Preface

As its title suggests, this book investigates reasoning about knowledge, in particular, reasoning about the knowledge of agents who reason about the world and each other's knowledge. This is the type of reasoning one often sees in puzzles or Sherlock Holmes mysteries, where we might have reasoning such as this:

> If Alice knew that Bob knew that Charlie was wearing a red shirt, then Alice would have known that Bob would have known that Charlie couldn't have been in the pantry at midnight. But Alice didn't know this ...

As we shall see, this type of reasoning is also important in a surprising number of other contexts. Researchers in a wide variety of disciplines, from philosophy to economics to cryptography, have all found that issues involving agents reasoning about other agents' knowledge are of great relevance to them. We attempt to provide here a framework for understanding and analyzing reasoning about knowledge that is intuitive, mathematically well founded, useful in practice, and widely applicable.

The book is almost completely self-contained. We do expect the reader to be familiar with propositional logic; a nodding acquaintance with distributed systems may be helpful to appreciate some of our examples, but it is not essential. Our hope is that the book will be accessible to readers from a number of different disciplines, including computer science, artificial intelligence, philosophy, and game theory. While proofs of important theorems are included, the non-mathematically-oriented reader should be able to skip them, while still following the main thrust of the book.

We have tried to make the book modular, so that, whenever possible, separate chapters can be read independently. At the end of Chapter 1 there is a brief overview of the book and a table of dependencies. Much of this material was taught a number of times by the second author in one-quarter courses at Stanford University and by the third author in one-semester courses at the Weizmann Institute of Science. Suggestions for subsets of material that can be covered can also be found at the end of Chapter 1.

Many of the details that are not covered in the main part of the text of each chapter are relegated to the exercises. As well, the exercises cover material somewhat tangential—but still of interest!—to the main thrust of the chapter. We recommend that the reader at least look over all the exercises in each chapter. Far better, of course, would be to do them all (or at least a reasonable subset). Problems that are somewhat more difficult are marked with $*$, and even more difficult problems are marked with $**$.

Each chapter ends with a section of notes. These notes provide references to the material covered in each chapter (as well as the theorems that are stated but not proved) and, occasionally, more details on some points not covered in the chapter. The references appearing in the notes are to the latest version of the material we could find. In many cases, earlier versions appeared in conference proceedings. The dates of the references that appear in the notes therefore do not provide a chronological account of the contributions to the field. While we attempt to provide reasonably extensive coverage of the literature in these notes, the field is too large for our coverage to be complete. We apologize for the inadvertent omission of relevant references.

The book concludes with a bibliography, an index, and a symbol index.

Many people helped us in many ways in the preparation of this book, and we are thankful to all of them. Daphne Koller deserves a very special note of thanks. She did a superb job of proofreading the almost-final draft of the book. Besides catching many typographical errors, she gave us numerous suggestions on improving the presentation in every chapter. We are very grateful to her. We would also like to thank Johan van Benthem, Mike Fischer, Adam Grove, Vassos Hadzilacos, Lane Hemaspaandra and the students of CS 487 at the University of Rochester, Wil Janssen, Hector Levesque, Murray Mazer, Ron van der Meyden, Jan Pachl, Karen Rudie, Ambuj Singh, Elias Thijsse, Mark Tuttle, and Lenore Zuck, for their useful comments and criticisms; Johan van Benthem, Brian Chellas, David Makinson, and Krister Segerberg for their help in tracking down the history of modal logic; and T. C. Chen and Brian Coan for pointing out the quotations at the beginning of Chapters 2 and 3, respectively. Finally, the second and third authors would like to thank the students of CS 356 (at Stanford in the years 1984–1989, 1991–1992, and 1994), CS 2422S (at Toronto in 1990) and the course on Knowledge Theory (at the Weizmann Institute of Science in the years 1987–1995), who kept finding typographical errors and suggesting improvements to the text (and wondering if the book would ever be completed), especially Gidi Avrahami, Ronen Brafman, Ed Brink, Alex Bronstein, Isis Caulder, Steve Cummings, John DiMarco, Kathleen Fisher, Steve Friedland, Tom Henzinger, David Karger, Steve Ketchpel, Orit Kislev, Christine Knight, Ronny Kohavi, Rick Kunin, Sherry Listgarten, Carlos Mendioroz,

Andres Modet, Shahid Mujtaba, Gal Nachum, Leo Novik, Raymond Pang, Barney Pell, Sonne Preminger, Derek Proudian, Omer Reingold, Tselly Regev, Gil Roth, Steve Souder, Limor Tirosh-Pundak-Mintz, Maurits van der Veen, Orli Waarts, Scott Walker, and Liz Wolf.

Finally, we wish to thank the institutions that supported this work for many years; the work of the first, second, and fourth authors was done at the IBM Almaden Research Center in San Jose, California, and the work of the third author was done at the Weizmann Institute of Science in Rehovot, Israel, and while on sabbatical at the Oxford University Computing Laboratory in Oxford, England. The work of the third author was supported in part by a Sir Charles Clore Post-Doctoral Fellowship, by an Alon Fellowship, and by a Helen and Milton A. Kimmelman Career Development Chair.

Chapter 1

Introduction and Overview

An investment in knowledge pays the best interest.

Benjamin Franklin, *Poor Richard's Almanac*, c. 1750

Epistemology, the study of knowledge, has a long and honorable tradition in philosophy, starting with the early Greek philosophers. Questions such as "What do we know?" "What can be known?" and "What does it mean to say that someone knows something?" have been much discussed in the philosophical literature. The idea of a formal logical analysis of reasoning about knowledge is somewhat more recent, but goes back at least to von Wright's work in the early 1950's. The first book-length treatment of *epistemic logic*—the logic of knowledge—is Hintikka's seminal work *Knowledge and Belief*, which appeared in 1962. The 1960's saw a flourishing of interest in this area in the philosophy community. The major interest was in trying to capture the inherent properties of knowledge. Axioms for knowledge were suggested, attacked, and defended.

More recently, researchers in such diverse fields as economics, linguistics, AI (artificial intelligence), and theoretical computer science have become interested in reasoning about knowledge. While, of course, some of the issues that concerned the philosophers have been of interest to these researchers as well, the focus of attention has shifted. For one thing, there are pragmatic concerns about the relationship between knowledge and action. What does a robot need to know in order to open a safe, and how does it know whether it knows enough to open it? At what point does an economic agent know enough to stop gathering information and make a decision? When should a database answer "I don't know" to a query? There are also concerns about the complexity of computing knowledge, a notion we can now quantify better

thanks to advances in theoretical computer science. Finally, and perhaps of most interest to us here, is the emphasis on considering situations involving the knowledge of a group of agents, rather than that of just a single agent.

When trying to understand and analyze the properties of knowledge, philosophers tended to consider only the single-agent case. But the heart of any analysis of a conversation, a bargaining session, or a protocol run by processes in a distributed system is the interaction between agents. The focus of this book is on understanding the process of reasoning about knowledge in a group and using this understanding to help us analyze complicated systems. Although the reader will not go far wrong if he or she thinks of a "group" as being a group of people, it is useful to allow a more general notion of "group," as we shall see in our applications. Our agents may be negotiators in a bargaining situation, communicating robots, or even components such as wires or message buffers in a complicated computer system. It may seem strange to think of wires as agents who know facts; however, as we shall see, it is useful to ascribe knowledge even to wires.

An agent in a group must take into account not only facts that are true about the world, but also the knowledge of other agents in the group. For example, in a bargaining situation, the seller of a car must consider what the potential buyer knows about the car's value. The buyer must also consider what the seller knows about what the buyer knows about the car's value, and so on. Such reasoning can get rather convoluted. Most people quickly lose the thread of such nested sentences as "Dean doesn't know whether Nixon knows that Dean knows that Nixon knows that McCord burgled O'Brien's office at Watergate." But this is precisely the type of reasoning that is needed when analyzing the knowledge of agents in a group.

A number of states of knowledge arise naturally in a multi-agent situation that do not arise in the one-agent case. We are often interested in situations in which *everyone* in the group knows a fact. For example, a society certainly wants all drivers to know that a red light means "stop" and a green light means "go." Suppose we assume that every driver in the society knows this fact and follows the rules. Will a driver then feel safe? The answer is no, unless she also knows that everyone else knows and is following the rules. For otherwise, a driver may consider it possible that, although she knows the rules, some other driver does not, and that driver may run a red light.

Even the state of knowledge in which everyone knows that everyone knows is not enough for a number of applications. In some cases we also need to consider the state in which simultaneously everyone knows a fact φ, everyone knows that everyone knows φ, everyone knows that everyone knows that everyone knows φ, and so on. In this case we say that the group has *common knowledge* of φ. This key notion was first studied by the philosopher David Lewis in the context of conventions. Lewis

pointed out that in order for something to be a convention, it must in fact be common knowledge among the members of a group. (For example, the convention that green means "go" and red means "stop" is presumably common knowledge among the drivers in our society.) John McCarthy, in the context of studying common-sense reasoning, characterized common knowledge as what "any fool" knows; "any fool" knows what is commonly known by all members of a society.

Common knowledge also arises in discourse understanding. Suppose Ann asks Bob "What did you think of the movie?" referring to a showing of *Monkey Business* they have just seen. Not only must Ann and Bob both know that "the movie" refers to *Monkey Business*, but Ann must know that Bob knows (so that she can be sure that Bob will give a reasonable answer to her question), Bob must know that Ann knows that Bob knows (so that Bob knows that Ann will respond appropriately to his answer), and so on. In fact, by a closer analysis of this situation, it can be shown that there must be common knowledge of what movie is meant for Bob to answer the question appropriately.

Finally, common knowledge also turns out to be a prerequisite for achieving agreement. This is precisely what makes it such a crucial notion in the analysis of interacting groups of agents.

At the other end of the spectrum from common knowledge is distributed knowledge. A group has distributed knowledge of a fact φ if the knowledge of φ is distributed among its members, so that by pooling their knowledge together the members of the group can deduce φ, even though it may be the case that no member of the group individually knows φ. For example, if Alice knows that Bob is in love with either Carol or Susan, and Charlie knows that Bob is not in love with Carol, then together Alice and Charlie have distributed knowledge of the fact that Bob is in love with Susan, although neither Alice nor Charlie individually has this knowledge. While common knowledge can be viewed as what "any fool" knows, distributed knowledge can be viewed as what a "wise man"—one who has complete knowledge of what each member of the group knows—would know.

Common knowledge and distributed knowledge are useful tools in helping us understand and analyze complicated situations involving groups of agents. The puzzle described in the next section gives us one example.

1.1 The Muddy Children Puzzle

Reasoning about the knowledge of a group can involve subtle distinctions between a number of states of knowledge. A good example of the subtleties that can arise is

given by the "muddy children" puzzle, which is a variant of the well known "wise men" or "cheating wives" puzzles.

> Imagine n children playing together. The mother of these children has told them that if they get dirty there will be severe consequences. So, of course, each child wants to keep clean, but each would love to see the others get dirty. Now it happens during their play that some of the children, say k of them, get mud on their foreheads. Each can see the mud on others but not on his own forehead. So, of course, no one says a thing. Along comes the father, who says, "At least one of you has mud on your forehead," thus expressing a fact known to each of them before he spoke (if $k > 1$). The father then asks the following question, over and over: "Does any of you know whether you have mud on your own forehead?" Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen?
>
> There is a "proof" that the first $k - 1$ times he asks the question, they will all say "No," but then the k^{th} time the children with muddy foreheads will all answer "Yes."
>
> The "proof" is by induction on k. For $k = 1$ the result is obvious: the one child with a muddy forehead sees that no one else is muddy. Since he knows that there is at least one child with a muddy forehead, he concludes that he must be the one. Now suppose $k = 2$. So there are just two muddy children, a and b. Each answers "No" the first time, because of the mud on the other. But, when b says "No," a realizes that he must be muddy, for otherwise b would have known the mud was on his forehead and answered "Yes" the first time. Thus a answers "Yes" the second time. But b goes through the same reasoning. Now suppose $k = 3$; so there are three muddy children, a, b, c. Child a argues as follows. Assume I do not have mud on my forehead. Then, by the $k = 2$ case, both b and c will answer "Yes" the second time. When they do not, he realizes that the assumption was false, that he is muddy, and so will answer "Yes" on the third question. Similarly for b and c.
>
> The argument in the general case proceeds along identical lines.

Let us denote the fact "at least one child has a muddy forehead" by p. Notice that if $k > 1$, i.e., more than one child has a muddy forehead, then every child can see at least one muddy forehead, and the children initially all know p. Thus, it would seem that the father does not provide the children with any new information, and so

he should not need to tell them that p holds when $k > 1$. But this is false! In fact, as we now show, if the father does not announce p, the muddy children are never able to conclude that their foreheads are muddy.

Here is a sketch of the proof: We prove by induction on q that, no matter what the situation is, i.e., no matter how many children have a muddy forehead, all the children answer "No" to the father's first q questions. Clearly, no matter which children have mud on their foreheads, all the children answer "No" to the father's first question, since a child cannot tell apart a situation where he has mud on his forehead from one that is identical in all respects except that he does not have a muddy forehead. The inductive step is similar: By the inductive hypothesis, the children answer "No" to the father's first q questions. Thus, when the father asks his question for the $(q+1)^{\text{st}}$ time, child i still cannot tell apart a situation where he has mud on his forehead from one that is identical in all respects except that he does not have a muddy forehead, since by the induction hypothesis, the children will answer "No" to the father's first q questions whether or not child i has a muddy forehead. Thus, again, he does not know whether his own forehead is muddy.

So, by announcing something that the children all know, the father somehow manages to give the children useful information! How can this be? Exactly what *is* the role of the father's statement? Of course, the father's statement did enable us to do the base case of the induction in the proof, but this does not seem to be a terribly satisfactory answer. It certainly does not explain what information the children gained as a result of the father's statement.

We can answer these questions by using the notion of common knowledge described in the previous section. Let us consider the case of two muddy children in more detail. It is certainly true that before the father speaks, everyone knows p. But it is not the case that everyone knows that everyone knows p. If Alice and Bob are the only children with muddy foreheads, then before the father speaks, Alice considers it possible that she does not have mud on her forehead, in which case Bob does not see anyone with a muddy forehead and so does not know p. After the father speaks, Alice does know that Bob knows p. After Bob answers "No" to the father's first question, Alice uses her knowledge of the fact that Bob knows p to deduce that her own forehead is muddy. (Note that if Bob did not know p, then Bob would have said "No" the first time even if Alice's forehead were clean.)

We have just seen that if there are only two muddy children, then it is not the case that everyone knows that everyone knows p before the father speaks. However, if there are three muddy children, then it *is* the case that everyone knows that everyone knows p before the father speaks. If Alice, Bob, and Charlie have muddy foreheads, then Alice knows that Bob can see Charlie's muddy forehead, Bob knows that Charlie

can see Alice's muddy forehead, etc. It is not the case, however, that everyone knows that everyone knows that everyone knows p before the father speaks. In general, if we let $E^k p$ represent the fact that everyone knows that everyone knows ... (k times) p, and let Cp represent the fact that p is common knowledge, then we leave it to the reader to check that if exactly k children have muddy foreheads, then $E^{k-1} p$ holds before the father speaks, but $E^k p$ does not. It turns out that when there are k muddy children, $E^k p$ suffices to ensure that the children with muddy foreheads will be able to figure it out, while $E^{k-1} p$ does not. The father's statement actually converts the children's state of knowledge from $E^{k-1} p$ to Cp. With this extra knowledge, they can deduce whether their foreheads are muddy.

The careful reader will have noticed that we made a number of implicit assumptions in the preceding discussion over and above the assumption made in the story that "the children are perceptive, intelligent, and truthful." Suppose again that Alice and Bob are the only children with muddy foreheads. It is crucial that both Alice and Bob *know* that the children are intelligent, perceptive, and truthful. For example, if Alice does not know that Bob is telling the truth when he answers "No" to the father's first question, then she cannot answer "Yes" to the second question (even if Bob is in fact telling the truth). Similarly, Bob must know that Alice is telling the truth. Besides its being known that each child is intelligent, perceptive, and truthful, we must also assume that each child knows that the others can see, that they all hear the father, that the father is truthful, and that the children can do all the deductions necessary to answer the father's questions.

Actually, even stronger assumptions need to be made. If there are k children with muddy foreheads, it must be the case that everyone knows that everyone knows ... ($k - 1$ times) that the children all have the appropriate attributes (they are perceptive, intelligent, all hear the father, etc.). For example, if there are three muddy children and Alice considers it possible that Bob considers it possible that Charlie might not have heard the father's statement, then she cannot say "Yes" to the father's third question (even if Charlie in fact did hear the father's statement and Bob knows this). In fact, it seems reasonable to assume that all these attributes are common knowledge, and, indeed, this assumption seems to be made by most people on hearing the story.

To summarize, it seems that the role of the father's statement was to give the children common knowledge of p (the fact that at least one child has a muddy forehead), but the reasoning done by the children assumes that a great deal of common knowledge already existed in the group. How does this common knowledge arise? Even if we ignore the problem of how facts like "all the children can see" and "all

the children are truthful" become common knowledge, there is still the issue of how the father's statement makes p common knowledge.

Note that it is not quite correct to say that p becomes common knowledge because all the children hear the father. Suppose that the father had taken each child aside individually (without the others noticing) and said "At least one of you has mud on your forehead." The children would probably have thought it a bit strange for him to be telling them a fact that they already knew. It is easy to see that p would not become common knowledge in this setting.

Given this example, one might think that the common knowledge arose because all the children *knew* that they all heard the father. Even this is not enough. To see this, suppose the children do not trust each other, and each child has secretly placed a miniature microphone on all the other children. (Imagine that the children spent the previous summer at a CIA training camp.) Again the father takes each child aside individually and says "At least one of you has a muddy forehead." In this case, thanks to the hidden microphones, all the children know that each child has heard the father, but they still do not have common knowledge.

A little more reflection might convince the reader that the common knowledge arose here because of the *public* nature of the father's announcement. Roughly speaking, the father's public announcement of p puts the children in a special situation, one with the property that all the children know both that p is true and that they are in this situation. We shall show that under such circumstances p is common knowledge. Note that the common knowledge does not arise because the children somehow deduce each of the facts $E^k p$ one by one. (If this were the case, then arguably it would take an infinite amount of time to attain common knowledge.) Rather, the common knowledge arises all at once, as a result of the children being in such a special situation. We return to this point in later chapters.

1.2 An Overview of the Book

The preceding discussion should convince the reader that the subtleties of reasoning about knowledge demand a careful formal analysis. In Chapter 2, we introduce a simple, yet quite powerful, formal semantic model for knowledge, and a language for reasoning about knowledge. The basic idea underlying the model is that of *possible worlds*. The intuition is that if an agent does not have complete knowledge about the world, she will consider a number of worlds possible. These are her candidates for the way the world actually is. The agent is said to *know* a fact φ if φ holds at all the worlds that the agent considers to be possible. Using this semantic model allows

us to clarify many of the subtleties of the muddy children puzzle in quite an elegant way. The analysis shows how the children's state of knowledge changes with each response to the father's questions, and why, if there are k muddy children altogether, it is only after the k^{th} question that the children with muddy foreheads can deduce this fact.

We should emphasize here that we do not feel that the semantic model we present in the next chapter is the unique "right" model of knowledge. We spend some time discussing the properties of knowledge in this model. A number of philosophers have presented cogent arguments showing that some of these properties are "wrong." Our concerns in this book are more pragmatic than those of the philosophers. We do not believe that there is a "right" model of knowledge. Different notions of knowledge are appropriate for different applications. The model we present in the next chapter is appropriate for analyzing the muddy children puzzle and for many other applications, even if it is not appropriate for every application. One of our goals in this book is to show how the properties of "knowledge" vary with the application.

In Chapter 3, we give a complete characterization of the properties of knowledge in the possible-worlds model. We describe two approaches to this characterization. The first approach is *proof-theoretic*: we show that all the properties of knowledge can be formally proved from the properties discussed in Chapter 2. The second approach is *algorithmic*: we study algorithms that can determine whether a given property holds under our definition of knowledge, and consider the computational complexity of doing this.

One of the major applications we have in mind is using knowledge to analyze *multi-agent systems*, be they systems of interacting agents or systems of computers in a network. In Chapter 4 we show how we can use our semantic model for knowledge to *ascribe* knowledge to agents in a multi-agent system. The reason that we use the word "ascribe" here is that the notion of knowledge we use in the context of multi-agent systems can be viewed as an *external* notion of knowledge. There is no notion of the agent computing his knowledge, and no requirement that the agent be able to answer questions based on his knowledge. While this may seem to be an unusual way of defining knowledge, we shall argue that it does capture one common usage of the word "know." Moreover, we give examples that show its utility in analyzing multi-agent systems.

In Chapter 5 we extend the model of Chapter 4 to consider *actions*, *protocols*, and *programs*. This allows us to analyze more carefully how changes come about in multi-agent systems. We also define the notion of a *specification* and consider what it means for a protocol or program to satisfy a specification.

In Chapter 6 we show how useful a knowledge-based analysis of systems can be. Our focus in this chapter is common knowledge, and we show how fundamental it is in various contexts. In particular, we show that it is a prerequisite for agreement and simultaneous coordinated action.

In Chapter 7 we extend our notions of programs to consider *knowledge-based* programs, which allow explicit tests for knowledge. Knowledge-based programs can be viewed as giving us a high-level language in which to program or specify a system. We give a number of examples showing the usefulness of thinking and programming at the knowledge level.

In Chapter 8 we consider the properties of knowledge and time, focusing on how knowledge evolves over time in multi-agent systems. We show that small changes in the assumptions we make about the interactions between knowledge and time in a system can have quite subtle and powerful effects on the properties of knowledge.

As we show in Chapter 2, one property that seems to be an inherent part of the possible-worlds model of knowledge is that agents are *logically omniscient.* Roughly speaking, this means they know all tautologies and all logical consequences of their knowledge. In the case of the muddy children puzzle we explicitly make the assumption that each child can do all the reasoning required to solve the puzzle. While this property may be reasonable for some applications, it certainly is not reasonable in general. After all, we cannot really hope to build logically omniscient robots. In Chapter 9 we describe several approaches for constructing abstract models that do not have the logical omniscience property.

As we have already discussed, our notion of knowledge in multi-agent systems is best understood as an external one, ascribed by, say, the system designer to the agents. We do not assume that the agents compute their knowledge in any way, nor do we assume that they can necessarily answer questions based on their knowledge. In a number of applications that we are interested in, agents need to *act* on their knowledge. In such applications, external knowledge is insufficient; an agent that has to act on her knowledge has to be able to compute this knowledge. The topic of knowledge and computation is the subject of Chapter 10.

In Chapter 11, we return to the topic of common knowledge. We suggested in the previous section that common knowledge arose in the muddy children puzzle because of the public nature of the father's announcement. In many practical settings such a public announcement, whose contents are understood simultaneously by many agents, is impossible to achieve. We show that, in a precise sense, common knowledge cannot be attained in these settings. This puts us in a somewhat paradoxical situation, in that we claim both that common knowledge is a prerequisite for agreement and coordinated action and that it cannot be attained. We examine this

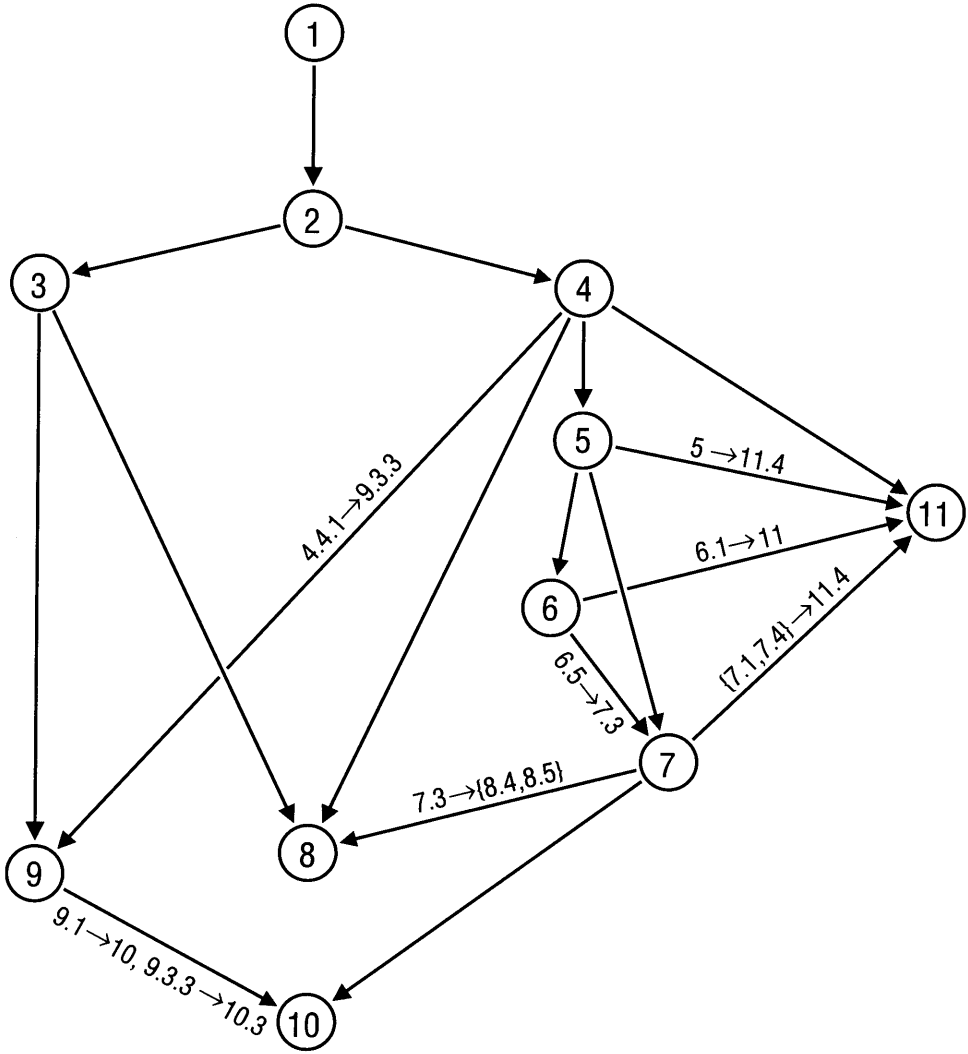

Figure 1.1 Dependency diagram

paradox in Chapter 11 and suggest a number of weaker notions that can be attained in practice and that are often sufficient for acting in the real world.

Although a considerable amount of the material in this book is based on previously published work, a number of elements are new. These include much of the material in Chapters 5, 7, 10, and some of Chapter 11. Specifically, the notions of contexts and programs in Chapter 5, and of knowledge-based programs and their implementation in Chapter 7, are new. Moreover, they play a significant role in the way we model and analyze knowledge and action in multi-agent systems.

We have tried as much as possible to write the book in a modular way, so that material in the later chapters can be read without having to read all the preceding chapters. Figure 1.1 describes the dependencies between chapters. An arrow from one chapter to another indicates that it is necessary to read (at least part of) the first chapter in order to understand (at least part of) the second. We have labeled the arrow if it is not necessary to read all of the first chapter to understand all of the second. For example, the label $9.1 \rightarrow 10, 9.3.3 \rightarrow 10.3$ on the arrow from Chapter 9 to Chapter 10 indicates that the only sections in Chapter 9 on which Chapter 10 depends are 9.1 and 9.3.3 and, moreover, the only section in Chapter 10 that depends on Section 9.3.3 is Section 10.3. Similarly, the label $5 \rightarrow 11.4$ on the arrow from Chapter 5 to Chapter 11 indicates that Section 11.4 is the only section in Chapter 11 that depends on Chapter 5, but it depends on the whole chapter.

Certain material can be skipped without losing a broad overview of the area. In particular, this is the case for Sections 3.3, 3.4, 4.5, 6.7, and 7.7. The second author covered a substantial portion of the remaining material (moving at quite a rapid pace) in a one-quarter course at Stanford University. A course designed to focus on the application of our approach to distributed systems could cover Chapters 1, 2, 4, 5, 6, 7, 10, and 11. Each chapter ends with exercises and bibliographic notes; these could be useful in a course based on this book. As we mentioned in the preface, we strongly recommend that the reader at least look over the exercises.

Exercises

1.1 The *aces and eights* game is a simple game that involves some sophisticated reasoning about knowledge. It is played with a deck consisting of just four aces and four eights. There are three players. Six cards are dealt out, two to each player. The remaining two cards are left face down. Without looking at the cards, each of the players raises them up to his or her forehead, so that the other two players can see

them but he or she cannot. Then all of the players take turns trying to determine which cards they're holding (they do not have to name the suits). If a player does not know which cards he or she is holding, the player must say so. Suppose Alice, Bob, and you are playing the game. Of course, it is common knowledge that none of you would ever lie, and that you are all perfect reasoners.

(a) In the first game, Alice, who goes first, holds two aces, and Bob, who goes second, holds two eights. Both Alice and Bob say that they cannot determine what cards they are holding. What cards are you holding? (Hint: consider what would have happened if you held two aces or two eights.)

(b) In the second game, you go first. Alice, who goes second, holds two eights. Bob, who goes third, holds an ace and an eight. No one is able to determine what he or she holds at his or her first turn. What do you hold? (Hint: by using part (a), consider what would have happened if you held two aces.)

(c) In the third game, you go second. Alice, who goes first, holds an ace and an eight. Bob, who goes third, also holds an ace and an eight. No one is able to determine what he or she holds at his or her first turn; Alice cannot determine her cards at her second turn either. What do you hold?

* **1.2** Show that in the aces and eights game of Exercise 1.1, someone will always be able to determine what cards he or she holds. Then show that there exists a situation where only one of the players will be able to determine what cards he or she holds, and the other two will never be able to determine what cards they hold, no matter how many rounds are played.

1.3 The *wise men puzzle* is a well-known variant of the muddy children puzzle. The standard version of the story goes as follows: There are three wise men. It is common knowledge that there are three red hats and two white hats. The king puts a hat on the head of each of the three wise men, and asks them (sequentially) if they know the color of the hat on their head. The first wise man says that he does not know; the second wise man says that he does not know; then the third wise man says that he knows.

(a) What color is the third wise man's hat?

(b) We have implicitly assumed in the story that the wise men can all see. Suppose we assume instead that the third wise man is blind and that it is common knowledge that the first two wise men can see. Can the third wise man still figure out the color of his hat?

Notes

The idea of a formal logical analysis of reasoning about knowledge seems to have first been raised by von Wright [1951]. As we mentioned in the text, Hintikka [1962] gave the first book-length treatment of epistemic logic. Lenzen [1978] gives an overview of the work in epistemic logic done in the 1960s and 1970s. He brings out the arguments for and against various axioms of knowledge. The most famous of these arguments is due to Gettier [1963], who argued against the classical interpretation of knowledge as true, justified belief; his work inspired many others. Gettier's arguments and some of the subsequent papers are discussed in detail by Lenzen [1978]. For recent reviews of the subject, see the papers by Halpern [1986, 1987, 1993a], by Meyer, van der Hoek, and Vreeswijk [1991a, 1991b], by Moses [1992], and by Parikh [1990].

As we mentioned, the original work on common knowledge was done by Lewis [1969] in the context of studying conventions. Although McCarthy's notion of what "any fool" knows goes back to roughly 1970, it first appears in a published paper in [McCarthy, Sato, Hayashi, and Igarishi 1979]. The notion of knowledge and common knowledge has also been of great interest to economists and game theorists, ever since the seminal paper by Aumann [1976]. Knowledge and common knowledge were first applied to multi-agent systems by Halpern and Moses [1990] and by Lehmann [1984]. The need for common knowledge in understanding a statement such as "What did you think of the movie?" is discussed by Clark and Marshall [1981]; a dissenting view is offered by Perrault and Cohen [1981]. Clark and Marshall also present an example of nested knowledge based on the Watergate scandal, mentioning Dean and Nixon. The notion of distributed knowledge was introduced by Halpern and Moses [1990]. They initially called it *implicit knowledge*, and the term "distributed knowledge" was suggested by Jan Pachl.

The muddy children puzzle is a variant of the "unfaithful wives" puzzle discussed by Gamow and Stern [1958]. Gardner [1984] also presents a variant of the puzzle, and a number of variants of the puzzle are discussed by Moses, Dolev, and Halpern [1986]. The version given here is taken almost verbatim from [Barwise 1981]. The aces and eights puzzle in Exercise 1.1 is taken from [Carver 1989]. Another related puzzle is the so-called "Conway paradox", which was first discussed by Conway, Paterson, and Moscow [1977], and later by Gardner [1977]. It was analyzed in an epistemic framework by van Emde Boas, Groenendijk, and Stokhof [1980]. An extension of this puzzle was considered by Parikh [1992].

See the bibliographic notes in later chapters for more references on the specific subjects discussed in these chapters.

Chapter 2

A Model for Knowledge

Chuangtse and Hueitse had strolled onto the bridge over the Hao, when the former observed, "See how the small fish are darting about! That is the happiness of the fish." "You are not a fish yourself," said Hueitse. "How can you know the happiness of the fish?" "And you not being I," retorted Chuangtse, "how can you know that I do not know?"

Chuangtse, c. 300 B.C.

2.1 The Possible-Worlds Model

As we said in Chapter 1, our framework for modeling knowledge is based on *possible worlds*. The intuitive idea behind the possible-worlds model is that besides the true state of affairs, there are a number of other possible states of affairs, or "worlds." Given his current information, an agent may not be able to tell which of a number of possible worlds describes the actual state of affairs. An agent is then said to *know* a fact φ if φ is true at all the worlds he considers possible (given his current information). For example, agent 1 may be walking on the streets in San Francisco on a sunny day but may have no information at all about the weather in London. Thus, in all the worlds that the agent considers possible, it is sunny in San Francisco. (We are implicitly assuming here that the agent does not consider it possible that he is hallucinating and in fact it is raining heavily in San Francisco.) On the other hand, since the agent has no information about the weather in London, there are worlds he considers possible in which it is sunny in London, and others in which it is raining in London. Thus, this agent knows that it is sunny in San Francisco,

but he does not know whether it is sunny in London. Intuitively, the fewer worlds an agent considers possible, the less his uncertainty, and the more he knows. If the agent acquires additional information—such as hearing from a reliable source that it is currently sunny in London—then he would no longer consider possible any of the worlds in which it is raining in London.

In a situation such as a poker game, these possible worlds have a concrete interpretation: they are simply all the possible ways the cards could have been distributed among the players. Initially, a player may consider possible all deals consistent with the cards in her hand. Players may acquire additional information in the course of the play of the game that allows them to eliminate some of the worlds they consider possible. Even if Alice does not know originally that Bob holds the ace of spades, at some point Alice might come to know it, if the additional information she obtains allows her to eliminate all the worlds (distributions of cards among players) where Bob does not hold the ace of spades.

Another example is provided by the muddy children puzzle we discussed in the previous chapter. Suppose Alice sees that Bob and Charlie have muddy foreheads and that all the other children do not have muddy foreheads. This allows her to eliminate all but two worlds: one in which she, Bob, and Charlie have muddy foreheads, and no other child does, and one in which Bob and Charlie are the only children with muddy foreheads. In all (i.e., both) of the worlds that Alice considers possible, Bob and Charlie have muddy foreheads and all the children except Bob, Charlie, and her have clean foreheads. Alice's only uncertainty is regarding her own forehead; this uncertainty is reflected in the set of worlds she considers possible. As we shall see in Section 2.3, on hearing the children's replies to the father's first two questions, Alice will be able to eliminate one of these two possible worlds and will know whether or not her own forehead is muddy.

To make these ideas precise, we first need a language that allows us to express notions of knowledge in a straightforward way. As we have already seen, English is not a particularly good language in which to carry out complicated reasoning about knowledge. Instead we use the language of *modal logic*.

Suppose we have a group consisting of n agents, creatively named $1, \ldots, n$. For simplicity, we assume these agents wish to reason about a world that can be described in terms of a nonempty set Φ of *primitive propositions*, typically labeled $p, p', q, q', \ldots$ These primitive propositions stand for basic facts about the world such as "it is sunny in San Francisco" or "Alice has mud on her forehead." To express a statement like "Bob *knows* that it is sunny in San Francisco," we augment the language by *modal* operators $K_1, \ldots, K_n$ (one for each agent). A statement like $K_1\varphi$ is then read "agent 1 knows φ."

Technically, a *language* is just a set of formulas. We can now describe the set of formulas of interest to us. We start with the primitive propositions in Φ and form more complicated formulas by closing off under negation, conjunction, and the modal operators $K_1, \ldots, K_n$. Thus, if φ and ψ are formulas, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, and $K_i\varphi$, for $i = 1, \ldots, n$. For the sake of readability, we omit the parentheses in formulas such as $(\varphi \wedge \psi)$ whenever it does not lead to confusion. We also use standard abbreviations from propositional logic, such as $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \Rightarrow \psi$ for $\neg\varphi \vee \psi$, and $\varphi \Leftrightarrow \psi$ for $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$. We take *true* to be an abbreviation for some fixed propositional tautology such as $p \vee \neg p$, and take *false* to be an abbreviation for $\neg$*true*.

We can express quite complicated statements in a straightforward way using this language. For example, the formula

$$K_1 K_2 p \wedge \neg K_2 K_1 K_2 p$$

says that agent 1 knows that agent 2 knows p, but agent 2 does not know that agent 1 knows that agent 2 knows p.

We view possibility as the dual of knowledge. Thus, agent 1 considers φ possible exactly if he does not know $\neg\varphi$. This situation can be described by the formula $\neg K_1 \neg\varphi$. A statement like "Dean doesn't know whether φ" says that Dean considers both φ and $\neg\varphi$ possible. Let's reconsider the sentence from the previous chapter: "Dean doesn't know whether Nixon knows that Dean knows that Nixon knows that McCord burgled O'Brien's office at Watergate." If we take Dean to be agent 1, Nixon to be agent 2, and p to be the statement "McCord burgled O'Brien's office at Watergate," then this sentence can be captured as

$$\neg K_1 \neg(K_2 K_1 K_2 p) \wedge \neg K_1 \neg(\neg K_2 K_1 K_2 p).$$

Now that we have described the *syntax* of our language (that is, the set of well-formed formulas), we need *semantics*, that is, a formal model that we can use to determine whether a given formula is true or false. One approach to defining semantics is, as we suggested, in terms of possible worlds, which we formalize in terms of *(Kripke) structures*. (In later chapters we consider other approaches to giving semantics to formulas.) A Kripke structure M for n agents over Φ is a tuple $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where S is a set of *states* or *possible worlds*, π is an *interpretation* that associates with each state in S a truth assignment to the primitive propositions of Φ (i.e., $\pi(s) : \Phi \rightarrow \{\textbf{true}, \textbf{false}\}$ for each state $s \in S$), and $\mathcal{K}_i$ is a binary relation on S, that is, a set of pairs of elements of S. We tend to use "state" and "world" interchangeably throughout the book.

The truth assignment $\pi(s)$ tells us whether p is true or false in state s. Thus, if p denotes the fact "it is raining in San Francisco," then $\pi(s)(p) = \mathbf{true}$ captures the situation in which it is raining in San Francisco in state s of structure M. The binary relation $\mathcal{K}_i$ is intended to capture the possibility relation according to agent i: $(s, t) \in \mathcal{K}_i$ if agent i considers world t possible, given his information in world s. We think of $\mathcal{K}_i$ as a *possibility* relation, since it defines what worlds agent i considers possible in any given world. Throughout most of the book (in particular, in this chapter), we further require that $\mathcal{K}_i$ be an *equivalence relation* on S. An equivalence relation $\mathcal{K}$ on S is a binary relation that is (a) *reflexive*, which means that for all $s \in S$, we have $(s, s) \in \mathcal{K}$, (b) *symmetric*, which means that for all $s, t \in S$, we have $(s, t) \in \mathcal{K}$ if and only if $(t, s) \in \mathcal{K}$, and (c) *transitive*, which means that for all $s, t, u \in S$, we have that if $(s, t) \in \mathcal{K}$ and $(t, u) \in \mathcal{K}$, then $(s, u) \in \mathcal{K}$. We take $\mathcal{K}_i$ to be an equivalence relation since we want to capture the intuition that agent i considers t possible in world s if in both s and t agent i has the same information about the world, that is, the two worlds are indistinguishable to the agent. Making $\mathcal{K}_i$ an equivalence relation seems natural, and it turns out to be the appropriate choice for many applications. For example, as we shall see in the next section, it is appropriate in analyzing the muddy children puzzle, while in Chapters 4 and 6 we show that it is appropriate for many multi-agent systems applications. We could equally well, however, consider possibility relations with other properties (for example, reflexive and transitive, but not symmetric), as we in fact do in Chapter 3.

We now define what it means for a formula to be true at a given world in a structure. Note that truth depends on the world as well as the structure. It is quite possible that a formula is true in one world and false in another. For example, in one world agent 1 may know it is sunny in San Francisco, while in another he may not. To capture this, we define the notion $(M, s) \models \varphi$, which can be read as "φ *is true at* (M, s)" or "φ *holds at* (M, s)" or "(M, s) *satisfies* φ." We define the $\models$ relation by induction on the structure of φ. That is, we start with the simplest formulas—primitive propositions—and work our way up to more complicated formulas φ, assuming that $\models$ has been defined for all the subformulas of φ.

The π component of the structure gives us the information we need to deal with the base case, where φ is a primitive proposition:

$(M, s) \models p$ (for a primitive proposition $p \in \Phi$) iff $\pi(s)(p) = \mathbf{true}$.

For conjunctions and negations, we follow the standard treatment from propositional logic; a conjunction $\psi \wedge \psi'$ is true exactly if both of the conjuncts ψ and ψ' are true, while a negated formula $\neg\psi$ is true exactly if ψ is not true:

$(M, s) \models \psi \wedge \psi'$ iff $(M, s) \models \psi$ and $(M, s) \models \psi'$

$(M, s) \models \neg\psi$ iff $(M, s) \not\models \psi$.

Note that the clause for negation guarantees that the logic is two-valued. For every formula ψ, we have either $(M, s) \models \psi$ or $(M, s) \models \neg\psi$, but not both.

Finally, we have to deal with formulas of the form $K_i\psi$. Here we try to capture the intuition that agent i knows ψ in world s of structure M exactly if ψ is true at all worlds that i considers possible in s. Formally, we have

$(M, s) \models K_i\psi$ iff $(M, t) \models \psi$ for all t such that $(s, t) \in \mathcal{K}_i$.

These definitions are perhaps best illustrated by a simple example. One of the advantages of a Kripke structure is that it can be viewed as a labeled graph, that is, a set of labeled nodes connected by directed, labeled edges. The nodes are the states of S; the label of state $s \in S$ describes which primitive propositions are true and false at s. We label edges by sets of agents; the label on the edge from s to t includes i if $(s, t) \in \mathcal{K}_i$. For example, suppose $\Phi = \{p\}$ and $n = 2$, so that our language has one primitive proposition p and there are two agents. Further suppose that $M = (S, \pi, \mathcal{K}_1, \mathcal{K}_2)$, where $S = \{s, t, u\}$, p is true at states s and u but false at t (so that $\pi(s)(p) = \pi(u)(p) =$ **true** and $\pi(t)(p) =$ **false**), agent 1 cannot distinguish s from t (so that $\mathcal{K}_1 = \{(s, s), (s, t), (t, s), (t, t), (u, u)\}$), and agent 2 cannot distinguish s from u (so that $\mathcal{K}_2 = \{(s, s), (s, u), (t, t), (u, s), (u, u)\}$). This situation can be captured by the graph in Figure 2.1. Note how the graph captures our assumptions about the $\mathcal{K}_i$ relations. In particular, we have a self-loop at each edge labeled by both 1 and 2 because the relations $\mathcal{K}_1$ and $\mathcal{K}_2$ are reflexive, and the edges have an arrow in each direction because $\mathcal{K}_1$ and $\mathcal{K}_2$ are symmetric.

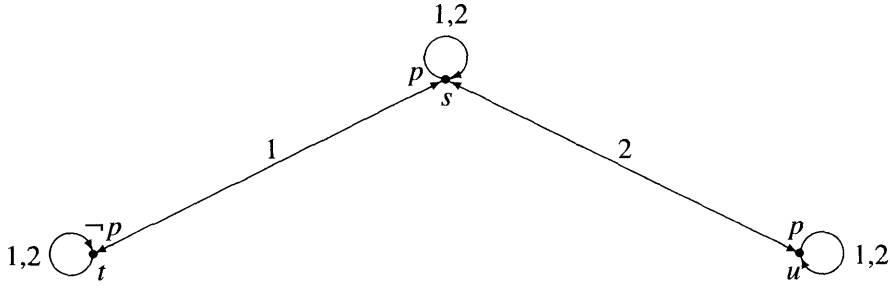

Figure 2.1 A simple Kripke structure

If we view p as standing for "it is sunny in San Francisco," then in state s it is sunny in San Francisco but agent 1 does not know it, since in state s he considers both s and t possible. (We remark that we used the phrase "agent 1 cannot distinguish s from t." Of course, agent 1 realizes perfectly well that s and t are different worlds. After all, it is raining in San Francisco in s, but not in t. What we really intend here is perhaps more accurately described by something like "agent 1's information is insufficient to enable him to distinguish whether the actual world is s or t." We continue to use the word "indistinguishable" in the somewhat looser sense throughout the book.) On the other hand, agent 2 does know in state s that it is sunny, since in both worlds that agent 2 considers possible at s (namely, s and u), the formula p is true. In state t, agent 2 also knows the true situation, namely, that it is not sunny. It follows that in state s agent 1 knows that agent 2 knows whether or not it is sunny in San Francisco: in both worlds agent 1 considers possible in state s, namely, s and t, agent 2 knows what the weather in San Francisco is. Thus, although agent 1 does not know the true situation at s, he does know that agent 2 knows the true situation. By way of contrast, although in state s agent 2 knows that it is sunny in San Francisco, she does not know that agent 1 does not know this fact. (In one world that agent 2 considers possible, namely u, agent 1 does know that it is sunny, while in another world agent 2 considers possible, namely s, agent 1 does not know this fact.) All of this relatively complicated English discussion can be summarized in one mathematical statement:

$$(M, s) \models p \wedge \neg K_1 p \wedge K_2 p \wedge K_1(K_2 p \vee K_2 \neg p) \wedge \neg K_2 \neg K_1 p.$$

Note that in both s and u, the primitive proposition p (the only primitive proposition in our language) gets the same truth value. One might think, therefore, that s and u are the same, and that perhaps one of them can be eliminated. This is not true! A state is not completely characterized by the truth values that the primitive propositions get there. The possibility relation is also crucial. For example, in world s, agent 1 considers t possible, while in u he does not. As a consequence, agent 1 does not know p in s, while in u he does.

We now consider a slightly more complicated example, which might provide a little more motivation for making the $\mathcal{K}_i$'s equivalence relations. Suppose we have a deck consisting of three cards labeled A, B, and C. Agents 1 and 2 each get one of these cards; the third card is left face down. A possible world is characterized by describing the cards held by each agent. For example, in the world (A, B), agent 1 holds card A and agent 2 holds card B (while card C is face down). There are clearly six possible worlds: (A, B), (A, C), (B, A), (B, C), (C, A), and (C, B). Moreover, it is clear that in a world such as (A, B), agent 1 thinks two worlds are possible:

(A, B) itself and (A, C). Agent 1 knows that he has card A, but he considers it possible that agent 2 could hold either card B or card C. Similarly, in world (A, B), agent 2 also considers two worlds: (A, B) and (C, B). In general, in a world (x, y), agent 1 considers (x, y) and (x, z) possible, and agent 2 considers (x, y) and (z, y) possible, where z is different from both x and y.

From this description, we can easily construct the $\mathcal{K}_1$ and $\mathcal{K}_2$ relations. It is easy to check that they are indeed equivalence relations, as required by the definitions. This is because an agent's possibility relation is determined by the information she has, namely, the card she is holding. This is an important general phenomenon: in any situation where an agent's possibility relation is determined by her information (and, as we shall see, there are many such situations), the possibility relations are equivalence relations.

The structure in this example with the three cards is described in Figure 2.2, where, since the relations are equivalence relations, we omit the self-loops and the arrows on edges for simplicity. (As we have observed, if there is an edge from state s to state t, there is bound to be an edge from t to s as well by symmetry.)

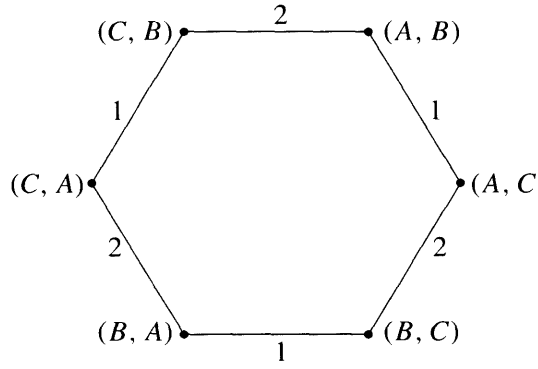

Figure 2.2 The Kripke structure describing a simple card game

This example points out the need for having worlds that an agent does not consider possible included in the structure. For example, in the world (A, B), agent 1 knows that the world (B, C) cannot be the case. (After all, agent 1 knows perfectly well that his own card is A.) Nevertheless, because agent 1 considers it possible that agent 2 considers it possible that (B, C) is the case, we must include (B, C) in the structure. This is captured in the structure by the fact that there is no edge from (A, B)

to (B, C) labeled 1, but there is an edge labeled 1 to (A, C), from which there is an edge labeled 2 to (B, C).

We still have not discussed the language to be used in this example. Since we are interested in reasoning about the cards held by agents 1 and 2, it seems reasonable to have primitive propositions of the form $1A$, $2A$, $2B$, and so on, which are to be interpreted as "agent 1 holds card A," "agent 2 holds card A," "agent 2 holds card B," and so on. Given this interpretation, we define π in the obvious way, and let M_c be the Kripke structure describing this card game. Then, for example, we have $(M_c, (A, B)) \models 1A \wedge 2B$. We leave it to the reader to check that we also have $(M_c, (A, B)) \models K_1(2B \vee 2C)$, which expresses the fact that if agent 1 holds A, then he knows that agent 2 holds either B or C. Similarly, we have $(M_c, (A, B)) \models K_1 \neg K_2(1A)$: agent 1 knows that agent 2 does not know that agent 1 holds A.

This example shows that our semantics does capture some of the intuitions we naturally associate with the word "knowledge." Nevertheless, this is far from a complete justification for our definitions, in particular, for our reading of the formula $K_i\varphi$ as "agent i knows φ." The question arises as to what would constitute a reasonable justification. We ultimately offer two justifications, which we hope the reader will find somewhat satisfactory. The first is by further examples, showing that our definitions correspond to reasonable usages of the word "know." One such example is given in Section 2.3, where we analyze the muddy children puzzle and show that the formula $K_i\varphi$ does capture our intuition regarding what child i knows. The second justification can be found in Section 2.4, where we consider some of the properties of this notion of knowledge and show that they are consistent with the properties that the knowledge of a perfect reasoner with perfect introspection might have. Of course, this does not imply that other reasonable notions of knowledge do not exist. Some of these are considered in later chapters.

We have also restricted attention here to *propositional* modal logic. We do not have first-order quantification, so that we cannot easily say, for example, that Alice knows the governors of all states. Such a statement would require universal and existential quantification. Roughly speaking, we could express it as $\forall x(State(x) \Rightarrow \exists y(K_{Alice}Governor(x, y))$: for all states x, there exists y such that Alice knows that the governor of x is y. We use only propositional modal logic throughout most of this book because it is sufficiently powerful to capture most of the situations we shall be interested in, while allowing us to avoid some of the complexities that arise in the first-order case. We briefly consider the first-order case in Section 3.7.

2.2 Adding Common Knowledge and Distributed Knowledge

The language introduced in the previous section does not allow us to express the notions of common knowledge and distributed knowledge that we discussed in Chapter 1. To express these notions, we augment the language with the modal operators E_G ("everyone in the group G knows"), C_G ("it is common knowledge among the agents in G"), and D_G ("it is distributed knowledge among the agents in G") for every nonempty subset G of $\{1, \ldots, n\}$, so that if φ is a formula, then so are $E_G\varphi$, $C_G\varphi$, and $D_G\varphi$. We often omit the subscript G when G is the set of all agents. In this augmented language we can make statements like $K_3\neg C_{\{1,2\}}p$ ("agent 3 knows that p is not common knowledge among agents 1 and 2") and $Dq \wedge \neg Cq$ ("q is distributed knowledge, but it is not common knowledge").

We can easily extend the definition of truth to handle common knowledge and distributed knowledge in a structure M. Since $E_G\varphi$ is true exactly if everyone in the group G knows φ, we have

$$(M, s) \models E_G\varphi \text{ iff } (M, s) \models K_i\varphi \text{ for all } i \in G.$$

The formula $C_G\varphi$ is true if everyone in G knows φ, everyone in G knows that everyone in G knows φ, etc. Let $E_G^0\varphi$ be an abbreviation for φ, and let $E_G^{k+1}\varphi$ be an abbreviation for $E_G E_G^k\varphi$. In particular, $E_G^1\varphi$ is an abbreviation for $E_G\varphi$. Then we have

$$(M, s) \models C_G\varphi \text{ iff } (M, s) \models E_G^k\varphi \text{ for } k = 1, 2, \ldots$$

Our definition of common knowledge has an interesting graph-theoretical interpretation, which turns out to be useful in many of our applications. Define a state t to be *G-reachable from state s in k steps* ($k \geq 1$) if there exist states $s_0, s_1, \ldots, s_k$ such that $s_0 = s$, $s_k = t$ and for all j with $0 \leq j \leq k-1$, there exists $i \in G$ such that $(s_j, s_{j+1}) \in \mathcal{K}_i$. We say t is *G-reachable from s* if t is G-reachable from s in k steps for some $k \geq 1$. Thus, t is G-reachable from s exactly if there is a path in the graph from s to t whose edges are labeled by members of G. In the particular case where G is the set of all agents, we say simply that t is *reachable from s*. Thus, t is reachable from s exactly if s and t are in the same connected component of the graph.

Lemma 2.2.1

(a) $(M, s) \models E_G^k\varphi$ *if and only if* $(M, t) \models \varphi$ *for all t that are G-reachable from s in k steps.*

(b) $(M, s) \models C_G\varphi$ *if and only if* $(M, t) \models \varphi$ *for all t that are G-reachable from s.*

Proof Part (a) follows from a straightforward induction on k, while part (b) is immediate from part (a). Notice that this result holds even if the $\mathcal{K}_i$'s are arbitrary binary relations; we do not need to assume that they are equivalence relations. ∎

A group G has distributed knowledge of φ if the "combined" knowledge of the members of G implies φ. How can we capture the idea of combining knowledge in our framework? In the Kripke structure in Figure 2.1, in state s agent 1 considers both s and t possible but does not consider u possible, while agent 2 considers s and u possible, but not t. Someone who could combine the knowledge of agents 1 and 2 would know that only s was possible: agent 1 has enough knowledge to eliminate u, and agent 2 has enough knowledge to eliminate t. In general, we combine the knowledge of the agents in group G by eliminating all worlds that some agent in G considers impossible. Technically, this is accomplished by *intersecting* the sets of worlds that each of the agents in the group considers possible. Thus we define

$$(M, s) \models D_G\varphi \text{ iff } (M, t) \models \varphi \text{ for all } t \text{ such that } (s, t) \in \cap_{i \in G}\mathcal{K}_i.$$

Returning to our card game example, let $G = \{1, 2\}$; thus, G is the group consisting of the two players in the game. Then it is easy to check (using Lemma 2.2.1) that $(M_c, (A, B)) \models C_G(1A \vee 1B \vee 1C)$: it is common knowledge that agent 1 holds one of the cards A, B, and C. Perhaps more interesting is $(M_c, (A, B)) \models C_G(1B \Rightarrow (2A \vee 2C))$: it is common knowledge that if agent 1 holds card B, then agent 2 holds either card A or card C. More generally, it can be shown that any fact about the game that can be expressed in terms of the propositions in our language is common knowledge.

What about distributed knowledge? We leave it to the reader to check that, for example, we have $(M_c, (A, B)) \models D_G(1A \wedge 2B)$. If the agents could pool their knowledge together, they would know that in world (A, B), agent 1 holds card A and agent 2 holds card B.

Again, this example does not provide complete justification for our definitions. But it should at least convince the reader that they are plausible. We examine the properties of common knowledge and distributed knowledge in more detail in Section 2.4.

2.3 The Muddy Children Revisited

In our analysis we shall assume that it is common knowledge that the father is truthful, that all the children can and do hear the father, that all the children can and do see

which of the other children besides themselves have muddy foreheads, that none of the children can see his or her own forehead, and that all the children are truthful and (extremely) intelligent.

First consider the situation before the father speaks. Suppose there are n children altogether. As before, we number them $1, \ldots, n$. Some of the children have muddy foreheads, and the rest do not. We can describe a possible situation by an n-tuple of 0's and 1's of the form $(x_1, \ldots, x_n)$, where $x_i = 1$ if child i has a muddy forehead, and $x_i = 0$ otherwise. Thus, if $n = 3$, then a tuple of the form (1, 0, 1) would say that precisely child 1 and child 3 have muddy foreheads. Suppose the actual situation is described by this tuple. What situations does child 1 consider possible before the father speaks? Since child 1 can see the foreheads of all the children besides herself, her only doubt is about whether she has mud on her own forehead. Thus child 1 considers two situations possible, namely, (1, 0, 1) (the actual situation) and (0, 0, 1). Similarly, child 2 considers two situations possible: (1, 0, 1) and (1, 1, 1). Note that in general, child i has the same information in two possible worlds exactly if they agree in all components except possibly the i^{th} component.

We can capture the general situation by a Kripke structure M consisting of 2^n states, one for each of the possible n-tuples. We must first decide what propositions we should include in our language. Since we want to reason about whether or not a given child's forehead is muddy, we take $\Phi = \{p_1, \ldots, p_n, p\}$, where, intuitively, p_i stands for "child i has a muddy forehead," and p stands for "at least one child has a muddy forehead." Thus, we define π so that $(M, (x_1, \ldots, x_n)) \models p_i$ if and only if $x_i = 1$, and $(M, (x_1, \ldots, x_n)) \models p$ if and only if $x_j = 1$ for some j. Of course, p is equivalent to $p_1 \vee \ldots \vee p_n$, so its truth value can be determined from the truth value of the other primitive propositions. There is nothing to prevent us from choosing a language where the primitive propositions are not independent. Since it is convenient to add a primitive proposition (namely p) describing the father's statement, we do so. Finally, we must define the $\mathcal{K}_i$ relations. Since child i considers a world possible if it agrees in all components except possibly the i^{th} component, we take $(s, t) \in \mathcal{K}_i$ exactly if s and t agree in all components except possibly the i^{th} component. Notice that this definition makes $\mathcal{K}_i$ an equivalence relation. This completes the description of M.

Although this Kripke structure may seem quite complicated, it actually has an elegant graphical representation. Suppose we ignore self-loops and the labeling on the edges for the moment. Then we have a structure with 2^n nodes, each described by an n-tuple of 0's and 1's, such that two nodes are joined by an edge exactly if they differ in one component. The reader with a good imagination will see that this

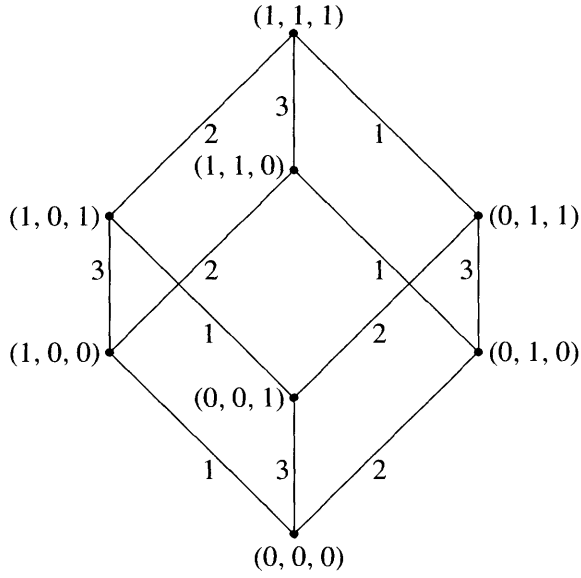

Figure 2.3 The Kripke structure for the muddy children puzzle with $n = 3$

defines an n-dimensional cube. The case $n = 3$ is illustrated in Figure 2.3 (where again we omit self-loops and the arrows on edges).

Intuitively, each child knows which of the other children have muddy foreheads. This intuition is borne out in our formal definition of knowledge. For example, it is easy to see that when the actual situation is $(1, 0, 1)$, we have $(M, (1, 0, 1)) \models K_1 \neg p_2$, since when the actual situation is $(1, 0, 1)$, child 2 does not have a muddy forehead in both worlds that child 1 considers possible. Similarly, we have $(M, (1, 0, 1)) \models K_1 p_3$: child 1 knows that child 3's forehead is muddy. However, $(M, (1, 0, 1)) \models \neg K_1 p_1$. Child 1 does not know that her own forehead is muddy, since in the other world she considers possible—(0,0,1)—her forehead is not muddy. In fact, it is common knowledge that every child knows whether every other child's forehead is muddy or not. Thus, for example, a formula like $p_2 \Rightarrow K_1 p_2$, which says that if child 2's forehead is muddy then child 1 knows it, is common knowledge. We leave it to the reader to check that $C(p_2 \Rightarrow K_1 p_2)$ is true at every state, as is $C(\neg p_2 \Rightarrow K_1 \neg p_2)$.

In the world (1,0,1), in which there are two muddy children, every child knows that at least one child has a muddy forehead even before the father speaks. And sure enough, we have $(M, (1, 0, 1)) \models Ep$. It follows, however, from Lemma 2.2.1 that

$(M, (1, 0, 1)) \models \neg E^2 p$, since p is not true at the world $(0, 0, 0)$ that is reachable in two steps from $(1, 0, 1)$. The reader can easily check that in the general case, if we have n children of whom k have muddy foreheads (so that the situation is described by an n-tuple exactly k of whose components are 1's), then $E^{k-1} p$ is true, but $E^k p$ is not, since each world (tuple) reachable in $k - 1$ steps has at least one 1 (and so there is at least one child with a muddy forehead), but the tuple $(0, \ldots, 0)$ is reachable in k steps.

Before we go on, the reader should note that there are a number of assumptions implicit in our representation. The fact that we have chosen to represent a world as an n-tuple in this way is legitimate if we can assume that all the information necessary for our reasoning already exists in such tuples. If there were some doubt as to whether child 1 was able to see, then we would have to include this information in the state description as well. Note also that the assumption that it is common knowledge that all the children can see is what justifies the choice of edges. For example, if $n = 3$ and if it were common knowledge that child 1 is blind, then, for example, in the situation $(1, 1, 1)$, child 1 would also consider $(1, 0, 0)$ possible. He would not know that child 2's forehead is muddy (see Exercises 2.1 and 2.2).

In general, when we choose to model a given situation, we have to put into the model everything that is relevant. One obvious reason that a fact may be "irrelevant" is because it does not pertain to the situation we are analyzing. Thus, for example, whether child 1 is a boy or a girl is not part of the description of the possible world. Another cause of irrelevance is that a fact may be common knowledge. If it is common knowledge that all the children can see, then there is no point in adding this information to the description of a possible world. It is true at all the possible worlds in the picture, so we do not gain anything extra by mentioning it. Thus, common knowledge can help to simplify our description of a situation.

We remark that throughout the preceding discussion we have used the term "common knowledge" in two slightly different, although related, senses. The first is the technical sense, where a formula φ in our language is common knowledge at a state s if it is true at all states reachable from s. The second is a somewhat more informal sense, where we say that a fact (not necessarily expressible in our language) is common knowledge if it is true at all the situations (states) in the structure. When we say it is common knowledge that at least one child has mud on his or her forehead, then we are using common knowledge in the first sense, since this corresponds to the formula Cp. When we say that it is common knowledge that no child is blind, we are using it in the second sense, since we do not have a formula q in the language that says that no child is blind. There is an obvious relationship between the two senses of the term. For example, if we enrich our language so that it does have a formula q

saying "no child is blind," then Cq actually would hold at every state in the Kripke structure. Throughout this book, we continue to speak of common knowledge in both senses of the term, and we hope that the reader can disambiguate if necessary.

Returning to our analysis of the puzzle, consider what happens after the father speaks. The father says p, which, as we have just observed, is already known to all the children if there are two or more children with muddy foreheads. Nevertheless, the state of knowledge changes, even if all the children already know p. If $n = 3$, in the world $(1, 0, 1)$ child 1 considers the situation $(0, 0, 1)$ possible. In that world, child 3 considers $(0, 0, 0)$ possible. Thus, in the world $(1, 0, 1)$, before the father speaks, although everyone knows that at least one child has a muddy forehead, child 1 thinks it possible that child 3 thinks it possible that none of the children has a muddy forehead. After the father speaks, it becomes *common knowledge* that at least one child has a muddy forehead. (This, of course, depends on our assumption that it is common knowledge that all the children can and do hear the father.) We can represent the change in the group's state of knowledge graphically (in the general case) by simply removing the point $(0, 0, \ldots, 0)$ from the cube, getting a "truncated" cube. (More accurately, what happens is that the node $(0, 0, \ldots, 0)$ remains, but all the edges between $(0, 0, \ldots, 0)$ and nodes with exactly one 1 disappear, since it is common knowledge that even if only one child has a muddy forehead, after the father speaks that child will not consider it possible that no one has a muddy forehead.) The situation is illustrated in Figure 2.4.

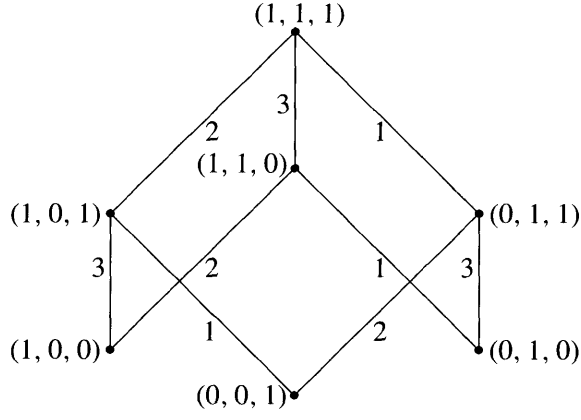

Figure 2.4 The Kripke structure after the father speaks

We next show that each time the children respond to the father's question with a "No", the group's state of knowledge changes and the cube is further truncated. Consider what happens after the children respond "No" to the father's first question. We claim that now all the nodes with exactly one 1 can be eliminated. (More accurately, the edges to these nodes from nodes with exactly two 1's all disappear from the graph.) Nodes with one or fewer 1's are no longer reachable from nodes with two or more 1's. The reasoning here parallels the reasoning in the "proof" given in the story. If the actual situation were described by, say, the tuple $(1, 0, \ldots, 0)$, then child 1 would initially consider two situations possible: $(1, 0, \ldots, 0)$ and $(0, 0, \ldots, 0)$. Once the father speaks it is common knowledge that $(0, 0, \ldots, 0)$ is not possible, so he would then know that the situation is described by $(1, 0, \ldots, 0)$, and thus would know that his own forehead is muddy. Once everyone answers "No" to the father's first question, it is common knowledge that the situation cannot be $(1, 0, \ldots, 0)$. (Note that here we must use the assumption that it is common knowledge that everyone is intelligent and truthful, and so can do the reasoning required to show $(1, 0, \ldots, 0)$ is not possible.) Similar reasoning allows us to eliminate every situation with exactly one 1. Thus, after all the children have answered "No" to the father's first question, it is common knowledge that at least *two* children have muddy foreheads.

Further arguments in the same spirit can be used to show that after the children answer "No" k times, we can eliminate all the nodes with at most k 1's (or, more accurately, disconnect these nodes from the rest of the graph). We thus have a sequence of Kripke structures, describing the children's knowledge at every step in the process. Essentially, what is going on is that if, in some node s, it becomes common knowledge that a node t is impossible, then for every node u reachable from s, the edge from u to t (if there is one) is eliminated. (This situation is even easier to describe once we add time to the picture. We return to this point in Chapter 7; see in particular Section 7.2.)

After k rounds of questioning, it is common knowledge that at least $k+1$ children have mud on their foreheads. If the true situation is described by a tuple with exactly $k+1$ entries of 1, then before the father asks the question for the $(k+1)^{\text{st}}$ time, those children with muddy foreheads will know the exact situation, and in particular will know their foreheads are muddy, and consequently will answer "Yes." Note that they could not answer "Yes" any earlier, since up to this point each child with a muddy forehead considers it possible that he or she does not have a muddy forehead.

There is actually a subtle point that should be brought out here. Roughly speaking, according to the way we are modeling "knowledge" in this context, a child "knows" a fact if the fact follows from his or her current information. But we could certainly imagine that if one of the children were not particularly bright, then he might not

be able to figure out that he "knew" that his forehead was muddy, even though in principle he had enough information to do so. To answer "Yes" to the father's question, it really is not enough for it to follow from the child's information whether the child has a muddy forehead. The child must actually be aware of the consequences of his information—that is, in some sense, the child must be able to compute that he has this knowledge—in order to act on it. Our definition implicitly assumes that all reasoners are *logically omniscient*, i.e., they are smart enough to compute all the consequences of the information that they have.

Now consider the situation in which the father does not initially say p. We claim that in this case the children's state of knowledge never changes, no matter how many times the father asks questions. It can always be described by the n-dimensional cube. We have already argued that before the father speaks the situation is described by the n-dimensional cube. When the father asks for the first time "Do any of you know whether you have mud on your own forehead?," clearly all the children say "No," no matter what the actual situation is, since in every situation each child considers possible a situation in which he or she does not have a muddy forehead. Since it is common knowledge before the father asks his question that the answer will be "No", no information is gained from this answer, so the situation still can be represented by the n-dimensional cube. Now a straightforward induction on m shows that it is common knowledge that the father's m^{th} question is also answered "No" (since at the point when the father asks this question, no matter what the situation is, each child will consider possible another situation in which he does not have a muddy forehead), and the state of knowledge after the father asks the m^{th} question is still described by the cube.

This concludes our analysis of the muddy children puzzle.

2.4 The Properties of Knowledge

In the first part of this chapter we described a language with modal operators such as K_i and defined a notion of truth that, in particular, determines whether a formula such as $K_i\varphi$ is true at a particular world. We suggested that $K_i\varphi$ should be read as "agent i knows φ." But is this a reasonable way of reading this formula? Does our semantics—that is, Kripke structures together with the definition of truth that we gave—really capture the properties of knowledge in a reasonable way? How can we even answer this question?

We can attempt to answer the question by examining what the properties of knowledge are under our interpretation. One way of characterizing the properties

of our interpretation of knowledge is by characterizing the formulas that are always true. More formally, given a structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, we say that φ is *valid in* M, and write $M \models \varphi$, if $(M, s) \models \varphi$ for every state s in S, and we say that φ is *satisfiable in* M if $(M, s) \models \varphi$ for some state s in S. We say that φ is *valid*, and write $\models \varphi$, if φ is valid in all structures, and that φ is *satisfiable* if it is satisfiable in some structure. It is easy to check that a formula φ is valid (resp. valid in M) if and only if $\neg\varphi$ is not satisfiable (resp. not satisfiable in M).

We now list a number of valid properties of our definition of knowledge and provide a formal proof of their validity. We then discuss how reasonable these properties are. As before, we assume throughout this section that the possibility relations $\mathcal{K}_i$ are equivalence relations.

One important property of our definition of knowledge is that each agent knows all the logical consequences of his knowledge. If an agent knows φ and knows that φ implies ψ, then both φ and $\varphi \Rightarrow \psi$ are true at all worlds he considers possible. Thus ψ must be true at all worlds that the agent considers possible, so he must also know ψ. It follows that

$$\models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi.$$

This axiom is called the *Distribution Axiom* since it allows us to distribute the K_i operator over implication. It seems to suggest that our agents are quite powerful reasoners.

Further evidence that our definition of knowledge assumes rather powerful agents comes from the fact that agents know all the formulas that are valid in a given structure. If φ is true at all the possible worlds of structure M, then φ must be true at all the worlds that an agent considers possible in any given world in M, so it must be the case that $K_i\varphi$ is true at all possible worlds of M. More formally, we have the following *Knowledge Generalization Rule*:

$$\text{For all structures } M, \text{ if } M \models \varphi, \text{ then } M \models K_i\varphi.$$

Note that from this we can deduce that if φ is valid, then so is $K_i\varphi$. This rule is very different from the formula $\varphi \Rightarrow K_i\varphi$, which says that if φ is true, then agent i knows it. An agent does not necessarily know all things that are true. (For example, in the case of the muddy children, it may be true that child 1 has a muddy forehead, but he does not necessarily know this.) However, agents do know all valid formulas. Intuitively, these are the formulas that are *necessarily* true, as opposed to the formulas that just happen to be true at a given world.

Although an agent may not know facts that are true, it is the case that if an agent knows a fact, then it is true. More formally, we have

$$\models K_i\varphi \Rightarrow \varphi.$$

This property, occasionally called the *Knowledge Axiom* or the *Truth Axiom* (for knowledge), has been taken by philosophers to be the major one distinguishing knowledge from *belief*. Although you may have false beliefs, you cannot know something that is false. This property follows because the actual world is always one of the worlds that an agent considers possible. If $K_i\varphi$ holds at a particular world (M, s), then φ is true at all worlds that i considers possible, so in particular it is true at (M, s).

The last two properties we consider say that agents can do introspection regarding their knowledge. They know what they know and what they do not know:

$$\models K_i\varphi \Rightarrow K_iK_i\varphi,$$
$$\models \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi.$$

The first of these properties is typically called the *Positive Introspection Axiom*, and the second is called the *Negative Introspection Axiom*.

The following theorem provides us with formal assurance that all the properties just discussed hold for our definition of knowledge.

Theorem 2.4.1 *For all formulas φ and ψ, all structures M where each possibility relation $\mathcal{K}_i$ is an equivalence relation, and all agents $i = 1, \ldots, n$,*

(a) $M \models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$,

(b) if $M \models \varphi$, *then* $M \models K_i\varphi$,

(c) $M \models K_i\varphi \Rightarrow \varphi$,

(d) $M \models K_i\varphi \Rightarrow K_iK_i\varphi$,

(e) $M \models \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$.

Proof

(a) If $(M, s) \models K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)$, then for all states t such that $(s, t) \in \mathcal{K}_i$, we have both that $(M, t) \models \varphi$ and $(M, t) \models \varphi \Rightarrow \psi$. By the definition of $\models$, we have that $(M, t) \models \psi$ for all such t, and therefore $(M, s) \models K_i\psi$.

(b) If $M \models \varphi$, then $(M, t) \models \varphi$ for all states t in M. In particular, for any fixed state s in M, it follows that $(M, t) \models \varphi$ for all t such that $(s, t) \in \mathcal{K}_i$. Thus, $(M, s) \models K_i\varphi$ for all states s in M, and hence $M \models K_i\varphi$.

(c) If $(M, s) \models K_i\varphi$, then for all t such that $(s, t) \in \mathcal{K}_i$, we have $(M, t) \models \varphi$. Since $\mathcal{K}_i$ is reflexive, it follows that $(s, s) \in \mathcal{K}_i$, so in particular $(M, s) \models \varphi$.

(d) Suppose $(M, s) \models K_i\varphi$. Consider any t such that $(s, t) \in \mathcal{K}_i$ and any u such that $(t, u) \in \mathcal{K}_i$. Since $\mathcal{K}_i$ is transitive, we have $(s, u) \in \mathcal{K}_i$. Since $(M, s) \models K_i\varphi$, it follows that $(M, u) \models \varphi$. Thus, for all t such that $(s, t) \in \mathcal{K}_i$, we have $(M, t) \models K_i\varphi$. It now follows that $(M, s) \models K_iK_i\varphi$.

(e) Suppose $(M, s) \models \neg K_i\varphi$. Then for some u with $(s, u) \in \mathcal{K}_i$, we must have $(M, u) \models \neg\varphi$. Suppose t is such that $(s, t) \in \mathcal{K}_i$. Since $\mathcal{K}_i$ is symmetric, $(t, s) \in \mathcal{K}_i$, and since $\mathcal{K}_i$ is transitive, we must also have $(t, u) \in \mathcal{K}_i$. Thus it follows that $(M, t) \models \neg K_i\varphi$. Since this is true for all t such that $(s, t) \in \mathcal{K}_i$, we obtain $(M, s) \models K_i\neg K_i\varphi$. ∎

The collection of properties that we have considered so far—the Distribution Axiom, the Knowledge Axiom, the Positive and Negative Introspection Axioms, and the Knowledge Generalization Rule—has been studied in some depth in the literature. For historical reasons, these properties are sometimes called the *S5 properties*. (Actually, S5 is an axiom system. We give a more formal definition of it in the next chapter.) How reasonable are these properties? The proof of Theorem 2.4.1 shows that, in a precise sense, the validity of the Knowledge Axiom follows from the fact that $\mathcal{K}_i$ is reflexive, the validity of the Positive Introspection Axiom follows from the fact that $\mathcal{K}_i$ is transitive, and the validity of the Negative Introspection Axiom follows from the fact that $\mathcal{K}_i$ is symmetric and transitive. Although taking $\mathcal{K}_i$ to be an equivalence relation seems reasonable for many applications we have in mind, we can certainly imagine other possibilities. As we show in Chapter 3, by modifying the properties of the $\mathcal{K}_i$ relations, we can get notions of knowledge that have different properties.

Two properties that seem forced on us by the possible-worlds approach itself are the Distribution Axiom and the Knowledge Generalization Rule. No matter how we modify the $\mathcal{K}_i$ relations, these properties hold. (This is proved formally in the next chapter.) These properties may be reasonable if we identify "agent i knows φ" with "φ follows from agent i's information," as we implicitly did when modeling the muddy children puzzle. To the extent that we think of knowledge as something acquired by agents through some reasoning process, these properties suggest that we must think

in terms of agents who can do perfect reasoning. While this may be a reasonable idealization in certain circumstances (and is an assumption that is explicitly made in the description of the muddy children puzzle), it is clearly not so reasonable in many contexts. In Chapters 9 and 10 we discuss how the possible-worlds model can be modified to accommodate imperfect, "non-ideal" reasoners.

The reader might wonder at this point if there are other important properties of our definition of knowledge that we have not yet mentioned. Although, of course, a number of additional properties follow from the basic S5 properties, in a precise sense the S5 properties completely characterize our definition of knowledge, at least as far as the K_i operators are concerned. This point is discussed in detail in Chapter 3.

We now turn our attention to the properties of the operators E_G, C_G, and D_G. Since $E_G\varphi$ is true exactly if every agent in G knows φ, we have

$$\models E_G\varphi \Leftrightarrow \bigwedge_{i\in G} K_i\varphi.$$

Recall that we said common knowledge could be viewed as what "any fool" knows. Not surprisingly, it turns out that common knowledge has all the properties of knowledge; axioms analogous to the Knowledge Axiom, Distribution Axiom, Positive Introspection Axiom, and Negative Introspection Axiom all hold for common knowledge (see Exercise 2.8). In addition, it is easy to see that common knowledge among a group of agents implies common knowledge among any of its subgroups, that is, $C_G\varphi \Rightarrow C_{G'}\varphi$ if $G \supseteq G'$ (again, see Exercise 2.8). It turns out that all these properties follow from two other properties, which in a precise sense capture the essence of common knowledge. We discuss these properties next.

Recall from Chapter 1 that the children in the muddy children puzzle acquire common knowledge of the fact p (that at least one child has a muddy forehead) because the father's announcement puts them in a situation where all the children know both that p is true and that they are in this situation. This observation is generalized in the following *Fixed-Point Axiom*, which says that φ is common knowledge among the group G if and only if all the members of G know that φ is true and is common knowledge:

$$\models C_G\varphi \Leftrightarrow E_G(\varphi \wedge C_G\varphi).$$

Thus, the Fixed-Point Axiom says that $C_G\varphi$ can be viewed as a *fixed point* of the function $f(x) = E_G(\varphi \wedge x)$, which maps a formula x to the formula $E_G(\varphi \wedge x)$. (We shall see a formalization of this intuition in Section 11.5.)

The second property of interest gives us a way of deducing that common knowledge holds in a structure.

For all structures M, if $M \models \varphi \Rightarrow E_G(\psi \wedge \varphi)$, then $M \models \varphi \Rightarrow C_G\psi$.

This rule is often called the *Induction Rule*. The proof that it holds shows why: the antecedent gives us the essential ingredient for proving, by induction on k, that $\varphi \Rightarrow E^k(\psi \wedge \varphi)$ is valid for all k.

We now prove formally that these properties do indeed hold for the operators E_G and C_G.

Theorem 2.4.2 *For all formulas φ and ψ, all structures M, and all nonempty $G \subseteq \{1, \ldots, n\}$,*

(a) $M \models E_G\varphi \Leftrightarrow \bigwedge_{i \in G} K_i\varphi$,

(b) $M \models C_G\varphi \Leftrightarrow E_G(\varphi \wedge C_G\varphi)$,

(c) if $M \models \varphi \Rightarrow E_G(\psi \wedge \varphi)$, *then* $M \models \varphi \Rightarrow C_G\psi$.

Proof Part (a) follows immediately from the semantics of E_G. To prove the other parts, we use the characterization of common knowledge provided by Lemma 2.2.1, namely, that $(M, s) \models C_G\varphi$ iff $(M, t) \models \varphi$ for all states t that are G-reachable from s. We remark for future reference that the proof we are about to give does not make use of the fact that the $\mathcal{K}_i$'s are equivalence relations; it goes through without change even if the $\mathcal{K}_i$'s are arbitrary binary relations.

For part (b), suppose $(M, s) \models C_G\varphi$. Thus $(M, t) \models \varphi$ for all states t that are G-reachable from s. In particular, if u is G-reachable from s in one step, then $(M, u) \models \varphi$ and $(M, t) \models \varphi$ for all t that are G-reachable from u. Thus $(M, u) \models \varphi \wedge C_G\varphi$ for all u that are G-reachable from s in one step, so $(M, s) \models E_G(\varphi \wedge C_G\varphi)$. For the converse, suppose $(M, s) \models E_G(\varphi \wedge C_G\varphi)$. Suppose that t is G-reachable from s and s' is the first node after s on a path from s to t whose edges are labeled by members of G. Since $(M, s) \models E_G(\varphi \wedge C_G\varphi)$, it follows that $(M, s') \models \varphi \wedge C_G\varphi$. Either $s' = t$ or t is reachable from s'. In the former case, $(M, t) \models \varphi$ since $(M, s') \models \varphi$, while in the latter case, $(M, t) \models \varphi$ by Lemma 2.2.1 and the fact that $(M, s') \models C_G\varphi$. Because $(M, t) \models \varphi$ for all t that are G-reachable from s, it follows that $(M, s) \models C_G\varphi$.

Finally, for part (c), suppose $M \models \varphi \Rightarrow E_G(\psi \wedge \varphi)$ and $(M, s) \models \varphi$. We show by induction on k that for all k we have $(M, t) \models \psi \wedge \varphi$ for all t that are G-reachable from s in k steps. Suppose t is G-reachable from s in one step. Since $M \models \varphi \Rightarrow E_G(\psi \wedge \varphi)$, we have $(M, s) \models E_G(\psi \wedge \varphi)$. Since t is G-reachable from s in one step, by Lemma 2.2.1, we have $(M, t) \models \psi \wedge \varphi$ as desired. If $k = k' + 1$, then there is some t' that is G-reachable from s in k' steps such that t is G-reachable from t' in one step. By the induction hypothesis, we have $(M, t') \models \psi \wedge \varphi$. Now the same argument as in the base case shows that $(M, t) \models \psi \wedge \varphi$. This completes

the inductive proof. Since $(M, t) \models \psi$ for all states t that are G-reachable from s, it follows that $(M, s) \models C_G\varphi$. ∎

Finally, we consider distributed knowledge. We mentioned in Chapter 1 that distributed knowledge can be viewed as what a "wise man" would know. So it should not be surprising that distributed knowledge also satisfies all the properties of knowledge. Distributed knowledge has two other properties that we briefly mention here. Clearly, distributed knowledge of a group of size one is the same as knowledge, so that we have

$$\models D_{\{i\}}\varphi \Leftrightarrow K_i\varphi.$$

The larger the subgroup, the greater the distributed knowledge of that subgroup:

$$\models D_G\varphi \Rightarrow D_{G'}\varphi \text{ if } G \subseteq G'.$$

The proof that all these properties of distributed knowledge are indeed valid is similar in spirit to the proof of Theorem 2.4.1, so we leave it to the reader (Exercise 2.10). We also show in Chapter 3 that these properties of common knowledge and distributed knowledge in a precise sense completely characterize all the relevant properties of these notions.

2.5 An Event-Based Approach

The approach to modeling knowledge presented in Section 2.1 has two components. It uses Kripke structures as a mathematical model for situations involving many agents, and it uses a logical language to make assertions about such situations. This language is based on a set of primitive propositions and is closed under logical operators. Thus, knowledge is expressed syntactically, by modal operators on formulas. We call this the *logic-based approach*. It is the approach that traditionally has been taken in philosophy, mathematical logic, and AI.

In this section, we describe an alternate approach to modeling knowledge, one that is typically used in the work on knowledge in game theory and mathematical economics. We call this the *event-based approach*. It differs from the logic-based approach in two respects. First, rather than use Kripke structures as the underlying mathematical model, the event-based approach uses closely related structures that we call *Aumann structures*. Second, and more important, in the spirit of probability theory, the event-based approach focuses on *events*, which are sets of possible worlds, and dispenses completely with logical formulas. Knowledge here is expressed as an

operator on events. We now review the event-based approach and discuss its close relationship to the logic-based approach.

As in the logic-based approach of Section 2.1, we start out with a universe S of *states*. An *event* is a set $e \subseteq S$ of states. We can talk, for example, about the event of its raining in London, which corresponds to the set of states where it is raining in London. We say that *event e holds at state s* if $s \in e$. Thus, if e_L is the event of its raining in London, then e_L holds at state s precisely if s is one of the states where it is raining in London. The conjunction of two events is given by their intersection. For example, the event of its raining in London and being sunny in San Francisco is the intersection of e_L with the event of its being sunny in San Francisco. Similarly, the negation of an event is given by the complement (with respect to S).

As we have mentioned, Aumann structures are used to provide a formal model for the event-based approach. Aumann structures are like Kripke structures, with two differences: The first is that there is no analogue to the π function, since in the event-based approach, there are no primitive propositions. The second difference is that, rather than use a binary relation $\mathcal{K}_i$ to define what worlds agent i considers possible, in Aumann structures there is a *partition* $\mathcal{P}_i$ of S for each agent i. (A *partition* of a set S is a set $\{S_1, \ldots, S_r\}$ of subsets of S such that the S_j's are disjoint and such that the union of the S_j's is the set S.) If $\mathcal{P}_i = \{S_1, \ldots, S_r\}$, then the sets S_j are called the *cells* of the partition $\mathcal{P}_i$, or the *information sets* of agent i. The intuition is that if S_j is an information set of agent i, and if $s \in S_j$, then the set of states that agent i considers possible (which corresponds to the information of agent i) is precisely S_j.

Formally, an Aumann structure A is a tuple $(S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$, where S is the set of states of the world and $\mathcal{P}_i$ is a partition of S for every agent i. We denote by $\mathcal{P}_i(s)$ the cell of the partition $\mathcal{P}_i$ in which s appears. Since $\mathcal{P}_i$ is a partition, it follows that for every agent i and every pair $s, t \in S$ of states, either $\mathcal{P}_i(s) = \mathcal{P}_i(t)$ or $\mathcal{P}_i(s) \cap \mathcal{P}_i(t) = \emptyset$. Intuitively, when s, t are in the same information set of agent i, then in state s agent i considers the state t possible. As we have already remarked, unlike a Kripke structure, in an Aumann structure there is no function π that associates with each state in S a truth assignment to primitive propositions. (Using terminology we introduce in the next chapter, this means that an Aumann structure is really a *frame*.)

How do we define knowledge in the event-based approach? Since the objects of interest in this approach are events, it should not be surprising that knowledge is defined in terms of events. Formally, given an Aumann structure $(S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$, we define knowledge operators $\mathsf{K}_i : 2^S \to 2^S$, for $i = 1, \ldots, n$, as follows:

$$\mathsf{K}_i(e) = \{s \in S \mid \mathcal{P}_i(s) \subseteq e\};$$

$\mathsf{K}_i(e)$ is called the event of *i knowing e*. Here 2^S is the set of all subsets of S. (Note that we use sans serif font for the knowledge operator K_i, in contrast to the italic font that we use for the modal operator K_i, and the script font we use for the binary relation $\mathcal{K}_i$.) It is easy to see that $\mathsf{K}_i(e)$ is the union of the information sets of agent i that are contained in e. The intuition is that agent i knows e at state s if e holds at every state that agent i considers possible at state s (namely, at all states of $\mathcal{P}_i(s)$). Thus, agent i knows that no matter what the actual state is, the event e holds there.

The event of *everyone in a group G knowing e* is captured by an operator $\mathsf{E}_G : 2^S \rightarrow 2^S$ defined as follows:

$$\mathsf{E}_G(e) = \bigcap_{i \in G} \mathsf{K}_i(e).$$

We can iterate the E_G operator, defining $\mathsf{E}_G^1(e) = \mathsf{E}_G(e)$ and $\mathsf{E}_G^{k+1}(e) = \mathsf{E}_G(\mathsf{E}_G^k(e))$ for $k \geq 1$. *Common knowledge of an event e among the agents in a group G*, denoted $\mathsf{C}_G(e)$, is the event of the players all knowing e, all knowing that all know it, and so on ad infinitum. Formally, we define

$$\mathsf{C}_G(e) = \bigcap_{k=1}^{\infty} \mathsf{E}_G^k(e).$$

Finally, *distributed knowledge of an event e among the agents in a group G*, denoted $\mathsf{D}_G(e)$, is defined by

$$\mathsf{D}_G(e) = \left\{ s \in S \mid \left(\bigcap_{i \in G} \mathcal{P}_i(s) \right) \subseteq e \right\}.$$

Intuitively, event e is distributed knowledge if e holds at all of the states that remain possible once we combine the information available to all of the agents.

Given two partitions $\mathcal{P}$ and $\mathcal{P}'$ of a set S, the partition $\mathcal{P}$ is said to be *finer* than $\mathcal{P}'$ (and $\mathcal{P}'$ to be *coarser* than $\mathcal{P}$) if $\mathcal{P}(s) \subseteq \mathcal{P}'(s)$ holds for all $s \in S$. Intuitively, if partition $\mathcal{P}$ is finer than partition $\mathcal{P}'$, then the information sets given by $\mathcal{P}$ give at least as much information as the information sets given by $\mathcal{P}'$ (since considering fewer states possible corresponds to having more information). The *meet* of partitions $\mathcal{P}$ and $\mathcal{P}'$, denoted $\mathcal{P} \sqcap \mathcal{P}'$, is the finest partition that is coarser than $\mathcal{P}$ and $\mathcal{P}'$; the *join* of $\mathcal{P}$ and $\mathcal{P}'$, denoted $\mathcal{P} \sqcup \mathcal{P}'$, is the coarsest partition finer than $\mathcal{P}$ and $\mathcal{P}'$. In the next proposition, we make use of the meet and the join to give nice characterizations of common knowledge and distributed knowledge.

Proposition 2.5.1 *Let $A = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ be an Aumann structure, let $G \subseteq \{1, \ldots, n\}$ be a group of agents, and let $e \subseteq S$. Then*

(a) $s \in \mathsf{C}_G(e)$ iff $(\sqcap_{i \in G} \mathcal{P}_i)(s) \subseteq e$.

(b) $s \in \mathsf{D}_G(e)$ iff $(\sqcup_{i \in G} \mathcal{P}_i)(s) \subseteq e$.

Proof See Exercise 2.15. ∎

It follows that the meet of the agents' partitions characterizes their common knowledge, and the join of the agents' partitions characterizes their distributed knowledge. Notice that Proposition 2.5.1(a) implies that verifying whether an event e is common knowledge at a given state s can be done by one simple check of inclusion between two well-defined sets; it is unnecessary to use the definition of common knowledge, which involves an infinitary intersection.

There is a close connection between the logic-based approach and the event-based approach, which we now formalize. There is a natural one-to-one correspondence between partitions on S and equivalence relations on S. Given a partition $\mathcal{P}$ of S, the corresponding equivalence relation $\mathcal{R}$ is defined by $(s, s') \in \mathcal{R}$ iff $\mathcal{P}(s) = \mathcal{P}(s')$. Similarly, given an equivalence relation $\mathcal{R}$ on S, the corresponding partition $\{S_1, \ldots, S_r\}$ of S is obtained by making each equivalence class of $\mathcal{R}$ into a cell S_j of the partition; that is, two states s, t are in the same cell of the partition precisely if $(s, t) \in \mathcal{R}$. It is thus easy to convert back and forth between the partition viewpoint and the equivalence relations viewpoint (see Exercise 2.16).

Assume now that we are given a Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where each $\mathcal{K}_i$ is an equivalence relation. We define the corresponding Aumann structure $A^M = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ (with the same set S of states) by taking $\mathcal{P}_i$ to be the partition corresponding to the equivalence relation $\mathcal{K}_i$. We want to show that M and A^M have the same "semantics." The semantics in M is defined in terms of formulas. The *intension* of a formula φ in structure M, denoted φ^M, is the set of states of M at which φ holds, i.e., $\varphi^M = \{s \mid (M, s) \models \varphi\}$. The semantics in A^M is defined in terms of events. For each primitive proposition p, define e_p^M to be the event that p is true; that is, $e_p^M = \{s \mid (M, s) \models p\}$. We can now define an event $\mathsf{ev}_M(\varphi)$ for each formula φ by induction on the structure of φ:

- $\mathsf{ev}_M(p) = e_p^M$
- $\mathsf{ev}_M(\psi_1 \wedge \psi_2) = \mathsf{ev}_M(\psi_1) \cap \mathsf{ev}_M(\psi_2)$
- $\mathsf{ev}_M(\neg\psi) = S - \mathsf{ev}_M(\psi)$
- $\mathsf{ev}_M(K_i\psi) = \mathsf{K}_i(\mathsf{ev}_M(\psi))$

- $\mathsf{ev}_M(C_G\psi) = \mathsf{C}_G(\mathsf{ev}_M(\psi))$
- $\mathsf{ev}_M(D_G\psi) = \mathsf{D}_G(\mathsf{ev}_M(\psi))$

Intuitively, $\mathsf{ev}_M(\varphi)$ is the event that φ holds. The following proposition shows that this intuition is correct, that is, that the formula φ holds at state s of the Kripke structure M iff $\mathsf{ev}_M(\varphi)$ holds at state s of the Aumann structure A^M.

Proposition 2.5.2 *Let M be a Kripke structure where each possibility relation $\mathcal{K}_i$ is an equivalence relation, and let A^M be the corresponding Aumann structure. Then for every formula φ, we have $\mathsf{ev}_M(\varphi) = \varphi^M$.*

Proof See Exercise 2.17. ∎

We have just shown how to go from a Kripke structure to a corresponding Aumann structure. What about the other direction? Let $A = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ be an Aumann structure. We want to define a corresponding Kripke structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ (with the same set S of states). Defining the $\mathcal{K}_i$'s is no problem: we simply take $\mathcal{K}_i$ to be the equivalence relation corresponding to the partition $\mathcal{P}_i$. What about the set Φ of primitive propositions and the function π that associates with each state in S a truth assignment to primitive propositions? Although an Aumann structure does not presuppose the existence of a set of primitive propositions, in concrete examples there typically are names for basic events of interest, such as "Alice wins the game" or "the deal is struck." These names can be viewed as primitive propositions. It is also usually clear at which states these named events hold; this gives us the function π. To formalize this, assume that we are given not only the Aumann structure A but also an arbitrary set Φ of primitive propositions and an arbitrary function π that associates with each state in S a truth assignment to primitive propositions in Φ. We can now easily construct a Kripke structure $M^{A,\pi}$, which corresponds to A and π. If $A = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$, then $M^{A,\pi} = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where $\mathcal{K}_i$ is the partition corresponding to $\mathcal{P}_i$, for $i = 1, \ldots, n$. It is straightforward to show that the Aumann structure corresponding to $M^{A,\pi}$ is A (see Exercise 2.18). Thus, by Proposition 2.5.2, the intensions of formulas in $M^{A,\pi}$ and the events corresponding to these formulas in A coincide.

Proposition 2.5.2 and the preceding discussion establish the close connection between the logic-based and event-based approaches that we claimed previously.

Exercises

2.1 Suppose it is common knowledge that all the children in the muddy children puzzle are blind. What would the graphical representation be of the Kripke structure describing the situation before the father speaks? What about after the father speaks?

* **2.2** Consider the following variant of the muddy children puzzle. Suppose that it is common knowledge that all the children except possibly child 1 are paying attention when the father speaks. Moreover, suppose the children have played this game with the father before, and it is common knowledge that when he speaks he says either "At least one of you has mud on your forehead" or a vacuous statement such as "My, this field is muddy." (Thus it is common knowledge that even if child 1 did not hear the father, he knows that the father made one of those statements.)

(a) Describe the situation (i.e., the Kripke structure) after the father's statement. (Hint: each possible world can be characterized by an $(n + 2)$-tuple, where n is the total number of children.) Draw the Kripke structure for the case $n = 2$.

(b) Can the children figure out whether or not they are muddy? (Hint: first consider the case where child 1 is not muddy, then consider the case where he is muddy and hears the father, and finally consider the case where he is muddy and does not hear the father.)

(c) Can the children figure out whether or not they are muddy if the father says at the beginning "Two or more of you have mud on your forehead"?

2.3 Here is yet another variant of the muddy children puzzle. Suppose that the father says "Child number 1 has mud on his forehead" instead of saying "At least one of you has mud on your forehead." It should not be too hard to convince yourself that now the children (other than child 1) cannot deduce whether they have mud on their foreheads. Explain why this should be so (i.e., why the children cannot solve the puzzle in a situation where they apparently have *more* information). This example shows that another assumption inherent in the puzzle is that all relevant information has been stated in the puzzle, and in particular, that the father said no more than "At least one of you has mud on your forehead."

* **2.4** In this exercise, we formalize the aces and eights game from Exercise 1.1.

(a) What are the possible worlds for this puzzle if the suit of the card matters? How many possible worlds are there?

(b) Now suppose we ignore the suit (so, for example, we do not distinguish a hand with the ace of clubs and the ace of hearts from a hand with the ace of spades and the ace of hearts). How many possible worlds are there in this case? Since the suit does not matter in the puzzle, we still get an adequate representation for the puzzle if we ignore it. Since there are so many fewer possible worlds to consider in this case, it is certainly a worthwhile thing to do.

(c) Draw the Kripke structure describing the puzzle.

(d) Consider the situation described in part (a) of Exercise 1.1. Which edges disappear from the structure when you hear that Alice and Bob cannot determine what cards they have? Explain why it is now crucial to think of edges, not nodes, as disappearing.

(e) Now consider the situation described in part (b) of Exercise 1.1 and show which edges disappear from the structure.

* **2.5** In this exercise, we formalize the wise men puzzle from Exercise 1.3.

(a) Consider the first version of the puzzle (as described in part (a) of Exercise 1.3). Draw the Kripke structure describing the initial situation. How does the structure change after the first wise man says that he does not know the color of the hat on his head? How does it change after the second wise man says that he does not know?

(b) How does the initial Kripke structure change if the third wise man is blind?

2.6 Show that G-reachability is an equivalence relation if the $\mathcal{K}_i$ relations are reflexive and symmetric.

2.7 Show that if t is G-reachable from s, then $(M, s) \models C_G\varphi$ iff $(M, t) \models C_G\varphi$, provided that the $\mathcal{K}_i$ relation is reflexive and symmetric.

2.8 Show that the following properties of common knowledge are valid, using semantic arguments as in Theorems 2.4.1 and 2.4.2:

(a) $(C_G\varphi \wedge C_G(\varphi \Rightarrow \psi)) \Rightarrow C_G\psi$,

(b) $C_G\varphi \Rightarrow \varphi$ (assuming that the $\mathcal{K}_i$ relations are reflexive),

(c) $C_G\varphi \Rightarrow C_G C_G\varphi$,

(d) $\neg C_G\varphi \Rightarrow C_G\neg C_G\varphi$ (assuming that the $\mathcal{K}_i$ relations are symmetric),

(e) $C_G\varphi \Rightarrow C_{G'}\varphi$ if $G \supseteq G'$.

As is shown in Exercise 3.11, these properties are actually provable from the properties of knowledge and common knowledge described in this chapter.

2.9 Show that if $M \models \varphi \Rightarrow \psi$, then

(a) $M \models K_i\varphi \Rightarrow K_i\psi$,

(b) $M \models C_G\varphi \Rightarrow C_G\psi$.

2.10 Show that the following properties of distributed knowledge are valid:

(a) $(D_G\varphi \wedge D_G(\varphi \Rightarrow \psi)) \Rightarrow D_G\psi$,

(b) $D_G\varphi \Rightarrow \varphi$ (assuming that the $\mathcal{K}_i$ relations are reflexive),

(c) $D_G\varphi \Rightarrow D_G D_G\varphi$ (assuming that the $\mathcal{K}_i$ relations are transitive),

(d) $\neg D_G\varphi \Rightarrow D_G\neg D_G\varphi$ (assuming that the $\mathcal{K}_i$ relations are symmetric and transitive),

(e) $D_{\{i\}}\varphi \Leftrightarrow K_i\varphi$,

(f) $D_G\varphi \Rightarrow D_{G'}\varphi$ if $G \subseteq G'$.

2.11 Prove using semantic arguments that knowledge and common knowledge distribute over conjunction; i.e., prove that the following properties are valid:

(a) $K_i(\varphi \wedge \psi) \Leftrightarrow (K_i\varphi \wedge K_i\psi)$,

(b) $C_G(\varphi \wedge \psi) \Leftrightarrow (C_G\varphi \wedge C_G\psi)$.

It can also be shown that these properties follow from the properties described for knowledge and common knowledge in the text (Exercise 3.31).

2.12 Prove that the following formulas are valid assuming that the $\mathcal{K}_i$ relations are equivalence relations:

(a) $\models \neg\varphi \Rightarrow K_i\neg K_i\varphi$,

(b) $\models \neg\varphi \Rightarrow K_{i_1} \ldots K_{i_k}\neg K_{i_k} \ldots K_{i_1}\varphi$ for any sequence $i_1, \ldots, i_k$ of agents,

(c) $\models \neg K_i \neg K_i \varphi \Leftrightarrow K_i \varphi$.

These formulas are also provable from the S5 properties we discussed; see Exercise 3.14.

2.13 Let $A = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ be an Aumann structure, and let $G \subseteq \{1, \ldots, n\}$. If s, t are states, we say that *t is G-reachable from s in A* if t is reachable from s in a Kripke structure $M^{A,\pi}$ corresponding to A. Prove that $t \in (\sqcap_{i \in G} \mathcal{P}_i)(s)$ iff t is G-reachable from s.

2.14 Let $A = (S, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ be an Aumann structure and let $G \subseteq \{1, \ldots, n\}$. Prove that $t \in (\sqcup_{i \in G} \mathcal{P}_i)(s)$ iff for every agent i we have $t \in \mathcal{P}_i(s)$.

2.15 Prove Proposition 2.5.1. (Hint: you may either prove this directly, or use Exercises 2.13 and 2.14.)

2.16 Show that the correspondence we have given between partitions and equivalence relations and the correspondence defined in the other direction are inverses. That is, show that $\mathcal{R}$ is the equivalence relation that we obtain from a partition $\mathcal{P}$ iff $\mathcal{P}$ is the partition that we obtain from the equivalence relation $\mathcal{R}$.

2.17 Let M be a Kripke structure where each possibility relation $\mathcal{K}_i$ is an equivalence relation, and let A be the corresponding Aumann structure.

(a) Prove that

(i) $s \in \mathsf{K}_i(\mathsf{ev}(\varphi))$ holds in A iff $(M, s) \models K_i \varphi$,

(ii) $s \in \mathsf{D}_G(\mathsf{ev}(\varphi))$ holds in A iff $(M, s) \models D_G \varphi$,

(iii) $s \in \mathsf{C}_G(\mathsf{ev}(\varphi))$ holds in A iff $(M, s) \models C_G \varphi$.

(b) Use part (a) to prove Proposition 2.5.2.

2.18 Show that the Aumann structure corresponding to the Kripke structure $M^{A,\pi}$ is A.

Notes

Modal logic was discussed by several authors in ancient times, notably by Aristotle in *De Interpretatione* and *Prior Analytics*, and by medieval logicians, but like most work before the modern period, it was nonsymbolic and not particularly systematic in approach. The first symbolic and systematic approach to the subject appears to be the work of Lewis beginning in 1912 and culminating in the book *Symbolic Logic* with Langford [1959]. Carnap [1946, 1947] suggested using possible worlds to assign semantics to modalities. Possible-worlds semantics was further developed independently by several researchers, including Bayart [1958], Hintikka [1957, 1961], Kanger [1957b], Kripke [1959], Meredith [1956], Montague [1960], and Prior [1962] (who attributed the idea to P. T. Geach), reaching its current form (as presented here) with Kripke [1963a]. Many of these authors also observed that by varying the properties of the $\mathcal{K}_i$ relations, we can obtain different properties of knowledge.

The initial work on modal logic considered only the modalities of possibility and necessity. As we mentioned in the bibliographic notes of Chapter 1, the idea of capturing the semantics of knowledge in this way is due to Hintikka, who also first observed the properties of knowledge discussed in Section 2.4.

The analysis of the muddy children puzzle in terms of Kripke structures is due to Halpern and Vardi [1991a]. Aumann structures were defined by Aumann [1976]. Aumann defines common knowledge in terms of the meet; in particular, the observation made in Proposition 2.5.1(a) is due to Aumann. A related approach, also defining knowledge as an operator on events, is studied by Orlowska [1989].

Chapter 3

Completeness and Complexity

There are four sorts of men:
He who knows not and knows not he knows not: he is a fool—shun him;
He who knows not and knows he knows not: he is simple—teach him;
He who knows and knows not he knows: he is asleep—wake him;
He who knows and knows he knows: he is wise—follow him.

Arabian proverb

In Chapter 2 we discussed the properties of knowledge (as well as of common knowledge and distributed knowledge). We attempted to characterize these properties in terms of valid formulas. All we did, however, was to list *some* valid properties. It is quite conceivable that there are additional properties of knowledge that are not consequences of the properties listed in Chapter 2. In this chapter, we give a complete characterization of the properties of knowledge. We describe two approaches to this characterization. The first approach is *proof-theoretic*: we show that all the properties of knowledge can be formally proved from the properties listed in Chapter 2. The second approach is *algorithmic*: we study algorithms that recognize the valid properties of knowledge. We also consider the computational complexity of recognizing valid properties of knowledge. Doing so will give us some insight into what makes reasoning about knowledge difficult.

When analyzing the properties of knowledge, it is useful to consider a somewhat more general framework than that of the previous chapter. Rather than restrict attention to the case where the possibility relations (the $\mathcal{K}_i$'s) are equivalence relations, we consider other binary relations as well. Although our examples show that taking the $\mathcal{K}_i$'s to be equivalence relations is reasonably well-motivated, particularly when

what an agent considers possible is determined by his information, there are certainly other choices possible. The real question is what we mean by "in world s, agent i considers world t possible."

Let us now consider an example where reflexivity might not hold. We can easily imagine an agent who refuses to consider certain situations possible, even when they are not ruled out by his information. Thus, Fred might refuse to consider it possible that his son Harry is taking illegal drugs, even if Harry is. Fred might claim to "know" that Harry is drug-free, since in all worlds Fred considers possible, Harry is indeed drug-free. In that case, Fred's possibility relation would not be reflexive; in world s where Harry is taking drugs, Fred would not consider world s possible. To see why symmetry might not hold, consider poor Fred again. Suppose in world s, Fred's wife Harriet is out visiting her friend Alice and told Fred that she would be visiting Alice. Fred, however, has forgotten what Harriet said. Without reflecting on it too much, Fred considers the world t possible, where Harriet said that she was visiting her brother Bob. Now, in fact, if Harriet had told Fred that she was visiting Bob, Fred would have remembered that fact, since Harriet had just had a fight with Bob the week before. Thus, in world t, Fred would not consider world s possible, since in world t, Fred would remember that Harriet said she was visiting Bob, rather than Alice. Perhaps with some introspection, Fred might realize that t is not possible, because in t he would have remembered what Harriet said. But people do not always do such introspection.

By investigating the properties of knowledge in a more general framework, as we do here, we can see how these properties depend on the assumptions we make about the possibility relations $\mathcal{K}_i$. In addition, we obtain general proof techniques, which in particular enable us to characterize in a precise sense the complexity of reasoning about knowledge.

This chapter is somewhat more technical than the previous ones; we have highlighted the major ideas in the text, and have left many of the details to the exercises. A reader interested just in the results may want to skip many of the proofs. However, we strongly encourage the reader who wants to gain a deeper appreciation of the techniques of modal logic to work through these exercises.

3.1 Completeness Results

As we said before, we begin by considering arbitrary Kripke structures, without the assumption that the possibility relations $\mathcal{K}_i$ are equivalence relations. Before we go on, we need to define some additional notation. Let $\mathcal{L}_n(\Phi)$ be the set of

formulas that can be built up starting from the primitive propositions in Φ, using conjunction, negation, and the modal operators $K_1, \ldots, K_n$. Let $\mathcal{L}_n^D(\Phi)$ (resp., $\mathcal{L}_n^C(\Phi)$) be the language that results when we allow in addition the modal operators D_G (resp., operators E_G and C_G), where G is a nonempty subset of $\{1, \ldots, n\}$. In addition, we consider the language $\mathcal{L}_n^{CD}(\Phi)$, where formulas are formed using all the operators C_G, D_G, and E_G. Let $\mathcal{M}_n(\Phi)$ be the class of all Kripke structures for n agents over Φ (with no restrictions on the $\mathcal{K}_i$ relations). Later we consider various subclasses of $\mathcal{M}_n(\Phi)$, obtained by restricting the $\mathcal{K}_i$ relations appropriately. For example, we consider $\mathcal{M}_n^{rst}(\Phi)$, the Kripke structures where the $\mathcal{K}_i$ relation is reflexive, symmetric, and transitive (i.e., an equivalence relation); these are precisely the structures discussed in the previous chapter. For notational convenience, we take the set Φ of primitive propositions to be fixed from now on and suppress it from the notation, writing $\mathcal{L}_n$ instead of $\mathcal{L}_n(\Phi)$, $\mathcal{M}_n$ instead of $\mathcal{M}_n(\Phi)$, and so on.

If A is a set, define $|A|$ to be the cardinality of A (i.e., the number of elements in A). We define $|\varphi|$, the *length* of a formula $\varphi \in \mathcal{L}_n^{CD}$, to be the number of symbols that occur in φ; for example, $|p \wedge E_{\{1,2\}}p| = 9$. In general, the length of a formula of the form $C_G\psi$, $E_G\psi$, or $D_G\psi$ is $2 + 2|G| + |\psi|$, since we count the elements in G as distinct symbols, as well as the commas and set braces in G. We also define what it means for ψ to be a *subformula* of φ. Informally, ψ is a subformula of φ if it is a formula that is a substring of φ. The formal definition proceeds by induction on the structure of φ: ψ is a subformula of $\varphi \in \mathcal{L}_n$ if either (a) $\psi = \varphi$ (so that φ and ψ are syntactically identical), (b) φ is of the form $\neg\varphi'$, $K_i\varphi'$, $C_G\varphi'$, $D_G\varphi'$, or $E_G\varphi'$, and ψ is a subformula of φ', or (c) φ is of the form $\varphi' \wedge \varphi''$ and ψ is a subformula of either φ' or φ''. Let $Sub(\varphi)$ be the set of all subformulas of φ. We leave it to the reader to check that $|Sub(\varphi)| \leq |\varphi|$; that is, the length of φ is an upper bound on the number of subformulas of φ (Exercise 3.1).

Although we have now dropped the restriction that the $\mathcal{K}_i$'s be equivalence relations, the definition of what it means for a formula φ in $\mathcal{L}_n^{CD}$ (or any of its sublanguages) to be true at a state s in the Kripke structure $M \in \mathcal{M}_n$ remains the same, as do the notions of validity and satisfiability. Thus, for example, $(M, s) \models K_i\varphi$ (i.e., agent i knows φ at state s in M) exactly if φ is true at all the states t such that $(s, t) \in \mathcal{K}_i$. We say that φ is *valid with respect to* $\mathcal{M}_n$, and write $\mathcal{M}_n \models \varphi$, if φ is valid in all the structures in $\mathcal{M}_n$. More generally, if $\mathcal{M}$ is some subclass of $\mathcal{M}_n$, we say that φ is *valid with respect to* $\mathcal{M}$, and write $\mathcal{M} \models \varphi$, if φ is valid in all the structures in $\mathcal{M}$. Similarly, we say that φ is satisfiable with respect to $\mathcal{M}$ if φ is satisfied in some structure in $\mathcal{M}$.

We are interested in characterizing the properties of knowledge in Kripke structures in terms of the formulas that are valid in Kripke structures. Note that we should

expect *fewer* formulas to be valid than were valid in the Kripke structures considered in the previous chapter, for we have now dropped the restriction that the $\mathcal{K}_i$'s are equivalence relations. The class $\mathcal{M}_n^{rst}$ of structures is a proper subclass of $\mathcal{M}_n$. Therefore, a formula that is valid with respect to $\mathcal{M}_n$ is certainly valid with respect to the more restricted class $\mathcal{M}_n^{rst}$. As we shall see, the converse does not hold.

We start by considering the language $\mathcal{L}_n$; we deal with common knowledge and distributed knowledge later on. We observed in the previous chapter that the Distribution Axiom and the Knowledge Generalization Rule hold no matter how we modify the $\mathcal{K}_i$ relations. Thus, the following theorem should not come as a great surprise.

Theorem 3.1.1 *For all formulas $\varphi, \psi \in \mathcal{L}_n$, structures $M \in \mathcal{M}_n$, and agents $i = 1, \ldots, n$,*

(a) if φ is an instance of a propositional tautology, then $\mathcal{M}_n \models \varphi$,

(b) if $M \models \varphi$ and $M \models \varphi \Rightarrow \psi$ then $M \models \psi$,

(c) $\mathcal{M}_n \models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$,

(d) if $M \models \varphi$ then $M \models K_i\varphi$.

Proof Parts (a) and (b) follow immediately from the fact that the interpretation of $\wedge$ and $\neg$ in the definition of $\models$ is the same as in propositional logic. The proofs of part (c) and (d) are identical to the proofs of parts (a) and (b) of Theorem 2.4.1. ∎

We now show that in a precise sense these properties completely characterize the formulas of $\mathcal{L}_n$ that are valid with respect to $\mathcal{M}_n$. To do so, we have to consider the notion of *provability*. An *axiom system* AX consists of a collection of *axioms* and *inference rules*. An axiom is a formula, and an inference rule has the form "from $\varphi_1, \ldots, \varphi_k$ infer ψ," where $\varphi_1, \ldots, \varphi_k, \psi$ are formulas. We are actually interested in (substitution) instances of axioms and inference rules (so we are really thinking of axioms and inference rules as *schemes*). For example, the formula $K_1q \vee \neg K_1q$ is an instance of the propositional tautology $p \vee \neg p$, obtained by substituting K_1q for p. A *proof* in AX consists of a sequence of formulas, each of which is either an instance of an axiom in AX or follows by an application of an inference rule. (If "from $\varphi_1, \ldots, \varphi_k$ infer ψ" is an instance of an inference rule, and if the formulas $\varphi_1, \ldots, \varphi_k$ have appeared earlier in the proof, then we say that ψ *follows by an application of an inference rule.*) A proof is said to be a *proof of the formula* φ if the last formula in the proof is φ. We say φ is *provable in* AX, and write $AX \vdash \varphi$, if there is a proof of φ in AX.

Consider the following axiom system K_n, which consists of the two axioms and two inference rules given below:

A1. All tautologies of propositional calculus

A2. $(K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$, $i = 1, \ldots, n$ (Distribution Axiom)

R1. From φ and $\varphi \Rightarrow \psi$ infer ψ (Modus ponens)

R2. From φ infer $K_i\varphi$ (Knowledge Generalization)

Recall that we are actually interested in instances of axioms and inference rules. For example,

$$(K_1(p \wedge q) \wedge K_1((p \wedge q) \Rightarrow \neg K_2 r)) \Rightarrow K_1 \neg K_2 r$$

is a substitution instance of the Distribution Axiom.

As a typical example of the use of K_n, consider the following proof of the formula $K_i(p \wedge q) \Rightarrow K_i p$. We give the axiom used or the inference rule applied and the lines it was applied to in parentheses at the end of each step:

1. $(p \wedge q) \Rightarrow p$ (A1)
2. $K_i((p \wedge q) \Rightarrow p)$ (1,R2)
3. $(K_i(p \wedge q) \wedge K_i((p \wedge q) \Rightarrow p)) \Rightarrow K_i p$ (A2)
4. $((K_i(p \wedge q) \wedge K_i((p \wedge q) \Rightarrow p)) \Rightarrow K_i p)$
   $\Rightarrow (K_i((p \wedge q) \Rightarrow p) \Rightarrow (K_i(p \wedge q) \Rightarrow K_i p))$
   (A1, since this is an instance of the propositional tautology
   $((p_1 \wedge p_2) \Rightarrow p_3) \Rightarrow (p_2 \Rightarrow (p_1 \Rightarrow p_3)))$
5. $K_i((p \wedge q) \Rightarrow p) \Rightarrow (K_i(p \wedge q) \Rightarrow K_i p)$ (3,4,R1)
6. $K_i(p \wedge q) \Rightarrow K_i p$ (2,5,R1)

This proof already shows how tedious the proof of even simple formulas can be. Typically we tend to combine several steps when writing up a proof, especially those that involve only propositional reasoning (A1 and R1).

The reader familiar with formal proofs in propositional or first-order logic should be warned that one technique that works in these cases, namely, the use of the *deduction theorem*, does *not* work for K_n. To explain the deduction theorem, we need one more definition. We generalize the notion of provability by defining φ to be *provable from* ψ *in the axiom system* AX, written $AX, \psi \vdash \varphi$, if there is a

sequence of steps ending with φ, each of which is either an instance of an axiom of AX, ψ itself, or follows from previous steps by an application of an inference rule of AX. The deduction theorem is said to hold for AX if $AX, \psi \vdash \varphi$ implies $AX \vdash \psi \Rightarrow \varphi$. Although the deduction theorem holds for the standard axiomatizations of propositional logic and first-order logic, it does not hold for K_n. To see this, observe that for any formula φ, by an easy application of Knowledge Generalization (R2) we have $\mathrm{K}_n, \varphi \vdash K_i\varphi$. However, we do not in general have $\mathrm{K}_n \vdash \varphi \Rightarrow K_i\varphi$: it is certainly not the case in general that if φ is true, then agent i knows φ. It turns out that the Knowledge Generalization Rule is essentially the cause of the failure of the deduction theorem for K_n. This issue is discussed in greater detail in Exercises 3.8 and 3.29.

We return now to our main goal, that of proving that K_n characterizes the set of formulas that are valid with respect to $\mathcal{M}_n$. An axiom system AX is said to be *sound* for a language $\mathcal{L}$ with respect to a class $\mathcal{M}$ of structures if every formula in $\mathcal{L}$ provable in AX is valid with respect to $\mathcal{M}$. The system AX is *complete* for $\mathcal{L}$ with respect to $\mathcal{M}$ if every formula in $\mathcal{L}$ that is valid with respect to $\mathcal{M}$ is provable in AX. We think of AX as characterizing the class $\mathcal{M}$ if it provides a sound and complete axiomatization of that class; notationally, this amounts to saying that for all formulas φ, we have $AX \vdash \varphi$ if and only if $\mathcal{M} \models \varphi$. Soundness and completeness provide a tight connection between the *syntactic* notion of provability and the *semantic* notion of validity.

We plan to show that K_n provides a sound and complete axiomatization for $\mathcal{L}_n$ with respect to $\mathcal{M}_n$. We need one more round of definitions in order to do this. Given an axiom system AX, we say a formula φ is *AX-consistent* if $\neg\varphi$ is not provable in AX. A finite set $\{\varphi_1, \ldots, \varphi_k\}$ of formulas is AX-consistent exactly if $\varphi_1 \wedge \ldots \wedge \varphi_k$ is AX-consistent, and an infinite set of formulas is AX-consistent exactly if all of its finite subsets are AX-consistent. Recall that a language is a set of formulas. A set F of formulas is a *maximal AX-consistent set* with respect to a language $\mathcal{L}$ if (1) it is AX-consistent, and (2) for all φ in $\mathcal{L}$ but not in F, the set $F \cup \{\varphi\}$ is not AX-consistent.

Lemma 3.1.2 *Suppose the language $\mathcal{L}$ consists of a countable set of formulas and is closed with respect to propositional connectives (so that if φ and ψ are in the language, then so are $\varphi \wedge \psi$ and $\neg\varphi$). In any axiom system AX that includes every instance of A1 and R1 for the language $\mathcal{L}$, every AX-consistent set $F \subseteq \mathcal{L}$ can be extended to a maximal AX-consistent set with respect to $\mathcal{L}$. In addition, if F is a maximal AX-consistent set, then it satisfies the following properties:*

(a) for every formula $\varphi \in \mathcal{L}$, exactly one of φ and $\neg\varphi$ is in F,

(b) $\varphi \wedge \psi \in F$ iff $\varphi \in F$ and $\psi \in F$,

(c) if φ and $\varphi \Rightarrow \psi$ are both in F, then ψ is in F,

(d) if φ is provable in AX, then $\varphi \in F$.

Proof Let F be an AX-consistent subset of formulas in $\mathcal{L}$. To show that F can be extended to a maximal AX-consistent set, we first construct a sequence $F_0, F_1, F_2, \ldots$ of AX-consistent sets as follows. Because $\mathcal{L}$ is a countable language, let $\psi_1, \psi_2, \ldots$ be an enumeration of the formulas in $\mathcal{L}$. Let $F_0 = F$, and inductively construct the rest of the sequence by taking $F_{i+1} = F_i \cup \{\psi_{i+1}\}$ if this set is AX-consistent and otherwise by taking $F_{i+1} = F_i$. It is easy to see that each set in the sequence $F_0, F_1, \ldots$ is AX-consistent, and that this is a nondecreasing sequence of sets. Let $F = \cup_{i=0}^{\infty} F_i$. Each finite subset of F must be contained in F_j for some j, and thus must be AX-consistent (since F_j is AX-consistent). It follows that F itself is AX-consistent. We claim that in fact F is a maximal AX-consistent set. For suppose $\psi \in \mathcal{L}$ and $\psi \notin F$. Since ψ is a formula in $\mathcal{L}$, it must appear in our enumeration, say as ψ_k. If $F_k \cup \{\psi_k\}$ were AX-consistent, then our construction would guarantee that $\psi_k \in F_{k+1}$, and hence that $\psi_k \in F$. Because $\psi_k = \psi \notin F$, it follows that $F_k \cup \{\psi\}$ is not AX-consistent. Hence $F \cup \{\psi\}$ is also not AX-consistent. It follows that F is a maximal AX-consistent set.

To see that maximal AX-consistent sets have all the properties we claimed, let F be a maximal AX-consistent set. If $\varphi \in \mathcal{L}$, we now show that one of $F \cup \{\varphi\}$ and $F \cup \{\neg\varphi\}$ is AX-consistent. For assume to the contrary that neither of them is AX-consistent. It is not hard to see that $F \cup \{\varphi \vee \neg\varphi\}$ is then also not AX-consistent (Exercise 3.2). So F is not AX-consistent, because $\varphi \vee \neg\varphi$ is a propositional tautology. This gives a contradiction. If $F \cup \{\varphi\}$ is AX-consistent, then we must have $\varphi \in F$ since F is a maximal AX-consistent set. Similarly, if $F \cup \{\neg\varphi\}$ is AX-consistent then $\neg\varphi \in F$. Thus, one of φ or $\neg\varphi$ is in F. It is clear that we cannot have both φ and $\neg\varphi$ in F, for otherwise F would not be AX-consistent. This proves (a).

Part (a) is enough to let us prove all the other properties we claimed. For example, if $\varphi \wedge \psi \in F$, then we must have $\varphi \in F$, for otherwise, as we just showed, we would have $\neg\varphi \in F$, and F would not be AX-consistent. Similarly, we must have $\psi \in F$. Conversely, if φ and ψ are both in F, we must have $\varphi \wedge \psi \in F$, for otherwise we would have $\neg(\varphi \wedge \psi) \in F$, and, again, F would not be AX-consistent. We leave the proof that F has properties (c) and (d) to the reader (Exercise 3.3). ∎

We can now prove that K_n is sound and complete.

Theorem 3.1.3 K_n *is a sound and complete axiomatization with respect to* $\mathcal{M}_n$ *for formulas in the language* $\mathcal{L}_n$.

Proof Using Theorem 3.1.1, it is straightforward to prove by induction on the length of a proof of φ that if φ is provable in K_n, then φ is valid with respect to $\mathcal{M}_n$ (see Exercise 3.4). It follows that K_n is sound with respect to $\mathcal{M}_n$.

To prove completeness, we must show that every formula in $\mathcal{L}_n$ that is valid with respect to $\mathcal{M}_n$ is provable in K_n. It suffices to prove that

$$\text{Every } \mathrm{K}_n\text{-consistent formula in } \mathcal{L}_n \text{ is satisfiable with respect to } \mathcal{M}_n. \qquad (*)$$

For suppose we can prove $(*)$, and φ is a valid formula in $\mathcal{L}_n$. If φ is not provable in K_n, then neither is $\neg\neg\varphi$, so, by definition, $\neg\varphi$ is K_n-consistent. It follows from $(*)$ that $\neg\varphi$ is satisfiable with respect to $\mathcal{M}_n$, contradicting the validity of φ with respect to $\mathcal{M}_n$.

We prove $(*)$ using a general technique that works for a wide variety of modal logics. We construct a special structure $M^c \in \mathcal{M}_n$, called the *canonical structure for* K_n. M^c has a state s_V corresponding to every maximal K_n-consistent set V. Then we show

$$(M^c, s_V) \models \varphi \text{ iff } \varphi \in V. \qquad (**)$$

That is, we show that a formula is true at a state s_V exactly if it is one of the formulas in V. Note that $(**)$ suffices to prove $(*)$, for by Lemma 3.1.2, if φ is K_n-consistent, then φ is contained in some maximal K_n-consistent set V. From $(**)$ it follows that $(M^c, s_V) \models \varphi$, and so φ is satisfiable in M^c. Therefore, φ is satisfiable with respect to $\mathcal{M}_n$, as desired.

We proceed as follows. Given a set V of formulas, define $V/K_i = \{\varphi \mid K_i\varphi \in V\}$. For example, if $V = \{K_1 p, K_2 K_1 q, K_1 K_3 p \wedge q, K_1 K_3 q\}$, then $V/K_1 = \{p, K_3 q\}$. Let $M^c = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where

$$\begin{aligned} S &= \{s_V \mid V \text{ is a maximal } \mathrm{K}_n\text{-consistent set}\} \\ \pi(s_V)(p) &= \begin{cases} \textbf{true} & \text{if } p \in V \\ \textbf{false} & \text{if } p \notin V \end{cases} \\ \mathcal{K}_i &= \{(s_V, s_W) \mid V/K_i \subseteq W\}. \end{aligned}$$

We now show that for all V we have $(M^c, s_V) \models \varphi$ iff $\varphi \in V$. We proceed by induction on the structure of formulas. More precisely, assuming that the claim holds for all subformulas of φ, we also show that it holds for φ.

If φ is a primitive proposition p, this is immediate from the definition of $\pi(s_V)$. The cases where φ is a conjunction or a negation are simple and left to the reader (Exercise 3.5). Assume that φ is of the form $K_i\psi$ and that $\varphi \in V$. Then $\psi \in V/K_i$ and, by definition of $\mathcal{K}_i$, if $(s_V, s_W) \in \mathcal{K}_i$, then $\psi \in W$. Thus, using the induction hypothesis, $(M^c, s_W) \models \psi$ for all W such that $(s_V, s_W) \in \mathcal{K}_i$. By the definition of $\models$, it follows that $(M^c, s_V) \models K_i\psi$.

For the other direction, assume $(M^c, s_V) \models K_i\psi$. It follows that the set $(V/K_i) \cup \{\neg\psi\}$ is not K_n-consistent. For suppose otherwise. Then, by Lemma 3.1.2, it would have a maximal K_n-consistent extension W and, by construction, we would have $(s_V, s_W) \in \mathcal{K}_i$. By the induction hypothesis we have $(M^c, s_W) \models \neg\psi$, and so $(M^c, s_V) \models \neg K_i\psi$, contradicting our original assumption. Since $(V/K_i) \cup \{\neg\psi\}$ is not K_n-consistent, there must be some finite subset, say $\{\varphi_1, \ldots, \varphi_k, \neg\psi\}$, which is not K_n-consistent. Thus, by propositional reasoning (Exercise 3.6), we have

$$\mathrm{K}_n \vdash \varphi_1 \Rightarrow \big(\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots)\big).$$

By R2, we have

$$\mathrm{K}_n \vdash K_i\big(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots))\big).$$

By induction on k, together with axiom A2 and propositional reasoning, we can show (Exercise 3.7)

$$\mathrm{K}_n \vdash K_i\big(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots))\big) \Rightarrow \big(K_i\varphi_1 \Rightarrow (K_i\varphi_2 \Rightarrow (\ldots \Rightarrow (K_i\varphi_k \Rightarrow K_i\psi)\ldots))\big).$$

Now from R1, we get

$$\mathrm{K}_n \vdash K_i\varphi_1 \Rightarrow \big(K_i\varphi_2 \Rightarrow (\ldots \Rightarrow (K_i\varphi_k \Rightarrow K_i\psi)\ldots)\big).$$

By part (d) of Lemma 3.1.2, it follows that

$$K_i\varphi_1 \Rightarrow \big(K_i\varphi_2 \Rightarrow (\ldots \Rightarrow (K_i\varphi_k \Rightarrow K_i\psi)\ldots)\big) \in V.$$

Because $\varphi_1, \ldots, \varphi_k \in V/K_i$, we must have $K_i\varphi_1, \ldots, K_i\varphi_k \in V$. By part (c) of Lemma 3.1.2, applied repeatedly, it follows that $K_i\psi \in V$, as desired. ∎

We have thus shown that K_n completely characterizes the formulas in $\mathcal{L}_n$ that are valid with respect to $\mathcal{M}_n$, where there are no restrictions on the $\mathcal{K}_i$ relations. What happens if we restrict the $\mathcal{K}_i$ relations? In Chapter 2, we observed that we do get extra properties if we take the $\mathcal{K}_i$ relations to be reflexive, symmetric, and transitive. These properties are the following:

A3. $K_i\varphi \Rightarrow \varphi$, $i = 1, \ldots, n$ (Knowledge Axiom)

A4. $K_i\varphi \Rightarrow K_iK_i\varphi$, $i = 1, \ldots, n$ (Positive Introspection Axiom)

A5. $\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$, $i = 1, \ldots, n$ (Negative Introspection Axiom)

We remarked earlier that axiom A3 has been taken by philosophers to capture the difference between knowledge and belief. From this point of view, the man we spoke of at the beginning of the chapter who "knew" his son was drug-free should really be said to *believe* his son was drug-free, but not to know it. If we want to model such a notion of belief, then (at least according to some philosophers) we ought to drop A3, but add an axiom that says that an agent does not believe *false*:

A6. $\neg K_i(\mathit{false})$, $i = 1, \ldots, n$ (Consistency Axiom)

It is easy to see that A6 is provable from A3, A1, and R1 (see Exercise 3.9).

Historically, axiom A2 has been called **K**, A3 has been called **T**, A4 has been called **4**, A5 has been called **5**, and A6 has been called **D**. We get different modal logics by considering various subsets of these axioms. In the case of one agent, the system with axioms and rules A1, A2, R1, and R2 has been called K. One approach to naming these logics is to name them after the significant axioms used. For example, the axiom system KD45 is the result of combining the axioms **K**, **D**, **4**, and **5** with A1, R1, and R2, while KT4 is the result of combining the axioms **K**, **T**, and **4** with A1, R1, and R2. Some of the axiom systems are commonly called by other names as well. The K is quite often omitted, so that KT becomes T, KD becomes D, and so on; KT4 has traditionally been called S4 and KT45 has been called S5. (The axioms **K**, **T**, **4**, and **5**, together with rule R2, are what we called the S5 properties in Chapter 2.) We stick with the traditional names here for those logics that have them, since they are in common usage, except that we use the subscript n to emphasize the fact that we are considering systems with n agents rather than only one agent. Thus, for example, we speak of the logics T_n or $\mathrm{S5}_n$. We occasionally omit the subscript if $n = 1$, in line with more traditional notation.

Philosophers have spent years arguing which of these axioms, if any, best captures the knowledge of an agent. We do not believe that there is one "true" notion of knowledge; rather, the appropriate notion depends on the application. As we said in Chapter 2, for many of our applications the axioms of S5 seem most appropriate (although philosophers have argued quite vociferously against them, particularly axiom A5). Rather than justify these axioms further, we focus here on the relationship between these axioms and the properties of the $\mathcal{K}_i$ relation, and on the effect of this relationship on the difficulty of reasoning about knowledge. (Some references on the

issue of justification of the axioms can be found in the bibliographic notes at the end of the chapter.) Since we do not have the space to do an exhaustive study of all the logics that can be formed by considering all possible subsets of the axioms, we focus on some representative cases here, namely K_n, T_n, $S4_n$, $S5_n$, and $KD45_n$. These provide a sample of the logics that have been considered in the literature and demonstrate some of the flexibility of this general approach to modeling knowledge. K_n is the minimal system, and it enables us to study what happens when there are in some sense as few restrictions as possible on the K_i operator, given our possible-worlds framework. The minimal extension of K_n that requires that what is known is necessarily true is the system T_n. Researchers who have accepted the arguments against A5 but have otherwise been happy with the axioms of $S5_n$ have tended to focus on $S4_n$. On the other hand, researchers who were willing to accept the introspective properties embodied by A4 and A5, but wanted to consider belief rather than knowledge have tended to consider KD45 or K45. For definiteness, we focus on KD45 here, but all our results for KD45 carry over with very little change to K45.

Theorem 3.1.3 implies that the formulas provable in K_n are precisely those that are valid with respect to $\mathcal{M}_n$. We want to connect the remaining axioms with various restrictions on the possibility relations $\mathcal{K}_i$. We have already considered one possible restriction on the $\mathcal{K}_i$ relations (namely, that they be reflexive, symmetric, and transitive). We now consider others. We say that a binary relation $\mathcal{K}$ on a set S is *Euclidean* if, for all $s, t, u \in S$, whenever $(s, t) \in \mathcal{K}$ and $(s, u) \in \mathcal{K}$, then $(t, u) \in \mathcal{K}$; we say that $\mathcal{K}$ is *serial* if, for all $s \in S$, there is some t such that $(s, t) \in \mathcal{K}$.

Some of the relationships between various conditions we can place on binary relations are captured in the following lemma, whose proof is left to the reader (Exercise 3.12).

Lemma 3.1.4

(a) If $\mathcal{K}$ is reflexive and Euclidean, then $\mathcal{K}$ is symmetric and transitive.

(b) If $\mathcal{K}$ is symmetric and transitive, then $\mathcal{K}$ is Euclidean.

(c) The following are equivalent:

(i) $\mathcal{K}$ is reflexive, symmetric, and transitive.

(ii) $\mathcal{K}$ is symmetric, transitive, and serial.

(iii) $\mathcal{K}$ is reflexive and Euclidean.

Let $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$; $\mathcal{M}_n^{rst}$; $\mathcal{M}_n^{elt}$) be the class of all structures for n agents where the possibility relations are reflexive (resp., reflexive and transitive; reflexive, symmetric, and transitive; Euclidean, serial, and transitive). As we observed earlier, since an equivalence relation is one that is reflexive, symmetric, and transitive, $\mathcal{M}_n^{rst}$ is precisely the class of structures we considered in Chapter 2.

The next theorem shows a close connection between various combinations of axioms, on the one hand, and various restrictions on the possibility relations $\mathcal{K}_i$, on the other hand. For example, axiom A3 (the Knowledge Axiom $K_i\varphi \Rightarrow \varphi$) corresponds to reflexivity of $\mathcal{K}_i$. To demonstrate one part of this correspondence, we now show that axiom A3 is valid in all structures in $\mathcal{M}_n^r$. If s is a world in a structure $M \in \mathcal{M}_n^r$, then agent i must consider s to be one of his possible worlds in s. Thus, if agent i knows φ in s, then φ must be true in s; i.e., $(M, s) \models K_i\varphi \Rightarrow \varphi$. Therefore, T_n is sound with respect to $\mathcal{M}_n^r$. We might hope that, conversely, every structure that satisfies all instances of axiom A3 is in $\mathcal{M}_n^r$. Unfortunately, this is not the case (we return to this point a little later). Nevertheless, as we shall see in the proof of the next theorem, axiom A3 forces the possibility relations in the canonical structure to be reflexive. As we shall see, this is sufficient to prove that T_n is complete with respect to $\mathcal{M}_n^r$.

Theorem 3.1.5 *For formulas in the language $\mathcal{L}_n$:*

(a) T_n *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^r$,

(b) $\mathrm{S4}_n$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rt}$,

(c) $\mathrm{S5}_n$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rst}$,

(d) $\mathrm{KD45}_n$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{elt}$.

Proof We first consider part (a). We already showed that T_n is sound with respect to $\mathcal{M}_n^r$. For completeness, we need to show that every T_n-consistent formula is satisfiable in some structure in $\mathcal{M}_n^r$. This is done exactly as in the proof of Theorem 3.1.3. We define a canonical structure M^c for T_n each of whose states corresponds to a maximal T_n-consistent set V of formulas. The $\mathcal{K}_i$ relations are defined as in the proof of Theorem 3.1.3, namely, $(s_V, s_W) \in \mathcal{K}_i$ in M^c exactly if $V/K_i \subseteq W$, where $V/K_i = \{\varphi \mid K_i\varphi \in V\}$. A proof identical to that of Theorem 3.1.3 can now be used to show that $\varphi \in V$ iff $(M^c, s_V) \models \varphi$, for all maximal T_n-consistent sets V. Furthermore, it is easy to see that every maximal T_n-consistent set V contains every instance of axiom A3. Therefore, all instances of axiom A3 are true at s_V. It follows immediately that $V/K_i \subseteq V$. So by definition of $\mathcal{K}_i$, it follows that $(s_V, s_V) \in \mathcal{K}_i$. So $\mathcal{K}_i$

is indeed reflexive, and hence $M^c \in \mathcal{M}_n^r$. Assume now that φ is a T_n-consistent formula. As in the proof of Theorem 3.1.3, it follows that φ is satisfiable in M^c. Since, as we just showed, $M^c \in \mathcal{M}_n^r$, it follows that φ is satisfiable in some structure in $\mathcal{M}_n^r$, as desired. This completes the proof of part (a).

To prove part (b), we show that just as axiom A3 corresponds to reflexivity, similarly axiom A4 corresponds to transitivity. It is easy to see that A4 is valid in all structures where the possibility relation is transitive. Moreover, A4 forces the possibility relations in the canonical structure to be transitive. To see this, suppose that $(s_V, s_W), (s_W, s_X) \in \mathcal{K}_i$ and that all instances of A4 are true at s_V. Then if $K_i\varphi \in V$, by A4 we have $K_iK_i\varphi \in V$, and, by the construction of M^c, we have $K_i\varphi \in W$ and $\varphi \in X$. Thus, $V/K_i \subseteq X$ and $(s_V, s_X) \in \mathcal{K}_i$, as desired. The proof is now very similar to that of part (a).

The proof of parts (c) and (d) go in the same way. Here the key correspondences are that axiom A5 corresponds to a Euclidean possibility relation and axiom A6 corresponds to a serial relation (Exercise 3.13). ∎

We say that a structure M is a *model of* K_n if every formula provable in K_n is valid in M. We can similarly say that a structure is a model of T_n, $S4_n$, $S5_n$, and $KD45_n$. The soundness part of Theorem 3.1.5 shows that every structure in $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$) is a model of T_n (resp., $S4_n$, $S5_n$, $KD45_n$). We might be tempted to conjecture that the converse also holds, so that, for example, if a structure is a model of $S5_n$, then it is in $\mathcal{M}_n^{rst}$. This is not quite true, as the following example shows. Suppose $n = 1$ and $\Phi = \{p\}$, and let M be the structure consisting of two states s and t, such that $\pi(s)(p) = \pi(t)(p) = \mathbf{true}$ and $\mathcal{K}_1 = \{(s, t), (t, t)\}$, as shown in Figure 3.1.

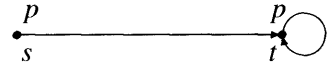

Figure 3.1 A model of $S5_1$ that is not in $\mathcal{M}_1^{rst}$

The structure M is not in $\mathcal{M}_1^r$, let alone $\mathcal{M}_1^{rst}$, but it is easy to see that it is a model of $S5_1$ and *a fortiori* a model of $S4_1$ and T_1 (Exercise 3.15). Nevertheless, the intuition behind the conjecture is almost correct. In fact, it is correct in two senses. If s is a state in a Kripke structure M, and s' is a state in a Kripke structure M', then we say that (M, s) and (M', s') are *equivalent*, and write $(M, s) \equiv (M', s')$, if they satisfy exactly the same formulas in the language $\mathcal{L}_n$. That is, $(M, s) \equiv (M', s')$ if, for all formulas $\varphi \in \mathcal{L}_n$, we have $(M, s) \models \varphi$ if and only if $(M', s') \models \varphi$. One sense

in which the previous conjecture is correct is that every model M of T_n (resp., $S4_n$, $S5_n$, $KD45_n$) can effectively be converted to a structure M' in $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$) with the same state space, such that $(M, s) \equiv (M', s)$ for every state s (see Exercise 3.16).

The second sense in which the conjecture is correct involves the notion of a *frame*. We define a *frame for n agents* to be a tuple $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where S is a set of states and $\mathcal{K}_1, \ldots, \mathcal{K}_n$ are binary relations on S. Thus, a frame is like a Kripke structure without the function π. Notice that the Aumann structures defined in Section 2.5 can be viewed as frames. We say that the Kripke structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is *based on* the frame $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n)$. A formula φ is *valid* in frame F if it is valid in every Kripke structure based on F. It turns out that if we look at the level of frames rather than at the level of structures, then we get what can be viewed as a partial converse to Theorem 3.1.5. For example, the $\mathcal{K}_i$'s in a frame F are reflexive *if and only if* every instance of the Knowledge Axiom A3 is valid in F. This suggests that the axioms are tied more closely to frames than they are to structures. Although we have shown that, for example, we can find a structure that is a model of $S5_n$ but is not in $\mathcal{M}_n^{rst}$ (or even $\mathcal{M}_n^r$), this is not the case at the level of frames. If a frame is a model of $S5_n$, then it must be in $\mathcal{F}_n^{rst}$. Conversely, if a frame is in $\mathcal{F}_n^{rst}$, then it is a model of $S5_n$. See Exercise 3.17 for more details.

The previous results show the connection between various restrictions on the $\mathcal{K}_i$ relations and properties of knowledge. In particular, we have shown that A3 corresponds to reflexive possibility relations, A4 to transitive possibility relations, A5 to Euclidean possibility relations, and A6 to serial possibility relations.

Up to now we have not considered symmetric relations. It is not hard to check (using arguments similar to those used previously) that symmetry of the possibility relations corresponds to the following axiom:

A7. $\varphi \Rightarrow K_i \neg K_i \neg \varphi, \quad i = 1, \ldots, n$

Axiom A7 can also easily be shown to be a consequence of A3 and A5, together with propositional reasoning (Exercise 3.18). This corresponds to the observation made in Lemma 3.1.4 that a reflexive Euclidean relation is also symmetric. Since a reflexive Euclidean relation is also transitive, the reader may suspect that A4 is redundant in the presence of A3 and A5. This is essentially true. It can be shown that A4 is a consequence of A1, A2, A3, A5, R1, and R2 (see Exercise 3.19). Thus we can obtain an axiom system equivalent to $S5_n$ by eliminating A4; indeed, by using the observations of Lemma 3.1.4, we can obtain a number of axiomatizations that are equivalent to $S5_n$ (Exercise 3.20).

Axiom	Property of $\mathcal{K}_i$
A3. $K_i\varphi \Rightarrow \varphi$	reflexive
A4. $K_i\varphi \Rightarrow K_iK_i\varphi$	transitive
A5. $\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$	Euclidean
A6. $\neg K_i false$	serial
A7. $\varphi \Rightarrow K_i\neg K_i\neg\varphi$	symmetric

Table 3.1 The correspondence between axioms and properties of $\mathcal{K}_i$

The preceding discussion is summarized by Table 3.1, which describes the correspondence between the axioms and the properties of the $\mathcal{K}_i$ relations.

We conclude this section by taking a closer look at the single-agent case of S5 and KD45. The following result shows that in the case of S5 we can further restrict our attention to structures where the possibility relation is *universal*; i.e., in every state, all states are considered possible. Intuitively, this means that in the case of S5 we can talk about *the* set of worlds the agent considers possible; this set is the same in every state and consists of all the worlds. Similarly, for KD45 we can restrict attention to structures with one distinguished state, which intuitively is the "real" world, and a set of states (which does not in general include the real world) corresponding to the worlds that the agent thinks possible in every state.

Proposition 3.1.6

(a) Assume that $M \in \mathcal{M}_1^{rst}$ and s is a state of M. Then there is a structure $M' = (S', \pi', \mathcal{K}_1')$, where $\mathcal{K}_1'$ is universal, i.e., $\mathcal{K}_1' = \{(s,t) \mid s, t \in S'\}$, and a state s' of M' such that $(M, s) \equiv (M', s')$.

(b) Assume that $M \in \mathcal{M}_1^{elt}$ and s is a state of M. Then there is a structure $M' = (\{s_0\} \cup S', \pi', \mathcal{K}_1')$, where S' is nonempty and $\mathcal{K}_1' = \{(s,t) : s \in \{s_0\} \cup S'$ and $t \in S'\}$, and a state s' of M' such that $(M, s) \equiv (M', s')$.

Proof We first consider part (b). Assume that $M = (S, \pi, \mathcal{K}_1) \in \mathcal{M}_1^{elt}$ and that $s_0 \in S$. Let $\mathcal{K}_1(s_0) = \{t \mid (s_0, t) \in \mathcal{K}_1\}$. Since $\mathcal{K}_1$ is serial, $\mathcal{K}_1(s_0)$ must be nonempty. It is also easy to check that since $\mathcal{K}_1$ is Euclidean, we have $(s,t) \in \mathcal{K}_1$ for all $s, t \in \mathcal{K}_1(s_0)$. Finally, since $\mathcal{K}_1$ is transitive, if $s \in \mathcal{K}_1(s_0)$ and $(s,t) \in \mathcal{K}_1$, then $t \in \mathcal{K}_1(s_0)$. Let $M' = (\{s_0\} \cup \mathcal{K}_1(s_0), \pi', \mathcal{K}_1')$, where π' is the restriction of π to $\{s_0\} \cup \mathcal{K}_1(s_0)$, and $\mathcal{K}_1' = \{(s,t) \mid s \in \{s_0\} \cup \mathcal{K}_1(s_0)$ and $t \in \mathcal{K}_1(s_0)\}$. By the previous observations, $\mathcal{K}_1'$ is the restriction of $\mathcal{K}_1$ to $\{s_0\} \cup \mathcal{K}_1(s_0)$. Note that $\mathcal{K}_1'$ is

serial (because $\mathcal{K}_1(s_0)$ is nonempty), Euclidean, and transitive. A straightforward induction on the structure of formulas now shows that for all $s \in \{s_0\} \cup \mathcal{K}_1(s_0)$ and all formulas $\varphi \in \mathcal{L}_n$, we have $(M, s) \models \varphi$ iff $(M', s) \models \varphi$. We leave details to the reader (Exercise 3.21).

For part (a), we proceed in the same way, except that we start with a structure $M \in \mathcal{M}_1^{rst}$. Using the fact that $\mathcal{K}_1$ is now reflexive, it is easy to show that the relation $\mathcal{K}_1'$ we construct is universal. The rest of the proof proceeds as before. ∎

It follows from Proposition 3.1.6 that we can assume without loss of generality that models of S5 have a particularly simple form, namely (S, π), where we do not mention the $\mathcal{K}_1$ relation but simply assume that $(s, t) \in \mathcal{K}_1$ for all $s, t \in S$. Similarly, we can take models of KD45 to have the form (s_0, S, π), where, as already discussed, the intuition is that s_0 is the "real" world, and S is the set of worlds that the agent considers possible. As we shall see, this simple representation of models for S5 and KD45 has important implications when it comes to the difficulty of deciding whether a formula is provable in S5 or KD45.

There is a similar simple representation for models of K45 (Exercise 3.22). We cannot in general get such simple representations for the other logics we have considered, nor can we get them even for $S5_n$ or $KD45_n$ if $n > 1$, i.e., if we have two or more agents in the picture. For more information on the single-agent case of S5, see Exercise 3.23.

3.2 Decidability

In the preceding section we showed that the set of valid formulas of $\mathcal{M}_n$ is indeed characterized by K_n, and that the valid formulas of various interesting subclasses of $\mathcal{M}_n$ are characterized by other systems, such as T_n, $S4_n$, and $S5_n$. Our results, however, were not constructive; they gave no indication of how to tell whether a given formula was indeed provable (and thus also valid in the appropriate class of structures).

In this section, we present results showing that the question of whether a formula is valid is *decidable*; that is, there is an algorithm that, given as input a formula φ, will decide whether φ is valid. (It is beyond the scope of this book to give a formal definition of decidability; references are provided in the notes at the end of the chapter.) An algorithm that recognizes valid formulas can be viewed as another characterization of the properties of knowledge, one that is complementary to the characterization in terms of a sound and complete axiom system.

A situation in which recognizing valid formulas is useful is where we have an agent whose information is characterized by a collection of formulas whose conjunction is φ. If the agent wants to know whether a formula ψ follows from the information he has, then he has to check whether $\varphi \Rightarrow \psi$ is valid. An example of this type of situation is if we take the agent to be a knowledge base. A knowledge base can draw conclusions about the state of the world based on the logical consequences of the information it has been told. (See Sections 4.4.1, 7.3, and 9.3.3 for further discussion of knowledge bases.)

A formula φ is valid in a certain class of Kripke structures if it is true in all states in all structures of that class. Thus, before examining the algorithmic aspects of validity, we consider the algorithmic aspects of truth. We refer to the problem of deciding if a formula is true in a given state of a given Kripke structure as the *model-checking problem.*

There is no general procedure for doing model checking in an infinite Kripke structure. Indeed, it is not even possible to represent arbitrary infinite structures effectively. On the other hand, in finite Kripke structures, model checking is relatively straightforward. Given a finite Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, define $||M||$, the *size* of M, to be the sum of the number of states in S and the number of pairs in $\mathcal{K}_i$, for $i = 1, \ldots, n$. In the following proposition (and in later results), we use the notation $O(f)$, read "*order* of f" or "(big) O of f" for a function f. This denotes some function g such that for each argument a, we have $g(a) \leq cf(a)$ for some constant c independent of a. Thus, for example, when we say that the running time of an algorithm is $O(||M|| \times |\varphi|)$, this means that there is some constant c, independent of the structure M and the formula φ, such that for all structures M and formulas φ the time to check if φ is satisfied in M is at most $c \times ||M|| \times |\varphi|$.

Proposition 3.2.1 *There is an algorithm that, given a structure M, a state s of M, and a formula $\varphi \in \mathcal{L}_n$, determines, in time $O(||M|| \times |\varphi|)$, whether $(M, s) \models \varphi$.*

Proof Let $\varphi_1, \ldots, \varphi_m$ be the subformulas of φ, listed in order of length, with ties broken arbitrarily. Thus we have $\varphi_m = \varphi$, and if φ_i is a subformula of φ_j, then $i < j$. There are at most $|\varphi|$ subformulas of φ (Exercise 3.1), so we must have $m \leq |\varphi|$. An easy induction on k shows that we can label each state s in M with φ_j or $\neg\varphi_j$, for $j = 1, \ldots, k$, depending on whether or not φ_j is true at s, in time $O(k||M||)$. The only nontrivial case is if φ_{k+1} is of the form $K_i\varphi_j$, where $j < k + 1$. We label a state s with $K_i\varphi_j$ iff each state t such that $(s, t) \in \mathcal{K}_i$ is labeled with φ_j. Assuming inductively that each state has already been labeled with φ_j or $\neg\varphi_j$, this step can clearly be carried out in time $O(||M||)$, as desired. ∎

Observe that this result holds independent of the number of agents. It continues to hold if we restrict attention to particular classes of structures, such as $\mathcal{M}_n^{rt}$ or $\mathcal{M}_n^{rst}$. The result can be easily extended to get a polynomial-time model-checking algorithm even if we have distributed knowledge or common knowledge in the language (Exercise 3.24). Finally, note that the algorithm can be easily modified to check whether φ holds at a particular state s in M.

It should be noted that Proposition 3.2.1 implicitly assumes a "reasonable" representation for Kripke structures. In particular, it assumes that, given a state s and a primitive proposition p, we can determine in constant time whether $\pi(s)$ assigns to p the truth value **true** or the truth value **false**. Such an assumption is not always appropriate. If s corresponds to a position in a chess game and p corresponds to "white can win from this position," then $\pi(s)(p)$ may be quite difficult to compute. Similarly, Proposition 3.2.1 requires some assumption on the time to "traverse" the edges of the Kripke structure; for example, it is sufficient to assume that given a state s where there are m edges $(s, t) \in \mathcal{K}_i$, we can find in time $O(m)$ all the states t such that $(s, t) \in \mathcal{K}_i$. These assumptions are fairly natural if we think of Kripke structures as labeled graphs, and we can read off the $\mathcal{K}_i$ relations and the states where the primitive propositions are true from the diagram describing the graph. Whenever we use a Kripke structure to model a specific situation, however, then we must reexamine these assumptions. In the case of the Kripke structure for the muddy children puzzle described in Chapter 2, it is easy to tell if a proposition p_i is true at a given state, and it is easy to compute the $\mathcal{K}_i$ relations from the descriptions of the states; in general, it may not be. We return to this issue in Chapter 10.

We now turn our attention to the problem of checking whether a given formula is provable. We start with K_n. Our first step is to show that if a formula is K_n-consistent, not only is it satisfiable in some structure (in particular, the canonical structure constructed in the proof of Theorem 3.1.3), but in fact it is also satisfiable in a finite structure (which the canonical structure is certainly not!). The proof is actually just a slight variant of the proof of Theorem 3.1.3. The idea is that rather than considering maximal K_n-consistent subsets of $\mathcal{L}_n$ when trying to construct a structure satisfying a formula φ, we restrict attention to sets of subformulas of φ.

Theorem 3.2.2 *If $\varphi \in \mathcal{L}_n$ is K_n-consistent, then φ is satisfiable in an $\mathcal{M}_n$ structure with at most $2^{|\varphi|}$ states.*

Proof Let $Sub^+(\varphi)$ consist of all the subformulas of φ and their negations, i.e., $Sub^+(\varphi) = Sub(\varphi) \cup \{\neg\psi \mid \psi \in Sub(\varphi)\}$. Let $Con(\varphi)$ be the set of maximal K_n-consistent subsets of $Sub^+(\varphi)$. A proof almost identical to that of Lemma 3.1.2

can be used to show that every K_n-consistent subset of $Sub^+(\varphi)$ can be extended to an element of $Con(\varphi)$. Moreover, a member of $Con(\varphi)$ contains either ψ or $\neg\psi$ for every formula $\psi \in Sub(\varphi)$ (but not both, for otherwise it would not be K_n-consistent). So the cardinality of $Con(\varphi)$ is at most $2^{|Sub(\varphi)|}$, which is at most $2^{|\varphi|}$, since $|Sub(\varphi)| \leq |\varphi|$.

We now construct a structure $M_\varphi = (S_\varphi, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$. The construction is essentially that of Theorem 3.1.3, except that we take $S_\varphi = \{s_V \mid V \in Con(\varphi)\}$. We can now show that if $V \in Con(\varphi)$, then for all $\psi \in Sub^+(\varphi)$ we have $(M_\varphi, s_V) \models \psi$ iff $\psi \in V$. The proof is identical to that of Theorem 3.1.3, and so is omitted here. ∎

From Theorem 3.2.2, we can get an effective (although not particularly efficient) procedure for checking if a formula φ is K_n-consistent (or equivalently, by Theorem 3.1.3, satisfiable with respect to $\mathcal{M}_n$). We simply construct every Kripke structure with $2^{|\varphi|}$ states. (The number of such structures is finite, albeit very large.) We then check if φ is true at some state of one of these structures. The latter check is done using the model-checking algorithm of Proposition 3.2.1. If φ is true at some state in one of these structures, then clearly φ is satisfiable with respect to $\mathcal{M}_n$. Conversely, if φ is satisfiable with respect to $\mathcal{M}_n$, then by Theorem 3.2.2 it must be satisfiable in one of these structures.

As a consequence, we can now show that the validity problem for $\mathcal{M}_n$ (or equivalently, by Theorem 3.1.3, the provability problem for K_n) is decidable.

Corollary 3.2.3 *The validity problem for $\mathcal{M}_n$ and the provability problem for K_n are decidable.*

Proof Since φ is provable in K_n iff $\neg\varphi$ is not K_n-consistent, we can simply check (using the aforementioned procedure) if $\neg\varphi$ is K_n-consistent. ∎

Note that by Corollary 3.2.3 we have a way of checking whether a formula is provable in K_n without deriving a single proof using the axiom system! (Actually, with some additional effort we can extend the ideas in the proof of Theorem 3.2.2 so that if a formula is provable in K_n, then we can effectively find a proof of it; see Exercise 3.25 for details.)

We can extend the arguments of Theorem 3.2.2 to the other logics we have been considering.

Theorem 3.2.4 *If φ is T_n- (resp., $S4_n$-, $S5_n$-, $KD45_n$-) consistent, then φ is satisfiable in a structure in $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$) with at most $2^{|\varphi|}$ states.*

Proof The proof in the case of T_n is identical to that of Theorem 3.2.2, except that we consider maximal T_n-consistent subsets of $Sub^+(\varphi)$ rather than maximal

K_n-consistent subsets of $Sub^+(\varphi)$. Note that in the case of T_n, the axiom $K_i\varphi \Rightarrow \varphi$ guarantees that $V/K_i \subseteq V$, so we get reflexivity of $\mathcal{K}_i$ even if we restrict attention to subsets of $Sub^+(\varphi)$.

The obvious modification of the proof of Theorem 3.2.2 does not work for $S4_n$, since the $\mathcal{K}_i$ relations may not be transitive if we define $(s_V, s_W) \in \mathcal{K}_i$ iff $V/K_i \subseteq W$. For example, if φ is the formula $K_1 p$, then the maximal $S4_n$-consistent subsets of $Sub^+(\varphi)$ are $V_1 = \{K_1 p, p\}$, $V_2 = \{\neg K_1 p, p\}$, and $V_3 = \{\neg K_1 p, \neg p\}$. Note that $V_1/K_1 \subseteq V_2$ and $V_2/K_1 \subseteq V_3$, but $V_1/K_1 \not\subseteq V_3$. Although $V_1/K_1 \subseteq V_2$, intuitively it should be clear that we do not want to have $(s_{V_1}, s_{V_2}) \in \mathcal{K}_1$. The reason is that every maximal $S4_n$-consistent extension of V_1 contains $K_1 K_1 p$; in such an extension, no $S4_n$-consistent extension of V_2 would be considered possible.

In the case of $S4_n$, we deal with this problem as follows: We repeat the construction of Theorem 3.2.2, except that we take $\mathcal{K}_i$ to be $\{(s_V, s_W) \mid V/K_i \subseteq W/K_i\}$. Clearly this definition guarantees that $\mathcal{K}_i$ is transitive. For $S5_n$, we take $\mathcal{K}_i$ to consist of $\{(s_V, s_W) \mid V/K_i = W/K_i\}$. This guarantees that $\mathcal{K}_i$ is an equivalence relation. In the case of $S4_n$ and $S5_n$, the axiom $K_i\varphi \Rightarrow \varphi$ guarantees that if $V/K_i \subseteq W/K_i$, then $V/K_i \subseteq W$, which we make use of in the proof. For $KD45_n$ we do not have this axiom, so we take $\mathcal{K}_i$ to consist of $\{(s_V, s_W) \mid V/K_i = W/K_i,\ V/K_i \subseteq W\}$. We leave it to the reader to check that with this definition, $\mathcal{K}_i$ is Euclidean, transitive, and serial. The proof in all cases now continues along the lines of Theorem 3.1.3; we leave details to the reader (Exercise 3.26). ∎

Just as in the case of K_n, we can use this result to give us an effective technique for deciding whether a formula is provable in T_n (resp., $S4_n$, $S5_n$, $KD45_n$).

Corollary 3.2.5 *The validity problem for $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$) and the provability problem for T_n (resp., $S4_n$, $S5_n$, $KD45_n$) are decidable.*

It turns out that in fact there are more efficient ways of checking whether a formula is provable than those suggested by the results we have just proved; we discuss this issue later in the chapter.

3.3 Incorporating Common Knowledge

We now turn our attention to axiomatizing the operators E_G and C_G. The operator C_G is "infinitary" in that it is defined as an infinite conjunction. This might suggest that we will not be able to characterize it with a finite set of axioms. Somewhat surprisingly, this turns out to be false. The axioms for common knowledge provided in Chapter 2

are complete, as we now show. This suggests that the characterization of common knowledge as a fixed point is somehow more fundamental than its characterization as an infinite conjunction. We return to this point in Chapter 11.

Let K_n^C (resp., T_n^C, $S4_n^C$, $S5_n^C$, $KD45_n^C$) consist of all the axioms of K_n (resp., T_n, $S4_n$, $S5_n$, $KD45_n$) together with the following two axioms and inference rule taken from Chapter 2:

C1. $E_G\varphi \Leftrightarrow \bigwedge_{i\in G} K_i\varphi$

C2. $C_G\varphi \Rightarrow E_G(\varphi \wedge C_G\varphi)$

RC1. From $\varphi \Rightarrow E_G(\psi \wedge \varphi)$ infer $\varphi \Rightarrow C_G\psi$ (Induction Rule)

As the following result shows, C1, C2, and RC1 completely characterize common knowledge.

Theorem 3.3.1 *For formulas in the language $\mathcal{L}_n^C$:*

(a) K_n^C *is a sound and complete axiomatization with respect to* $\mathcal{M}_n$,

(b) T_n^C *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^r$,

(c) $S4_n^C$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rt}$,

(d) $S5_n^C$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rst}$,

(e) $KD45_n^C$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{elt}$.

Proof We consider the case of K_n^C here. We can get all the other cases by modifying the proof just as we modified the proof of Theorem 3.1.3 to prove Theorem 3.1.5.

The validity of axioms C1 and C2 with respect to $\mathcal{M}_n^{rst}$, and the fact that RC1 preserves valid formulas with respect to $\mathcal{M}_n^{rst}$, was shown in Theorem 2.4.2. Although that proof was done in the context of $\mathcal{M}_n^{rst}$, as we remarked in the proof, the proof did not make use of the fact that the possibility relations were reflexive, symmetric, and transitive, and therefore it goes through without change even if we drop this assumption. So soundness follows.

For completeness, we proceed as in the proof of Theorem 3.1.3 to show that if φ is K_n^C-consistent, then φ is satisfiable in some structure in $\mathcal{M}_n$. For technical reasons that are explained below, we need to restrict to finite structures as is done in Theorem 3.2.2.

We define $Sub_C(\varphi)$ to consist of all subformulas of φ together with the formulas $E_G(\psi \wedge C_G\psi)$, $\psi \wedge C_G\psi$, and $K_i(\psi \wedge C_G\psi)$ for each subformula $C_G\psi$ of φ and

$i \in G$, and the formulas $K_i\psi$ for each subformula $E_G\psi$ of φ and $i \in G$. We define $Sub_C^+(\varphi)$ to consist of all the formulas in $Sub_C(\varphi)$ and their negations, and define $Con_C(\varphi)$ to consist of all maximal K_n^C-consistent subsets of $Sub_C^+(\varphi)$. Let $M_\varphi = (S_\varphi, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where S_φ consists of $\{s_V \mid V \in Con_C(\varphi)\}$, $\pi(s_V)(p) = \textbf{true}$ iff $p \in V$, and $\mathcal{K}_i = \{(s_V, s_W) \mid V/K_i \subseteq W\}$, $i = 1, \ldots n$. Clearly S_φ is finite; in fact, it is not hard to show that $|S_\varphi| \leq 2^{|\varphi|}$ (see Exercise 3.27).

We again want to show that for every formula $\varphi' \in Sub_C^+(\varphi)$, we have $(M_\varphi, s_V) \models \varphi'$ iff $\varphi' \in V$. We proceed by induction on the structure of formulas. The argument in the case that φ' is a primitive proposition, a conjunction, a negation, or of the form $K_i\psi$ is essentially identical to that used in Theorem 3.1.3; we do not repeat it here.

Suppose φ' is of the form $E_G\psi$. Assume that $\varphi' \in V$. Since V is a maximal K_n^C-consistent subset of $Sub_C^+(\varphi)$, and since $Sub_C^+(\varphi)$ includes (by definition) all formulas $K_i\psi$ for $i \in G$, by C1 we get that $K_i\psi \in V$ for all $i \in G$. So by the induction hypothesis, $(M_\varphi, s_V) \models K_i\psi$ for each $i \in G$. Therefore, $(M_\varphi, s_V) \models E_G\psi$. We have shown that if $E_G\psi \in V$, then $(M_\varphi, s_V) \models E_G\psi$. The proof of the converse is very similar.

Finally, we must consider the case that φ' is of the form $C_G\psi$. That is, we need to prove that $C_G\psi \in V$ iff $(M_\varphi, s_V) \models C_G\psi$. We begin with the "only if" direction. If $C_G\psi \in V$, we show by induction on k that if s_W is G-reachable from s_V in k steps, then both ψ and $C_G\psi$ are in W. For $k = 1$, observe that axiom C2 and the fact that $V \in Con_C(\varphi)$ together imply that $E_G(\psi \wedge C_G\psi) \in V$. Now our construction guarantees that if s_W is G-reachable from s_V in one step (so that $(s_V, s_W) \in \mathcal{K}_i$ for some $i \in G$), we have $(\psi \wedge C_G\psi) \in W$. Since $W \in Con_C(\varphi)$, it follows that both ψ and $C_G\psi$ are in W. Now assume that the claim holds for k; we prove it for $k + 1$. If s_W is G-reachable from s_V in k steps, then there exists W' such that $s_{W'}$ is G-reachable from s_V in k steps, and s_W is reachable from $s_{W'}$ in one step. By the induction hypothesis, both ψ and $C_G\psi$ are in W'. The argument for the base case now shows that both $C_G\psi$ and ψ are in W. Our argument shows that, in particular, $\psi \in W$ for all W such that s_W is G-reachable from s_V. By our main induction hypothesis, $(M_\varphi, s_W) \models \psi$ for all s_W that are G-reachable from s_V. Thus, $(M_\varphi, s_V) \models C_G\psi$.

The proof of the converse, that if $(M_\varphi, s_V) \models C_G\psi$ then $C_G\psi \in V$, is the hardest part of the proof. Assume that $(M_\varphi, s_V) \models C_G\psi$. We can describe each state s_W of M_φ by the conjunction of the formulas in W. This conjunction, which we denote by φ_W, is a formula in $\mathcal{L}_n^C$, since W is a finite set. Here we make crucial use of the fact that we restrict to formulas in $Sub_C^+(\varphi)$, a finite set, rather than consider maximal K_n^C-consistent subsets of $\mathcal{L}_n^C$, which would have been the straightforward analogue to the proof of Theorem 3.1.3. Let $\mathcal{W} = \{W \in Con_C(\varphi) \mid (M_\varphi, s_W) \models C_G\psi\}$. Define

$\varphi_{\mathcal{W}}$ to be $\bigvee_{W \in \mathcal{W}} \varphi_W$. Thus, $\varphi_{\mathcal{W}}$ is the disjunction of the description of all of the states s_W where $C_G\psi$ holds, and can be thought of as the formula that characterizes these states. Since the set $\mathcal{W}$ is finite, it follows that $\varphi_{\mathcal{W}}$ is a formula in $\mathcal{L}_n^C$. The key step in the proof is to make use of the Induction Rule (RC1), where $\varphi_{\mathcal{W}}$ plays the role of φ. In Exercise 3.28, we prove that

$$\mathrm{K}_n^C \vdash \varphi_{\mathcal{W}} \Rightarrow E_G(\psi \wedge \varphi_{\mathcal{W}}). \tag{3.1}$$

Therefore, by the Induction Rule, we obtain

$$\mathrm{K}_n^C \vdash \varphi_{\mathcal{W}} \Rightarrow C_G\psi.$$

Since $V \in \mathcal{W}$, we have $\mathrm{K}_n^C \vdash \varphi_V \Rightarrow \varphi_{\mathcal{W}}$, so

$$\mathrm{K}_n^C \vdash \varphi_V \Rightarrow C_G\psi. \tag{3.2}$$

It follows that $C_G\psi \in V$, as desired. For if $C_G\psi \notin V$, then $\neg C_G\psi \in V$, and it is easy to see that this, together with (3.2), would imply that V is not K_n^C-consistent, a contradiction. ∎

Notice that our proof shows that if a formula φ is satisfiable at all, it is satisfiable in a finite structure (in fact, with at most $2^{|\varphi|}$ states.) Thus, using the techniques of the previous section, we again get that the validity problem for K_n^C (resp., T_n^C, $\mathrm{S4}_n^C$, $\mathrm{S5}_n^C$, $\mathrm{KD45}_n^C$) is decidable.

3.4 Incorporating Distributed Knowledge

The last operator we would like to axiomatize is D_G. The major new properties of distributed knowledge were discussed in Chapter 2:

D1. $D_{\{i\}}\varphi \Leftrightarrow K_i\varphi, \quad i = 1, \ldots, n$

D2. $D_G\varphi \Rightarrow D_{G'}\varphi$ if $G \subseteq G'$

In addition, the D_G operator has all the properties of the knowledge operator. What these are depends on the system we consider. Thus, for example, in all cases A2 applies to D_G, so that the following axiom is valid:

$$\big(D_G\varphi \wedge D_G(\varphi \Rightarrow \psi)\big) \Rightarrow D_G\psi.$$

If in addition we take the $\mathcal{K}_i$ relations to be reflexive, so that knowledge satisfies A3, then so does distributed knowledge; i.e., we get in addition the axiom $D_G\varphi \Rightarrow \varphi$. Similar remarks hold for A4 and A5. Let K_n^D (resp., T_n^D, $\mathrm{S4}_n^D$, $\mathrm{S5}_n^D$, $\mathrm{KD45}_n^D$) be the system that results from adding axioms D1, D2 to K_n (resp., T_n, $\mathrm{S4}_n$, $\mathrm{S5}_n$, $\mathrm{KD45}_n$), and assuming that all of the other axioms apply to the modal operators D_G as well as K_i. Thus, for example, $\mathrm{S4}_n^D$ includes the axiom $D_G\varphi \Rightarrow D_G D_G\varphi$.

Theorem 3.4.1 *For formulas in the language $\mathcal{L}_n^D$*

(a) K_n^D *is a sound and complete axiomatization with respect to* $\mathcal{M}_n$,

(b) T_n^D *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^r$,

(c) $\mathrm{S4}_n^D$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rt}$,

(d) $\mathrm{S5}_n^D$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{rst}$,

(e) $\mathrm{KD45}_n^D$ *is a sound and complete axiomatization with respect to* $\mathcal{M}_n^{elt}$.

Proof The proof of soundness is straightforward (see Exercise 2.10). Although the basic ideas of the completeness proof are similar to those we have encountered before, the details are somewhat technical. We just sketch the main ideas here, leaving the details to the exercises. We consider only the case of K_n^D here.

We start with a canonical structure constructed just as in the proof of Theorem 3.1.3, treating the distributed knowledge operators D_G exactly like the K_i operators. That is, for each maximal K_n^D-consistent set V, we define the set V/D_G in the obvious way and define binary relations $\mathcal{K}_G$ on S via $(s_V, s_W) \in \mathcal{K}_G$ iff $V/D_G \subseteq W$. From axiom D1 it follows that $\mathcal{K}_{\{i\}}$ (the binary relation derived using D_G, where G is the singleton set $\{i\}$) is equal to $\mathcal{K}_i$ (the binary relation derived using K_i). It can be shown that $\mathcal{K}_G \subseteq \bigcap_{i\in G} \mathcal{K}_i$; however, in general, equality does not hold. By making multiple copies of states in the canonical structure that are in $\bigcap_{i\in G} \mathcal{K}_i$ and not in $\mathcal{K}_G$, it is possible to construct a structure at which the same formulas are true in corresponding states, and in which $\bigcap_{i\in G} \mathcal{K}_i$ and $\mathcal{K}_G$ coincide. This gives us the desired structure. (See Exercise 3.30 for further details.) ∎

We have considered axiom systems for the languages $\mathcal{L}_n^C$ and $\mathcal{L}_n^D$. It is not too hard to show that we can get sound and complete axiomatizations for the language $\mathcal{L}_n^{CD}$, which has modal operators for common knowledge and distributed knowledge, by combining the axioms for common knowledge and distributed knowledge. It can also be shown that the validity problem is decidable. There are no interesting new ideas involved in doing this, so we shall not carry out that exercise here.

3.5 The Complexity of the Validity Problem

In earlier sections, we have shown that the validity problem for the various logics we have been considering is decidable. In this section, we examine the issue more carefully. In particular, we attempt to completely characterize the inherent difficulty of deciding validity for all the logics we consider. This will give us some insight into which features of a logic make deciding validity difficult. We characterize the "inherent difficulty" of a problem in terms of computational complexity. We briefly review the necessary notions here.

Formally, we view everything in terms of the difficulty of determining membership in a set. Thus, the validity problem is viewed as the problem of determining whether a given formula φ is an element of the set of formulas valid with respect to a class of structures. The difficulty of determining set membership is usually measured by the amount of time and/or space (memory) required to do this, as a function of the input size. Since the inputs we consider in this section are formulas, we will typically be interested in the difficulty of determining whether a formula φ is valid or satisfiable as a function of $|\varphi|$.

We are usually most interested in *deterministic* computations, where at any point in a computation, the next step of the computation is uniquely determined. However, thinking in terms of *nondeterministic* computations—ones where the program may "guess" which of a finite number of steps to take—has been very helpful in classifying the intrinsic difficulty of a number of problems. A deterministic algorithm must conclude by either accepting the input (intuitively, saying "Yes, the formula that is the input is valid") or rejecting the input (intuitively, saying "No, the formula that is the input is not valid"). A nondeterministic algorithm is said to accept an input if it accepts for some appropriate sequence of guesses.

The complexity classes we are most concerned with here are *P*, *NP*, *PSPACE*, *EXPTIME*, and *NEXPTIME*: those sets such that determining whether a given element x is a member of the set can be done in deterministic polynomial time, nondeterministic polynomial time, deterministic polynomial space, deterministic exponential time (where exponential in n means in time 2^{dn} for some constant d), and nondeterministic exponential time, respectively (as a function of the size of x). It is not hard to show that $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME$. It is also known that $P \neq EXPTIME$ and that $NP \neq NEXPTIME$. While it is conjectured that all the other inclusions are strict, proving this remains elusive. The $P = NP$ problem is currently considered the most important open problem in the field of computational complexity. Interestingly, it is known that nondeterminism does not add any power

at the level of polynomial space: nondeterministic polynomial space is equivalent to deterministic polynomial space.

Given a complexity class $\mathcal{C}$, the class co-$\mathcal{C}$ consists of all of the sets whose *complement* is a member of $\mathcal{C}$. Notice that if we have a deterministic algorithm $\mathbf{A}$ for deciding membership in a set A, then it is easy to convert it to an algorithm $\mathbf{A}'$ for deciding membership in the complement of A that runs in the same space and/or time bounds: $\mathbf{A}'$ accepts an input x iff $\mathbf{A}$ rejects. It follows that $\mathcal{C} = \text{co-}\mathcal{C}$ must hold for every deterministic complexity class $\mathcal{C}$, in particular, for *P*, *PSPACE* and *EXPTIME*. This is not necessarily the case for a nondeterministic algorithm. There is no obvious way to construct an algorithm $\mathbf{A}'$ that will accept an element of the complement of A by an appropriate sequence of guesses. Thus, in particular, it is not known whether *NP* = co-*NP*. Clearly, if *P* = *NP*, then it would immediately follow that *NP* = co-*NP*, but it is conjectured that in fact *NP* $\neq$ co-*NP*.

Roughly speaking, a set A is said to be *hard* with respect to a complexity class $\mathcal{C}$ (e.g., *NP*-hard, *PSPACE*-hard, etc.) if every set in $\mathcal{C}$ can be efficiently *reduced* to A; i.e., for every set B in $\mathcal{C}$, an algorithm deciding membership in B can be easily obtained from an algorithm for deciding membership in A. A set is *complete* with respect to a complexity class $\mathcal{C}$, or $\mathcal{C}$-*complete* if it is both in $\mathcal{C}$ and $\mathcal{C}$-hard.

The problem of determining whether a formula of propositional logic is satisfiable (i.e., the problem of determining whether a given propositional formula is in the set of satisfiable propositional formulas) is *NP*-complete. In particular, this means that if we could find a polynomial-time algorithm for deciding satisfiability for propositional logic, we would also have polynomial-time algorithms for all other *NP* problems. This is considered highly unlikely.

What about the complexity of determining validity? Notice that satisfiability and validity are complementary problems. A formula φ is valid exactly if $\neg\varphi$ is not satisfiable. Thus, if the satisfiability problem for a logic is complete for some complexity class $\mathcal{C}$, then the validity problem must be complete for co-$\mathcal{C}$. In particular, this means that the validity problem for propositional logic is co-*NP*-complete.

The complexity of the satisfiability and validity problem for numerous logics other than propositional logic has been studied. It is remarkable how many of these problems can be completely characterized in terms of the complexity classes discussed here. In particular, this is true for the logics we consider here. (We remark that when we speak of a *logic*, we typically mean an axiom system together with a corresponding class of structures. We usually refer to a logic by the name of the axiom system. Thus, when we speak of "the satisfiability problem for (the logic) $\mathrm{S4}_n$," we mean the problem of determining if a formula $\varphi \in \mathcal{L}_n$ is satisfiable with respect to $\mathcal{M}_n^{rt}$ or, equivalently, the problem of determining if φ is $\mathrm{S4}_n$-consistent.)

$S5_1$, $KD45_1$	K_n, T_n, $S4_n$, $n \geq 1$; $S5_n$, $KD45_n$, $n \geq 2$	K_n^C, T_n^C, $n \geq 1$; $S4_n^C$, $S5_n^C$, $KD45_n^C$, $n \geq 2$
NP-complete	*PSPACE*-complete	*EXPTIME*-complete

Table 3.2 The complexity of the satisfiability problem for logics of knowledge

The situation is summarized in Table 3.2. The results in the table are stated in terms of the satisfiability problem. Since φ is valid iff $\neg\varphi$ is not satisfiable, a solution to the validity problem quickly leads to a solution to the satisfiability problem, and vice versa. In particular, in those cases where the satisfiability problem is *PSPACE*- or *EXPTIME*-complete, the validity problem has the same complexity as the satisfiability problem. In the cases where the satisfiability problem is *NP*-complete, the validity problem is co-*NP*-complete.

As can be seen from the table, without common knowledge in the language, the satisfiability problem is in general *PSPACE*-complete. In the case of $S4_2$, for example, this means that there is an algorithm for deciding whether a formula is satisfiable with respect to $\mathcal{M}_2^n$ (or, equivalently, whether it is $S4_2$-consistent) that runs in polynomial space, and every *PSPACE* problem can be efficiently reduced to the satisfiability problem for $S4_2$. The only exception to the *PSPACE* result is in the case of S5 and KD45 (for only one agent), where the satisfiability problem is *NP*-complete. This says that in the case of S5 and KD45, going from one agent to many agents increases the complexity of the logic (provided that *PSPACE* $\neq$ *NP*). Adding common knowledge causes a further increase in complexity, to *EXPTIME*-complete.

We remark that we do not mention languages involving distributed knowledge in our table. This is because adding distributed knowledge to the language does not affect the complexity. Thus, for example, the complexity of the satisfiability problem for $S5_n$ is the same as that for $S5_n^D$. We also do not mention the single-agent case for $S4^C$, $S5^C$, and $KD45^C$, because in these cases common knowledge reduces to knowledge.

In the next section, we prove *NP*-completeness for S5 and KD45 in detail. The proofs for the *PSPACE* upper and lower bounds are quite technical and are beyond the scope of this book. (See the notes for references.) We remark that our lower bounds suggest that we cannot hope for automatic theorem provers for these logics that are guaranteed to work well (in the sense of providing the right answer quickly) for all inputs.

It is interesting to compare the results mentioned in the table with those proved in Section 3.2. The results there give us a nondeterministic exponential time algorithm for deciding satisfiability: given a formula φ, we simply guess an exponential-sized structure satisfying φ (if φ is satisfiable, then there must be such a structure by the results of Section 3.2) and then verify (using the model-checking algorithm) that the structure indeed satisfies φ. Since the structure is exponential-sized, the model-checking can be done in exponential time. The algorithm is in nondeterministic exponential time because of the guess made at the beginning. Because, as we mentioned earlier, it is strongly suspected that exponential time, and hence nondeterministic exponential time, is worse than polynomial space, this suggests that the algorithm of Section 3.2 is not optimal.

3.6 *NP*-Completeness Results for S5 and KD45

In this section, we focus on the single-agent case of $S5_n$ and $KD45_n$, namely S5 and KD45. It is clear that the satisfiability problem for S5 and KD45 must be at least *NP*-hard, since it is at least as hard as the satisfiability problem for propositional logic. It is somewhat surprising that reasoning about knowledge in this case does not add any further complexity. We start with S5 here.

Theorem 3.6.1 *The satisfiability problem for S5 is NP-complete (and thus the validity problem for S5 is co-NP-complete).*

The key step in the proof of Theorem 3.6.1 lies in showing that satisfiable S5 formulas can be satisfied in structures with very few states.

Proposition 3.6.2 *An S5 formula φ is satisfiable if and only if it is satisfiable in a structure in $\mathcal{M}_1^{rst}$ with at most $|\varphi|$ states.*

Proof Suppose $(M, s) \models \varphi$. By Proposition 3.1.6, we can assume without loss of generality that $M = (S, \pi, \mathcal{K}_1)$, where $\mathcal{K}_1$ is a universal relation, so that $(t, t') \in \mathcal{K}_1$ for all $t, t' \in S$. Let F be the set of subformulas of φ of the form $K_1\psi$ for which $(M, s) \models \neg K_1\psi$; i.e., F is the set of subformulas of φ that have the form $K_1\psi$ and are false at the state s. For each formula $K_1\psi \in F$, there must be a state $s_\psi \in S$ such that $(M, s_\psi) \models \neg\psi$. Let $M' = (S', \pi', \mathcal{K}_1')$, where (a) $S' = \{s\} \cup \{s_\psi \mid \psi \in F\}$, (b) π' is the restriction of π to S', and (c) $\mathcal{K}_1' = \{(t, t') \mid t, t' \in S'\}$. Since $|F| < |Sub(\varphi)| \leq |\varphi|$, it follows that $|S'| \leq |\varphi|$. We now show that for all states $s' \in S'$ and for all subformulas ψ of φ (including φ itself), $(M, s') \models \psi$ iff $(M', s') \models \psi$. As usual,

we proceed by induction on the structure of ψ. The only nontrivial case is when ψ is of the form $K_1\psi'$. Suppose $s' \in S'$. If $(M, s') \models K_1\psi'$, then $(M, t) \models \psi'$ for all $t \in S$, so, in particular, $(M, t) \models \psi'$ for all $t \in S'$. By the induction hypothesis, $(M', t) \models \psi'$ for all $t \in S'$, so $(M', s') \models K_1\psi'$. For the converse, suppose that $(M, s') \not\models K_1\psi'$. Then $(M, t) \models \neg\psi'$ for some $t \in S$. Because $\mathcal{K}_1$ is the universal relation, it follows that $(s, t) \in \mathcal{K}_1$, so $(M, s) \models \neg K_1\psi'$. But then it follows that $K_1\psi' \in F$, and $(M, s_{\psi'}) \models \neg\psi'$. By construction, $s_{\psi'} \in S'$, and by the induction hypothesis, we also have $(M', s_{\psi'}) \models \neg\psi'$. Because $(s', s_{\psi'}) \in \mathcal{K}_1'$, we have $(M', s') \models \neg K_1\psi'$, and so $(M', s') \not\models K_1\psi'$ as desired. This concludes the proof that $(M, s') \models \psi$ iff $(M', s') \models \psi$ for all subformulas ψ of φ and all $s' \in S'$. Since $s \in S'$ and $(M, s) \models \varphi$ by assumption, we also have $(M', s) \models \varphi$. ∎

Proof of Theorem 3.6.1 As we have already mentioned, S5 satisfiability is clearly *NP*-hard. We now give an *NP* algorithm for deciding S5 satisfiability. Intuitively, given a formula φ, we simply guess a structure $M \in \mathcal{M}_n^{rst}$ with a universal possibility relation and at most $|\varphi|$ states, and verify that φ is satisfied in M. More formally, we proceed as follows. Given a formula φ, where $|\varphi| = m$, we nondeterministically guess a structure $M = (S, \pi, \mathcal{K}_1)$, where S is a set of $k \leq m$ states, $(s, t) \in \mathcal{K}_1$ for all $s, t \in S$, and for all $s \in S$ and all primitive propositions p not appearing in φ, we have $\pi(s)(p) =$ **false**. (Note that the only "guessing" that enters here is the choice of k and the truth values $\pi(s)(q)$ that the primitive propositions q appearing in φ have in the k states of S.) Since at most m primitive propositions appear in φ, guessing such a structure can be done in nondeterministic time $O(m^2)$ (i.e., at most cm^2 steps for some constant c). Next, we check whether φ is satisfied at some state $s \in S$. By Proposition 3.2.1, this can be done deterministically in time $O(m^3)$. By Proposition 3.6.2, if φ is satisfiable, it must be satisfiable in one of the structures of the kind we guessed. (Of course, if φ is not satisfiable, no guess will be right.) Thus, we have a nondeterministic $O(m^3)$ algorithm for deciding if φ is satisfiable. ∎

We can prove essentially the same results for KD45 as for S5. Using Proposition 3.1.6, we can show the following:

Proposition 3.6.3 *A KD45 formula φ is satisfiable iff it is satisfiable in a structure in $\mathcal{M}_1^{elt}$ with at most $|\varphi|$ states.*

Proof See Exercise 3.34. ∎

Using Proposition 3.6.3 just as we used Proposition 3.2.1, we can now prove the following theorem:

Theorem 3.6.4 *The satisfiability problem for KD45 is NP-complete (and thus the validity problem for KD45 is co-NP-complete).*

Proof See Exercise 3.34. ∎

We remark that results similar to Proposition 3.6.3 and Theorem 3.6.4 can also be proved for K45 (Exercise 3.35).

3.7 The First-Order Logic of Knowledge

So far, we have considered only *propositional* modal logic. That is, the formulas we have allowed contain only primitive propositions, together with propositional connectives such as $\wedge$ and $\neg$, and modal operators such as K_i and C. *First-order logic* has greater expressive power than propositional logic. It allows us to reason about individuals in a domain and properties that they have. Among other things, first-order logic allows us to express properties that all individuals have and that some individuals have, by using a universal quantifier ($\forall$, or "for all") and an existential quantifier ($\exists$, or "there exists"). For example, we can say that Pete is the governor of California using a formula such as $Governor(California, Pete)$. To say that every state has a governor, we might write the first-order formula $\forall x(State(x) \Rightarrow \exists y Governor(x, y))$: for all states x, there exists y such that the governor of x is y. First-order logic is, in a precise sense, expressive enough to capture all of propositional modal logic (see Exercise 3.37). By combining the quantifiers of first-order logic and the modal operators of propositional modal logic, we get a yet more expressive logic, *first-order modal logic*. For example, neither in first-order logic nor in propositional modal logic can we say that Alice knows that California has a governor. We can, however, say this in first-order modal logic with the formula

$$K_{Alice}\exists y Governor(California, y).$$

There are some subtleties involved in combining first-order quantifiers with modal operators. We briefly discuss them in this section, to give the reader a feeling for the issues that arise.

Despite its additional power, we make relatively little use of first-order modal logic in the rest of the book, both because most of the examples that we discuss can be expressed using propositional modal logic and because most of the issues that we are interested in already arise in the propositional case. Nevertheless, the first-order case may well be important for more complex applications.

3.7.1 First-Order Logic

In this section we briefly review first-order logic. The reader familiar with first-order logic may still want to skim this section to get acquainted with our notation.

In propositional logic, we start with a set Φ of primitive propositions. In first-order logic, we start with a set $\mathcal{T}$ of *relation symbols*, *function symbols*, and *constant symbols*. Each relation symbol and function symbol has some *arity*, which intuitively corresponds to the number of arguments it takes. If the arity is k, then we refer to the symbol as k-ary. In our earlier example, the relation symbol *Governor* has arity 2: that is, $Governor(x, y)$ has two arguments, x and y. A function symbol *Gov*, where intuitively $Gov(x) = y$ means that the governor of state x is person y, has arity 1, since it takes only one argument, namely x. We refer to the set of relation symbols, function symbols, and constant symbols as the *vocabulary*.

We assume an infinite supply of *variables*, which we usually write as x and y, possibly along with subscripts. Constant symbols and variables are both used to denote individuals in the domain. We can also form more complicated terms denoting individuals by using function symbols. Formally, the set of *terms* is formed by starting with variables and constant symbols, and closing off under function application, so that if f is a k-ary function symbol, and if $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term. We need terms to define formulas. An *atomic formula* is either of the form $P(t_1, \ldots, t_k)$, where P is a k-ary relation symbol and $t_1, \ldots, t_k$ are terms, or of the form $t_1 = t_2$, where t_1 and t_2 are terms. As in propositional logic, if φ and ψ are formulas, then so are $\neg\varphi$ and $\varphi \wedge \psi$. In addition, we can form new formulas using quantifiers. If φ is a formula and x is a variable, then $\exists x\varphi$ is also a formula. We use the same abbreviations as we did in the propositional case. For example, $\varphi_1 \vee \varphi_2$ is an abbreviation for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$. Furthermore, we write $\forall x\varphi$ as an abbreviation for $\neg\exists x\neg\varphi$.

We give semantics to first-order formulas using *relational structures*. Roughly speaking, a relational structure consists of a domain of individuals and a way of associating with each of the elements of the vocabulary corresponding entities over the domain. Thus, a constant symbol is associated with an element of the domain, a function symbol is associated with a function on the domain, and so on. More precisely, fix a vocabulary $\mathcal{T}$. A *relational $\mathcal{T}$-structure* (which we sometimes simply call a relational structure or just a structure) $\mathcal{A}$ consists of a nonempty set, denoted $dom(\mathcal{A})$, called the *domain*, an assignment of a k-ary relation $P^{\mathcal{A}} \subseteq dom(\mathcal{A})^k$ to each k-ary relation symbol P of $\mathcal{T}$, an assignment of a k-ary function $f^{\mathcal{A}} : dom(\mathcal{A})^k \rightarrow dom(\mathcal{A})$ to each k-ary function symbol f of $\mathcal{T}$, and an assignment of a member $c^{\mathcal{A}}$ of the domain to each constant symbol c. We refer to $P^{\mathcal{A}}$ as the *interpretation* of P in $\mathcal{A}$, and similarly for function symbols and constant symbols.

As an example, suppose that $\mathcal{T}$ consists of one binary relation symbol E. In that case, a $\mathcal{T}$-structure is simply a graph. (Recall that a graph consists of a set of nodes, some of which are connected by edges.) The domain is the set of nodes of the graph, and the interpretation of E is the edge relation of the graph, so that there is an edge from a_1 to a_2 exactly if $(a_1, a_2) \in E^{\mathcal{A}}$.

Notice that a relational structure does not provide an interpretation of the variables. Technically, it turns out to be convenient to have a separate function that does this. A *valuation* V on a structure $\mathcal{A}$ is a function from variables to elements of $dom(\mathcal{A})$. Recall that we suggested that terms are intended to represent members of the domain of a structure $\mathcal{A}$. Given a structure $\mathcal{A}$ and a valuation V on $\mathcal{A}$, we can inductively extend V in a straightforward way to a function, denoted $V^{\mathcal{A}}$ (although we frequently omit the superscript $\mathcal{A}$ when it is clear from context), that maps terms to elements of $dom(\mathcal{A})$, as follows. Define $V^{\mathcal{A}}(c) = c^{\mathcal{A}}$ for each constant symbol c, and then extend the definition by induction on the structure of terms, by taking $V^{\mathcal{A}}(f(t_1, \ldots, t_k)) = f^{\mathcal{A}}(V^{\mathcal{A}}(t_1), \ldots, V^{\mathcal{A}}(t_k))$.

With these definitions in hand, we can now define what it means for a formula to be true in a relational structure. Before we give the formal definition, we consider a few examples. Let *Tall* be a unary relation symbol, and let *President* be a constant symbol. What does it mean for *Tall*(*President*) to be true in the structure $\mathcal{A}$? If we think of the domain of $\mathcal{A}$ as consisting of people, then the interpretation $Tall^{\mathcal{A}}$ of the *Tall* relation can be thought of intuitively as the set of all tall people in the domain. Assume that $President^{\mathcal{A}} = Bill$, so that, intuitively, the president is Bill. Then *Tall*(*President*) is true in the structure $\mathcal{A}$ precisely if *Bill* is in the relation $Tall^{\mathcal{A}}$, that is, intuitively, if Bill is tall. How should we deal with quantification? In particular, what should it mean for $\exists x Tall(x)$ to be true in the structure $\mathcal{A}$? Intuitively, this formula is true when there exists a tall person in the domain of $\mathcal{A}$. Formally, $\exists x Tall(x)$ is true in the structure $\mathcal{A}$ precisely if the relation $Tall^{\mathcal{A}}$ is nonempty. Similarly, $\forall x Tall(x)$ is true in the structure $\mathcal{A}$ precisely if the relation $Tall^{\mathcal{A}}$ contains every member of the domain of $\mathcal{A}$, that is, if everyone is tall. As a final example, consider the formula $Governor(c, x)$, where c is a constant symbol and x is a variable. It does not make sense to ask whether or not the formula $Governor(c, x)$ is true in a structure $\mathcal{A}$ without knowing what value x takes on. Here we make use of valuations, which assign to each variable a member of the domain of the structure. Thus, rather than ask whether $Governor(c, x)$ is true in a structure $\mathcal{A}$, we instead ask whether $Governor(c, x)$ is true in a structure $\mathcal{A}$ *under a given valuation* V. Assume that $c^{\mathcal{A}} = California$ and $V(x) = Pete$, so that c takes the value *California* in the structure $\mathcal{A}$ and x takes the value *Pete* under V. Then we say that $Governor(c, x)$ is true in the structure $\mathcal{A}$ under the valuation V

precisely if $(V(c), V(x)) = (c^{\mathcal{A}}, Pete) = (California, Pete) \in Governor^{\mathcal{A}}$. Intuitively, $Governor(c, x)$ is true in the structure $\mathcal{A}$ under the valuation V iff Pete is the governor of California according to the structure $\mathcal{A}$.

We now give the formal semantics. Fix a relational structure $\mathcal{A}$. For each valuation V on $\mathcal{A}$ and formula φ, we define what it means for φ to be true in $\mathcal{A}$ under the valuation V, written $(\mathcal{A}, V) \models \varphi$. If V is a valuation, x is a variable, and $a \in dom(\mathcal{A})$, then let $V[x/a]$ be the valuation V' such that $V'(y) = V(y)$ for every variable y except x, and $V'(x) = a$. Thus, $V[x/a]$ agrees with V, except possibly on x, and it assigns the value a to x.

$(\mathcal{A}, V) \models P(t_1, \ldots, t_k)$, where P is a k-ary relation symbol and $t_1, \ldots, t_k$ are terms, iff $(V(t_1), \ldots, V(t_k)) \in P^{\mathcal{A}}$

$(\mathcal{A}, V) \models (t_1 = t_2)$, where t_1 and t_2 are terms, iff $V(t_1) = V(t_2)$

$(\mathcal{A}, V) \models \neg\varphi$ iff $(\mathcal{A}, V) \not\models \varphi$

$(\mathcal{A}, V) \models \varphi_1 \wedge \varphi_2$ iff $(\mathcal{A}, V) \models \varphi_1$ and $(\mathcal{A}, V) \models \varphi_2$

$(\mathcal{A}, V) \models \exists x\varphi$ iff $(\mathcal{A}, V[x/a]) \models \varphi$ for some $a \in dom(\mathcal{A})$.

Recall that $\forall x\varphi$ is taken to be an abbreviation for $\neg\exists x\neg\varphi$. It is easy to see that $(\mathcal{A}, V) \models \forall x\varphi$ iff $(\mathcal{A}, V[x/a]) \models \varphi$ for every $a \in dom(\mathcal{A})$ (Exercise 3.38).

To decide whether the formula $Tall(President)$ is true in the structure $\mathcal{A}$ under the valuation V, the role of V is irrelevant. That is, $(\mathcal{A}, V) \models Tall(President)$ iff $(\mathcal{A}, V') \models Tall(President)$, where V and V' are arbitrary valuations. A similar comment applies to the formula $\exists x Tall(x)$. However, this is not the case for the formula $Governor(c, x)$, where c is a constant symbol and x is a variable. There may be valuations V and V' such that $(\mathcal{A}, V) \models Governor(c, x)$ but $(\mathcal{A}, V') \not\models Governor(c, x)$, so that $V(x)$ is the governor of California, but $V'(x)$ is not.

To understand better what is going on here, we need some definitions. Roughly speaking, we say that an occurrence of a variable x in φ is *bound* by the quantifier $\forall x$ in a formula such as $\forall x\varphi$ or by $\exists x$ in $\exists x\varphi$, and that an occurrence of a variable in a formula is *free* if it is not bound. (A formal definition of what it means for an occurrence of a variable to be free is given in Exercise 3.39.) A formula in which no occurrences of variables are free is called a *sentence*. Observe that x is free in the formula $Governor(c, x)$, but no variables are free in the the formulas $Tall(President)$ and $\exists x Tall(x)$, so the latter two formulas are sentences. It is not hard to show that the valuation does not affect the truth of a sentence. That is, if φ is a sentence, and V and V' are valuations on the structure $\mathcal{A}$, then $(\mathcal{A}, V) \models \varphi$ iff $(\mathcal{A}, V') \models \varphi$ (Exercise 3.39). In other words, a sentence is true or false in a structure, independent of the valuation used.

3.7.2 First-Order Modal Logic

Just as the syntax of propositional modal logic is obtained from that of propositional logic by adding the modal operators K_i, we get the syntax of first-order modal logic from that of first-order logic by adding the modal operators K_i. Thus, we define the language of first-order modal logic by taking the definition we gave for first-order formulas and also closing off under the modal operators $K_1, \ldots, K_n$, so that if φ is a first-order modal formula, then so is $K_i\varphi$. For example, $\forall x(K_1 Red(x))$ is a first-order modal formula, which intuitively says that for every x, agent 1 knows that x is red.

The semantics of first-order modal logic uses *relational Kripke structures*. In a (propositional) Kripke structure, each world is associated with a truth assignment to the primitive propositions via the function π. In a relational Kripke structure, the π function associates with each world a relational structure. Formally, we define a relational Kripke structure for n agents over a vocabulary $\mathcal{T}$ to be a tuple $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where S is a set of *states*, π associates with each state in S a $\mathcal{T}$-structure (i.e., $\pi(s)$ is a $\mathcal{T}$-structure for each state $s \in S$), and $\mathcal{K}_i$ is a binary relation on S.

The semantics of first-order modal logic is, for the most part, the result of combining the semantics of first-order logic and the semantics of modal logic in a straightforward way. But there are a few subtleties, as we shall see. We begin with some examples. Just as in the propositional case, we would like a formula $K_i\varphi$ to be true at a state s of a relational Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ precisely if φ is true at every state t such that $(s, t) \in \mathcal{K}_i$. As an example, let φ be the formula $Tall(President)$. In some states t of the relational Kripke structure the president might be Bill (that is, $President^{\pi(t)} = Bill$), and in some states t the president might be George. We would like the formula $K_i Tall(President)$ to be true if the president is tall in every possible world, even if the president is a different person in different worlds. It is quite possible for agent i to know that the president is tall without knowing exactly who the president is.

What about the formula $K_i Tall(x)$, where x is a variable? Since x is a variable, we need a valuation to decide whether this formula is true at a given state. Assume that $V(x) = Bill$. For $K_i Tall(x)$ to be true, we want Bill to be tall in every possible world. But now there is a problem: although *Bill* may be a member of the domain of the relational structure $\pi(s)$, it is possible that *Bill* is not a member of the domain of $\pi(t)$ for some state t that agent i considers possible at state s. To get around this problem, we restrict attention for now to *common-domain Kripke structures*, that is, relational Kripke structures where the domain is the same at every state. We discuss the implications of this restriction in more detail later.

Under the restriction to common-domain structures, defining truth of formulas becomes quite straightforward. Fix a relational Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where the states have common domain U. A *valuation* V on M is a function that assigns to each variable a member of U. This means that $V(x)$ is independent of the state, although the interpretation of, say, a constant c does depend on the state. As we shall see, this lets us identify the same individual in the domain at different states and plays an important role in the expressive power of first-order modal logic. For a state s of M, a valuation V on M, and a formula φ, we define what it means for φ to be true at the state s of M under the valuation V, written $(M, s, V) \models \varphi$. Most of the definitions are just as in the first-order case, where the role of $\mathcal{A}$ is played by $\pi(s)$. For example,

$(M, s, V) \models P(t_1, \ldots, t_k)$, where P is a k-ary relation symbol and $t_1, \ldots, t_k$ are terms, iff $(V^{\pi(s)}(t_1), \ldots, V^{\pi(s)}(t_k)) \in P^{\pi(s)}$.

In the case of formulas $K_i\varphi$, the definition is just as in the propositional case in Chapter 2:

$(M, s, V) \models K_i\varphi$ iff $(M, t, V) \models \varphi$ for every t such that $(s, t) \in \mathcal{K}_i$.

As we said earlier, first-order modal logic is significantly more expressive than either first-order logic or propositional modal logic. One important example of its extra expressive power is that it allows us to distinguish between "knowing that" and "knowing who." For example, the formula $K_{Alice}\exists x(x = \mathit{President})$ says that Alice knows that someone is the president. This formula is valid (since $\exists x(x = \mathit{President})$ is valid). In particular, the formula is true even in a world where Alice does not know whether Bill or George is the president; she may consider one world possible where Bill is the president, and consider another world possible where George is the president. Thus, although Alice knows that there is a president, she may not know exactly who the president is. To say that Alice knows who the president is, we would write $\exists x K_{Alice}(x = \mathit{President})$. Because a valuation is independent of the state, it is easy to see that this formula says that there is one particular person who is president in every world that Alice considers possible. Notice that the fact that the valuation is independent of the state is crucial in allowing us to distinguish "knowing that" from "knowing who."

3.7.3 Assumptions on Domains

We restricted attention in the previous section to common-domain Kripke structures. This means that although we allow the interpretations of relation symbols, of function

symbols, and of constant symbols to vary from state to state in a given relational Kripke structure, we do not allow the domains to vary. Essentially, this assumption says that the domain is common knowledge. This assumption is quite reasonable in many applications. When analyzing a card game, players typically have common knowledge about which cards are in the deck. Nevertheless, there are certainly applications where the domain is not common knowledge. For example, although there are no unicorns in this world, we might like to imagine possible worlds where unicorns exist. On a more practical level, if our agent is a knowledge base reasoning about the employees in a company, then the agent may not know exactly how many employees the company has.

As we saw earlier, this assumption of a common domain arose in response to a technical problem, that of making sense of the truth value of a formula where a free variable appears in the scope of a modal operator, such as in the formula $K_i Tall(x)$. Without the common-domain assumption, to decide if $K_i Tall(x)$ is true at a state s under a valuation V where $V(x) = Bill$, we have to consider the truth of $Tall(x)$ at a state t where *Bill* may not be in the domain. It is not clear what the truth value of $K_i Tall(x)$ should be in this case.

We can solve this problem by making a somewhat weaker assumption than the common-domain assumption. It suffices to assume that if world t is considered possible in world s, then the domain corresponding to s is a subset of the domain corresponding to t. Formally, this assumption says that if $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is a relational Kripke structure and $(s, t) \in \mathcal{K}_i$, then $dom(\pi(s)) \subseteq dom(\pi(t))$. Informally, this assumption says that every object that exists in a world s also exists in every world considered possible at s. We call this the *domain-inclusion assumption*.

While the domain-inclusion assumption lets us deal with more cases than the common-domain assumption, and does avoid the technical problems discussed above, it certainly does not handle all problems. For one thing, an agent cannot consider possible a world with fewer domain elements. This means that if we take the $\mathcal{K}_i$'s to be equivalence relations, as we have claimed that we want to do for many applications, or even just Euclidean relations, then the domain-inclusion assumption implies that in all worlds considered possible from a given world the domains must be the same. Thus, with the additional assumption that the relation is Euclidean, we cannot model in this framework the examples that we mentioned earlier involving unicorns or employees in a company.

Many solutions have been proposed for how to give a semantics without any assumptions whatsoever about relationships between domains of worlds within a relational Kripke structure. Nevertheless, it is fair to say that no solution has been universally accepted. Each proposed solution suffers from various problems. One proposed solution and a problem from which it suffers are discussed in Exercise 3.40.

3.7.4 Properties of Knowledge in Relational Kripke Structures

We now consider the properties of knowledge in relational Kripke structures. Just as before, we do this in terms of the formulas that are valid in relational Kripke structures. In the first-order case, we say that a formula is valid if it is true at every state of every relational Kripke structure under every valuation. To simplify the discussion, we assume a common domain.

In the propositional case, we saw that a sound and complete axiomatization could be obtained by considering all tautologies of propositional logic, together with some specific axioms about knowledge. It is easy to see that all the axioms of K_n are still valid in relational Kripke structures (Exercise 3.41). It is also intuitively clear that these axioms are not complete. We clearly need some axioms for first-order reasoning.

We might hope that we can get a complete axiomatization by adding all (substitution instances of) valid first-order formulas. Unfortunately, this results in an unsound system. There are two specific axioms of first-order logic that cause problems.

To state them we need a little notation. Suppose $\varphi(x)$ is a first-order formula in which some occurrences of x are free. Let t, t_1, and t_2 be terms, and let $\varphi(t)$ be the result of substituting t for all free occurrences of x. Assume for simplicity that no variable occurring in t, t_1, or t_2 is quantified in φ (so that, for example, for every variable y in t there is no subformula of φ of the form $\exists y\psi$; without this assumption, we may have inadvertent binding of the y in t by $\exists y$). Consider the following two axioms:

$$\varphi(t) \Rightarrow \exists x \varphi(x) \tag{3.3}$$

$$(t_1 = t_2) \Rightarrow \big(\varphi(t_1) \Leftrightarrow \varphi(t_2)\big) \tag{3.4}$$

It is easy to see that both of these axioms are valid in relational structures (Exercise 3.42). For the first one, if $\varphi(t)$ is true, then there is certainly some value we can assign to x that makes $\varphi(x)$ true, namely, the interpretation of t. Axiom (3.4) just says that "equals can be replaced by equals." As an example, taking $\varphi(x)$ to be $Governor(California, x)$, we have that $((x_1 = x_2) \Rightarrow (Governor(California, x_1)) \Leftrightarrow Governor(California, x_2))$ is valid. Although these axioms are valid in relational Kripke structures if $\varphi(x)$ is a first-order formula, we now show that neither axiom is valid if we allow φ to be an arbitrary modal formula.

We start with the first axiom. Let $\varphi(x)$ be the modal formula $K_i(Tall(x))$ and let t be *President*. With this substitution, the axiom becomes

$$K_i\big(Tall(President)\big) \Rightarrow \exists x\, K_i\big(Tall(x)\big). \tag{3.5}$$

As we noted earlier, the left-hand side of (3.5) is true if, intuitively, the president is tall in every possible world, even if the president is a different person in different worlds. However, the right-hand side of (3.5) is false if there is no one person who is tall in every possible world. Since it is possible simultaneously for the left-hand side of (3.5) to be true and the right-hand side to be false, it follows that (3.5) is not valid.

What is going on is that the valuation is independent of the state, and hence under a given valuation, a variable x is a *rigid designator*, that is, it denotes the same domain element in every state. On the other hand, a constant symbol such as *President* can denote different domain elements in distinct states. It is easy to see that (3.3) is valid if we restrict the term t to being a variable. More generally, we can show that (3.3) is valid if t is a rigid designator (Exercise 3.42).

To see that the second axiom is not valid in relational Kripke structures, let φ be $K_i(t_1 = x)$. Then the axiom becomes

$$(t_1 = t_2) \Rightarrow \big(K_i(t_1 = t_1) \Leftrightarrow K_i(t_1 = t_2)\big).$$

It is easy to see that $K_i(t_1 = t_1)$ is valid, so the axiom reduces to

$$(t_1 = t_2) \Rightarrow K_i(t_1 = t_2). \tag{3.6}$$

There is a famous example from the philosophical literature that shows that this is not valid. Because of its brightness, the planet Venus is called the morning star (at sunrise, when it appears in the east), and it is also called the evening star (at sunset, when it appears in the west). Ancient astronomers referred to the morning star as Phosphorus, and the evening star as Hesperus, and were unaware that Phosphorus and Hesperus were one and the same. Let the constant symbol *Phosphorus* play the role of t_1 in (3.6), let the constant symbol *Hesperus* play the role of t_2, and let agent i be an ancient astronomer. Then (3.6) is falsified: although Hesperus and Phosphorus are equal in the real world, the astronomer does not know this.

Notice that, again, the problem here arises because t_1 and t_2 may not be rigid designators. If we restrict attention to terms that are rigid designators, and, in particular, to variables, then (3.4) is valid in all relational Kripke structures (Exercise 3.42). It follows that the following special case of (3.6), called *Knowledge of Equality*, is valid:

$$(x_1 = x_2) \Rightarrow K_i(x_1 = x_2). \tag{3.7}$$

We remark that (3.3) and (3.4) are the only axioms of first-order logic that are not valid in relational Kripke structures. More precisely, there is a complete axiomatization of first-order logic that includes (3.3) and (3.4) as axioms such that all

substitution instances of all axioms besides (3.3) and (3.4) are valid in relational Kripke structures.

Suppose we restrict (3.3) and (3.4) so that if φ is a modal formula (that is, it has occurrences of K_i operators), then the terms t, t_1, and t_2 must be variables; we henceforth call these the *restricted versions* of (3.3) and (3.4). Note that the restricted versions of (3.3) and (3.4) are valid in relational Kripke structures. We might hope that by taking (substitution instances of) the axioms of first-order logic, using only the restricted versions of (3.3) and (3.4), together with the axioms and inference rules of K_n, we would have a sound and complete axiomatization for knowledge in first-order relational structures. The resulting system is sound, but it is not complete; there are two additional axioms we must add.

One new axiom arises because of the interaction between the first-order quantifier $\forall$ and the modal operator K_i, which can be thought of as a "knowledge quantifier." Consider the following formula, sometimes called the *Barcan formula*:

$$\forall x_1 \ldots \forall x_k K_i \varphi \Rightarrow K_i \forall x_1 \ldots \forall x_k \varphi.$$

It is fairly easy to see that the Barcan formula is valid (Exercise 3.43). Its validity, however, depends crucially on the common-domain assumption. For example, consider a relational Kripke structure whose common domain consists of precisely three elements, a_1, a_2, and a_3. Assume that Alice knows that a_1 is red, that a_2 is red, and that a_3 is red. Then, for all x, Alice knows that x is red; that is, $\forall x(K_A Red(x))$ holds. From the Barcan formula it follows that Alice knows that for every x, x is red; that is, $K_A(\forall x Red(x))$ holds. Without the common-domain assumption, we might argue intuitively that Alice does not know that every object is red, since Alice might consider it possible that there is a fourth object a_4 that is blue. In the presence of the common-domain assumption, Alice knows that a_1, a_2, and a_3 are the only domain elements, so this argument cannot be applied. One the other hand, the Barcan formula is not valid under the domain-inclusion assumption that we discussed earlier, where there really can be a fourth (non-red) object a_4 in another world (Exercise 3.44).

The second new axiom arises because of the interaction between the K_i operator and equality. This axiom, which is analogous to Knowledge of Equality (3.7), is called *Knowledge of Inequality*:

$$(x_1 \neq x_2) \Rightarrow K_i(x_1 \neq x_2). \tag{3.8}$$

Like Knowledge of Equality, this axiom is valid (Exercise 3.45). Unlike Knowledge of Equality, this axiom does not follow from the other axioms.

It turns out that no further new axioms beyond the Barcan formula and Knowledge of Inequality are needed to get a sound and complete axiomatization for the first-order theory of knowledge. Such an axiomatization (for structures with n agents) is obtained by combining

(a) the axiom system K_n,

(b) the axiom system for first-order logic referred to previously, except that we use the restricted versions of (3.3) and (3.4),

(c) the Barcan formula, and

(d) Knowledge of Inequality.

Notice that if we do not allow function or constant symbols in the vocabulary, then the only terms are variables, which are rigid designators. In this case, all substitution instances of axioms (3.3) and (3.4) are valid (in fact, the restricted versions of (3.3) and (3.4) are identical to the unrestricted versions), so we can simplify the statement of (b) above.

We have already seen that for Kripke structures $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, additional properties of the $\mathcal{K}_i$ relations give us additional axioms for knowledge. Not surprisingly, the same is true for relational Kripke structures. For example, if each $\mathcal{K}_i$ is an equivalence relation, then we can modify the sound and complete axiomatizations we just described by replacing the axiom system K_n by the axiom system $\mathrm{S5}_n$. It is interesting that in this case we do not need to include the Barcan formula or the Knowledge of Inequality axiom, since they turn out to be consequences of the axioms of first-order logic, along with $\mathrm{S5}_n$ (see Exercises 3.43 and 3.45).

Exercises

3.1 Show that $|Sub(\varphi)| \leq |\varphi|$.

3.2 Show that if neither $F \cup \{\varphi\}$ nor $F \cup \{\neg\varphi\}$ is AX-consistent, then neither is $F \cup \{\varphi \vee \neg\varphi\}$.

3.3 Prove that a maximal AX-consistent set has properties (c) and (d), as claimed in the statement of Lemma 3.1.2.

3.4 Prove that K_n is sound for $\mathcal{M}_n$, using Theorem 3.1.1.

3.5 In the proof of Theorem 3.1.3, prove that $(M^c, s_V) \models \varphi$ iff $\varphi \in V$, in the case that φ is a conjunction or a negation.

3.6 Show that if $\{\varphi_1, \ldots, \varphi_k, \neg\psi\}$ is not K_n-consistent, then

$$\mathrm{K}_n \vdash \varphi_1 \Rightarrow \big(\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots)\big).$$

3.7 Prove, using induction on k together with axiom A2 and propositional reasoning, that

$$\mathrm{K}_n \vdash K_i\big(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots))\big) \Rightarrow \big(K_i\varphi_1 \Rightarrow (K_i\varphi_2 \Rightarrow (\ldots \Rightarrow (K_i\varphi_k \Rightarrow K_i\psi)\ldots))\big).$$

* **3.8** Let K_n' be the variant of K_n consisting of one inference rule, modus ponens (R1), and the following two axioms:

A1′. $K_{i_1} \ldots K_{i_k}\varphi$, where φ is an instance of a tautology of propositional calculus, $k \geq 0$, and $i_1, \ldots, i_k$ are arbitrary (not necessarily distinct) agents in $\{1, \ldots, n\}$,

A2′. $K_{i_1} \ldots K_{i_k}[(K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi]$, $i = 1, \ldots, n$, where, again, $k \geq 0$ and $i_1, \ldots, i_k$ are arbitrary (not necessarily distinct) agents in $\{1, \ldots, n\}$.

Thus, A1′ and A2′ look just like A1 and A2, except that a string of knowledge operators has been appended to the beginning of each formula. If we take $k = 0$ in each of A1′ and A2′, we get back A1 and A2.

Show that K_n is equivalent to K_n'; that is, show that a formula φ is provable in K_n iff it is provable in K_n'. Then show that the deduction theorem holds for K_n'. Find similar (equivalent) variants of T_n, $\mathrm{S4}_n$, $\mathrm{S5}_n$, and $\mathrm{KD45}_n$ for which the deduction theorem holds.

This shows that it is essentially R2—Knowledge Generalization—that causes the deduction theorem to fail for the logics that we have been considering.

3.9 Show that A6 is provable from A3, A1, and R1.

3.10 In this exercise, we consider when an agent can know both φ and $\neg\varphi$, or both φ and the fact that he does not know φ.

(a) Show that $K_1\varphi \wedge K_1\neg\varphi$ is consistent with K_n by constructing a Kripke structure that satisfies, for example, $K_1 p \wedge K_1\neg p$.

(b) Show that $K_n + \{A6\} \vdash \neg(K_i\varphi \wedge K_i\neg\varphi)$. Show as a consequence that it follows that $AX \vdash \neg(K_i\varphi \wedge K_i\neg\varphi)$ if AX is any one of T_n, $S4_n$, $S5_n$, or $KD45_n$.

(c) Show that $AX \vdash \neg K_i(\varphi \wedge \neg K_i\varphi)$ although $\varphi \wedge \neg K_i\varphi$ is consistent with AX, where AX is any of T_n, $S4_n$, $S5_n$, or $KD45_n$. Thus, although it is consistent in each of these logics for φ to be true but agent i not to know it, it is impossible for i to know this fact.

* **3.11** Give syntactic proofs of the following properties of common knowledge:

(a) $K_n^C \vdash (C_G\varphi \wedge C_G(\varphi \Rightarrow \psi)) \Rightarrow C_G\psi$,

(b) $T_n^C \vdash C_G\varphi \Rightarrow \varphi$,

(c) $K_n^C \vdash C_G\varphi \Rightarrow C_GC_G\varphi$ (note that the analogous axiom A4 is *not* needed),

(d) $S5_n^C \vdash \neg C_G\varphi \Rightarrow C_G\neg C_G\varphi$ (hint: show $S5_n^C \vdash \neg C_G\varphi \Leftrightarrow K_i\neg C_G\varphi$ for all $i \in G$),

(e) $S4_n^C \not\vdash \neg C_G\varphi \Rightarrow C_G\neg C_G\varphi$ (hint: show that $\neg C_G\varphi \wedge \neg C_G\neg C_G\varphi$ is satisfiable in some structure in $\mathcal{M}_n^{rt}$),

(f) $K_n^C \vdash C_G\varphi \Rightarrow C_{G'}\varphi$ if $G \supseteq G'$.

3.12 Prove Lemma 3.1.4.

3.13 In this exercise, we focus on the connection between axiom systems and possibility relations.

(a) Show that axiom A4 is valid in all structures where the possibility relation is transitive.

(b) Show that axiom A5 is valid in all structures where the possibility relation is Euclidean.

(c) Show that axiom A5 forces the possibility relation in the canonical structure to be Euclidean; in particular, show that if all instances of A5 are true at a state s_V in the canonical structure and (s_V, s_W), $(s_V, s_X) \in \mathcal{K}_i$, then $(s_W, s_X) \in \mathcal{K}_i$.

(d) Show that axiom A6 is valid in all structures where the possibility relation is serial.

(e) Show that axiom A6 forces the possibility relation in the canonical structure to be serial; in particular, show that if all instances of A6 are true at a state s_V, then there must be some state s_W such that $(s_V, s_W) \in \mathcal{K}_i$.

* **3.14** In this exercise, we show that the formulas proved valid in Exercise 2.12 are provable in $S5_n$. Prove the following:

(a) $S5_n \vdash \neg\varphi \Rightarrow K_i \neg K_i \varphi$,

(b) $S5_n \vdash \neg\varphi \Rightarrow K_{i_1} \ldots K_{i_k} \neg K_{i_k} \ldots K_{i_1} \varphi$ for any sequence $i_1, \ldots, i_k$ of agents,

(c) $S5_n \vdash \neg K_i \neg K_i \varphi \Leftrightarrow K_i \varphi$.

(Hint: for part (a), use the Knowledge Axiom and the Negative Introspection Axiom. For part (b), use part (a), induction, the Knowledge Generalization Rule, and the Distribution Axiom.)

3.15 Prove that the structure M in Figure 3.1 is a model of $S5_1$. (Hint: show that there is a Kripke structure M' with a single state s' such that for every formula $\varphi \in \mathcal{L}_1(\{p\})$ we have $(M, s) \models \varphi$ iff $(M, t) \models \varphi$ iff $(M', s') \models \varphi$.)

3.16 In this exercise, we show that there is a construction that converts a model M of T_n (resp., $S4_n$, $S5_n$, $KD45_n$) to a model M' in $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$) that in a precise sense is equivalent to M. Given a Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, let $M^r = (S, \pi, \mathcal{K}_1^r, \ldots, \mathcal{K}_n^r)$, where $\mathcal{K}_i^r$ is the reflexive closure of $\mathcal{K}_i$; i.e., $\mathcal{K}_i^r = \mathcal{K}_i \cup \{(s, s) \mid s \in S\}$. Similarly, let M^{rt} (resp., M^{rst}, M^{et}) be the structure obtained from M by replacing the $\mathcal{K}_i$ relations by their reflexive, transitive closures (resp., reflexive, symmetric and transitive closures; Euclidean and transitive closures). Note that we have M^{et} rather than M^{elt}, since it does not make sense to take the serial closure.

Prove the following:

(a) $M^r \in \mathcal{M}_n^r$; and if M is a model of T_n, then $(M, s) \equiv (M^r, s)$ for all states s in M.

(b) $M^{rt} \in \mathcal{M}_n^{rt}$; and if M is a model of $S4_n$, then $(M, s) \equiv (M^{rt}, s)$ for all states s in M.

(c) $M^{rst} \in \mathcal{M}_n^{rst}$; and if M is a model of $S5_n$, then $(M, s) \equiv (M^{rst}, s)$ for all states s in M.

(d) If M is a model of $KD45_n$, then so is M^{et}; moreover, in this case, $M^{et} \in \mathcal{M}_n^{elt}$ and $(M, s) \equiv (M^{et}, s)$ for all states s in M. Note that this case is slightly different from the previous cases, since it is not necessarily true in general that $M^{et} \in \mathcal{M}_n^{elt}$.

3.17 Let $\mathcal{F}_n$ be the class of all Kripke frames. Just as for structures, we can consider subclasses of $\mathcal{F}_n$ such as $\mathcal{F}_n^r$, $\mathcal{F}_n^{rt}$, $\mathcal{F}_n^{rst}$, and $\mathcal{F}_n^{elt}$. We say that a frame F is a *model of* T_n (resp., $S4_n$, $S5_n$, $KD45_n$) if every structure based on F is a model of T_n (resp., $S4_n$, $S5_n$, $KD45_n$). Prove the following:

(a) F is a model of T_n iff $F \in \mathcal{F}_n^r$,

(b) F is a model of $S4_n$ iff $F \in \mathcal{F}_n^{rt}$,

(c) F is a model of $S5_n$ iff $F \in \mathcal{F}_n^{rst}$,

(d) F is a model of $KD45_n$ iff $F \in \mathcal{F}_n^{elt}$.

3.18 In this exercise, we take a closer look at axiom A7.

(a) Show that axiom A7 is provable from the system consisting of A1, A3, A5, and R1.

(b) Show that axiom A7 forces the possibility relation in the canonical structure to be symmetric.

* **3.19** Show that A4 is provable from the system consisting of A1, A2, A3, A5, R1, and R2. (Hint: use Exercise 3.18 to show that $K_i\varphi \Rightarrow K_i\neg K_i\neg K_i\varphi$ is provable, and use A5 and propositional reasoning to show that $\neg K_i\neg K_i\varphi \Rightarrow K_i\varphi$ is also provable.)

3.20 Prove, using Lemma 3.1.4 and the techniques of Theorem 3.1.5, that the following axiom systems are equivalent (i.e., precisely the same formulas are provable in all of these systems):

(a) $S5_n$,

(b) the system consisting of {A1,A2,A4,A6,A7,R1,R2},

(c) the system consisting of {A1,A2,A3,A5,R1,R2},

(d) the system consisting of {A1,A2,A3,A4,A7,R1,R2}.

(Note that this exercise gives us an indirect proof of the preceding exercise.)

3.21 Fill in the missing details in the proof of Proposition 3.1.6. In particular, show that the relation $\mathcal{K}_1'$ defined in the the proof of part (b) has the properties claimed for it, and show that $(M, s) \equiv (M', s)$ for all states $s \in \{s_0\} \cup \mathcal{K}_1'(s_0)$ and all formulas ψ.

3.22 Show that an analogue to Proposition 3.1.6(b) holds for K45. (Hint: the only difference is that we can now take the set S to be empty.)

* **3.23** The *depth* of a formula is the depth of nesting of the K_i operators in the formula. Formally, we define depth by induction on structure of formulas. We define $depth(p) = 0$ for a primitive proposition p, $depth(\neg\varphi) = depth(\varphi)$, $depth(\varphi \wedge \psi) = \max(depth(\varphi), depth(\psi))$, and $depth(K_i\varphi) = depth(\varphi) + 1$.

(a) Show that for every formula $\varphi \in \mathcal{L}_1$ we can effectively find a formula φ' of depth 1 such that S5 $\vdash \varphi \Leftrightarrow \varphi'$. That is, for every formula in $\mathcal{L}_1$ we can effectively find an equivalent formula that is a Boolean combination of propositional formulas and formulas of the form $K_1\psi$, where ψ is propositional. (Hint: use the fact that $K_1(\varphi_1 \vee K_1\varphi_2) \Leftrightarrow (K_1\varphi_1 \vee K_1\varphi_2)$ is valid in $\mathcal{M}_1^{rst}$.)

(b) Show that for every formula in $\mathcal{L}_1$ of the form $K_1\varphi$ we can effectively find an equivalent formula that is a Boolean combination of formulas of the form $K_1\psi$, where ψ is propositional.

3.24 Extend Proposition 3.2.1 to deal with formulas in the language $\mathcal{L}_n^{CD}$. (We remark that once we have common knowledge in the language, the algorithm will no longer run in time $O(||M|| \times |\varphi|)$, but will still run in time polynomial in $||M||$ and $|\varphi|$.)

** **3.25** In this exercise, we sketch the details of how to construct effectively the proof of a valid formula. (By "construct effectively," we mean that there is an algorithm that takes as input a formula φ and gives as output a proof of φ, if φ is valid, and halts, say with the output "not valid," if φ is not valid.) We work with K_n here, but the proof can be easily modified to deal with all the other logics we have been considering. Using the notation of Theorem 3.2.2, let $Sub^+(\varphi)$ consist of all the subformulas of φ and their negations. Let $\mathcal{V}$ consist of all subsets V of $Sub^+(\varphi)$ such that (a) $\psi \in V$

iff $\neg\psi \notin V$ for each subformula ψ of φ and (b) $\psi \wedge \psi' \in V$ iff $\psi, \psi' \in V$. Let $M^0 = (S^0, \pi^0, \mathcal{K}_1^0, \ldots, \mathcal{K}_n^0)$, where $S^0 = \{s_V \mid V \in \mathcal{V}\}$ and $\pi^0, \mathcal{K}_1^0, \ldots, \mathcal{K}_n^0$ are constructed as in the proof of Theorem 3.1.3. We construct inductively, for $k = 0, 1, 2, \ldots$, a sequence of structures $M^k = (S^k, \pi^k, \mathcal{K}_1^k, \ldots, \mathcal{K}_n^k)$. Suppose we have already constructed M^k. Let S^{k+1} consist of those states $s_V \in S^k$ such that if there is a formula of the form $\neg K_i\psi \in V$, then there is a state $s_W \in S^k$ such that $(s_V, s_W) \in \mathcal{K}_i^k$ and $\neg\psi \in W$. Let $\pi^{k+1}, \mathcal{K}_1^{k+1}, \ldots, \mathcal{K}_n^{k+1}$ be the restrictions of $\pi^k, \mathcal{K}_1^k, \ldots, \mathcal{K}_1^k$ to S^{k+1}. Note that this construction is effective. Moreover, since $|S^0| \leq 2^{|\varphi|}$ and $S^{k+1} \subseteq S^k$, there must be some point, say k_0, such that $S^{k_0+1} = S^{k_0}$.

(a) Prove that φ is satisfiable iff for some state $s_V \in S^{k_0}$ we have $\varphi \in V$. (Note that this gives us another proof that if φ is satisfiable, it is satisfiable in a finite structure.)

(b) Let φ_V be the conjunction of all the formulas in the set V. Prove that if $s_V \in (S^k - S^{k+1})$, then $\mathrm{K}_n \vdash \neg\varphi_V$; moreover, show that the proof of $\neg\varphi_V$ can be effectively constructed. (Hint: show by induction on k that $\mathrm{K}_n \vdash \bigvee_{s_V \in S^k} \varphi_V$, and that the proof of $\bigvee_{s_V \in S^k} \varphi_V$ can be constructed effectively.)

(c) Note that if φ is valid, then if we apply the previous construction to $\neg\varphi$, eventually we eliminate every state s_V such that $\neg\varphi \in V$. Use this observation and parts (a) and (b) to show that we can effectively construct a proof of φ.

3.26 Complete the details of the proof of Theorem 3.2.4.

3.27 In the proof of Theorem 3.3.1, show that $|S_\varphi| \leq 2^{|\varphi|}$. (Hint: recall that $|C_G\varphi| = 2 + 2|G| + |\varphi|$.)

* **3.28** In this exercise, we fill in some of the details of the proof of Claim 3.1 in the proof of Theorem 3.3.1. Assume that $(M_\varphi, s_V) \models C_G\psi$. Let $\mathcal{W}$ be as in the proof of Theorem 3.3.1. We wish to prove that

$$\mathrm{K}_n^C \vdash \varphi_{\mathcal{W}} \Rightarrow E_G(\psi \wedge \varphi_{\mathcal{W}}).$$

(a) Prove that if $i \in G$ and $W \in \mathcal{W}$, then $\mathrm{K}_n^C \vdash \varphi_W \Rightarrow K_i\psi$. (Hint: assume that $W/K_i = \{\varphi_1, \ldots, \varphi_k\}$. Use an argument like that in the proof of Theorem 3.1.3, where W here plays the role that V plays there, and use the fact that $(M_\varphi, s_W) \models K_i\psi$, to show that

$$\mathrm{K}_n^C \vdash K_i\varphi_1 \Rightarrow \big(K_i\varphi_2 \Rightarrow (\ldots \Rightarrow (K_i\varphi_k \Rightarrow K_i\psi)\ldots)\big).$$

Now use the fact that $K_i\varphi_j \in W$, for $j = 1, \ldots, k$.)

(b) Define $\overline{\mathcal{W}}$ to be $Con_C(\varphi) - \mathcal{W}$. Show that if $i \in G$, $W \in \mathcal{W}$, and $W' \in \overline{\mathcal{W}}$, then $\mathrm{K}_n^C \vdash \varphi_W \Rightarrow K_i \neg \varphi_{W'}$. (Hint: by definition of $\mathcal{W}$, show that $(M_\varphi, s_W) \models C_G \psi$ and $(M_\varphi, s_{W'}) \not\models C_G \psi$. Conclude that $s_{W'}$ is not G-reachable from s_W and, in particular, $(s_W, s_{W'}) \notin \mathcal{K}_i$. By definition of $\mathcal{K}_i$, conclude that $W/K_i \not\subseteq W'$, so there is a formula ψ' such that $K_i \psi' \in W$ and $\psi' \notin W'$. Since $\psi' \notin W'$, show that $\mathrm{K}_n^C \vdash \psi' \Rightarrow \neg \varphi_{W'}$. From this, show $\mathrm{K}_n^C \vdash K_i \psi' \Rightarrow K_i \neg \varphi_{W'}$. Since $K_i \psi' \in W$, conclude that $\mathrm{K}_n^C \vdash \varphi_W \Rightarrow K_i \neg \varphi_{W'}$.)

(c) Conclude from parts (a) and (b) that

$$\mathrm{K}_n^C \vdash \varphi_W \Rightarrow K_i \Big(\psi \wedge \Big(\bigwedge_{W' \in \overline{\mathcal{W}}} \neg \varphi_{W'} \Big) \Big).$$

(d) Prove that

$$\mathrm{K}_n^C \vdash \varphi_{\mathcal{W}} \Leftrightarrow \Big(\bigwedge_{W' \in \overline{\mathcal{W}}} \neg \varphi_{W'} \Big).$$

(Hint: first show that $\mathrm{K}_n^C \vdash \bigvee_{W \in Con_C(\varphi)} \varphi_W$.)

(e) Use parts (c) and (d) to show that $\mathrm{K}_n^C \vdash \varphi_W \Rightarrow K_i(\psi \wedge \varphi_{\mathcal{W}})$.

(f) Conclude from part (e) that $\mathrm{K}_n^C \vdash \varphi_{\mathcal{W}} \Rightarrow E_G(\psi \wedge \varphi_{\mathcal{W}})$.

* **3.29** This exercise provides a weak form of the deduction theorem for languages with common knowledge. Let $G = \{1, \ldots, n\}$. Show that if $\mathrm{K}_n^C, \varphi \vdash \psi$, then $\mathrm{K}_n^C \vdash C_G \varphi \Rightarrow C_G \psi$. Observe that similar results hold for T_n^C, $\mathrm{S4}_n^C$, $\mathrm{S5}_n^C$, and $\mathrm{KD45}_n^C$.

* **3.30** In this exercise, we fill in some details of the proof of Theorem 3.4.1.

(a) Show that $\mathcal{K}_G \subseteq \bigcap_{i \in G} \mathcal{K}_i$.

(b) Construct an example where $\mathcal{K}_G \neq \bigcap_{i \in G} \mathcal{K}_i$.

(c) Show that the canonical structure (or any other structure for that matter) can be *unwound* to get a structure whose graph looks like a tree, in such a way that the same formulas are true in corresponding states. (More formally, given a structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, there is another structure $M' = (S', \pi', \mathcal{K}'_1, \ldots, \mathcal{K}'_n)$ and a function $f : S' \to S$ such that (i) the graph of M' looks like a tree, in that for all states s', t' in M', there is at most one

path from s' to t', and no path from s' back to itself, (ii) if $(s', t') \in \mathcal{K}_i'$ then $(f(s'), f(t')) \in \mathcal{K}_i$, (iii) $\pi'(s') = \pi(f(s'))$, and (iv) f is onto, so that for all $s \in S$ there exists $s' \in S'$ such that $f(s') = s$. Moreover, we have $(M', s') \models \varphi$ iff $(M, f(s')) \models \varphi$ for all states $s' \in S'$ and all formulas φ.

(d) Show that we can unwind the canonical structure in such a way as to get a structure M' where $\mathcal{K}_G = \bigcap_{i \in G} \mathcal{K}_i$.

3.31 Prove syntactically (i.e., from the axioms) that knowledge distributes over conjunctions. That is, prove the following:

(a) $\mathrm{K}_n \vdash K_i(\varphi \wedge \psi) \Leftrightarrow (K_i\varphi \wedge K_i\psi)$ (hint: use the observation that $\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\varphi_1 \wedge \varphi_2))$ is a propositional tautology),

(b) $\mathrm{K}_n^C \vdash E_G(\varphi \wedge \psi) \Leftrightarrow (E_G\varphi \wedge E_G\psi)$,

(c) $\mathrm{K}_n^C \vdash C_G(\varphi \wedge \psi) \Leftrightarrow (C_G\varphi \wedge C_G\psi)$,

(d) $\mathrm{K}_n^D \vdash D_G(\varphi \wedge \psi) \Leftrightarrow (D_G\varphi \wedge D_G\psi)$.

3.32 In Chapter 2, we said that distributed knowledge could be viewed as the knowledge the agents would have by pooling their individual knowledge together. This suggests the following inference rule:

RD1. From $(\psi_1 \wedge \ldots \wedge \psi_k) \Rightarrow \varphi$ infer $(K_{i_1}\psi_1 \wedge \ldots \wedge K_{i_k}\psi_k) \Rightarrow D_G\varphi$, for $G = \{i_1, \ldots, i_k\}$.

Intuitively, RD1 says that if $\psi = \psi_1 \wedge \ldots \wedge \psi_k$ implies φ, and if each of the agents in G knows a "part" of ψ (in particular, agent i_j knows ψ_j), then together they have distributed knowledge of ψ, and thus distributed knowledge of φ.

(a) Prove that RD1 preserves validity with respect to $\mathcal{M}_n$.

(b) Show that RD1 is derivable from axiom A2 (with D_G substituted for K_i), D1, and D2, using propositional reasoning. (Hint: you will also need the results of Exercise 3.31.)

3.33 We say that φ is a *pure knowledge formula* if φ is a Boolean combination of formulas of the form $K_i\psi$ (that is, it is formed from formulas of the form $K_i\psi$ using $\wedge$, $\neg$, and $\vee$). For example, $K_2p \vee (K_1\neg K_3p \wedge \neg K_2\neg p)$ is a pure knowledge formula, but $p \wedge \neg K_1p$ is not. Show that if φ is a pure knowledge formula, then $\mathrm{K}_n^D \vdash \varphi \Rightarrow D_G\varphi$.

3.34 Fill in the details of the proofs of Proposition 3.6.3 and Theorem 3.6.4.

3.35 Prove analogues to Proposition 3.6.3 and Theorem 3.6.4 for K45. (Hint: use Exercise 3.22.)

* **3.36** Show that $K_1\varphi$ is S4-consistent iff $K_1\varphi$ is S5-consistent. Conclude that the satisfiability problem for S4 for formulas of the form $K_1\varphi$ is *NP*-complete (although the general satisfiability problem for S4 is *PSPACE*-complete).

* **3.37** In this exercise, we show that first-order logic is, in a precise sense, expressive enough to capture propositional modal logic (without common knowledge). Given a set Φ of primitive propositions, let the vocabulary Φ^* consist of a unary predicate P corresponding to each primitive proposition p in Φ, as well as binary predicates $R_1, \ldots, R_n$, one for each agent. We now define a translation from formulas in $\mathcal{L}_n(\Phi)$ to first-order formulas over Φ^*, so that for every formula $\varphi \in \mathcal{L}_n(\Phi)$ there is a corresponding first-order formula φ^* with one free variable x:

- $p^* = P(x)$ for a primitive proposition p
- $(\neg\varphi)^* = \neg(\varphi^*)$
- $(\varphi \wedge \psi)^* = \varphi^* \wedge \psi^*$
- $(K_i\varphi)^* = \forall y(R_i(x, y) \Rightarrow \varphi^*(y))$, where y is a new variable not appearing in φ^* and $\varphi^*(y)$ is the result of replacing all occurrences of x in φ^* by y.

Next, we provide a mapping from a Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n) \in \mathcal{M}_n(\Phi)$ to a relational Φ^*-structure M^*. The domain of M^* is S. For each primitive proposition $p \in \Phi$, we let $P^{M^*} = \{s \in S : \pi(s)(p) = \textbf{true}\}$, and let $R_i^{M^*} = \mathcal{K}_i$.

(a) Show that $(M, s) \models \varphi$ iff $(M^*, V) \models \varphi^*(x)$, where $V(x) = s$. Intuitively, this says that φ^* is true of exactly the domain elements corresponding to states s for which $(M, s) \models \varphi$.

(b) Show that φ is valid with respect to $\mathcal{M}_n(\Phi)$ iff $\forall x\varphi^*(x)$ is a valid first-order formula. (Hint: use the fact that the mapping from structures in $\mathcal{M}_n(\Phi)$ to relational Φ^*-structures is invertible.)

(c) Show how to modify this construction to capture validity with respect to structures in $\mathcal{M}_n^r$ (resp., $\mathcal{M}_n^{rt}$, $\mathcal{M}_n^{rst}$, $\mathcal{M}_n^{elt}$).

Given this translation, we might wonder why we should consider propositional modal logic at all. There are four main reasons for this. First, the syntax of modal logic allows us to more directly capture the types of statements regarding knowledge that we typically want to make. Second, the semantics of modal logic in terms of possible-worlds structures better represents our intuitions (and, as we shall see, directly corresponds to a standard representation of a system, one of our major application areas). Third, the translation fails for common knowledge. (That is, there is no first-order formula corresponding to common knowledge. This follows from the fact that transitive closure cannot be expressed in first-order logic.) Finally, by moving to first-order logic, we lose the nice complexity properties that we have for propositional modal logic. First-order logic is undecidable; there is no algorithm that can effectively decide whether a first-order formula is valid.

3.38 Show that $(\mathcal{A}, V) \models \forall x\varphi$ iff $(\mathcal{A}, V[x/a]) \models \varphi$ for every $a \in dom(\mathcal{A})$.

3.39 Inductively define what it means for an occurrence of a variable x to be free in a formula as follows:

- if φ is an atomic formula ($P(t_1, \ldots, t_k)$ or $t_1 = t_2$), then every occurrence of x in φ is free,
- an occurrence of x is free in $\neg\varphi$ iff the corresponding occurrence of x is free in φ,
- an occurrence of x is free in $\varphi_1 \wedge \varphi_2$ iff the corresponding occurrence of x in φ_1 or φ_2 is free,
- an occurrence of x is free in $\exists y\varphi$ iff the corresponding occurrence of x is free in φ and x is different from y.

A *sentence* is a formula in which no occurrences of variables are free.

(a) Show that if φ is a formula, and V and V' are valuations that agree on all of the variables that are free in φ, then $(\mathcal{A}, V) \models \varphi$ iff $(\mathcal{A}, V') \models \varphi$.

(b) Show that if φ is a sentence, and V and V' are valuations on the structure $\mathcal{A}$, then $(\mathcal{A}, V) \models \varphi$ iff $(\mathcal{A}, V') \models \varphi$.

3.40 In this exercise, we consider a semantics without any assumptions whatsoever about relationships between domains of worlds within a relational Kripke structure. For simplicity, we assume that there are no function symbols. Given a relational

Kripke structure M, we now take a valuation V on M to be a mapping from variables to the union of the domains at the states of M. Then, as we saw, to define what it means for a formula such as $K_i Tall(x)$ to hold at a state s of a relational Kripke structure M under a valuation V such that $V(x) = Bill$, we may have to decide if $Tall(x)$ is true at a state t such that *Bill* is not in the domain of the relational structure $\pi(t)$. One solution to this problem is to note that if *Bill* is not in the domain of $\pi(t)$, then certainly $Bill \notin Tall^{\pi(t)}$. Therefore, we define a new semantics where we simply say $(M, t, V) \not\models Tall(x)$ if $V(x)$ is not in the domain of $\pi(t)$. Similarly, we say $(M, t, V) \not\models x = y$ if $V(x)$ or $V(y)$ is not in the domain of $\pi(t)$. Further, we modify our standard semantics by saying $(M, s, V) \models \exists x\varphi$ iff $(M, s, V[x/a]) \models \varphi$ for some $a \in dom(\pi(s))$. Although this semantics has some attractive features, it has its problems, as we now show.

The *universal closure* of a formula φ is the formula $\forall x_1 \ldots \forall x_k \varphi$, where $x_1, \ldots, x_k$ are all of the variables that occur free in φ. In first-order logic, it is easy to see that a formula is valid if and only if its universal closure is valid. The next two parts of the exercise, however, show that this is not the case for the semantics of this exercise.

(a) Show that $\forall x(x = x)$ is valid under the semantics of this exercise.

(b) Show that $x = x$ is not valid under the semantics of this exercise.

The fact that $x = x$ is not valid in this semantics is certainly undesirable. Of course, the formula $x = x$ is not a sentence. The next part of this exercise gives an example of a sentence that is not valid in this logic, which we might hope would be valid, namely, the universal closure of the Knowledge of Equality axiom.

(c) Show that the formula $\forall x K_i(x = x)$ is not valid.

The failure of formulas such as those in (b) and (c) to be valid have led most researchers to reject this semantics as a general solution.

We might hope to solve this problem by redefining $(M, t, V) \models (x = y)$ iff $V(x) = V(y)$, irrespective of whether x or y is in domain of $\pi(t)$. While this change "solves" the problems of (b) and (c), other problems remain.

(d) Show that under this redefinition, neither $\exists y(x = y)$ nor $\forall x K_i \exists y(x = y)$ is valid.

The problems that arise in part (d) are due to the fact that $\exists x\varphi$ is true at s if φ holds for some a in $dom(\pi(s))$. We could solve this problem by taking $\exists x\varphi$ to hold if φ holds for any a in the union of the domains at all states. This indeed solves all the problems we have raised but effectively puts us back in the common-domain setting. The semantics is now equivalent to that which would be obtained by taking the same domain at all states, namely, the union of all the domains.

3.41 Show that K_n is sound for relational Kripke structures with n agents. Show that if each $\mathcal{K}_i$ is an equivalence relation, then $S5_n$ is sound.

3.42 In this exercise, we consider when axioms (3.3) and (3.4) from Section 3.7.4 are valid.

(a) Show that both axioms are valid with respect to relational structures.

(b) We say that a constant, function, or relation symbol is a *rigid designator* if it takes on the same value in every state. We say that a term is a rigid designator if all the constant and function symbols that appear in it are rigid designators. Show that both axioms are valid with respect to relational Kripke structures if t, t_1, and t_2 are rigid designators.

We remark that in certain applications it may be useful to designate some of the symbols as rigid designators, while others are allowed to vary. For example, we may want the interpretation of constants such as 0 and 1 and of functions such as $+$ and $\times$ to be independent of the state.

* **3.43** In this exercise, we consider the Barcan formula.

(a) Show that the Barcan formula is valid.

(b) Show that this axiom is a consequence of the axioms and rules of $S5_n$ together with the axioms and rules of first-order logic. (Hint: first-order logic has analogues to the Distribution Axiom A2 and the Knowledge Generalization Rule R2 for universal quantification: $(\forall x\varphi \wedge \forall x(\varphi \Rightarrow \psi)) \Rightarrow \forall x\psi$ is an axiom, and "from φ infer $\forall x\varphi$" is an inference rule. In addition, there is the following axiom:

$$\varphi \Rightarrow \forall x\varphi \text{ if } \varphi \text{ has no free occurrences of } x.$$

Using these, the restricted version of (3.4), and the axioms of $S5_n$, prove

$$\neg K_i \neg \forall x_1 \ldots \forall x_k K_i \varphi \Rightarrow \forall x_1 \ldots \forall x_k \varphi.$$

Then show that, using the axioms and rules of $S5_n$, from $\neg K_i \neg \psi_1 \Rightarrow \psi_2$ we can prove $\psi_1 \Rightarrow K_i \psi_2$.)

3.44 Show that under the domain-inclusion assumption

(a) the Barcan formula is not valid,

(b) the converse of the Barcan formula, namely

$$K_i \forall x_1 \ldots \forall x_k \varphi \Rightarrow \forall x_1 \ldots \forall x_k K_i \varphi,$$

is valid,

(c) if the $\mathcal{K}_i$ relations are equivalence relations, then the Barcan formula is valid.

3.45 In this exercise, we consider the Knowledge of Inequality axiom.

(a) Show that the Knowledge of Inequality axiom (3.8) is valid.

(b) Show that this axiom is a consequence of the axioms and rules of $S5_n$ together with the axioms and rules of first-order logic. (Hint: show that $K_i(\varphi \Rightarrow K_i\varphi) \Rightarrow K_i(\neg\varphi \Rightarrow K_i\neg\varphi)$ is valid in $S5_n$, and hence provable in $S5_n$. Now take φ to be $x_1 = x_2$, and apply Knowledge of Equality.)

Notes

A discussion of different varieties of modal logic can be found in some of the standard texts in the area, such as [Hughes and Cresswell 1968] and [Chellas 1980]. The historical names S4 and S5 are due to Lewis, and are discussed in his book with Langford [1959]. The names K and T are due to Lemmon [1977], as is the idea of naming the logic for the significant axioms used. Arguments for using logics weaker than S5 in game theory can be found in, for example, [Samet 1987] and [Geanakopolos 1989].

The treatment of completeness and complexity issues in this chapter largely follows that of [Halpern and Moses 1992]. The technique for proving completeness using canonical structures seems to have been worked out independently by Makinson [1966], Kaplan [1966], and Lemmon and/or Scott [Lemmon 1977]. An algebraic approach to the semantics of modal logic is described by Lemmon [1977]. Frames were introduced by Lemmon and Scott [Lemmon 1977], who called them "world systems." The term "frame" is due to Segerberg [1968]. The idea of using frames to characterize axiom systems (as in Exercise 3.17) is well known in modal logic; it appears, for example, in [Goldblatt 1992] and [Hughes and Cresswell 1984].

Although we restrict our attention in this book to languages $\mathcal{L}$ with a countable set of formulas, this is not really necessary. For example, we make this restriction in

Lemma 3.1.2 only to simplify the proof. Indeed, Lemma 3.1.2 is a standard result in the model-theoretic literature and is known as *Lindenbaum's Theorem* [Chang and Keisler 1990, Proposition 1.3.11].

As we mentioned in the notes to Chapter 1, Lenzen's overview article [1978] has a good discussion and review of philosophers' arguments for and against various axioms of knowledge. In the next chapter we present our model of knowledge in multi-agent systems for which $S5_n$ is an appropriate axiomatization. Other axiom systems for knowledge have been used in various contexts. Moore [1985] uses S4 in his theory of knowledge and action. Since the knowledge represented in a knowledge base is typically not required to be true, axiom A3 has been thought inappropriate for these applications; thus KD45 is considered, for example, by Levesque [1984a]. KD45 has also been considered, for example, by Fagin and Halpern [1988a] and by Levesque [1984b], to be an appropriate logic for characterizing the beliefs of an agent, who might believe things that in fact turn out to be false.

There has been a great deal of interest recently in having a system with modal operators for knowledge and belief where, typically, the belief operator satisfies the axioms of KD45 and the knowledge operator satisfies the axioms of S5. The focus has been on the interaction between these operators (for example, if agent i believes φ, does she know that she believes φ?) and on defining belief in terms of knowledge. Further details can be found in [Friedman and Halpern 1994], [Kraus and Lehmann 1988], [Moses and Shoham 1993], and [Voorbraak 1992].

The formula $K_i(\varphi \wedge \neg K_i\varphi)$ discussed in part (c) of Exercise 3.10 has been called a "pragmatically paradoxical formula." It was first introduced by Moore (see [Hintikka 1962]).

Axioms for common knowledge appear in [Lehmann 1984], [Milgrom 1981], and [McCarthy, Sato, Hayashi, and Igarishi 1979]. (In these papers, only the modal operator C, referring to common knowledge of all the agents in the system, was used, rather than the indexed modal operator C_G.) The essential ideas for extending the canonical structure technique to languages including common knowledge are due to Kozen and Parikh [1981], who proved completeness results for the logic *PDL* (Propositional Dynamic Logic) in this way. The idea for proving completeness for the language including distributed knowledge is due to Halpern and Moses [1992]; a formal completeness proof (as in Exercise 3.30) can be found in [Fagin, Halpern, and Vardi 1992a] and [Hoek and Meyer 1992].

An excellent introduction to complexity theory is given by Hopcroft and Ullman [1979]. The fact that satisfiability for propositional logic is *NP*-complete was proved by Cook [1971], who in fact introduced the notions of *NP* and *NP*-completeness. Ladner [1977] proved that the satisfiability problem for S5 is *NP*-complete, and that

satisfiability for the logics K, T, and S4 is *PSPACE*-complete. The results in the multi-agent case are from [Halpern and Moses 1992]. The exponential time results for logics involving common knowledge are based on similar results for PDL. The lower bound for PDL is due to Fischer and Ladner [1979]; the matching upper bound is due to Pratt [1979]. Details of the proofs of the complexity results not included here can be found in [Halpern and Moses 1992]. A general framework for studying the complexity of modal logics is described by Vardi [1989].

An excellent introduction to first-order logic is [Enderton 1972]; this book also provides a nice discussion of issues of decidability and undecidability. The translation from modal logic to first-order logic (Exercise 3.37) is another notion that seems to have been developed independently by a number of people. The first treatment of these ideas in print seems to be due to van Benthem [1974]; details and further discussion can be found in his book [1983]. The distinction between "knowing that" and "knowing who" is related to an old and somewhat murky distinction between knowledge *de dicto* (literally, "knowledge of words") and knowledge *de re* (literally, "knowledge of things"). Plantinga [1974] discusses these terms in more detail.

The example of the morning star and the evening star is due to Frege [1892], and its implications for first-order modal logic were first discussed by Quine [1947]. The idea of dealing with the morning-star paradox by restricting substitutions so that they are not allowed within the scope of knowledge operators K_i is due to Kanger [1957a], and the idea of using rigid designators is due to Kaplan [1969].

The Barcan formula (or actually, a formula equivalent to it) was introduced by Barcan [1946]. Prior [1956] showed that the Barcan formula is a consequence of the axioms of first-order logic, along with S5; see [Hughes and Cresswell 1968, page 145] for a proof. Prior [1957] also made an early objection to it. Kripke [1963b] introduced structures equivalent to relational Kripke structures, but where the domains of distinct worlds can be unrelated, and so the Barcan formula is violated. He gave a completeness proof for a first-order modal logic in the S5 case [1959]. Barcan [1947] showed the validity of the Knowledge of Equality axiom.

Detailed discussions of first-order modal logic, along with completeness proofs, appear in Hughes and Cresswell's book [1968] and Garson's article [1977]. Much of the information we have given about first-order modal logic (including bibliographic references) is from Hughes and Cresswell's book. They, along with Garson, discuss and prove sound and complete axiomatizations under a variety of assumptions, including cases where formulas involving equality are not allowed Garson discusses in detail a number of ways of dealing with what is called the problem of "quantifying-in": how to give semantics to a formula such as $\exists x K_i(P(x))$ without the common domain assumption.

An axiomatization of first-order logic that includes (3.3) and a slightly stronger version of (3.4) appear in [Enderton 1972]. This stronger version says that if φ' is the result of replacing some occurrences of t_1 in $\varphi(t_1)$ by t_2, then $(t_1 = t_2) \Rightarrow (\varphi(t_1) \Leftrightarrow \varphi(t_2))$. It is not hard to show that in the presence of the other axioms, this stronger version is implied by our (3.4). We remark that in [Enderton 1972], the Rule of Universal Generalization ("from φ infer $\forall x\varphi$") is not used. Instead, all axioms are viewed as universally quantified. We do assume this rule here. (Alternatively, we would have to universally quantify all the free variables in the axioms of K_n or $S5_n$.)

Chapter 4

Knowledge in Multi-Agent Systems

The following four propositions, which appear to the author to be incapable of formal proof, are presented as Fundamental Postulates upon which the entire superstructure of General Systemantics ... is based ...

1. *EVERYTHING IS A SYSTEM.*
2. *EVERYTHING IS PART OF A LARGER SYSTEM.*
3. *THE UNIVERSE IS INFINITELY SYSTEMATIZABLE, BOTH UPWARD (LARGER SYSTEMS) AND DOWNWARD (SMALLER SYSTEMS).*
4. *ALL SYSTEMS ARE INFINITELY COMPLEX. (The illusion of simplicity comes from focusing attention on one or a few variables.)*

John Gall, *Systemantics*, 1975

4.1 Runs and Systems

In this chapter we consider one of the major application areas of reasoning about knowledge: multi-agent systems. We subscribe to the spirit of the quote above: in this book, we view any collection of interacting agents as a multi-agent system. For example, we shall view the children (and the father) in the muddy children puzzle as agents in a multi-agent system. We also want to be able to model a game such as poker as a multi-agent system. A distributed system consisting of processes in a computer network running a particular protocol forms another example of a multi-agent system. Although we typically call the entities in a multi-agent system

"agents," we occasionally refer to them as "players" (particularly in the context of game-theoretic examples) and "processes" or "sites" (particularly in the context of distributed systems examples).

A good case can be made that, as suggested in the quote at the beginning of the chapter, systems are extremely complex. It is hard enough to reason about the behavior of one agent. If we have a system of interacting agents, things get much worse. Consider, for example, the muddy children puzzle. If we attempt to model the system in full detail, we would have to include all that happened to each of the children throughout their lives, a detailed description of their visual and auditory systems and how they operate, details of the weather, etc.; the list is potentially endless. All of these factors could, in principle, influence the behavior of the children.

The first step in dealing with the complexity is also suggested by the quote: we focus attention on only a few of the details, and hope that these cover everything that is relevant to our analysis. The second step consists of finding good ways to think about a situation in order to minimize its complexity. Our goal in this chapter (and in much of the rest of the book) is to show that reasoning about systems in terms of knowledge can be very helpful in this regard. To do that, we need a formal model of multi-agent systems. We want a framework that is general enough to allow us to capture all the important features of multi-agent systems, without getting too bogged down in details.

One key assumption we make is that if we look at the system at any point in time, each of the agents is in some *state*. We refer to this as the agent's *local* state, in order to distinguish it from a *global* state, which we define shortly. We assume that an agent's local state encapsulates all the information to which the agent has access. In our abstract framework, we do not make any additional assumptions about the state. In the case of the muddy children, the state of a child might encode what the child has seen and heard, that is, which of the other children have muddy foreheads and which do not, the father's initial statement, and the responses of each of the children to the father's questions so far. If we are modeling a poker game, a player's state might consist of the cards he currently holds, the bets made by the other players, any other cards he has seen, and any information he may have about the strategies of the other players (for example, Bob may know that Alice likes to bluff, while Charlie tends to bet conservatively).

As this example already indicates, representing the states of the agents can be highly nontrivial. The first problem is deciding what to include in the state. Certainly if the system is made up of interacting people, then it becomes a rather difficult problem to decide where to draw the line. In the poker example, should we include the fact that Bob had an unhappy childhood as part of his state? If so, how do we

capture this? Once we have solved the problem of what to include in the state, we then have to decide how to *represent* what we do include. If we decide that Bob's childhood is relevant, how do we describe the relevant features of his childhood in a reasonable way? In our abstract framework we sidestep these difficulties, and simply assume that at each point in time, each agent in the system is in some unique state. Of course, we do have to confront these difficulties when dealing with concrete examples. These problems tend to be somewhat easier to solve when dealing with processes in a distributed system rather than people, but, as we shall soon see, even in this simpler setting there can be difficult choices to make.

Once we think in terms of each agent having a state, it is but a short step to think of the whole system as being in some state. The first thought might be to make the system's state be a tuple of the form $(s_1, \ldots, s_n)$, where s_i is agent i's state. But, in general, more than just the local states of the agents may be relevant when analyzing a system. If we are analyzing a message-passing system where processes send messages back and forth along communication lines, we might want to know about messages that are in transit or about whether a communication line is up or down. If we are considering a system of sensors observing some terrain, we might need to include features of the terrain in a description of the state of the system.

Motivated by these observations, we conceptually divide a system into two components: the agents and the *environment*, where we view the environment as "everything else that is relevant." In many ways the environment can be viewed as just another agent, though it typically plays a special role in our analyses. We define a *global state* of a system with n agents to be an $(n + 1)$-tuple of the form $(s_e, s_1, \ldots, s_n)$, where s_e is the state of the environment and s_i is the local state of agent i.

A given system can be modeled in many ways. How we divide the system into agents and environment depends on the system being analyzed. In a message-passing system, we can view a message buffer, which stores messages not yet delivered, either as a process (i.e., an agent), and have its state encode which messages have been sent and not yet delivered, or as part of the environment. Similarly, we can view a communication line as an agent whose local state might describe (among other things) whether or not it is up, or we can have the status of the communication lines be part of the environment.

A global state describes the system at a given point in time. But a system is not a static entity; it constantly changes. Since we are mainly interested in how systems change over time, we need to build time into our model. We define a *run* to be a function from time to global states. Intuitively, a run is a complete description of how the system's global state evolves over time. In this book, we take time to range

over the natural numbers. Thus, $r(0)$ describes the initial global state of the system in a possible execution r, the next global state is $r(1)$, and so on.

Our assumption that time ranges over the natural numbers seems to be quite a strong one. In particular, it means that time steps are discrete and that time is infinite. We have made this choice mainly for definiteness, but also because it seems appropriate for many of our applications. Most of our results and comments hold with little or no change if we assume instead that time is continuous (and ranges over, say, the real numbers or the nonnegative real numbers), or if we assume that time is finite. Although we typically think of time as being continuous, assuming that time is discrete is quite natural. Computers proceed in discrete time steps, after all. Even when analyzing situations involving human agents, we can often usefully imagine that the relevant actions are performed at discrete time instances, as in the case of the muddy children puzzle. Allowing time to be infinite makes it easier to model situations where there is no *a priori* time bound on how long the system will run. The muddy children puzzle again provides an example of this phenomenon, since when we start to analyze the puzzle, it is not clear how many steps it will take the children to figure out whether they have mud on their forehead; indeed, in some variants of the puzzle, they never figure it out. And if we do want to model a system that runs for a bounded number of steps, we can typically capture this by assuming that the system remains in the same global state after it has stopped.

We assume that time is measured on some clock external to the system. We do *not* assume that agents in the system necessarily have access to this clock; at time m measured on the external clock, agent i need not know it is time m. If an agent does know the time, then this information would be encoded in his local state (we return to this issue later). This external clock need not measure "real time." For example, in the case of the muddy children puzzle, there could be one "tick" of the clock for every round of questions by the father and every round of answers to the father's question. If we are analyzing a poker game, there could be one tick of the clock each time someone bets or discards. In general, we model the external clock in whatever way makes it easiest for us to analyze the system.

A system can have many possible runs, since the system's global state can evolve in many possible ways: there are a number of possible initial states and many things that could happen from each initial global state. For example, in a poker game, the initial global states could describe the possible deals of the hand, with player i's local state s_i describing the cards held initially by player i. For each fixed deal of the cards, there may still be many possible betting (and discarding) sequences, and thus many runs. In a message-passing system, a particular message may or may not be lost, so again, even with a fixed initial global state, there are many possible runs.

To capture this, we formally define a *system* to be a *nonempty set of runs*. Notice how this definition abstracts our intuitive view of a system as a collection of interacting agents. Instead of trying to model the system directly, our definition models the possible *behaviors* of the system. The requirement that the set of runs be nonempty captures the intuition that the system we are modeling has *some* behaviors. Our approach lets us use the same formal model to describe systems of great diversity; a computer system and a poker game are modeled similarly. Throughout the book we will use the term *system* in two ways: as the "real-life" collection of interacting agents or as a set of runs. Our precise intention should be clear from the context.

In more detail, we proceed as follows. Let L_e be a set of possible states for the environment and let L_i be a set of possible local states for agent i, for $i = 1, \ldots, n$. We take $\mathcal{G} = L_e \times L_1 \times \cdots \times L_n$ to be the set of global states. A *run over* $\mathcal{G}$ is a function from the time domain—the natural numbers in our case—to $\mathcal{G}$. Thus, a run over $\mathcal{G}$ can be identified with a sequence of global states in $\mathcal{G}$. We refer to a pair (r, m) consisting of a run r and time m as a *point*. If $r(m) = (s_e, s_1, \ldots, s_n)$ is the global state at the point (r, m), we define $r_e(m) = s_e$ and $r_i(m) = s_i$, for $i = 1, \ldots, n$; thus, $r_i(m)$ is agent i's local state at the point (r, m). A *round* takes place between two time points. We define round m in run r to take place between time $m - 1$ and time m. It is often convenient for us to view an agent as performing an *action* during a round. (We discuss actions in detail in Chapter 5.) A *system* $\mathcal{R}$ *over* $\mathcal{G}$ is a set of runs over $\mathcal{G}$. We say that (r, m) is a *point in system* $\mathcal{R}$ if $r \in \mathcal{R}$. In practice, the appropriate set of runs will be chosen by the system designer or the person analyzing the system, both of whom presumably have a model of what this set should be.

The following simple example describes a scenario that we call the *bit-transmission problem*, and to which we shall often return in this chapter as well as in Chapters 5 and 7, should give the reader a better feeling for some of these definitions.

Example 4.1.1 Imagine we have two processes, say a *sender* S and a *receiver* R, that communicate over a communication line. The sender starts with one bit (either 0 or 1) that it wants to communicate to the receiver. Unfortunately, the communication line is faulty, and it may lose messages in either direction in any given round. That is, there is no guarantee that a message sent by either S or R will be received. For simplicity, we assume that a message is either received in the same round that it is sent, or lost altogether. (Since in this example a message may be received in the same round it is sent, we are implicitly assuming that rounds are long enough for a message to be sent and delivered.) We assume that this type of message loss is the only possible faulty behavior in the system. Because of the uncertainty regarding

possible message loss, S sends the bit to R in every round, until S receives a message from R acknowledging receipt of the bit. We call this message from R an *ack* message. R starts sending the *ack* message in the round after it receives the bit. To allow S to stop sending the bit, R continues to send the *ack* repeatedly from then on.

This informal description gives what we call a *protocol* for S and R: it is a specification of what they do at each step. (We discuss protocols in much greater detail in Chapter 5.) The protocol dictates that S must continue sending the bit to R until S receives the *ack* message; roughly speaking, this is because before it receives the *ack* message, S does not know whether R received the bit. On the other hand, in this protocol R never knows for certain that S actually received its acknowledgment. Note the usage of the word "know" in the two previous sentences. This, of course, is not an accident. One of our main claims is that this type of protocol is best thought of in terms of knowledge.

Returning to the protocol, note that R does know perfectly well that S stops sending messages after it receives an *ack* message. But even if R does not receive messages from S for a while, from R's point of view this is not necessarily because S received an *ack* message from R; it could be because the messages that S sent were lost in the communication channel. We could have S send an *ack-ack* message—an acknowledgment to the acknowledgment—so that R could stop sending the acknowledgment once it receives an *ack-ack* message from S. But this only pushes the problem up one level: S will not be able to safely stop sending *ack-ack* messages, since S has no way of knowing that R has received an *ack-ack* message. As we show later in this chapter (Theorem 4.5.4) this type of uncertainty is inherent in systems such as the one we have just described, where communication is not guaranteed. For now, we focus on the protocol that we described, where R continues to send *ack* messages in every round, and S stops as soon as it receives one of them.

The situation that we have just described informally can be formalized as a system. To describe the set of runs that make up this system, we must make a number of choices regarding how to model the local states of S, R, and the environment. It seems reasonable to assume that the value of the bit should be part of S's local state, and it should be part of R's local state as soon as R receives a message from S with the value. Should we include in S's state the number of times that S has sent the bit or the number of times that S receives an *ack* message from R? Similarly, should we include in R's state the number of times R has sent the *ack* message or the number of times R has received the bit from S? Perhaps we should include in the local state a representation of the protocol being used? Our choice is to have the local states of S and R include very little information; essentially, just enough to allow us to carry out our analysis. On the other hand, as we shall see in Example 4.2.1,

it is useful to have the environment's state record the events taking place in the system. Thus, we take L_S, the possible local states of S, to be $\{0, 1, (0, ack), (1, ack)\}$, where, intuitively, S's local state is k if its initial bit is k and it has not received an *ack* message from R, while S's local state is (k, ack) if its initial bit is k and it has received an *ack* message from R, for $k = 0, 1$. Similarly, $L_R = \{\lambda, 0, 1\}$, where λ denotes the local state where R has received no messages from S, and k denotes the local state where R received the message k from S, for $k = 0, 1$. The environment's local state is used to record the history of events taking place in the system. At each round, either (a) S sends the bit to R and R does nothing, (b) S does nothing and R sends an *ack* to S, or (c) both S and R send messages. We denote these three possibilities by (sendbit, Λ), (Λ, sendack), (sendbit, sendack) respectively. Thus, we let the environment's state be a sequence of elements from the set {(sendbit, Λ), (Λ, sendack), (sendbit, sendack)}. Here the m^{th} member of the sequence describes the actions of the sender and receiver in round m.

There are many possible runs in this system, but these runs must all satisfy certain constraints. Initially, the system must start in a global state where nothing has been recorded in the environment's state, neither S nor R has received any messages, and S has an initial bit of either 0 or 1. Thus, the initial global state of every run in the system has the form $(\langle\,\rangle, k, \lambda)$, where $\langle\,\rangle$ is the empty sequence and k is either 0 or 1. In addition, consecutive global states $r(m) = (s_e, s_S, s_R)$ and $r(m+1) = (s'_e, s'_S, s'_R)$ in a run r are related by the following conditions:

- If $s_R = \lambda$, then $s'_S = s_S$, $s'_e = s_e \cdot$ (sendbit, Λ) (where $s_e \cdot$ (sendbit, Λ) is the result of appending (sendbit, Λ) to the sequence s_e), and either $s'_R = \lambda$ or $s'_R = s_S$. (Before R receives a message, it sends no messages; as a result, S receives no message, so it continues to send the bit and its state does not change. R may or may not receive the message sent by S in round $m + 1$.)

- If $s_S = s_R = k$, then $s'_R = k$, $s'_e = s_e \cdot$ (sendbit, sendack), and either $s'_S = k$ or $s'_S = (k, ack)$. (After R has received S's bit, it starts sending acknowledgments, and its state undergoes no further changes. S continues to send the bit, and it may or may not receive the acknowledgment sent by R in round $m + 1$.)

- if $s_S = (k, ack)$, then (a) $s'_e = s_e \cdot (\Lambda,$ sendack), (b) $s'_S = s_S$, and (c) $s'_R = s_R$. (Once S has received R's acknowledgement, S stops sending the bit and R continues to send acknowledgements. The local states of S and R do not change any more.)

We take the system $\mathcal{R}^{bt}$ describing the bit-transmission problem to consist of all the runs meeting the constraints just described. ∎

Example 4.1.1 shows how many choices have to be made in describing a system, even in simple cases. The example also suggests that the process of describing all the runs in a system of interest can be rather tedious. As we said before, getting a good representation of a system can be difficult. The process is far more of an art than a science. We shall return to this point in Chapter 5, where we extend the framework to deal with protocols and programs. This will give us a relatively straightforward way of describing systems in many applications of interest.

4.2 Incorporating Knowledge

We already saw in our discussion of the bit-transmission problem (Example 4.1.1) that we were making statements such as "R does not *know* for certain that S received its acknowledgment." A central thesis of this book is that we often want to think of an agent's actions as depending on her knowledge. Indeed, our framework has been designed so that knowledge can be incorporated in a straightforward way. The basic idea is that a statement such as "R does not know φ" means that, as far as R is concerned, the system could be at a point where φ does not hold. The way we capture that "as far as R is concerned, the system could be at a point where φ does not hold" is closely related to the notion of possible worlds in Kripke structures. We think of R's knowledge as being determined by its local state, so that R cannot distinguish between two points of the system in which it has the same local state, and it can distinguish points in which its local state differs. We now formalize these ideas.

As we shall see, a system can be viewed as a Kripke structure except that we have no function π telling us how to assign truth values to the primitive propositions. (In the terminology of Section 3.1, a system can be viewed as a *frame*.) To view a system as a Kripke structure, we assume that we have a set Φ of primitive propositions, which we can think of as describing basic facts about the system. In the context of distributed systems, these might be such facts as "the value of the variable x is 0," "process 1's initial input was 17," "process 3 sends the message μ in round 5 of this run," or "the system is deadlocked." (For simplicity, we are assuming that we can describe the basic properties of the system adequately using propositional logic; the extension of the framework to use first-order logic is straightforward.) An *interpreted system* $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system over a set $\mathcal{G}$ of global states and π is an interpretation for the propositions in Φ over $\mathcal{G}$, which assigns truth values to the primitive propositions at the global states. Thus, for every $p \in \Phi$ and state $s \in \mathcal{G}$, we have $\pi(s)(p) \in \{\textbf{true}, \textbf{false}\}$. Of course, π induces also an interpretation over

the points of $\mathcal{R}$; simply take $\pi(r, m)$ to be $\pi(r(m))$. Notice that Φ and π are not intrinsic to the system $\mathcal{R}$. They constitute additional structure on top of $\mathcal{R}$ that we, as outside observers, add for our convenience, to help us analyze or understand the system better. We refer to the points and states of the system $\mathcal{R}$ as points and states, respectively, of the interpreted system $\mathcal{I}$. That is, we say that the point (r, m) is in the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ if $r \in \mathcal{R}$, and similarly, we say that $\mathcal{I}$ is a system over state space $\mathcal{G}$ if $\mathcal{R}$ is.

To define knowledge in interpreted systems, we associate with an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ a Kripke structure $M_{\mathcal{I}} = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ in a straightforward way: We simply take S to consist of the points in $\mathcal{I}$, and take $\mathcal{K}_1, \ldots, \mathcal{K}_n$ to be some binary relations on S. Note that there is no possibility relation $\mathcal{K}_e$ for the environment; this is because we are not usually interested in what the environment knows. For the possibility relation $\mathcal{K}_i$ we choose a specific relation, which we now describe. If $s = (s_e, s_1, \ldots, s_n)$ and $s' = (s'_e, s'_1, \ldots, s'_n)$ are two global states in $\mathcal{R}$, then we say that s and s' are *indistinguishable to agent* i, and write $s \sim_i s'$, if i has the same state in both s and s', i.e., if $s_i = s'_i$. We can extend the indistinguishability relation $\sim_i$ to points: we say that two points (r, m) and (r', m') are *indistinguishable to* i, and write $(r, m) \sim_i (r', m')$, if $r(m) \sim_i r'(m')$ (or, equivalently, if $r_i(m) = r'_i(m')$). Clearly $\sim_i$ is an equivalence relation on points. When we speak of knowledge in interpreted systems, we assume that the $\mathcal{K}_i$ relation in $M_{\mathcal{I}}$ is defined by $\sim_i$. Intuitively, agent i considers a state s' possible in a state s if s and s' are indistinguishable to agent i. Thus, the agents' knowledge is completely determined by their local states.

Recall from Chapter 3 that we denote by $\mathcal{L}_n(\Phi)$ the set of formulas obtained by starting with the primitive propositions in Φ, and closing off under conjunction, negation, and the modal operators $K_1, \ldots, K_n$, and that we usually omit the Φ when it is clear from context, writing just $\mathcal{L}_n$. Similarly, we denote by $\mathcal{L}_n^C$, $\mathcal{L}_n^D$, and $\mathcal{L}_n^{CD}$ the languages that result by adding, respectively, the modal operators for common knowledge, distributed knowledge, and both common and distributed knowledge. We can now define what it means for a formula $\varphi \in \mathcal{L}_n^{CD}$ to be true at a point (r, m) in an interpreted system $\mathcal{I}$ by applying the definitions of Chapter 2 to the related Kripke structure $M_{\mathcal{I}}$. Thus, we say that $(\mathcal{I}, r, m) \models \varphi$ exactly if $(M_{\mathcal{I}}, s) \models \varphi$, where $s = (r, m)$. For example, we have

$(\mathcal{I}, r, m) \models p$ (for $p \in \Phi$) iff $\pi(r, m)(p) = \mathbf{true}$, and

$(\mathcal{I}, r, m) \models K_i\varphi$ iff $(\mathcal{I}, r', m') \models \varphi$ for all (r', m') such that $(r, m) \sim_i (r', m')$.

The obvious definition of $\models$ for formulas involving common knowledge and distributed knowledge are left to the reader.

Since π is a function on global states, the truth of a primitive proposition q at a point (r, m) depends only on the global state $r(m)$. This seems like a natural assumption; the global state is meant to capture everything that is relevant about the current situation. Quite often, in fact, the truth of a primitive proposition q of interest depends, not on the whole global state, but only on the component of some particular agent. For example, the truth of a statement such as "process 2 received process 1's message" might depend only on process 2's state. In that case, we expect π to respect the *locality* of q, that is, if $s, s' \in \mathcal{G}$, and $s \sim_i s'$, then $\pi(s)(q) = \pi(s')(q)$.

We can also imagine statements that depend on more than just the global state. Consider, for example, a statement such as "eventually (at some later point in the run) the variable x is set to 5." There could well be two points (r, m) and (r', m') with the same global state, such that this statement is true at (r, m) and false at (r', m'). Thus, such a temporal statement cannot be represented by a primitive proposition in our framework. Indeed, it cannot be represented by any formula in our language; it is easy to see that, for every formula $\varphi \in \mathcal{L}_n^{CD}$, if $r(m) = r'(m')$, then $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}, r', m') \models \varphi$ (Exercise 4.1). While we could deal with this problem by allowing the truth of a primitive proposition to depend on the point, and not just the global state, the more appropriate way to express such temporal statements is to add modal operators for time into the language. We do this in Section 4.3.

In analogy to our previous definitions, we say that φ is *valid in the interpreted system* $\mathcal{I}$ and write $\mathcal{I} \models \varphi$, if $(\mathcal{I}, r, m) \models \varphi$ for all points (r, m) in $\mathcal{I}$. For a class $\mathcal{C}$ of interpreted systems, we say that a formula φ is *valid in* $\mathcal{C}$, and write $\mathcal{C} \models \varphi$, if $\mathcal{I} \models \varphi$ for every interpreted system $\mathcal{I} \in \mathcal{C}$.

We now have a concrete interpretation for knowledge in multi-agent systems. As we said in Chapter 1, this interpretation of knowledge is an *external* one, ascribed to the agents by someone reasoning about the system. We do not assume that the agents compute their knowledge in any way, or that they can necessarily answer questions based on their knowledge. Note that this notion of knowledge satisfies all the S5 properties as described in Chapter 2, since $\sim_i$ is an equivalence relation. In particular, the Distribution Axiom and the Rule of Knowledge Generalization both hold: agents know all logical consequences of their knowledge, and they know all valid formulas. As we observed in Section 2.4, these properties hold in every Kripke structure. In particular, they would hold no matter how we defined the $\mathcal{K}_i$ relation in $M_{\mathcal{I}}$.

Recall that we allow the agents in our system to be processes in a distributed system. It may seem strange to view such inanimate agents as possessing knowledge and, in fact, as being "logically omniscient." Nevertheless, our usage of the word "knowledge" is consistent with at least one way it is used in practice. For example,

when someone analyzing a distributed protocol says "process 2 does not know that process 3 is faulty at the end of round 5 in run r," what is often meant is that there is a point at which process 3 is faulty, which is indistinguishable to process 2 from the point $(r, 5)$.

We shall see many examples throughout the book where this notion of knowledge is useful for analyzing multi-agent systems. There are certainly applications, however, for which the externally ascribed knowledge is inappropriate. For example, later in this chapter we consider an example involving knowledge bases, where it may be more appropriate to consider the knowledge base's *beliefs*, rather than its knowledge. As we shall see, by using a slightly different $\mathcal{K}_i$ relation instead of $\sim_i$, we do get a reasonable notion of belief. Of course, we still have logical omniscience. We explore techniques for dealing with logical omniscience in later chapters of the book. For now we content ourselves with showing how the external notion of knowledge can be applied to better analyze multi-agent systems.

Example 4.2.1 Consider the bit-transmission problem (Example 4.1.1) again. We can take Φ here to consist of six primitive propositions: $bit = 0$, $bit = 1$, *recbit*, *recack*, *sendbit*, and *sendack*, representing the assertions that the value of S's initial bit is 0, the value of S's initial bit is 1, R has received S's message, S has received R's acknowledgment, S has just sent a message, and R has just sent a message, respectively. The appropriate interpreted system is $\mathcal{I}^{bt} = (\mathcal{R}^{bt}, \pi^{bt})$, where $\mathcal{R}^{bt}$ consists of the set of runs described in Example 4.1.1, and π^{bt} is such that

- $(\mathcal{I}^{bt}, r, m) \models bit = k$ exactly if $r_S(m)$ is either k or (k, ack), for $k = 0, 1$,
- $(\mathcal{I}^{bt}, r, m) \models recbit$ if $r_R(m) \neq \lambda$,
- $(\mathcal{I}^{bt}, r, m) \models recack$ if $r_S(m) = (0, ack)$ or $r_S(m) = (1, ack)$,
- $(\mathcal{I}^{bt}, r, m) \models sendbit$ if the last tuple in $r_e(m)$ is either (sendbit, Λ) or (sendbit, sendack), and
- $(\mathcal{I}^{bt}, r, m) \models sendack$ if the last tuple in $r_e(m)$ is either (Λ, sendack) or (sendbit, sendack).

Note that the truth value of all these primitive propositions is completely determined by the global state, since we assumed the environment's state records the events taking place in the system. In fact, it is easy to see that $bit = 0$, $bit = 1$, and *recack* are local to S—they depend only on S's local state—whereas *recbit* is local to R. For the remainder of our discussion in this example, we need only the primitive

propositions $bit = 0$ and $bit = 1$; however, the other primitive propositions will be useful later. Just as the way we choose to model the local states in the system depends on the analysis we plan to carry out, so too does the choice of primitive propositions.

Intuitively, after R receives S's bit, then R *knows* the value of the bit. And indeed, it is easy to check that if (r, m) is a point such that $r_R(m) = k$, for $k \neq \lambda$ (so that R has received S's bit by that point), then $(\mathcal{I}, r, m) \models K_R(bit = k)$. This is because at all other points (r', m'), if $r_R(m) = r'_R(m')$, then S must have initial bit k at (r', m'). Similarly, when S receives R's *ack* message, then S knows that knows the initial bit. More formally, if $r_S(m) = (k, ack)$, then $(\mathcal{I}, r, m) \models K_S K_R(bit = k)$. It is easy to see that, in this setting, if S stops sending messages to R before S knows that R knows the value of the bit, i.e., before either $K_S K_R(bit = 0)$ or $K_S K_R(bit = 1)$ holds, then it is possible that R will never receive the bit. Although we do not provide a formal proof of this fact here, this observation already suggests the power of the knowledge-based approach. It allows us to relate actions, such as sending a message or receiving a message, to states of knowledge, and then use the states of knowledge as a guide to what actions should be performed. We investigate these issues in greater detail in the next few chapters. ∎

4.3 Incorporating Time

As it stands, our language is not expressive enough to handle conveniently the full complexity of even the simple situation of Example 4.1.1. For example, we might well want to make statements like "the receiver eventually knows the sender's initial bit." As we have already observed, we cannot express such temporal statements in our language.

To be able to make temporal statements, we extend our language by adding *temporal operators*, which are new modal operators for talking about time. We focus attention here on four temporal operators: $\Box$ ("always"), its dual $\Diamond$ ("eventually"), $\bigcirc$ ("next time"), and U ("until"). Intuitively, $\Box\varphi$ is true if φ is true now and at all later points; $\Diamond\varphi$ is true if φ is true at some point in the future; $\bigcirc\varphi$ is true if φ is true at the next step; and $\varphi U \psi$ is true if φ is true until ψ is true. More formally, in interpreted systems, we have

$$
\begin{aligned}
(\mathcal{I}, r, m) &\models \Box\varphi \quad \text{iff} \quad (\mathcal{I}, r, m') \models \varphi \text{ for all } m' \geq m, \\
(\mathcal{I}, r, m) &\models \Diamond\varphi \quad \text{iff} \quad (\mathcal{I}, r, m') \models \varphi \text{ for some } m' \geq m, \\
(\mathcal{I}, r, m) &\models \bigcirc\varphi \quad \text{iff} \quad (\mathcal{I}, r, m+1) \models \varphi, \text{ and} \\
(\mathcal{I}, r, m) &\models \varphi U \psi \quad \text{iff} \quad (\mathcal{I}, r, m') \models \psi \text{ for some } m' \geq m \text{ and} \\
&\qquad\qquad\qquad\quad (\mathcal{I}, r, m'') \models \varphi \text{ for all } m'' \text{ with } m \leq m'' < m'.
\end{aligned}
$$

Note that our interpretation of $\bigcirc\varphi$ as "φ holds at the next step" makes sense because our notion of time is discrete. All the other temporal operators make perfect sense even for continuous notions of time. It is easy to check that $\Diamond\varphi$ is equivalent to $true\, U \varphi$, while $\Box\varphi$ is equivalent to $\neg\Diamond\neg\varphi$ (see Exercise 4.2); thus, we can take $\bigcirc$ and U as our basic temporal operators, and define $\Diamond$ and $\Box$ in terms of U.

We saw earlier that if two points (r, m) and (r', m') in an interpreted system $\mathcal{I}$ have the same global state, then they agree on all formulas in $\mathcal{L}_n^{CD}$; i.e., if $r(m) = r'(m')$, then $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}, r', m') \models \varphi$, for all formulas $\varphi \in \mathcal{L}_n^{CD}$. Once we add time to the language, this is no longer true. For example, it is easy to construct an interpreted system $\mathcal{I}$ and two points (r, m) and (r', m') in $\mathcal{I}$ such that $r(m) = r'(m')$, but $(\mathcal{I}, r, m) \models \Diamond p$ and $(\mathcal{I}, r', m') \models \neg\Diamond p$ (Exercise 4.3).

In general, temporal operators are used for reasoning about events that happen along a single run. For example, in the bit-transmission problem, the formula $\Box(recbit \Rightarrow \Diamond recack)$ says that if at some point along a run the receiver receives the bit sent by the sender, then at some point in the future the sender will receive the acknowledgment sent by the receiver. By combining temporal and knowledge operators, we can make assertions about the evolution of knowledge in the system. For example, we mentioned earlier that in the context of the bit-transmission problem we may want to make statements such as "the receiver eventually knows the sender's initial bit." This statement can now be expressed by the formula

$$\Diamond(K_R(bit = 0) \vee K_R(bit = 1)).$$

Once we have temporal operators, there are a number of important notions that we can express. We have already seen the usefulness of $\Diamond$ and $\Box$. We briefly mention two other useful notions here, obtained by combining $\Diamond$ and $\Box$:

- The formula $\Box\Diamond\varphi$ is true if φ occurs *infinitely often*; i.e., $(\mathcal{I}, r, m) \models \Box\Diamond\varphi$ exactly if the set $\{m' \mid (\mathcal{I}, r, m') \models \varphi\}$ is infinite (Exercise 4.4).

- The formula $\Diamond\Box\varphi$ is true if φ is true *almost everywhere*; i.e., $(\mathcal{I}, r, m) \models \Diamond\Box\varphi$ if for some m' and all $m'' \geq m'$, we have $(\mathcal{I}, r, m'') \models \varphi$.

The temporal operators that we have defined can talk about events that happen only in the present or future, not events that happen in the past. While these suffice for many of our applications, we could certainly add temporal operators for reasoning about the past, for example, an analogue to $\Diamond$ that says "at some time in the past." We have not done so here simply to avoid introducing a plethora of new notation.

4.4 Examples of Systems

At the beginning of this chapter we said that in this book, we view any collection of interacting agents as a multi-agent system. In this section, we give examples of systems that arise in a number of different contexts, and show how they can be modeled in a straightforward way in our framework. In addition, we show how a number of standard assumptions that are made can be expressed in our framework.

4.4.1 Knowledge Bases

Informally, we can view a *knowledge base* (KB for short) as a system that is told facts about an external world, and is asked queries about that world. The standard approach in the AI community to modeling a KB is just to identify it with a formula, or set of formulas, that can informally be thought of as describing what the KB knows. When the KB is asked a query ψ, it computes (using some computational procedure) whether ψ follows from the information it has been given.

In this section, we model the KB in our framework. As we shall see, this gives us a number of advantages. For one thing, we describe assumptions about how the KB obtains its knowledge. For another, we can relate what the KB is told to what is true in the world. The first step in modeling the KB in our framework is to decide who the agents are and what the role of the environment is. The KB is clearly an agent in the system. In addition, we choose to have another agent called the *Teller*; this is the agent that *tells* the KB facts about the external world. We use the environment to model the external world. It is possible to use the environment to also model the Teller, but, as we shall see later on, our approach offers certain advantages. We want to view the environment's state as providing a complete description of (the relevant features of) the external world, the local state of the KB as describing the information that the KB has about the external world, and the local state of the Teller as describing the information that the Teller has about the external world and about the KB. This allows us to distinguish what is true (as modeled by the environment's state) from what is known to the Teller (as modeled by the Teller's state) and from what the KB is told (as modeled by the KB's state).

That still gives us quite a bit of freedom in deciding how to model the global states. If we can describe all the relevant features of the external world by using a set Φ of primitive propositions, then we can take the environment to be just a truth assignment to the primitive propositions in Φ. If, instead, we need to use first-order information to describe the world, then we can take the environment to be a relational structure, as discussed in Section 3.7.

What about the KB's local state? We want it to represent all the relevant information that the KB has learned. One option is to take the local state to consist of the sequence of facts that the KB has been told and queries that it has been asked. If we assume that the sequence of queries does not carry any information about the external world, then we can simplify this representation by including in the local state only the sequence of facts that the KB has been told, and ignoring the queries. This is in fact what we do.

Finally, the Teller's state has to describe the Teller's information about the external world and about the KB. Note that the Teller has complete information about the KB, since the Teller is the sole source for the KB's information. Thus, the Teller's local state contains a description of its information about external world as well as the sequence of facts that the KB has been told.

What does the KB know after it has been told some fact φ? Assuming that what it has been told is true, it may seem reasonable to say that the KB knows φ. This is clearly false, however, if the external world can change. It might well be the case that φ was true when the KB was told it, and is no longer true afterwards. For definiteness, we assume that the external world is stable. As we shall see, even with this assumption, if φ can include facts about the KB's knowledge, then φ may be true when the KB is told it, but not afterwards.

To get a feeling for some of the issues involved, we focus first on modeling a fairly simple concrete situation (where, in particular, it cannot happen that a formula φ is true when the KB is told it, but becomes false later). We consider later what happens when we weaken these assumptions. We assume that

1. the external world can be described propositionally, using the propositions in a finite set Φ,

2. the external world is stable, so that the truth values of the primitive propositions describing the world do not change over time,

3. the Teller has complete information about the external world,

4. the KB is told and asked facts only about the external world, and not facts about its own knowledge, and these facts are expressed as propositional formulas,

5. everything the KB is told is true, and

6. there is no *a priori* initial knowledge about the external world, or about what the KB will be told.

The first assumption tells us that we can represent the environment's state as a truth assignment α to the primitive propositions in Φ. The second assumption tells us that in each run r, the environment's state $r_e(m)$ is independent of m; the environment's state does not change over time. The third assumption tells us that the Teller's state includes the truth assignment α, which describes the external world. Given that we are representing the KB's local state as a sequence of facts that it has been told, the fourth assumption tells us that this local state has the form $\langle \varphi_1, \ldots, \varphi_k \rangle$, $k \geq 0$, where $\varphi_1, \ldots, \varphi_k$ are propositional formulas. We assume that the Teller's local state has a similar form, and consists of the truth assignment that describes the real world, together with the sequence of facts it has told the KB. Thus, we take the Teller's local state to be of the form $(\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle)$, where α is a truth assignment and $\varphi_1, \ldots, \varphi_k$ are propositional formulas. Since the Teller's state is simply the pair consisting of the environment's state and the KB's state, we do not represent it explicitly, but rather denote a global state by $(\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \cdot)$. The fifth assumption tells us that everything the KB is told is true. This means that in a global state of the form $(\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \cdot)$, each of $\varphi_1, \ldots, \varphi_k$ must be true under truth assignment α. The part of the sixth assumption that says that there is no initial knowledge of the world is captured by assuming that the initial state of every run has the form $(\alpha, \langle\,\rangle, \cdot)$, and that for every truth assignment α', there is some run with initial global state $(\alpha', \langle\,\rangle, \cdot)$. We capture the second half of the sixth assumption—that there is no knowledge about what information will be given—by not putting any further restrictions on the set of possible runs. We discuss this in more detail later.

To summarize, we claim our assumptions are captured by the interpreted system $\mathcal{I}^{kb} = (\mathcal{R}^{kb}, \pi^{kb})$, where $\mathcal{R}^{kb}$ consists of all runs r such that for some sequence $\varphi_1, \varphi_2, \ldots$ of propositional formulas and for some truth assignment α, we have

KB1. $r(0) = (\alpha, \langle\,\rangle, \cdot)$

KB2. if $r(m) = (\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \cdot)$, then

(a) either $r(m+1) = r(m)$, or $r(m+1) = (\alpha, \langle \varphi_1, \ldots, \varphi_k, \varphi_{k+1} \rangle, \cdot)$,

(b) $\varphi_1 \wedge \ldots \wedge \varphi_k$ is true under truth assignment α, and

(c) $\pi^{kb}(r, m) = \alpha$, that is, π^{kb} is defined so that the truth assignment at (r, m) is given by the environment's state.

Our assumption that $\mathcal{R}$ consists of *all* runs that satisfy KB1 and KB2 also captures the assumption that there is no knowledge about what information will be given. This is perhaps best understood by example. There may be *a priori* knowledge that, if p is true, then this is the first thing the KB will be told. This places a restriction on the set

of possible runs, eliminating runs with global states of the form $(\alpha, \langle\varphi_1, \ldots, \varphi_k\rangle, \cdot)$ such that $k \geq 1$ and p is true under the truth assignment α, but $\varphi_1 \neq p$. It is easy to construct other examples of how what information is given or the order in which it is given might impart knowledge beyond the facts themselves. By allowing all runs consistent with KB1 and KB2 in $\mathcal{R}$, we are saying that there is no such knowledge.

Having defined the system $\mathcal{I}^{kb}$, we can now see how the KB answers queries. Suppose that at a point (r, m) the KB is asked a query ψ, where ψ is a propositional formula. Since the KB does not have direct access to the environment's state, ψ should be interpreted not as a question about the external world, but rather as a question about the KB's knowledge of the external world. Thus, the KB should answer "Yes" exactly if $(\mathcal{I}^{kb}, r, m) \models K_{KB}\psi$ holds (taking K_{KB} to denote "the KB knows"), "No" exactly if $(\mathcal{I}^{kb}, r, m) \models K_{KB}\neg\psi$ holds, and "I don't know" otherwise.

It turns out that the KB essentially knows precisely the conjunction of what it has been told. Suppose the KB is in local state $\langle\varphi_1, \ldots, \varphi_k\rangle$. We can view the formula $\kappa = \varphi_1 \wedge \ldots \wedge \varphi_k$ as a summary of its knowledge about the world; the KB knows only what follows from this. This could be interpreted in two ways: the KB could answer "Yes" exactly if ψ is a consequence of κ, or if $K_{KB}\psi$ is a consequence of $K_{KB}\kappa$. As the following result shows, these two interpretations are equivalent.

Recall from Chapter 3 that $\mathcal{M}_n^{rst}$ consists of all Kripke structures where the $\mathcal{K}_i$ relations are equivalence classes, and we write $\mathcal{M}_n^{rst} \models \varphi$ if φ is valid in all Kripke structures in $\mathcal{M}_n^{rst}$.

Proposition 4.4.1 *Suppose that* $r_{KB}(m) = \langle\varphi_1, \ldots, \varphi_k\rangle$. *Let* $\kappa = \varphi_1 \wedge \ldots \wedge \varphi_k$ *and let* ψ *be a propositional formula. The following are equivalent:*

(a) $(\mathcal{I}^{kb}, r, m) \models K_{KB}\psi$.

(b) $\kappa \Rightarrow \psi$ *is a propositional tautology.*

(c) $\mathcal{M}_n^{rst} \models K_{KB}\kappa \Rightarrow K_{KB}\psi$.

Proof First, it can be shown that if κ and ψ are arbitrary propositional formulas, then $\kappa \Rightarrow \psi$ is a propositional tautology iff $\mathcal{M}_n^{rst} \models K_i\kappa \Rightarrow K_i\psi$ (see Exercise 4.5). This yields the equivalence of (b) and (c).

We now show that (b) implies (a). Assume that $\kappa \Rightarrow \psi$ is a propositional tautology. If $(r, m) \sim_{KB} (r', m')$, then $r'(m') = (\alpha', \langle\varphi_1, \ldots, \varphi_k\rangle, \cdot)$ for some α'. Since everything that the KB is told is true, κ is true under the truth assignment α', so we must have $(\mathcal{I}^{kb}, r', m') \models \kappa$. Since $\mathcal{M}_n^{rst} \models \kappa \Rightarrow \psi$, it follows that $(\mathcal{I}^{kb}, r', m') \models \psi$, and thus $(\mathcal{I}^{kb}, r, m) \models K_{KB}\psi$.

Finally, we show that (a) implies (b). Assume that (b) fails, that is, $\kappa \Rightarrow \psi$ is not a propositional tautology. This means that there must be a truth assignment α' under which the formula $\kappa \wedge \neg\psi$ is true. Since $\mathcal{R}$ consists of all runs satisfying properties KB1 and KB2, it is easy to show that there must be a point (r', m') in $\mathcal{R}$ such that $r'(m') = (\alpha', \langle\varphi_1, \ldots, \varphi_k\rangle, \cdot)$. Since $(\mathcal{I}^{kb}, r', m') \models \neg\psi$, and $(r, m) \sim_{KB} (r', m')$, it follows that $(\mathcal{I}^{kb}, r, m) \not\models K_{KB}\psi$. So (a) fails. ∎

Thus, Proposition 4.4.1 shows that under our assumptions, we can model the KB in the standard AI manner: as a formula. Moreover, in order to answer a query, the KB must compute what follows from the formula that represents its knowledge.

Proposition 4.4.1 characterizes how the KB answers propositional queries. How should the KB handle non-propositional queries such as $(p \Rightarrow K_{KB}p)$ ("if p is the case, then the KB knows it")? Here also we want the KB to answer "Yes" to a query φ exactly if $(\mathcal{I}^{kb}, r, m) \models K_{KB}\varphi$, "No" exactly if $(\mathcal{I}^{kb}, r, m) \models K_{KB}\neg\varphi$ holds, and "I don't know" otherwise. When does the formula $K_{KB}(p \Rightarrow K_{KB}p)$ hold? It is not hard to show that this formula is equivalent to $K_{KB}p \vee K_{KB}\neg p$, so the answer to this query already follows from Proposition 4.4.1: the answer is "Yes" if either p follows from what the KB has been told, or $\neg p$ does, and "I don't know" otherwise. Note that it is not possible here for the answer to be "No", since if φ is $(p \Rightarrow K_{KB}p)$, then $K_{KB}\neg\varphi$ is equivalent to the formula $K_{KB}(p \wedge \neg K_{KB}p)$, which is inconsistent with S5 (Exercise 3.10).

We are mainly interested in what can be said about formulas that involve only the KB's knowledge, because we view the Teller as being in the background here. We define a *KB-formula* to be one in which the only modal operator is K_{KB}; a *KB-query* is a query which is a KB-formula. For every KB-formula of the form $K_{KB}\varphi$ we can effectively find an equivalent formula that is a Boolean combination of formulas of the form $K_{KB}\psi$, where ψ is propositional (Exercise 3.23). It follows that the way that the KB responds to KB-queries can already be determined from how it responds to propositional queries. The reason is as follows. To decide on its answer to the query φ, we must determine whether $K_{KB}\varphi$ holds and whether $K_{KB}\neg\varphi$ holds. As we just noted, we can effectively find a formula equivalent to $K_{KB}\varphi$ that is a Boolean combination of formulas of the form $K_{KB}\psi$, where ψ is propositional, and similarly for $K_{KB}\neg\varphi$. We then need only evaluate formulas of the form $K_{KB}\psi$, where ψ is propositional. Thus, using Proposition 4.4.1, we can compute how the KB will answer KB-queries from the conjunction of the formulas that the KB has been told (Exercise 4.6).

There is another way of characterizing how the KB will answer KB-queries. Given a propositional formula φ, let S^{φ} consist of all truth assignments α such that φ is

true under truth assignment α. Let $M^\varphi = (S^\varphi, \pi, \mathcal{K}_{KB})$ be the Kripke structure such that $\pi(\alpha) = \alpha$ and $\mathcal{K}_{KB}$ is the universal relation (so that for all $\alpha, \beta \in S^\varphi$, we have $(\alpha, \beta) \in \mathcal{K}_{KB}$). In a sense, we can think of M^φ as a *maximal* model of φ, since all truth assignments consistent with φ appear in M^φ. As the following result shows, if κ is the conjunction of the formulas that the KB has been told, then for an arbitrary formula ψ, the KB knows ψ exactly if $K_{KB}\psi$ holds in the maximal model for κ. Intuitively, if the KB was told κ, then *all* that the KB knows is κ. The maximal model for κ is the model that captures the fact that κ is all that the KB knows.

Proposition 4.4.2 *Suppose that* $r_{KB}(m) = \langle \varphi_1, \ldots, \varphi_k \rangle$ *and* $\kappa = \varphi_1 \wedge \ldots \wedge \varphi_k$. *Then for all KB-formulas* ψ, *we have* $(\mathcal{I}^{kb}, r, m) \models \psi$ *iff* $(M^\kappa, r_e(m)) \models \psi$.

Proof See Exercise 4.7. ∎

In particular, Proposition 4.4.2 shows that the KB can answer a KB-query ψ by evaluating whether $(M^\kappa, r_e(m)) \models K_{KB}\psi$. Notice that the truth of $K_{KB}\psi$ in $(M^\kappa, r_e(m))$ is independent of $r_e(m)$. Thus, we can still view κ as a representation of the KB's knowledge. We remark that the ideas in this proposition can be extended to handle arbitrary non-propositional queries as well, rather than just KB-queries (Exercise 4.10).

Our discussion so far illustrates that it is possible to model a standard type of knowledge base within our framework. But what do we gain by doing so? For one thing, it makes explicit the assumptions underlying the standard representation. In addition, we can talk about what the KB knows regarding its knowledge, as shown in Proposition 4.4.2. Beyond that, as we now show, it allows us to capture in a straightforward way some variants of these assumptions. The flexibility of the model makes it easier to deal with issues that arise when we modify the assumptions.

We begin by considering situations where there is some prior knowledge about what information will be given. As we observed earlier, the fact that we consider *all* runs in which KB1 and KB2 are true captures the assumption that no such knowledge is available. But, in practice, there may well be default assumptions that are encoded in the conventions by which information is imparted. We earlier gave an example of a situation where there is a convention that if p is true, then the KB will be told p first. Such a convention is easy to model in our framework: it simply entails a restriction on the set of runs in the system. Namely, the restriction is that for every point (r, m) in the system where $r(m) = (\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \cdot)$ for some $k \geq 1$, we have $\varphi_1 = p$ precisely when p is true under α. Recall that the order in which the KB is given information is part of its local state. In a precise sense, therefore, the KB knows what this order is. In particular, it is straightforward to show that if there is a convention

that the KB will be told p first if it is is true, then the KB either knows p or knows $\neg p$ once it has been told at least one fact (Exercise 4.8).

In a similar fashion, it is easy to capture the situation where there is some *a priori* knowledge about the world, by modifying the set of runs in $\mathcal{I}^{kb}$ appropriately. Suppose, for example, that it is known that the primitive proposition p must be true. In this case, we consider only runs r such that $r_e(0) = \alpha$ for some truth assignment α that makes p true. An analogue to Proposition 4.4.1 holds: now the KB will know everything that follows from p and the facts that it has been told (see Exercise 4.9).

Next, let us consider the situation where the Teller does not have complete information about the world (though it still has complete information about the KB). We model this by including in the Teller's state a nonempty set $\mathcal{T}$ of truth assignments. Intuitively, $\mathcal{T}$ is the set of possible external worlds that the Teller considers possible. The set $\mathcal{T}$ replaces the single truth assignment that described the actual external world. Since we are focusing on knowledge here, we require that $\alpha \in \mathcal{T}$; this means that the true external world is one of the Teller's possibilities. The Teller's state also includes the sequence of facts that the KB has been told. To avoid redundancy, we denote the Teller's state by $\langle \mathcal{T}, \cdot \rangle$. Global states now have the form $(\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \langle \mathcal{T}, \cdot \rangle)$. We still require that everything the KB is told be true; this means that the Teller should tell φ to the KB only if φ is true in all the truth assignments in $\mathcal{T}$. It is easy to see that this means that the Teller says φ only if $K_T \varphi$ holds (taking K_T to denote "the Teller knows"). Not surprisingly, Propositions 4.4.1 and 4.4.2 continue to hold in this setting, with essentially no change in proof.

We remark that once we allow the Teller to have incomplete information, it becomes more interesting to consider situations with several Tellers. This is the situation that is perhaps most realistic. In practice, there may be several sources of information for the KB. In fact, in this situation the distinction between the Tellers and the KB blurs, and each agent may be viewed as both a Teller and a KB. We discuss this setting further in Section 7.3.

Up to now we have assumed that the actual world is one of the worlds in $\mathcal{T}$, the set of worlds that the Teller considers possible. It is but a short step to allow the Teller to have false beliefs, which amounts to allowing $\mathcal{T}$ not to include the actual world. We would still require that the Teller tell φ to the KB only if φ is true in all the truth assignments in $\mathcal{T}$. This means that the Teller only *believes* φ to be the case; its beliefs may be wrong. We can best capture this situation by using a possibility relation other than $\sim_{KB}$, one that corresponds to belief rather than knowledge; we leave the details to Exercise 4.11.

We have been assuming that the KB is told only propositional facts. Things get somewhat more complicated if the KB is given information that is not purely

propositional. For example, suppose the KB is told $p \Rightarrow K_{KB}p$. This says that if p is true, then the KB knows it. Such information can be quite useful, assuming that the KB can actually check what it knows and does not know. In this case, the KB can check if it knows p; if it does not, it can then conclude that p is false. As this example shows, once we allow the KB to be given information that relates its knowledge to the external world, then it may be able to use its introspective abilities to draw conclusions about the external world.

If the KB is told non-propositional facts, then we can no longer represent the KB's knowledge simply by the conjunction of facts that it has been told. In fact, in the non-propositional case, the KB may be told a fact that was true when it was told, but does not remain true once the KB is told it. For example, suppose the primitive proposition p is true of the external world, and the KB has not been given any initial information. In this situation, the formula $p \wedge \neg K_{KB}p$ is certainly true. But after the KB is told this, then it is certainly not the case that the KB knows $p \wedge \neg K_{KB}p$; indeed, as we noted earlier, $K_{KB}(p \wedge \neg K_{KB}p)$ is inconsistent with S5. Nevertheless, the KB certainly learns something as a result of being told this fact: it learns that p is true. As a result, $K_{KB}p$ should hold after the KB is told $p \wedge \neg K_{KB}p$.

Using our framework, we can still describe the system that results when we allow the KB to be told facts that include statements about its knowledge. As before, we take the KB's local state to consist of a sequence of formulas, except that we now allow the formulas to be modal formulas, which can talk about the KB's knowledge, not just propositional formulas. The only difficulty arises in restricting to runs in which the KB is told only true formulas. Because we are now interested in formulas that involve knowledge, it is not clear that we can decide whether a given formula is true without having the whole system in hand. But our problem is to construct the system in the first place! Doing this appropriately requires a little more machinery, which we present in Chapter 7. We thus defer further discussion of this issue until then.

4.4.2 Game Trees

The goal of game theory is to understand *games* and how they should be played. To a game theorist, a game is an abstraction of a situation where players interact by making "moves." Based on the moves made by the players, there is an outcome, or payoff, to the game. It should be clear that standard games such as poker, chess, and bridge are games in this sense. For example, the "moves" in bridge consist of bidding and playing the cards. There are rules for computing how many points each side gets at the end of a hand of bridge; this is the payoff. The game theorists' notion of game, however, encompasses far more than what we commonly think of as games.

Standard economic interactions such as trading and bargaining can also be viewed as games, where players make moves and receive payoffs.

Games with several moves in sequence are typically described by means of a *game tree*. A typical game tree is given in Figure 4.1. In the game G_1, there are two players, 1 and 2, who move alternately. Player 1 moves first, and has a choice of taking action a_1 or a_2. This is indicated by labeling the root of the tree with a 1, and labeling the two edges coming out of the root with a_1 and a_2 respectively. After player 1 moves, it is player 2's turn. In this game, we assume that player 2 knows the move made by player 1 before she moves. At each of the nodes labeled with a 2, player 2 can choose between taking action b_1 or b_2. (In general, player 2's set of possible actions after player 1 takes the action a_1 may be different from player 2's set of possible actions after player 1 takes the action a_2.) After these moves have been made, the players receive a payoff. The leaves of the tree are labeled with the payoffs. In the game G_1, if player 1 takes action a_1 and player 2 takes action b_1, then player 1 gets a payoff of 3, while player 2 gets a payoff of 4 (denoted by the pair (3, 4) labeling this leaf of the tree). A *play* of the game corresponds to a path in the game tree, that is, it is a complete sequence of moves by the players from start to finish.

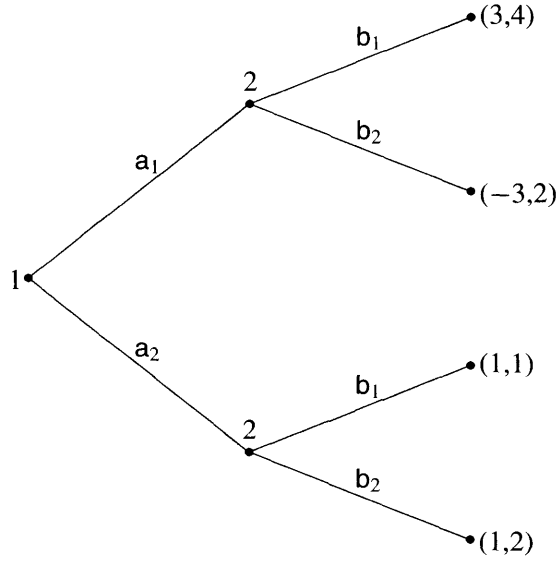

Figure 4.1 A game tree for G_1

It should be clear that, at least in principle, chess could also be described by a game tree. The nodes represent board positions, and the leaves of the tree represent positions where the game has ended. If we suppose that all games are played to the end (so no one either resigns or offers a draw), then the moves at each node are the legal chess moves in that position. There are only three possible outcomes: a win for White (player 1), a win for Black (player 2), or a draw. The plays in this game tree correspond to the possible (complete) games of chess.

The game represented in Figure 4.1 is called a game of *perfect information*; intuitively, every event relevant to the game takes place in public. A player knows all the moves that have been made before she moves. Chess is another example of a game with perfect information. By way of contrast, bridge and poker are not games of perfect information.

One of the key issues studied by game theorists is how the information available to the players when they move affects the outcome of the game. In particular, game theorists are quite interested in games where agents do not have perfect information. Consider, for example, the game tree for the game G_2 depicted in Figure 4.2. It is identical to the game tree in Figure 4.1, except that the two nodes where player 2 moves are enclosed by an oval. This oval is meant to indicate that the two nodes are indistinguishable to player 2, or, as game theorists would say (see Section 2.5), they are in the same information set. This means that when player 2 makes her move in this game, she does not know whether player 1 chose action a_1 or a_2. (Recall that in the first game we assumed that when player 2 made her move, she did know which action player 1 had chosen.) In general, game theorists use the information set to represent the information that a player has at a given point in the game. It is assumed that a player always knows when it is her turn to move, so that there cannot be two nodes in player i's information set, such that player i is supposed to move at one of the nodes and not at the other. (In fact, game theorists are typically interested in player i's information sets only at nodes where it is player i's turn to move.) Moreover, it is assumed that player i has the same choices of actions at all the nodes in her information set. It does not make sense for a player to be able to perform different actions at nodes that she cannot distinguish. As we can see from Figure 4.2, the set of actions from which player 2 must choose is identical at the two nodes where she moves. In contrast to game G_1, where the set of possible moves did not *have* to be identical, here they do.

Games in the sense of game theorists are certainly systems in our sense of the word. Not surprisingly, it is straightforward to model the two games we have described as systems according to our formal definition.

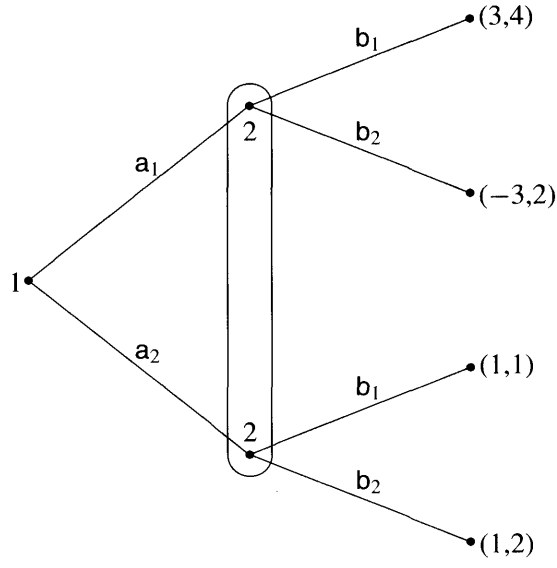

Figure 4.2 A game tree for G_2

Consider the first game, G_1. Perhaps the most obvious way of modeling it as a system is to have each play correspond to a run. Since we assume that G_1 is a game of perfect information, what happens at each move must be reflected in the players' states. Each player's initial state in this game has the form $\langle\,\rangle$, representing the fact that nothing has yet happened. After player 1's move, we assume that both players' local states encode the move that player 1 has made; thus, the local state has the form $\langle \mathsf{a}_i \rangle$, for $i = 1, 2$. Finally, after player 2's move, we assume that both players' states include player 2's move, and the payoff. We can ignore the environment's state here; we presume that the players' local states include all the information of interest to us. Thus, we take the environment's state to be λ. (We remark that in a more general setting, game theorists view "nature," or the environment, as another player in the game. In this case it may be appropriate to have a more complicated environment state.) Call the resulting system $\mathcal{R}_1$.

Notice that as we have described $\mathcal{R}_1$, both players have identical local states at all points. This is the formal counterpart to our assumption that G_1 is a game of perfect information. It is straightforward to show that the moves that have taken place, as well as the payoffs received by the players, are common knowledge once they take place (Exercise 4.12). In games of perfect information there is very little

uncertainty, which leads to such a simple model. We remark that even in games of perfect information such as this one, the players usually follow particular strategies, which are not necessarily common knowledge. Defining strategies and capturing them in the model will be one of the subjects treated in Chapter 5.

What system does G_2 correspond to? Again, we assume that there is a run for each play of the game, and the player's initial states have the form $\langle\,\rangle$. Just as in $\mathcal{R}_1$, we can also assume that player 1's local state after his move includes the move that he has made. We do not, however, want player 2's local state to include this information. The key difference between G_1 and G_2 is that player 2 does not know what player 1's move is after he has made it. Nevertheless, player 2's state must encode the fact that player 1 has moved. Player 2's state must be different before player 1 moves and after player 1 moves, for otherwise she would not know that it is her turn to move. For definiteness, we assume that immediately after player 1's move, player 2's state has the form $\langle move\rangle$; essentially, the *move* is just an indication that it is player 2's move. We assume that both players' states after player 2's move include the move, and the payoff. This gives us the system $\mathcal{R}_2$.

It is not hard to see that in the system $\mathcal{R}_2$, player 2 is guaranteed to discover after she moves what player 1's move was. Indeed, we now have the machinery to formally prove that at time 2, player 2 knows what move player 1 chose (Exercise 4.13). This, of course, depends on the fact that we are assuming that the players are notified after the second move what the final payoffs are. It should be obvious at this point that we can capture a situation in which players are not informed about the payoffs immediately, or perhaps that each player is informed of her or his own payoff and not the other's. All we need to do is modify what goes into a player's local state.

The systems $\mathcal{R}_1$ and $\mathcal{R}_2$ correspond to the games G_1 and G_2 described by the game trees in Figures 4.1 and 4.2 in that each play of the game is captured by one of the runs, and every run captures a possible play of the game. Of course, these systems are not the only possible representations of these games. For example, we could have just as well have used the information sets of the agents as local states. In the next chapter, we consider another possible representation, one that contains more information, namely, a representation of the strategy being used by the players.

4.4.3 Synchronous Systems

A standard assumption in many systems is that agents have access to a shared clock, or that actions take place in rounds or steps, and agents know what round it is at all times. Put another way, it is implicitly assumed that the time is common knowledge,

so that all the agents are running in synchrony. This assumption has already arisen in some of the systems we have considered. In particular, we implicitly made this assumption in our presentation of the muddy children puzzle and of the two games G_1 and G_2 of Section 4.4.2. Indeed, although synchrony is not a necessary assumption when modeling games, it is often assumed by game theorists. When linguists analyze a conversation, it is also typically assumed (albeit implicitly) that the agents share a clock or that the conversation proceeds in structured steps. In computer science, many protocols are designed so that they proceed in rounds (where no agent starts round $m + 1$ before all agents finish round m).

How can we capture synchrony in our framework? Since an agent's knowledge is determined by his local state, his knowledge of the time must be encoded somehow in the local state. This global clock need not measure "real time." Formally, $\mathcal{R}$ is a *synchronous system* if for all agents i and points (r, m) and (r', m') in $\mathcal{R}$, if $(r, m) \sim_i (r', m')$, then $m = m'$. This captures our intuition that in a synchronous system, each agent i knows what time it is; at all points that i considers possible at the point (r, m), the time (on the system's shared clock) is m. In particular, this means that i can distinguish points in the present from points in the future; i has a different local state at every point (r, m) in a run r. We say that an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is synchronous if $\mathcal{R}$ is synchronous.

It is easy to see that the systems $\mathcal{R}_1$ and $\mathcal{R}_2$ corresponding to the games G_1 and G_2 discussed in Section 4.4.2 are indeed synchronous in the sense of our formal definition (Exercise 4.14). Intuitively, it should also be clear that the muddy children puzzle should be modeled as a synchronous system, since each time the father asks a question or the children answer can be viewed as starting a new "round." As we shall see in Chapter 7, the system that we use to model the muddy children puzzle is indeed synchronous. On the other hand, the system $\mathcal{I}^{kb}$ that we use to model the knowledge base is not synchronous. Synchrony was not a major issue in that case. We could, of course, make it synchronous, either by adding a clock to the KB's and Teller's local states, or assuming that the Teller tells the KB a new formula at every step (so that the number of formulas in the KB's and Teller's local states encodes the time).

4.4.4 Perfect Recall

According to our definition of knowledge in a system, an agent's knowledge is determined by his local state. We might expect that, over time, an agent's local state might "grow" to reflect the new knowledge he acquires, while still keeping track of all the old information he had. We do not require this in our definition. It is quite

possible according to our definition that information encoded in $r_i(m)$—i's local state at time m in run r—no longer appears in $r_i(m+1)$. Intuitively, this means that agent i has lost or "forgotten" this information. There are often scenarios of interest where we want to model the fact that certain information is discarded. In practice, for example, an agent may simply not have enough memory capacity to remember everything he has learned.

While we view the ability to model such forgetting as a feature of our framework, there are many instances where it is natural to model agents as if they do not forget, that is, they have *perfect recall*. Perfect recall is sufficiently common in applications to warrant a definition and to be studied as a separate property of systems.

Perfect recall means, intuitively, that an agent's local state encodes everything that has happened (from that agent's point of view) thus far in the run. Among other things, this means that the agent's state at time $m+1$ contains at least as much information as his state at time m. Put another way, an agent with perfect recall should, essentially, be able to reconstruct his complete local history. In the case of synchronous systems, since an agent's local state changes with every tick of the external clock, agent i's having perfect recall would imply that the sequence $\langle r_i(0), \ldots, r_i(m)\rangle$ must be encoded in $r_i(m+1)$. In systems that are not synchronous, however, agents are not necessarily affected by the passage of time on the external clock. Roughly speaking, an agent can sense that something has happened only when there is a change in his local state. This motivates the following definitions.

Let *agent i's local-state sequence at the point* (r, m) be the sequence of local states she has gone through in run r up to time m, without consecutive repetitions. Thus, if from time 0 through time 4 in run r agent i has gone through the sequence $\langle s_i, s_i, s_i', s_i, s_i\rangle$ of local states, where $s_i \neq s_i'$, then her local-state sequence at $(r, 4)$ is $\langle s_i, s_i', s_i\rangle$. Process i's local-state sequence at a point (r, m) essentially describes what has happened in the run up to time m, from i's point of view.

Intuitively, an agent has perfect recall if her current local state encodes her whole local-state sequence. More formally, we say that *agent i has perfect recall in system $\mathcal{R}$* if at all points (r, m) and (r', m') in $\mathcal{R}$, if $(r, m) \sim_i (r', m')$, then agent i has the same local-state sequence at both (r, m) and (r', m'). Thus, agent i has perfect recall if she "remembers" her local-state sequence at all times. In a system with perfect recall, $r_i(m)$ encodes i's local-state sequence in that, at all points where i's local state is $r_i(m)$, she has the same local-state sequence. Notice that the systems $\mathcal{R}_1$ and $\mathcal{R}_2$ that we used to model the games G_1 and G_2 assume perfect recall, since the players keep track of all the moves that they make (Exercise 4.15). In fact, perfect recall is a standard assumption made by game theorists. It also holds in $\mathcal{I}^{kb}$, our representation

of knowledge bases (Exercise 4.16). As we shall see, perfect recall is an assumption made, either explicitly or implicitly, in a number of other contexts as well.

One might expect that in a system $\mathcal{I}$ where agents have perfect recall, if an agent knows a fact φ at a point (r, m), then she will know φ at all points in the future, that is, we might expect that $\mathcal{I} \models K_i\varphi \Rightarrow \Box K_i\varphi$. This is not quite true. One problem arises with statements talking about the situation "now." For example, if φ is the statement "it is currently time 0," then at time 0, an agent i may know φ (say, if she has access to a clock) but agent i will certainly not always know that it is time 0! Another problem comes from knowledge about ignorance. Suppose φ is the formula $\neg K_1 p$. Then it is not hard to construct a system where agents have perfect recall such that agent 1 initially does not know p, but she later learns p. Thus, we have $\neg K_1 p$, and hence $K_1 \neg K_1 p$, holding at time 0, but we certainly do not have $K_1 \neg K_1 p$ holding at all times in the future, since, by assumption, eventually $K_1 p$ holds (see Exercise 4.17). So, knowledge about ignorance does not necessarily persist in the presence of perfect recall. Nevertheless, the intuition that in systems where agents have perfect recall, once an agent knows φ, then she never forgets φ, is essentially correct. Namely, it is true for all *stable* formulas, where a stable formula is one that, once true, remains true (see Exercises 4.18 and 4.19 for more details; we return to this issue in Section 8.2).

How reasonable is the assumption of perfect recall? This, of course, depends on the application we have in mind, and on the model we choose. It is easy to see that perfect recall requires every agent to have a number of local states at least as large as the number of distinct local-state sequences she can have in the system. In systems where agents change state rather infrequently, this may not be too unreasonable. On the other hand, if we consider systems where there are frequent state changes, or look at systems over long stretches of time, then perfect recall may require a rather large (possibly infinite) number of states. This typically makes perfect recall an unreasonable assumption over long periods of time, although it is often a convenient idealization, and may be quite reasonable over short time periods.

The simple protocol considered in Example 4.1.1 is one in which the sender and receiver undergo very few state changes. In the system that models the protocol, the states of S and R do not reflect every separate sending or receipt of a message. The states change only when a message is *first* received. According to our definitions, both S and R have perfect recall in this case, despite the fact that neither S nor R remember how many times they have received or sent messages. The point is that S and R recall everything that was ever encoded in their states.

Often it is not clear until after the fact what information is relevant to an analysis. Frequently, a simple way to avoid deciding what to include in the state is simply to

have the state record all events that the agent is involved in, and assume that agents have perfect recall. If we can gain a reasonable understanding of the system under the assumption of perfect recall, we can then consider to what extent forgetting can be allowed without invalidating our analysis.

4.4.5 Message-Passing Systems

In many situations, particularly when analyzing protocols run by processes in a distributed system, we want to focus on the communication aspects of the system. We capture this in the notion of a *message-passing system*, where the most significant actions are sending and receiving messages. In message-passing systems, we view a process's local state as containing information about its initial state, the messages that it has *sent* and *received*, and what *internal actions* it has taken. Because we are interested only in the communication aspects of the system, the details of the internal actions are not relevant here. The internal actions can include such things as changing the value of a variable or reading or writing a value.

More formally, suppose we fix a set Σ_i of initial states for process i, a set INT_i of internal actions for i, and a set MSG of messages. Then a *history* for process i (over Σ_i, INT_i, and MSG) is a sequence whose first element is in Σ_i, and whose later elements consist of nonempty sets with elements of the form $send(\mu, j, i)$, $receive(\mu, j, i)$, or $int(\mathsf{a}, i)$, where $\mu \in MSG$ and $\mathsf{a} \in INT_i$. We think of $send(\mu, j, i)$ as representing the event "message μ is sent to j by i"; similarly, $receive(\mu, j, i)$ represents the event "message μ is received from j by i"; finally, $int(\mathsf{a}, i)$ represents the event "internal action a is performed by i." The details of Σ_i and INT_i are not relevant here. Intuitively, i's history at (r, m) consists of i's initial state, followed by the sequence describing i's actions up to time m. Thus, at the point $(r, 0)$, process i's history consists only of its initial state. If, for example, in round $m > 0$ of run r, process i performs the action of sending process j the message μ and also performs some internal action a, then i's history at the point (r, m) is the result of appending the set $\{send(\mu, j, i), int(\mathsf{a}, i)\}$ to its history at $(r, m-1)$. Similarly, if i's only action in round m is that of receiving the message μ' from j, then its history at (r, m) is the result of appending $\{receive(\mu', j, i)\}$ to its history at (r, m). If i performs no actions in round m, then its history at (r, m) is the same as its history at $(r, m-1)$. (Notice that we are distinguishing performing no action from performing some kind of null action to indicate that time has passed; a null action would be modeled as an internal action.)

In message-passing systems, we speak of $send(\mu, j, i)$, $receive(\mu, j, i)$, and $int(\mathsf{a}, i)$ as *events*. We say that an event *occurs* in round $m + 1$ of run r if it appears

in some process's history in $(r, m+1)$, but not in any process's history in (r, m). Although, strictly speaking, $r_i(m)$ is a sequence of sets of events, we often identify it with the set consisting of the union of all the events in all the sets in the sequence. We talk of an event being *in* $r_i(m)$ if it is in this set. This notion of event can be viewed as a special case of the notion of events that was discussed in Section 2.5, where an event was simply taken to be a set of possible worlds. Here we can associate with an event e the set of all points (r, m) where e occurs in round m of run r. For example, we can identify the event $send(\mu, j, i)$ with the set of points (i.e., possible worlds) where process i sends message μ to j.

In a message-passing system, the process's local state at any point is its history. Of course, if h is the history of process i at the point (r, m), then we want it to be the case that h describes what happened in r up to time m from i's point of view. To do this, we need to impose some consistency conditions on global states. In particular, we want to ensure that message histories grow (or at least do not shrink) over time, and that every message received in round m corresponds to a message that was sent at round m or earlier. Given sets Σ_i of initial states and INT_i of internal actions for processes $1, \ldots, n$, and a set MSG of messages, we define a *message-passing system* (over $\Sigma_1, \ldots, \Sigma_n, INT_1, \ldots, INT_n$, and MSG) to be a system such that for each point (r, m), the following constraints are satisfied:

MP1. $r_i(m)$ is a history over Σ_i, INT_i, and MSG,

MP2. for every event $receive(\mu, j, i)$ in $r_i(m)$ there exists a corresponding event $send(\mu, i, j)$ in $r_j(m)$, and

MP3. $r_i(0)$ is a sequence of length one (intuitively consisting of process i's initial state) and $r_i(m+1)$ is either identical to $r_i(m)$ or the result of appending a set of events to $r_i(m)$.

MP1 says that a process's local state is its history, MP2 guarantees that every message received at round m corresponds to one that was sent at round m or earlier, and MP3 guarantees that histories do not shrink. We have ignored the environment's state up to now. This is with good reason; the details of the environment's state are not relevant in this chapter, so we suppress them. The environment's state will become more important when we consider protocols in Chapters 5 and 7. We remark that by making each process's local state be its history, we ensure that processes have perfect recall in message-passing systems (see Exercise 4.22). In practice, we may want to add further requirements. For example, a *reliable* message-passing system is one where *communication is guaranteed*: every message sent is eventually received. Formally, a reliable message-passing system $\mathcal{R}$ is one that satisfies conditions MP1–MP3 as well as the following additional condition:

MP4. for all processes i, j, and all points (r, m) in $\mathcal{R}$, if $send(\mu, j, i)$ is in $r_i(m)$, then there exists an $m' \geq m$ such that $receive(\mu, i, j)$ is in $r_j(m')$.

We can easily impose further requirements on message-passing systems, such as a requirement that messages arrive in the order in which they are sent. We can force the system to be synchronous by assuming that $r_i(m) \neq r_i(m+1)$ for all i. This forces i to take some action at every step; i can then compute the time by looking at the length of its local history. Other requirements that we can impose on a system are discussed in Exercise 4.23. The key point is that the model is flexible enough to let us capture a wide range of assumptions quite easily.

4.4.6 Asynchronous Message-Passing Systems

In synchronous systems (it is common knowledge that) processes know exactly what time it is. Although the assumption of synchrony is widespread, it is by no means universal. While synchrony essentially implies that all processes share a global clock, in many computer science applications it is inappropriate to assume a global clock. Indeed, there may be very little information about time; we may not have any idea (or not be prepared to make any assumptions) about the relative speed of processes. For example, if a process becomes overloaded with work (and this may happen in an unpredictable fashion), the process may suddenly slow down relative to other processes. Similarly, we may not have upper bounds on message delivery times. To abstract this type of situation, and to understand the role of synchrony, computer scientists study systems that in some sense are as far away from being synchronous as possible. To minimize synchrony assumptions, one assumes that processes may work at arbitrary rates relative to each other and that there is no bound on message delivery times. We consider such asynchrony in this section, in the context of message-passing systems.

To capture these assumptions formally, we proceed much as in the previous section. As before, we assume that each process's local state consists of its history. As was the case in the bit-transmission problem, we use the environment's state to record the events that have taken place thus far, and the relative order in which they occur. The exact form of the environment's state plays only a minor role in this chapter, so we defer a detailed definition of the environment's state to Chapter 5. We also make the two following simplifying assumptions:

- We assume that at each step, at most one event takes place for each process. This means that we now take a history to be a sequence starting with an initial state, and followed by events (we blur here the distinction between an event e

and the singleton set $\{e\}$ containing the event e), rather than sets of events. This is a reasonable assumption if we model time at a sufficiently fine level of granularity.

- We assume that all the events in a given process's history are distinct. Our motivation for this is that we want to consider occurrences of events and their temporal relationship. It makes the exposition much easier if we do not have to distinguish different occurrences of the same event in a given run. In particular, this assumption has the effect that a process can never perform the same action twice in a given run, since if the internal action a is performed by process i both in round m and in round m' of run r, where $m < m'$, then the event $int(\mathsf{a}, i)$ would appear twice in process i's history at (r, m'). (If we want to consider systems where a process can perform the same internal action more than once, we could simply change our representation of the internal event a so that it is a triple $int(k, \mathsf{a}, i)$, where the first component k denotes the k^{th} occurrence of the internal event a performed by process i. Similar techniques could be used to deal with the possibility of i receiving the same message from j or sending the same message to j a number of times.)

In any message-passing system, a process knows at least what is in its history. It may well know more. For example, in a system where it is common knowledge that all processes perform an action at every step, a process can certainly deduce information about progress made by other processes from the amount of progress it has made. To eliminate all such additional knowledge, we consider possible *all* runs consistent with the assumptions MP1, MP2, and MP3. To make this precise, let us define a set V of histories to be *prefix-closed* if whenever h is a history in V, then every prefix of h other than the empty sequence is also in V. Let $V_1, \ldots, V_n$ be prefix-closed sets of histories for processes $1, \ldots, n$ respectively. Intuitively, V_i consists of all histories that process i could have. Define $\mathcal{R}(V_1, \ldots, V_n)$ to consist of all runs satisfying MP1, MP2, and MP3 such that all of process i's local states are in V_i. We define an *asynchronous message-passing system* (or a.m.p. system, for short) to be one of the form $\mathcal{R}(V_1, \ldots, V_n)$ for some choice of $V_1, \ldots, V_n$.

We remark that we can also consider systems that satisfy MP4 in addition to MP1–3. We call such a system an *asynchronous reliable message-passing system* (or a.r.m.p. system, for short). An a.m.p. system in which communication takes place can never be an a.r.m.p. system. Intuitively, this is because in an a.m.p. system we must allow for the possibility that a message will never be received, whereas in an a.r.m.p. system this situation is impossible. To see this, suppose that $\mathcal{R}(V_1, \ldots, V_n)$ is an a.m.p. system that includes a run r such that i sends j the message μ in

round m of r. Then there must be a run r' in $\mathcal{R}(V_1, \ldots, V_n)$ that agrees with r up to the beginning of round m, process i still sends μ to j in round m of r', but j never receives μ (Exercise 4.26). Because μ is never received in r', it follows that $\mathcal{R}(V_1, \ldots, V_n)$ is not an a.r.m.p. system. In the rest of this section, we focus on a.m.p. systems.

The fact that a.m.p. systems consist of *all* runs satisfying MP1–3 for some choice of $V_1, \ldots, V_n$ allows us to show that for every run $r \in \mathcal{R}$, a number of runs related to r must also be in $\mathcal{R}$. For example, suppose $r \in \mathcal{R}$ and r^1 is the run in which, intuitively, all events in r are stretched out by a factor of two. Thus, in r^1, all processes start in the same initial state as in r, no events occur in odd rounds of run r^1, and, for all m, the same events occur in round $2m$ of run r^1 as in round m of run r (so that for all times m, we have $r^1(2m) = r^1(2m+1) = r(m)$). It is easy to check that r^1 satisfies conditions MP1–3 (since r does), so r^1 must also be in $\mathcal{R}$. Similarly, any run that is like r except that there are arbitrarily long "silent intervals" between the events of r is also in $\mathcal{R}$. This shows that in a precise sense time is meaningless in a.m.p. systems (Exercise 4.24).

As we now show, an a.m.p. system is asynchronous in a strong sense: the only information a process has about the ordering of events is what follows from the order in its own history, together with the fact that a message must be sent before it is received. To make this precise, we define a notion of *potential causality* between events. This is intended to capture the intuition that event e might have caused event e'; in particular, this means that e *necessarily* occurred no later than e'. For events e and e' in a run r, we write $e \xrightarrow{r} e'$ if either

1. e' is a *receive* event and e is the corresponding *send* event,
2. for some process i, events e and e' are both in i's history at some point (r, m) and either $e = e'$ or e precedes e' (i.e., comes earlier in the history), or
3. for some event e'' we have $e \xrightarrow{r} e''$ and $e'' \xrightarrow{r} e'$.

Notice that condition (3) here guarantees that we have the transitive closure of conditions (1) and (2). These three properties together say that the only way that e could have affected e' is if either $e = e'$ or there is a sequence $e_1, \ldots, e_k$ of events such that $e = e_1$, $e' = e_k$, and for each consecutive pair e_h, e_{h+1}, either e_h and e_{h+1} are in the history of the same process, with e_h preceding e_{h+1}, or e_h is a *send* and e_{h+1} is the corresponding *receive*. Note that $\xrightarrow{r}$ is an *anti-symmetric* relation; we cannot have both $e \xrightarrow{r} e'$ and $e' \xrightarrow{r} e$ (which means that e is a potential cause of e' and e' is a potential cause of e) unless $e = e'$. It is worth pointing out, however, that this would not be the case if we allowed an event to occur more than once in a history.

The following result makes precise the degree to which an a.m.p. is asynchronous. It says that the potential causality relation $\xrightarrow{r}$ is the closest we can come in an a.m.p. system to defining a notion of ordering of events. That is, even if the processes could combine all their knowledge, they could not deduce any more about the ordering of events in run r than is implied by the $\xrightarrow{r}$ relation. For the following proposition, we assume that for each pair of events e and e', we have a proposition $Prec(e, e')$ in Φ. We say that the interpretation of $Prec(e, e')$ in the interpreted a.m.p. system $\mathcal{I} = (\mathcal{R}, \pi)$ is *standard* if $\pi(r(m))(Prec(e, e')) = \mathbf{true}$ exactly if events e and e' both occur by round m of run r, and e occurs no later than e' in r. (Note that our assumption that the environment keeps track of the events that have occurred means that π is well defined.)

Proposition 4.4.3 *Let G be the group of all processes, let $\mathcal{R}$ be an a.m.p. system, and assume that the interpretation of $Prec(e, e')$ in $\mathcal{I} = (\mathcal{R}, \pi)$ is standard. Then $(\mathcal{I}, r, m) \models D_G(Prec(e, e'))$ iff e and e' have both occurred by round m of r and $e \xrightarrow{r} e'$.*

Proof See Exercise 4.25. ∎

As we shall see in the next section, there are even closer connections between the potential causality ordering and knowledge.

4.5 Knowledge Gain in A.M.P. Systems

Given that the only interaction between the processes in an a.m.p. system is through sending and receiving messages, it is perhaps not surprising that there is a close relationship between knowledge and communication in an a.m.p. system. By better understanding this relationship, we shall be able to prove bounds on the number of messages required to gain certain levels of knowledge, and hence on the amount of communication required to achieve certain goals. As we shall see, the relationship between knowledge and communication is mediated by the causality relationship $\xrightarrow{r}$ described in Section 4.4.6. Roughly speaking, the only way for process i to gain knowledge about process j is to receive a message. Although this message does not have to come directly from process j, it should be the last in a chain of messages, the first of which was sent by j. The chain may, however, pass through a number of other processes before it reaches i. This observation motivates the next definition.

Suppose $i_1, \ldots, i_k$ is a sequence of processes, with repetitions allowed, r is a run, and $m < m'$, We say that $\langle i_1, \ldots, i_k \rangle$ *is a process chain in* $(r, m\,..\,m')$ if there exist events $e_1, \ldots, e_k$ in run r such that event e_1 occurs at or after round $m + 1$

in run r, event e_k occurs at or before round m', event e_j is in process i_j's history for $j = 1, \ldots, k$, and $e_1 \xrightarrow{r} \cdots \xrightarrow{r} e_k$. For example, suppose that in run r, process 1 sends the message μ to process 2 in round 1, process 2 receives μ in round 2, process 2 sends the message μ' to process 1 in round 3, and μ' is received by process 1 in round 3. Then $\langle 1, 2, 2, 1\rangle$ is a process chain in $(r, 0\,..\,3)$ (as is $\langle 1, 2, 1\rangle$). We say that $\langle i_1, \ldots, i_k\rangle$ is a process chain in r if it is a process chain in $(r, m\,..\,m')$ for some $m < m'$.

This example suggests that process chains are intimately linked to the sending and receiving of messages. It is easy to see that if $\langle 1, 2, 1\rangle$ is a process chain in run r corresponding to events e_1, e_2, and e_3 that occur in rounds m_1, m_2, and m_3 respectively, then there must have been a message sent by process 1 between rounds m_1 and m_2 inclusive (that is, at or after round m_1, and at or before round m_2) and a message sent by process 2 between rounds m_2 and m_3 inclusive. More generally, we have the following lemma.

Lemma 4.5.1 *Suppose $\langle i_1, \ldots, i_k\rangle$ is a process chain in $(r, m\,..\,m')$, with $i_j \neq i_{j+1}$ for $1 \leq j \leq k-1$. Then there must be a sequence of messages $\mu_1, \ldots, \mu_{k-1}$ sent in r such that μ_1 is sent by i_1 at or after round $m+1$, and μ_j is sent by i_j strictly after μ_{j-1} is sent by i_{j-1} for $1 < j \leq k-1$. In particular, at least $k-1$ messages must be sent in run r between rounds $m+1$ and m' inclusive.*

Proof See Exercise 4.27. ∎

Note that it is not necessarily the case that μ_j is sent by i_j to i_{j+1}. There may be a sequence of messages $\mu_{j,1}, \ldots, \mu_{j,l}$ such that $\mu_{j,1} = \mu_j$ is sent by i_j and $\mu_{j,l}$ is received by i_{j+1}.

The next definition is the key to relating knowledge and communication. If $i_1, \ldots, i_k$ is a sequence of processes, we write $(r, m) \sim_{i_1,\ldots,i_k} (r', m')$, and say that (r', m') *is $(i_1, \ldots, i_k)$-reachable from* (r, m), if there exist points $(r_0, m_0), \ldots, (r_k, m_k)$ such that $(r, m) = (r_0, m_0)$, $(r', m') = (r_k, m_k)$, and $(r_{j-1}, m_{j-1}) \sim_{i_j} (r_j, m_j)$ for $j = 1, \ldots, k$. Thus $(r, m) \sim_{i_1,\ldots,i_k} (r', m')$ if at the point (r, m) process i_1 considers it possible that i_2 considers it possible ... that i_k considers it possible that (r', m') is the current point. (Despite the notation, $\sim_{i_1,\ldots,i_k}$ is not in general an equivalence relation if $k > 1$.) The following lemma tells us that for every run r and for all times m, m', either (r, m') is $(i_1, \ldots, i_k)$-reachable from (r, m) or $\langle i_1, \ldots, i_k\rangle$ is a process chain in $(r, m\,..\,m')$. Putting this together with Lemma 4.5.1, this says that message transmission is essentially the only mechanism that blocks reachability.

Lemma 4.5.2 *Let $\mathcal{R}$ be an a.m.p. system, let r be a run in $\mathcal{R}$, and let $m < m'$. For all $k \geq 1$ and all sequences $i_1, \ldots, i_k$ of processes, either $(r, m) \sim_{i_1,\ldots,i_k} (r, m')$ or $\langle i_1, \ldots, i_k\rangle$ is a process chain in $(r, m\,..\,m')$.*

Proof We proceed by induction on k. If $k = 1$ and $\langle i_1 \rangle$ is not a process chain in $(r, m\,..\,m')$, then it must be the case that no events occur in process i_1's history in r between rounds $m+1$ and m' inclusive. It follows that $(r, m) \sim_{i_1} (r, m')$, as desired.

Suppose $k > 1$ and $\langle i_1, \ldots, i_k \rangle$ is not a process chain in $(r, m\,..\,m')$. Let e^* be the last event in i_k's history at the point (r, m'). We now define a new run r'. Intuitively, r' consists of all the events that occurred in r up to and including round m, together with all events that occurred in r after round m that potentially caused e^*. The run r' agrees with r up to time m (so that $r'(m'') = r(m'')$ for $0 \leq m'' \leq m$). For $m < m'' \leq m'$ and each process i, we define $r'_i(m'')$ to be the sequence that results from appending to $r'_i(m)$ (in the order they occurred) all events e in $r_i(m)$ that occurred between rounds $m+1$ and m'' (inclusive) such that $e \xrightarrow{r} e^*$. Finally, we take $r'(m'') = r'(m')$ for $m'' > m'$, that is, no event takes place in r' after time m'. It is easy to check that $r'_i(m'')$ is a prefix (not necessarily strict) of $r_i(m'')$ for all $m'' \geq 0$, because if e' occurs in $r_i(m'')$ before e and $e \xrightarrow{r} e^*$, then we also have $e' \xrightarrow{r} e^*$. It is now not hard to show that (1) $r' \in \mathcal{R}$ (that is, that r' satisfies the the conditions MP1–3 in definition of message-passing systems), (2) $(r', m') \sim_{i_k} (r, m')$, (3) $(r, m) \sim_{i_1} (r', m)$, and (4) $\langle i_1, \ldots, i_{k-1} \rangle$ is not a process chain in $(r', m\,..\,m')$ (Exercise 4.29). From (4) and the induction hypothesis it follows that $(r', m) \sim_{i_1,\ldots,i_{k-1}} (r', m')$. Applying (2) and (3) we now immediately get that $(r, m) \sim_{i_1,\ldots,i_k} (r, m')$, as desired. ∎

The two conditions in Lemma 4.5.2 are not mutually exclusive. It is possible to construct a run r such that, for example, $\langle 1, 2 \rangle$ is a process chain in $(r, 0\,..\,4)$ and $(r, 0) \sim_{1,2} (r, 4)$ (Exercise 4.30).

Lemma 4.5.2 allows us to relate message passing to knowledge in a.m.p. systems. One direct consequence is stated in the following theorem, which essentially says that processes can gain or lose knowledge only by sending and receiving messages.

Theorem 4.5.3 *Let r be a run in an interpreted a.m.p. system $\mathcal{I}$, and assume that $m < m'$.*

(a) If $(\mathcal{I}, r, m) \models \neg K_{i_k}\varphi$ and $(\mathcal{I}, r, m') \models K_{i_1} \ldots K_{i_k}\varphi$, then $\langle i_k, \ldots, i_1 \rangle$ is a process chain in $(r, m\,..\,m')$.

(b) If $(\mathcal{I}, r, m) \models K_{i_1} \ldots K_{i_k}\varphi$ and $(\mathcal{I}, r, m') \models \neg K_{i_k}\varphi$, then $\langle i_1, \ldots, i_k \rangle$ is a process chain in $(r, m\,..\,m')$.

Proof We prove (b) here; the proof of (a) is similar. Suppose, in order to obtain a contradiction, that $\langle i_1, \ldots, i_k \rangle$ is not a process chain in $(r, m\,..\,m')$. By Lemma 4.5.2, we have $(r, m) \sim_{i_1,\ldots,i_k} (r, m')$. Thus, by definition, there exist points $(r_0, m_0), \ldots, (r_k, m_k)$ such that $(r, m) = (r_0, m_0)$, $(r, m') = (r_k, m_k)$, and for

$j = 1, \ldots, k$ we have $(r_{j-1}, m_{j-1}) \sim_{i_j} (r_j, m_j)$. We can now show, by a straightforward induction on j, that $(\mathcal{I}, r_j, m_j) \models K_{i_j} \ldots K_{i_k}\varphi$ for $j = 1, \ldots, k$. In particular, it follows that $(\mathcal{I}, r, m') \models K_{i_k}\varphi$, a contradiction. ∎

Part (a) of Theorem 4.5.3 seems quite intuitive: it says that knowledge gain can occur only as the result of receiving messages. Part (b), however, may at first glance seem somewhat counterintuitive. It says that knowledge loss can occur only as the result of sending messages. How can processes *lose* knowledge by sending messages? To see how this can happen, suppose that process 1 sends process 2 a message saying "Hello," and that this is the first message sent from process 1 to process 2. Before process 1 sends the message, 1 knows that 2 has not received any messages from it. After it sends the messages, it loses this knowledge.

More deeply nested knowledge can be lost as well. Consider the following (somewhat contrived) example. Imagine a database system where at most one process at a time may have control of the database. (Intuitively, a process having control may update the database. We do not want more than one process trying to update the database at a time.) If no process has control, we say that the system is *freely available*. Control of our database is handled as follows. Intuitively, when a process has control, it puts a lock on the database, so no other process can access it. If only one process has a lock on the database, it can either release the database, so that it becomes freely available, or pass control on to a second process. That process can either return control to the first process, or pass control on to a third process, and so on. In general, a process either returns control to the process that passed control to it (intuitively, releasing its lock), or passes control on to a new process (after putting a lock on all the other locks already in place, and passing the information about all the existing locks to the new process). Since we are considering a.m.p. systems, this passing on of control is done by sending messages. Let φ be the fact "the database is not freely available." Suppose we have a situation where originally process 1 took control of the database, passed control on to process 2, who then passed control on to process 3. At that point we have $K_3K_2K_1\varphi$. Process 3 releases control by sending a message to process 2. Immediately after process 3 sends the message, we have $\neg K_3K_2K_1\varphi$. The reason is that because we are in an asynchronous system, process 3 has no way of knowing how much time has passed, so for all process 3 knows, process 2 got its message and released control to process 1, who then made the system freely available. In fact, $K_2K_1\varphi$ continues to hold until process 2 gets process 3's message and then sends a message to process 1 releasing control. The point is that in an asynchronous system, without getting further messages, process 3 has no way of knowing exactly when or whether this happens. Again, knowledge is lost as the result of sending messages.

There is another way that processes can lose knowledge by sending messages. It is possible that the knowledge they lose is knowledge about lack of knowledge; in this case part (b) of Theorem 4.5.3 can really be understood as a special case of part (a). For example, consider a formula p that intuitively says "the value of variable x is 0," where x is a variable local to process 3 (so that it can only be affected by 3's internal actions). Suppose that $(\mathcal{I}, r, 0) \models \neg p$, so that at time 0 in run r, variable x does not have value 0. By the results of Exercise 2.12, we know that $(\mathcal{I}, r, 0) \models K_1 K_2 \neg K_2 K_1 p$. (The intuitive argument for this is as follows: Clearly $\neg K_1 p$ holds, since process 1 cannot know a false fact. Thanks to negative introspection, we have also have $K_1 \neg K_1 p$. By similar reasoning, from $\neg K_1 p$, we can deduce both $\neg K_2 K_1 p$ and $K_2 \neg K_2 K_1 p$. But because process 1 *knows* $\neg K_1 p$, process 1 also knows $K_2 \neg K_2 K_1 p$, i.e., $K_1 K_2 \neg K_2 K_1 p$ holds.) Now take φ to be the formula $\neg K_2 K_1 p$. We have just shown that $(\mathcal{I}, r, 0) \models K_1 K_2 \varphi$. Suppose at some later time, say at the point $(r, 6)$, process 2 knows that process 1 knows p, i.e., we have $(\mathcal{I}, r, 6) \models K_2 K_1 p$. By definition of φ, we have $(\mathcal{I}, r, 6) \models \neg\varphi$. Therefore, we also have $(\mathcal{I}, r, 6) \models \neg K_2 \varphi$, since process 2 cannot know a false fact. Thus the knowledge loss—from $K_1 K_2 \varphi$ at time 0 to $\neg K_2 \varphi$ at time 6—is just a knowledge gain in disguise—from $\neg K_1 p$ at time 0 to $K_2 K_1 p$ at time 6. Whether we apply part (a) or part (b) of the theorem, it follows that there must be a process chain $\langle 1, 2 \rangle$ in $(r, 0\,..\,6)$. The same sequence of messages causes the knowledge gain and the knowledge loss.

One consequence of Theorem 4.5.3 is that common knowledge can be neither gained nor lost in a.m.p. systems.

Theorem 4.5.4 *Suppose $\mathcal{I}$ is an interpreted a.m.p. system, r is a run in $\mathcal{I}$, and G is a group of processes with $|G| \geq 2$. Then for all formulas φ and all times $m \geq 0$, we have $(\mathcal{I}, r, m) \models C_G \psi$ iff $(\mathcal{I}, r, 0) \models C_G \psi$.*

Proof Suppose $(\mathcal{I}, r, m) \models C_G \psi$ and $(\mathcal{I}, r, 0) \models \neg C_G \psi$. Suppose that exactly l messages are sent between rounds 1 and m inclusive. Since $(\mathcal{I}, r, 0) \models \neg C_G \psi$, there must be some sequence $i_1, \ldots, i_k$ of pairwise distinct processes in G such that $(\mathcal{I}, r, 0) \models \neg K_{i_k} \ldots K_{i_1} \psi$. Let i and j be distinct processes in G such that $j \neq i_k$. (Such processes must exist since $|G| \geq 2$.) Since $(\mathcal{I}, r, m) \models C_G \psi$, it follows that $(\mathcal{I}, r, m) \models (K_i K_j)^l K_{i_k} \ldots K_{i_1} \psi$, where we define $(K_i K_j)^l \psi'$ for any formula ψ' inductively in the obvious way. By part (a) of Theorem 4.5.3, where the role of φ is played by $K_{i_{k-1}} \ldots K_1 \psi$, it follows that $\langle i_k, j, i \ldots, j, i \rangle$ is a process chain in $(r, 0\,..\,m)$, where there are l occurrences of j, i in this chain. By Lemma 4.5.1, at least $2l$ messages must be sent in round r between rounds 1 and m. But this contradicts our assumption that exactly l messages are sent. Thus, common knowledge cannot be

gained. The proof that common knowledge cannot be lost proceeds along identical lines, using part (b) of Theorem 4.5.3. ∎

As we shall see in Chapter 6, the fact that common knowledge cannot be gained or lost in a.m.p. systems has important consequences for the possibility of coordinating actions in a.m.p. systems. We remark that Theorem 4.5.3, and hence Theorem 4.5.4, hold with almost no change in proof for a.r.m.p. systems as well (Exercise 4.32). This shows that reliability considerations do not play an important role in these results. It is the fact that there is no bound on message delivery time, rather than the question of whether the message will be delivered, that is crucial here.

Using Theorem 4.5.3, we can also prove a number of lower bounds on the number of messages required to solve certain problems. Consider the problem of *mutual exclusion*. Intuitively, the situation here is that from time to time a process tries to access some shared resource that may be accessed by only one process at a time. (For example, the process may try to change the value of a shared variable.) We say that the process is in its *critical section* when it has access to the shared resource. We say that $\mathcal{R}$ is a *system with mutual exclusion* if in every run of $\mathcal{R}$, no two processes are simultaneously in their critical sections.

If $\mathcal{R}$ is a system with mutual exclusion and r is a run in $\mathcal{R}$ in which $i_1, i_2, \ldots, i_k$ (with $i_j \neq i_{j+1}$ for $1 \leq j < k$) enter their critical section in sequence, then $\langle i_1, i_2, \ldots, i_k \rangle$ is a process chain in r. For suppose that i_j enters its critical section at time m_j in r, for $j = 1, \ldots, k$. For each process i, let cs_i be a primitive proposition denoting that i is in its critical section. Formally, we consider the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$, where π makes cs_i true at exactly those global states where process i is in its critical section. From the assumption that $\mathcal{R}$ is a system with mutual exclusion, it follows that $cs_i \Rightarrow \neg cs_j$ is valid in $\mathcal{I}$ if $i \neq j$. Moreover, because the truth of cs_i is determined by i's local state, we must have that both $cs_i \Rightarrow K_i(cs_i)$ and $\neg cs_i \Rightarrow K_i(\neg cs_i)$ are valid in $\mathcal{I}$. Since we have assumed that process i_j enters its critical section at time m_j in r, we have $(\mathcal{I}, r, m_j) \models cs_{i_j}$. It follows from the previous observations and the S5 properties of our definition of knowledge that for $j = 1, \ldots, k-1$, we have

$$(\mathcal{I}, r, m_j) \models K_{i_j} K_{i_{j+1}} \neg cs_{i_{j+1}} \text{ and}$$
$$(\mathcal{I}, r, m_{j+1}) \models \neg K_{i_{j+1}} \neg cs_{i_{j+1}}$$

(Exercise 4.34). Thus, by Theorem 4.5.3(b), $\langle i_j, i_{j+1} \rangle$ is a process chain in $(r, m_j \,..\, m_{j+1})$. It immediately follows that $\langle i_1, i_2, \ldots, i_k \rangle$ is a process chain in r. By Lemma 4.5.1, the existence of this process chain in r implies that at least $k-1$

messages are sent in r. This gives us a lower bound on the number of messages required for mutual exclusion: for k processes to enter their critical sections, at least $k - 1$ messages are required.

Exercises

4.1 Show that for all formulas $\varphi \in \mathcal{L}_n^{CD}$ and all interpreted systems $\mathcal{I}$, if (r, m) and (r', m') are points in $\mathcal{I}$ such that $r(m) = r'(m')$, then $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}, r', m') \models \varphi$.

4.2 Prove that $(\mathcal{I}, r, m) \models \Diamond\varphi$ iff $(\mathcal{I}, r, m) \models true\, U \varphi$, and $(\mathcal{I}, r, m) \models \Box\varphi$ iff $(\mathcal{I}, r, m) \models \neg\Diamond\neg\varphi$.

4.3 Construct an interpreted system $\mathcal{I}$ and two points (r, m) and (r', m') in $\mathcal{I}$ such that $r(m) = r'(m')$, but $(\mathcal{I}, r, m) \models \Diamond p$ and $(\mathcal{I}, r', m') \models \neg\Diamond p$.

4.4 Show that $(\mathcal{I}, r, m) \models \Box\Diamond\varphi$ exactly if the set $\{m' \mid (\mathcal{I}, r, m') \models \varphi\}$ is infinite.

4.5 In this exercise, we fill in some of the missing details in the proof of Proposition 4.4.1 (among other things). Recall from Chapter 3 that $\mathcal{M}_n$ consists of all Kripke structures with no constraints on the $\mathcal{K}_i$ relations, while $\mathcal{M}_n^{rst}$ consists of all Kripke structures where the $\mathcal{K}_i$'s are equivalence relations. We write $\mathcal{M}_n \models \varphi$ (resp., $\mathcal{M}_n^{rst} \models \varphi$) if φ is valid in all Kripke structures in $\mathcal{M}_n$ (resp., in $\mathcal{M}_n^{rst}$).

(a) Show that for arbitrary formulas φ and ψ, we have $\mathcal{M}_n \models \varphi \Rightarrow \psi$ iff $\mathcal{M}_n \models K_i\varphi \Rightarrow K_i\psi$.

(b) Show that for propositional formulas φ and ψ, we have $\mathcal{M}_n^{rst} \models \varphi \Rightarrow \psi$ iff $\mathcal{M}_n^{rst} \models K_i\varphi \Rightarrow K_i\psi$. Moreover, show that $\mathcal{M}_n^{rst} \models \varphi \Rightarrow \psi$ iff $\varphi \Rightarrow \psi$ is a propositional tautology.

(c) Show that for arbitrary formulas φ and ψ, we have $\mathcal{M}_n^{rst} \models \varphi \Rightarrow \psi$ implies $\mathcal{M}_n^{rst} \models K_i\varphi \Rightarrow K_i\psi$, but that there exist formulas φ and ψ such that $\mathcal{M}_n^{rst} \models K_i\varphi \Rightarrow K_i\psi$, but $\mathcal{M}_n^{rst} \not\models \varphi \Rightarrow \psi$.

4.6 Show how to use Proposition 4.4.1 and Exercise 3.23 to compute how the KB can answer KB-queries from the conjunction of the formulas that the KB has been told.

4.7 Prove Proposition 4.4.2.

4.8 Suppose there is a default rule saying that if p is true, then it is the first fact the KB is told. Show how $\mathcal{I}^{kb}$ can be modified to take this into account, and prove that for the appropriate modification, if $r_{KB}(m) = \langle \varphi_1, \ldots, \varphi_k \rangle$ for some $k \geq 1$ and $\varphi_1 \neq p$, then $(\mathcal{I}^{kb}, r, m) \models K_{KB} \neg p$.

4.9 Suppose that the KB has *a priori* knowledge that the propositional formula φ holds. Show how $\mathcal{I}^{kb}$ can be modified to take this into account, and show that the analogue of Proposition 4.4.1 holds: the KB will know everything that follows from what it has been told and φ.

* **4.10** Suppose that $r_{KB}(m) = \langle \varphi_1, \ldots, \varphi_k \rangle$, and let $\kappa = \varphi_1 \wedge \ldots \wedge \varphi_k$. By extending the ideas of Proposition 4.4.2, show how we can construct a Kripke structure $(M^\kappa)^+$ which has an accessibility relation corresponding to the Teller's knowledge, as well as one corresponding to the KB's knowledge, such that for all formulas ψ, we have $(\mathcal{I}^{kb}, r, m) \models \psi$ iff $((M^\kappa)^+, r_e(m)) \models \psi$.

4.11 In this exercise, we consider the implications of allowing the the Teller to have false beliefs. Suppose that, as before, the Teller's state includes the set $\mathcal{T}$ of worlds that it considers possible, but we allow the real world not to be in $\mathcal{T}$. We require, however, that the Teller tells φ to the KB only if φ is true in all the truth assignments in $\mathcal{T}$. Rather than use the $\sim_T$ relation to define the Teller's knowledge in the Kripke structure associated with the system $\mathcal{I}^{kb}$, suppose we use the relation $\mathcal{K}_T$ defined by $(r(m), r'(m')) \in \mathcal{K}_T$ if (1) $r_T(m) = r'_T(m')$ and (2) if $r'(m') = (\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \langle \mathcal{T}, \cdot \rangle)$, then $\alpha \in \mathcal{T}$. That is, the only worlds that the Teller considers possible are ones corresponding to its beliefs as captured by $\mathcal{T}$.

(a) Show that if we define knowledge using the $\mathcal{K}_T$ relation instead of $\sim_T$, and the Teller tells φ to the KB at the point (r, m), then $(\mathcal{I}^{kb}, r, m) \models K_T \varphi$.

(b) Show that if we define knowledge using the $\mathcal{K}_T$ relation, then the Teller's knowledge satisfies all the S5 properties except the Knowledge Axiom $K_i \varphi \Rightarrow \varphi$. Instead, the Teller's knowledge satisfies the property $\neg K_T(\mathit{false})$ (and thus is characterized by the axiom system KD45 of Chapter 3). This gives us a natural way of defining belief in this context.

(c) Suppose the KB *believes* that everything it is told is true. Since the Teller also believes that everything it says is true, we can capture this by using the

relation $\mathcal{K}_{KB}$ defined by $(r(m), r'(m')) \in \mathcal{K}_{KB}$ if (1) $r_{KB}(m) = r'_{KB}(m')$ and (2) if $r'(m') = (\alpha, \langle\varphi_1, \ldots, \varphi_k\rangle, \langle\mathcal{T}, \cdot\rangle)$, then $\alpha \in \mathcal{T}$. Notice that the second condition guarantees that at all points the KB considers possible, the Teller's beliefs are correct, so that everything the KB is told is true. Show that if we use $\mathcal{K}_{KB}$ to define the KB's knowledge, then K_{KB} also satisfies the axioms of KD45.

(d) Show that analogues to Propositions 4.4.1 and 4.4.2 hold in this context as well.

4.12 This exercise deals with the game G_1 of Section 4.4.2. Let Φ consist of primitive propositions of the form $act_i(\mathsf{a})$ for each $\mathsf{a} \in \{\mathsf{a}_1, \mathsf{a}_2, \mathsf{b}_1, \mathsf{b}_2\}$ and $i \in \{1, 2\}$. Let $\mathcal{I}_1 = (\mathcal{R}_1, \pi)$ be an interpreted system corresponding to the game G_1, where $\mathcal{R}_1$ is as described in the text and π gives the following interpretation to the primitive propositions: $\pi(r, m)(act_i(\mathsf{a})) =$ **true** exactly if i has taken action a in the global state $r(m)$. Prove that for all runs $r \in \mathcal{R}_1$ and actions a we have

(a) $(\mathcal{I}_1, r, 1) \models act_1(\mathsf{a}) \Leftrightarrow C(act_1(\mathsf{a}))$,

(b) $(\mathcal{I}_1, r, 2) \models act_2(\mathsf{a}) \Leftrightarrow C(act_2(\mathsf{a}))$.

This shows that, in this game, players' moves are common knowledge once they take place.

4.13 This exercise deals with the game G_2 of Section 4.4.2. Let Φ and π be defined as in Exercise 4.12, and let $\mathcal{I}_2 = (\mathcal{R}_2, \pi)$ be an interpreted system corresponding to the game G_2, where $\mathcal{R}_2$ is as described in the text. Prove that for every run $r \in \mathcal{R}_2$ and actions $\mathsf{a} \in \{\mathsf{a}_1, \mathsf{a}_2\}$ and $\mathsf{b} \in \{\mathsf{b}_1, \mathsf{b}_2\}$ we have:

(a) $(\mathcal{I}_2, r, 1) \models \neg K_2(act_1(\mathsf{a}))$,

(b) $(\mathcal{I}_2, r, 2) \models act_1(\mathsf{a}) \Leftrightarrow K_2(act_1(\mathsf{a}))$,

(c) $(\mathcal{I}_2, r, 2) \models (act_1(\mathsf{a}) \wedge act_2(\mathsf{b})) \Leftrightarrow C(act_1(\mathsf{a}) \wedge act_2(\mathsf{b}))$.

Although, as part (a) shows, the moves of player 1 are not known to player 2 when she moves (as shown in Exercise 4.12, this is in contrast to the situation in the game G_1), part (c) shows that all the moves made by both players are common knowledge at the end of every play.

4.14 Show that, according to our formal definitions, the systems $\mathcal{R}_1$ and $\mathcal{R}_2$ representing the games G_1 and G_2 of Section 4.4.2 are synchronous.

4.15 Show that both players have perfect recall in the systems $\mathcal{R}_1$ and $\mathcal{R}_2$ representing the games G_1 and G_2 of Section 4.4.2.

4.16 Show that in the system $\mathcal{I}^{kb}$ representing a knowledge base, the agents have perfect recall.

4.17 Show that knowledge of ignorance does not necessarily persist in systems where agents have perfect recall. In particular, construct an interpreted system $\mathcal{I}$ where agents have perfect recall and a run r in $\mathcal{I}$ such that $(\mathcal{I}, r, 0) \models \neg K_1 p$ and $(\mathcal{I}, r, m) \models K_1 p$ for $m > 0$.

4.18 A formula φ is said to be *stable (with respect to the interpreted system $\mathcal{I}$)* if once φ is true it remains true; i.e., if we have $\mathcal{I} \models \varphi \Rightarrow \Box\varphi$. Assume that φ_1 and φ_2 are stable.

(a) Show that $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ are stable.

(b) If, in addition, $\mathcal{I}$ is a system where agents have perfect recall, show that $K_i\varphi_1$ and $C_G\varphi_1$ are stable. Thus, in a system where agents have perfect recall, if an agent knows a stable formula at some point, then he knows it from then on.

(c) Show that if we assume in addition that the system is synchronous, then $D_G\varphi_1$ is stable.

* **4.19** This exercise considers whether the stability of formulas is preserved by the knowledge and distributed knowledge operators.

(a) Show by example that there is an interpreted system where agents have perfect recall and there is a stable formula φ_1 where $D_G\varphi_1$ is *not* stable. This contrasts with the situation in Exercise 4.18(c), where we also assume that the system is synchronous.

(b) Show by example that $K_1\varphi_2$ is not necessarily stable (even in synchronous systems where agents have perfect recall) if φ_2 is not stable.

4.20 Let $\mathcal{R}$ be a system in which agents have perfect recall.

(a) Show that if $r \in \mathcal{R}$ and $r_i(m) = r_i(m')$, then $r_i(m) = r_i(m'')$ for all m'' with $m \leq m'' \leq m'$.

(b) We say that agent i *considers run* r' *possible at the point* (r, m) if $(r, m) \sim_i (r', m')$ for some m'. Show that the set of runs that an agent considers possible in $\mathcal{R}$ does not increase over time; i.e., show that if $r \in \mathcal{R}$ and $m' > m$, then the set of runs agent i considers possible at (r, m') is a subset of the set of runs i considers possible at (r, m).

4.21 Consider a synchronous message-passing system with two processes, where process 1 sends process 2 a message μ_1 in round 1 and a message μ_2 in round 2. Process 2 does nothing. Describe L_1 and L_2 and the set of runs in each of the following situations:

(a) messages are received in the same round that they are sent,

(b) messages are either received in the same round that they are sent or not received at all,

(c) messages are guaranteed to be received eventually (but there is no upper bound on message delivery time).

4.22 Show that processes have perfect recall in message-passing systems.

4.23 Show how the following requirements can be imposed on message-passing systems:

(a) a requirement that agents take some action at least once every k steps,

(b) a requirement that messages arrive within k rounds.

4.24 In this exercise, we show that if a given run r is in an a.m.p. system, then so are a number of variants of r. A function $f : N \to N$ is called a *stuttering* function if, intuitively, f is monotone nondecreasing, and increases by increments of at most 1 at a time. More formally, f is a stuttering function if $f(0) = 0$ and $f(m + 1)$ satisfies $f(m) \leq f(m + 1) \leq f(m) + 1$ for all $m \geq 0$. An example of the range of a stuttering function would be 00111223444455 .... Given a run r, we say that r' is a *stuttered version* of r if, for some stuttering function f, we have $r'(m) = r(f(m))$ for all $m \geq 0$.

Prove that a.m.p. systems are closed under stuttering. Namely, show that if r is a run of an a.m.p. system $\mathcal{R}$ and r' is a stuttered version of r, then r' is a run of $\mathcal{R}$. Note that it immediately follows that if r is a run in an a.m.p. system $\mathcal{R}$, then so is the run r^1 described in Section 4.4.6 in which $r^1(2m) = r^1(2m + 1) = r(m)$, so that all events in r are stretched out by a factor of two. Similarly, any run in which there are arbitrarily long "silent intervals" between the events of r will also be in $\mathcal{R}$.

4.25 Part (a) of this exercise, like the previous exercise, involves showing that if a given run r is in an a.m.p. system, then so are a number of variants of r. This turns out to be the key step in proving Proposition 4.4.3.

(a) Assume that r is a run in an a.m.p. system $\mathcal{R}$, that e is an event, and that k is a positive integer. Let r' be a run defined intuitively as follows: For every event e' occurring in r, if $e \xrightarrow{r} e'$, then e' occurs k rounds later in r' than in r, and otherwise e' occurs in r' in the same round as in r. More formally, let i be an agent. If there is no event e' and time m_i where $e \xrightarrow{r} e'$ and e' is in $r_i(m_i)$, then let $r'_i(m) = r_i(m)$ for every m. Otherwise, let m_i be minimal such that there is an event e' where $e \xrightarrow{r} e'$ and e' is in $r_i(m_i)$. Then define

$$r'_i(m) = \begin{cases} r_i(m) & \text{if } m < m_i \\ r_i(m_i - 1) & \text{if } m_i \leq m < m_i + k \\ r_i(m - k) & \text{if } m_i + k \leq m. \end{cases}$$

Show that r' is a run in $\mathcal{R}$.

(b) Prove Proposition 4.4.3. (Hint: showing that if $e \xrightarrow{r} e'$ and e and e' have occurred by time m, then $(\mathcal{I}, r, m) \models D_G Prec(e, e')$ is straightforward. For the converse, if it is not the case that $e \xrightarrow{r} e'$, use part (a) to find a run $r' \in \mathcal{R}$ and a time m' such that $(r, m) \sim_i (r', m')$ for all processes i and $(\mathcal{I}, r', m') \models \neg Prec(e, e')$.)

4.26 Let $\mathcal{R}$ be an a.m.p. system and let r be a run in $\mathcal{R}$ such that i sends j the message μ in round m of r. Show that there must be a run r' in $\mathcal{R}$ that agrees with r up to the beginning of round m, where process i still sends μ to j in round m of r', but where j never receives μ in r'. (Hint: use Exercises 4.24 and 4.25(a).)

4.27 Prove Lemma 4.5.1.

4.28 Define a *consistent cut* of a run $r \in \mathcal{R}$ to be a tuple $\langle s_1, \ldots, s_n \rangle$ of local states such that (a) there exist $m_1, \ldots, m_n$ such that $r_i(m_i) = s_i$, for $i = 1, \ldots, n$ (so that each of these local states appears at some point in the run) and (b) there exists a run $r' \in \mathcal{R}$ and a time m such that $r'_i(m) = s_i$, for $i = 1, \ldots, n$. Thus, a consistent cut is essentially one that could have occurred as a global state at some point in the system. Show that $\langle s_1, \ldots, s_n \rangle$ is a consistent cut of a run r in an a.m.p. system $\mathcal{R}$ iff (a) there exist $m_1, \ldots, m_n$ such that $r_i(m_i) = s_i$, for $i = 1, \ldots, n$ and (b) these states are closed downwards with respect to $\xrightarrow{r}$; i.e., if $e \xrightarrow{r} e'$, and e' is an event in some s_i (this makes sense in an a.m.p. system since the local states are essentially sequences of events), then e is also an event in some s_i.

* **4.29** Fill in the details of the proof of Lemma 4.5.2. In particular, show that r' satisfies the four conditions stated near the end of Lemma 4.5.2.

4.30 Show that there is a run r in an a.m.p. system $\mathcal{R}$ such $\langle 1, 2\rangle$ is a process chain in $(r, 0\,..\,4)$ and $(r, 0) \sim_{1,2} (r, 4)$.

4.31 This exercise presents a variant of Theorem 4.5.3. Let r be a run in an interpreted a.m.p. system $\mathcal{I}$ and let $m < m'$.

(a) Show that if $(\mathcal{I}, r, m) \models \neg K_{i_k} K_{i_{k+1}} \varphi$ and $(\mathcal{I}, r, m') \models K_{i_1} \ldots K_{i_k} K_{i_{k+1}} \varphi$, then $\langle i_{k+1}, i_k, \ldots, i_1\rangle$ is a process chain in $(r, m\,..\,m')$.

(b) Show that if $(\mathcal{I}, r, m) \models K_{i_1} \ldots K_{i_k} K_{i_{k+1}} \varphi$ and $(\mathcal{I}, r, m') \models \neg K_{i_k} K_{i_{k+1}} \varphi$, then $\langle i_1, \ldots, i_k, i_{k+1}\rangle$ is a process chain in $(r, m\,..\,m')$.

4.32 Show that Theorems 4.5.3 and 4.5.4 hold for a.r.m.p. systems.

4.33 This exercise considers an alternative proof of Theorem 4.5.4. Show that if $\mathcal{R}$ is an a.m.p. system and G is a group of processes with $|G| \geq 2$, then for all points (r, m) in $\mathcal{R}$, we have that $(r, 0)$ is G-reachable from (r, m). Use this observation, together with the fact that the same formulas are common knowledge among G at two states one of which is G-reachable from the other (Exercise 2.7), to provide an alternative proof of Theorem 4.5.4.

4.34 Prove the claim in the discussion of mutual exclusion in Section 4.5; i.e., for $j = 1, \ldots, k-1$, prove:

(a) $(\mathcal{I}, r, m_j) \models K_{i_j} K_{i_{j+1}} \neg cs_{i_{j+1}}$,

(b) $(\mathcal{I}, r, m_{j+1}) \models \neg K_{i_{j+1}} \neg cs_{i_{j+1}}$.

Notes

The first person to talk in terms of ascribing mental qualities such as knowledge and belief to machines was McCarthy [1979]. Newell [1982] also talks about analyzing systems at the "knowledge level," although not quite in the way we do here. The general framework presented here for ascribing knowledge in multi-agent systems

originated with Halpern and Moses [1990] and Rosenschein [1985]. Slight variants of this framework were also introduced by Fischer and Immerman [1986], Halpern and Fagin [1989], Parikh and Ramanujam [1985], and Rosenschein and Kaelbling [1986]. Our presentation here is based on that of [Halpern and Fagin 1989], where the reader is referred for more details and further discussion. (We remark that Halpern and Fagin allowed π to depend on the point, not just the global state.) Many other approaches to modeling distributed systems have been described in the literature. Some of the better-known approaches include those of Hoare [1985], Lamport [1986], Lynch and Fischer [1981], Lynch and Tuttle [1989], Milner [1980], and Pratt [1985].

We have made a number of significant assumptions in the way we have chosen to model multi-agent systems. We have already discussed to some extent the implications of having time range over the natural numbers. One implication that we did not discuss is that it assumes that time is linear—there is a unique next step—rather than "branching," where there may be several possible next steps. This choice has been discussed in the philosophical literature in some detail (see, for example, [Thomason 1984]); its implications for computer science have been discussed by Clarke, Emerson, and Sistla [1986], Emerson and Halpern [1986], Halpern and Vardi [1989], and Lamport [1980, 1985], among others. We could imagine that instead of time being linear (as it is in a run), it is branching. We can then consider appropriate *branching* temporal operators; this is done, for example, by Emerson and Halpern [1985, 1986]. Many other temporal operators besides the ones we have used here are possible. In particular, all the operators we have used here involve the future; we could also have past-time operators, as is done, for example, by Lichtenstein, Pnueli, and Zuck [1985]. An excellent introduction to temporal logic can be found in [Manna and Pnueli 1992].

Another assumption we have made regarding time is that it is discrete and ranges over the natural numbers. Our framework changes very little if we allow continuous time. Knowledge-based analyses in models in which time is assumed to be continuous, ranging over the real numbers, are presented in Brafman, Latombe, Moses, and Shoham [1994] and in Moses and Bloom [1994].

Besides our assumptions about time, we have assumed that it makes sense to talk about the global state of the system, and that the set of agents is fixed and their names $(1, \ldots, n)$ are commonly known. While these assumptions can be relaxed, doing so adds a number of complexities. Lamport [1985] and Pratt [1982] present some arguments against the existence of global states. Moses and Roth (see [Roth 1989]) and Grove and Halpern [1991] generalize the framework presented here so that it can deal with the case where the set of agents is not fixed and is not common knowledge.

Our examples of systems were taken from a number of different sources. The notion of a knowledge base that is told facts about the world and is asked queries was first formalized by Levesque [1984a]. Halpern and Vardi [1991b] were the first to model knowledge bases using our notion of system. Making sense out of "all the KB knows is κ" can be quite difficult, although it is straightforward if κ is a propositional formula, as was the case in Proposition 4.4.2. See [Halpern 1993b], [Halpern and Moses 1984], [Lakemeyer 1993], [Levesque 1990], [Parikh 1991], [Stark 1981], and [Vardi 1985] for further discussion of this issue.

Merritt and Taubenfeld [1991] consider a type of system we have not discussed here: a *shared-memory* system. As the name suggests, in such a system, all processes have access to a shared memory. Merritt and Taubenfeld present a model for knowledge in shared-memory systems and show how a number of well-known results regarding such systems can be generalized by using statements involving the difficulty of attaining particular states of knowledge in this setting. Parikh and Krasucki [1992] define the notion of a *level of knowledge* as a set of formulas of the form $K_i K_j \ldots \varphi$ that are satisfied at a given point. They study the attainable levels of knowledge under various assumptions regarding the synchrony or asynchrony of the system.

The notion of game tree is standard in the economics literature. We have provided only a cursory discussion of game trees here. For more details, and a far more detailed introduction to ideas of game theory, see, for example, [Fudenberg and Tirole 1991] and [Rasmusen 1989]. The two games discussed here are slight variations of ones considered by Rasmusen [1989]. The formal definition of synchronous systems and of systems where agents have perfect recall is taken from [Halpern and Vardi 1989]. For a discussion of perfect recall in the context of game theory, see [Fudenberg and Tirole 1991]. Our discussion of a.m.p. systems is based on the work of Chandy and Misra [1986] (although their definitions have been slightly modified to fit our framework, and our presentation follows closely that of Halpern [1987]).

Lemma 4.5.2 and Theorem 4.5.3 are due to Chandy and Misra [1986], as well as the application to the mutual exclusion problem. Extensions to Theorem 4.5.3 were proved by Hadzilacos [1987] and Mazer [1990]. Lamport [1978] discusses the $\xrightarrow{r}$ causality relation. The fact that common knowledge cannot be gained (or lost) in many multi-agent systems was first shown by Halpern and Moses [1990]; the observation that, in particular, this is so in a.m.p. systems (Theorem 4.5.4) is due to Chandy and Misra [1986]. Hadzilacos [1987] considers a number of requirements that we might impose on a.m.p. systems, and examines the properties of knowledge that arise as a result of imposing these requirements.

The notion of a consistent cut described in Exercise 4.28 was introduced by Lamport [1978]. People who feel that global states are inappropriate often consider consistent cuts to be a more reasonable alternative. Panangaden and Taylor [1992] present a definition of knowledge with respect to consistent cuts rather than points. Exercise 4.19(a) shows what can be viewed as an undesirable property of distributed knowledge: We can have a stable formula φ for which $D_G\varphi$ is not stable. Moses and Bloom [1994] define an alternative notion they call *inherent knowledge*, which they argue is more appropriate than distributed knowledge in systems that are not synchronous. They use this notion to study real-time systems, where they generalize Proposition 4.4.3 and use the more general version to discuss clock synchronization.

Chapter 5

Protocols and Programs

Think globally, act locally.

René Dubos

In our discussion of runs in Chapter 4 we avoided consideration of where the runs came from. Starting in some initial global state, what causes the system to change state? Intuitively, it is clear that this change occurs as a result of *actions* performed by the agents and the environment. Furthermore, the agents typically perform their actions deliberately, according to some *protocol*, which is often represented as a *program*. The study of protocols and programs is the subject of this chapter.

5.1 Actions

We already saw several examples of actions taken by agents in multi-agent systems. For example, in message-passing systems (Section 4.4.5), the actions include sending and receiving messages (and possibly some unspecified internal actions). In the games G_1 and G_2 of Section 4.4.2, the actions were a_1, a_2, b_1, and b_2. In general, we assume that for each agent i there is a set ACT_i of actions that can be performed by i. For example, in a distributed system, an action $\mathsf{send}(x, j, i)$—intuitively, this action corresponds to i sending j the value of variable x—might be in ACT_i if x is a local variable of i. On the other hand, if x is not a local variable of i, then it would usually be inappropriate to include $\mathsf{send}(x, j, i)$ in ACT_i.

In keeping with our policy of viewing the environment as an agent (albeit one whose state of knowledge is not of interest), we allow the environment to perform

actions from a set ACT_e. In message-passing systems, it is perhaps best to view message delivery as an action performed by the environment. If we consider a system of sensors observing a terrain, we may want to view a thunderstorm as an action performed by the environment. For both the agents and the environment, we allow for the possibility of a special *null* action Λ, which corresponds to the agents or the environment performing no action.

Knowing which action was performed by a particular agent is typically not enough to determine how the global state of the system changes. Actions performed simultaneously by different agents in a system may interact. If two agents simultaneously pull on opposite sides of a door, the outcome may not be easily computed as a function of the outcomes of the individual actions when performed in isolation. If two processes try simultaneously to write a value into a register, it is again not clear what will happen. To deal with potential interaction between actions, we consider *joint actions*. A joint action is a tuple of the form $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$, where a_e is an action performed by the environment and a_i is an action performed by agent i.

How do joint actions cause the system to change state? We would like to associate with each joint action $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$ a *global state transformer* $\mathcal{T}$, where a global state transformer is simply a function mapping global states to global states, i.e., $\mathcal{T} : \mathcal{G} \rightarrow \mathcal{G}$. Joint actions cause the system to change state via the associated global state transformers; if the system is in global state s when the action $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$ is being performed, then the system changes its state to $\mathcal{T}(s)$. Thus, whenever we discuss actions we will also have a mapping τ that associates with each joint action $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$, a global state transformer $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$. The mapping τ is called the *transition function*. Note that our definition requires that $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(s_e, s_1, \ldots, s_n)$ be defined for each joint action $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$ and each global state $(s_e, s_1, \ldots, s_n)$. In practice, not all joint actions and all global states are going to be of interest when we analyze a multi-agent system, since certain combinations of actions or certain combinations of local states will never actually arise. In such cases, we can let $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(s_e, s_1, \ldots, s_n)$ be defined arbitrarily.

Example 5.1.1 Let us return to the bit-transmission problem (Example 4.1.1). Recall that the sender S either sends its bit or does nothing. Thus, we can take ACT_S, the set of S's actions, to be {sendbit, Λ}. Similarly, the set ACT_R is {sendack, Λ}. The environment determines whether a message is delivered or lost. Recall that we assumed that all messages are either received in the same round that they are sent, or are lost. Thus, we view the environment as nondeterministically performing one of the four actions of the form (a, b), where a is either $\mathsf{deliver}_S(\mathit{current})$ or Λ_S, while b is either $\mathsf{deliver}_R(\mathit{current})$ or Λ_R. For example, if the environment performs

an action of the form $(\Lambda_S, \mathsf{deliver}_R(current))$, then R receives whatever message S sends in that round, if there is one, but S does not receive any message, and if R did send a message in that round, then that message is lost. It is easy to describe formally the effect of each joint action on the global state so as to capture the intuitions just described. We leave the details to the reader (Exercise 5.1). ∎

Example 5.1.2 In the previous example, the environment could either deliver the message currently being sent by either S or R, or it could lose it altogether. In the asynchronous message-passing systems considered in Section 4.4.6, the environment has a wider repertoire of possible behaviors. For example, the environment can decide to deliver a message an arbitrary number of rounds after it has been sent. It is also useful to think of the environment in an a.m.p. system as doing more than just deciding when messages will be delivered. Recall that in a.m.p. systems we make no assumptions on the relative speeds of processes. This means that there may be arbitrarily long intervals between actions taken by processes. One way to capture this assumption is to allow the environment to decide when the process is allowed to take an action.

More formally, in an a.m.p. system we assume that ACT_e consists of actions a_e of the form $\langle \mathsf{a}_{e1}, \ldots, \mathsf{a}_{en} \rangle$, where a_{ei} is either $\mathsf{deliver}_i(current, j)$, $\mathsf{deliver}_i(\mu, j)$, go_i, or nogo_i. (The reader should note that i, j, and μ are parameters, where i and j range over processes, and μ ranges over messages, whereas *current* is a dummy parameter.) Intuitively, $\mathsf{deliver}_i(current, j)$ has the same effect as $\mathsf{deliver}_R(current)$ in Example 5.1.1: any message sent by j to i in that round is received; $\mathsf{deliver}_i(\mu, j)$ means that i will receive the message μ previously sent by j; go_i means that process i is allowed to perform an action (either an internal action or sending a message); finally, nogo_i means that i will neither receive a message nor perform an action. The set ACT_i of possible actions for process i consists of send actions of the form $\mathsf{send}(\mu, j)$ and all the internal actions in INT_i.

Recall that in Chapter 4 we took the state of each process in an a.m.p. system to be its history, and said that the environment's state records the events that have taken place, but we did not describe the environment's state in detail. Now that we have formally defined what actions can be performed in a.m.p. systems, we can take the environment's state to be the sequence of joint actions performed thus far. Recall that a history is a sequence whose first element is an initial state and whose later elements consist of nonempty sets with events of the form $send(\mu, j, i)$, $receive(\mu, j, i)$, or $int(\mathsf{a}, i)$, where μ is a message and a is an internal action. We say that the event $send(\mu, j, i)$ *corresponds* to the action $\mathsf{send}(\mu, j)$ by process i, and the event $int(\mathsf{a}, i)$ *corresponds* to the internal action a by process i. The transition

function τ simply updates the processes' and the environment's local states to reflect the actions performed. Suppose $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$, and $\mathsf{a}_e = (\mathsf{a}_{e1}, \ldots, \mathsf{a}_{en})$. Then $(s'_e, s'_1, \ldots, s'_n)$ must satisfy the following constraints, for $i = 1, \ldots, n$:

- s'_e is the result of appending $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$ to s_e,
- if $\mathsf{a}_{ei} = \mathsf{go}_i$ and a_i is an internal action or a send action $\mathsf{send}(\mu, j)$, then s'_i is the result of appending the event corresponding to a_i to the history s_i,
- if $\mathsf{a}_{ei} = \mathsf{deliver}_i(\mathit{current}, j)$, $\mathsf{a}_{ej} = \mathsf{go}_j$, and a_j is $\mathsf{send}(\mu, i)$, then s'_i is the result of appending $\mathit{receive}(\mu, j, i)$ to s_i,
- if $\mathsf{a}_{ei} = \mathsf{deliver}_i(\mu, j)$, then s'_i is the result of appending $\mathit{receive}(\mu, j, i)$ to s_i,
- in all other cases, $s'_i = s_i$.

Notice how the actions in a joint tuple interact. For example, unless $\mathsf{a}_{ei} = \mathsf{go}_i$, the effect of a_i is nullified; and in order for a message sent by j to i to be received by i in the current round, we must have both $\mathsf{a}_{ej} = \mathsf{go}_j$ and $\mathsf{a}_{ei} = \mathsf{deliver}_i(\mathit{current}, j)$. We chose message delivery to be completely under the control of the environment. We could instead assume that when the environment chooses to deliver a message from i to j, it puts it into a buffer (which is a component of its local state). In this case, i would receive a message only if it actually performed a receive action. We have chosen the simpler way of modeling message delivery, since it suffices for our examples. ∎

These examples should make it clear how much freedom we have in choosing how to model a system. The effect of a joint action will, of course, be very dependent on our choice. For example, in the bit-transmission problem, we chose to record in the local state of S only whether or not S has received an *ack* message, and not how many *ack* messages S receives; the delivery of an *ack* message may have no effect on S's state. If we had chosen instead to keep track of the number of messages S received, then every message delivery would have caused a change in S's state. Ideally, we choose a model that is rich enough to capture all the relevant details, but one that makes it easy to represent state transitions. As Example 5.1.2 shows, by representing a process's state in an a.m.p. system as its history, modeling the effect of a joint action becomes quite straightforward.

5.2 Protocols and Contexts

Agents usually perform actions according to some *protocol*, which is a rule for selecting actions. In Example 4.1.1, the receiver's protocol involves sending an *ack* message after it has received a bit from the sender.

Intuitively, a *protocol* for agent i is a description of what actions agent i may take, as a function of her local state. We formally define a *protocol* P_i for agent i to be a function from the set L_i of agent i's local states to nonempty sets of actions in ACT_i. The fact that we consider a set of possible actions allows us to capture the possible nondeterminism of the protocol. Of course, at a given step of the protocol, only one of these actions is actually performed; the choice of action is nondeterministic. A *deterministic* protocol is one that maps states to actions, i.e., it prescribes a unique action for each local state. Formally, P_i is deterministic if $|P_i(s_i)| = 1$ for each local state $s_i \in L_i$. We remark that if P_i is deterministic, then we typically write $P_i(s_i) = \mathsf{a}$ rather than $P_i(s_i) = \{\mathsf{a}\}$. If we had wanted to consider probabilistic protocols (which we do not here, because it would only complicate the exposition), we would need to put a probability distribution on the set of actions that an agent can perform at a given state. This would then generate a probability space on the set of possible runs of the protocol.

Just as it is useful to view the environment as performing an action, it is also useful to view the environment as running a protocol. We define a protocol for the environment to be a function from L_e to nonempty subsets of ACT_e. For example, in a message-passing system, we can use the environment's protocol to capture the possibility that messages are lost or that messages may be delivered out of order. If all the agents and the environment follow deterministic protocols, then there is only one run of the protocol for each initial global state. In most of our examples, the agents follow deterministic protocols, but the environment does not.

While our notion of protocol is quite general, there is a crucial restriction: a protocol is a function on *local* states, rather than a function on *global* states. This captures our intuition that all the information that the agent has is encoded in his local state. Thus, what an agent does can depend only on his local state, and not on the whole global state. Our definition of protocol is so general that we allow protocols that are arbitrary functions on local states, including ones that cannot be computed. Of course, in practice we are typically interested in *computable* protocols. These are protocols that are computable functions, i.e., protocols for which there is an algorithm that takes a local state as input and returns the set of actions prescribed by the protocol in that state. (A full formalization of this notion is beyond the scope of this book; see the notes at the end of the chapter.)

Processes do not run their protocols in isolation; it is the combination of the protocols run by all agents that causes the system to behave in a particular way. We define a *joint protocol* P to be a tuple $(P_1, \ldots, P_n)$ consisting of protocols P_i, for each of the agents $i = 1, \ldots, n$. Note that while we did include the environment's action in a joint action, we do not include the environment's protocol in a joint protocol. This is because of the environment's special role; we usually design and analyze the agents' protocols, taking the environment's protocol as a given. In fact, when designing multi-agent systems, the environment is often seen as an *adversary* who may be trying to cause the system to behave in some undesirable way. In other words, the joint protocol P and the environment protocol P_e can be viewed as the strategies of opposing players.

The joint protocol P and the environment's protocol prescribe the behavior of all "participants" in the system and therefore, intuitively, should determine the complete behavior of the system. On closer inspection, the protocols describe only the actions taken by the agents and the environment. To determine the behavior of the system, we also need to know the "context" in which the joint protocol is executed. What does such a context consist of? Clearly, the environment's protocol P_e should be part of the context, since it determines the environment's contribution to the joint actions. In addition, the context should include the transition function τ, because it is τ that describes the results of the joint actions. Furthermore, the context should contain the set $\mathcal{G}_0$ of *initial* global states, because this describes the state of the system when execution of the protocol begins. These components of the context provide us with a way of describing the environment's behavior at any single step of an execution.

There are times when we wish to consider more global constraints on the environment's behavior, ones that are not easily captured by P_e, τ, and $\mathcal{G}_0$. To illustrate this point, recall from Example 5.1.2 that in an a.m.p. system, we allow the environment to take actions of the form $(\mathsf{a}_{e1}, \ldots, \mathsf{a}_{en})$, where a_{ei} is one of nogo_i go_i, $\mathsf{deliver}_i(\mathit{current}, j)$, or $\mathsf{deliver}_i(\mu, j)$. In an a.m.p. system, we can think of the environment's protocol as prescribing a nondeterministic choice among these actions at every step, subject to the requirement that a message is delivered only if it has been sent earlier, but not yet delivered. Now suppose we consider an a.r.m.p. system, where all message delivery is taken to be reliable. Note that this does not restrict the environment's actions in any given round; the environment can always postpone message delivery to a later round. The most straightforward way to model an a.r.m.p. is to leave the environment's protocol unchanged, and place an additional restriction on the acceptable behaviors of the environment. Namely, we require that all messages sent must eventually be delivered by the environment.

There are a number of ways that we could capture such a restriction on the environment's behavior. Perhaps the simplest is to specify a condition Ψ on runs, one that tells us which ones are "acceptable." Formally, Ψ is a set of runs; $r \in \Psi$ if r satisfies the condition Ψ. Notice that while the environment's protocol can be thought of as describing a restriction on the environment's behavior at any given point in time, the reliable delivery of messages is a restriction on the environment's "global" behavior, namely, on the acceptable infinite behaviors of the environment. Indeed, often the condition Ψ can be characterized by one (or a collection of) formulas of temporal logic, and the runs in Ψ are those that satisfy these formulas. For example, to specify reliable message-passing systems, we could use the condition $Rel = \{r \mid$ all messages sent in r are eventually received$\}$. Let $send(\mu, j, i)$ be a proposition that is interpreted to mean "message μ is sent to j by i" and let $receive(\mu, i, j)$ be a proposition that is interpreted to mean "message μ is received from i by j." Then a run r is in Rel precisely if $\Box(send(\mu, j, i) \Rightarrow \Diamond receive(\mu, i, j))$ holds at $(r, 0)$ (and thus at every point in r) for each message μ and processes i, j. Another condition of interest is *True*, the condition consisting of all runs; this is the appropriate condition to use if we view all runs as "good."

Formally, we define a *context* γ to be a tuple $(P_e, \mathcal{G}_0, \tau, \Psi)$, where $P_e : L_e \rightarrow 2^{ACT_e} - \{\emptyset\}$ is a protocol for the environment, $\mathcal{G}_0$ is a nonempty subset of $\mathcal{G}$, τ is a transition function, and Ψ is a condition on runs. (Notice that by including τ in the context, we are also implicitly including the sets $L_e, L_1, \ldots, L_n$ of local states as well as the sets $ACT_e, ACT_1, \ldots, ACT_n$ of actions, since the set of joint actions is the domain of τ and the set of global states is the domain of the transition functions yielded by τ. To minimize notation, we do not explicitly mention the state sets and action sets in the context. We shall, however, refer to these sets and to the set $\mathcal{G} = L_e \times L_1 \times \cdots \times L_n$ of global states as if they were part of the context.) It is only in a context that a joint protocol describes the behavior of a system. As we shall see later on, the combination of a context γ and a joint protocol P for the agents uniquely determines a set of runs, which we shall think of as the system representing the execution of the joint protocol P in the context γ.

Contexts provide us with a formal way to capture our assumptions about the systems under consideration. We give two examples of how this can be done here; many others appear in the next few chapters.

Example 5.2.1 In the bit-transmission problem (Example 4.1.1) and in a.m.p. systems (Example 5.1.2), we assumed that the environment keeps track of the sequence of joint actions that were performed. We can formalize this in terms of contexts. We say that $(P_e, \mathcal{G}_0, \tau, \Psi)$ is a *recording context* if

1. the environment's state is of the form $\langle \ldots, h, \ldots \rangle$, where h is a sequence of joint actions,

2. in all global states in $\mathcal{G}_0$, the sequence h of joint actions in the environment's state is the empty sequence $\langle\,\rangle$ (so that no actions have been performed initially), and

3. if $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$, then we require that the sequence h' of joint actions that appears in s'_e is the result of appending $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)$ to the corresponding sequence h of s_e.

For another example, consider message-passing systems, as discussed in Section 4.4.5. Fix a set Σ_i of initial states for process i, a set INT_i of internal actions for process i, and a set MSG of messages. A context $(P_e, \mathcal{G}_0, \tau, \Psi)$ is a *message-passing context* (over Σ_i, INT_i, and MSG, for $i = 1, \ldots, n$) if

1. process i's actions are sets consisting of elements of the form $\mathsf{send}(\mu, j)$ for $\mu \in MSG$, or a for $\mathsf{a} \in INT_i$,

2. process i's local states are histories,

3. for every global state $(s_e, s_1, \ldots, s_n) \in \mathcal{G}_0$, we have that $s_i \in \Sigma_i$ for $i = 1, \ldots, n$, and

4. if $\tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$, then we require that either $s'_i = s_i$ or s'_i is the result of appending to s_i the set consisting of the events corresponding to the actions in a_i, perhaps together with some receive events that correspond to messages that were sent earlier to i by some process j. Intuitively, the state s'_i is the result of appending to s_i the additional events that occurred from process i's point of view in the most recent round. These consist of the actions performed by i, together with the messages received by i. We allow $s'_i = s_i$ to accommodate the possibility that the environment performs a nogo_i action, as in Example 5.1.2.

Notice that we have placed no restrictions on P_e, Ψ, or the form of the environment's local states here, although in practice we often take message-passing contexts to be recording contexts as well. (In particular, as we shall see in Example 5.2.4, this is the case for contexts that capture a.m.p. systems.) ∎

In many cases we have a particular collection Φ of primitive propositions and a particular interpretation π for Φ over $\mathcal{G}$ in mind when we define a context. Just as

we went from systems to interpreted systems, we can go from contexts to *interpreted contexts*; an interpreted context is a pair (γ, π) consisting of a context γ and an interpretation π. (We do not explicitly include Φ here, just as we did not in the case of interpreted systems and Kripke structures.)

We have already observed that when describing a system, we often have some flexibility in our choice of global states. We also have flexibility in describing the other components of a context. We typically think of $\mathcal{G}_0$ as describing the initial conditions, while τ and P_e describe the system's local behavior, and Ψ describes all other relevant aspects of the environment's behavior. To describe the behavior of the system we have to decide what the actions performed by the environment are (this is part of P_e) and how these actions interact with the actions of the agents (this is described by τ).

There is often more than one way in which this can be done. For example, we chose earlier to model message delivery by an explicit deliver action by the environment rather than as the direct result of a send action by the agents. Although we motivated the condition Ψ by the need to be able to capture global aspects of the environment's behavior, we have put no constraints on the condition Ψ. As a result, it is possible (although not advisable) to place much of the burden of determining the initial conditions and local behavior of the environment on the choice of Ψ. Thus, for example, we could do away with $\mathcal{G}_0$ altogether, and have Ψ consist only of runs r whose initial state would have been in $\mathcal{G}_0$. In a well-chosen context we would expect the components to be orthogonal. In particular, we expect that P_e would specify local aspects of the environment's protocol, and that Ψ would capture the more global properties of the environment's behavior over time (such as "all messages are eventually delivered"). In Theorem 7.2.4 of Chapter 7, we shall see an example of how, in order to prove general properties of contexts, we sometimes need to make further restrictions on the condition Ψ to ensure that it does not interact with the other components of the context in undesirable ways.

We can now talk about the runs of the protocol in a given context. We say that a run r is *consistent with a joint protocol* $P = (P_1, \ldots, P_n)$ *in context* $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ if

1. $r(0) \in \mathcal{G}_0$ (so $r(0)$ is a legal initial state),

2. for all $m \geq 0$, if $r(m) = (s_e, s_1, \ldots, s_n)$, then there is a joint action $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n) \in P_e(s_e) \times P_1(s_1) \times \cdots \times P_n(s_n)$ such that $r(m+1) = \tau(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n)(r(m))$ (so $r(m+1)$ is the result of transforming $r(m)$ by a

joint action that could have been performed from $r(m)$ according to P and P_e), and

3. $r \in \Psi$ (so that, intuitively, r conforms to the restriction made by Ψ).

Thus, r is consistent with P in context γ if r is a possible behavior of the system under the actions prescribed by P in γ. We say that r is *weakly consistent* with P in context γ if it satisfies the first two of the three conditions required for consistency, but is not necessarily in Ψ. Intuitively, this means that r is consistent with the step-by-step behavior of P in context γ. We do not use weak consistency in this chapter, but the concept will prove to be useful in Chapter 7. Note that while we are always guaranteed to have runs that are weakly consistent with P in γ, it is possible that there is no run r that is consistent with P in γ. This happens precisely if there is no run in Ψ that is weakly consistent with P in γ. In such a case we say that P is *inconsistent* with γ; otherwise, P is *consistent* with γ. For example, all joint protocols are inconsistent with a context γ in which Ψ contains no run whose initial state is in $\mathcal{G}_0$. We take a situation where the joint protocol is inconsistent with the context as an indication of bad modeling, and always assume that the joint protocols we are considering are consistent with their contexts.

We say that a system $\mathcal{R}$ (resp., an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$) is *consistent with a protocol P in context γ* (resp., interpreted context (γ, π)) if every run $r \in \mathcal{R}$ is consistent with P in γ. Because systems are nonempty sets of runs, this requires that P be consistent with γ. Typically, there will be many systems consistent with a protocol in a given context. However, when we think of running a protocol in a given context, we usually have in mind the system where all possible behaviors of the protocol are represented. We define $\mathbf{R}^{rep}(P, \gamma)$ to be the system consisting of all runs consistent with P in context γ, and call it the *system representing protocol P in context γ*. Similarly, we say that $\mathbf{I}^{rep}(P, \gamma, \pi) = (\mathbf{R}^{rep}(P, \gamma), \pi)$ is *the interpreted system representing P in interpreted context* (γ, π). Notice that $\mathcal{R}$ is consistent with P in γ iff $\mathcal{R} \subseteq \mathbf{R}^{rep}(P, \gamma)$, so that $\mathbf{R}^{rep}(P, \gamma)$ is the maximal system consistent with P in γ.

While we are mainly interested in the (interpreted) system representing P in a given (interpreted) context, there is a good reason to look at some of the other systems consistent with P in that context as well. We may start out considering one context γ, and then be interested in what happens if we restrict γ in some way. For example, we may wish to restrict attention to a particular set of initial states, or may wish to see what happens if we limit the behavior of the environment. The following definitions make precise the idea of "restricting" a context γ. We say that the environment protocol P'_e is a *restriction* of P_e, written $P'_e \sqsubseteq P_e$, if $P'_e(s_e) \subseteq P_e(s_e)$ holds for every

local state $s_e \in L_e$. We say that a context $\gamma' = (P'_e, \mathcal{G}'_0, \tau, \Psi')$ is a *subcontext* of $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$, denoted by $\gamma' \sqsubseteq \gamma$, if (a) $P'_e \sqsubseteq P_e$, (b) $\mathcal{G}'_0 \subseteq \mathcal{G}_0$, and (c) $\Psi' \subseteq \Psi$. Similarly, we say that an interpreted context (γ', π) is a subcontext of (γ, π) if γ' is a subcontext of γ. As the following lemma shows, the systems consistent with P in context γ are precisely those that represent P in some subcontext of γ.

Lemma 5.2.2 *$\mathcal{R}$ is consistent with P in context γ if and only if $\mathcal{R}$ represents P in some subcontext $\gamma' \sqsubseteq \gamma$.*

Proof If $\mathcal{R}$ represents P in the subcontext $\gamma' \sqsubseteq \gamma$, then it is easy to see that every run of $\mathcal{R}$ must be consistent with P in context γ. Thus, $\mathcal{R}$ is consistent with P in context γ. Conversely, suppose $\mathcal{R}$ is consistent with P in context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$. Let $\gamma_{\mathcal{R}} = (P_e, \mathcal{G}_0, \tau, \mathcal{R})$. Since $\mathcal{R}$ must be a subset of Ψ, it follows that $\gamma_{\mathcal{R}} \sqsubseteq \gamma$. It is easy to see that $\mathcal{R}$ represents P in γ_R. ∎

Example 5.2.3 What are the sender's and receiver's protocols in our bit-transmission example? Recall from Example 4.1.1 that the sender S is in one of four states—0, 1, (0, *ack*), or (1, *ack*)—and its possible actions are sendbit and Λ. Its protocol P_S^{bt} is quite straightforward to describe:

- $P_S^{bt}(0) = P_S^{bt}(1) = \mathsf{sendbit}$
- $P_S^{bt}(0, ack) = P_S^{bt}(1, ack) = \Lambda$.

This is a formalization of the informal description that we gave in Chapter 4; the sender sends its bit to the receiver until it receives an acknowledgment, at which point it stops sending (and does nothing).

Recall that the receiver R is in one of three states—λ, 0, or 1—and its possible actions are sendack and Λ. The receiver's protocol P_R^{bt} is

- $P_R^{bt}(\lambda) = \Lambda$
- $P_R^{bt}(0) = P_R^{bt}(1) = \mathsf{sendack}$.

We now need to describe a context for the joint protocol $P^{bt} = (P_S^{bt}, P_R^{bt})$. Recall that the environment's state is a sequence that records the events taking place in the system, and the environment's four actions are of the form (a, b), where a is either $\mathsf{deliver}_S(current)$ or Λ_S, while b is either $\mathsf{deliver}_R(current)$ or Λ_R. We view the environment as following the nondeterministic protocol P_e^{bt}, according to which, at every state, it nondeterministically chooses to perform one of these four actions.

The set $\mathcal{G}_0$ of initial states is the cross product $\{\langle\rangle\} \times \{0, 1\} \times \{\lambda\}$, i.e., initially the environment's and the receiver's states record nothing, and the sender's state records the input bit. The context capturing the situation described in Example 4.1.1 is $\gamma^{bt} = (P_e^{bt}, \mathcal{G}_0, \tau, \mathit{True})$, where the definition of τ was left as an exercise to the reader (Exercise 5.1). Moreover, the system $\mathcal{R}^{bt}$ described in Example 4.1.1 is precisely $\mathbf{R}^{rep}(P^{bt}, \gamma^{bt})$.

When analyzing the protocol of Example 4.1.1, we may want to restrict our analysis to the situation in which the system's communication channel is *fair* in the sense that every message sent infinitely often is eventually delivered. In our case this would imply that R eventually does receive S's bit, and S eventually receives an *ack* message sent by R. One motivation for restricting to runs that are fair in this sense comes from the following observation: if we assume that there is a positive probability of at least $\alpha > 0$ of a given message being delivered in any given round (if there is a message that can be delivered), and that these probabilities are independent from round to round, then the set of runs that are fair in this sense has probability one. To capture this, we use the condition *Fair*, where a run is in *Fair* exactly if every message that is repeatedly sent in the run is eventually delivered. Thus, a run is in *Fair* if it satisfies the formula $(\Box\Diamond \mathit{sendbit} \Rightarrow \Diamond \mathit{recbit}) \wedge (\Box\Diamond \mathit{sendack} \Rightarrow \Diamond \mathit{recack})$. Let γ_{fair}^{bt} be the context we get by replacing *True* in γ^{bt} by *Fair*. The system $\mathcal{R}^{fair}$ that represents P^{bt} in a fair setting is then $\mathbf{R}^{rep}(P^{bt}, \gamma_{fair}^{bt})$. ∎

Example 5.2.4 Consider a.m.p. systems over $\Sigma_1, \ldots, \Sigma_n$, $INT_1, \ldots, INT_n$, and *MSG*. As we now show, these can be characterized by a particular context of the form $(P_e^{amp}, \mathcal{G}_0, \tau, \mathit{True})$. This context is both a recording context and a message-passing context (as defined in Example 5.2.1). All we need to do to complete its description is to describe the environment's actions and local states. Since it is a recording context, the environment's state must include the sequence of joint actions performed thus far. In fact, here we take the environment's state to be precisely this sequence. $\mathcal{G}_0$ is $\{\langle\rangle\} \times \Sigma_1 \times \cdots \times \Sigma_n$. We discussed in Example 5.1.2 what the environment's actions are and how the transition function τ is defined. Finally, P_e^{amp} simply nondeterministically chooses one of the environment's actions, except that we require that if $\mathsf{deliver}_i(\mu, j)$ is performed, then the message μ must have been sent earlier by i to j, and not yet received. (Note that the environment can determine this by looking at its state, since its state records the sequence of actions that have been performed thus far.) Call this context γ^{amp}.

In what sense does γ^{amp} characterize a.m.p. systems? Suppose we are given a prefix-closed set V_i of histories for process i. As we now show, V_i determines a protocol P_i for process i. If $h \in V_i$ and $\mathsf{a} \in ACT_i$, we denote by $h \cdot a$ the history

that results from appending to h the event a corresponding to the action a. We then define $P_i(h) = \{\mathsf{a} \in ACT_i : h \cdot a \in V_i\}$. Intuitively, $P_i(h)$ consists of all allowable actions according to the set V_i. Let $P^{amp}(V_1, \ldots, V_n)$ be the joint protocol $(P_1, \ldots, P_n)$, that is, $P^{amp}(V_1, \ldots, V_n)$ is the joint protocol that corresponds to the sets $V_1, \ldots, V_n$ of histories. It is not hard to show that $\mathcal{R}(V_1, \ldots, V_n)$ is essentially $\mathbf{R}^{rep}(P^{amp}(V_1, \ldots, V_n), \gamma^{amp})$ (Exercise 5.2). This suggests that the right way to think about the a.m.p. system $\mathcal{R}(V_1, \ldots, V_n)$ is as the system that results when the processes run the joint protocol $P^{amp}(V_1, \ldots, V_n)$ in a setting that captures the appropriate asynchrony.

Notice that if we wanted to consider a.r.m.p. systems rather than a.m.p. systems, we would simply replace the condition *True* in the context γ^{amp} by the condition *Rel* discussed earlier. We may also want to require that the environment follows a *fair schedule*, in the sense that no process is blocked from moving from some point on. Formally, we can capture this by the condition *FS* that holds of a run if there are infinitely many go_i actions, for each process i. Thus, if we add a proposition go_i that is true at a state exactly if a go_i action was performed by the environment in the preceding round, then *FS* can be characterized by the formula $\Box\Diamond go_1 \wedge \ldots \wedge \Box\Diamond go_n$. ∎

Example 5.2.5 Let us reconsider the game-theoretic framework of Section 4.4.2. There, we described systems that model all possible plays of a game by including a run for each path in the game tree. We did not, however, attempt to model the *strategies* of the players, which are the major focus in game theory. A strategy is a function that tells a player which move to choose based on the player's "current information" about the game. (In game theory, such strategies are called *pure*, as opposed to *mixed* strategies, which are probability distributions over pure strategies.) In our model, a player's current information is completely captured by his local state; thus, a strategy for player i is simply a deterministic protocol for player i, i.e., a function from his local state to actions.

Let us consider the game G_1 in Section 4.4.2. What are the possible strategies of player 1 in this game? Because player 1 takes an action only at the first step, he has only two possible strategies: "choose a_1" and "choose a_2." We call these two strategies σ_1 and σ_2. Player 2 has four strategies in the first game, since her choice of action can depend on what player 1 did. These strategies can be described by the pairs $(\mathsf{b}_1, \mathsf{b}_1)$, $(\mathsf{b}_1, \mathsf{b}_2)$, $(\mathsf{b}_2, \mathsf{b}_1)$, and $(\mathsf{b}_2, \mathsf{b}_2)$. The first strategy corresponds to "choose b_1 no matter what." The second strategy corresponds to "choose b_1 if player 1 chose a_1, and choose b_2 if player 1 chose a_2." We can similarly describe the third and fourth strategies. Call these four strategies σ_{11}, σ_{12}, σ_{21}, and σ_{22}. Even though there are eight pairs of strategies (for the two players), there are only four

different plays. For example, the pair (σ_1, σ_{11}) and the pair (σ_1, σ_{12}) result in the same play: $\langle \mathsf{a}_1, \mathsf{b}_1 \rangle$.

Recall that the system $\mathcal{R}_1$ that corresponds to the game G_1 contains four runs, one run for each path in the game tree. The local states of the players essentially consist of sequences of moves in the game. For example, the local state of both players at the start is the empty sequence $\langle\,\rangle$ and their local state after player 1 chooses a_1 is the sequence $\langle \mathsf{a}_1 \rangle$. We would like to define protocols for the players that capture the strategies that they follow; however, there is a difficulty. After player 1 chooses a_1, player 2's local state is $\langle \mathsf{a}_1 \rangle$. Thus, a deterministic protocol would tell player 2 to choose either b_1 or b_2. But in $\mathcal{R}_1$, player 2 chooses b_1 in one run and b_2 in another. Does this mean that player 2 is not following a deterministic protocol? No. Rather, it means that our description of player 2's local state is incomplete; it does not include everything that determines her choice of action.

We now present a system $\mathcal{R}_1'$ that enriches the players' local states so that they include not only the history of the game, but also a representation of the strategy of the player. Thus, the set of local states of player 1 includes states such as $(\sigma_1, \langle\,\rangle)$, $(\sigma_1, \langle \mathsf{a}_1, \mathsf{b}_1 \rangle)$, $(\sigma_2, \langle \mathsf{a}_2 \rangle)$, etc. Similarly, the set of local states of player 2 includes states such as $(\sigma_{11}, \langle\,\rangle)$, $(\sigma_{12}, \langle \mathsf{a}_2 \rangle)$, $(\sigma_{21}, \langle \mathsf{a}_1, \mathsf{b}_1 \rangle)$, etc. Again, all the relevant information in the system is captured by the players' local states, so we can take the environment's state to be the constant λ. There are eight initial states, corresponding to all pairs of strategies, so $\mathcal{G}_0$ consists of these eight states. The actions of the players are a_1, a_2, b_1, b_2, and Λ; the last is the null action and the other actions have the obvious meaning. The environment plays no role here. Its only action is Λ; that is, $P_e(\lambda) = \Lambda$. We leave it to the reader to define the transition function τ in this system formally (see Exercise 5.3). The context $\gamma = (P_e, \mathcal{G}_0, \tau, \mathit{True})$ describes the setting in which this game is played.

We can now define the protocols for the players; these protocols essentially say "choose an action according to your strategy." The protocol P_1 for player 1 is as follows:

- $P_1(\sigma_i, \langle\,\rangle) = \mathsf{a}_i$, for $i \in \{1, 2\}$,
- $P_1(\sigma_i, h) = \Lambda$ if $h \neq \langle\,\rangle$, for $i \in \{1, 2\}$.

The protocol P_2 for player 2 is as follows:

- $P_2(\sigma_{ij}, \langle \mathsf{a}_1 \rangle) = \mathsf{b}_i$, for $i, j \in \{1, 2\}$,
- $P_2(\sigma_{ij}, \langle \mathsf{a}_2 \rangle) = \mathsf{b}_j$, for $i, j \in \{1, 2\}$,
- $P_2(\sigma_{ij}, h) = \Lambda$ if $h \notin \{\langle \mathsf{a}_1 \rangle, \langle \mathsf{a}_2 \rangle\}$, for $i, j \in \{1, 2\}$.

The system $\mathcal{R}'_1$ consists of all runs that start from an initial state and are consistent with the joint protocol $P = (P_1, P_2)$, i.e., $\mathcal{R}'_1 = \mathbf{R}^{rep}(P, \gamma)$.

How does the game G_2 of Section 4.4.2 get modeled in this more refined approach? Again, player 1 has two possible strategies, σ_1 and σ_2. But now player 2 also has only two strategies, which we call σ'_1 and σ'_2. Running σ'_1, player 2 chooses action b_1, and running σ'_2, she chooses b_2. There is no strategy corresponding to σ_{12}, since player 2 does not know what action player 1 performed at the first step, and thus her strategy cannot depend on this action. We can define a system $\mathcal{R}'_2$ that models this game and captures the players' strategies (Exercise 5.3).

The approach to modeling game trees just discussed, where the players' local states contain information about what strategy the player is using, is somewhat more complicated than that discussed in Section 4.4.2. It does, however, offer some advantages. Because it captures the strategies used by the players, it enables us to reason about what players know about each other's strategies, an issue of critical importance in game theory. For example, a standard assumption made in the game-theory literature is that players are *rational*. To make this precise, we say that a strategy σ for player i *(strictly) dominates* a strategy σ' if, no matter what strategy the other players are using, player i gets at least as high a payoff using strategy σ as using strategy σ', and there is some strategy that the other players could use whereby i gets a strictly higher payoff by using σ than by using σ'. According to one notion of rationality, a rational player never uses a strategy if there is another strategy that dominates it. For example, in the game G_1, strategy σ_{12} dominates all other strategies for player 2, so that if player 2 were rational, then she would use σ_{12}.

To reason formally about rationality, we introduce the propositions $rational_i$, for $i = 1, 2$, where $rational_i$ holds at a point if player i's strategy at that point is not dominated by another strategy. For player 1 to know that player 2 is rational means that $K_1(rational_2)$ holds. The players can use their knowledge of rationality to eliminate certain strategies. For example, in the game G_1, if player 1 knows that player 2 is rational, then he knows that she would use the strategy σ_{12}. With this knowledge, σ_1 dominates σ_2 for player 1. Thus, if player 1 is rational, he would then use σ_1. (Notice that if player 1 thinks player 2 is not rational, it may make sense for 1 to use σ_2 instead, since it guarantees a better payoff in the worst case.) It follows that if players 1 and 2 are both rational, and player 1 knows that player 2 is rational, then their joint strategy must be (σ_1, σ_{12}) and the payoff is (3, 4). By way of contrast, even if we assume that rationality is common knowledge in the game G_2 (an assumption that is frequently made by game theorists), it is easy to see that neither players 1 nor player 2 has a dominated strategy, and so no strategy for either player is eliminated because of a rationality assumption. ∎

The previous examples show how we can view a context as a description of a class of systems of interest. The context describes the setting in which a protocol can be run, and by running distinct protocols in the same context we can generate different systems, all of which share the characteristics of the underlying context. We will see several examples of classes of systems described by contexts in Chapter 6.

5.3 Programs

As discussed earlier, a protocol is a function from local states to sets of actions. We typically describe protocols by means of *programs* written in some programming language. Consider the receiver R in Example 4.1.1, which starts sending an *ack* message after it has received a bit from the sender S. This can be described by a program such as "**if** *recbit* **do** sendack" (recall from Example 4.2.1 that *recbit* is a primitive proposition that holds at points where R has received S's message). The essential feature of this statement is that the program selects an action based on the result of a test that depends solely on the local state.

We now describe a simple programming language, which is still rich enough to describe protocols, and whose syntax emphasizes the fact that an agent performs actions based on the result of a test that is applied to her local state. A (*standard*) *program* for agent i is a statement of the form:

case of
  if t_1 **do** a_1
  if t_2 **do** a_2
  ...
end case

where the t_j's are *standard tests* for agent i and the a_j's are actions of agent i (i.e., $\mathsf{a}_j \in ACT_i$). (We call such programs "standard" to distinguish them from the *knowledge-based* programs that we introduce in Chapter 7. We typically omit the **case** statement if there is only one clause.) A standard test for agent i is simply a propositional formula over a set Φ_i of primitive propositions. Intuitively, once we know how to evaluate the tests in the program at the local states in L_i, we can convert this program to a protocol over L_i: at a local state ℓ, agent i nondeterministically chooses one of the (possibly infinitely many) clauses in the **case** statement whose test is true at ℓ, and executes the corresponding action.

We want to use an interpretation π to tell us how to evaluate the tests. However, not just any interpretation will do. We intend the tests in a program for agent i to

be *local*, that is, to depend only on agent i's local state. It would be inappropriate for agent i's action to depend on the truth value of a test that i could not determine from her local state. We say that an interpretation π on the global states in $\mathcal{G}$ is *compatible* with a program Pg_i for agent i if every proposition that appears in Pg_i is local to i in the sense described in Section 4.2; that is, if q appears in Pg_i, the states s and s' are in $\mathcal{G}$, and $s \sim_i s'$, then $\pi(s)(q) = \pi(s')(q)$. If φ is a propositional formula all of whose primitive propositions are local to agent i, and ℓ is a local state of agent i, then we write $(\pi, \ell) \models \varphi$ if φ is satisfied by the truth assignment $\pi(s)$, where $s = (s_e, s_1, \ldots, s_n)$ is a global state such that $s_i = \ell$. Because all the primitive propositions in φ are local to i, it does not matter which global state s we choose, as long as i's local state in s is ℓ. Given a program Pg_i for agent i and an interpretation π compatible with Pg_i, we define a protocol that we denote Pg_i^π by setting

$$\mathsf{Pg}_i^\pi(\ell) = \begin{cases} \{\mathsf{a}_j : (\pi, \ell) \models t_j\} & \text{if } \{j : (\pi, \ell) \models t_j\} \neq \emptyset \\ \{\Lambda\} & \text{if } \{j : (\pi, \ell) \models t_j\} = \emptyset. \end{cases}$$

Intuitively, Pg_i^π selects all actions from the clauses that satisfy the test, and selects the null action Λ if no test is satisfied. In general, we get a nondeterministic protocol, since more than one test may be satisfied at a given state.

Many of the definitions that we gave for protocols have natural analogues for programs. We define a *joint* program to be a tuple $\mathsf{Pg} = (\mathsf{Pg}_1, \ldots, \mathsf{Pg}_n)$, where Pg_i is a program for agent i. We say that an interpretation π is *compatible* with Pg if π is compatible with each of the Pg_i's. From Pg and π we get a joint protocol $\mathsf{Pg}^\pi = (\mathsf{Pg}_1^\pi, \ldots, \mathsf{Pg}_n^\pi)$. We say that an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ *represents* (resp., is *consistent with*) a joint program Pg in the interpreted context (γ, π) exactly if π is compatible with Pg and $\mathcal{I}$ represents (resp., is consistent with) the corresponding protocol Pg^π. We denote the interpreted system representing Pg in (γ, π) by $\mathbf{I}^{rep}(\mathsf{Pg}, \gamma, \pi)$. Of course, this definition only makes sense if π is compatible with Pg. From now on we always assume that this is the case.

Notice that the syntactic form of our standard programs is in many ways more restricted than that of programs in common programming languages such as C or FORTRAN. In such languages, one typically sees constructs such as **for**, **while**, or **if**... **then**... **else**..., which do not have syntactic analogues in our formalism. The semantics of programs containing such constructs depends on the local state containing an implicit *instruction counter*, specifying the command that is about to be executed at the current local state. Since we model the local state of a process explicitly, it is possible to simulate these constructs in our framework by having an explicit variable in the local state accounting for the instruction counter. The local

tests t_j used in a program can then reference this variable explicitly, and the actions a_j can include explicit assignments to this variable. Given that such simulation can be carried out in our framework, there is no loss of generality in our definition of standard programs.

It is easy to see that every protocol is induced by a standard program if we have a rich enough set of primitive propositions (Exercise 5.4). As a result, our programming language is actually more general than many other languages; a program may induce a non-computable protocol. Typically, however, we are interested in programs that induce computable protocols. In fact, standard programs usually satisfy a stronger requirement: they have finite descriptions, and they induce deterministic protocols.

Example 5.3.1 Let us return to the bit-transmission problem yet again. We saw the sender S's protocol in Example 5.2.3. The sender S can be viewed as running the following program BT_S, which uses the proposition *recack* that we introduced in Example 4.2.1:

if $\neg$*recack* **do** sendbit.

(Note that if *recack* holds, then, according to our definitions, the action Λ is selected.)

Similarly, the receiver R can be viewed as running the following program BT_R:

if *recbit* **do** sendack.

Let $\mathsf{BT} = (\mathsf{BT}_S, \mathsf{BT}_R)$. Recall that we gave an interpretation π^{bt} in Example 4.2.1 describing how the propositions in BT_S and BT_R are to be interpreted. It is easy to see that π^{bt} is compatible with BT, and that $\mathsf{BT}^{\pi^{bt}}$ is the joint protocol P^{bt} described in Example 5.2.3 (Exercise 5.5). ∎

5.4 Specifications

When designing or analyzing a multi-agent system, we typically have in mind some property that we want the system to satisfy. Very often we start with a desired property and then design a protocol to satisfy this property. For example, in the bit-transmission problem the desired property is that the sender communicate the bit to the receiver. We call this desired property the *specification* of the system or protocol under consideration. A specification is typically given as a description of the "good" systems. Thus, a specification can be identified with a class of interpreted systems, the ones that are "good." An interpreted system $\mathcal{I}$ *satisfies* a specification σ if it is in the class, that is, if $\mathcal{I} \in \sigma$.

Many specifications that arise in practice are of a special type that we call *run-based*. A run-based specification is a specification that is given as a property of runs. Quite often run-based specifications can be expressed using formulas with temporal operators (with no modal operators for knowledge). A system is said to satisfy a run-based specification if all its runs do. For example, a possible specification for the bit-transmission problem is: "the receiver eventually receives the bit from the sender, and the sender eventually stops sending the bit"; this can be expressed as $\Diamond recbit \wedge \Diamond\Box\neg sendbit$. The truth of this specification can be decided for each run with no consideration of the system in which the run appears.

Although run-based specifications arise often in practice, there are reasonable specifications that are not run based. For example, in the muddy children puzzle, the natural specification of the children's behavior is: "a child says 'Yes' if he knows whether he is muddy, and says 'No' otherwise." This specification is given in terms of the children's knowledge, which depends on the whole system and cannot be determined by considering individual runs in isolation. We view such a specification as a *knowledge-based* specification. More generally, we call a specification that is expressible in terms of epistemic (and possibly other) modal operators a knowledge-based specification. Unlike run-based specifications, which specify properties of runs, knowledge-based specifications specify properties of interpreted systems.

What should it mean for a protocol P to satisfy a specification σ in an interpreted context (γ, π)? We say that P *satisfies* σ in (γ, π) or is *correct* with respect to σ in (γ, π), precisely if the interpreted system representing P in (γ, π) satisfies σ; i.e., if $\mathbf{I}^{rep}(P, \gamma, \pi) \in \sigma$.

We are often interested in the correctness of a protocol not just with respect to one context, but with respect to some collection Γ of contexts. This collection of contexts corresponds to the various settings in which we want to run the protocol. Typically, the contexts in Γ are subcontexts of a single context γ, in the sense defined in Section 5.2. This leads us to consider a stronger notion of correctness: We say P *strongly satisfies* σ in (γ, π), or P is *strongly correct with respect to* σ in (γ, π), if every interpreted system consistent with P in (γ, π) satisfies σ. By Lemma 5.2.2, P is strongly correct with respect to σ in (γ, π) exactly if every interpreted system that represents P in a subcontext γ' of γ satisfies σ. Thus, P is strongly correct with respect to σ in (γ', π) exactly if P is correct with respect to σ in (γ', π) for every subcontext γ' of γ.

There is one important case where correctness and strong correctness coincide: when σ is a run-based specification (Exercise 5.6). This follows from the fact a system is consistent with a protocol if and only if it is a subset of the unique system representing the protocol. In general, of course, correctness and strong correctness

do not coincide. When they do not coincide, it can be argued that strong correctness may be too strong a notion. After all, even if we are interested in proving correctness with respect to certain subcontexts of γ, we are not interested in *all* subcontexts of γ. In practice, however, it is often just as easy to prove strong correctness with respect to a context γ as it is to prove correctness for a restricted set of subcontexts of γ.

As before, all our definitions for protocols have natural analogues for programs. In particular, we say that a program Pg (strongly) satisfies σ in an interpreted context (γ, π) if the protocol Pg^π (strongly) satisfies σ in the interpreted context (γ, π).

Example 5.4.1 Let σ' be the run-based specification for the bit-transmission problem described earlier: $\Diamond recbit \wedge \Diamond\Box\neg sendbit$. In Example 5.3.1, we described a standard program $\mathsf{BT} = (\mathsf{BT}_S, \mathsf{BT}_R)$ for this problem. We also described an interpreted context (γ^{bt}, π^{bt}) for BT. It is easy to see that BT does not satisfy σ' in (γ^{bt}, π^{bt}), for there are runs consistent with BT^π in γ^{bt} in which the messages sent by S are never received by R. As we observed earlier, however, we are often interested in assuming that the communication channel is fair. Recall from Example 5.2.3 that γ^{bt}_{fair} is the result of replacing the condition *True* in γ^{bt} by *Fair*. Thus, γ^{bt}_{fair} differs from γ^{bt} in that it ensures that communication delivery satisfies the fairness condition. It is not hard to verify that BT does indeed satisfy σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$ (Exercise 5.7). Since σ' is a run-based specification, this means that BT strongly satisfies σ' as well. It thus follows that as long as the communication channel is fair, BT works fine.

We can also give a knowledge-based specification for the bit-transmission problem. Let σ'' be the knowledge-based specification: "eventually S knows that R knows the value of the bit, and S stops sending messages when it knows that R knows the value of the bit." Using our language, we can express σ'' as $\Diamond K_S K_R(bit) \wedge \Box(K_S K_R(bit) \Rightarrow \neg sendbit)$. This specification is more abstract than σ', because it does not refer to the manner in which the agents gain their knowledge. It is not hard to see that BT satisfies σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$ (Exercise 5.7). BT, however, does *not* strongly satisfy σ'' in this context. To prove this, let γ^{bt}_{ck} be the context where it is common knowledge that S's initial value is 1, and the communication channel is fair. That is, γ^{bt}_{ck} is just like γ^{bt}_{fair}, except that the only initial state is $(\lambda, 1, \lambda)$. Clearly we have $\gamma^{bt}_{ck} \sqsubseteq \gamma^{bt}_{fair}$. In this context, the sender knows from the outset that the receiver knows the bit. Nevertheless, following BT, the sender would send the bit to the receiver in the first round, and would keep sending messages until it receives an acknowledgment. This does not conform to the requirement made in σ'' that if S knows that R knows the bit, then S does not send a message. It follows

that BT does not satisfy σ'' in $(\gamma_{ck}^{bt}, \pi^{bt})$. An advantage of σ'' is that it can be satisfied without the sender having to send any message in contexts such as $(\gamma_{ck}^{bt}, \pi^{bt})$ in which the value of the initial bit is common knowledge.

Notice that the specification σ'' is not run-based. To tell whether $\Diamond K_S K_R(bit)$ holds, we need to consider the whole system, not just a run. Knowledge-based specifications such as σ'' are quite important in practice. If a system satisfies σ'', then we know that in a certain sense no unnecessary messages are sent; this is an assurance we do not have if we know only that the system satisfies σ'. ∎

Exercises

5.1 Give a formal description of the effect of each joint action as a global state transformer in the system corresponding to the bit-transmission problem described in Examples 4.1.1 and 5.1.1.

5.2 Suppose we are given a prefix-closed set V_i of histories for process i whose initial states are precisely Σ_i, for all processes $i = 1, \ldots, n$. Let $\mathcal{R} = \mathcal{R}(V_1, \ldots, V_n)$ and $\mathcal{R}' = \mathbf{R}^{rep}(P^{amp}(V_1, \ldots, V_n), \gamma^{amp})$. We would like to show that $\mathcal{R} = \mathcal{R}'$. We cannot quite do this, however, since we put no constraints on the form of the environment state in $\mathcal{R}(V_1, \ldots, V_n)$. Show instead that (a) for each run $r \in \mathcal{R}$ there is a run $r' \in \mathcal{R}'$ such that $r_i(m) = r'_i(m)$ holds for all processes i and times m, and, similarly, (b) for each run $r' \in \mathcal{R}'$ there is a run $r \in \mathcal{R}$ such that $r_i(m) = r'_i(m)$ for all processes i and times m.

5.3 This exercise fills in some details in Example 5.2.5.

(a) Define the transition function of the system $\mathcal{R}'_1$.

(b) Give a complete description of the system $\mathcal{R}'_2$.

(c) Show that neither player 1 nor player 2 has any dominated strategies in the game G_2.

5.4 Show that every protocol P_i for agent i over the local state space L_i is induced by some program Pg_i. (Hint: for every local state $s \in L_i$, introduce a local proposition of the form "*state is s*," and have π_i assign **true** to this proposition only in the state s.)

5.5 This exercise discusses aspects of the bit-transmission problem. Show that the interpretation π^{bt} presented in Example 4.2.1 is compatible with the joint program $\mathsf{BT} = (\mathsf{BT}_S, \mathsf{BT}_R)$, and that $\mathsf{BT}^{\pi^{bt}}$ is the joint protocol $P^{bt} = (P_S^{bt}, P_R^{bt})$ described in Example 5.2.3.

5.6 Let σ be a run-based specification, let P be a joint protocol, and let (γ, π) be an interpreted context. Prove that P satisfies σ in (γ, π) if and only if P strongly satisfies σ in (γ, π).

5.7 Using the terminology of Example 5.4.1, prove:

(a) BT strongly satisfies σ' in $(\gamma_{fair}^{bt}, \pi^{bt})$,

(b) BT satisfies σ'' in $(\gamma_{fair}^{bt}, \pi^{bt})$.

Notes

A formal model of actions and protocols was introduced by Halpern and Fagin [1989]. This model is related to that of Shapley [1953], although the latter deals with a stochastic setting in which actions are chosen probabilistically. Our model in this chapter extends that of Halpern and Fagin by adding the notion of contexts and programs. Halpern and Tuttle [1993] discuss how probability can be added to the framework (and the subtleties of doing so); their work is based on Fagin and Halpern's formal model of knowledge and probability [1994].

The reader interested in computability can consult the extensive study by Rogers [1967]. The idea of using knowledge to specify distributed programs was suggested by Halpern and Moses [1990]. The first explicit use of knowledge-based specifications for protocol design seems to be that of Afrati, Papadimitriou, and Papageorgiou [1988]. Sanders [1991] points out a number of differences between knowledge-based specifications and run-based specifications. Halpern [1991] discusses the issues raised by Sanders in some detail, and gives perhaps the first explicit definition of knowledge-based specification. The idea of a knowledge-based specification is certainly implicit in earlier work, including that of Mazer [1991]. Katz and Taubenfeld [1986] develop formal methods for using assertions involving knowledge in verifying that distributed programs satisfy given specifications.

Chapter 6

Common Knowledge and Agreement

Agreement consists in disagreement.

M. A. L. Lucan, *The Civil War*, c. A.D. 50

We can't disagree forever.

J. Geanakoplos and H. Polemarchakis, 1982

The discussion in Chapters 1 and 2 shows that common knowledge plays an important role in the muddy children puzzle. Common knowledge, however, is far more than just a curiosity that arises in puzzles. As we show in this chapter, it is a fundamental notion of group knowledge, one which is relevant in many applications. In particular, we show that common knowledge is a necessary and sometimes even sufficient condition for reaching agreement and for coordinating actions. We illustrate the role of common knowledge by examining three well-known problems from the literature, known as *coordinated attack*, *agreeing to disagree*, and *simultaneous Byzantine agreement*.

Before we turn to specific examples, let us consider the relationship between common knowledge and agreement. How can we capture the fact that two players, say Alice and Bob, agree on some statement, which for simplicity we represent by a formula ψ? Suppose $agree(\psi)$ is a formula that is true at states in which the players have agreed on ψ. While we do not attempt to characterize agreement completely, we expect that if Alice and Bob agree on ψ, then each of them knows that they have agreed on ψ. This is a key property of agreement: in order for there to be agreement, every participant in the agreement must know that there is agreement. Thus, we expect $agree(\psi) \Rightarrow E(agree(\psi))$ to be valid. The Induction Rule for

common knowledge tells us that if this is the case, then $agree(\psi) \Rightarrow C(agree(\psi))$ is also valid. Hence, agreement implies common knowledge.

Now suppose that Alice and Bob are trying to coordinate their actions, i.e., they want to ensure that Alice performs action a precisely when Bob performs action b. Clearly, this involves the agents' agreeing on when to perform the actions; as our analysis shows, this requires common knowledge. Unlike agreement, which we treat as an intuitive notion, coordination can be defined formally. We establish a formal connection between coordination and common knowledge in the analysis later in this chapter of coordinated attack and simultaneous Byzantine agreement. (We explore the connection between coordination and common knowledge in more detail in Chapter 11.)

As we shall see, the connection between agreement and common knowledge provides us with a sharp tool with which to analyze agreement problems. We can use this connection to prove impossibility results, namely, to prove that there are no protocols for solving certain agreement problems, such as coordinated attack or agreeing to disagree. We can also use this connection in a positive manner, as a tool for the design of efficient protocols for reaching simultaneous Byzantine agreement.

6.1 Coordinated Attack

Communication plays an important role in facilitating coordination between agents. How, other than by means of communication, can an agent arrange to coordinate his actions with the actions of other agents in cases when the coordination was not fixed in advance? It is perhaps not surprising that *guaranteed* coordination may require some degree of reliability of the communication medium. Indeed, unreliable communication renders such coordination impossible. This is particularly well illustrated by the *coordinated attack* problem, a well-known problem from the distributed systems folklore. The problem can be described informally as follows:

> Two divisions of an army, each commanded by a general, are camped on two hilltops overlooking a valley. In the valley awaits the enemy. It is clear that if both divisions attack the enemy simultaneously they will win the battle, while if only one division attacks it will be defeated. As a result, neither general will attack unless he is absolutely sure that the other will attack with him. In particular, a general will not attack if he receives no messages. The commanding general of the first division wishes to coordinate a simultaneous attack (at some time the next day).

> The generals can communicate only by means of messengers. Normally, it takes a messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will it take them to coordinate an attack?

Suppose a messenger sent by General A reaches General B with a message saying "*attack at dawn.*" Should General B attack? Although the message was in fact delivered, General A has no way of knowing that it was delivered. A must therefore consider it possible that B did not receive the message (in which case B would definitely not attack). Hence A will not attack given his current state of knowledge. Knowing this, and not willing to risk attacking alone, B cannot attack based solely on receiving A's message. Of course, B can try to improve matters by sending the messenger back to A with an acknowledgment. Imagine that the messenger is again successful and delivers the acknowledgment. When A receives this acknowledgment, can he then attack? A here is in a similar position to the one B was in when he received the original message. This time B does not know that the acknowledgment was delivered. Since B knows that without receiving the acknowledgment A will not attack, B cannot attack as long as he considers it possible that A did not receive the acknowledgment. Hence, A cannot attack before he ensures that B knows the acknowledgment has been delivered. At this point, A might try to improve matters by sending the messenger back to B with an acknowledgment to the acknowledgment. Unfortunately, similar reasoning shows that this again will not suffice. As the reader can check, this time the problem is that A does not know that B knows that A knows that B received A's initial message. Indeed, it is possible to show (and will follow from our later results) that no number of successful deliveries of acknowledgments to acknowledgments can allow the generals to attack. Note that the problem is not caused by what actually happens, but by the uncertainty regarding what might have happened. In the scenario we have just considered, communication proceeds as smoothly as we could hope—all the acknowledgments sent are received—and still coordination is not attained.

If we let *delivered* represent the fact that at least one message was delivered, it is not too hard to show that when B gets A's initial message, $K_B(\mathit{delivered})$ holds. Moreover, when A gets B's acknowledgment, $K_A K_B(\mathit{delivered})$ holds, when B gets A's acknowledgment of the acknowledgment, $K_B K_A K_B(\mathit{delivered})$ holds, and so on (Exercise 6.1). However, even if all the acknowledgments sent are received, common knowledge of *delivered* never holds.

The fact that common knowledge of *delivered* does not hold even if all acknowledgments sent are received is not an accident. As we are about to show, in any system

with unreliable communication, there can never be any common knowledge about message delivery. Is this a problem for the generals? After all, they are interested in coordinating an attack, not in attaining common knowledge. Unfortunately for the generals, as we suggested earlier (and are about to prove), common knowledge is a prerequisite for coordination, in particular, the type of coordination required in the coordinated attack problem. Thus, coordinated attack is not possible in systems with unreliable communication.

Our first step in proving these results is to define a class of contexts in which it makes sense to talk about the agents' knowledge regarding message delivery. To make our results as general as possible, we want to assume as little as possible about these contexts. In particular, we do not want to assume anything about the internal actions agents can perform or the form of their local states. Also, we do not want to assume anything about the environment's states and actions, beyond assuming that message delivery events can take place, and that the environment records the events taking place in the system.

Formally, we call an interpreted context (γ, π) a *message-delivery context* if it satisfies the following assumptions:

- The environment and/or some of the agents have actions that we designate as message-delivery actions; intuitively, these actions result in messages being delivered to agents.

- γ is a recording context (as defined in Example 5.2.1), so that the environment's state includes the sequence of joint actions that have been performed so far, and τ updates states appropriately.

- The language includes the proposition *delivered*. As we said earlier, we intend *delivered* to be true if at least one message has been delivered, i.e., if at least one message-delivery action has been performed. Because the environment's state includes the sequence of joint actions performed, it is easy to define π to enforce this.

As discussed in Chapter 5, we can use a context to characterize a class of systems. A *message-delivery system* is a system of the form $\mathbf{I}^{rep}(P, \gamma, \pi)$, where (γ, π) is a message-delivery context and P is a protocol that can be run in context γ. In a message-delivery system, we can talk about message delivery and what the agents know about it.

What can we say about the agents' knowledge of *delivered* in a message-delivery system? The formula *delivered* is necessarily false at the beginning of a run (since

no messages have been delivered by time 0). It immediately follows that *delivered* cannot be common knowledge at time 0. Recall from Theorem 4.5.4 that in an asynchronous message-passing system common knowledge cannot be gained or lost. Thus, in an a.m.p. system the agents never attain common knowledge of *delivered*. In fact, as we now show, *delivered* can never become common knowledge even in synchronous systems, as long as message delivery is sufficiently unreliable.

What should it mean for message delivery to be "sufficiently unreliable"? Intuitively, we take this to mean that there may be unbounded message delivery, so that it can take arbitrarily long for a message to arrive. As a consequence, the only way an agent (other than the recipient) can find out about successful message delivery is through the receipt of other messages. In particular, if $\mathcal{R}$ has unbounded message delivery, i receives a message at a point (r, l) in $\mathcal{R}$, and no agent receives a message from i in run r between times l and m, then all the other agents will consider it possible at time m that i has not yet received the message (since they have no reason to believe otherwise).

We formalize the notion of unbounded message delivery as a richness condition on the set of runs. Let $\mathcal{R}$ be a system such that, for an appropriately chosen π, the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is a message-delivery system. Given a run $r \in \mathcal{R}$, we write $d(r, m) = k$ if exactly k messages have been delivered in the first m rounds of r. Clearly, we always have $d(r, 0) = 0$. We say that such a system $\mathcal{R}$ *displays umd* (umd stands for *unbounded message delivery*) if for all points (r, m) in $\mathcal{R}$ with $d(r, m) > 0$, there exists an agent i and a run $r' \in \mathcal{R}$ such that (1) for all agents $j \neq i$ and times $m' \leq m$ we have $r'_j(m') = r_j(m')$ and (2) $d(r', m) < d(r, m)$. Intuitively, we can think of i as the last agent to receive a message in r at or before round m, and r' as a run that is like r except that i does not receive this last message by round m. Clause (1) ensures that no other agent can tell by round m that i has not received this message. Because the last message to i in r is not delivered in r', we have $d(r', m) < d(r, m)$, as required by clause (2).

A number of systems of interest display umd. For example, it is easy to see that every a.m.p. system displays umd, as does every a.r.m.p. system (Exercise 6.2). In fact, we can make a stronger statement. We say that a *context* γ *displays umd* if all systems described by γ display umd, that is, if $\mathbf{R}^{rep}(P, \gamma)$ displays umd for every protocol P that can be run in context γ. It is easy to see that the context γ^{amp} characterizing a.m.p. systems introduced in Example 5.2.4 displays umd, as do the contexts that arise by replacing the condition *True* in γ^{amp} by *Rel* or *Fair*. Finally, the context implicitly characterized by the coordinated attack story also displays umd.

The umd condition is just what we need to show that common knowledge of message delivery is not attainable.

Theorem 6.1.1 *Let $\mathcal{I} = (\mathcal{R}, \pi)$ be a message-delivery system such that $\mathcal{R}$ displays umd, and let G be a set of two or more agents. Then*

$$\mathcal{I} \models \neg C_G(\mathit{delivered}).$$

Proof For every point (r, m) in $\mathcal{I}$, we prove that $(\mathcal{I}, r, m) \models \neg C_G(\mathit{delivered})$ by induction on $d(r, m)$. The case $d(r, m) = 0$ is trivial. Let $d(r, m) = k+1$ and assume that the claim is true for all points (r', m') with $d(r', m') \leq k$. Let r' and i be the run and agent, respectively, guaranteed to exist by the umd condition. Let $j \neq i$ be an agent in G. The umd condition guarantees that $d(r', m) < d(r, m)$. By the induction hypothesis, we have that $(\mathcal{I}, r', m) \models \neg C_G(\mathit{delivered})$. But since $r'_j(m) = r_j(m)$, we have that (r', m) is G-reachable from (r, m). Thus, by Exercise 2.7, it follows that $(\mathcal{I}, r, m) \models \neg C_G(\mathit{delivered})$, and we are done. ∎

Note that the form of Theorem 6.1.1 is somewhat weaker than that of Theorem 4.5.4. Unlike a.m.p. systems, it is not necessarily the case in a system satisfying umd that *no* common knowledge can be gained. For example, in a synchronous system satisfying umd, at 2 o'clock it is always common knowledge that the time is 2 o'clock. Rather, Theorem 6.1.1 essentially implies that *communication* in such systems cannot make formulas common knowledge. A formula that is common knowledge at some point must also be common knowledge at a point where no messages have been delivered. Of course, Theorem 6.1.1 is not strictly weaker than Theorem 4.5.4, because Theorem 6.1.1 applies in a much wider class of contexts than Theorem 4.5.4 does. As an example of an application of this theorem, we now use it to prove the impossibility of coordinated attack.

To be able to discuss a coordinated attack by the generals, we define a corresponding class of contexts. An interpreted context (γ, π) is *ca-compatible* if it is a message-delivery context in which two of the agents are the generals A and B, and for each $i \in \{A, B\}$, one of General i's actions is denoted attack_i. Moreover, we require that there be propositions $\mathit{attacked}_i$, for $i \in \{A, B\}$. We take $\mathit{attacked}_i$ to be true at a point if attack_i was performed at some point in the past, i.e., if attack_i is recorded in the environment's state. (Recall that γ is a recording context.) We take $\mathit{attacking}_i$ to be an abbreviation for $\neg \mathit{attacked}_i \wedge \bigcirc \mathit{attacked}_i$. Thus, $\mathit{attacking}_i$ is true if General i's next action is to attack. Finally, we take *attack* to be an abbreviation for $\mathit{attacking}_A \wedge \mathit{attacking}_B$, so *attack* is true if both generals are about to attack. Notice that the definition of ca-compatible contexts makes no assumptions whatsoever about the form of the generals' local states.

We can now formally capture the requirements of coordinated attack in terms of a specification. Let the specification σ^{ca} consist of all ca-compatible interpreted systems $\mathcal{I}$ such that

1. $\mathcal{I} \models attacking_A \Leftrightarrow attacking_B$,
2. $\mathcal{I} \models \neg delivered \Rightarrow \neg attack$, and
3. $(\mathcal{I}, r, m) \models attack$ for at least one point (r, m) of $\mathcal{I}$.

The first condition captures the key requirement for coordinated attack: It says that General A attacks at (r, m) iff General B attacks at (r, m). In particular, both generals must attack *simultaneously*. The second requirement captures the statement in the story that no attack is carried out if no messages are delivered. Finally, the third condition prevents the trivial solution to the problem where no general ever attacks. Notice that the first two conditions are run-based; the third, however, is not. We say that P is a *protocol for coordinated attack* in a ca-compatible interpreted context (γ, π) if P satisfies σ^{ca} in (γ, π).

We can now make precise our earlier claim that common knowledge is a prerequisite for coordinated attack. We start by showing that when the generals attack, it must be common knowledge that they are attacking. To simplify notation, we use C rather than $C_{\{A,B\}}$ to represent common knowledge among the generals. Similarly, we use E instead of $E_{\{A,B\}}$. We focus here on the case in which the protocols the generals follow are *deterministic*. (We briefly comment on the nondeterministic case at the end of the section.)

Proposition 6.1.2 *Let (γ, π) be a ca-compatible interpreted context and let P be a deterministic protocol. If $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca}, then*

$$\mathcal{I} \models attack \Rightarrow C(attack).$$

Proof We first show that $\mathcal{I} \models attack \Rightarrow E(attack)$. Suppose $(\mathcal{I}, r, m) \models attack$, so that $(\mathcal{I}, r, m) \models attacking_A \wedge attacking_B$. In particular, this means that $P_A(r_A(m)) = \mathsf{attack}_A$, so that A attacks in round $m + 1$ of r. Let (r', m') be a point of $\mathbf{R}^{rep}(P, \gamma)$ that A considers possible at (r, m), i.e., $r_A(m) = r'_A(m')$. Because P_A is deterministic, it follows that $P_A(r'_A(m')) = P_A(r_A(m)) = \mathsf{attack}_A$. Hence, $(\mathcal{I}, r', m') \models attacking_A$ as well. Since $\mathcal{I}$ satisfies σ^{ca}, we have that $(\mathcal{I}, r', m') \models attacking_B$. Thus, $(\mathcal{I}, r', m') \models attack$. This means that *attack* holds at all points that A considers possible at (r, m), so $(\mathcal{I}, r, m) \models K_A(attack)$. An analogous argument shows that $(\mathcal{I}, r, m) \models K_B(attack)$. Hence we get $(\mathcal{I}, r, m) \models E(attack)$. Because (r, m) was an arbitrary point, we get $\mathcal{I} \models attack \Rightarrow E(attack)$, as desired. By the Induction Rule, it now follows that $\mathcal{I} \models attack \Rightarrow C(attack)$. ∎

Proposition 6.1.2 draws a formal connection between an action (attacking in this case) and a state of knowledge (common knowledge of attack). We stress that the

generals need not be doing any reasoning for this result to hold; and even if they do reason, they need not be aware of the notion of common knowledge. Nevertheless, when they attack they *must* have common knowledge of the fact they are attacking, according to our external definition of knowledge.

Because the successful delivery of at least one message is a prerequisite of an attack, we obtain the following:

Corollary 6.1.3 *Let (γ, π) be a ca-compatible interpreted context and let P be a deterministic protocol. If $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca}, then*

$$\mathcal{I} \models attack \Rightarrow C(delivered).$$

Proof The second requirement of σ^{ca} is equivalent to $\mathcal{I} \models attack \Rightarrow delivered$. From Exercise 2.9, we have that $\mathcal{I} \models C(attack) \Rightarrow C(delivered)$. The result now follows from Proposition 6.1.2. ∎

Corollary 6.1.3 and Theorem 6.1.1 together imply that the generals in the coordinated attack problem are never able to attack. More generally, there is no deterministic protocol for coordinated attack in a system that displays umd.

Corollary 6.1.4 *If (γ, π) is a ca-compatible interpreted context such that γ displays umd, then there is no deterministic protocol P that satisfies σ^{ca} in (γ, π).*

Because in the coordinated attack example we are talking about a context that displays unbounded message delivery, Corollary 6.1.4 says that there is no deterministic protocol for the coordinated attack problem in such a context! It might not be too surprising that coordinated attack is not attainable in some runs of a protocol (in particular, runs where the messenger gets lost and does not deliver too many messages). Corollary 6.1.4, however, makes a far stronger claim: it says that an attack is never attainable in *any* run of *any* deterministic protocol for coordinated attack. Thus, even if every message is delivered, coordinated attack is not possible, as long as there is the possibility that messages will not be delivered.

The fact that coordinated attack implies common knowledge (Proposition 6.1.2) depends on our requirement that the coordinated attack must be simultaneous and the assumption that the generals are using deterministic protocols. In practice, simultaneity might be too strong a requirement. A protocol that guarantees that the generals attack within a short time of each other may be quite satisfactory. In a system where the generals attack within a short time of each other, attacking does not necessarily imply common knowledge of the attack. Nevertheless, in Chapter 11

we use similar arguments to show that even such weaker forms of coordination are unattainable if communication is unreliable.

While the assumption that the generals are following deterministic protocols is quite reasonable in practice, it is instructive to understand where we use it in the proof of Proposition 6.1.2. Essentially, the fact that P is deterministic makes $attacking_i$ depend on i's local state, for $i \in \{A, B\}$. As a result, we have that $\mathcal{I} \models attacking_A \Rightarrow K_A(attacking_A)$. To see how this may fail for nondeterministic protocols, consider the following protocol P: General A simply sends a message saying *attack*; after that he nondeterministically chooses in each round whether or not to attack. After receiving the message, General B nondeterministically chooses in each round whether or not to attack. Suppose we are given a ca-compatible context (γ, π), where $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$. If Ψ does not put any constraints on the set of acceptable runs, then it is clear that $\mathbf{I}^{rep}(P, \gamma, \pi)$ will not satisfy σ^{ca}; there will be many runs where one general attacks and the other does not. It is possible, however, to choose Ψ in such a way that $\mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca} and $\mathbf{R}^{rep}(P, \gamma)$ displays umd (see Exercise 6.3). The formula $attack \Rightarrow C(attack)$ is not valid in $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$. In fact, even $attacking_A \Rightarrow K_A(attacking_A)$ is not valid there; because of the nondeterminism, at a point where General A is about to attack, he does not know that he is about to attack. The reason that $\mathcal{I}$ still manages to satisfy σ^{ca} is that Ψ here "magically" rejects all runs where the generals do not coordinate. As we show in Chapter 7, by putting reasonable constraints on Ψ, we can ensure that this does not happen and that the analogue of Proposition 6.1.2 holds even for nondeterministic protocols (Exercise 7.9).

There is another way in which we could have proved impossibility of coordinated attack with respect to nondeterministic protocols. Notice that in the definition of a ca-compatible context we did not assume that a general records the fact that he has just attacked in his local state. By assuming that the generals do keep track of the fact they have attacked, we would make $attacked_i$ be a formula that depends on i's local state. This would make it possible to prove analogues of Proposition 6.1.2 and Corollary 6.1.3 for arbitrary protocols and contexts, using $attacked = (attacked_A \wedge attacked_B)$ instead of *attack*. As a result, we could again use Theorem 6.1.1 to obtain an analogue of Corollary 6.1.4. This would prove the impossibility of solving coordinated attack when umd holds in this type of ca-compatible contexts, even for nondeterministic protocols (Exercise 6.4).

6.2 Agreeing to Disagree

The analysis in the previous section demonstrated a formal connection between agreement and common knowledge. To coordinate their attack, the generals have to agree to attack together at a particular time, and our analysis shows that common knowledge is a necessary and sufficient condition for such an agreement to hold. This intimate connection between agreement and common knowledge has surprising consequences in applications in which the players are attempting to agree to take *different* actions (unlike the coordinated-attack situation in which they are attempting to agree to take essentially the same action).

An example of such an application is trading in the stock market, where a transaction occurs when one side *buys* and the other side *sells*. Why do people trade? Some trades are certainly due to the fact that people may have different utilities for having money at a given moment: one person may need to make a big payment and will therefore want to sell stock, while the other may have just received a large sum of money and may wish to invest some of it in the stock market. A great deal of trade, however, takes place for purely speculative reasons. The seller thinks the price of a given stock is likely to go down, while the buyer believes it will go up. Perhaps the buyer has some information leading him to believe that the company that issued the stock is going to do well in the next year, while the seller has information indicating that the company might fail.

For a trade to take place, the buyer and seller have to agree to the transaction, which means that they have to reach common knowledge that the trade takes place. But then part of the information that the buyer has is the fact that the seller is willing to sell and, similarly, part of the seller's information is that the buyer is willing to buy. How should this information affect their decisions? To take a somewhat extreme example, say the seller is Ms. X, a top executive in the company whose stock is being traded. "Clearly," the intended buyer should reason, "if Ms. X is selling, then the stock price is likely to drop. Thus, if she is willing to sell the stock for $\$k$, then I should not buy it for that amount." Since the participants in the trade have reached common knowledge when the trade takes place, they should make use of this knowledge when making their decisions. Somewhat surprisingly, as we show in this section, if they do use this knowledge, then the trade cannot take place! More precisely, we show that if both sides in the trade act according to the same rules, then the common knowledge that would arise should the trade take place prevents the trade from taking place. Roughly speaking, the result says that players cannot "agree to disagree", that is, they cannot have common knowledge that they are taking different actions, such as buying and selling. (Notice that the word "agree" plays two different

roles in the phrase "agree to disagree"; "agree" refers to common knowledge, while "disagree" refers to reaching different decisions.)

To prove this result, we need a few definitions. As described in Chapter 5, the actions taken by the players are prescribed by their protocols, where a protocol for player i is a function of player i's local state. In many applications, it is more appropriate to view the player's actions as depending not on her local state, but on the set of points she considers possible. For example, suppose a player wants to maximize some payoff that depends on the point. Since the player does not know what the actual point is, her decision actually depends on the set of points that she considers possible. In two different systems, she may have the same local state, but consider a different set of points possible, and thus take different actions. This is a phenomenon we examine more carefully in Chapter 7, when we consider knowledge-based programs. For now, we generalize our notion of protocol in an attempt to capture this intuition.

Given a local state ℓ for player i in a system $\mathcal{R}$, let $\mathbf{IS}_i(\ell, \mathcal{R})$ denote the set of points (r, m) in $\mathcal{R}$ such that $r_i(m) = \ell$. If $\mathcal{I} = (\mathcal{R}, \pi)$ is an interpreted system, we identify $\mathbf{IS}_i(\ell, \mathcal{I})$ with $\mathbf{IS}_i(\ell, \mathcal{R})$. In the terminology of Section 2.5, $\mathbf{IS}_i(\ell, \mathcal{R})$ is the *information set* of player i when in local state ℓ in the interpreted system $\mathcal{R}$. As we said earlier, we want to view player i's action as a function of her information set rather than as a function of her local state. Essentially, this amounts to making a player's action a function of her *knowledge*.

To make this precise, given a set $\mathcal{G}$ of global states, let S be the set of points over $\mathcal{G}$ (i.e., the set of points (r, m) where r is a run over $\mathcal{G}$). If ACT_i is the set of actions for player i, we define a *decision function* for player i (over $\mathcal{G}$) to be a function whose domain consists of some subsets of S and whose range is ACT_i. Thus, a decision function prescribes an action for the subsets of S in its domain.

We have another method for prescribing actions for player i: namely, by means of a protocol. How are decision functions and protocols related?

In a precise sense, we can view decision functions as more general than protocols. To know what action a decision function prescribes for player i in a given local state, we need to know what player i's information set is. But this depends not just on player i's local state, but on the whole system. We may be able to associate a protocol with a given decision function once we have a system in hand to determine what the information sets are. Given a decision function D for player i and a system $\mathcal{R}$, we say that a protocol P_i for player i is *compatible with* D *in* $\mathcal{R}$ if $P_i(\ell) = D(\mathbf{IS}_i(\ell, \mathcal{R}))$ for all $\ell \in L_i$. (Note that this requires that the domain of D includes all the information sets of player i in $\mathcal{R}$.) It is not hard to see that every deterministic protocol is compatible

with some decision function D; i.e., if P_i is a deterministic protocol, then there is a decision function D such that P_i is compatible with D in all systems $\mathcal{R}$ (Exercise 6.5).

As our discussion suggests, we are mainly interested in applying decision functions to information sets. We have, however, allowed decision functions to be defined on arbitrary sets of points. There are many sets of points that cannot be information sets (in particular, any set that includes points (r, m) and (r', m') such that $r_i(m) \neq r'_i(m')$). Why are we allowing decision functions to be defined on such sets? For one thing, this makes it possible for us to talk about all players using the *same* decision function, as we do in the examples we provide later in this section. In addition, as these examples show, it is often the case that the decision function we have in mind is most naturally thought of as a function on arbitrary sets of points.

As we said, we are interested in situations where all players use the same decision function. Thus, we assume for the remainder of this section that the players' actions are all taken from the same set ACT. We say that the joint protocol $P = (P_1, \ldots, P_n)$ *implements* the decision function D in context γ if P_i is compatible with D in $\mathbf{R}^{rep}(P, \gamma)$, for $i = 1, \ldots, n$. Thus, if P implements D, then the actions prescribed by both P and D agree in the system representing P.

We are now almost ready to state and prove the *Agreement Theorem*, a celebrated result in game theory. What we want to show is that if two players use the same decision function, then they cannot agree to perform different actions. To capture this formally, we restrict attention to *interpreted contexts* (γ, π^{ag}) *for agreement.* Again, we want to put as few restrictions on such contexts as possible: just enough so that we can talk about the actions performed by the players. We assume that

- the players' actions are all taken from the same set ACT,
- γ is a recording context,
- for each action a, there is a primitive proposition $perf_i(\mathsf{a})$, and
- $\pi^{ag}(s)(perf_i(\mathsf{a}))$ is true in a state s if the action a was performed by player i, as recorded in the environment's state. As we did for coordinated attack, we take $act_i(\mathsf{a})$ to be an abbreviation for $\neg perf_i(\mathsf{a}) \wedge \bigcirc perf_i(\mathsf{a})$. Thus, $act_i(\mathsf{a})$ is true if player i is about to perform action a.

We need one more technical definition before we can state the Agreement Theorem. A decision function D is said to be *union consistent* if it satisfies the following condition: for every action a and for every collection $T_1, T_2, \ldots$ of pairwise disjoint subsets of S, if $D(T_j) = \mathsf{a}$ for all j, then $\bigcup_j T_j$ is in the domain of D and

$D(\bigcup_j T_j) = \mathsf{a}$. Intuitively, the function D is union consistent if, whenever it prescribes the same action a for disjoint sets of points, it prescribes a for the union of these sets as well. Union consistency seems fairly reasonable: intuitively, it says that if a player performs the action a whenever she considers T_j to be the set of possible worlds, then she should also perform a if she considers all the points in $\cup_j T_j$ possible. Recall that we observed that any deterministic protocol can be obtained from some decision function. In fact, it can be shown that any deterministic protocol can be obtained from some *union-consistent* decision function (Exercise 6.5). We give some examples of union-consistent decision functions after proving the theorem.

We can now formally state the Agreement Theorem. As we did in our discussion of coordinated attack, we use C rather than $C_{\{1,2\}}$ to represent common knowledge among the two players.

Theorem 6.2.1 *Suppose* $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi^{ag})$, *where* P *is a joint protocol and* (γ, π^{ag}) *is an interpreted context for agreement. If* P *implements some union-consistent decision function in context* γ, *then for all actions* $\mathsf{a}, \mathsf{b} \in ACT$, *if* $\mathcal{I} \models C(act_1(\mathsf{a}) \wedge act_2(\mathsf{b}))$, *then* $\mathsf{a} = \mathsf{b}$.

Proof Suppose P implements the union-consistent decision function D in context γ. Suppose $(\mathcal{I}, r, m) \models C(act_1(\mathsf{a}) \wedge act_2(\mathsf{b}))$; we want to show that $\mathsf{a} = \mathsf{b}$. Let S' consist of all points that are $\{1, 2\}$-reachable from (r, m). Suppose $(r', m') \in S'$ and $r'_1(m') = \ell$. By the definition of reachability, if $(r'', m'') \sim_1 (r', m')$, then $(r'', m'') \in S'$. Thus, $\mathbf{IS}_1(\ell, \mathcal{I}) \subseteq S'$. Moreover, by the definition of $\mathbf{IS}_1(\ell, \mathcal{I})$, if $\ell \neq \ell'$ for some $\ell' \in L_1$, then $\mathbf{IS}_1(\ell, \mathcal{I}) \cap \mathbf{IS}_1(\ell', \mathcal{I}) = \emptyset$. It follows that S' is a disjoint union of sets of the form $\mathbf{IS}_1(\ell, \mathcal{I})$. Since $(\mathcal{I}, r, m) \models C(act_1(\mathsf{a}))$, we must have that $(\mathcal{I}, r', m') \models act_1(\mathsf{a})$, so that $P_1(\ell) = \mathsf{a}$. The fact that P implements D implies that $D(\mathbf{IS}_1(\ell, \mathcal{I})) = \mathsf{a}$. By the union consistency of D, we can now conclude that $D(S') = \mathsf{a}$. A completely analogous argument with respect to player 2 and action b yields that $D(S') = \mathsf{b}$. Thus, $\mathsf{a} = D(S') = \mathsf{b}$ as desired. ∎

Thus, if two agents use the same union-consistent decision function, i.e., they act according to the same rules, then they cannot have common knowledge that they are taking different actions. That is, they cannot agree to disagree.

We observed earlier that every protocol for player i is compatible with *some* union-consistent decision function. The crux of the Agreement Theorem is the requirement that both players use the *same* union-consistent decision function. How reasonable is this requirement? We now describe three examples of situations where this arises.

For the first example, suppose two players each perform an action and receive a payoff as a result of that action. Moreover, suppose that the payoff to player i

depends solely on the action that player i performs and the global state at which the action is performed. This means that, in particular, the payoff is independent of the action that the other player performs and both players receive the same payoff if they perform the same action. For example, if the two players are betting on numbers in a roulette game, and we assume that the winning number is completely determined by the global state, then each player's payoff depends only on the global state (and not on the bet made by the other player), and the players receive the same payoff if they make the same bet.

Of course, the problem is that, in such scenarios, the players do not know what the actual global state is, so that they do not know what their payoff will be. Suppose that the players are both *risk averse*, which means that they choose the action for which the worst-case payoff is maximal. Formally, if $payoff(s, \mathsf{a})$ is the payoff for performing action a at the global state s and S' is a set of points, define $payoff(S', \mathsf{a}) = \min_{(r,m)\in S'}(payoff(r(m), \mathsf{a}))$. (To be precise, we should use the infimum rather than the minimum in this expression, since a minimum over an infinite set of payoffs may not exist.) Intuitively, $payoff(S', \mathsf{a})$ is the worst-case payoff if the action a is performed in the set S'. For simplicity, we assume that ACT is finite and that for all subsets S' of points and for all distinct pairs of actions a and b, we have $payoff(S', \mathsf{a}) \neq payoff(S', \mathsf{b})$. Under these assumptions, we define $D^{ra}(S')$ to be the unique action a such that $payoff(S', \mathsf{a}) > payoff(S', \mathsf{b})$ for all $\mathsf{b} \neq \mathsf{a}$. (The *ra* in D^{ra} stands for *risk averse*.) Thus, according to the decision function D^{ra}, the action that is chosen in S' is the one that maximizes the worst-case payoff in S'. It is easy to check that D^{ra} is a union-consistent decision function (Exercise 6.6). It follows that if the players are both risk averse in this sense, they cannot agree to disagree. Thus, if they discuss their actions until they have common knowledge of the actions they are about to perform, then these actions must be the same.

In our remaining two examples, the players' decisions are defined in terms of probability. A thorough discussion of probability in our framework is beyond the scope of this book. We provide a brief, somewhat informal, discussion of these examples here; further references can be found in the notes. (These examples can be skipped by a reader unfamiliar with probability theory.)

Suppose we have a probability distribution Pr defined on certain subsets of S, the set of points. If e is a fixed subset of points (in the terminology of Section 2.5, e is an event) and ACT, the set of actions, just consists of all numbers in the interval $[0, 1]$, let the decision function D^{pr} be defined on the subsets S' of S for which $\Pr(e|S')$ is defined. On these subsets, we define $D^{pr}(S') = \Pr(e|S')$. Thus, $D^{pr}(S')$ is the conditional probability of e given S'. It is easy to show that D^{pr} is union consistent (Exercise 6.7). If player i is in local state ℓ, then his estimate of the probability

of e is given by the conditional probability $\Pr(e|\mathbf{IS}_i(\ell, \mathcal{R}))$. Thus, according to the Agreement Theorem, if the players have common knowledge of their estimates of the probability of e, then these estimates must be the same.

To bring out the perhaps surprising nature of this example, suppose the players start with the same probability distribution on the set of points and then receive some information that causes them to revise their estimate of the probability of e, using conditioning. They can then exchange their estimates of the probability of e. Doing so can cause them to revise further their estimates of e, since their information changes. Suppose that after a while they reach steady state, and no further exchanges of their estimates can cause them to revise these estimates. It is possible to show that in a large class of contexts, the players are in fact guaranteed to reach steady state and to attain common knowledge of the fact that they are in steady state (see Exercise 6.8 for an example). Once this happens, their estimates are common knowledge, so according to the Agreement Theorem, they must be the same. Thus, although the players might have different information, they cannot agree to disagree on their estimates of the probability of e.

For our final example, we return to the setting of the first example but, as in the second example, we assume that we have a probability distribution on the set of points. Rather than being risk averse, as in our first example, suppose that the players perform the action that has the highest expected rate of return. That is, we now define $payoff'(S', \mathsf{a})$ to be the expected payoff of the action a over the set S'. Again, we assume for simplicity that ACT is finite and for all distinct actions a and b, we have $payoff'(S', \mathsf{a}) \neq payoff'(S', \mathsf{b})$. Under these assumptions, we define $D^{um}(S')$ to be the unique action a such that $payoff(S', \mathsf{a}) > payoff(S', \mathsf{b})$ for all $\mathsf{b} \neq \mathsf{a}$. (The *um* in D^{um} stands for *utility maximizer*.) It is easy to check that D^{um} is union consistent (Exercise 6.7). Again, the Agreement Theorem tells us that if the players' protocols are consistent with D^{um}, then they cannot agree to disagree.

If we take the actions in the first and third examples to be *buy* and *sell*, then we are back to the scenario with which we began this section. Thus, in this setting, the Agreement Theorem tells us that speculative trading between players who follow the same rules (e.g, have the same payoff function and are both risk averse in the first example, or have the same payoff function and probability distribution and are both utility maximizers in the third example) is impossible. This certainly seems counterintuitive. There has been a lot of work in game theory on trying to understand the implications of this result and to avoid the apparent paradox. Some of the approaches involve limited reasoners, a topic we discuss in Chapters 9 and 10.

6.3 Simultaneous Byzantine Agreement

In Sections 6.1 and 6.2, common knowledge served as a tool for proving impossibility results, namely, the fact that there are no protocols solving the coordinated-attack problem or for agreeing to disagree. We now present a case in which common knowledge is used in a positive manner, as a tool for the design of efficient protocols.

The coordinated attack problem deals with the impact that unreliable communication has on coordination in multi-agent protocols. Another major source of difficulty in distributed systems is the fact that processes may fail during the execution of a protocol. This can cause particular difficulties when it comes to coordinating actions between different sites in such a system. We do not want two sites in an airline reservation system to sell the same seat to two different people; a bank must ensure that every transaction made at one of its automated tellers is appropriately recorded in its central database. Because the components of such a system do fail occasionally for various reasons, it is important to program them in such a way that the overall behavior of the system will not be jeopardized by the failure of a small number of its components.

The paradigmatic problem concerning reaching agreement at different sites in a system in the presence of failures is the *Byzantine agreement problem*. The Byzantine agreement problem can be informally described as follows: We have n generals, up to t of which might be traitors. Each general initially prefers to either attack or retreat (although they are willing to do something other than what they initially prefer). At the end of the protocol, they must reach agreement, so that the loyal generals either all attack or all retreat. (The traitors can do whatever they like; we have no control over them.) Although the generals can talk to each other (over reliable channels), there is no broadcast facility. It is not possible for a general to take a loudspeaker and announce his vote to all the others. There is a trivial protocol satisfying these conditions: one where the generals retreat, no matter what their initial preference. In practice, however, this is not a satisfactory solution. A natural additional property to demand is that if all generals initially prefer to do the same thing (either attack or retreat), this is what they will agree to do. This condition eliminates trivial solutions such as retreating no matter what.

If we are guaranteed that there are no traitors in the system, then reaching agreement is a trivial matter. The generals all talk to each other, find out what each of them initially prefers to do, and use some uniform rule for reaching a decision (for example, the rule might be to attack if any general initially prefers to attack). In the presence of traitors, however, assuming we have no way of knowing beforehand who the traitors are, the situation becomes considerably more complicated. What do we

do if a traitor tells one loyal general that he wants to attack and tells another that he wants to retreat?

This problem turns out to be remarkably subtle and has been studied at length in the literature. We focus here on one particular variant of the problem, where we require that not only do the generals decide, but that they decide simultaneously. We refer to this variant as the *simultaneous Byzantine agreement (SBA)* problem. Just as in the case of coordinated attack, we can show that in a large class of interpreted contexts, the requirement of simultaneity here leads to common knowledge.

We now define this class of interpreted contexts, which we call *ba-compatible*. These ba-compatible contexts share many of the features of ca-compatible contexts. Again, we want to make minimal assumptions about the processes' (i.e., generals') local states and have the environment's state record the actions performed. The main difference is that we now want to allow the environment to specify which processes are faulty and how they are faulty. This means that there are more possible actions that the environment can perform. For now, we do not go into the details of what these actions are; we defer that discussion to Section 6.5. Formally, we say that an interpreted context (γ, π) is *ba-compatible* if it satisfies the following assumptions:

- For each process $i = 1, \ldots, n$, one of i's actions is denoted $\mathsf{decide}_i(y)$, for $y \in 0, 1$. Intuitively, this means that process i has decided on the value y. (We can think of $\mathsf{decide}_i(0)$ as representing a decision to retreat, while $\mathsf{decide}_i(1)$ represents a decision to attack.)

- The environment's actions include ones of the form $(\mathsf{a}_{e1}, \ldots, \mathsf{a}_{en})$. The components a_{ei} themselves are tuples, which describe which messages sent by process j are delivered to process i in that round, whether or not i fails in that round, which we capture by a fail_i action, and its faulty behavior if it does fail. We discuss later how this faulty behavior is described, since this depends on the type of faulty behavior we allow. We say that process i *fails in round k of run r* if the environment's action at this round has a fail_i component. We say that process i is *faulty* in round k of run r (or at the point (r, k)) if process i failed in some round $k' \leq k$ of run r. Otherwise, we say that i is *nonfaulty* or *correct*.

- γ is a recording context. Note that this means that we can tell from the environment's state which processes are faulty, by seeing which fail_i actions have been performed.

- Process i's initial state is a tuple of the form $(x_i, \ldots)$, where x_i is either 0 or 1. Intuitively, x_i represents process i's initial preference to either retreat (0) or

attack (1). We also assume that process i's local state records whether process i has performed an action of the form $\mathsf{decide}_i(y)$. All the results of this chapter hold even without this assumption, but it turns out to be convenient for our results in Section 7.4. We make no further assumptions about the form of the processes' local states.

- The environment's state includes the tuple $(x_1, \ldots, x_n)$ of the processes' initial preferences.

- The language includes the propositions $decided_i(y)$, $decided_{\mathcal{N}}(y)$, and $\exists y$, for $i = 1, \ldots, n$ and $y = 0, 1$. We define π so that (1) $decided_i(y)$ is true at a global state if the action $\mathsf{decide}_i(y)$ was performed at any previous round (note that since this fact is recorded in process i's local state, $decided_i(y)$ is a proposition local to process i), (2) $decided_{\mathcal{N}}(y)$ is true at a global state if each nonfaulty process i has performed the action $\mathsf{decide}_i(y)$ (recall that the nonfaulty processes can be determined from the environment's local state), and (3) $\exists y$ is true if $x_i = y$ for some process i, so that some process initially preferred to decide on the value y. Similarly to what we did in the case of coordinated attack, we take $deciding_{\mathcal{N}}(y)$ to be an abbreviation for $\neg decided_{\mathcal{N}}(y) \wedge \bigcirc decided_{\mathcal{N}}(y)$.

A *ba-compatible interpreted system* is one of the form $\mathbf{I}^{rep}(P, \gamma, \pi)$, where P is a protocol and (γ, π) is a ba-compatible interpreted context. In a ba-compatible interpreted system, we can talk about the processes' initial preferences and their decisions, so it makes sense to talk about SBA.

As we said earlier, we intend to focus here on *simultaneous* Byzantine agreement (SBA). The specification σ^{sba} of SBA is run-based. It is satisfied by all ba-compatible interpreted systems $\mathcal{I}$ such that each run r in $\mathcal{I}$ satisfies

- *Decision:* every process i that is nonfaulty in r performs exactly one decide_i action in r;

- *Agreement:* the nonfaulty processes all decide on the same value;

- *Validity:* if all the processes have the same initial preference x, then all nonfaulty processes decide on the value x; and

- *Simultaneity:* the nonfaulty processes all decide simultaneously, i.e., in the same round.

The first clause ensures that the nonfaulty processes decide exactly once, the second ensures that their decisions are in agreement, the third ensures the decision is

related to the initial preferences in a nontrivial way, and the fourth guarantees that the decision is simultaneous. Notice that the third clause prevents trivial solutions such as one in which everyone simply always decides on the value 0 in the first round and halts. Indeed, the third clause ensures that for any given $y \in \{0, 1\}$ the processes may decide on the value y only if at least one process had y as its initial preference. We say that P is a *protocol for SBA* or that P *attains SBA*, in a ba-compatible interpreted context (γ, π) if P satisfies σ^{sba} in (γ, π). If r is a run in $\mathbf{I}^{rep}(P, \gamma, \pi)$, we say that P attains SBA *in k rounds in r* if $(\mathcal{I}, r, k) \models \mathit{deciding}_{\mathcal{N}}$. (Note that this means that a nonfaulty process i actually performs the action decide_i in round $k + 1$.) P attains SBA in k rounds if P attains SBA in k rounds in all runs of $\mathbf{I}^{rep}(P, \gamma, \pi)$.

It should be clear from the specifications and the descriptions of the problems that SBA and coordinated attack are closely related. Indeed, we can reformulate coordinated attack slightly to resemble SBA even more as follows. We can assume that each general i has an initial preference x_i regarding whether or not he would like to attack. We could then restate the coordinated attack problem by requiring that if both generals initially prefer to attack, then they should attack, while if both initially prefer to retreat, they should retreat. While this version of the coordinated attack problem is slightly different from the one we considered, we can easily prove results analogous to Theorems 6.1.1 and Corollary 6.1.4 for it. In fact, it is easy to show that if P is a deterministic protocol for this modified version of coordinated attack in an interpreted context (γ, π) allowing all four configurations of initial preferences, then $\mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca} (Exercise 6.9).

Despite these similarities, there are some significant differences between SBA and coordinated attack, at least in the contexts of most interest to us. In coordinated attack, both generals are assumed to be reliable; the problem is with the communication links. In SBA, we are mainly interested in contexts where correct generals have no problem communicating. Thus, we focus on contexts where communication is *reliable* and *immediate*, so that a message is guaranteed to arrive in the same round in which it is sent, provided that neither the sender nor the intended recipient of the message is faulty. The problem in these contexts is not with communication, but with the faulty processes.

The Byzantine agreement problem is sensitive to the type of faulty behavior a faulty process can display. The literature has concentrated on three basic failure modes:

1. *Crash failures:* a faulty process behaves according to the protocol, except that it might crash at some point, after which it sends no messages. In the round in which a process fails, the process may perform an arbitrary subset of the

actions it is supposed to perform, according to the protocol it is following. In particular, it may send only a subset of the messages that it is supposed to send according to its protocol.

2. *Omission failures:* a faulty process behaves according to the protocol, except that it may omit to send or receive an arbitrary set of messages in any given round. We sometimes refer to this case as the *general-omission* failure mode. Another variant of this mode in which faulty processes omit only to send messages is called the *sending-omission* failure mode.

3. *Byzantine failures:* faulty processes may deviate from the protocol in an arbitrary fashion; they may "lie," send deceiving messages, and collude to fool the nonfaulty processes in the most malicious ways.

In practice, crash failures occur quite regularly, as a result of mechanical and electric failures. Omission failures are often the result of communications problems. Finally, Byzantine failures represent the worst possible failures, where we can make no assumption on the behavior of faulty processes. Crash failures can be viewed as a restricted type of omission failures (a process omits to send all messages from a certain point on), and omission failures in turn can be viewed as a restricted type of Byzantine failures.

We model these failure modes in terms of the environment's actions. We defer the technical details to Section 6.5.

As we said earlier, the problem of SBA has been well studied in the literature. It is known that there are protocols that attain SBA in $t + 1$ rounds in all these failure modes, provided that communication is reliable and immediate. (In the case of Byzantine failures, we also have some constraints on the relationship between n, the total number of processes in the system, and t, the upper bound on the number of faulty processes. There is a protocol for SBA in this case if and only if $n > 3t$.) Moreover, there is no protocol that attains SBA in fewer than $t + 1$ rounds. In fact, it is known that any protocol for SBA in one of these failure modes requires at least $t + 1$ rounds to attain SBA in runs where there are no failures at all.

It might seem surprising that, even if we consider such relatively benign failures as crash failures, we still have to take $t + 1$ rounds to reach agreement in runs where there are no faulty processes. As the following example shows, the problem here, as in the case of coordinated attack, is not what does happen, but what *might* have happened.

Example 6.3.1 Suppose that $n = 3$ and $t = 1$, and we restrict to crash failures. Consider a run where all the processes have initial preference 0 and there are no

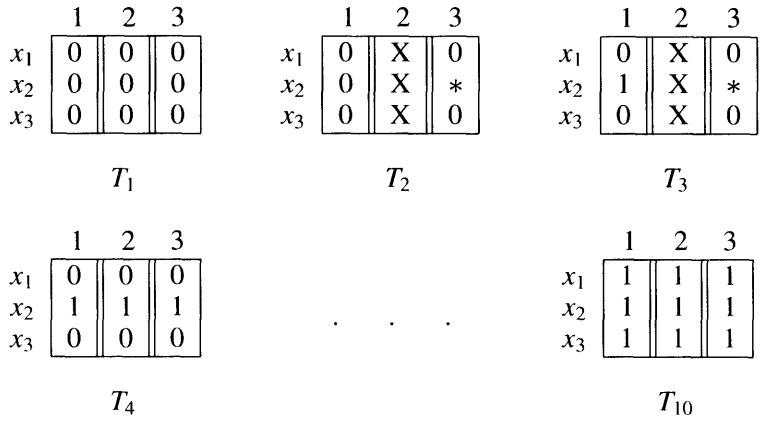

Figure 6.1 An example for $n = 3$ and $t = 1$

failures. By the validity requirement, this means that all the processes must decide on the value 0. Suppose that in round 1 every process sends its initial preference to every other process. Thus, at time 1, all the processes know that every process initially preferred 0. Since we are considering only crash failures, all the processes are telling the truth here. This means that all the processes know at time 1 that they must all ultimately decide on the value 0. Why are they not able to decide on the value 0 right away?

We can represent the situation at time 1 in a run using a 3×3 table, where each column represents the information that one process has at time 1 about the initial preferences x_1, x_2, and x_3 (see Figure 6.1). A failed process is assumed to have no information, and we mark this by the letter X. Otherwise, an entry in the table can be either 0, 1, or $*$, where we use a $*$ in row i of column j to represent the fact that j did not receive information about i's initial preference x_i (because process i crashed before sending j a message). In the run we just described, the situation is represented by a table all of whose entries are 0; this is table T_1 in Figure 6.1.

Recall that the nonfaulty processes must all decide simultaneously. As we suggested earlier, the problem is not with what actually happens, but with the uncertainty about what might have happened. In particular, although in the situation depicted by T_1 process 1 received a 0 from all processes, it is possible, as far as process 1 is concerned, that process 3 did not receive a message at all from process 2; this could happen if 2 crashed after sending a message to 1, but before sending a message to 3; this situation is depicted by table T_2. (Note that in T_1 process 1 does not consider it possible that process 2 told 3 that its initial preference is 1. We are not allowing lying

here; this would require Byzantine failures.) Clearly, process 1 cannot distinguish T_1 from T_2; it has the same local state in both situations (essentially described by the first column of the table). Now in the situation described by T_2, process 3, which did not get a message from process 2, considers it possible that 2's initial preference x_2 is 1, and that 2 passed x_2 on to process 1. Thus, in the situation described by T_2, process 3 considers the situation described by table T_3 possible. Finally, in T_3 process 1 does not know of any failure, and considers T_4 a possibility. Notice that, intuitively, our tour from T_1 to T_4 involved "silencing" process 2, changing its initial preference x_2, and "reviving" the process. By applying this type of reasoning it is possible to do the same to process 3 and then to process 1. As a result, we can construct a sequence of tables $T_1, \ldots, T_{10}$ such that T_1 is the table all of whose entries are 0 and T_{10} is a table all of whose entries are 1, and for each consecutive pair of tables T_l, T_{l+1}, with $l < 10$, there is some process that cannot distinguish the situations described by T_l and T_{l+1} and is correct in both of these situations (Exercise 6.10).

Now suppose that some correct process decides at time 1 on the value 0 in the situation described by table T_1. By the agreement and simultaneity requirements of SBA, all the processes must decide at time 1 on the value 0 in this situation. Since process 1 cannot distinguish T_1 from T_2, process 1 must decide at time 1 on the value 0 in the situation described by T_2. Again, by the agreement and simultaneity requirements, the two processes that are correct in this case (1 and 3) must decide at time 1 on the value 0. Similarly, we get that processes 1 and 3 decide at time 1 on the value 0 in the situation described by T_3. Continuing this argument, we get that all processes decide at time 1 on the value 0 in the situation described by T_{10}. But in the situation described by T_{10}, all the processes are correct and have 1 as their initial preference. The fact that they decide on the value 0 in T_{10} contradicts the validity requirement!

As should be clear from our discussion, simultaneity plays a crucial role here. Our proof would not hold (nor would the claim) had we not required simultaneity. To see why, suppose again that process 1 decides on the value 0 in the situation at time 1 described by table T_1. Since process 1 cannot distinguish the situation described by T_1 from that described by T_2, we know that process 1 also decides on the value 0 in the latter situation. Without the requirement of simultaneity, we cannot, however, conclude that process 3 decides on the value 0 in the situation described by T_2, although we can conclude that process 3 will eventually decide on the value 0 in this situation. (This may, however, require further messages from process 1.) More significantly, we cannot conclude that any process decides 0 in the situation described by T_3. ∎

We have stated (and shall prove formally later in this chapter) that $t + 1$ rounds are required to attain SBA even if there are no failures. It might seem that if we need $t + 1$ rounds if there are no failures, things could only be worse if there are failures. As we show, this is not the case. In fact, by doing a knowledge-based analysis of SBA, we can completely characterize the number of rounds that are required to reach agreement.

6.4 Nonrigid Sets and Common Knowledge

The validity requirement of SBA implies that if all initial preferences are 1, then the nonfaulty processes should decide on the value 1. In particular, for a process to decide on the value 0, the process must know that not all initial preferences are 1. Since the only possible initial preferences are 0 and 1, this says that for a process to decide on the value 0, the process must know that some initial preference is 0. Of course, knowledge that some initial preference is v is not sufficient for a process to decide on the value v. Otherwise, a process could simply always decide on its initial preference, creating a violation of the agreement property in cases where two processes had different initial preferences. In fact, Example 6.3.1 shows that even if a process knows that all of the initial preferences are 0, this is not sufficient to decide on the value 0. What other knowledge do the processes need?

Just as SBA requires simultaneity (namely, simultaneous agreement), so too the coordinated attack problem requires simultaneity (namely, simultaneous attacking). Proposition 6.1.2 tells us that if the generals attack in the coordinated attack problem, then the fact that they are both attacking must be common knowledge. It is natural to expect SBA to similarly require attaining common knowledge. The question is, which group of processes actually attains common knowledge? It is not the set of *all* processes, since we do not place any requirements on the actions of the faulty processes. The SBA problem specification σ^{sba} requires only that the *nonfaulty* processes decide on an appropriate value. SBA therefore involves coordinating the actions of the nonfaulty processes. Thus, we expect that the nonfaulty processes will need to attain common knowledge. Notice, however, that the set of nonfaulty processes is not fixed, but varies from one point of the system to another. A set whose identity is not fixed and can depend on the point is called a *nonrigid* set. Formally, given a system $\mathcal{R}$, a nonrigid set $\mathcal{S}$ of processes in the system is a function associating with every point of the system a subset of the processes. In other words, $\mathcal{S}(r, m)$ is a (possibly different) set of processes for every point (r, m) of $\mathcal{R}$. We use the formula $i \in \mathcal{S}$ to denote that i is in the nonrigid set $\mathcal{S}$. We take $i \in \mathcal{S}$ to be true at a point (r, m) if $i \in \mathcal{S}(r, m)$.

Nonrigid sets arise naturally in the analysis of a variety of problems. For example, when we consider a system in which processes can join and leave the system dynamically, the set of the processes in the system is a nonrigid set. Similarly, the set of processes that have direct communication links to a given process in such a system is a nonrigid set. The nonrigid set of most interest to us here is the set of nonfaulty processes, which we denote by $\mathcal{N}$. Thus, $\mathcal{N}(r, m)$ consists of the set of processes that are not faulty at the point (r, m).

Before we can relate SBA to common knowledge, we need to extend the definition of common knowledge to nonrigid sets. How should we do this? Given a nonrigid set $\mathcal{S}$, a natural candidate would be to define $E_{\mathcal{S}}\varphi$ as $\bigwedge_{i \in \mathcal{S}} K_i \varphi$. According to this definition, $E_{\mathcal{S}}\varphi$ would hold at a point (r, m) if all the processes in $\mathcal{S}(r, m)$ know φ at (r, m). $C_{\mathcal{S}}$ would then be defined in terms of $E_{\mathcal{S}}$ as usual. Note that in a formula such as $E_{\mathcal{S}}E_{\mathcal{S}}\varphi$, or even $K_i E_{\mathcal{S}}\varphi$, the value that the nonrigid set $\mathcal{S}$ takes on may vary. For example, in evaluating the truth of $K_i E_{\mathcal{S}}\varphi$, the value of $\mathcal{S}$ may be different at different points that i considers possible.

How can we judge whether the proposed definitions of $E_{\mathcal{S}}\varphi$ and $C_{\mathcal{S}}\varphi$ are appropriate? One criterion that is important for our applications is whether we can prove, as we did in the case of coordinated attack, that if the members of $\mathcal{S}$ have coordinated their actions, then this fact is guaranteed to be common knowledge among the members of $\mathcal{S}$. Does the definition of $C_{\mathcal{S}}$ we have just given have this property? For example, in SBA, is it necessarily the case that when a nonfaulty process decides, then it is common knowledge among the nonfaulty processes that the nonfaulty processes are deciding? We would expect this to be the case, by analogy with the situation for coordinated attack (Proposition 6.1.2). As we shall show later, if a nonfaulty process is guaranteed to know that it is nonfaulty, then there is indeed such common knowledge among the nonfaulty processes. In general, however, a process does not know whether it is faulty (at least, not in the round that it fails). The consequences of this lack of knowledge can perhaps most easily be seen in the case of general-omission failures. In this case, it is quite easy to construct a run in which a nonfaulty process decides on the value 0 and yet considers it possible that the nonfaulty processes are deciding on the value 1, exactly because it does not know whether it or the other processes are faulty (Exercise 6.17). We now define a notion of common knowledge that is appropriate even when the nonfaulty processes do not necessarily know that they are nonfaulty.

Notice that while the nonfaulty process in the example in the previous paragraph does not know that the nonfaulty processes are all deciding on the value 0, it might know that *if it is nonfaulty*, then they are all deciding on the value 0. This motivates

the following definition: Given a nonrigid set $\mathcal{S}$ and a process i, define $B_i^{\mathcal{S}}\varphi$ to be an abbreviation for $K_i(i \in \mathcal{S} \Rightarrow \varphi)$. Thus,

$$(\mathcal{I}, r, m) \models B_i^{\mathcal{S}}\varphi \text{ iff } (\mathcal{I}, r', m') \models \varphi \text{ for all } (r', m') \text{ such that}$$
$$r_i(m) = r'_i(m') \text{ and } i \in \mathcal{S}(r', m').$$

Thus, $B_i^{\mathcal{S}}\varphi$ holds if and only if i knows that *if it is in* $\mathcal{S}$, then φ holds. It is easy to check that $B_i^{\mathcal{S}}$ satisfies the S5 properties as discussed in Chapter 2, except for the Knowledge Axiom ($B_i^{\mathcal{S}}\varphi \Rightarrow \varphi$). Nevertheless, the Knowledge Axiom is satisfied at points where i is in the nonrigid set $\mathcal{S}$. That is, if $i \in \mathcal{S}(r, m)$, then $(\mathcal{I}, r, m) \models B_i^{\mathcal{S}}\varphi \Rightarrow \varphi$ (see Exercise 6.11). In general, it may be better to view $B_i^{\mathcal{S}}$ as a notion of *belief* rather than knowledge.

Corresponding to the nonrigid set $\mathcal{S}$, we add two modal operators to the language, $E_{\mathcal{S}}$ and $C_{\mathcal{S}}$. We define $E_{\mathcal{S}}\varphi$ as $\bigwedge_{i\in\mathcal{S}} B_i^{\mathcal{S}}\varphi$. Thus,

$$(\mathcal{I}, r, m) \models E_{\mathcal{S}}\varphi \text{ iff } (\mathcal{I}, r, m) \models B_i^{\mathcal{S}}\varphi \text{ for all } i \in \mathcal{S}(r, m).$$

In other words, *everyone in* $\mathcal{S}$ *knows* φ at the point (r, m) exactly if every process in $\mathcal{S}(r, m)$ knows that if it is in $\mathcal{S}$, then φ holds. Notice that if $\mathcal{S}(r, m)$ is empty, then by definition $E_{\mathcal{S}}\varphi$ holds. The notion of $C_{\mathcal{S}}\varphi$ is now defined as an infinite conjunction in terms of $E_{\mathcal{S}}\varphi$. Defining $E_{\mathcal{S}}^{k+1}\varphi$ inductively as an abbreviation for $E_{\mathcal{S}}E_{\mathcal{S}}^{k}\varphi$, we have

$$(I, r, m) \models C_{\mathcal{S}}\varphi \text{ iff } (\mathcal{I}, r, m) \models E_{\mathcal{S}}^{k}\varphi \text{ for } k = 1, 2, \ldots$$

It is easy to see that if $\mathcal{S}$ is a fixed nonrigid set G, so that $\mathcal{S}(r, m) = G$ for all points (r, m), then $C_{\mathcal{S}}\varphi \equiv C_G\varphi$ (Exercise 6.12). Thus, this definition extends our original definition of $C_G\varphi$ to nonrigid sets. Let us now reconsider the case we mentioned earlier where a nonfaulty process is guaranteed to know that it is nonfaulty. In this case, when $\mathcal{S}$ is the nonrigid set of nonfaulty processes, it is clear that if process i is nonfaulty, then $B_i^{\mathcal{S}}\varphi$ is equivalent to $K_i\varphi$, for every formula φ. Therefore, results we obtain later about common knowledge among nonfaulty processes being attained in SBA (Theorem 6.4.2 and Corollary 6.4.3) would hold also in this case had we used the first definition of common knowledge for nonrigid sets.

Just as with C_G, we can relate $C_{\mathcal{S}}$ to a notion of reachability. Define a point (r', m') to be *$\mathcal{S}$-reachable from a point (r, m) in k steps* ($k \geq 1$) if there exist points $(r_0, m_0), (r_1, m_1), \ldots, (r_k, m_k)$ such that $(r, m) = (r_0, m_0)$, $(r', m') = (r_k, m_k)$ and for all l with $0 \leq l \leq k - 1$, there exists $i \in \mathcal{S}(r_l, m_l) \cap \mathcal{S}(r_{l+1}, m_{l+1})$ such that $(r_l, m_l) \sim_i (r_{l+1}, m_{l+1})$. We say (r', m') is *$\mathcal{S}$-reachable from* (r, m) if (r', m') is $\mathcal{S}$-reachable from (r, m) in k steps for some $k \geq 1$. Now we get the following analogue to Lemma 2.2.1:

Lemma 6.4.1 $(\mathcal{I}, r, m) \models C_{\mathcal{S}}\varphi$ *iff* $(\mathcal{I}, r', m') \models \varphi$ *for all points* (r', m') *that are* $\mathcal{S}$*-reachable from* (r, m).

Proof See Exercise 6.13. ∎

Using Lemma 6.4.1, we can also show that $C_{\mathcal{S}}$ satisfies many of the properties that we showed in Chapter 2 were satisfied by the common knowledge operator C_G (Exercise 6.13). In particular, it satisfies all the properties of S5, except possibly the Knowledge Axiom (with K_i replaced by $C_{\mathcal{S}}$), and it also satisfies the Fixed-Point Axiom and Induction Rule. Moreover, if $\mathcal{S}(r, m) \neq \emptyset$ for all points (r, m) in an interpreted system $\mathcal{I}$, then $C_{\mathcal{S}}$ satisfies the Knowledge Axiom in $\mathcal{I}$ as well. Finally, $C_{\mathcal{S}}$ satisfies the following property, of which we make frequent use:

$$\models i \in \mathcal{S} \Rightarrow (B_i^{\mathcal{S}} C_{\mathcal{S}}\varphi \Leftrightarrow C_{\mathcal{S}}\varphi).$$

It turns out to be useful to define a nonrigid version of distributed knowledge as well as a nonrigid version of common knowledge. If $\mathcal{S}$ is a nonrigid set, we define $D_{\mathcal{S}}$ as follows:

$$(\mathcal{I}, r, m) \models D_{\mathcal{S}}\varphi \quad \text{iff} \quad (\mathcal{I}, r, m) \models D_G(G \subseteq \mathcal{S} \Rightarrow \varphi) \text{ for } G = \mathcal{S}(r, m).$$

Thus, for example, $D_{\mathcal{N}}\varphi$ holds at the point (r, m) if the nonfaulty processes at (r, m) have distributed knowledge of the fact that if they are nonfaulty, then φ holds. The condition $G \subseteq \mathcal{S}$ here is analogous to $i \in \mathcal{S}$ in the definition of $B_i^{\mathcal{S}}\varphi$. Although $D_{\mathcal{S}}\varphi$ is defined in terms of $D_G\varphi$, the fact that $\mathcal{S}$ is a nonrigid set causes $D_{\mathcal{S}}$ and D_G to have quite different properties. For example, negative introspection does not hold in general for $D_{\mathcal{S}}$, although it does for D_G (Exercises 2.10 and 6.14). The differences between $D_{\mathcal{S}}$ and D_G are investigated in Exercise 6.14.

Using $C_{\mathcal{N}}$, we can get an analogue of Proposition 6.1.2 for SBA.

Theorem 6.4.2 *Let* (γ, π) *be a ba-compatible interpreted context and let* P *be a deterministic protocol. If* $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ *satisfies* σ^{sba}*, then* $\mathcal{I} \models \mathit{deciding}_{\mathcal{N}}(y) \Rightarrow C_{\mathcal{N}}(\mathit{deciding}_{\mathcal{N}}(y))$.

Proof The proof of this theorem is, not surprisingly, similar to that of Proposition 6.1.2. We leave details to the reader (Exercise 6.15). ∎

As with coordinated attack, Theorem 6.4.2 does not hold for nondeterministic protocols (Exercise 6.16). And again, as we shall see in in Chapter 7, by putting some further (but quite reasonable) restrictions on the contexts we consider (in particular,

the Ψ condition in these contexts), we can extend Theorem 6.4.2 so that it holds for nondeterministic protocols as well (Exercise 7.9).

Theorem 6.4.2 states that it is valid in a system $\mathcal{I}$ satisfying σ^{sba} that whenever the nonfaulty processes decide on the value y, then the fact that they are deciding is common knowledge among the nonfaulty processes, that is, $C_{\mathcal{N}}(\mathit{deciding}_{\mathcal{N}}(y))$ holds. The theorem tells us that, just as in the case of coordinated attack, common knowledge plays a fundamental role in SBA. Unlike coordinated attack though, in the contexts of interest to us for SBA, common knowledge will be attainable. Our interest lies in *how long* it will take to attain it.

In the coordinated attack problem we required that the generals attack only if at least one message is delivered; as a consequence, we could prove a corollary to Proposition 6.1.2 showing that an attack requires common knowledge of the fact that at least one message has been delivered. In the case of SBA, the processes are not allowed to decide on the value 1 when all initial preferences are 0. This implies that when they decide on the value 1 it must be the case that some process has an initial preference of 1. We thus obtain the following corollary of Theorem 6.4.2.

Corollary 6.4.3 *Let (γ, π) be ba-compatible and let P be a deterministic protocol. If $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{sba}, then $\mathcal{I} \models \mathit{deciding}_{\mathcal{N}}(y) \Rightarrow C_{\mathcal{N}}(\exists y)$.*

Note that we can also prove that $C_{\mathcal{N}}(\mathit{delivered})$ must hold when the processes decide. But in the contexts of most interest to us here, communication is reliable and immediate. In these contexts there is no difficulty attaining $C_{\mathcal{N}}(\mathit{delivered})$. In fact, if $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$, where P is a protocol that requires that every process send a message in the first round and (γ, π) is a context where communication is reliable and immediate, then $C_{\mathcal{N}}(\mathit{delivered})$ holds at time 1 in every run of $\mathcal{I}$.

We remark that neither Theorem 6.4.2 nor Corollary 6.4.3 would have held in the case of general-omission failures had we used the first definition of common knowledge for nonrigid sets (see Exercise 6.17). On the other hand, there are variants of SBA for which the first definition is appropriate (see Exercise 6.18).

6.5 Attaining SBA

Corollary 6.4.3 shows that attaining common knowledge that some process had initial preference y is necessary in order to decide on the value y. One of our goals here is to show that it is a sufficient condition as well, by describing a program that attains SBA by deciding which of $C_{\mathcal{N}}(\exists 0)$ or $C_{\mathcal{N}}(\exists 1)$ holds.

Let $decided_i$ be an abbreviation of $decided_i(0) \vee decided_i(1)$, so that $decided_i$ is true if process i has made a decision. Process i's program would have the following form:

case of
if $\neg decided_i \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ **do** $\mathsf{decide}_i(0)$
if $\neg decided_i \wedge \neg B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ **do** $\mathsf{decide}_i(1)$
...

The parts of the program that are not shown (the "...") describe the actions that should be taken in case neither $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ nor $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ holds. We assume that none of these actions include $\mathsf{decide}_i(0)$ or $\mathsf{decide}_i(1)$; thus, process i decides on a value only in the first two lines of the program. The first two lines of the program are not symmetric in the role of 0 and 1. Indeed, it is possible that $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ and $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ could both hold; the asymmetry assures that in this case all the agents decide on the value 0, rather than some deciding on the value 0 and some deciding on the value 1. As it stands, this program is not a (standard) program of the form introduced in Chapter 5; in standard programs, we do not allow tests for knowledge. This is what we call a *knowledge-based* program. We discuss such programs in detail in Chapter 7 and, in particular, present a knowledge-based program for SBA along the lines of this program. Here we discuss this program at an intuitive level, as motivation for what we do in the rest of this chapter.

Informally, we can argue that for a program of this form, provided that the decision property holds (so that all nonfaulty processes do eventually decide), the agreement, validity, and simultaneity properties must hold as well. By the properties of $B_i^{\mathcal{N}}$ and $C_{\mathcal{N}}$ discussed in Section 6.4, if i is nonfaulty at a point (r, m), then $(\mathcal{I}, r, m) \models B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1) \Rightarrow \exists 1$, and $\exists 1$ clearly does not hold if no process has initial preference 1. Hence, if all processes have initial preference 0, then the processes cannot decide on the value 1. A similar comment applies when we reverse the roles of 0 and 1. Therefore, if the processes do decide, then the validity property must hold. For the simultaneity and agreement properties, suppose i and j are nonfaulty, m is the first round where some nonfaulty process makes a decision, and i decides on the value 0 at round m. According to the program, this means that $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ must hold. Using the properties of $C_{\mathcal{N}}$ discussed in Exercise 6.13, it follows that $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \Rightarrow C_{\mathcal{N}}(\exists 0)$ and $C_{\mathcal{N}}(\exists 0) \Rightarrow B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$. Thus, j also decides on the value 0 in round m. Similar arguments can be used to show that if i decides on the value 1 at round m, then so does j. These informal arguments can be completely formalized once we give precise semantics to knowledge-based programs; we defer further discussion of this

issue to Section 7.4 (see, in particular, Theorem 7.4.1). We now turn our attention to the problem of attaining common knowledge.

We are interested not only in finding protocols that attain SBA, but in finding ones that attain SBA as soon as possible, in a sense to be made precise shortly. Our discussion suggests that this reduces to finding protocols that attain common knowledge of some initial preference as soon as possible. In this section, we focus on the problem of how soon such common knowledge can be attained in three contexts of interest. These are contexts where communication is reliable and immediate, and where, in addition, processes suffer either (a) crash failures, (b) sending-omission failures, or (c) general-omission failures. In the literature these three failure types have been called *benign*, since for these failure types the processes are not "actively" trying to disrupt the system.

We begin by defining the three contexts of interest, γ^{cr}, γ^{som}, and γ^{gom}, that capture crash failures, sending-omission failures, and general-omission failures, respectively. Actually, each of these is not a single context, but a family of contexts, one for each pair (n, t) with $t < n$. The appropriate n and t should always be clear from context. We denote the set $\{\gamma^{cr}, \gamma^{som}, \gamma^{gom}\}$ by Γ^{sba}. For $fm \in \{cr, som, gom\}$, the context γ^{fm} has the form $(P_e^{fm}, \mathcal{G}_0, \tau, True)$. Thus, these contexts differ only in the environment's protocol. All these contexts are recording message-passing contexts (as defined in Example 5.2.1). In particular, this means that the processes' local states are histories, so that the processes have perfect recall. We assume that process i's initial state has the form x_i, where x_i represents i's initial preference. We take the environment's state to be of the form $((x_1, \ldots, x_n), h)$, where h is the sequence of joint actions performed thus far. The set $\mathcal{G}_0$ of initial global states consists of the 2^n tuples of the form $(((x_1, \ldots, x_n), \langle\,\rangle), x_1, \ldots, x_n)$.

As in all message-passing contexts, the processes' actions consist of sending messages and internal actions; the only internal action we allow here, however, is decide_i. (We could, of course, allow other internal actions, but decide_i is the only internal action necessary for designing a protocol that attains SBA.) We assume that the environment's actions have the form $(\mathsf{a}_{e1}, \ldots, \mathsf{a}_{en})$, where a_{ei} is either $\mathsf{block}_i(Q)$ or $(\mathsf{fail}_i, \mathsf{block}_i(Q))$, for some subset Q of processes. Intuitively, $\mathsf{block}_i(Q)$ results in the processes in Q not receiving i's messages; $(\mathsf{fail}_i, \mathsf{block}_i(Q))$ results in a situation where process i fails and the only processes to receive its last message before failing are the processes not in Q. We assume that τ is such that all messages sent by nonfaulty processes to nonfaulty processes are delivered in the same round they are sent. (Note that message delivery here is the result of send actions by the processes; we do not assume that the environment performs deliver actions.) It is the environment's protocol that captures the type of faulty behavior allowed and describes what happens to messages sent by or to faulty processes.

All that remains is to describe the environment's protocol in each of these three contexts. We start with the crash-failure case. We assume that any message sent by a nonfaulty process to a nonfaulty process is received in the same round it is sent. That is, a_{ei} is $\mathsf{block}(\emptyset)$ if i is nonfaulty. Thus, the only interesting thing that the environment can do is to decide which processes fail, when they fail, and the subset of processes that receive a message from a failed process in the round that it fails. Since we are considering crash failures, if a process fails in round m, it sends no messages in later rounds. P_e^{cr} can now be easily described: in local state s_e, this protocol nondeterministically performs an action $(\mathsf{a}_{e1}, \ldots, \mathsf{a}_{en})$ of the form $\mathsf{block}_i(Q)$ or $(\mathsf{fail}_i, \mathsf{block}_i(Q))$, with the restrictions that (1) if i has not failed yet according to s_e, then the component a_{ei} of $P_e(s_e)$ is either $\mathsf{block}_i(\emptyset)$ or $(\mathsf{fail}_i, \mathsf{block}_i(Q))$, for some arbitrary subset Q of processes, (2) if process i has already failed according to s_e, then the component a_{ei} of $P_e(s_e)$ must be $\mathsf{block}_i(\{1, \ldots, n\})$, and (3) the total number of failures cannot exceed t. The effect of clause (2) is to guarantee that no process receives messages from a process after the round in which it crashes. Thus, we are modeling crashed processes as if they continue to function, although their attempts at communication do not succeed. We could have chosen other ways of modeling the situation (for example, by assuming that once a process crashes, its local state becomes a distinguish *crashed* state); our choice results in a smoother transition to omission failures.

In the sending-omissions failure case, the environment's protocol P_e^{som} is the same as P_e^{cr}, except that we modify clause (2) to allow a_{ei} to have the form $\mathsf{block}_i(Q)$ after process i has failed, for any subset Q of processes. That is, some (perhaps even all) messages sent by a process after it has suffered a sending-omission failure may still be received.

In the general-omissions failure case, the environment's protocol P_e^{gom} is the same as P_e^{som}, except that now we allow a_{ei} to have the form $\mathsf{block}_i(Q)$ for $Q \neq \emptyset$ even if i has not failed, with the restriction that if i has not failed, then all the processes in Q must be ones that have already failed. This is how we allow the environment to stop faulty processes from receiving messages. Thus, messages can be blocked by a fault in either the sender or the receiver.

This completes the description of the three contexts of interest. Since all these contexts share a common global state space, we can define a single interpretation π^{sba} such that (γ, π^{sba}) is ba-compatible for $\gamma \in \Gamma^{sba}$.

Now that we have defined these contexts, we can formalize our discussion from Section 6.3 stating that SBA can be attained in $t + 1$ rounds but not any faster.

Theorem 6.5.1 *There are deterministic protocols that attain SBA in $t + 1$ rounds in each of the contexts in Γ^{sba}.*

Theorem 6.5.2 *If P is a deterministic protocol that satisfies σ^{sba} in a context $\gamma \in \Gamma^{sba}$, r is a failure-free run in $\mathbf{R}^{rep}(P, \gamma)$, and P attains SBA in t' rounds in run r, then $t' \geq t + 1$.*

As we said earlier, our main interest lies in finding protocols that attain SBA as soon as possible in the contexts in Γ^{sba}. We wish this to be true in every run, not just the failure-free runs. Moreover, we want to characterize how soon processes can decide in these contexts. A knowledge-based analysis will help us do this. As a by-product of our analysis, we provide proofs of Theorems 6.5.1 and 6.5.2 in the case of crash failures.

We first need some means of comparing the performance of two protocols. Given two protocols P and P' and a context $\gamma \in \Gamma^{sba}$, we say that a run $r \in \mathbf{R}^{rep}(P, \gamma)$ and a run $r' \in \mathbf{R}^{rep}(P', \gamma)$ are *corresponding runs* if they have the same initial state (i.e., $r(0) = r'(0)$) and for all rounds m, the environment performs the same action at round m in r and r'. Since the environment performs the same actions in two corresponding runs, this means that the same processes fail at the same times in each of them. If P is a deterministic protocol and $\gamma \in \Gamma^{sba}$, then a run of $\mathbf{R}^{rep}(P, \gamma)$ is completely determined by its initial state and the actions performed by the environment. Thus, if P and P' are both deterministic protocols and $r \in \mathbf{R}^{rep}(P, \gamma)$, then we can talk about *the* corresponding run $r' \in \mathbf{R}^{rep}(P', \gamma)$.

Using the notion of corresponding runs, we can compare the performance of different protocols. Assume that $\gamma \in \Gamma^{sba}$ and that P and P' are deterministic protocols that satisfy σ^{sba} in (γ, π^{sba}). We say that P *dominates* P' if for every run $r \in \mathbf{R}^{rep}(P, \gamma)$, if the nonfaulty processes decide in round m of r, then they decide no earlier than round m of any corresponding run $r' \in \mathbf{R}^{rep}(P', \gamma)$. P *strictly dominates* P' if P dominates P' and P' does not dominate P. A protocol is called *optimal* for SBA in context γ if it is not strictly dominated by any other protocol for SBA in this context. Finally, a protocol P is called an *optimum* protocol for SBA in γ if P dominates all other protocols for SBA in this context.

We want to show that optimum protocols for SBA exist, and to find them. As a first step, we define a protocol that is optimum as far as attaining knowledge and common knowledge of certain facts of interest.

As we noted, in the knowledge-based program informally presented at the beginning of this section, we did not describe what happens if neither $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ nor $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ holds. Intuitively, if this is the case, then process i should send the other processes messages. But what should these messages say? As we now show, to attain common knowledge as quickly as possible, the best thing to do is for the processes to tell each other everything they know. We make this precise by considering

a particular protocol called the *full-information protocol*. The protocol for process i, denoted FIP_i, is simple: in local state ℓ, process i performs the action $\mathsf{sendall}_i(\textit{local state})$. This action has the effect of sending each process other than i the message ℓ if process i's local state is ℓ.

We use FIP to denote the joint protocol $(FIP_1, \ldots, FIP_n)$. Notice that FIP is a communication protocol; the only actions are the sending of messages. Running FIP, the processes tell each other everything they know at every step. This suggests that processes attain knowledge about the system at least as fast running FIP as they would with any other protocol. As we now show, this is indeed the case.

We say that formula φ is *determined by the initial state* in a ba-compatible interpreted system $\mathcal{I}$ if for every point (r, m) in $\mathcal{I}$, the truth of φ at (r, m) is uniquely determined by the initial global state $r(0)$. Note that in ba-compatible interpreted systems, the formulas $\exists 0$ and $\exists 1$ are determined by the initial state. We say that φ is a *basic formula* if it is of the form $K_i\psi$, $D_{\mathcal{N}}\psi$, $C_{\mathcal{N}}\psi$, or $B_i^{\mathcal{N}}\psi$, where ψ is determined by the initial state.

We can now make precise our intuition that FIP is an optimum protocol for attaining knowledge.

Theorem 6.5.3 *Assume that $\gamma \in \Gamma^{sba}$ and that φ is a basic formula. Also assume that P is a deterministic protocol, $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi^{sba})$, and $\mathcal{I}' = \mathbf{I}^{rep}(FIP, \gamma, \pi^{sba})$. Let $r \in \mathbf{R}^{rep}(P, \gamma)$ and $r' \in \mathbf{R}^{rep}(FIP, \gamma)$ be corresponding runs. Then for all $m \geq 0$, if $(\mathcal{I}, r, m) \models \varphi$ then $(\mathcal{I}', r', m) \models \varphi$.*

Proof See Exercise 6.22. ∎

Corollary 6.5.4 *If $\gamma \in \Gamma^{sba}$ and $\mathcal{I} = \mathbf{I}^{rep}(FIP, \gamma, \pi^{sba})$, then for all runs r in $\mathcal{I}$, there is a time $m \leq t + 1$ such that $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$.*

Proof Fix $\gamma \in \Gamma^{sba}$ and let r be a run in $\mathbf{I}^{rep}(FIP, \gamma, \pi^{sba})$. By Theorem 6.5.1, there is a protocol P that attains SBA by round $t + 1$ in context γ. Let r' be the run corresponding to r in $\mathcal{I}' = \mathbf{I}^{rep}(P, \gamma, \pi^{sba})$. Suppose the processes decide on the value y in round $m \leq t + 1$ of r'. By Corollary 6.4.3, it follows that $(\mathcal{I}', r', m) \models C_{\mathcal{N}}(\exists y)$. By Theorem 6.5.3, it follows that $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists y)$. ∎

Corollary 6.4.3 implies that attaining common knowledge of the existence of a particular initial preference y is a necessary condition for deciding in SBA. Corollary 6.5.4 shows that in the contexts in Γ^{sba}, the processes attain this common knowledge when running the full-information protocol. By Theorem 6.5.3, they in fact attain it as soon as possible when running the full-information protocol. This suggests that to design an optimum protocol for SBA, we should run a full-information

protocol, testing for common knowledge as we go, and decide as sketched in the knowledge-based program at the beginning of this section. As we show in Section 7.4, this approach indeed works. To apply this approach, however, we need some means of testing, and preferably testing *efficiently*, when and whether the common knowledge holds.

It is not hard to see that the processes can compute when $C_{\mathcal{N}}(\exists y)$ holds. Since the number of initial global states with given values of n and t is finite and there are only finitely many patterns of faulty behavior that can occur in the first m rounds for each fixed m, it follows that there are only finitely many distinct prefixes of runs through time m. In fact, there are less than 2^{2mn^2+n} such runs, even in the case of general-omission failures (see Exercise 6.23). For similar reasons, for any fixed local state, the set of points in which a process has this local state is also finite. As a result, we can explicitly perform model checking as described in Section 3.2 to test whether $C_{\mathcal{N}}(\exists y)$ holds. In general, this operation requires an exponential amount of computation (as a function of n). Can we do better? That is the subject of the next section.

6.6 Attaining Common Knowledge

In this section, we focus on two major issues. First, we study when $C_{\mathcal{N}}(\exists y)$ becomes true in runs of the full-information protocol. Second, we consider how hard it is to decide whether $C_{\mathcal{N}}(\exists y)$ holds. The techniques we develop to deal with the first issue allow us to deal with the second one as well. Our analysis focuses on the crash-failure mode. Thus, we mainly study the interpreted system $\mathcal{I}^{cr} = \mathbf{I}^{rep}(FIP, \gamma^{cr}, \pi^{sba})$. We start by considering two aspects of runs in the crash-failure mode that play an important role in determining when common knowledge arises. Detailed proofs of the technical results of Sections 6.6.1 and 6.6.2 will be given in Section 6.7.

6.6.1 Clean Rounds

A *clean round* is a round in which no new process failure becomes distributed knowledge among the nonfaulty processes. More formally, let *faulty*(i) be a proposition that holds at a point (r, m) exactly if i is faulty at that point. (We remark that in the course of this section, we shall define a number of new propositions such as *faulty*(i), and extend π^{sba} to give them meaning.) We formally define round m to be clean in run r in an interpreted system $\mathcal{I}$ if, for every process i, if $(\mathcal{I}, r, m) \models D_{\mathcal{N}}(faulty(i))$ then $(\mathcal{I}, r, m-1) \models D_{\mathcal{N}}(faulty(i))$. (Notice that although we are checking for the

truth of the same formula—$D_{\mathcal{N}}(faulty(i))$—at both points, the set $\mathcal{N}$ may refer to different sets of processes at each of these points.) Clean rounds resemble, but do not coincide with, rounds in which no processes actually fail. A round in which a process fails may be clean, provided no process nonfaulty at the end of the round has noticed the failure. Conversely, a round in which no failure occurs might not be clean if a failure that occurred in the previous round is discovered for the first time in that round. Because we are dealing with crash failures, every failure will be discovered at most one round after it has occurred (since no messages are received from a faulty process in the rounds subsequent to its failure).

The importance of clean rounds in the crash-failure mode stems from the fact that whatever is distributed knowledge among the nonfaulty processes at the start of a clean round is known by all the nonfaulty processes following the clean round. (This will be formalized in Theorem 6.7.2.) Using this observation, we can show that once it is common knowledge that a clean round has occurred, then any formula determined by the initial state that is distributed knowledge is also common knowledge. Thus, after the existence of a clean round is common knowledge, common knowledge of many new formulas can easily be attained. This statement is formalized in the next theorem, whose proof we defer to the next section. Let *clean* be a formula that is true at (r, m) if some round $m' \leq m$ is clean in run r. Note that *clean* can be determined from the global state $r(m)$, since we can reconstruct what has happened in the run up to time m by looking at the environment's state.

Theorem 6.6.1 *If φ is a formula determined by the initial state, then*

$$\mathcal{I}^{cr} \models (C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi) \Rightarrow C_{\mathcal{N}}\varphi.$$

Proof See Section 6.7. ∎

Since there are at most t failures, it is easy to see that one of the first $t + 1$ rounds of every run must be clean. Hence it is common knowledge at time $t + 1$ that a clean round has occurred. Since one of $\exists 0$ and $\exists 1$ must be distributed knowledge among the nonfaulty processes, we immediately get the following corollary to Theorem 6.6.1:

Corollary 6.6.2 *Let r be a run of $\mathcal{I}^{cr}$. Then $(\mathcal{I}^{cr}, r, t + 1) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$.*

As we showed earlier, SBA can be attained as soon as either $C_{\mathcal{N}}(\exists 0)$ or $C_{\mathcal{N}}(\exists 1)$ holds. Therefore, in the case of crash failures, it follows from Corollary 6.6.2 that SBA can always be attained by time $t + 1$. This proves Theorem 6.5.1 in the case of crash failures.

Theorem 6.6.1 suggests that in some cases $C_{\mathcal{N}}(\exists y)$ may be attainable even before time $t + 1$. We next consider how soon such common knowledge can occur.

6.6.2 Waste

Obviously, runs in which no failures occur are reasonably well-behaved. It may seem natural to expect that in such runs $C_{\mathcal{N}}(\exists y)$ should be easy to attain, while having many failures in a run could only make things worse. This turns out not to be the case. For example, consider a run r^{bad} in which process 1 detects t failures in the first round. Since no further failures can occur or become distributed knowledge, the second round of the run r^{bad} must be clean. In addition, at the end of the second round all other nonfaulty processes will know that process 1 detected t failures in round 1, since process 1 will send a message about this fact to the other processes. It is not hard to see that it is *common knowledge* at $(r^{bad}, 2)$ that process 1 detected t failures in the first round (Exercise 6.27). It then follows that it is common knowledge at $(r^{bad}, 2)$ that round 2 of r^{bad} was clean. By Theorem 6.6.1, any formula determined by the initial state that is distributed knowledge among the nonfaulty processes is common knowledge at $(r^{bad}, 2)$. In particular, this is the case for at least one of $\exists 0$ and $\exists 1$.

Suppose we view SBA as a game, where an adversary is trying to force the nonfaulty processes to take as many rounds as possible before reaching agreement by choosing the behavior of the faulty processes appropriately. It turns out that, roughly speaking, the adversary's best strategy is to minimize the number of failures, since this increases the processes' uncertainty. As the previous example shows, the adversary's worst strategy is to have all the faulty processes fail right at the beginning of a run. A closer analysis shows that, in fact, if more than m failures have become distributed knowledge by the end of round m, then from the point of view of the ability to delay the first clean round, failures have been "wasted." In particular, if $m + k$ failures are discovered by the end of round m, then there must be a clean round by time $t + 1 - k$; in fact, there must be a clean round between round $m + 1$ and round $t + 1 - k$. Moreover, after the clean round, all correct processes will know that at some point the waste was at least k.

These comments motivate the following definitions: We add $t + 1$ propositions $\#Failed \geq k$ to the language, for $k = 0, \ldots, t$. We interpret $\#Failed$ to be the number of processes that have failed, so that $(\mathcal{I}^{cr}, r, m) \models \#Failed \geq k$ if at least k processes have failed by the end of round m in run r. (Recall that this information is encoded in the environment's state.) We denote by $\#KnownFailed(r, m)$ the number of processes whose failure is distributed knowledge among the nonfaulty processes at the point (r, m). More formally:

$$\#KnownFailed(r, m) =_{\text{def}} \max\{k \mid (\mathcal{I}^{cr}, r, m) \models D_{\mathcal{N}}(\#Failed \geq k)\}.$$

(We remark that we use $\#Failed \geq k$ here rather than $\#Failed = k$, since processes do not in general know the exact number of processes that have failed, just a lower bound on this number.) It is easy to see that the fact that process i has failed is distributed knowledge among the nonfaulty processes at (r, m) exactly if at least one of the nonfaulty processes knows that i has failed. Moreover, if i fails in round m of run r, then by round $m + 1$ of run r all the nonfaulty processes know that i has failed, since they do not receive messages from i in round $m + 1$. Note that $\#KnownFailed(r, 0) = 0$, since by definition no processes have failed at time 0. We define the *difference at* (r, m), denoted $diff(r, m)$, by

$$diff(r, m) =_{\text{def}} \#KnownFailed(r, m) - m.$$

Finally, we define the *waste* (or the *wastefulness*) of a run r of $\mathcal{I}^{cr}$, denoted $\mathcal{W}(r)$, by

$$\mathcal{W}(r) =_{\text{def}} \max_{m \geq 0} diff(r, m).$$

Since $diff(r, 0) = 0$, it follows that $\mathcal{W}(r) \geq 0$.

The greater the wastefulness of a run, the sooner it becomes common knowledge that there has been a clean round. For example, if r^{bad} is the run described at the beginning of this subsection, where t failures are detected in round 1 by process 1, then we have $\mathcal{W}(r^{bad}) = t - 1$, and at $(r^{bad}, 2)$ it is common knowledge among the nonfaulty processes that round 2 must be clean. In general, it can be shown that in every run r, by time $t + 1 - \mathcal{W}(r)$ it is common knowledge among the nonfaulty processes that a clean round has occurred.

Theorem 6.6.3 *If r is a run in $\mathcal{I}^{cr}$, then $(\mathcal{I}^{cr}, r, t + 1 - \mathcal{W}(r)) \models C_{\mathcal{N}}(clean)$.*

Proof This follows from Theorem 6.7.4 below and the fact that a clean round is guaranteed to occur in r by time $t + 1 - \mathcal{W}(r)$. ∎

Theorem 6.6.3 tells us that at time $t + 1 - \mathcal{W}(r)$ it is common knowledge among the nonfaulty processes that a clean round has occurred. In addition, Theorem 6.6.1 tells us that if φ is a formula determined by the initial state, and if it is common knowledge among the nonfaulty processes that a clean round has occurred, then $D_{\mathcal{N}}\varphi \Rightarrow C_{\mathcal{N}}\varphi$ holds, that is, distributed knowledge of φ (among the nonfaulty processes) implies common knowledge of φ. We conclude that $D_{\mathcal{N}}\varphi \Rightarrow C_{\mathcal{N}}\varphi$ holds at time $t + 1 - \mathcal{W}(r)$. We record this fact in the following corollary.

Corollary 6.6.4 *Let r be a run of $\mathcal{I}^{cr}$. If φ is a formula determined by the initial state, then $(\mathcal{I}^{cr}, r, t + 1 - \mathcal{W}(r)) \models D_{\mathcal{N}}\varphi \Rightarrow C_{\mathcal{N}}\varphi$.*

Since, as before, one of $\exists 0$ and $\exists 1$ must be distributed knowledge among the nonfaulty processes, we obtain from Corollary 6.6.4 a strengthening of Corollary 6.6.2:

Corollary 6.6.5 *Let r be an arbitrary run of $\mathcal{I}^{cr}$. Then*

$$(\mathcal{I}^{cr}, r, t + 1 - \mathcal{W}(r)) \models C_{\mathcal{N}'}(\exists 0) \vee C_{\mathcal{N}'}(\exists 1).$$

We can slightly strengthen Corollary 6.6.4 in the case that $t = n - 1$. In this case, a slightly more subtle analysis shows that formulas determined by the initial state that are distributed knowledge become common knowledge at time $n - 1 - \mathcal{W}(r)$ (Exercise 6.28), i.e., at time $t - \mathcal{W}(r)$, rather than at time $t + 1 - \mathcal{W}(r)$. This gives us:

Corollary 6.6.6 *Let r be a run of $\mathcal{I}^{cr}$ and set $T = \min(t, n - 2)$. If φ is determined by the initial state, then $(\mathcal{I}^{cr}, r, T + 1 - \mathcal{W}(r)) \models D_{\mathcal{N}'}\varphi \Rightarrow C_{\mathcal{N}'}\varphi$.*

In fact, the bound implied by Corollary 6.6.6 is tight, in that formulas that are determined by the initial state do not become common knowledge any sooner unless they were common knowledge to start with.

Theorem 6.6.7 *Let $T = \min\{t, n - 2\}$. If φ is a formula determined by the initial state and $m < T + 1 - \mathcal{W}(r)$, then $(\mathcal{I}^{cr}, r, m) \models C_{\mathcal{N}'}\varphi$ iff $(\mathcal{I}^{cr}, r, 0) \models C_{\mathcal{N}'}\varphi$.*

Proof The claim is an immediate corollary of Lemma 6.7.7. ∎

Corollary 6.6.6 and Theorem 6.6.7 show that the wastefulness of a run of *FIP* uniquely determines when the existence of a particular initial preference becomes common knowledge. In particular, in runs with waste $t - 1$, it happens after two rounds; in runs with waste 0, it happens after $t + 1$ rounds. In general, it happens after k rounds if the waste is $t + 1 - k$. Since attaining common knowledge of an initial preference is a necessary and sufficient condition for attaining SBA, this gives us a complete characterization of the number of rounds necessary to attain SBA.

These results show that the greater the wastefulness of a run, the earlier the processes attain common knowledge. Roughly speaking, the more processes are known to have failed, the less a nonfaulty process's uncertainty about what can go wrong, and hence the easier it is to attain common knowledge. This can be viewed as illustrating a weakness of a model that presumes (common knowledge of) an upper bound on the number of possible failures. Consider again the run r^{bad} in which process 1 detects t failures in round 1. We have already observed that $\mathcal{W}(r^{bad}) = t - 1$. If some nonfaulty process in r^{bad} has initial preference 0, then it follows from Corollary 6.6.6 that $(\mathcal{I}^{cr}, r^{bad}, 2) \models C_{\mathcal{N}}(\exists 0)$. As we shall see, this means that

the nonfaulty processes can decide on the value 0 in round 3. Anthropomorphizing somewhat, this means that the nonfaulty processes should be thrilled if t processes fail in the first round. Intuitively, they are taking this as proof that no further failures will take place. In practice, of course, if t processes fail in the first round, the processes should expect more processes to fail.

On the other hand, the assumption of (common knowledge of) an upper bound on the number of possible failures is not so unreasonable in practice. A program specification often requires that the program work correctly so long as the number of failures is bounded and/or the type of failures is restricted. If the bounds/restrictions are chosen appropriately, the runs of the resulting system should represent all but some exceedingly unlikely behaviors of the physical system.

6.6.3 Computing Common Knowledge

As we promised earlier, we now consider how hard it is for a nonfaulty process to test whether $C_{\mathcal{N}}(\exists y)$ holds in a run of $\mathcal{I}^{cr}$. At the end of Section 6.5, we discussed a model-checking algorithm that in general requires an exponential amount of computation. We now discuss a much more efficient approach.

Our first observation is that computing whether $C_{\mathcal{N}}(\exists y)$ holds in a run of *FIP* is bound to take at least exponential time (as a function of n), for what seem to be the wrong reasons. Consider a run in which there are no failures. In this case, each process receives a message consisting of the local state of every other process in every round. An easy argument by induction shows that the messages sent in round k must have length at least $(n-1)^{k-1}$. We have already observed that if there are no failures, then $C_{\mathcal{N}}(\exists y)$ does not hold until time $t+1$. By this time, the processes' local states have size at least $(n-1)^{t+1}$. If n is $O(t)$, then just *reading* a local state will take exponential time!

We can solve this problem quite easily. Rather than send its local state at each round, each process sends a short description of its local state. This message conveys the same information as sending the full local state, without being exponentially long. The basic idea is that all a process needs to know is which processes communicated with each other in previous rounds. This information can be represented quite succinctly.

We represent the first m rounds of the run r of *FIP* in a context $\gamma \in \Gamma^{sba}$ in terms of a graph $G(r, m)$. For each process i and time k with $0 \leq k \leq m$ there is a node $\langle i, k \rangle$ in the graph. Moreover, (1) each node of the form $\langle i, 0 \rangle$ is labeled by process i's initial state $r_i(0)$, (2) there is an edge between nodes $\langle i, k-1 \rangle$ and $\langle j, k \rangle$ labeled "+" if j receives a message from i in round k of r, and (3) there is an edge between nodes

$\langle i, k-1\rangle$ and $\langle j, k\rangle$ labeled "$-$" if j does not receive a message from i in round k of r. If $\ell = r_i(m)$, let $G(\ell)$ be the subgraph of $G(r, m)$ that describes what i has learned about the situation thus far. That is, $G(\ell)$ is identical to $G(r, m)$ except that some edges are missing (because i may not know that certain messages have or have not been received) and some nodes of the form $\langle j, 0\rangle$ may not be labeled (because i may not know some of the initial states of other processes). Notice that $G(\ell)$ has $n(m+1)$ nodes and size $O(mn^2)$ (that is, at most $O(mn^2)$ nodes and edges). Thus, for the first n rounds (which turn out to be the only rounds of interest to us), $G(\ell)$ has size at most $O(n^3)$. (See Exercise 6.24 for more details on the graph $G(r, m)$.) Let FIP'_i be the protocol according to which process i sends the message $G(\ell)$ in local state ℓ, rather than the message ℓ, as in FIP_i. Let FIP' denote the joint protocol $(FIP'_1, \ldots, FIP'_n)$. In the first n rounds of runs of FIP', the processes' local states are of size $O(n^5)$ (since at each step, process i sends and receives at most n messages that have size $O(n^3)$ each, and these are recorded in its history), which at least allows the possibility of polynomial-time algorithms for checking when common knowledge occurs.

It should be clear from the construction of FIP' that the processes do not lose any information by running FIP' rather than FIP. In particular, it is easy to show the following theorem.

Theorem 6.6.8 *Assume that $\gamma \in \Gamma^{sba}$, and let φ be a basic formula. Let r and r' be corresponding runs of $\mathcal{I} = \mathbf{I}^{rep}(FIP, \gamma, \pi^{sba})$ and $\mathcal{I}' = \mathbf{I}^{rep}(FIP', \gamma, \pi^{sba})$, respectively. Then $(\mathcal{I}, r, m) \models \varphi$ if and only if $(\mathcal{I}', r', m) \models \varphi$.*

Proof See Exercise 6.26. ∎

As we already observed (see Exercise 6.13), if i is nonfaulty, then $C_{\mathcal{N}}(\exists y)$ holds iff $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$ holds. Thus, the problem reduces to checking when $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$ holds. This can be easily done. First, process i computes what it knows about the waste. That is, at the point (r, m), process i computes how many processes it knows were faulty at the point (r, m') for each $m' \leq m$, by checking $G(r_i(m))$. (Note that at (r, m), process i may know more about the processes that were faulty at (r, m') for $m' < m$ than it did at (r, m').) It then computes its best estimate for the waste of run r, by subtracting m' from the number of processes it knows were faulty at (r, m'), for all $m' \leq m$. If at the point (r, m), process i knows that the waste is at least $T + 1 - m$, where $T = \min(t+1, n-2)$, then $m \geq T + 1 - \mathcal{W}(r)$, and it follows from Corollary 6.6.6 that for every value y for which process i knows $\exists y$, it also knows that $C_{\mathcal{N}}(\exists y)$ holds.

For $fm \in \{cr, som, gom\}$, let $\mathcal{I}^{fm'} = \mathbf{I}^{rep}(FIP', \gamma^{fm}, \pi^{sba})$. We thus have the following theorem.

Theorem 6.6.9 *There is an algorithm that, given input $r_i(m)$, with $m < n$, decides in time polynomial in n whether $(\mathcal{I}^{cr'}, r, m) \models B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$.*

What happens in the case of sending-omission failures or the case of general-omission failures? Although the details are beyond the scope of this book, it turns out that we can perform a combinatorial analysis of the runs of $\mathcal{I}^{som'}$ and $\mathcal{I}^{gom'}$ similar to the analysis we have just carried out for $\mathcal{I}^{cr'}$. In the case of $\mathcal{I}^{som'}$, we can again show that there is a polynomial-time algorithm for testing common knowledge. In the case of $\mathcal{I}^{gom'}$, however, the problem of testing common knowledge is *NP*-hard. (Recall that complexity-theoretic notions such as *NP*-hardness were discussed in Section 3.5.) The situation is summarized by the following theorem.

Theorem 6.6.10 *There is an algorithm that, given input $r_i(m)$, with $m < n$, decides in time polynomial in n whether $(\mathcal{I}^{som'}, r, m) \models B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$. The corresponding problem for the system $\mathcal{I}^{gom'}$ is NP-hard.*

Theorem 6.6.10 shows that in the case of general-omission failures, resource-bounded processes that are restricted to doing polynomial-time computations are unlikely to be able compute when they have common knowledge of an initial preference. (If they could, then it would follow that $P = NP$, which, as we mentioned in Section 3.5, is considered highly unlikely.) A process that did not suffer from such resource limitations could, of course, compute when this common knowledge arises. As we mentioned earlier, this can be done in exponential time, using the model-checking algorithm of Section 3.2. In fact, this computation can be done in space polynomial in n (see Exercise 6.25).

6.7 Detailed Proofs

In this section, we provide the details of some of the proofs we omitted in Section 6.6. The results of this section are not needed elsewhere in the book.

Before beginning to prove the statements made in Section 6.6, we need a few definitions. First, the notion of a *failure pattern* will play an important role in this section. A failure pattern is a description of which processes are faulty and how the faulty processes behave. In the case of crash failures, a failure pattern can be represented as a set of triples of the form $\langle i, m, Q\rangle$, where i is a process, m is a round number, and Q is a set of processes. A run $r \in \mathbf{R}^{rep}(P, \gamma^{cr})$ *displays the failure pattern* f if f consists of precisely those triples $\langle i, m, Q\rangle$ such that the environment performed the action $(\mathsf{fail}_i, \mathsf{block}_i(Q))$ at round m of run r.

We use $X = (x_1, \ldots, x_n)$ to denote the list of the initial preferences of the processes. Notice that a deterministic protocol P, a failure pattern f, and a vector X uniquely determine a run in the context γ^{cr}. This is the run of $\mathbf{R}^{rep}(P, \gamma^{cr})$ with failure pattern f and initial preferences as determined by X. We denote the run determined by P, f and X by $P(X, f)$.

We start with a preliminary result about distributed knowledge, which is interesting in its own right. Roughly speaking, it shows that distributed knowledge of formulas determined by the initial state cannot be gained in the case of crash failures. (We remark that distributed knowledge can be lost. For example, suppose that in run r of a system $\mathcal{I}$, process 1 is the only process with initial preference 0, and process 1 fails in round 1 and no process receives messages from process 1. By definition, no process is faulty at time 0, so $(\mathcal{I}, r, 0) \models D_{\mathcal{N}}(\exists 0)$, but $(\mathcal{I}, r, 1) \not\models D_{\mathcal{N}}(\exists 0)$.)

Theorem 6.7.1 *Suppose $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma^{cr}, \pi^{sba})$ and φ is a formula determined by the initial state. Then $\mathcal{I} \models \neg D_{\mathcal{N}}\varphi \Rightarrow \Box \neg D_{\mathcal{N}}\varphi$.*

Proof Assume $(\mathcal{I}, r, l) \models \neg D_{\mathcal{N}}\varphi$. We want to show $(\mathcal{I}, r, m) \models \neg D_{\mathcal{N}}\varphi$ for all $m > l$. It clearly suffices to show $(\mathcal{I}, r, l+1) \models \neg D_{\mathcal{N}}\varphi$, since once we show this, the desired result follows by a straightforward induction. From the semantic definition of $D_{\mathcal{N}}\varphi$, it follows that $(\mathcal{I}, r', l) \models \neg\varphi$ for some run r' of $\mathcal{I}$ such that $\mathcal{N}(r, l) \subseteq \mathcal{N}(r', l)$ and $r_i(l) = r'_i(l)$ for all processes $i \in \mathcal{N}(r, l)$. Let $r = P(X, f)$ and $r' = P(X', f')$. Let f'' be a failure pattern such that

- f'' is identical to f' up to round l (so that $\langle i, k, Q\rangle \in f'$ iff $\langle i, k, Q\rangle \in f''$ for $k \leq l$),
- according to f'', all of the processes in $\mathcal{N}(r', l)$ not in $\mathcal{N}(r, l)$ (if any) crash in round $l+1$ and none of their round $l+1$ messages are delivered (so that $\langle i, l+1, \{1, \ldots, n\}\rangle \in f''$ for $i \in \mathcal{N}(r', l) - \mathcal{N}(r, l)$),
- f'' is identical to f from round $l+1$ on for processes not in $\mathcal{N}(r', l) - \mathcal{N}(r, l)$ (so that $\langle i, k, Q\rangle \in f$ iff $\langle i, k, Q\rangle \in f''$ for $k \geq l+1$, $i \notin \mathcal{N}(r', l) - \mathcal{N}(r, l)$).

It is easy to see that the same processes fail in f and f''; hence, no more than t processes fail in f''. Let $r'' = P(X', f'')$. Given that no more than t processes fail in f'', we immediately have that r'' is a run of $\mathcal{I}$. We claim that (a) $(\mathcal{I}, r'', l+1) \models \neg\varphi$, (b) $\mathcal{N}(r, l+1) = \mathcal{N}(r'', l+1)$, and (c) $r_i(l+1) = r''_i(l+1)$ for all $i \in \mathcal{N}(r, l+1)$. We have (a) because $r''(0) = r'(0)$, φ is determined by the initial state, and $(\mathcal{I}, r', l) \models \neg\varphi$. Property (b) follows from the construction of f''. Finally, for property (c), notice that every process $i \in \mathcal{N}(r, l)$ has the same local states at (r, l) and

at (r'', l), since $r_i(l) = r_i'(l)$, and $r_i'(l) = r_i''(l)$. Therefore, all processes in $\mathcal{N}(r, l)$ send the same messages (according to P) in round $l + 1$ of both runs. Furthermore, f'' is constructed in such a way that each nonfaulty process receives messages from the same set of processes in round $l + 1$ of both runs. It thus follows that $(\mathcal{I}, r, l + 1) \models \neg D_{\mathcal{N}}\varphi$, and we are done. ∎

We now examine the relationship between clean rounds, distributed knowledge, and common knowledge. As we mentioned earlier, our interest in clean rounds is motivated by the observation that many formulas that are distributed knowledge before a clean round are known by all the nonfaulty processes after a clean round. This observation is captured by the following theorem. (Recall that a formula φ is stable in an interpreted system $\mathcal{I}$ if once true, it remains true; i.e., if $\mathcal{I} \models \varphi \Rightarrow \Box\varphi$; see Section 4.4.4 and Exercises 4.18 and 4.19.)

Theorem 6.7.2 *Suppose that φ is stable in $\mathcal{I}^{cr}$, and that $(\mathcal{I}^{cr}, r, m - 1) \models D_{\mathcal{N}}\varphi$. If round m of run r is clean, then $(\mathcal{I}^{cr}, r, m) \models E_{\mathcal{N}}\varphi$.*

Proof See Exercise 6.29. ∎

In the interpreted system $\mathcal{I}^{cr}$, processes have perfect recall. As a consequence, if φ is stable, so are $E_{\mathcal{N}}\varphi$ and $C_{\mathcal{N}}\varphi$ (Exercise 6.30). Since formulas determined by the initial state are stable, Theorem 6.7.2 implies that after a clean round all processes know all formulas determined by the initial state that were distributed knowledge at the beginning of the round. Furthermore, Theorem 6.7.1 implies that no additional formulas determined by the initial state will later become known. Thus, following a clean round, all processes will forever have identical knowledge about the initial state. As a result, once it is common knowledge that a clean round has occurred, it is also common knowledge that the processes have an identical view of the initial state. These observations suffice to prove Theorem 6.6.1 from the previous section. We repeat the statement here for convenience.

Theorem 6.6.1 *If φ is a formula determined by the initial state, then*

$$\mathcal{I}^{cr} \models (C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi) \Rightarrow C_{\mathcal{N}}\varphi.$$

Proof Let φ be a formula determined by the initial state. Let (r, m) be a point such that $(\mathcal{I}^{cr}, r, m) \models clean \wedge D_{\mathcal{N}}\varphi$. It follows from Theorem 6.7.1 that $(\mathcal{I}^{cr}, r, l) \models D_{\mathcal{N}}\varphi$ for all $l \leq m$. Since $(\mathcal{I}^{cr}, r, m) \models clean$, some round $l \leq m$ of r must be clean. Notice that $l > 0$, since round 1 takes place between times 0 and time 1. Since $(\mathcal{I}^{cr}, r, l - 1) \models D_{\mathcal{N}}\varphi$, by Theorem 6.7.2 we have that $(\mathcal{I}^{cr}, r, l) \models E_{\mathcal{N}}\varphi$. Since

φ is stable, so is $E_{\mathcal{N}}\varphi$ (Exercise 6.30), and therefore $(\mathcal{I}^{cr}, r, m) \models E_{\mathcal{N}}\varphi$. Since $\mathcal{I}^{cr} \models E_{\mathcal{N}}\varphi \Rightarrow E_{\mathcal{N}}D_{\mathcal{N}}\varphi$ (Exercise 6.31), we have that $(\mathcal{I}^{cr}, r, m) \models E_{\mathcal{N}}D_{\mathcal{N}}\varphi$. It follows that $\mathcal{I}^{cr} \models (clean \wedge D_{\mathcal{N}}\varphi) \Rightarrow E_{\mathcal{N}}D_{\mathcal{N}}\varphi$. Since $C_{\mathcal{N}}(clean) \Rightarrow E_{\mathcal{N}}C_{\mathcal{N}}(clean)$ is valid by the Fixed-Point Axiom (which holds by Exercise 6.13), we have that $\mathcal{I}^{cr} \models C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi \Rightarrow E_{\mathcal{N}}(C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi)$, so from the Induction Rule we get $\mathcal{I}^{cr} \models (C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi) \Rightarrow C_{\mathcal{N}}(C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi)$. It is easy to check that $\models C_{\mathcal{N}}(C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi) \Rightarrow C_{\mathcal{N}}\varphi$. Thus, we get $\mathcal{I}^{cr} \models (C_{\mathcal{N}}(clean) \wedge D_{\mathcal{N}}\varphi) \Rightarrow C_{\mathcal{N}}\varphi$, as desired. ∎

We next want to prove Theorem 6.6.3; thus, we want to show that by round $t + 1 - \mathcal{W}(r)$ of a run r, it is common knowledge that a clean round has occurred. We need to prove some preliminary results first.

We add $t+1$ propositions of the form $\mathcal{W}_{curr} \geq k$, for $k = 0, \ldots, t$ to the language. If $k < t$, we take $\mathcal{W}_{curr} = k$ to be an abbreviation for $(\mathcal{W}_{curr} \geq k) \wedge \neg(\mathcal{W}_{curr} \geq k+1)$; we identify $\mathcal{W}_{curr} = t$ with $\mathcal{W}_{curr} \geq t$. We use $\mathcal{W}_{curr}$ to refer to the waste in the "current" run. Thus, for example, $(\mathcal{I}^{cr}, r, l) \models (\mathcal{W}_{curr} = k)$ holds exactly if $\mathcal{W}(r) = k$. Notice that since the processes generally do not know what run they happen to be in, their knowledge about the waste is only in terms of $\mathcal{W}_{curr}$. We remark that for all runs r, we have $diff(r, 0) = 0$ and that $diff(r, l) \leq t - 1$ for all l. Thus, $0 \leq \mathcal{W}(r) \leq t - 1$. Moreover, since $diff(r, t + 1) < 0$, it follows that at some time $\widehat{l}$ between time 0 and time t the difference reaches its maximum—the wastefulness of the run—for the last time. Round $\widehat{l}+1$ must be clean, since otherwise $diff(r, \widehat{l}+1) \geq diff(r, \widehat{l})$, contradicting the choice of $\widehat{l}$. This leads us to the following:

Lemma 6.7.3 *If* $\mathcal{W}(r) \geq w$ *then* $(\mathcal{I}^{cr}, r, t + 1 - w) \models C_{\mathcal{N}}(\mathcal{W}_{curr} \geq w)$.

Proof Let r be an arbitrary run of $\mathcal{I}^{cr}$ and suppose $\mathcal{W}(r) = w' \geq w$. We start by showing that $(\mathcal{I}^{cr}, r, t + 1 - w) \models E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w)$. Let $\widehat{l}$ be the largest l satisfying $diff(r, l) = w'$. By definition of $\widehat{l}$, we have $\#KnownFailed(r, \widehat{l}) - \widehat{l} = w'$. Since $\#KnownFailed(r, \widehat{l}) \leq t$, it follows that $\widehat{l} \leq t - w'$, and hence also $\widehat{l}+1 \leq t+1-w$. Now observe that round $\widehat{l}+1$ of r must indeed be clean, since $diff(r, \widehat{l}+1) < diff(r, \widehat{l})$. Since $\#KnownFailed(r, \widehat{l}) = \widehat{l} + w'$, we have that $(\mathcal{I}^{cr}, r, \widehat{l}) \models D_{\mathcal{N}}(\#Failed \geq \widehat{l}+w')$. As a result, we obtain by definition of $\mathcal{W}$ that $(\mathcal{I}^{cr}, r, \widehat{l}) \models D_{\mathcal{N}}(\mathcal{W}_{curr} \geq w')$. Notice that the truth of $\mathcal{W}_{curr} \geq w'$ is determined by the run, so that $\mathcal{W}_{curr} \geq w'$ is a stable formula. Since round $\widehat{l} + 1$ of r is clean, we have by Theorem 6.7.2 that $(\mathcal{I}^{cr}, r, \widehat{l} + 1) \models E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w')$. Since $\widehat{l} + 1 \leq t + 1 - w$, and since $E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w')$ is stable (by Exercise 6.30, since $\mathcal{W}_{curr} \geq w'$ is stable), it follows that $(\mathcal{I}^{cr}, r, t + 1 - w) \models E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w')$. Moreover, since $w' \geq w$, we obtain that $(\mathcal{I}^{cr}, r, t + 1 - w) \models E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w)$, as desired.

Let r be a run with $\mathcal{W}(r) \geq w$ and let r' be a run such that $(r', t+1-w)$ is $\mathcal{N}$-reachable from $(r, t+1-w)$. (Recall that $\mathcal{I}^{cr}$ is synchronous, so that all points $\mathcal{N}$-reachable from $(r, t+1-w)$ are points at time $t+1-w$.) We prove by induction on k that if $(r', t+1-w)$ is $\mathcal{N}$-reachable from $(r, t+1-w)$ in k steps, then $(\mathcal{I}^{cr}, r', t+1-w) \models \mathcal{W}_{curr} \geq w$. The case $k = 0$ is immediate, since then $r' = r$ and the claim holds by assumption. Assume the claim is true for k, and let $(r', t+1-w)$ be $\mathcal{N}$-reachable from $(r, t+1-w)$ in $k+1$ steps. Let r'' be a run such that $(r'', t+1-w)$ is $\mathcal{N}$-reachable from $(r, t+1-w)$ in k steps and $(r', t+1-w)$ is $\mathcal{N}$-reachable from $(r'', t+1-w)$ in one step. By the induction hypothesis, we have that $(\mathcal{I}^{cr}, r'', t+1-w) \models \mathcal{W}_{curr} \geq w$. This means that $\mathcal{W}(r'') \geq r$. Therefore, by our previous claim, we have that $(\mathcal{I}^{cr}, r'', t+1-w) \models E_{\mathcal{N}}(\mathcal{W}_{curr} \geq w)$. It follows that $(\mathcal{I}^{cr}, r', t+1-w) \models \mathcal{W}_{curr} \geq w$. It now follows from Lemma 6.4.1 that $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}(\mathcal{W}_{curr} \geq w)$. ∎

We can now use Lemma 6.7.3 to prove that the precise waste of a run becomes common knowledge at time $t+1-\mathcal{W}_{curr}$.

Theorem 6.7.4 *If $\mathcal{W}(r) = w$ then $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}(\mathcal{W}_{curr} = w)$.*

Proof Given Lemma 6.7.3, it suffices to show that $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}\neg(\mathcal{W}_{curr} \geq w+1)$. Assume by way of contradiction that $(\mathcal{I}^{cr}, r, t+1-w) \not\models C_{\mathcal{N}}\neg(\mathcal{W}_{curr} \geq w+1)$. Then there is a point $(r', t+1-w)$ that is $\mathcal{N}$-reachable from $(r, t+1-w)$ for which $\mathcal{W}(r') \geq w+1$. By Lemma 6.7.3 we have $(\mathcal{I}^{cr}, r', t+1-w') \models C_{\mathcal{N}}(\mathcal{W}_{curr} \geq w+1)$. Since $(r, t+1-w)$ is $\mathcal{N}$-reachable from $(r', t+1-w)$, it follows that $\mathcal{W}(r) \geq w+1$. But this contradicts our assumption that $\mathcal{W}(r) = w$. ∎

We can now prove Theorem 6.6.3, which we restate.

Theorem 6.6.3 *If r is a run in $\mathcal{I}^{cr}$, then $(\mathcal{I}^{cr}, r, t+1-\mathcal{W}(r)) \models C_{\mathcal{N}}(clean)$.*

Proof Assume that $\mathcal{W}(r) = w$. Then there must be a clean round in r by time $t+1-w$ (Exercise 6.33). Thus, $(\mathcal{I}^{cr}, r, t+1-w) \models (\mathcal{W}_{curr} = w) \Rightarrow clean$. Hence, by Exercise 6.13, $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}(\mathcal{W}_{curr} = w) \Rightarrow C_{\mathcal{N}}(clean)$. By Theorem 6.7.4, we know that $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}(\mathcal{W}_{curr} = w)$. Hence, $(\mathcal{I}^{cr}, r, t+1-w) \models C_{\mathcal{N}}(clean)$, as desired. ∎

Finally, we prove Theorem 6.6.7, which states that formulas determined by the initial state do not become common knowledge in a run r in $\mathcal{I}^{cr}$ in fewer than $t+1-\mathcal{W}(r)$ rounds unless they were common knowledge to start with. This result

follows from three basic lemmas. The first is somewhat technical. It states that before time $t + 1 - \mathcal{W}(r)$, a point differing from the current point only in that the last process known to have failed never fails, is $\mathcal{N}$-reachable from the current point. To make this precise, we give the following definition: Given a failure pattern f, the failure pattern f^{-i} is defined to be $f - \{\langle i, k, Q\rangle\}$ if there is a triple of the form $\langle i, k, Q\rangle$ in f, and to be f if i is not designated to fail according to f. Given a run $r = FIP(X, f)$, we define r^{-i} to be $FIP(X, f^{-i})$. Finally, we say that the failure of j is *discovered at time m* in a given run if time m is the first time at which j's failure is distributed knowledge among the nonfaulty processes. Recall that this means that m is the first time that some nonfaulty process knows that j is faulty. We can now show the following lemma.

Lemma 6.7.5 *Let r be a run of $\mathcal{I}^{cr}$ and let $T = \min\{t, n-2\}$. If $\mathcal{W}(r) \leq T - m$ and no process failure is discovered in r at a later time than process i's failure, then (r, m) is $\mathcal{N}$-reachable from (r^{-i}, m).*

Proof For ease of exposition, we prove the claim assuming $t \leq n-2$. The case of $t = n-1$ is left as Exercise 6.34. Thus, $T = t$, and we use t for T in the proof. If i does not fail in r then $r = r^{-i}$, and the claim trivially holds. So assume that i fails in r, and let l be the round in which i's failure is discovered in r. By assumption, no process failure is discovered in r at a later time. If $l > m$ then for every nonfaulty process $j \in \mathcal{N}(r, m)$ we have $r_j(m) = r_j^{-i}(m)$ and thus clearly (r, m) is $\mathcal{N}$-reachable from (r^{-i}, m). It remains to prove the result for $l \leq m$. We do this by induction on $k = m - l$.

Case $k = 0$ (i.e., $l = m$): First suppose that some process $j \in \mathcal{N}(r, m)$ received a message from i in round m. Then clearly j cannot distinguish (r, m) from (r^{-i}, m), and we are done. So suppose that no process in $\mathcal{N}(r, m)$ received a message from i in round m. Let $j \neq j'$ be distinct processes such that $j, j' \in \mathcal{N}(r, m)$. Such processes exist by the assumption that $t \leq n-2$. Clearly, $r_j(m)$ is independent of whether or not j' receives a message from i in round m. Thus, (r, m) is $\mathcal{N}$-reachable from (r', m), where r' differs from r only in that i fails in round l of r' (i may have failed in round $l-1$ of r and still sent messages to all the nonfaulty processes), but j' does receive a message from i in round m of r'. By our earlier arguments, since j' receives a message from i in r', we have that $((r')^{-i}, m)$ is $\mathcal{N}$-reachable from (r', m). By the transitivity of $\mathcal{N}$-reachability, it follows that $((r')^{-i}, m)$ is $\mathcal{N}$-reachable from (r, m). Finally, observe that $r^{-i} = (r')^{-i}$. Thus, it follows that (r^{-i}, m) is $\mathcal{N}$-reachable from (r, m).

Case $k > 0$ (i.e., $l < m$): Assume inductively that the claim holds for $k-1$. Let $Q = \{j_1, \ldots, j_h\}$ consist of the processes in $\mathcal{N}(r, l)$ to which process i's round l messages are blocked in run r (i.e., $Q = \mathcal{N}(r, l) \cap Q'$ where $\langle i, l, Q' \rangle \in f(r)$). We prove our claim by induction on h. First observe that the case $h = 0$ is vacuously true, since if $h = 0$, then i's failure is discovered only at time $l+1$, which contradicts our assumption about l. Now assume that $h > 0$ and that the claim holds for $h-1$. Let r' be a run that is identical to r except that i fails in round l of r' and $\{j_1, \ldots, j_{h-1}\}$ is the set of processes to which i's round l messages are blocked in run r'. Clearly $(r')^{-i} = r^{-i}$ and, by the induction hypothesis, (r^{-i}, m) is $\mathcal{N}$-reachable from (r', m). Thus, all we have to do is to show that (r', m) is $\mathcal{N}$-reachable from (r, m). We do this by constructing two intermediate runs r_h and r'_h, and showing that (r', m) is $\mathcal{N}$-reachable from (r'_h, m), which is $\mathcal{N}$-reachable from (r_h, m), which in turn is $\mathcal{N}$-reachable from (r, m). The construction of r_h, which involves having process j_h fail, depends on the waste being small enough to give us enough freedom to play with failures between rounds $l+1$ and m. We proceed as follows.

Let r_h be the run that is identical to r up to time l; furthermore, in r_h, process j_h fails in round $l+1$ of r_h, all of the round $l+1$ messages it sends are blocked, and no other processes fail in r_h after round l. We claim that no more than t processes fail in r_h. Since the number of processes that fail in r_h is one more than the number of processes that fail in r, it suffices to show that at most $t-1$ processes fail in r. First notice that the number of processes that fail in r is $\#KnownFailed(r, l)$, since if a process failed at or before round l in r and its failure was not discovered by time l, then its failure would have been discovered by time m, contradicting our assumption that no process failure is discovered in r at a later time than i's failure. Now, $diff(r, l) \leq \mathcal{W}(r) \leq t - m$. Thus, $\#KnownFailed(r, l) = diff(r, l) + l \leq t - m + l = t - (m - l)$. Since we are assuming that $l < m$, we have $\#KnownFailed(r, l) < t$. It now follows that there are no more than t failures in r_h. Since r_h has the same initial state as r and no more than t failures, it follows that r_h is a run of the system.

Clearly $\mathcal{W}(r_h) \leq t - m$, since $diff(r_h, l') = diff(r, l') \leq t - m$ for all $l' \leq l$, and $diff(r_h, l+1) = \#KnownFailed(r_h, l+1) - (l+1) = \#KnownFailed(r_h, l) + 1 - (l+1) = diff(r_h, l) \leq \mathcal{W}(r) \leq t - m$. Notice also that no process fails in r_h after round $l+1$. Thus, since $r = r_h^{-j_h}$ and j_h is the last process whose failure is discovered in r_h, by the induction hypothesis on $k-1$ (applied with respect to j_h), we have that (r_h, m) is $\mathcal{N}$-reachable from (r, m).

Let r'_h be identical to r_h except that i fails in round l and does send a message to j_h (although i still does not send messages to $j_1, \ldots, j_{h-1}$ in round l of r'_h). Choose $j \in \mathcal{N}(r_h, m)$ such that $j \neq j_h$. (Again, this is possible since we are assuming that $t \leq n - 2$.) Clearly j cannot distinguish (r_h, m) from (r'_h, m), so (r'_h, m)

is $\mathcal{N}$-reachable from (r_h, m). Since $r' = r_h'^{-j_h}$, we can now apply the induction hypothesis for $k-1$ to conclude that (r', m) is $\mathcal{N}$-reachable from (r_h', m).

Thus, we have shown that (r', m) is $\mathcal{N}$-reachable from (r_h', m), which is $\mathcal{N}$-reachable from (r_h, m), which in turn is $\mathcal{N}$-reachable from (r, m). Therefore, (r', m) is $\mathcal{N}$-reachable from (r, m). As we showed, this is sufficient to complete the proof. ∎

We can use Lemma 6.7.5 and the fact that all 2^n possible initial states appear in runs of the system, to show that before time $t+1$, all failure-free runs are reachable from each other:

Lemma 6.7.6 *Let $T = \min\{t, n-2\}$. If $m \leq T$ and both r and r' are failure-free runs in $\mathcal{I}^{cr}$, then (r, m) is $\mathcal{N}$-reachable from (r', m).*

Proof Let r and r' be failure-free runs in $\mathcal{I}^{cr}$. We want to show that (r, m) is $\mathcal{N}$-reachable from (r', m). Let $Q = \{j_1, \ldots, j_h\}$ be the set of processes whose initial states in r and r' differ. We prove by induction on h that (r, m) is $\mathcal{N}$-reachable from (r', m). If $h = 0$, then $r = r'$ and we are done. Let $h > 0$ and assume inductively that for all failure-free runs r'' that differ from r' in the initial state of no more than $h-1$ processes, (r', m) is $\mathcal{N}$-reachable from (r'', m). Let r_h be a run with the same initial state as r, in which j_h fails in the first round with all of its round 1 messages being blocked, and no other process fails. Clearly $\mathcal{W}(r_h) = 0 \leq T - m$. Since we also have $r_h^{-j_h} = r$, it follows from Lemma 6.7.5 that (r_h, m) is $\mathcal{N}$-reachable from (r, m). Clearly (r_h, m) is $\mathcal{N}$-reachable from (r_h', m), where r_h' differs from r_h only in that the initial state of j_h in r_h' is as in r'. (Notice that r_h' is a run of the system because all possible initial states appear in the system.) Again by Lemma 6.7.5 we have that (r_h', m) is $\mathcal{N}$-reachable from $(\widehat{r}, m)$, where the initial state of $\widehat{r}$ is as in r_h', and $\widehat{r}$ is failure-free. Since $\widehat{r}$ differs from r' only in the initial states of $h-1$ processes, by the induction hypothesis we have that (r', m) is $\mathcal{N}$-reachable from $(\widehat{r}, m)$ and, by the symmetry and transitivity of $\mathcal{N}$-reachability, we have that (r, m) is $\mathcal{N}$-reachable from (r', m), as desired. ∎

Notice that in the case of crash failures, Theorem 6.5.2 is a corollary of this lemma. For we know by Corollary 6.4.3 that in order for the nonfaulty processes to decide on the value y, it must be the case that $C_{\mathcal{N}}(\exists y)$ holds. But Lemma 6.7.6 implies that $C_{\mathcal{N}}(\exists y)$ cannot hold in a failure-free run before time $t+1$, since all failure free runs are $\mathcal{N}$-reachable from each other before this time.

The next result extends Lemma 6.7.6 to the case where there are failures. Theorem 6.6.7 is a straightforward consequence of this lemma.

Lemma 6.7.7 *Let $T = \min\{t, n-2\}$ and let r and r' be runs of $\mathcal{I}^{cr}$. If $\mathcal{W}(r) \leq T - m$ and $\mathcal{W}(r') \leq T - m$, then (r, m) is $\mathcal{N}$-reachable from (r', m).*

Proof Suppose r and r' are as in the statement of the lemma. Let r_1 be the failure-free run with the same initial state as r and let r_2 be the failure-free run with the same initial state as r'. Observe that for any run r, we have $\mathcal{W}(r^{-i}) \leq \mathcal{W}(r)$. Thus, by repeated applications of Lemma 6.7.5, we have that (r_1, m) is $\mathcal{N}$-reachable from (r, m) and that (r_2, m) is $\mathcal{N}$-reachable from (r', m). By Lemma 6.7.6, we have that (r_1, m) is $\mathcal{N}$-reachable from (r_2, m). The result now follows from the symmetry and transitivity of $\mathcal{N}$-reachability. ∎

Exercises

6.1 Construct a set of runs corresponding to the coordinated attack story described in the text. Show that in the run where all the messages are delivered, after each acknowledgment the depth of knowledge increases by one. That is, show that when B gets A's initial message, $K_B(\mathit{delivered})$ holds; when A gets B's acknowledgment, $K_A K_B(\mathit{delivered})$ holds; when B gets A's acknowledgment of the acknowledgment, $K_B K_A K_B(\mathit{delivered})$ holds; and so on. Show that at no point in this run does $C(\mathit{delivered})$ hold.

6.2 Show that a.m.p. systems display umd, as do a.r.m.p. systems.

6.3 Show that for the nondeterministic protocol P described at the end of Section 6.1, it is possible to construct a ca-compatible interpreted context (γ, π) such that $\mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca} and $\mathcal{R}(P, \gamma)$ displays umd.

6.4 This exercise shows that the umd condition makes coordinating an attack impossible for arbitrary protocols, if we assume that the generals record in their local state whether or not they have attacked. Suppose that $\mathcal{I}$ is a ca-compatible system such that $\mathcal{I} \models \mathit{attacked}_i \Rightarrow K_i \mathit{attacked}_i$, for $i \in \{A, B\}$. (Clearly if the generals record the fact that they have attacked in their local states, then this property will hold.) Define $\mathit{attacked} = (\mathit{attacked}_A \wedge \mathit{attacked}_B)$. Prove that if $\mathcal{I}$ satisfies σ^{ca}, then $\mathcal{I} \models \mathit{attacked} \Rightarrow C(\mathit{attacked})$ and hence that $\mathcal{I} \models \mathit{attacked} \Rightarrow C(\mathit{delivered})$. (Notice that this result shows that analogues of Proposition 6.1.2 and Corollary 6.1.3 hold hold even if $\mathcal{I}$ is of the form $\mathbf{I}^{rep}(P, \gamma, \pi)$, where P is nondeterministic, as long as the generals record the fact that they have attacked. Using Theorem 6.1.1, we immediately get that in the presence of the umd condition, coordinated attack is impossible if the generals record the fact that they have attacked, even if they are following a nondeterministic protocol.)

6.5 Let $\mathcal{G}$ be a set of global states. Show that if P_i is a deterministic protocol, then there is a decision function D such that P_i is compatible with D in all systems $\mathcal{R}$ over $\mathcal{G}$. Show, moreover, that we can take D to be union consistent.

6.6 Prove that the decision function D^{ra} is union consistent.

6.7 Prove that both D^{pr} and D^{um} are union consistent.

* **6.8** This exercise describes a class of scenarios in which the players reach steady state (in the sense mentioned in Section 6.2) and attain common knowledge of the fact that they are in steady state.

Given a set S' of points, define $Init(S') = \{r(0) \mid (r, m) \in S' \text{ for some } m \geq 0\}$. Thus, $Init(S')$ is a set of initial global states, and an initial global state s is in this set exactly if there is a point $(r, m) \in S'$ such that $r(0) = s$. We say that a decision function D on set S *depends on initial states* if $Init(T) = Init(U)$ implies that $D(T) = D(U)$ for all $T, U \subseteq S$.

Let (γ, π^{ag}) be an interpreted context for agreement, such that $\gamma = (P_e, \mathcal{G}_0, \tau, True)$ is a message-passing context in which the only actions are send actions and in which P_e reliably delivers messages in the round they are sent. Let D be a decision function that is defined on all subsets of points and depends on initial states in γ.

(a) Prove that there is a unique joint protocol $P = (P_1, P_2)$ that implements D in context γ. Let $\mathcal{I}^D = \mathbf{I}^{rep}(P, \gamma, \pi^{ag})$.

(b) Show that if $\mathcal{G}_0$ is finite, then for every run r of $\mathcal{I}^D$ there exist actions a and b such that the players eventually attain common knowledge of the fact that they are in steady state and performing these actions. That is, show that

$$\mathcal{I}^D \models \Diamond C\Box(act_1(\mathsf{a}) \wedge act_2(\mathsf{b})).$$

Conclude that for each run r, there is a time m such that

$$(\mathcal{I}^D, r, m) \models C(act_1(\mathsf{a}) \wedge act_2(\mathsf{b})).$$

Thus, at some point in each run it is common knowledge what actions the players stabilize on.

(c) Given a run r of $\mathcal{I}^D$, let $steady(r)$ be the smallest m for which $(\mathcal{I}^D, r, m) \models C\Box(act_1(\mathsf{a}) \wedge act_2(\mathsf{b}))$. Give an optimal bound on the size of $steady(r)$ in $\mathcal{I}^D$, and prove that it is optimal. (Hint: consider the number of information sets each player has initially.)

(d) Show that without the assumption that $\mathcal{G}_0$ is finite, the result of part (b) might be false.

(e) Show that if we allow internal actions as well as send actions, the result of part (b) might be false.

6.9 Suppose we modify the coordinated attack problem so that it becomes more like SBA by (a) assuming that each general initially either prefers to attack or prefers not to attack, and (b) requiring that the generals do what they initially prefer in runs in which they have the same initial preferences. Let (γ, π) be a context compatible with this scenario in which the set $\mathcal{G}_0$ of initial global states contains all four configurations of initial preferences. Show that if P is a deterministic protocol that attains this modified version of coordinated attack in (γ, π), then $\mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{ca}.

6.10 Construct the tables $T_5, \ldots, T_9$ missing from Figure 6.1 in Example 6.3.1.

6.11 Let $\mathcal{S}$ be a nonrigid set of processes in an interpreted system $\mathcal{I}$. Prove that $B_i^{\mathcal{S}}$ satisfies all the S5 properties as defined in Chapter 2 (with K_i replaced by $B_i^{\mathcal{S}}$), except for the Knowledge Axiom. Give an example where the Knowledge Axiom fails, and prove that $B_i^{\mathcal{S}}$ satisfies the following weakening of the Knowledge Axiom:

$$i \in \mathcal{S} \Rightarrow (B_i^{\mathcal{S}}\varphi \Rightarrow \varphi).$$

6.12 Show that if $\mathcal{S}(r, m) = G$ for all points (r, m) in an interpreted system $\mathcal{I}$, then

(a) $\mathcal{I} \models B_i^{\mathcal{S}}\varphi \Leftrightarrow K_i\varphi$ for $i \in G$,

(b) $\mathcal{I} \models E_{\mathcal{S}}\varphi \Leftrightarrow E_G\varphi$,

(c) $\mathcal{I} \models C_{\mathcal{S}}\varphi \Leftrightarrow C_G\varphi$.

6.13 This exercise considers some properties of the operator $C_{\mathcal{S}}$.

(a) Prove Lemma 6.4.1.

(b) Prove that $C_{\mathcal{S}}$ satisfies all the S5 properties as defined in Chapter 2 (with K_i replaced by $C_{\mathcal{S}}$), except for the Knowledge Axiom.

(c) Show that $C_{\mathcal{S}}$ also satisfies the analogue of the Knowledge Axiom in a system $\mathcal{I}$ such that $\mathcal{S}(r, m) \neq \emptyset$ for all points in $\mathcal{I}$. In particular, show that our assumption that $t < n$ implies that $C_{\mathcal{N}}$ satisfies the analogue of the Knowledge Axiom.

(d) Prove that $C_{\mathcal{S}}$ satisfies the Fixed-Point Axiom and Induction Rule of Chapter 2, with C_G replaced by $C_{\mathcal{S}}$ and E_G replaced by $E_{\mathcal{S}}$.

(e) Prove that $\models i \in \mathcal{S} \Rightarrow (B_i^{\mathcal{S}} C_{\mathcal{S}} \varphi \Leftrightarrow C_{\mathcal{S}} \varphi)$.

6.14 This exercise considers some properties of distributed knowledge for nonrigid sets. Let $\mathcal{S}$ be a nonrigid set of processes. Prove that $D_{\mathcal{S}}$ satisfies all the S5 properties as defined in Chapter 2, except the Negative Introspection Axiom (i.e., it satisfies the axioms of the system $S4_n$, as defined in Chapter 3). Provide a counterexample to negative introspection.

6.15 Prove Theorem 6.4.2.

6.16 Show that Theorem 6.4.2 does not hold in general for nondeterministic protocols, by considering a context much like that used in Exercise 6.3.

6.17 Suppose that P is a protocol for SBA in the general-omission failure mode such that $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma^{gom}, \pi^{sba})$ satisfies σ^{sba}.

(a) Show that there are two runs r_1 and r_2 in $\mathcal{I}$ such that

(i) in both runs, process i decides on the value 0 and has the same local state when it decides,

(ii) in r_1 process i is nonfaulty, while in r_2 it is faulty, and in r_2 the nonfaulty processes decide on the value 1.

(b) Use part (a) to prove that $\mathcal{I} \not\models (i \in \mathcal{N} \wedge \mathit{deciding}_{\mathcal{N}}(0)) \Rightarrow K_i \mathit{deciding}_{\mathcal{N}}(0)$. By way of contrast, show $\mathcal{I} \models (i \in \mathcal{N} \wedge \mathit{deciding}_{\mathcal{N}}(0)) \Rightarrow B_i^{\mathcal{N}} \mathit{deciding}_{\mathcal{N}}(0)$.

Part (b) suggests that our definition of $C_{\mathcal{N}}$ is the appropriate one for SBA. (Hint for part (a): choose $n = 2t$, and construct runs that differ only in the identity of the faulty processes and the blame for undelivered messages.)

* **6.18** Define *uniform SBA* just like SBA, but replace the agreement and simultaneity requirements by

- *Uniformity: All* the processes that decide on a value (whether faulty or correct), decide on the same value; moreover, their decision is performed in the same round.

We denote the resulting specification for uniform SBA by σ^{usba}. Uniformity may be a reasonable requirement in the case of sending or general-omission failures; in these cases, we may want all the processes that actually decide (even ones that may have omitted to send or receive a message) to decide on the same value. Given a nonrigid set $\mathcal{S}$, define $(\mathcal{I}, r, m) \models E'_{\mathcal{S}}\varphi$ if $(\mathcal{I}, r, m) \models K_i\varphi$ for all $i \in \mathcal{S}(r, m)$. In addition, define $(\mathcal{I}, r, m) \models C'_{\mathcal{S}}\varphi$ precisely if $(\mathcal{I}, r, m) \models (E'_{\mathcal{S}})^k\varphi$ holds for all $k > 0$. (Note that this is precisely the definition we showed in Exercise 6.17 was inappropriate for SBA in the general-omission failure mode.)

(a) Let (γ, π) be a ba-compatible interpreted context and let P be a deterministic protocol. Suppose that $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ satisfies σ^{usba}. Prove that $\mathcal{I} \models deciding_{\mathcal{N}}(y) \Rightarrow C'_{\mathcal{N}}(\exists y)$.

(b) Let $\gamma \in \{\gamma^{cr}, \gamma^{som}\}$ and let $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi^{sba})$. Assuming that $t \leq n - 2$, prove that for all formulas φ, we have $\mathcal{I} \models C_{\mathcal{N}}\varphi \Leftrightarrow C'_{\mathcal{N}}\varphi$.

(c) Let $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma^{gom}, \pi^{sba})$ and suppose $n > 2t$. Prove that for all formulas φ, we have $\mathcal{I} \models C_{\mathcal{N}}\varphi \Leftrightarrow C'_{\mathcal{N}}\varphi$.

6.19 What can you conclude from Exercises 6.17 and 6.18 about the attainability of uniform SBA in the various failure modes?

6.20 We defined SBA in terms of the processes needing to decide on one of two values: 0 or 1. There are cases in which it is natural to consider agreements on a decision among more than two values. In the following, let *many-valued SBA* refer to a variant of SBA differing from SBA only in that the processes must decide on one of $k > 2$ values.

(a) Show that Corollary 6.4.3 fails for many-valued SBA.

(b) Let $all(y)$ be a proposition that is true at a point (r, m) if all processes have initial preference y in r. Let $\mathcal{I}$ be an interpreted system satisfying the specification of many-valued SBA and suppose $x \neq y$. Prove that $\mathcal{I} \models deciding_{\mathcal{N}}(y) \Rightarrow C_{\mathcal{N}}(\neg all(x))$.

6.21 Define a *receiving-omission* failure to be one where a faulty process fails to receive a message. Describe the context γ^{rom} that captures receiving omissions, and let $\mathcal{I}^{rom} = \mathbf{I}^{rep}(FIP', \gamma^{rom}, \pi^{sba})$. Show that for every run r in $\mathcal{I}^{rom}$, we have $(\mathcal{I}^{rom}, r, 1) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$. We remark that this shows that agreement is attainable in one round if we restrict to receiving-omission faults. (Hint: find an appropriate formula to which the Induction Rule can be applied to show that common knowledge among the nonfaulty processes holds.)

6.22 Suppose P is a deterministic protocol, $\gamma \in \Gamma^{sba}$, $\mathcal{R} = \mathbf{R}^{rep}(FIP, \gamma)$, and $\widehat{\mathcal{R}} = \mathbf{R}^{rep}(P, \gamma)$. Let r and r' be runs in $\mathcal{R}$ and let $\widehat{r}$ and $\widehat{r'}$ be the corresponding runs of $\widehat{\mathcal{R}}$.

(a) Prove that if $r_i(k) = r'_i(k)$ then $\widehat{r}_i(k) = \widehat{r'}_i(k)$.

(b) Use part (a) to prove Theorem 6.5.3.

* **6.23** Show that there are at most 2^{2mn^2+n} distinct prefixes of runs through time m in the system $\mathbf{I}^{rep}(FIP, \gamma^{gom}, \pi^{sba})$. Show that there are at least 2^{n^2-n} runs through time 1 in the system $\mathbf{I}^{rep}(FIP, \gamma^{cr}, \pi^{sba})$.

6.24 The communication graphs $G(r, m)$ were used in Section 6.6.3 to present efficient implementations of the full-information protocol *FIP*. This exercise considers how to define succinct representations of local states using these graphs.

(a) Formally define the graph $G(r, m)$ described in Section 6.6.3.

(b) Define the graph $G(r_i(m))$ corresponding to what process i has learned by the point (r, m).

(c) Describe how the graph $G(r_i(m + 1))$ is obtained from $G(r_i(m))$ and the messages that i receives in round $m + 1$.

(d) What is the size of $G(r_i(m))$ and what is the complexity of constructing $G(r_i(m + 1))$?

* **6.25** The purpose of this exercise is to show that it is possible to test whether $C_{\mathcal{N}'}(\exists y)$ holds at a point (r, m) of $\mathcal{I}^{fm'}$, for $fm \in \{cr, som, gom\}$, using polynomial space computations. (Recall that polynomial space and related complexity-theoretic notions were defined in Section 3.5.)

(a) Describe an effective construction that, given a graph of the form $G(r_i(m))$, yields a graph of the form $G(r', m)$ for some run r' such that $r_i(m) = r'_i(m)$.

(b) Describe a nondeterministic polynomial space algorithm that, given as input two graphs $G(r', m)$ and $G(r'', m)$, accepts if and only if (r'', m) is $\mathcal{N}'$-reachable from (r', m) in $\mathcal{I}^{fm'}$.

(c) Show how parts (a) and (b) can be used to test, given a graph of the form $G(r_i(m))$ capturing i's local state at (r, m), whether $(\mathcal{I}^{fm'}, r, m) \models B_i^{\mathcal{N}'} C_{\mathcal{N}'}(\exists y)$, for $y = 0, 1$.

(d) Finally, prove that testing for $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$ can be performed by a polynomial space computation in $\mathcal{I}^{fm'}$. (Hint: use the fact that polynomial space is the same as nondeterministic polynomial space.)

6.26 Prove Theorem 6.6.8.

6.27 Consider the run r^{bad} described at the beginning of Section 6.6.2. Show that in the case of either crash or sending-omission failures, at the point $(r^{bad}, 2)$ it is common knowledge among the nonfaulty processes that process i detected t failures, and that the second round is clean. (Hint: find an appropriate formula to which the Induction Rule can be applied to show that common knowledge among the nonfaulty processes holds.)

* **6.28** The analysis of clean rounds is slightly different for the case $t = n - 1$. Here, $n - 1$ assumes the role that $t + 1$ has in the case of $t \leq n - 2$. Let $t = n - 1$ and let φ be a formula determined by the initial state.

(a) Prove that $(\mathcal{I}^{cr}, r, n - 1) \models D_{\mathcal{N}}\varphi \Rightarrow C_{\mathcal{N}}\varphi$. (Hint: define *clean'* to be true if either there is only one nonfaulty process or a clean round has occurred, and show that an analogue to Theorem 6.6.1 holds for *clean'*. Then show that $C_{\mathcal{N}}(clean')$ holds at time $n - 1$ when $t = n - 1$.)

(b) Prove the analogue of Theorem 6.6.3 for *clean'*, namely, show that for every run r in $\mathcal{I}^{cr}$, we have $(\mathcal{I}^{cr}, r, n - 1 - \mathcal{W}(r)) \models C_{\mathcal{N}}(clean')$. (Hint: show how to modify the proofs of Lemma 6.7.3 and Theorem 6.7.4 appropriately.)

(c) Using part (b), show that $(\mathcal{I}, r, n - 1 - \mathcal{W}(r)) \models D_{\mathcal{N}}\varphi \Rightarrow C_{\mathcal{N}}\varphi$.

6.29 Prove Theorem 6.7.2.

6.30 In this exercise, we extend some results on stable formulas stated in Exercise 4.18 to nonrigid sets. Prove that in a system where processes have perfect recall, if φ is stable, then so are $E_{\mathcal{N}}\varphi$ and $C_{\mathcal{N}}\varphi$. Show, however, that $D_{\mathcal{N}}\varphi$ might not be stable.

6.31 Prove that $\mathcal{I}^{cr} \models E_{\mathcal{N}}\varphi \Rightarrow E_{\mathcal{N}} D_{\mathcal{N}}\varphi$.

6.32 Prove that $\mathcal{W}(r)$ depends only on the failure pattern of r for every run r in $\mathcal{I}^{cr}$. (Hint: consider two runs r and r' of the full-information protocol that have the same failure pattern, and prove by induction on m that $\#KnownFailed(r, m) = \#KnownFailed(r', m)$.)

6.33 Show that if $\mathcal{W}(r) = w$, then there must be a clean round in r by time $t + 1 - w$.

* **6.34** Show how to modify the proof of Lemma 6.7.5 given for $t \leq n - 2$ for the case of $t = n - 1$.

** **6.35** While our focus here has been on the crash- and omission-failure modes, we can prove corresponding results for the Byzantine failure mode as well.

(a) Define a context γ^{byz} analogous to the contexts in Γ^{sba} but where Byzantine failures are allowed.

(b) Define a notion of corresponding run for the context γ^{byz}.

(c) Prove an analogue of Theorem 6.5.3 for the interpreted context $(\gamma^{byz}, \pi^{sba})$.

6.36 The problem of *weak* SBA, denoted WSBA, differs from SBA in that the validity requirement is changed so that the nonfaulty processes are required to decide on a value v only if all initial preferences are v and no process fails.

(a) Formalize WSBA along the lines of our formalization of SBA.

(b) Prove that, just as in the case of SBA, there is no deterministic protocol P that attains WSBA in fewer than $\min(n - 1 - \mathcal{W}(r), t + 1 - \mathcal{W}(r))$ rounds in any run $r \in \mathbf{R}^{rep}(P, \gamma^{cr})$.

This exercise and the next one show the essential commonality among a number of different variants of SBA that have been considered in the literature. On the other hand, Exercise 6.38 shows that not all variants are identical.

6.37 In the *Byzantine Generals* problem (BG), only one process (the *source*) initially has a value, and the nonfaulty processes need to decide on this value if the source does not fail, and on the same value otherwise. We denote by SBG the simultaneous version of BG, where the processes are required to decide simultaneously.

(a) Formalize SBG along the lines of our formalization of SBA.

(b) Prove that, just as in the case of SBA, there is no deterministic protocol P that attains SBG in fewer than $\min(n - 1 - \mathcal{W}(r), t + 1 - \mathcal{W}(r))$ rounds in any run $r \in \mathbf{R}^{rep}(P, \gamma^{cr})$.

* **6.38** *Simultaneous bivalent agreement* is defined just like SBA except that the validity requirement is replaced by

- *Validity'*: There is at least one run in which the nonfaulty processes decide on the value 0, and at least one in which the nonfaulty processes decide on the value 1.

Construct a protocol for simultaneous bivalent agreement that always halts in two rounds.

* **6.39** *Eventual Byzantine agreement* (EBA) is defined by dropping the simultaneity requirement from SBA: the processes' decisions need not be simultaneous. Suppose P is a protocol for EBA in the case of crash failures. Prove that

(a) there is a run r of P such that $\mathcal{W}(r) = 0$ and not all processes decide before round $t + 1$ in r,

(b) for all j, there is a run r of P such that $\mathcal{W}(r) = j$ and not all processes decide before round $t + 1 - j$ in r.

* **6.40** This exercise considers optimality of protocols with respect to the EBA problem of Exercise 6.39.

(a) Define a notion of optimal and optimum protocols for EBA.

(b) Prove that no optimum protocol for EBA exists even in γ^{cr}. (Hint: for every $y \in \{0, 1\}$, design an EBA protocol that decides very quickly if at least one nonfaulty process has initial preference y. Use the lower bounds given in Sections 6.6 and 6.7 to prove that no protocol can dominate both of the protocols you designed.)

Notes

The coordinated attack problem was introduced by Gray [1978]. It has become part of the folklore of distributed systems; a formal proof of its impossibility (by induction on the number of messages) is given by Yemini and Cohen [1979]. Its relationship to common knowledge was established by Halpern and Moses [1990].

A probabilistic version of the Agreement Theorem (essentially corresponding to the second example of a union-consistent function in Section 6.2) was proved by Aumann [1976]. Aumann's result was the basis for a number of so-called *no-speculation* theorems regarding speculative trading in the stock market, such as that of

Milgrom and Stokey [1982], showing that speculative trading was impossible under certain conditions (along the lines of our third example). Aumann's result was later generalized to the case of decision functions by Cave [1983] and Bacharach [1985]. The first and third examples of union-consistent functions presented here are due to Cave and Bacharach. The intuition we gave for union-consistency is close to that given by Savage [1954] for the *Sure Thing Principle*. Moses and Nachum [1990] discuss the relationship between union consistency and the Sure Thing Principle; in addition, they discuss the fact that the proof of the Agreement Theorem relies heavily on the value of the decision function on sets that are *not* information sets of the players and consider the implications of this fact for the applicability of the Agreement Theorem.

Parikh and Krasucki [1990] study conditions under which an analogue of the Agreement Theorem holds in circumstances where there are more than two players, and they interact in pairs (without public announcements). They show that union consistency is not a sufficient condition in that case, but other conditions are. For more on this and other generalizations of the Agreement Theorem, see the survey paper by Geanakoplos [1992], as well as the work of Rubinstein and Wolinsky [1990]. Geanakoplos and Polemarchakis [1982] prove that if the set of runs is finite, then a protocol by which players repeatedly announce their estimates of the probability of an event e is guaranteed to lead to common knowledge of their respective estimates of the probability of e in finite time. Exercise 6.8 is based on a more general version of this theorem, given in Geanakoplos's survey paper.

The Byzantine agreement problem dates back to the seminal paper of Pease, Shostak, and Lamport [1980], and has attracted a great deal of interest in the literature. A good overview of early work on Byzantine agreement, as well as further pointers to the literature, can be found in Fischer's survey paper [1983]. The fact that requiring *simultaneous* agreement makes a difference was first observed by Dolev, Reischuk, and Strong [1990].

The fact that $t + 1$ rounds are required to attain SBA was proved by Fischer and Lynch [1982] for the case of Byzantine failures, by DeMillo, Lynch, and Merritt [1982] for the case of Byzantine failures with a special mechanism that allows processes to sign the their messages in an "unforgeable" manner, and by Dolev and Strong [1982] in the case of crash failures. The proof given in Section 6.7 (essentially that of Dwork and Moses [1990]) simplifies and generalizes the result of Dolev and Strong.

The relationship between SBA and common knowledge in the crash-failure mode was established by Dwork and Moses [1990]. Their work is the source of the analysis in Sections 6.6 and 6.7. (The communication graph introduced in Section 6.6.3

is due to Merritt [1984].) This work was extended to variants of the omission failure mode by Moses and Tuttle [1988], who proved Theorem 6.6.10. Moses and Tuttle also defined a wide class of problems, called *simultaneous choice problems*, involving simultaneous actions. This class generalizes SBA and includes many related problems. They showed that appropriate common knowledge is a necessary and sufficient condition for the solution of all such problems, and derived optimum knowledge-based and standard programs for all of the problems in this class in the crash- and omission-failure modes.

While *nonrigid* constants, whose meaning changes from world to world, have been studied at length in the context of modal logic (see, for example, the discussion in [Hughes and Cresswell 1968] and [Garson 1977] as well as the discussion in Section 3.7), Dwork and Moses [1990] seem to have been the first to define common knowledge with respect to a nonrigid set of agents. The definitions used in Section 6.4 are taken from [Moses and Tuttle 1988] (except that they used the term "indexical" instead of "nonrigid"). A deeper exploration of the logical issues involved in using modal operators parameterized by nonrigid sets can be found in the work of Grove and Halpern [1991]; Moses and Roth (see [Roth 1989]) consider further applications of nonrigid names to the analysis of distributed systems.

While our focus here has been on the crash- and omission-failure modes, as we mentioned in Exercise 6.35 it is possible to define a notion of corresponding runs for the Byzantine case as well. Michel [1989a, 1989b] considers optimum protocols for SBA and analogues of the protocols FIP and FIP' in the case of Byzantine failures. Michel argues that for Byzantine failures, in contrast to the other failure types, any optimum protocol for SBA must use exponential communication. Michel also looks at protocols from a category-theoretic perspective and proves that the full-information is a *universal* element in an appropriately defined category of protocols. This can be viewed as a generalization of Theorem 6.5.3.

Exercises 6.17 and 6.18 are taken from [Moses and Tuttle 1988] and [Neiger and Tuttle 1993], respectively. Uniform SBA was introduced and studied by Neiger and Toueg [1990].

The *weak* SBA problem of Exercise 6.36 was defined by Lamport; Lamport and Fischer [1982] proved that $t + 1$ rounds are required to attain weak SBA. The EBA problem of Exercise 6.39 was introduced by Pease, Shostak, and Lamport [1980] and further studied by Dolev, Reischuk, and Strong [1990]. Exercises 6.36–6.39 are from [Dwork and Moses 1990], while Exercises 6.21, 6.25, and 6.40 are from [Moses and Tuttle 1988]. A knowledge-based analysis of optimal protocols for EBA was carried out by Halpern, Moses, and Waarts [1990].

Chapter 7

Knowledge-Based Programming

. . . you act, and you know why you act, but you don't know why you know that you know what you do.

Umberto Eco, *The Name of the Rose*

7.1 Knowledge-Based Programs

Our notion of standard programs, in which agents perform actions based on the results of tests that are applied to their local state, is very simple. We argued, however, in Chapter 5 that this notion is rich enough to describe protocols. Nevertheless, standard programs cannot be used to describe the relationships between knowledge and action that we would often like to capture. We already observed this to some extent in our discussion of simultaneous Byzantine agreement in Section 6.5 (we return to SBA in Section 7.4). The issue is perhaps best understood by considering the muddy children puzzle again.

Recall that in the muddy children puzzle, the children are asked by the father if they know whether they have mud on their foreheads. If so, they are supposed to answer "Yes"; otherwise they should answer "No." If, as in Section 2.3, we take the proposition p_i to represent "child i has mud on his forehead," then it seems quite reasonable to think of child i as following the program MC_i:

case of
    if $childheard_i \wedge (K_i p_i \vee K_i \neg p_i)$ **do** say "Yes"
    if $childheard_i \wedge \neg K_i p_i \wedge \neg K_i \neg p_i$ **do** say "No"
end case.

Here $childheard_i$ is a primitive proposition that is true at a given state if child i heard the father's question, "Does any of you know whether you have mud on your own forehead?" in the previous round. Unfortunately, MC_i is not a program as we have defined it. Besides propositional tests, it has tests for knowledge such as $K_i p_i \vee K_i \neg p_i$. Moreover, we cannot use our earlier techniques to associate a protocol with a program, since the truth value of such a knowledge test cannot be determined by looking at a local state in isolation.

We call a program of this form a *knowledge-based* program, to distinguish it from the standard programs defined in Chapter 5. Formally, a knowledge-based program for agent i has the form

case of
  if $t_1 \wedge k_1$ **do** a_1
  if $t_2 \wedge k_2$ **do** a_2
  ...
end case

where the t_j's are standard tests, the k_j's are *knowledge tests* for agent i, and the a_j's are actions of agent i. A knowledge test for agent i is a Boolean combination of formulas of the form $K_i\varphi$, where φ can be an arbitrary formula that may include other modal operators, including common knowledge and temporal operators. Intuitively, the agent selects an action based on the result of applying the standard test to her local state and applying the knowledge test to her "knowledge state." In the program MC_i the test $childheard_i$ is a standard test, while $K_i p_i \vee K_i \neg p_i$ and $\neg K_i p_i \wedge \neg K_i \neg p_i$ are knowledge tests. In any given clause, we can omit either the standard test or the knowledge test; thus, a standard program is a special case of a knowledge-based program. We define a *joint* knowledge-based program to be a tuple $\mathsf{Pg} = (\mathsf{Pg}_1, \ldots, \mathsf{Pg}_n)$, with one knowledge-based program for each agent.

The muddy children example shows that knowledge-based programs are better than standard programs in capturing our intuitions about the relationship between knowledge and action. Even when we can capture our intuitions using a standard program, it may be useful to think at the knowledge level. As the following example shows, knowledge-based programs allow us to look at things in a more high-level way, abstracting away irrelevant details.

Example 7.1.1 Consider the bit-transmission problem from the knowledge-based perspective. The sender S's protocol is to keep sending the bit until an acknowledgment is received from the receiver R. The purpose of the acknowledgment is to inform S that the bit was received by R. Thus, another way to describe the sender's

behavior is to say that S keeps sending the bit until it *knows* that the bit was received by R. This can be described by the knowledge-based program BT'_S:

if $\neg K_S(recbit)$ **do** sendbit.

The advantage of this program over the standard program BT_S in Example 5.3.1 is that it abstracts away the mechanism by which S learns that the bit was received by R. For example, if messages from S to R are guaranteed to be delivered in the same round in which they are sent, then S knows that R received the bit even if S does not receive an acknowledgment.

Now consider the receiver R, which keeps sending acknowledgments after it receives the bit. Surely, if R *knew* that S received the acknowledgment, then R would stop sending it. Thus, it makes sense to replace the standard program BT_R described in Example 5.3.1 by the knowledge-based program BT'_R:

if $recbit \wedge \neg K_R(recack)$ **do** sendack.

An advantage of this program is that if messages from R to S are guaranteed to be delivered in the same round in which they are sent, then R needs to send only one acknowledgment.

The knowledge-based framework enables us to abstract even further. The reason that S keeps sending the bit to R is that S wants R to know the value of the bit. The reason that R keeps sending the acknowledgment to S is that R wants S to know that R knows the value of the bit. Thus, intuitively, S should keep sending the bit until it knows that R knows its value. Let $K_R(bit)$ be an abbreviation for $K_R(bit = 0) \vee K_R(bit = 1)$. Thus, $K_R(bit)$ is true precisely if R knows the value of the bit. The sender's behavior can be described by the knowledge-based program BT''_S:

if $\neg K_S K_R(bit)$ **do** sendbit,

and the receiver's behavior by the knowledge-based program BT''_R:

if $K_R(bit) \wedge \neg K_R K_S K_R(bit)$ **do** sendack.

This program abstracts away the manner in which S learns that R knows the value of the bit and the manner in which R learns that S knows that R knows the value of the bit. If messages are guaranteed to be delivered in the same round that they are sent, then S has to send the bit only once, and R need not send any acknowledgment. Furthermore, if the value of the bit is common knowledge, then BT''_S does not require S to send any messages. In contrast, BT and BT' require S to send messages even in such contexts. Thus, programming at the knowledge level enables us to design more efficient programs. ∎

We have described the syntax of knowledge-based programs, and have provided (by example) some intuition for how knowledge-based programs can be used to give high-level descriptions of the agents' behavior. It remains to give formal semantics to knowledge-based programs. Just as we think of a standard program as inducing a protocol that determines an agent's actions in a given context, we also want to think of a knowledge-based program as inducing a protocol. It is not obvious, however, how to associate a protocol with a knowledge-based program. A protocol is a function from local states to actions. To go from a standard program to a protocol, all we needed to do was to evaluate the standard tests at a given local state, which we did using interpretations. In a knowledge-based program, we also need to evaluate the knowledge tests. But in our framework, a knowledge test depends on the whole interpreted system, not just the local state. It may well be the case that agent i is in the same local state ℓ in two different interpreted systems $\mathcal{I}_1$ and $\mathcal{I}_2$, and the test $K_i p$ may turn out to be true at the local state ℓ in $\mathcal{I}_1$, and false at the local state ℓ in $\mathcal{I}_2$.

To deal with this problem, we proceed as follows. Given an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$, we associate with a joint knowledge-based program $\mathsf{Pg} = (\mathsf{Pg}_1, \ldots, \mathsf{Pg}_n)$ a joint protocol that we denote $\mathsf{Pg}^{\mathcal{I}} = (\mathsf{Pg}_1^{\mathcal{I}}, \ldots, \mathsf{Pg}_n^{\mathcal{I}})$. Intuitively, we evaluate the standard tests in Pg according to π, and evaluate the knowledge tests in Pg according to $\mathcal{I}$. As in the case of standard programs, we require that π be compatible with Pg, that is, that every proposition appearing in a standard test in Pg_i should be local to i. Note that we place the locality requirement only on the propositions appearing in the standard tests, not the propositions appearing in the knowledge tests. We wish to define $\mathsf{Pg}_i^{\mathcal{I}}(\ell)$ for all local states ℓ of agent i. To define this, we first define when a test φ holds in a local state ℓ with respect to an interpreted system $\mathcal{I}$, denoted $(\mathcal{I}, \ell) \models \varphi$.

If φ is a standard test and $\mathcal{I} = (\mathcal{R}, \pi)$, then in analogy to Section 5.3, we define

$$(\mathcal{I}, \ell) \models \varphi \text{ iff } (\pi, \ell) \models \varphi.$$

Since φ is a standard test in Pg_i, it must be local to agent i, so this definition makes sense. If φ is a knowledge test of the form $K_i\psi$, we define

$$(\mathcal{I}, \ell) \models K_i\psi \text{ iff } (\mathcal{I}, r, m) \models \psi \text{ for all points } (r, m) \text{ of } \mathcal{I} \text{ such that } r_i(m) = \ell.$$

Finally, for conjunctions and negations, we follow the standard treatment (see Section 2.1).

Note that $(\mathcal{I}, \ell) \models \varphi$ is defined even if the local state ℓ does not occur in $\mathcal{I}$. In this case it is almost immediate from the definitions that $(\mathcal{I}, \ell) \models K_i(\mathit{false})$, so the Knowledge Axiom fails. On the other hand, if ℓ does occur in $\mathcal{I}$, then K_i behaves in

the standard way. This follows, since if $\ell = r_i(m)$ for some point (r, m) in $\mathcal{I}$, then it is not hard to show that $(\mathcal{I}, \ell) \models K_i\psi$ iff $(\mathcal{I}, r, m) \models K_i\psi$ (Exercise 7.1).

We can now define

$$\mathsf{Pg}_i^{\mathcal{I}}(\ell) = \begin{cases} \{\mathsf{a}_j : (\mathcal{I}, \ell) \models t_j \wedge k_j\} & \text{if } \{j : (\mathcal{I}, \ell) \models t_j \wedge k_j\} \neq \emptyset \\ \{\Lambda\} & \text{if } \{j : (\mathcal{I}, \ell) \models t_j \wedge k_j\} = \emptyset. \end{cases}$$

Intuitively, the actions prescribed by i's protocol $\mathsf{Pg}_i^{\mathcal{I}}$ are exactly those prescribed by Pg_i in the interpreted system $\mathcal{I}$.

Let Pg be a standard program. Then Pg is also a knowledge-based program, with no knowledge tests. Consider an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$. We can associate a protocol with Pg in two ways. We can think of Pg as a standard program, and associate with it the protocol Pg^{π}, or we can think of Pg as a knowledge-based program and associate with it the protocol $\mathsf{Pg}^{\mathcal{I}}$. Our definitions guarantee that these protocols are identical (Exercise 7.2).

The mapping from knowledge-based programs to protocols allows us to define what it means for an interpreted system to be consistent with or to represent a knowledge-based program in a given interpreted context by reduction to the corresponding definitions for protocols. Thus, we say that an interpreted system $\mathcal{I}$ is *consistent with the knowledge-based program* Pg in an interpreted context (γ, π) if π is compatible with Pg and if $\mathcal{I}$ is consistent with the protocol $\mathsf{Pg}^{\mathcal{I}}$ in (γ, π); similarly, $\mathcal{I}$ *represents* Pg in (γ, π) if π is compatible with Pg and if $\mathcal{I}$ represents $\mathsf{Pg}^{\mathcal{I}}$ in (γ, π). This means that to check if $\mathcal{I}$ represents (resp., is consistent with) Pg, we check if $\mathcal{I}$ represents (resp., is consistent with) the protocol obtained by evaluating the knowledge tests in Pg with respect to $\mathcal{I}$ itself. Because of the circularity of the definition, it is not necessarily the case that there is a unique interpreted system representing Pg in a given interpreted context. There may be more than one or there may be none, as we shall see in Section 7.2. In contrast, there can be at most one interpreted system that represents a standard program.

Just as in the case of standard programs, we say that a knowledge-based program Pg *satisfies* the specification σ, or is *correct* with respect to σ, in the interpreted context (γ, π), precisely if every interpreted system representing Pg in (γ, π) satisfies σ. Of course, there may be more than one such system, or none. (We can also define a notion of *strong correctness* for knowledge-based programs analogous to the definition we gave for standard programs. This notion is discussed in Section 7.5.)

We have already seen some examples that demonstrate that knowledge-based programs can be viewed as programs in a very high-level programming language. While this means that knowledge-based programming can be a powerful tool, it has one

significant disadvantage: knowledge-based programs are not directly "executable." To "execute" a program, a process has to be able to evaluate the tests with respect to its local state, but knowledge tests cannot be evaluated with respect to the local state. Thus, we need a way to implement knowledge-based programs by both protocols and standard programs.

We say that a protocol P *is an implementation of*, or *implements*, the knowledge-based program Pg_{kb} in the interpreted context (γ, π) if P coincides with the protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$, where $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ is the interpreted system that represents P in (γ, π). Intuitively, P implements Pg_{kb} if P and Pg_{kb} prescribe the same actions in the interpreted system that represents P. In particular, if P implements Pg_{kb} in (γ, π), then the interpreted system $\mathbf{I}^{rep}(P, \gamma, \pi)$ representing P in (γ, π) also represents Pg_{kb} in this interpreted context. Thus, if Pg_{kb} satisfies some specification σ, then P also satisfies σ.

We can now define implementation by standard programs. We say that a standard program Pg_s *is an implementation of*, or *implements*, the knowledge-based program Pg_{kb} in the interpreted context (γ, π) if the protocol Pg_s^{π}, obtained from Pg_s by using the interpretation π, implements Pg_{kb} in (γ, π). Note that, in general, just as there may not be a unique interpreted system representing a knowledge-based program, there may be more than one standard program implementing a knowledge-based program, or there may be none.

Example 7.1.2 Let us return to the bit-transmission problem once more. Recall from Example 5.4.1 that we considered two specifications: σ' is the run-based specification $\Diamond recbit \wedge \Diamond\Box\neg sendbit$, and σ'' is the knowledge-based specification $\Diamond K_S K_R(bit) \wedge \Box(K_S K_R(bit) \Rightarrow \neg sendbit)$. We saw a standard program BT for the bit-transmission problem in Example 5.3.1, and two knowledge-based programs, BT' and BT'', in Example 7.1.1. We observed in Example 5.4.1 that BT satisfies both σ' and σ'' in the interpreted context $(\gamma_{fair}^{bt}, \pi^{bt})$, where all messages sent infinitely often are eventually delivered.

What can we say about the knowledge-based programs? We are interested in two questions: whether the standard program BT implements BT' and BT'', and whether BT' and BT'' satisfy the specifications σ' and σ''. The answer, of course, depends on the context. We sketch some aspects of the situation here, leaving details to the reader (Exercise 7.3).

Consider the interpreted context $(\gamma_{fair}^{bt}, \pi^{bt})$. The standard program BT implements both of the knowledge-based programs BT' and BT'' in $(\gamma_{fair}^{bt}, \pi^{bt})$. We shall give a result (Theorem 7.2.4) from which it follows that there is a unique interpreted system representing each of BT' and BT'' in this interpreted context; thus,

this interpreted system must in fact be $\mathbf{I}^{rep}(\mathsf{BT}, \gamma^{bt}_{fair}, \pi^{bt})$. Since, as we showed in Example 5.4.1, BT satisfies both σ' and σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$, it follows that BT′ and BT″ satisfy both specifications in this interpreted context as well. ∎

7.2 Getting Unique Representations

As we mentioned in the previous section, in general there is no unique interpreted system that represents a knowledge-based program in a given context. The following example illustrates how this can occur.

Example 7.2.1 Suppose we have a system consisting of only one agent, agent 1, who has a bit that is initially set to 0. Suppose agent 1 runs the following simple knowledge-based program NU (for "not unique"):

if $K_1(\Diamond(bit = 1))$ **do** $bit := 1$.

Intuitively, $bit := 1$ has the effect of assigning the value 1 to the bit. According to NU, agent 1 sets the bit to 1 if she knows that eventually the bit is 1, and otherwise does nothing. It should be clear that there are two ways that agent 1 could be consistent with the program: either by never setting the bit to 1 or by setting the bit to 1 in the first round. We can formalize this by considering the context $\gamma^{nu} = (P_e, \mathcal{G}_0, \tau, True)$, defined as follows: We take agent 1's local state to be either 0 or 1; we think of this local state as representing the value of the bit. We take the environment's state to always be λ (the environment plays no role in this example). Since the bit is initially 0, we take $\mathcal{G}_0 = \{(\lambda, 0)\}$. We assume that the environment's action is always Λ, so $P_e(\lambda) = \Lambda$. The agent's action is either Λ or $bit := 1$. The effect of τ is to reset the bit as appropriate; thus, $\tau(\Lambda, \Lambda)(\lambda, k) = (\lambda, k)$ and $\tau(\Lambda, bit := 1)(\lambda, k) = (\lambda, 1)$. This completes the description of γ^{nu}. Finally, we define π^{nu} in the obvious way: $\pi^{nu}((\lambda, k))(bit = 1)$ is true exactly if $k = 1$.

Let r^0 be the run where agent 1 does nothing, starting in the initial state $(\lambda, 0)$; thus, $r^0(m) = (\lambda, 0)$ for all $m \geq 0$. Let r^j, for $j \geq 1$, be the run where agent 1 sets the bit to 1 in round j, after starting in the initial state $(\lambda, 0)$; thus, $r^j(m) = (\lambda, 0)$ for $m < j$, and $r^j(m) = (\lambda, 1)$ for $m \geq j$. It is easy to see that the only runs that we can have in context γ^{nu} are of the form r^j. It is also not hard to see that no run of the form r^j for $j > 1$ can be in an interpreted system consistent with NU. For if r^j is in an interpreted system $\mathcal{I}$ consistent with NU, then since agent 1 sets the bit to 1 in round j of r^j, it must be the case that $(\mathcal{I}, r^j, j - 1) \models K_1(\Diamond(bit = 1))$. But clearly $(r^j, 0) \sim_1 (r^j, j - 1)$. Thus, $(\mathcal{I}, r^j, 0) \models K_1(\Diamond(bit = 1))$. Since $\mathcal{I}$ is

consistent with $\mathsf{NU}^{\mathcal{I}}$, this means that agent 1 should have set the bit to 1 in round 1 of r^j, a contradiction. Thus, the set of runs in any interpreted system consistent with NU must be a nonempty subset of $\{r^0, r^1\}$. Let $\mathcal{R}^j$ be the system consisting of the single run r^j, for $j = 0, 1$, and let $\mathcal{I}^j = (\mathcal{R}^j, \pi^{nu})$. We claim that both $\mathcal{I}^0$ and $\mathcal{I}^1$ represent NU in the context (γ^{nu}, π^{nu}). Clearly in $\mathcal{I}^1$, agent 1 knows $\Diamond(bit = 1)$, since it is true at every point in $\mathcal{I}^1$, so the only possible action she can take is to set the bit to 1 in round 1, which is precisely what she does in r^1. On the other hand, in $\mathcal{I}^0$, agent 1 never knows $\Diamond(bit = 1)$, since it is false at all points in r^0. This means that according to the protocol $\mathsf{NU}^{\mathcal{I}^0}$, agent 1 never sets the bit to 1, so the only run consistent with $\mathsf{NU}^{\mathcal{I}^0}$ is r^0. In fact, it is easy to see that if $\mathcal{R}^2 = \{r^0, r^1\}$, then the interpreted system $\mathcal{I}^2 = (\mathcal{R}^2, \pi^{nu})$ is not consistent with NU (Exercise 7.4), so that in fact $\mathcal{I}^0$ and $\mathcal{I}^1$ are the only interpreted systems consistent with NU.

Now consider the program that intuitively says "set the bit to 1 exactly if you know you will never set the bit to 1." No interpreted system can be consistent with this program, since it amounts to saying "set the bit to 1 exactly if you know you should not." We can capture this intuition by means of the following knowledge-based program NU′:

if $K_1(\neg\Diamond(bit = 1))$ **do** $bit := 1$.

There can be no interpreted system consistent with NU′ in the context (γ^{nu}, π^{nu}): Arguments similar to those used before show that the only runs that can be in an interpreted system consistent with NU′ are r^0 and r^1. Thus, $\mathcal{I}^0$, $\mathcal{I}^1$, and $\mathcal{I}^2$ are the only possible candidates for interpreted systems consistent with NU′. It is straightforward to show that none of these interpreted systems in fact are consistent with NU′ (Exercise 7.4). Hence, there is no interpreted system that is consistent with or represents NU′. We take this to mean that the program NU′ is inconsistent with the interpreted context (γ^{nu}, π^{nu}). ∎

In Example 7.2.1, we saw programs that determine an agent's current actions as a function of his knowledge about the actions that he will perform in the future. This direct reference to knowledge about the future seemed to make it possible to define both nonsensical programs such as NU′, which cannot be implemented by any standard program, and ambiguous programs such as NU, which can be implemented in a number of different ways. We remark that the explicit use of future temporal operators such as $\Diamond$ is not crucial to this example. Essentially the same effect can be achieved without such operators (Exercise 7.5). The programs in Example 7.2.1 are somewhat contrived, and were designed specifically for the purpose of this example. We do not come across such programs in common applications. We now consider an

example from the realm of robot motion planning in which a program with multiple implementations arises in a fairly natural setting.

Example 7.2.2 Consider a mobile robot that travels on a track. For simplicity, we assume that the track has discrete locations numbered 0,1,2,... The robot starts out at location 0 and can move only in the positive direction. Our robot has an imperfect location sensor. For every location $q \geq 0$, it is guaranteed that whenever the robot is at location q, the sensor-reading α will be one of $\{q-1, q, q+1\}$. This sensor provides the robot with information about its current location. We assume that the robot's motion is determined by the environment, and that the only actions the robot can take are halt and the null action Λ. Once the robot performs the halt action, it no longer moves. We are interested in studying programs that the robot can use to ensure that it halts in a given region. We assume that the robot's local state consists only of α, the current reading of the sensor. Moreover, we assume that the environment can, at any given point, choose either to move the robot one step in the positive direction or to let it stay at the same location. In addition, the environment determines the robot's sensor reading, subject to the constraint that the reading is within ± 1 of the robot's actual position. Finally, we assume that the environment is *fair*, in the sense that in runs in which the robot never performs the halt action, the robot moves an infinite number of times.

To reason about this system, we use a language with five primitive propositions: *halted*, p, p', $\alpha = 3$, and $\alpha \in \{3, 4, 5\}$, where, intuitively, *halted* is true if the robot has halted, p is true if the robot's location is one of $\{2, 3, 4\}$, p' is true if the robot's location is one of $\{2, 4, 6\}$, $\alpha = 3$ is true if the robot's sensor reading is 3, and $\alpha \in \{3, 4, 5\}$ is true if the robot's sensor reading is one of 3, 4, or 5. Let (γ, π) be a recording context corresponding to this description. The formal definition of (γ, π) is left to the reader (Exercise 7.6).

Suppose the robot follows the trivial program P^{Λ},

if *true* **do** Λ,

and hence never performs the halt action. Let $\mathcal{I}^{\Lambda}$ be the interpreted system representing this program in the interpreted context (γ, π). It is straightforward to verify that $\mathcal{I}^{\Lambda} \models (K_r p \Leftrightarrow \alpha = 3)$, so that the only points of $\mathcal{I}^{\Lambda}$ at which the robot knows p are those at which the sensor reading is 3. In addition, it can be shown that $\mathcal{I}^{\Lambda} \models \neg K_r p'$: the robot never knows p' in $\mathcal{I}^{\Lambda}$ (Exercise 7.6).

Suppose the robot has a goal of stopping in the region $\{2, 3, 4\}$ (intuitively, at a location where p is true). Formally, we can capture this goal by the specification σ^{mp} (where the superscript *mp* stands for *motion planning*), which is defined by the

formula $\Box(halted \Rightarrow p) \wedge \Diamond(halted)$. The first conjunct of σ^{mp}, which is called the *safety property*, says that the robot halts only in the goal region, while the second conjunct, which is called the *liveness property*, says that the robot eventually halts. Note that σ^{mp} is a run-based specification.

How can we design a program to satisfy this specification? Intuitively, as long as the robot does not know that p holds, it should not halt. On the other hand, once the robot does know that p holds, it should be able to safely perform the halt action and stop in the desired region. Thus, $K_r p$ seems to be both a necessary and a sufficient condition for stopping in the goal region.

Consider the knowledge-based program MP

if $K_r p$ **do** halt.

Clearly MP satisfies the safety property, in that in any interpreted system that represents it, the robot halts only in the goal region. Does it satisfy the liveness property? Not necessarily.

Recall that $K_r p$ is guaranteed to hold when $\alpha = 3$. This follows directly from the precision of the sensor. This suggests that the following standard program, denoted MP_s,

if $\alpha = 3$ **do** halt,

should be an implementation of MP in (γ, π). Indeed, it is not hard to check that this is the case (Exercise 7.6). Unfortunately, the program MP_s does not satisfy the liveness condition of σ^{mp} in (γ, π). There are many runs of MP_s in this context in which $\alpha \neq 3$ holds throughout the run, despite the fact that the robot crosses the goal region and exits it. It follows that, in spite of its "obviousness," the knowledge-based program MP does not satisfy the liveness condition of σ^{mp} in (γ, π) either.

Now consider the program MP'_s:

if $\alpha \in \{3, 4, 5\}$ **do** halt.

This program guarantees that the robot will not stop before reaching position $q = 2$, because $\alpha \in \{3, 4, 5\}$ is not satisfied when $q < 2$. Moreover, when following this program, the robot *is* guaranteed to stop if it ever reaches the position $q = 4$, since at that point the sensor reading must satisfy $\alpha \in \{3, 4, 5\}$. By the fairness assumption about the environment, the robot must be moved at least four times if it never performs the halt action, in which case it would eventually reach the position $q = 4$. It follows that MP'_s satisfies σ^{mp} in (γ, π).

It can be shown (Exercise 7.6) that MP'_s is also an implementation of MP in (γ, π). Thus, despite depending only on what the robot knows about its current position and

making no reference to the robot's future actions, the knowledge-based program MP does not have a unique representation in (γ, π). Interestingly, MP_s and MP'_s are, in a precise sense, the only implementations of MP in (γ, π) (Exercise 7.6).

Notice that by assuming that the robot's local state consists only of the sensor reading, we have implicitly assumed that the robot is not aware of whether it has halted. Having the robot be aware of whether it has halted turns out not to make any difference in this case. It is not hard to show that we would still have the same two distinct implementations of MP even if we add to the robot's local state a *halted* bit, which starts out with value 0 and is set to 1 once the robot has performed the halt action, so that the robot does know whether or not it has halted (Exercise 7.6).

A change that does affect the outcome would be to modify the robot's local state so that it contains the global time m, in addition to the sensor reading α. (Thus, the local states would have the form (m, α).) Let (γ', π') be the interpreted context obtained from (γ, π) by making this change. With this change, we are guaranteed that every system representing MP in the new context is synchronous. The added information that the robot obtains from the fact that the system is synchronous together with the fact that the robot cannot be moved more than one step per round is enough to ensure that MP_s does not implement MP in (γ', π'). Thus, for example, when $m = 3$ and $\alpha = 4$ the robot knows that its position is 3, since at time $m = 3$ it is guaranteed that $q \leq 3$, whereas when $\alpha = 4$ it is guaranteed that $q \geq 3$. It is possible to show that MP has a unique representation in (γ', π'). Indeed, MP'_s implements MP in this context, and as a result MP does satisfy σ^{mp} in (γ', π') (Exercise 7.6). In the sequel we will see that although synchrony by itself is not enough to guarantee that a knowledge-based program is represented by a unique system, and although the lack of future temporal operators is also not sufficient, taken together they do guarantee (for reasonable contexts) that a knowledge-based program is represented by a unique system. ∎

Examples 7.2.1 and 7.2.2 show that a knowledge-based program may have zero, one, or more than one interpreted systems representing it. Is this a problem? Not necessarily. A knowledge-based program should be viewed as a high-level specification; the systems that represent it can be viewed as those systems that satisfy the specification. For example, consider the knowledge-based program NU from Example 7.2.1:

if $K_1(\Diamond(bit = 1))$ **do** $bit := 1$.

This program can be viewed as saying: "if you know that you are going to take an action, then take it as soon as possible." Appropriately, as we have shown, this

program is represented by two interpreted systems, one in which the action is taken and one in which the action is never taken. On the other hand, the program NU′ can be viewed as saying "if you know that you are not going to take an action, then take it as soon as possible." Not surprisingly, there are no systems representing this program; it is an unsatisfiable specification.

A standard program (in a given interpreted context) is a complete description of the behavior of the agents; this is not the case in general with a knowledge-based program. In many situations, however, there is a strong intuition that a knowledge-based program does completely describe the behavior of the agents, and consequently, the program ought to be represented by a unique interpreted system. For example, in the case of the muddy children puzzle, we expect the behavior of the children following the knowledge-based program MC (described at the beginning of the chapter) to be uniquely determined. This also seems to be the case for the programs BT′ and BT″ for the bit-transmission problem. In the remainder of this section, we describe circumstances under which there will be a unique interpreted system representing a knowledge-based program. We start with an informal discussion and then make things more formal.

Why may one feel that there should be a unique interpreted system representing MC? Intuitively, it is because, once we fix the initial set of states, we can start running the program step by step, generating the run as we go. If r is a run over $\mathcal{G}$, the *prefix of r through time m*, denoted $Pref_m(r)$, is the sequence of the first $m+1$ global states in r, i.e., it is a function ρ from $\{0, \ldots, m\}$ to $\mathcal{G}$ such that $\rho(k) = r(k)$ for $k = 0, \ldots, m$. If $\mathcal{R}$ is a set of runs, then $Pref_m(\mathcal{R})$ is the set of prefixes through time m of the runs in $\mathcal{R}$, i.e., $Pref_m(\mathcal{R}) = \{Pref_m(r) \mid r \in \mathcal{R}\}$. If $\mathcal{I} = (\mathcal{R}, \pi)$, we define $Pref_m(\mathcal{I}) = (Pref_m(\mathcal{R}), \pi)$. Suppose we can generate all prefixes of runs through time m. Once we have all prefixes through time m, at any given point (r, m), the children in that situation can determine whether they do indeed know whether their own forehead is muddy, and thus can take the appropriate action at the next step. This allows us to generate all prefixes through time $m+1$.

The key reason that this idea works is that the prefixes that we have already constructed are sufficient to determine the truth of the knowledge tests in the children's program. In general, this might not be the case. To understand why, suppose we have a knowledge-based program $\mathsf{Pg} = (\mathsf{Pg}_1, \ldots, \mathsf{Pg}_n)$, and Pg_i includes a test such as $K_i\varphi$. Suppose that we have indeed constructed all the prefixes of runs of Pg through time m. For agent i to know what actions to perform next at a point (r, m), the knowledge test $K_i\varphi$ has to be evaluated. As long as this can be done solely by considering points of the form (r', m') with $m' \leq m$—intuitively, these are the points we have already constructed—then there is no problem. If, on the other hand, φ is

a temporal formula such as the formula $\Diamond(bit = 1)$ that appears in the program NU in Example 7.2.1, then we may not be able to evaluate the truth of φ in the prefixes we have constructed thus far. Even if φ is a nontemporal formula, there may be a problem. For example, suppose the time m is encoded in the environment's state, and φ is the formula $m \leq 1$, which is true at all time m points with m less than or equal to 1. Then $K_1(m \leq 1)$ may be false at a point $(r, 1)$ if agent 1 does not know the time, i.e., if $(r, 1) \sim_1 (r', k)$ for some point (r', k) where $k > 1$. Note, however, that there is no point that occurs in a prefix through time 1 and "witnesses" the fact that $K_1(m \leq 1)$ fails at $(r, 1)$, since the formula $m \leq 1$ is true at all points of the form $(r', 0)$ or $(r', 1)$. This discussion suggests that to make the inductive construction work, if a test $K_i\varphi$ in the program is false, there must be a "witness" to its falsity in some prefix we have already constructed.

Subtle problems can arise in the interaction between the assumption that witnesses are provided in the sense that we have just described (informally) and the condition Ψ on runs in a context. Suppose we are interested in running the knowledge-based program Pg in the interpreted context (γ, π), where $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$. What should the system representing Pg be? Intuitively, it should consist of all runs in Ψ whose prefixes arise in the inductive construction. Even if, at every step of the inductive construction, the prefixes constructed provide witnesses, it is possible that the condition Ψ does not include a run with a prefix ρ that arises in the construction. This means that we cannot include a run with prefix ρ in the system. This, in turn, might mean that a witness that we counted on in the course of the inductive construction may not occur in the system, thus undermining our evaluation of the tests.

We now show that there is a unique system representing Pg if "witnesses are provided" and if the condition Ψ is "reasonable." Intuitively, the condition we shall require Ψ to satisfy ensures that for every prefix that arises in the inductive construction, there is some run in Ψ with that prefix that we can include in the system we are constructing. We now give some technical definitions that capture these intuitions.

We first formalize what it means for a condition Ψ in a context to be well behaved. Recall that a run r is weakly consistent with a protocol P in context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ if r is consistent with P except that it may not be in Ψ. Intuitively, Ψ is "reasonable" if it does not rule out prefixes that are weakly consistent with P in γ. Formally, we say that a context γ is *nonexcluding* if (a) $\mathcal{G}_0 \cap Pref_0(\Psi) \neq \emptyset$ (note that $Pref_0(\Psi)$ can be viewed as a set of states), and (b) for every protocol P, if a run r is weakly consistent with P in the context γ, and the prefix ρ of r through time m is in $Pref_m(\Psi)$, then there is a run r' with prefix ρ that is consistent with P in γ. Note

that condition (a) gets our inductive construction started (since Ψ cannot exclude all the initial states), and condition (b) guarantees that Ψ does not prevent a prefix ρ that we have constructed in our inductive construction from being extended to a run. Although it may seem difficult to check whether a context is nonexcluding, many contexts of interest are easily shown to be nonexcluding. For one thing, a context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ is guaranteed to be nonexcluding if Ψ is *True*. More generally, in many contexts of interest Ψ constrains only the "limit" behavior of the run, e.g., *Fair* or *Rel* in message-passing contexts. In such cases, it is often not hard to show that the context under consideration is nonexcluding. For example, the context γ^{bt}_{fair} from Example 5.4.1 is nonexcluding (see Exercise 7.7). We remark, however, that the property of being nonexcluding is a property of the context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ as a whole and not a property of Ψ by itself; a context with the condition *Fair* or *Rel* is not necessarily nonexcluding (see Exercise 7.8).

Our main motivation in defining the notion of a nonexcluding context was to provide a sufficient condition for a knowledge-based program to be represented by a unique interpreted system. It turns out, however, that this definition also suffices to enforce a number of other intuitions we have associated with a context (especially the Ψ component of a context) being "reasonable"; see Exercise 7.9.

We now formalize the notion of "witnesses being provided". We say that an interpreted system $\mathcal{I}$ *provides witnesses* for $K_i\varphi$ if, for every point (r, m) of $\mathcal{I}$, if $(\mathcal{I}, r, m) \models \neg K_i\varphi$, then there is a point (r', m') in $\mathcal{I}$ with $m' \leq m$, such that $(r, m) \sim_i (r', m')$ and $(\mathcal{I}, r', m') \models \neg\varphi$. Notice that if $\mathcal{I} = (\mathcal{R}, \pi)$ provides witnesses for $K_i\varphi$ and φ contains no temporal operators, then the truth of $K_i\varphi$ at a point (r, m) of $\mathcal{I}$ is uniquely determined by $Pref_m(\mathcal{R})$—the set of prefixes up to time m of the runs of $\mathcal{I}$. We say that $\mathcal{I}$ provides witnesses for a knowledge-based program Pg if $\mathcal{I}$ provides witnesses for every formula $K_i\varphi$ that appears as a subformula of a knowledge test in Pg.

Our sufficient condition for the existence of a unique representation depends on the assumption that the system representing Pg provides witnesses for Pg. Unfortunately, we do not know in advance what system this is. Thus, we require that a large collection of systems all provide witnesses for Pg. If a system representing Pg exists, it is guaranteed to be in this collection.

We say that an interpreted context (γ, π) provides witnesses for a knowledge-based program Pg if every system $\mathcal{I}'$ of the form $\mathcal{I}' = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$, where $\mathcal{I} = (\mathcal{R}, \pi)$, provides witnesses for Pg. The system $\mathcal{R}$ here can be arbitrary, except that we require that it consist of runs over the global states that arise in the context γ; the interpretation π is the same as in the interpreted context (γ, π). In other words, the context (γ, π) provides witnesses for Pg if all of the systems representing standard

programs of the form $\mathsf{Pg}^{\mathcal{I}}$ in this context provide witnesses for Pg. Note that if (γ, π) provides witnesses for Pg, then we do indeed have the property we desired that any system representing Pg in (γ, π) provides witnesses for Pg, since if $\mathcal{I}$ represents Pg, then, by definition, $\mathcal{I} = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$.

In general, it may be difficult to tell if a given interpreted context (γ, π) provides witnesses for a program Pg. As we shall see, however, if the tests in the program are sufficiently simple, it need not be very difficult. The following lemma gives a useful sufficient condition that guarantees that a context provides witnesses for a program.

Lemma 7.2.3 *Let (γ, π) be an interpreted context. If every system of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$ is synchronous, then (γ, π) provides witnesses for* Pg.

Proof We need to show that every system $\mathcal{I}'$ of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$ provides witnesses for Pg. Thus, we must show that if $K_i\varphi$ is a subformula of a knowledge test in Pg, then $\mathcal{I}'$ provides witnesses for $K_i\varphi$. Since $\mathcal{I}'$ is synchronous by assumption, it certainly suffices to show that every synchronous system provides witnesses for every formula of the form $K_i\varphi$. As we now show, this is almost immediate from the definition of a synchronous system. Recall that in synchronous systems, if $(r, m) \sim_i (r', m')$ then $m = m'$. Suppose $(\mathcal{I}', r, m) \models \neg K_i\varphi$. Then by assumption there is a point (r', m) in $\mathcal{I}'$ with $(r', m) \sim_i (r, m)$ such that $(\mathcal{I}', r', m) \models \neg\varphi$. It follows that $\mathcal{I}'$ provides witnesses for $K_i\varphi$, and we are done. ∎

The second condition in Lemma 7.2.3 requires that every interpreted system of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$ be synchronous. This holds, for example, if the program Pg (in the interpreted context (γ, π)) prescribes that each agent performs exactly one action in every round and if (according to γ) the agents keep track of the actions they have performed in their local states (as is the case in message-passing systems). Under these assumptions, systems of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$ are necessarily synchronous, since an agent can determine the time by looking at his local state.

Lemma 7.2.3 is only a sufficient condition; for example, the context $(\gamma^{bt}_{fair}, \pi^{bt})$ provides witnesses for the programs BT′ and BT″, even though the systems that represent these programs are not synchronous (Exercise 7.7).

We are now ready to state a theorem that gives a useful sufficient condition on a knowledge-based program Pg in an interpreted context (γ, π) that guarantees that there is a unique interpreted system representing Pg in (γ, π). We denote this interpreted system by $\mathbf{I}^{rep}(\mathsf{Pg}, \gamma, \pi)$.

Theorem 7.2.4 *Let* Pg *be a knowledge-based program in which the tests do not involve temporal operators, let γ be a nonexcluding context, and assume that the context (γ, π) provides witnesses for* Pg. *Then there is a unique interpreted system $\mathbf{I}^{rep}(\mathsf{Pg}, \gamma, \pi)$ representing* Pg *in (γ, π).*

Proof The proof formalizes the construction we sketched informally at the beginning of the section. We inductively construct interpreted systems $\mathcal{I}^m = (\mathcal{R}^m, \pi)$ for $m = 0, 1, 2, \ldots$, where π is the interpretation that appears in the statement of the theorem. The prefixes $Pref_m(\mathcal{R}^m)$ correspond to the prefixes in our informal construction. Suppose $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$. The set $\mathcal{R}^0$ consists of all runs r in Ψ such that $r(0) \in \mathcal{G}_0$. (Note that $\mathcal{R}^0$ is nonempty, since γ is nonexcluding.) Let $\mathcal{I}^0 = (\mathcal{R}^0, \pi)$. Suppose we have constructed $\mathcal{I}^m$; we then define $\mathcal{I}^{m+1} = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}^m}, \gamma, \pi)$. We define the system R^ω to consist of all runs r such that $Pref_m(r)$ is in $Pref_m(\mathcal{R}^m)$ for all $m \geq 0$, and let $\mathcal{I}^\omega = (\mathcal{R}^\omega, \pi)$. We would like to say at this point that $\mathcal{I}^\omega$ is the unique interpreted system representing Pg in the interpreted context (γ, π). This would be the case if $\mathcal{I}^\omega$ were of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}'}, \gamma, \pi)$ for some $\mathcal{I}'$. Since, in general, it is not, we need to take one extra step: Let $\mathcal{I}^{\omega+1} = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}^\omega}, \gamma, \pi)$. It turns out that $\mathcal{I}^{\omega+1}$ is the unique interpreted system representing Pg in the interpreted context (γ, π). The actual details of the proof are described in Exercise 7.11.

Notice that our construction is somewhat more complicated than the intuition we gave at the beginning of the section. Our discussion there was in terms of prefixes of runs. The idea was that by inductively assuming that we have defined all prefixes through time m, we could then construct all prefixes through time $m + 1$. The runs of our system would be those all of whose prefixes were constructed. Unfortunately, in our actual proof, we must work directly with systems consisting not of prefixes of runs, but of (full) runs; this is because we have no definition of $\mathsf{Pg}^{Pref_m(\mathcal{I})}$. Hence, in our proof, we used the full system $\mathcal{R}^m$ rather than $Pref_m(\mathcal{R}^m)$. On the other hand, our assumption that (γ, π) provides witnesses for Pg guarantees that all that matters about $\mathcal{R}^m$ is $Pref_m(\mathcal{R}^m)$; the proof shows that we could have replaced $\mathcal{R}^m$ in the construction by any other system $\mathcal{R}'$ such that $Pref_m(\mathcal{R}') = Pref_m(\mathcal{R}^m)$ (see part (b) of Exercise 7.11). ∎

Using Theorem 7.2.4, we can show that many knowledge-based programs of interest have unique representations. For example, since, as we have already mentioned, $(\gamma^{bt}_{fair}, \pi)$ provides witnesses for BT′ and BT″, and γ^{bt}_{fair} is a nonexcluding context, it follows that BT′ and BT″ have unique representations in this context. Another obvious application is the muddy children puzzle.

Example 7.2.5 We now want to take a more careful look at the knowledge-based program MC run by the muddy children. We start by formally describing the context (γ^{mc}, π^{mc}) corresponding to our intuitive description of the muddy children puzzle. The agents here are the children and the father. We can view $\gamma^{mc} = (P^{mc}_e, \mathcal{G}_0, \tau, \mathit{True})$ as a recording message-passing context, in which whatever an

agent (the father or one of the children) says in a given round is a message that is delivered in the same round to all other agents. The initial states of the children and the father describe what they see; later states describe everything they have heard. Thus, $\mathcal{G}_0$ consists of all 2^n tuples of the form $(\langle\,\rangle, X^{-1}, \ldots, X^{-n}, X)$, where $X = (x_1, \ldots, x_n)$ is a tuple of 0's and 1's, with $x_i = 0$ meaning that child i is clean, and $x_i = 1$ meaning that he has a muddy forehead, and $X^{-i} = (x_1, \ldots, x_{i-1}, *, x_{i+1}, \ldots, x_n)$, i.e., it differs from X only in that it contains a $*$ in the i^{th} component. Intuitively, X^{-i} describes what child i sees given that X describes the true situation, where $*$ means "no information." Only the father sees all the children, so his initial local state is X. The initial local state of the environment is the empty history $\langle\,\rangle$. The only actions performed by the children and the father are the sending of messages, and these actions have the obvious results of changing their local states and the local state of the environment. The environment's protocol P_e^{mc} is simply to deliver all messages in the same round in which they are sent.

The children run the knowledge-based programs MC_i described in the beginning of the chapter. The father runs the following (standard) program:

case of
  if *initial* $\wedge \bigvee_{i=1}^{n} p_i$ **do**
    say "At least one of you has mud on your forehead; does any of you know whether you have mud on your own forehead?"
  if *initial* $\wedge \neg \bigvee_{i=1}^{n} p_i$ **do**
    say "Does any of you know whether you have mud on your own forehead?"
  if *childrenanswered* **do**
    say "Does any of you know whether you have mud on your own forehead?"
end case.

Here *initial* is a primitive proposition that is true in the initial state, i.e., before any communication has taken place, and *childrenanswered* is a primitive proposition that is true if the father heard the children's answers in the previous round. Thus, in round 1, if there is at least one muddy child, a message to this effect is sent to all children. In the odd-numbered rounds 1, 3, 5, ..., the father sends to all children the message "Does any of you know whether you have mud on your own forehead?" The children respond "Yes" or "No" in the even-numbered rounds. Finally, the interpretation π^{mc} interprets the propositions p_i, *childheard*$_i$, *initial*, and *childrenanswered* in the obvious way. It is straightforward to see that in the interpreted context (γ^{mc}, π^{mc}), the knowledge-based program MC satisfies the knowledge-based

specification σ^{mc}: "a child says 'Yes' if he knows whether he is muddy, and says 'No' otherwise" (Exercise 7.13).

We now want to apply Theorem 7.2.4 to show that there is a unique interpreted system representing MC. Since the condition in γ^{mc} is *True*, it easily follows that γ^{mc} is nonexcluding. Clearly there are no temporal operators in the tests in MC. Moreover, notice that the father and the children each either send a message or receive one in every round, and they keep track of the messages they send and receive in their local states. As we observed in the discussion following Lemma 7.2.3, it follows that every interpreted system of the form $\mathbf{I}^{rep}(\mathsf{MC}^{\mathcal{I}}, \gamma^{mc}, \pi^{mc})$ is synchronous. Thus, we can apply Lemma 7.2.3 to conclude that (γ^{mc}, π^{mc}) provides witnesses for MC.

The same arguments show that the hypotheses of Theorem 7.2.4 also hold for any subcontext $\gamma' \sqsubseteq \gamma^{mc}$ obtained by restricting the set of initial states, that is, by replacing $\mathcal{G}_0$ by some subset of $\mathcal{G}_0$. Restricting the set of initial states corresponds to changing the puzzle by making certain information common knowledge. For example, eliminating the initial states where child 3's forehead is clean corresponds to making it common knowledge that child 3's forehead is muddy. It can be shown that MC satisfies σ^{mc} in any interpreted context (γ', π^{mc}) where γ' is a subcontext of γ^{mc}. (See Exercise 7.13 for more discussion on the effect of passing to subcontexts.)

Let MC_s be the standard program for the muddy children implicitly described in Chapters 1 and 2. Namely, if the father initially says that at least one child has a muddy forehead, then a child that sees k muddy children responds "No" to the father's first k questions and "Yes" to the $(k+1)^{\text{st}}$ question (and to all the questions after that). Finally, let $\mathcal{I}^{mc} = \mathbf{I}^{rep}(\mathsf{MC}_s, \gamma^{mc}, \pi^{mc})$. It is straightforward to show that $\mathcal{I}^{mc}$ represents MC in (γ^{mc}, π^{mc}) (Exercise 7.13), and hence, by our previous argument, is the unique such interpreted system. In fact, MC_s implements MC in (γ^{mc}, π^{mc}). There are, however, contexts in which MC_s does not implement MC. For example, consider the context where it is common knowledge that the children all have muddy foreheads. This is the subcontext $\gamma' \sqsubseteq \gamma^{mc}$ in which we replace $\mathcal{G}_0$ by the singleton set $\{(\langle\rangle, X^{-1}, \ldots, X^{-n}, X)\}$, where $X = (1, \ldots, 1)$. We leave it to the reader to check that in the unique interpreted system $\mathcal{I}'$ representing MC in (γ', π^{mc}), all the children respond "Yes" to the father's first question. Clearly MC_s does not implement MC in this context. ∎

In the remainder of this chapter, we show the power of programming at the knowledge level, by considering a number of other examples of knowledge-based programs. As we shall see, these programs all have unique representations in the contexts of interest. This will make it easier to implement them by standard programs.

7.3 Knowledge Bases Revisited

We now return to modeling a knowledge base KB, as first discussed in Section 4.4.1. As we observed, there are difficulties in modeling a situation where the Teller gives the KB information that is not purely propositional, and includes information about its knowledge. We now have the tools to model this situation.

We start by describing a context that captures the situation. Let $\gamma^{kb} = (P_e^{kb}, \mathcal{G}_0, \tau, \mathit{True})$ where, as before, we take the KB's local state to consist of the sequence of formulas it has been told and the environment's state to consist of a truth assignment describing the external world. We start by considering the basic situation described in Section 4.4.1, where the Teller is assumed to have complete information about the world, so that the Teller's state consists of the truth assignment that describes the external world and the sequence of formulas the Teller has told the KB. Thus, as in Section 4.4.1, we denote a global state by a tuple of the form $(\alpha, \langle \varphi_1, \ldots, \varphi_k \rangle, \cdot)$, where the φ_i's can be arbitrary formulas about the world and the KB's knowledge. The initial states have the form $(\alpha, \langle\,\rangle, \cdot)$. The Teller's actions are to send messages, which are formulas. We assume that the environment delivers all messages in the same round that they are sent; i.e., using our earlier terminology, the only action of the environment is $\mathsf{deliver}_{KB}(\mathit{current})$, and it performs this action at every round. We leave it to the reader to describe τ and fill in the remaining details of γ^{kb} (Exercise 7.14). As before, we want the interpretation π^{kb} to be such that $\pi^{kb}(\alpha, \ldots) = \alpha$.

The Teller runs a simple knowledge-based program TELL_T. It consists of an infinite number of clauses. For each KB-formula φ (i.e., a formula φ in which the only modal operator is K_{KB}), there is a clause

if $K_T\varphi$ **do** $\mathsf{send}(\varphi)$.

In addition the program has a clause

if *true* **do** Λ.

That is, at each step, either the Teller does nothing, or it nondeterministically chooses some information involving the KB's knowledge and the external world (but not the Teller's knowledge) and sends it to the KB. If, instead of having a clause for each KB-formula, we have a clause for each propositional formula, then we get a program $\mathsf{TELLPROP}_T$. This is essentially the special case that we focused on in Section 4.4.1.

Let TELL (resp., TELLPROP) be the joint knowledge-based program where the Teller runs TELL_T (resp., $\mathsf{TELLPROP}_T$) and the KB does nothing (that is, its program is to perform the action Λ at every step). It should be clear that these programs

precisely capture our intuition regarding what the Teller can do. It can be shown, using Theorem 7.2.4, that there are unique interpreted systems $\mathcal{I}^{tell}$ and $\mathcal{I}^{tellprop}$ representing TELL and TELLPROP in (γ^{kb}, π^{kb}), respectively (Exercise 7.16). It turns out that the interpreted system $\mathcal{I}^{kb}$ (defined in Section 4.4.1), which captures the interaction of the knowledge base with the Teller, is precisely the system $\mathcal{I}^{tellprop}$ (Exercise 7.17). This observation provides support for our intuition that $\mathcal{I}^{tell}$ appropriately captures the situation where the Teller tells the KB formulas that may involve the KB's knowledge, even in the non-propositional case. An important advantage of using a knowledge-based program here is that it allows us to characterize the system $\mathcal{I}^{tell}$ without describing it explicitly.

In Section 4.4.1, we considered a number of variants of the basic situation. These variants are easily modeled in the more general setting we are considering here, where we allow the Teller to give information about the KB's knowledge. For example, suppose we have a default assumption such as "if p is true then the first thing the KB will be told is p." Clearly, this corresponds to an obvious modification of the Teller's program (Exercise 7.18).

As we already observed in Section 4.4.1, once we allow the Teller to tell the KB formulas that refer to the KB's knowledge, we can no longer represent the KB's knowledge by the conjunction of the formulas it has been told. Interestingly, we can still show that there is a single propositional formula that characterizes what the KB knows.

Theorem 7.3.1 *Suppose that* $r_{KB}(m) = \langle \varphi_1, \ldots, \varphi_k \rangle$. *We can effectively find a propositional formula* φ *such that for all propositional formulas* ψ *we have* $(\mathcal{I}^{tell}, r, m) \models K_{KB}\psi$ *if and only if* $\mathcal{M}_n^{rst} \models \varphi \Rightarrow \psi$.

Proof See Exercise 7.19. ∎

Theorem 7.3.1 tells us that there is a single propositional formula that determines how the KB answers propositional queries. As we observed in Section 4.4.1, once we know how the KB answers propositional queries, we can determine how the KB answers arbitrary KB-queries.

In more complicated applications, one cannot divide the world neatly into a KB and a Teller. Rather, one often has many agents, each of which plays both the role of the KB and the Teller. More specifically, suppose we have n agents each of whom makes an initial observation of the external world, and then communicates with the others. We assume that the agents are truthful, but they do not necessarily know or tell the "whole truth." We refer to such agents as *observing agents*.

For example, assume that there are three observing agents, namely Alice, Bob, and Charlie, and assume that the external world can be described by exactly two primitive propositions, p and q. Let the external world be characterized by the truth assignment where p and q are both true. Alice's initial information is that p and q are either both true or both false. Bob, on the other hand, has no nontrivial initial information about nature. Thus, as far as Bob is concerned, any of the four truth assignments are possible. Finally, Charlie's initial information is that p is true, but he has no initial information about q. Alice, Bob, and Charlie now begin sending each other messages. Alice sends Charlie a message saying that she knows $p \Rightarrow q$. Assume that Charlie receives this message; he then combines his initial information with the information he gained from Alice's message, and thereby knows that p and q are both true. Charlie then sends a message to Alice saying that he (Charlie) knows that Bob does not know that q is false. Even though Charlie has not received any messages from Bob, this is a truthful message: Charlie knows that q is true, and so Charlie knows that Bob could not possibly know that q is false.

We can see already in this example that the agents do not necessarily tell everything they know. For example, when Alice says that she knows that p implies q, she is making a true statement, but is not telling everything she knows. This is quite reasonable in practice, of course. Even truthful people hide some private information from others. (On the other hand, as we saw in Section 6.5, telling all you know often does lead to protocols where agents acquire information as quickly as possible.)

We formalize the observing agents setting as a synchronous, recording, message-passing system, where all the agents are following a simple knowledge-based program, much like that of the Teller. At every round, agent i nondeterministically selects, for each agent j, a formula φ_j that i knows to be true, and sends the message "φ_j" to j. Formally, agent i's program consists of all clauses of the form

if $K_i\varphi_1 \wedge \ldots \wedge K_i\varphi_k$ **do** $\mathsf{send}(\varphi_1, j_1); \ldots; \mathsf{send}(\varphi_k, j_k)$,

where, just as with a.m.p. systems, we take $\mathsf{send}(\varphi_l, j_l)$ to be the action sending the message φ_l to agent j_l. We assume that $k \geq 1$, so agent i has to send at least one message to some agent j at each round. We allow the messages φ to be arbitrary formulas in $\mathcal{L}_n(\Phi)$, that is, formulas with modal operators for knowledge, but no temporal operators. Φ here is the set of primitive propositions that describe the external world. Note that the tests in the program guarantee that the agents send only messages that they know to be true. We denote the joint program followed by the observing agents by OA.

We now briefly describe the interpreted context (γ^{oa}, π^{oa}), in which we are interested. The initial local states of agent i are of the form $(\mathcal{T}_1, \langle\rangle)$, where $\mathcal{T}_i$ is

a set of truth assignments over Φ. Intuitively, $\mathcal{T}_i$ describes the states of nature that agent i considers possible as a result of her initial observation. Agent i's local state is a pair $(\mathcal{T}_i, h_i)$, where $\mathcal{T}_i$ is her initial state and h_i describes the rest of her history: what actions she took in each round (and in particular, what messages she sent in each round, and to which agents) and what messages she received in each round (and from which agents she received them). This explains why the empty sequence $\langle\,\rangle$ is part of agent i's initial state.

The environment's state is a pair (α, h), where α is the truth assignment that describes the external world (which we assume does not change over time) and h is the sequence of joint actions performed thus far. We take the set $\mathcal{G}_0$ of initial states to be all the global states of the form $((\alpha, \langle\,\rangle), (\mathcal{T}_1, \langle\,\rangle), \ldots, (\mathcal{T}_n, \langle\,\rangle))$, where $\alpha \in \mathcal{T}_i$ for $i = 1, \ldots, n$. The fact that $\mathcal{T}_i$ includes α says that while the agent's initial observation may not completely specify the exact state of nature, it does give her *correct* information about the true situation; the observing agents do not have false beliefs.

The environment nondeterministically chooses what messages will be delivered at each round. We allow messages to be delayed for an arbitrary length of time, messages to be lost, and messages to be delivered several times. We omit a formal description of the environment's protocol here, but it is much like what we described in the case of a.m.p. systems in Section 5.1 (except that there are no go_i or nogo_i actions, since the agents are enabled in every round). We also omit description of the transition function τ, although again it is much like the transition function in γ^{amp}. A joint action has the obvious effect on a global state, namely, that of extending the histories as appropriate. The actions do not change the truth assignment α in the environment's component, since we assume that the external world does not change during the interaction among the agents. We take the condition on runs to be *True*; that is, we place no restrictions on the allowable runs. Finally, π^{oa} is much like π^{kb}: If $s = ((\alpha, h), (\mathcal{T}_1, h_1), \ldots, (\mathcal{T}_n, h_n))$, then we define $\pi^{oa}(s) = \alpha$.

The assumption that each agent sends at least one message in every round was one we did not make in the case of a single KB. Of course, the message may be simply *true*, so that the agent receiving the message may not learn anything from it. Nevertheless, this assumption is not innocuous. It forces every interpreted system of the form $\mathbf{I}^{rep}(\mathsf{OA}^{\mathcal{I}}, \gamma^{oa}, \pi^{oa})$ to be synchronous. Since there are no temporal operators in the tests in OA, it follows from Lemma 7.2.3 that (γ^{oa}, π^{oa}) provides witnesses for OA. Since γ^{oa} is clearly nonexcluding, we can thus apply Theorem 7.2.4 to conclude that there is a unique system representing OA in context (γ^{oa}, π^{oa}). We call this system (with n observing agents) $\mathcal{I}_n^{oa}(\Phi)$. (As we shall see Chapter 8, Φ plays an important role in $\mathcal{I}_n^{oa}(\Phi)$.) In the case of a single KB, we observed that every system

of the form $\mathbf{I}^{rep}(\mathsf{TELL}^{\mathcal{I}}, \gamma^{kb}, \pi^{kb})$ provides witnesses for TELL, even without the assumption that a message is sent in every round. In contrast, it can be shown that if we modify OA so as to allow a null action, then (γ^{oa}, π^{oa}) does *not* provide witnesses for the resulting knowledge-based program (Exercise 7.20). Similarly, if we allow messages about the future, such as "you will eventually know about p," then again the context (γ^{oa}, π^{oa}) does not provide witnesses for the resulting knowledge-based program.

The system $\mathcal{I}_n^{oa}(\Phi)$ is certainly an idealization of a rather simple scenario of observing agents exchanging messages, but it turns out to involve some interesting subtleties. Although knowledge in $\mathcal{I}_n^{oa}(\Phi)$ satisfies the S5 properties, it has additional properties of interest. This point is discussed in detail in Chapter 8, where we study the evolution of the knowledge in $\mathcal{I}_n^{oa}(\Phi)$.

7.4 A Knowledge-Based Program for SBA

We already saw in Section 6.5 that designing programs for SBA is best done by reasoning in terms of the processes' knowledge. In fact, we even sketched there an outline of a knowledge-based program for SBA. We can now formally define such a program and prove that it really does attain SBA in contexts of interest. We can also use the analysis of Section 6.6 to show how the knowledge-based program can be implemented efficiently. We consider a knowledge-based program SBA that is based on the full-information protocol *FIP* of Section 6.5. We describe process i's program SBA_i in Figure 7.1. As in Section 6.5, we use $decided_i$ as an abbreviation for $decided_i(0) \vee decided_i(1)$.

case of
  if $\neg decided_i \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$
    do $\mathsf{decide}_i(0)$
  if $\neg decided_i \wedge \neg B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$
    do $\mathsf{decide}_i(1)$
  if $\neg decided_i \wedge \neg B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge \neg B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$
    do $\mathsf{sendall}_i(\textit{local state})$
end case

Figure 7.1 The knowledge-based program SBA_i

Recall that the action $\mathsf{sendall}_i(\textit{local state})$ has the effect of sending each process other than i the message ℓ if process i's local state is ℓ. Thus, messages in SBA_i are sent according to *FIP*. Note that since $B_i^{\mathcal{N}}\varphi$ is an abbreviation for $K_i(i \in \mathcal{N} \Rightarrow \varphi)$, tests such as $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ and $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ are indeed knowledge tests.

We argued informally in Section 6.5 that, provided that all nonfaulty processes do eventually decide, then a program of this form satisfies the specification σ^{sba}. The next theorem makes this informal argument precise.

Theorem 7.4.1 *If (γ, π) is a ba-compatible interpreted context, $\mathcal{I}$ is consistent with the program* SBA *in (γ, π), and $C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$ is attained in every run of $\mathcal{I}$, then $\mathcal{I}$ satisfies σ^{sba}. Moreover, the processes decide in a run r of $\mathcal{I}$ at the round following the first time that $C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$ is attained.*

Proof Suppose $\mathcal{I}$ satisfies the hypotheses of the theorem. Fix a run r of $\mathcal{I}$. We want to show that all the properties required by σ^{sba} hold of r. Recall (see Exercise 6.13) that

$$\mathcal{I} \models i \in \mathcal{N} \Rightarrow (C_{\mathcal{N}}\varphi \Leftrightarrow B_i^{\mathcal{N}} C_{\mathcal{N}}\varphi). \qquad (*)$$

Suppose processes i_1 and i_2 are nonfaulty throughout run r. Let $m \geq 0$ be the smallest integer such that $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$. The existence of m is guaranteed by the hypotheses of the theorem. From $(*)$, it follows that if $k < m$, then $(\mathcal{I}, r, k) \models \neg B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge \neg B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ for $j = i_1, i_2$, so neither process decides before round $m + 1$. Notice that we must have either $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists 0)$ or $(\mathcal{I}, r, m) \models \neg C_{\mathcal{N}}(\exists 0) \wedge C_{\mathcal{N}}(\exists 1)$. In the first case, from $(*)$ it follows that $(\mathcal{I}, r, m) \models B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ for $j = i_1, i_2$, so both processes decide 0 at round $m + 1$. In the second case, from $(*)$ it follows that $(\mathcal{I}, r, m) \models \neg B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge B_j^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$ for $j = i_1, i_2$, so both processes decide 1 at round $m + 1$. Clearly if $k > m$, then $(\mathcal{I}, r, k) \models \textit{decided}_j$ for $j = i_1, i_2$, so the program guarantees that neither process decides after round $m + 1$. This gives us the decision, agreement, and simultaneity properties. For validity, suppose that all initial values were 0 (a similar argument holds if all initial values were 1). Since $t < n$ we are guaranteed that $\mathcal{N} \neq \emptyset$ (see Exercise 6.13), so that $\mathcal{I} \models C_{\mathcal{N}}(\exists 1) \Rightarrow \exists 1$. It follows that if all processes start with initial value 0, then $\exists 1$ is false, and 0 is the value the nonfaulty processes decide on. Since the run r was chosen arbitrarily, we thus obtain that all runs of $\mathcal{I}$ satisfy the decision, agreement, validity and simultaneity properties, so that $\mathcal{I}$ satisfies σ^{sba}. ∎

Notice that the fact that SBA is a knowledge-based program makes the proof of Theorem 7.4.1 direct and simple. We now want to use Theorem 7.4.1 to prove that SBA indeed satisfies the specification σ^{sba} in certain contexts. As expected,

the contexts we consider are those in $\Gamma^{sba} = \{\gamma^{cr}, \gamma^{som}, \gamma^{gom}\}$, i.e., the contexts corresponding to crash failures, sending-omission failures, and general-omission failures. Before proceeding with the proof, we need some preliminary observations. The first observation is that, in these contexts, there is a unique interpreted system that represents the program, i.e., the program SBA does completely describe the processes' behavior.

Proposition 7.4.2 *For $\gamma \in \Gamma^{sba}$, there is a unique interpreted system representing* SBA *in (γ, π^{sba}).*

Proof Let $\gamma \in \Gamma^{sba}$. Clearly γ is nonexcluding, since the condition on runs in γ is *True*. It is also easy to show that (γ, π^{sba}) provides witnesses for SBA (Exercise 7.21). The result now follows from Theorem 7.2.4. ∎

Let $\mathcal{I}^{fm}_{sba}$ be the interpreted system representing SBA in (γ^{fm}, π^{sba}), for $fm \in \{cr, gom, som\}$. In analyzing SBA we need to know when basic formulas hold in $\mathcal{I}^{fm}_{sba}$. (Recall that in Section 6.5 we defined a basic formula to be one of the form $\mathcal{K}_i\varphi$, $D_{\mathcal{N}}\varphi$, $C_{\mathcal{N}}\varphi$, or $B_i^{\mathcal{N}}\varphi$, where φ is a formula determined by the initial state.) We note that the interpreted system $\mathcal{I}^{fm}_{sba}$ is closely related to the interpreted system $\mathcal{I}^{fm} = \mathbf{I}^{rep}(FIP, \gamma^{fm}, \pi^{sba})$, which we analyzed in Section 6.6: $\mathcal{I}^{fm}_{sba}$ represents SBA, whereas $\mathcal{I}^{fm}$ represents *FIP*. The difference between these interpreted systems is that in $\mathcal{I}^{fm}$, the only actions are sending the description of local states, whereas in $\mathcal{I}^{fm}_{sba}$ we also have decide_i actions. Therefore, up to the time that the processes decide, we would expect that the same basic formulas are true in $\mathcal{I}^{fm}$ and in $\mathcal{I}^{fm}_{sba}$. The next lemma says that this is indeed the case.

Lemma 7.4.3 *Let φ be a basic formula. For $fm \in \{cr, gom, som\}$, let r be a run in $\mathcal{I}^{fm}$, and let r' be the corresponding run in $\mathcal{I}^{fm}_{sba}$. Let m_r be the least m such that $(\mathcal{I}^{fm}, r, m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$. Then for all $m \leq m_r$, we have $(\mathcal{I}^{fm}, r, m) \models \varphi$ iff $(\mathcal{I}^{fm}_{sba}, r', m) \models \varphi$. In particular, we have that $(\mathcal{I}^{fm}_{sba}, r', m_r) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$.*

Proof See Exercise 7.22. ∎

We can now prove that SBA really does satisfy σ^{sba} in the contexts of interest. Moreover, it leads to an *optimum* protocol. This protocol is $\mathsf{SBA}^{\mathcal{I}^{fm}_{sba}}$, i.e., the protocol obtained from SBA in the interpreted system $\mathcal{I}^{fm}_{sba}$. For ease of notation, we denote this protocol by SBA^{fm}.

Theorem 7.4.4 *For each fm* $\in \{cr, gom, som\}$, *the interpreted system* $\mathcal{I}^{fm}_{sba}$ *satisfies the specification* σ^{sba} *in the context* (γ^{fm}, π^{sba}). *Moreover, the protocol* SBA^{fm} *is an optimum protocol for SBA in the context* γ^{fm}.

Proof Fix $fm \in \{cr, gom, som\}$. By Corollary 6.5.4 we know that $C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$ is attained in every run of $\mathcal{I}^{fm}$. By Lemma 7.4.3 it follows that $C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$ is attained in every run of $\mathcal{I}^{fm}_{sba}$. By Theorem 7.4.1, it follows that $\mathcal{I}^{fm}_{sba}$ satisfies σ^{sba}.

Now let P be a deterministic protocol that satisfies σ^{sba} in (γ^{fm}, π^{sba}). Let r be a run of $\mathcal{I}^{fm}$, let r' be the corresponding run of $\mathcal{I}^{fm}_{sba}$, and let r'' be the corresponding run of $\mathcal{I}' = \mathbf{I}^{rep}(P, \gamma^{fm}, \pi^{sba})$. Suppose that m is the first time such that $(\mathcal{I}^{fm}, r, m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$. By Lemma 7.4.3, m is also the first time such that $(\mathcal{I}^{fm}_{sba}, r', m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$. By Theorem 7.4.1, the nonfaulty processes decide at round $m+1$ of run r'.

Suppose that the nonfaulty processes decide y in round $m'+1$ of run r''. From Corollary 6.4.3, it follows that $(\mathcal{I}', r'', m') \models C_{\mathcal{N}}(\exists y)$. By Theorem 6.5.3, it follows that $(\mathcal{I}^{fm}, r, m') \models C_{\mathcal{N}}(\exists y)$. Thus, we must have $m \leq m'$. It follows that SBA^{fm} dominates P, as desired. ∎

These results show how doing a knowledge-based analysis of a problem can clarify what is going on and help in designing protocols. The key role of common knowledge in SBA was already brought out by our analysis in Chapter 6. Here we exploited these results to design a knowledge-based program that was easily proved correct and led to an optimum protocol for contexts in Γ^{sba}.

This analysis has shown us how to obtain optimum protocols for SBA, but these protocols are not particularly efficient. We already observed in Section 6.6 that if we use *FIP* (or a program based on *FIP*, as SBA is), then the messages, and hence processes' local states, grow exponentially large. In Section 6.6, we also considered the protocol *FIP'*, which still transmits all the information that *FIP* does, but does so using much shorter messages, by sending a compact representation of the local state. Let SBA' be the knowledge-based program that is based on *FIP'* in the same way that SBA is based on *FIP*. To be precise, SBA' is the same as SBA, except that the action sendall(*local state*) is replaced by sendall(G(*local state*)), which has the effect of sending each process other than i the message $G(\ell)$ if process i's local state is ℓ. (Recall from Section 6.6 that $G(\ell)$ is a compact representation of the local state ℓ.)

It is easy to see that we get the following analogue to Proposition 7.4.2.

Proposition 7.4.5 *For $\gamma \in \Gamma^{sba}$, there is a unique interpreted system representing* SBA′ *in* (γ, π^{sba}).

Proof See Exercise 7.23. ∎

Let $\mathcal{I}^{fm}_{sba'}$ be the unique interpreted system representing SBA′ in (γ^{fm}, π^{sba}), for $fm \in \{cr, gom, som\}$. Using Theorem 6.6.8, we immediately get the following analogue to Theorem 7.4.4. Here we use $(\mathsf{SBA}')^{fm}$ to denote the protocol $(\mathsf{SBA}')^{\mathcal{I}^{fm}_{sba'}}$, i.e., the protocol obtained from SBA′ in the interpreted system $\mathcal{I}^{fm}_{sba'}$.

Theorem 7.4.6 *For each $fm \in \{cr, gom, som\}$, the interpreted system $\mathcal{I}^{fm}_{sba'}$ satisfies the specification σ^{sba} in the context (γ^{fm}, π^{sba}). Moreover, the protocol* $(\mathsf{SBA}')^{fm}$ *is an optimum protocol for SBA in the context γ^{fm}.*

Proof See Exercise 7.24. ∎

In Section 6.6 we showed that there are polynomial-time algorithms to compute whether $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$ holds at a given point in the systems $\mathcal{I}^{cr}$ and $\mathcal{I}^{som}$ (see Theorems 6.6.9 and 6.6.10). This means that we can replace the tests for knowledge in SBA′ by polynomial-time standard tests. Thus we get:

Theorem 7.4.7 *There are polynomial-time optimum protocols for SBA in the contexts (γ^{cr}, π^{sba}) and $(\gamma^{som}, \pi^{sba})$.*

What about general-omission failures? According to Theorem 6.6.10, in the interpreted system $\mathcal{I}^{gom}$, the problem of computing when $C_{\mathcal{N}}(\exists y)$ holds is NP-hard. This does not immediately imply that there cannot be an optimum polynomial-time protocol for SBA in the context $(\gamma^{gom}, \pi^{sba})$. It is possible that an optimum protocol for SBA does not compute when $C_{\mathcal{N}}(\exists y)$ holds. Nevertheless, it can be shown that from an optimum polynomial-time protocol for SBA in γ^{gom} we can obtain a polynomial-time algorithm that tests for $C_{\mathcal{N}}(\exists y)$ in $\mathcal{I}^{gom}$. As a result, we get

Theorem 7.4.8 *If $P \neq NP$, then no polynomial-time protocol can be optimum for SBA in $(\gamma^{gom}, \pi^{sba})$.*

7.5 Strong Correctness

As we observed in Section 5.4, we are often interested in the correctness of protocols and programs with respect to all subcontexts of some context γ. Thus, we

defined the notion of strong correctness (for standard programs) as follows: a standard program Pg strongly satisfies a specification σ or is strongly correct with respect to σ in (γ, π) if every interpreted system consistent with Pg in (γ, π) satisfies σ. By Lemma 5.2.2, this means that every interpreted system that represents Pg in a subcontext $\gamma' \sqsubseteq \gamma$ satisfies σ.

In the case of a standard program Pg, there is a well-defined relationship between the (unique) system that represents Pg in a given interpreted context (γ, π), and the systems consistent with Pg in (γ, π): a system consistent with Pg must be a subset of the system that represents Pg, because the system that represents Pg consists of *all* runs that are consistent with Pg in (γ, π). As we shall see in Example 7.5.2, no such relationship exists in the case of knowledge-based programs, even in situations where there is a unique interpreted system representing the program. In fact, there is no simple relationship between the systems that are consistent with Pg and the systems that represent Pg in (γ, π), except for the fact that a system that represents Pg in (γ, π) is of course also consistent with Pg in (γ, π). Nevertheless, the connection described in Lemma 5.2.2 between the systems that are consistent with Pg in (γ, π) and the systems that represent Pg in subcontexts of (γ, π) still holds:

Lemma 7.5.1 *$\mathcal{I}$ is consistent with the knowledge-based program* Pg *in the interpreted context (γ, π) if and only if $\mathcal{I}$ represents* Pg *in some subcontext $\gamma' \sqsubseteq \gamma$.*

Proof See Exercise 7.25. ∎

In analogy to our definitions with standard programs, we say that Pg *strongly satisfies* σ or is *strongly correct* with respect to σ in the interpreted context (γ, π) if every interpreted system consistent with Pg in (γ, π) satisfies σ. Our motivation for considering strong correctness here is the same as it was in the case of standard programs: By Lemma 7.5.1, it follows that by proving strong correctness with respect to an interpreted context we prove correctness with respect to all subcontexts, as we did, for example, in Example 7.2.5 for the knowledge-based program MC.

In the case of standard programs, we observed that correctness and strong correctness coincide for run-based specifications. This depends crucially on the fact that a system consistent with a protocol is a subset of the unique system representing the protocol. As we have already mentioned, the analogous property does not hold for knowledge-based programs. As Example 7.5.2 will show, even if we consider only run-based specifications, correctness and strong correctness do not coincide for knowledge-based programs.

Example 7.5.2 Let us return to the bit-transmission problem for the last time. Recall from Example 7.1.2 that we are considering the standard program BT, the knowledge-based programs BT′ and BT″, and the specifications σ' and σ''. We already observed that BT, BT′ and BT″ satisfy both σ' and σ'' in the interpreted context $(\gamma^{bt}_{fair}, \pi^{bt})$.

Since σ' is a run-based specification and BT is a standard program, BT *strongly* satisfies σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$. Indeed, satisfaction and strong satisfaction coincide in the case of run-based specifications and standard programs (see Exercise 5.6). This is not so for knowledge-based programs. It can be shown that BT′ also strongly satisfies σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$ (Exercise 7.26), but BT″ does not. This is a significant difference between standard and knowledge-based programs. To see that BT″ does not strongly satisfy σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$, consider again the context $\gamma^{bt}_{ck} \sqsubseteq \gamma^{bt}_{fair}$ defined in Section 5.4, where it is common knowledge that S's initial value is 1. Let r^{ck} be the run starting with initial state $(\langle\,\rangle, 1, \lambda)$ (the only one possible in γ^{bt}_{ck}) in which neither S nor R send any messages. Let $\mathcal{R}^{ck} = \{r^{ck}\}$ and let $\mathcal{I}^{ck} = (\mathcal{R}^{ck}, \pi^{bt})$. It is easy to see that $\mathcal{I}^{ck}$ is the unique system representing BT″ in the interpreted context $(\gamma^{bt}_{ck}, \pi^{bt})$. Now $\mathcal{I}^{ck}$ does not satisfy σ', because no messages are sent in r^{ck}. It follows that BT″ does not strongly satisfy σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$. Notice that r^{ck} is not in $\mathbf{I}^{rep}(\mathsf{BT}'', \gamma^{bt}_{fair}, \pi^{bt}) = \mathbf{I}^{rep}(\mathsf{BT}, \gamma^{bt}_{fair}, \pi^{bt})$. This shows that by passing from γ^{bt}_{fair} to its subcontext γ^{bt}_{ck}, we get a system that is not a subset of $\mathcal{R}^{fair}$. By contrast, with a standard program, we always obtain a subset of the runs when we pass to a subcontext.

What about the knowledge-based specification σ''? We already observed in Example 5.4.1 that BT does not strongly satisfy σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$, because it does not satisfy σ'' in the subcontext $(\gamma^{bt}_{ck}, \pi^{bt})$. Similarly, BT′ does not strongly satisfy σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$ either. To see this, consider the context γ^{bt}_{ck} again. In this context, according to BT′, S still starts by sending R messages, since it must send messages until it knows that R received the bit. But S knows that R knows (in γ^{bt}_{ck}) that the value of the initial bit is 1 even before R receives a message from S, since this fact is common knowledge. Thus, S should not send any messages according to σ''. The program BT″, however, does satisfy σ'' in $(\gamma^{bt}_{ck}, \pi^{bt})$. In fact, BT″ strongly satisfies σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$ (Exercise 7.26). ∎

7.6 The Sequence-Transmission Problem

We now focus on an extension of the bit-transmission problem called the *sequence-transmission problem*, and we show how a knowledge-based analysis can help to clarify many issues here as well.

The sequence-transmission problem is a standard problem of data communication. Just as in the bit-transmission problem, we have a *sender* S and a *receiver* R. The sender S has an input tape with an infinite sequence $X = \langle x_0, x_1, \ldots \rangle$ of data elements. S reads these data elements and tries to transmit them to R. R must write these data elements onto an output tape. We would like a solution that satisfies (a) at any time the sequence of data elements written by R is a prefix of X and (b) every data element x_i in the sequence X is eventually written by R (or more precisely, every finite prefix of X is eventually a prefix of the data elements written by R). In analogy to our terminology in Example 7.2.2, property (a) is called the safety property, while (b) is called the liveness property. Note that both safety and liveness are run-based specifications.

The sequence-transmission problem clearly has a trivial solution if we assume that messages sent by S cannot be lost, corrupted, duplicated, or reordered. S simply sends $x_0, x_1, \ldots$ in order, and R writes them out as it receives them. Once we consider a faulty communication medium, however, the problem becomes far more complicated. The sequence-transmission problem over a faulty communication medium has been extensively studied in the literature, and solutions for several communication models have been proposed. (We mention some of these solutions in the notes at the end of the chapter.) These solutions were all designed individually, on an *ad hoc* basis. Instead of giving individual solutions for a number of different communication models, we describe here one high-level knowledge-based solution. Solutions for various communication models can be derived from our solution. Thus, the knowledge-based approach enables us to design programs at a higher level of abstraction.

Generalizing our ideas about the bit-transmission problem, there is a very simple knowledge-based solution to the sequence-transmission problem. S reads the i^{th} data element, and repeatedly sends it to R until S knows that R has received it and that R knows that it is the i^{th} element. At that point, S reads the $(i+1)^{\text{st}}$ element, and so on. R writes the data elements as it learns about them, and it then requests S to send the next data element. A knowledge-based program ST that captures this intuition is described in Figure 7.2. For simplicity, we assume here that the input sequence consists only of 0's and 1's, although we could easily deal with any finite data domain.

ST_S:

case of
  if $\neg K_S K_R(x_{@i})$ **do** send($\langle i, y \rangle$)
  if $K_S K_R(x_{@i})$ **do** $i := i + 1$; read
end case

ST_R:

case of
  if $\neg K_R(x_j)$ **do** send(j)
  if $K_R(x_j = 0)$ **do** write(0); $j := j + 1$
  if $K_R(x_j = 1)$ **do** write(1); $j := j + 1$
end case

Figure 7.2 The knowledge-based program ST

A few words are now in order on the notation we use in the program. The variable y refers to the most recent data element read and is initialized with the first data element; i is a counter, initialized to 1, that keeps track of how many data elements have been read by S; and j is a counter, initialized to 0, that keeps track of the number of data elements written by R. The effect of the "read" action in S's program is to read the value of the current data element (where "current" is determined by the counter i). The effect of the "write(0)" and "write(1)" actions in R's program is to write an output value. As in message-passing systems, neither S nor R takes a receive action; we leave message delivery to the environment's protocol.

Generalizing what we did in the knowledge-based program for the bit transmission problem (Example 7.1.1), we take $K_R(x_i)$ to be an abbreviation for $K_R(x_i = 0) \vee K_R(x_i = 1)$. Thus, if $K_R(x_i)$ holds, then R knows the value of x_i. Recall that S sends the i^{th} data element until it knows that R knows the value of this data element. Intuitively, this suggests that S should test if $K_S K_R(x_i)$ holds; if it does not, then S should continue to send the i^{th} data element; otherwise S can increment i and read the next data element. There is, however, a subtle problem with this intuition, caused by the fact that i is a variable local to S, and, consequently, its value may not be known to R. Roughly speaking, the problem arises because we cannot substitute equals for equals inside the scope of a K operator. For example, suppose we are at a point where $i = 3$. What S really wants to do is to continue sending the value of x_3 until $K_S K_R(x_3)$ holds. This is not the same as sending it the value of x_3 until $K_S K_R(x_i)$ holds. Put another way, $(i = 3) \wedge K_S K_R(x_i)$ is not equivalent to $(i = 3) \wedge K_S K_R(x_3)$. The problem is that R may know the value

of x_3 without knowing the value of x_i (or, for that matter, without even knowing the value of i), since the variable i may take on different values in the global states that R considers possible. In the terminology of Section 3.7.4, the number 3 is a *rigid designator*, while i is not. In a state where $i = 3$, we want S to continue sending x_i until $K_S K_R(x_3)$ holds, not until $K_S K_R(x_i)$ holds. To achieve this effect, we take the clause "**if** $K_S K_R(x_{@i})$ **do** ..." to be an abbreviation for the infinite collection of clauses "**if** $i = k \wedge K_S K_R(x_k)$ **do** ...," for $k = 0, 1, \cdots$. Similarly, we view "**if** $\neg K_S K_R(x_{@i})$ **do** ..." as an abbreviation for the infinite collection of clauses "**if** $i = k \wedge \neg K_S K_R(x_k)$ **do** ...." As we saw in Section 3.7.4, the semantics of first-order modal logic forces free variables to act as rigid designators. Thus, if we had allowed first-order tests in knowledge-based programs, we could have replaced the test $K_S K_R(x_{@i})$ by the first-order test $\forall k(k = i \Rightarrow K_S K_R(x_k))$, and achieved the same effect (see Exercise 7.27). (We could similarly have used $K_R(x_{@j})$ instead of $K_R(x_j)$ in R's program ST_R, but there is no need. Since j is part of R's local state, it is easy to see that $K_R(x_{@j})$ and $K_R(x_j)$ are equivalent.)

It should now be clear—at least at an intuitive level—that the knowledge-based program ST does what we want: S sends $\langle i, y \rangle$, where y is the i^{th} bit, as long as S does not know that R knows the i^{th} bit. R's **send**(j) can be interpreted both as an acknowledgment for receiving the $(j-1)^{\text{st}}$ bit and as a request for the j^{th} bit. Given this intuition, we would expect that ST solves the sequence-transmission problem in a wide variety of contexts. This is what we now want to prove. In particular, we want to show that ST is correct if messages can be deleted, duplicated, reordered, or detectably corrupted. We also want to prove that it is correct if there is some common knowledge about the sequence of data elements. To do this, we prove that ST is strongly correct in a very general context. All the other contexts of interest will be subcontexts of this general context. The interpreted context that we study is (γ^{st}, π^{st}), where $\gamma^{st} = (P_e^{st}, \mathcal{G}_0, \tau, Fair^{st})$. We now describe each of the components in more detail.

We first consider the local states for S and R. It should come as no surprise at this point that there is no unique way to represent these local states. Not all choices are equally appropriate; indeed, the correctness of ST depends crucially on some of the choices we make. Many of our choices express assumptions that are implicit in the text of the program. For example, we assumed that S keeps track of the values it has read and R keeps track of the values it has written. To see where we use this, suppose R sends S the message $k+1$ at the point (r, m) and S receives this message at some later point (r, m'). As we show in the next section (Lemma 7.7.2), R knows the value of x_k when it sends this message; i.e., $K_R(x_k)$ holds at (r, m). To prove that the program is correct, we need to show that $K_S K_R(x_k)$ holds when S receives

the message. If, however, R "forgets" the value of x_k earlier, then $K_R(x_k)$ would no longer hold at (r, m'), and hence neither would $K_S K_R(x_k)$. By having R keep track of all the values it has written, we assure that such forgetting does not occur.

Motivated by this discussion, we choose to model the sender and the receiver as processes with perfect recall. As with message-passing contexts (see Section 5.2), we assume that the local states of the processes consist of their histories. Recall that the history of a process is a sequence whose first element is an initial state and whose later elements are sets of messages (sent and delivered) and internal actions. Here we take the history to include also the sequence of input/output values. Thus, the sender's history includes also the sequence of values read, and the receiver's history includes also the sequence of values written. It is convenient to view a local state of each process as a pair (h_1, h_2), where h_1 is the history of messages and internal actions and h_2 is the input/output history. Thus, y refers to the last element of h_2 in S's state, i refers to the length of h_2 is S's state, and j refers to the length of h_2 in R's state.

What about the environment's states? As in Example 5.2.1, we assume that the context is recording. We take the environment's state to consist of the input sequence and the sequence of joint actions that has been performed thus far. Thus, L_e, the set of possible states for the environment, consists of states of the form (X, h_e), where h_e is a sequence of joint actions. We describe the form of these joint actions later on.

As we mentioned earlier, we assume that the initial state of S includes the first input value. Thus, we take $\mathcal{G}_0$, the set of initial global states of ST, to consist of all global states of the form (s_e, s_S, s_R), where s_e is of the form $(X, \langle\rangle)$, s_S is of the form $(\langle\rangle, \langle x_0\rangle)$, and s_R is of the form $(\langle\rangle, \langle\rangle)$, where X is an infinite sequence of 0's and 1's. Thus, in an initial state, the sender S starts out reading the first element of the sequence X, the receiver R has not written any elements, and neither S nor R has sent or received any messages.

To complete the description of the environment's state (and to describe the environment's protocol), we need to describe the possible joint actions. The set of actions for S and R is immediate from the knowledge-based program ST. S can perform two actions: send$(\langle i, y\rangle)$ and $i := i + 1$; read. Similarly, R can perform three actions: write(0); $j := j + 1$, write(1); $j := j + 1$, and send(j). The actions we allow the environment to perform are similar in spirit to those that the environment performs in a.m.p. systems (see Example 5.1.2). Essentially, the environment decides when to enable each of S and R and when messages sent by S and R will be delivered. In particular, some messages may never be delivered, and messages may not arrive in the order sent. Also, the environment can duplicate messages, so that the same message may be delivered several times, and the environment can corrupt messages

in a detectable way. We capture these conditions by taking the environment's action to have the form $(\mathsf{a}_{eS}, \mathsf{a}_{eR})$, where

- a_{eS} has the form nogo_S, go_S, or $\mathsf{deliver}_S(\mu_S)$ for $\mu_S \in \{*, \lambda, \mathit{current}\} \cup \{0, 1, 2, \ldots\}$, and
- a_{eR} has the form nogo_R, go_R, or $\mathsf{deliver}_R(\mu_R)$ for $\mu_R \in \{*, \lambda, \mathit{current}\} \cup \{\langle i, y\rangle \mid i \in \{0, 1, 2, \ldots\}\, y \in \{0, 1\}\}$.

Thus, each process can be disabled by the environment's nogo action. If $\mathsf{a}_{eS} = \mathsf{nogo}_S$, then S's action is disabled. If, on the other hand, $\mathsf{a}_{eS} = \mathsf{go}_S$, then S's action is enabled. Finally, if $\mathsf{a}_{eS} = \mathsf{deliver}_S(\mu_S)$, then S's action is enabled and it also receives the message μ_S if $\mu_s \neq \mathit{current}$. (Thus, unlike in a.m.p. systems, a process can perform an action and receive a message in the same round.) If $\mathsf{a}_{eS} = \mathsf{deliver}_S(\mathit{current})$, then S receives the message that R is currently sending, provided that R is currently sending a message and that R's action is enabled, i.e., $\mathsf{a}_{eR} \neq \mathsf{nogo}_R$. We view λ as the special "empty" message. It models a detectable message "nondelivery." This enables us to model situations in which a process tries but fails to receive a message. This is different from nondelivery in which a process simply does not receive a message. For example, if $\mathsf{a}_{eS} = \mathsf{go}_S$, then S does not receive a message. The message $*$, on the other hand, is the special "detectably corrupted" message; this is how we model message corruption. The effect of a_{eR} is similar to a_{eS}; we omit details here. Notice that our model allows the environment to deliver only one message at a time. At the price of slightly complicating the model, we could easily allow the delivery of sets of messages (see Exercise 7.28).

We now have to define the transition function τ. As in Example 5.1.2, the definition is completely straightforward, although tedious to write down. For example, consider the joint action $(\mathsf{a}_e, \mathsf{a}_S, \mathsf{a}_R)$, where $\mathsf{a}_e = (\mathsf{deliver}_S(*), \mathsf{nogo}_R)$, $\mathsf{a}_S = \mathsf{send}(\langle i, y\rangle)$, and $\mathsf{a}_R = \mathsf{send}(j)$. The effect of this joint action is that the tuple $(\mathsf{a}_e, \mathsf{a}_S, \mathsf{a}_R)$ is appended to the environment's history, and $\mathit{receive}(*)$ is appended to the sender's history. Note that $\mathit{send}(j)$ is not appended to the receiver's history, because of the nogo_R action by the environment. We leave further details to the reader (see Exercise 7.29).

The environment's protocol P_e^{st} is straightforward: the environment nondeterministically chooses some action a_e at every state, with the only constraint being that it does not deliver a message that was not sent earlier by an enabled process. Note that we allow the same message to be delivered several times.

Finally, we take Fair^{st} to be the set of runs where both S and R are scheduled infinitely often, and message delivery is fair, in that a message sent infinitely often is eventually delivered.

Next, we need to define an interpretation π^{st} for the propositions $x_j = 0$, $x_j = 1$, $i = 0$, $i = 1$, $i = 2, \ldots$. (Recall that propositions of the form $i = k$ arise when we replace $K_R(x_{@i})$ by the infinite set of clauses that it abbreviates.) We give these propositions the obvious interpretation, where x_j refers to the j^{th} element of X and i refers to the number of data elements read by S. This completes the description of the interpreted context (γ^{st}, π^{st}).

We leave it to the reader to check that by appropriately restricting the environment's protocol γ^{st}, we can construct contexts where message delivery is guaranteed, messages are not corrupted, messages are received in the order in which they are sent, and so on. Similarly, by appropriately restricting the set of initial states, we can capture the fact that, for example, the value of the first data element is common knowledge. These are all subcontexts of γ^{st}.

Recall that a run for the sequence-transmission problem satisfies the safety property if the sequence of elements written is always a prefix of the input sequence, and it satisfies the liveness property if every element in the input sequence is eventually written. Consider the run-based specification σ^{st} consisting of those interpreted systems all of whose runs satisfy both the safety and the liveness properties. We want to show that ST satisfies σ^{st} in the context (γ^{st}, π^{st}). But, as we said before, we actually would like to show more. We want to show that ST still satisfies σ^{st} if we pass to subcontexts. All of this is a consequence of the following result:

Theorem 7.6.1 *The program* ST *strongly satisfies* σ^{st} *in the interpreted context* (γ^{st}, π^{st}).

We defer the proof of Theorem 7.6.1 to Section 7.7, and continue here with our discussion of ST. It is not hard to show that (γ^{st}, π^{st}) provides witnesses for the knowledge tests in ST, and that γ^{st} is nonexcluding (Exercise 7.30). Indeed, this is also true if we restrict the environment's protocol as previously described, or if we restrict attention to a subset of initial states. By Theorem 7.2.4, this implies that there exists a unique system representing ST in all these contexts.

We next consider a standard program ST′ that implements ST in (γ^{st}, π^{st}). ST′ is described in Figure 7.3. As the reader can see, the programs ST and ST′ are very similar syntactically. In fact, the only difference is that knowledge tests in ST are replaced by standard tests in ST′. The standard tests in ST′ use the variables z and z'. The variable z refers to the last message received by S, and the variable z' refers to the last message received by R. The function $proj_i$ returns the i^{th} component of its argument. As we prove in the next section, in the case of the program ST, the following are equivalent: (a) S knows that R knows the value of x_i, and (b) $z = i + 1$. Thus, we replace the test $\neg K_S K_R(x_{@i})$ in ST by $z \neq i + 1$ in ST′. Similarly, $K_R(x_j)$

holds if and only if R receives a message of the form $\langle j, y \rangle$, i.e., when $proj_1(z') = j$. In this case, R writes $proj_2(z')$ on the output tape.

S's program (ST'_S):

case of
  if $z \neq i + 1$ **do** send($\langle i, y \rangle$)
  if $z = i + 1$ **do** $i := i + 1$; read
end case

R's program (ST'_R):

case of
  if $proj_1(z') \neq j$ **do** send(j)
  if $proj_1(z') = j \wedge proj_2(z') = 0$ **do** write(0); $j := j + 1$
  if $proj_1(z') = j \wedge proj_2(z') = 1$ **do** write(1); $j := j + 1$
end case

Figure 7.3 The standard program ST'

We extend π^{st} to interpret the propositions "$z = i + 1$," "$proj_1(z') = j$," "$proj_2(z') = 0$," and "$proj_2(z') = 1$" in the obvious way. It is not hard to show that ST' implements ST in (γ^{st}, π^{st}) (Exercise 7.31). Notice that since σ^{st} is a run-based specification and ST' satisfies σ^{st} in (γ^{st}, π^{st}), it follows that ST' strongly satisfies σ^{st} in (γ^{st}, π^{st}). Thus, for every subcontext $\gamma \sqsubseteq \gamma^{st}$ we have that ST' satisfies σ^{st} in (γ', π^{st}).

ST' implements ST not only in (γ^{st}, π^{st}), but also in a number of other contexts of interest. For example, if we restrict the environment's protocol so that S and R are always scheduled, or so that there is no message corruption, or that messages are always delivered in the order in which they are sent, then ST' still can be shown to implement ST (Exercise 7.31).

On the other hand, there are contexts where ST' does *not* implement ST. Roughly speaking, ST' cannot take advantage of some information regarding message delivery or common knowledge about the data sequence. For example, suppose it is common knowledge that the initial value in the sequence is 1. Running ST_S, the sender will not bother sending x_0 since $K_S K_R(x_0)$ will hold. On the other hand, the sender does not take advantage of such knowledge when running ST'; the sender S sends R every value that it reads, even if R already knows that value. Formally, if γ^1 is the context that results by replacing $\mathcal{G}_0$ in γ^{st} by that subset consisting of the initial states where the first element in the data sequence is 1, it is easy to see

that $\mathbf{I}^{rep}(\mathsf{ST}', \gamma^1, \pi^{st}) \neq \mathbf{I}^{rep}(\mathsf{ST}, \gamma^1, \pi^{st})$. Similarly, it can be shown that ST′ does not implement ST in a context where all messages are guaranteed to be delivered in precisely five rounds (Exercise 7.31).

7.7 Proving Strong Correctness of ST

In this section, we provide a formal proof of Theorem 7.6.1. This section can be skipped on a first reading of the book; none of the results will be used in later sections.

We want to show that ST strongly satisfies the specification σ^{st} in the interpreted context (γ^{st}, π^{st}). (Recall that σ^{st} is the run-based specification consisting of those interpreted systems all of whose runs satisfy both the safety and the liveness properties.) By definition, this means we must show that every interpreted system consistent with ST satisfies σ^{st} in this interpreted context. Fix a system $\mathcal{I} = (\mathcal{R}, \pi^{st})$ that is consistent with ST in the interpreted context (γ^{st}, π^{st}). With the help of a few lemmas, we prove that $\mathcal{I}$ satisfies σ^{st}. Before we prove these lemmas, we need to introduce some notation. Assume that $r \in \mathcal{R}$. For every $k \geq 0$ and every variable v of the programs ST and ST′, we let $v^{r(k)}$ denote the value of v at the global state $r(k)$. For example, $i^{r(k)}$ denotes the value of i at $r(k)$. Also, we take $b_S^{r(k)}$ to consist of the messages sent by S up to the (r, k), we take $b_R^{r(k)}$ to consist of the messages sent by R up to this point, we take $Y^{r(k)}$ to be the sequence of values written by R up to this point, and we take X^r to be the input sequence in the run r. Finally, if Z and Z' are two sequences, we write $Z \preceq Z'$ if Z is a prefix (not necessarily strict) of Z'.

Intuitively, safety for ST is obvious, since R writes a data element only if R knows its value. This intuition is formalized in the following lemma.

Lemma 7.7.1 *For all runs $r \in \mathcal{R}$ and all times $m \geq 0$, $|Y^{r(m)}| = j^{r(m)}$ and $Y^{r(m)} \preceq X^r$.*

Proof Let $r \in \mathcal{R}$. We proceed by induction on m. For $m = 0$, the claim follows from our characterization of $\mathcal{G}_0$, the set of initial global states. To see that for every $m > 0$ we have $|Y^{r(m)}| = j^{r(m)}$, note that from the semantics of ST, it is immediate that $|Y|$ increases by one if and only if j does. For the inductive step of the second part, assume that the claim is established for every $l < m$. If $j^{r(m)} = j^{r(m-1)}$, then $Y^{r(m)} = Y^{r(m-1)}$, so that the claim trivially follows from the induction hypothesis. If $j^{r(m)} \neq j^{r(m-1)}$, then $(\mathcal{I}, r, m-1) \models K_R(x_l)$, where $l = j^{r(m-1)}$ and $j^{r(m)} = j^{r(m-1)} + 1$. If $x_l = 0$, then $(\mathcal{I}, r, m-1) \models K_R(x_l = 0)$, so that $Y^{r(m)} = Y^{r(m-1)} \cdot 0$. (As before, $Y^{r(m-1)} \cdot 0$ denotes the result of appending 0

to the sequence $Y^{r(m-1)}$.) Similarly, if $x_l = 1$, then $(\mathcal{I}, r, m-1) \models K_r(x_l = 1)$, so that $Y^{r(m)} = Y^{r(m-1)} \cdot 1$. In either case, it is immediate that the claim holds. ∎

To prove liveness, we need two preliminary lemmas.

Lemma 7.7.2 *For every $l \geq 0$ and every $d \in \{0, 1\}$, the following all hold:*

(a) If $\langle l, d\rangle$ is in $b_S^{r(m)}$ then $(\mathcal{I}, r, m) \models (x_l = d)$.

(b) if l is in $b_R^{r(m)}$ and $l \geq 1$, then $(\mathcal{I}, r, m) \models \bigwedge_{k=0}^{l-1} K_R(x_k)$.

(c) If $(z')^{r(m)} = \langle l, d\rangle$, then $(\mathcal{I}, r, m) \models K_R(x_l = d)$.

(d) if $z^{r(m)} = l$ and $l \geq 1$, then $(\mathcal{I}, r, m) \models \bigwedge_{k=0}^{l-1} K_S K_R(x_k)$.

Proof

(a) Suppose $\langle l, d\rangle$ is in $b_S^{r(m)}$ and let $k \leq m$ be the least integer such that $\langle l, d\rangle$ is in $b_S^{r(k)}$. From the semantics of ST it follows that the message was sent by S at the point $(r, k-1)$; moreover, $i^{r(k-1)} = l$ and $d = x_l$. Hence, $(\mathcal{I}, r, k-1) \models (x_l = d)$. The definition of π^{st} guarantees that $(\mathcal{I}, r, m) \models (x_l = d)$.

(b) Suppose l is in $b_R^{r(m)}$, $l \geq 1$, and $(r, m) \sim_R (r', m')$. We want to show that the values of $x_0, \ldots, x_{l-1}$ are the same in both $r(m)$ and $r'(m')$. Note that from the semantics of ST it follows that there exists $m'' \leq m$ such that l was sent by R at the point (r, m'') and $j^{r(m'')} = l$. By Lemma 7.7.1, we have that $|Y^{r(m'')}| = l$ and $Y^{r(m'')} \preceq X^r$. We clearly have $Y^{r(m'')} \preceq Y^{r(m)}$, so $|Y^{r(m)}| \geq l$. Since R records the sequence of values it has written in its local state, we must have $Y^{r'(m')} = Y^{r(m)}$. By Lemma 7.7.1 again, $Y^{r'(m')} \preceq X^{r'}$ and $Y^{r(m)} \preceq X^r$. It follows that the values of $x_0, \ldots, x_{l-1}$ are the same in both $r(m)$ and $r'(m')$. Thus, $(\mathcal{I}, r, m) \models \bigwedge_{k=0}^{l-1} K_R(x_k)$.

(c) Suppose that $(z')^{r(m)} = \langle l, d\rangle$ and that $(r', m') \sim_R (r, m)$. Then $(z')^{r'(m')} = \langle l, d\rangle$. Thus, we must have that $\langle l, d\rangle$ is in $b_S^{r'(m')}$, since $\langle l, d\rangle$ must have been sent at some time $m'' \leq m'$ in run r'. From part (a), it follows that $(\mathcal{I}, r', m') \models (x_l = d)$. Thus $(\mathcal{I}, r, m) \models K_R(x_l = d)$.

(d) This proof follows the same lines as that for part (c), using part (b) instead of part (a); we leave details to the reader (Exercise 7.32). ∎

Lemma 7.7.3 *For all runs r of $\mathcal{I}$ and all $k \geq 0$, there exists $m_k \geq 0$ such that $j^{r(m_k)} = k$.*

Proof We prove the lemma by induction on k. For the base case we can take $m_0 = 0$; the result follows, since $j^{r(0)} = 0$ by our assumptions about the initial states. For the inductive case, assume that for some $m_k \geq 0$, we have $j^{r(m_k)} = k$. We want to show that there exists $m > m_k$ such that $j^{r(m)} = k + 1$. Assume, by way of contradiction, that for all $m \geq m_k$, we have $j^{r(m)} = k$. The semantics of ST implies that, for all $m \geq m_k$, we must have $(\mathcal{I}, r, m) \models \neg K_R(x_k)$. Since all runs of $\mathcal{I}$ are fair, R sends infinitely many k messages to S in r, and these are received at infinitely many points by S. From Lemma 7.7.2 we get that if $z^{r(m)} = k$ and $k \geq 1$, then $(\mathcal{I}, r, m) \models \bigwedge_{l=0}^{k-1} K_S K_R(x_l)$. It is clear from the semantics of ST that $i^{r(0)} = 0$, and i increases by at most 1 at each round, and never decreases. We claim that at some time n' we must have $i^{r(n')} = k$. To see this, note that at every step, S evaluates the test $\neg K_S K_R(x_{@i})$. By our earlier remarks, this formula will be false at every point (r, n) such that $z^{r(n)} = k$ and $i^{r(n)} < k$. Moreover, each time it is false, the variable i is increased by 1. Since $z^{r(n)} = k$ infinitely often, there must indeed be some point (r, n') such that $i^{r(n')} = k$. Since $(\mathcal{I}, r, m) \models \neg K_R(x_k)$ for all $m \geq m_k$, it follows from the Knowledge Axiom that $(\mathcal{I}, r, m) \models \neg K_S K_R(x_k)$ for all $m \geq m_k$. Thus, S performs the action $\mathsf{send}(\langle k, y \rangle)$ at all times $n'' \geq n'$. Since r is a fair run, there must be some $m' \geq 0$ such that $z^{r(m')} = \langle k, y \rangle$. By Lemma 7.7.2, this implies that $(\mathcal{I}, r, m') \models K_R(x_k)$, which is a contradiction. ∎

The fact that $\mathcal{I}$ satisfies σ^{st} now follows easily. Safety follows from Lemma 7.7.1. For liveness, suppose r is a run in $\mathcal{I}$. We want to show that every data element x_k in X^r is eventually written. By Lemma 7.7.3, there is some m_k such that $j^{r(m_k)} = k$. By Lemma 7.7.1, we have that $|Y^{r(m_k)}| = k$ and that $Y^{r(m_k)} \preceq X^r$, so that x_k is written by time m_k.

Exercises

7.1 Show that if φ is either a standard test or a knowledge test that appears in the knowledge-based program Pg_i for agent i, and $\ell = r_i(m)$ for some point (r, m) in $\mathcal{I}$, then $(\mathcal{I}, \ell) \models \varphi$ iff $(\mathcal{I}, r, m) \models \varphi$. (Hint: use the fact that the propositions in φ are local to i.)

7.2 Let Pg be a standard program, and assume that $\mathcal{I} = (\mathcal{R}, \pi)$. Show that $\mathsf{Pg}^{\mathcal{I}} = \mathsf{Pg}^{\pi}$.

7.3 This exercise completes the details of Example 7.1.2.

(a) Show that $\mathbf{I}^{rep}(\mathsf{BT}, \gamma^{bt}_{fair}, \pi^{bt})$ represents each of BT' and BT'' in the interpreted context $(\gamma^{bt}_{fair}, \pi^{bt})$.

(b) Show further that BT implements BT' and BT'' in the interpreted context $(\gamma^{bt}_{fair}, \pi^{bt})$.

7.4 This exercise completes the details of Example 7.2.1.

(a) Show that the interpreted system $\mathcal{I}^2$ is inconsistent with NU in the interpreted context (γ^{nu}, π^{nu}).

(b) Show that no interpreted system is consistent with NU' in the context (γ^{nu}, π^{nu}), by showing that none of $\mathcal{I}^0$, $\mathcal{I}^1$, nor $\mathcal{I}^2$ are consistent with it.

7.5 Show that even without temporal operators in the knowledge tests, we can still find a knowledge-based program that is represented by more than one interpreted system, and a knowledge-based program that is not represented by any interpreted system in an appropriate context. (Hint: modify the context γ^{nu} described in Example 7.2.1 so that the environment's state includes the time. Modify the transition function τ so that the time m is updated at every step. Then consider the programs

$$\textbf{if } K_1(m \neq 0 \Rightarrow x = 1) \textbf{ do } x := 1$$

and

$$\textbf{if } K_1(m \neq 0 \Rightarrow x \neq 1) \textbf{ do } x := 1\,.)$$

7.6 This exercise completes the details of Example 7.2.2.

(a) Define the recording context (γ, π) corresponding to the description in the example.

(b) Prove that $\mathcal{I}^{\Lambda} \models (K_r p \Leftrightarrow \alpha = 3)$ and that $\mathcal{I}^{\Lambda} \models \neg K_r p'$.

(c) Prove that the program MP_s implements MP in the context (γ, π).

(d) Prove that the program MP'_s implements MP in the context (γ, π).

(e) Prove that every system representing MP in the context (γ, π) coincides with one of the two systems $\mathbf{I}^{rep}(\mathsf{MP}_s, \gamma, \pi)$ and $\mathbf{I}^{rep}(\mathsf{MP}'_s, \gamma, \pi)$ representing MP_s and MP'_s, respectively. (Intuitively, this means that MP_s and MP'_s are the only implementations of MP in this context.)

(f) Let (γ'', π'') be the interpreted context resulting from adding a *halted* bit to the robot's local state, as described in the example. Prove that (c), (d), and (e) still hold when we substitute the context (γ'', π'') for (γ, π).

(g) Describe the context (γ', π') of the example, and prove that there is only one interpreted system representing MP in this context.

7.7 Show that $(\gamma_{fair}^{bt}, \pi^{bt})$ provides witnesses for BT′ and BT″. Moreover, show that γ_{fair}^{bt} is a nonexcluding context.

7.8 Define P_e, $\mathcal{G}_0$, and τ such that neither $(P_e, \mathcal{G}_0, \tau, Fair)$ nor $(P_e, \mathcal{G}_0, \tau, Rel)$ is a nonexcluding context. (Hint: consider a variant of γ^{amp}, where the environment's protocol is such that it never performs the action $\mathsf{deliver}_1(\mu, 2)$ for some message $\mu \in MSG$.)

7.9 In Chapter 6 we observed that the connection between common knowledge, coordinated attack, and SBA described in Proposition 6.1.2 and Theorem 6.4.2 holds only for deterministic protocols. (See the discussion at the end of Section 6.1.) Show that Proposition 6.1.2 and Theorem 6.4.2 hold for nondeterministic protocols as well if we further restrict the contexts to be nonexcluding.

7.10 Show that if P is a protocol and r is a run such that $Pref_m(r) \in Pref_m(\mathbf{R}^{rep}(P, \gamma))$ for all m, then r is weakly consistent with P.

** **7.11** This exercise fills in some of the details of the proof of Theorem 7.2.4.

(a) Show that if P is a protocol and $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ is nonexcluding, then $Pref_0(\mathbf{R}^{rep}(P, \gamma)) = \mathcal{G}_0 \cap Pref_0(\Psi)$.

Next is the key step, which shows that our inductive construction has the right properties. Intuitively, this step shows that, for each interpreted system $\mathcal{I}'$ of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}}, \gamma, \pi)$, the actions of the protocol $\mathsf{Pg}^{\mathcal{I}'}$ at time m depend only on the prefixes of $\mathcal{I}'$ through time m. This is the only place in the proof where we use the assumption that (γ, π) provides witnesses for Pg; this and part (a) are the only places where we use the assumption that γ is nonexcluding.

(b) Assume that $\mathcal{I}_1$ and $\mathcal{I}_2$ provide witnesses for Pg and that $Pref_m(\mathcal{I}_1) = Pref_m(\mathcal{I}_2) = Pref_m(\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}_1}, \gamma, \pi)) = Pref_m(\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}_2}, \gamma, \pi))$. Show that $Pref_{m+1}(\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}_1}, \gamma, \pi)) = Pref_{m+1}(\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}_2}, \gamma, \pi))$. (Hint: suppose $\rho \in Pref_{m+1}(\mathbf{R}^{rep}(\mathsf{Pg}^{\mathcal{I}_1}, \gamma))$, so that there is a run $r \in \mathbf{R}^{rep}(\mathsf{Pg}^{\mathcal{I}_1}, \gamma)$ such

that $\rho = Pref_{m+1}(r)$. Suppose $r(m) = (\ell_e, \ell_1, \ldots, \ell_n)$. It follows that there must be a tuple $(\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n) \in P_e(\ell_e) \times \mathsf{Pg}_1^{\mathcal{I}_1}(\ell_1) \times \cdots \times \mathsf{Pg}_n^{\mathcal{I}_1}(\ell_n)$ such that $r(m+1) = \tau((\mathsf{a}_e, \mathsf{a}_1, \ldots, \mathsf{a}_n))(r(m))$. Show that $\mathsf{a}_i \in \mathsf{Pg}_i^{\mathcal{I}_2}(\ell_i)$ for each agent i. Use this to show that there is a run with prefix ρ that is weakly consistent with $\mathsf{Pg}^{\mathcal{I}_2}$ in context γ. Use the fact that γ is nonexcluding to conclude that there is a run with prefix ρ that is consistent with $\mathsf{Pg}^{\mathcal{I}_2}$ in context γ.)

(c) Show that if $0 \leq m \leq m' < \omega$, then $Pref_m(\mathcal{I}^{m'}) = Pref_m(\mathcal{I}^m)$.

(d) Assume that $\mathcal{I}_1$ is of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}'_1}, \gamma, \pi)$ and $\mathcal{I}_2$ is of the form $\mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}'_2}, \gamma, \pi)$. Show that if $Pref_m(\mathcal{I}_1) = Pref_m(\mathcal{I}_2)$ for all m, then $\mathcal{I}_1 = \mathcal{I}_2$.

Recall that $\mathcal{I}^{\omega+1} = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}^\omega}, \gamma, \pi)$. Define $\mathcal{I}^{\omega+2} = \mathbf{I}^{rep}(\mathsf{Pg}^{\mathcal{I}^{\omega+1}}, \gamma, \pi)$. Our goal now is to prove that $\mathcal{I}^{\omega+1} = \mathcal{I}^{\omega+2}$.

(e) Show that $\mathcal{I}^\omega$ provides witnesses for Pg. (Hint: assume that $\mathcal{I}$ is either $\mathcal{I}^\omega$ or $\mathcal{I}^m$ for some m, and similarly for $\mathcal{I}'$. Let φ be a formula with no temporal operators, such that φ is a subformula or the negation of a subformula of a test in Pg. Prove by induction on the structure of φ that (i) for every run r of $\mathcal{I}$ and run r' of $\mathcal{I}'$ such that $Pref_m(r) = Pref_m(r')$, we have $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}', r, m) \models \varphi$, and (ii) if φ is of the form $K_i\psi$, then $\mathcal{I}$ provides witnesses for φ.)

(f) Show that $Pref_m(\mathcal{I}^{\omega+1}) = Pref_m(\mathcal{I}^{\omega+2})$ for all m. (Hint: show that $Pref_m(\mathcal{I}^{\omega+1}) = Pref_m(\mathcal{I}^m)$ for each m, by using parts (a), (b), (c), and (e). Similarly, show that $Pref_m(\mathcal{I}^{\omega+2}) = Pref_m(\mathcal{I}^m)$ for each m.)

(g) Prove that $\mathcal{I}^{\omega+1} = \mathcal{I}^{\omega+2}$.

(h) Show that $\mathcal{I}^{\omega+1}$ represents Pg in (γ, π).

We have shown that some system represents Pg in (γ, π). We now show uniqueness.

(i) Show that if $\mathcal{I}_1$ and $\mathcal{I}_2$ are systems that represent Pg in (γ, π), then $Pref_m(\mathcal{I}_1) = Pref_m(\mathcal{I}_2)$ for all m.

(j) Show that there is at most one system that represents Pg in (γ, π).

*** 7.12** This exercise provides a weakening of the notion of providing witnesses that is still sufficient to guarantee that there is at most one interpreted system that represents a knowledge-based program Pg in the interpreted context (γ, π) (though it

does not guarantee the existence of such a system). We say that the context (γ, π) provides witnesses *in the weak sense* for Pg if every system $\mathcal{I}$ that represents Pg in (γ, π) provides witnesses for Pg. Show that there is at most one interpreted system representing Pg in (γ, π) if (γ, π) provides witnesses in the weak sense for Pg and γ is a nonexcluding context. (Hint: show by induction on m that if $\mathcal{I} = (\mathcal{R}, \pi)$ and $\mathcal{I}' = (\mathcal{R}', \pi)$ are two systems representing Pg in (γ, π), then $Pref_m(\mathcal{R}) = Pref_m(\mathcal{R}')$ for all m.)

* **7.13** This exercise completes the details of Example 7.2.5.

(a) Show that MC satisfies σ^{mc} in any interpreted context (γ', π^{mc}) where γ' is a subcontext of γ^{mc}.

(b) Show that we can find sets Ψ' and Ψ'' of runs such that $Pref_0(\Psi') = Pref_0(\Psi'') = \mathcal{G}_0$, and if γ' (resp., γ'') is the context that results from replacing *True* in γ^{mc} by Ψ' (resp., Ψ''), then there are two interpreted systems representing MC in (γ', π^{mc}) and no interpreted systems representing MC in (γ'', π^{mc}). Of course, it follows from Theorem 7.2.4 that neither γ' nor γ'' can be nonexcluding. (Hint: define Ψ' to be the set of runs that actually arise in the muddy children puzzle, along with the unique run in the system where it is common knowledge that all of the children have a muddy forehead. Define Ψ'' to be the set of all runs in which the children answer "Yes" to all the father's questions.)

(c) Describe the standard program MC_s formally. (Hint: you need to introduce new primitive propositions.)

(d) Show that $\mathcal{I}^{mc}$ represents MC in (γ^{mc}, π^{mc}) and that MC_s implements MC in (γ^{mc}, π^{mc}).

7.14 Fill in the remaining details of the context γ^{kb} for knowledge bases in Section 7.3.

7.15 Suppose that $\mathcal{I}$ is a system of the form $\mathbf{I}^{rep}(\mathsf{TELL}^{\mathcal{I}'}, \gamma^{kb}, \pi^{kb})$, and at some point (r, m) in $\mathcal{I}$, we have $r_{KB}(m) = \langle \varphi_1, \ldots, \varphi_k \rangle$. Show that there is a run r' in $\mathcal{I}$ such that $r'_{KB}(k) = r_{KB}(m)$. Thus, although the KB may not get a message in every round, it considers it possible that it did get a message in every round.

7.16 Show that there is a unique interpreted system representing TELL in (γ^{kb}, π^{kb}). (Hint: using Theorem 7.2.4, it clearly suffices to show that this context provides

witnesses for TELL. To do this, suppose that $\mathcal{I} = (\mathcal{R}, \pi^{kb})$ and $\mathcal{I}' = (\mathcal{R}', \pi^{kb})$ are systems of the form $\mathbf{I}^{rep}(\mathsf{TELL}^{\hat{\mathcal{I}}}, \gamma^{kb}, \pi^{kb})$ and $\mathbf{I}^{rep}(\mathsf{TELL}^{\hat{\mathcal{I}}'}, \gamma^{kb}, \pi^{kb})$, respectively, such that $Pref_m(\mathcal{R}) = Pref_m(\mathcal{R}')$, and r and r' are two runs in $\mathcal{I}$ and $\mathcal{I}'$ respectively such that $Pref_m(r) = Pref_m(r')$. It clearly suffices to show that for all formulas $\varphi \in \mathcal{L}_n$, we have $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}', r', m) \models \varphi$. The proof proceeds by induction on the structure of φ. The only nontrivial case is if φ is of the form $\mathcal{K}_{KB}\varphi'$. To deal with this case, use Exercise 7.15.)

7.17 Show that $\mathcal{I}^{kb} = \mathbf{I}^{rep}(\mathsf{TELLPROP}, \gamma^{kb}, \pi^{kb})$.

7.18 In the formalization of knowledge bases in Section 4.4.1 in terms of knowledge-based programs, show how we can capture the default assumption that if p is true, then the first thing the KB will be told is p. (Hint: introduce a new proposition *told*, which is initially false and becomes true after the Teller tells the KB at least one formula.)

* **7.19** Prove Theorem 7.3.1. (Hint: define the propositional formula $\varphi = \langle \varphi_1, \ldots, \varphi_k \rangle^*$ by induction on k. We can clearly take $\langle\,\rangle^* = \mathit{true}$, while $\langle \varphi_1, \ldots, \varphi_k, \varphi_{k+1} \rangle^*$ has the form $\langle \varphi_1, \ldots, \varphi_k \rangle^* \wedge \varphi'_{k+1}$. Thus, the problem reduces to defining φ'_{k+1}. By Exercise 3.23, it suffices to assume that φ_{k+1} is of depth 1. Now proceed by induction on the structure of φ_{k+1}.)

* **7.20** This exercise deals with variants of the knowledge-based program OA.

(a) Let OA′ be the knowledge-based program that is obtained by modifying OA so as to allow the agents to perform null actions in every round, just as in TELL. Show that (γ^{oa}, π^{oa}) does *not* provide witnesses for OA′.

(b) Let OA″ be the knowledge-based program that is obtained by modifying OA so as to allow the agents to send formulas with temporal operators. Show that (γ^{oa}, π^{oa}) does *not* provide witnesses for OA″.

7.21 Show that (γ, π^{sba}) provides witnesses for the knowledge-based program SBA, for every $\gamma \in \Gamma^{sba}$.

7.22 Prove Lemma 7.4.3.

7.23 Prove Proposition 7.4.5.

7.24 Prove Theorem 7.4.6.

7.25 Prove Lemma 7.5.1.

7.26 This exercise completes the details of Example 7.5.2.

(a) Show that BT' strongly satisfies σ' in $(\gamma^{bt}_{fair}, \pi^{bt})$.

(b) Show that BT'' strongly satisfies σ'' in $(\gamma^{bt}_{fair}, \pi^{bt})$.

7.27 Show that the test $K_S K_R(x_{@i})$ in ST could be replaced by $\forall k(k = i \Rightarrow K_S K_R(x_k))$ if we had allowed first-order tests in knowledge-based programs.

7.28 Discuss how the context of ST would have to be modified if the environment could deliver more than one message at a time to a given process.

7.29 Given the context γ^{st} for ST described in Section 7.6, describe the effect of $\tau(\mathsf{a}_e, \mathsf{a}_R, \mathsf{a}_S)$, where (1) a_e is $(\mathsf{deliver}_S(*), \mathsf{deliver}_R(\mathit{current}))$, (2) a_S is $i := i + 1$; read, and (3) a_R is write(1); $j := j + 1$.

7.30 Prove that (γ^{st}, π^{st}) provides witnesses for the program ST, and that γ^{st} is nonexcluding. (Hint: the proof that (γ^{st}, π^{st}) provides witnesses for ST is similar in spirit to the proof in Exercise 7.16 that (γ^{kb}, π^{kb}) provides witnesses for TELL in that it uses an analogue of Exercise 7.15.)

7.31 In this exercise, we consider in what contexts ST' implements ST:

(a) Show that ST' implements ST in (γ^{st}, π^{st}).

(b) Show that there are a number of subcontexts $\gamma \sqsubseteq \gamma^{st}$ such that ST' implements ST in (γ, π^{st}). In particular, describe γ and show that this is the case if

  (i) γ corresponds to restricting the environment's protocol so that S and R are always scheduled,

  (ii) γ corresponds to restricting the environment's protocol so that there is no message corruption,

  (iii) γ corresponds to restricting the environment's protocol so that messages are always delivered in the order in which they are sent.

(c) Show that ST' does not implement ST in a context where all messages are guaranteed to be delivered in precisely five rounds.

7.32 Complete the proof of Lemma 7.7.2.

Notes

The notion of a *knowledge-based protocol*, where an agent's actions depended not just on his local state, but also on his knowledge, was introduced by Halpern and Fagin [1989], and studied further by Neiger and Toueg [1993]. Although the idea of a *knowledge-based program*, in the sense of a syntactic object of the type we have here, with explicit tests for knowledge, was implicit in the discussion of [Halpern and Fagin 1985] (an earlier version of [Halpern and Fagin 1989]) and knowledge-based programs were used informally by Dwork and Moses [1990], Halpern and Fagin [1989], Halpern and Zuck [1992], and Moses and Tuttle [1988], the first formal definition of a knowledge-based program seems to have been given by Kurki-Suonio [1986] and by Shoham [1993]. Kurki-Suonio and Shoham, however, did not work with interpreted systems. Rather, they assumed that an agent's knowledge was explicitly encoded in his local state (and thus, in our terminology, was independent of the interpreted system). This means that their knowledge-based programs are really more like our standard programs, although some of the tests in their programs are intuitively thought of as tests for knowledge. Our definition of $(\mathcal{I}, \ell) \models \varphi$ is related to a similar notion used by Neiger [1988].

Sanders [1991] extended the syntax of the programming language UNITY [Chandy and Misra 1988], to allow for explicit tests for knowledge. The observation that even a run-based specification that is satisfied by a knowledge-based program Pg in an interpreted context is not necessarily satisfied by Pg in a subcontext is essentially due to her. Our discussion of the unique system representing a knowledge-based program (including Theorem 7.2.4 and various generalizations) is based on [Fagin, Halpern, Moses, and Vardi 1994]. It is in the spirit of the discussion of canonical models for knowledge-based protocols in [Halpern and Fagin 1989].

Finding weaker conditions that guarantee the existence of at least one, or exactly one, interpreted system representing a given knowledge-based program is an important open problem. In particular, while we conjecture that there is in fact a unique interpreted system representing the variants of OA discussed in Exercise 7.20, we have not been able to prove this. The discussion of a knowledge-based program with multiple implementations in the simple motion-planning context of Example 7.2.2 is based on the work of Brafman, Latombe, Moses, and Shoham [1994].

The model of observing agents was introduced by Fagin and Vardi [1986], who called them communicating scientists. Fagin and Halpern [1988b] examined the role of truthfulness in multi-agent systems in more detail.

The notion of implementation we define here is much stronger than other notions that have been proposed in the literature. Halpern and Fagin [1989], Lamport [1986], and Mazer [1991] discuss other notions of implementation.

Knowledge-based analyses of protocols were carried out by (among others) Bazzi and Neiger [1992], Brafman, Latombe, Moses and Shoham [1994], Chandy and Misra [1986], Dwork and Moses [1990], Hadzilacos [1987], Halpern, Moses, and Waarts [1990], Halpern and Zuck [1992], Fischer and Immerman [1986], Kurki-Suonio [1986], Mazer [1990], Mazer and Lochovsky [1990], Merritt and Taubenfeld [1991], Michel [1989b], Moses and Kislev [1993], Moses and Roth [1989], Moses and Tuttle [1988], Neiger [1988], Neiger and Bazzi [1992], Neiger and Toueg [1993], and Neiger and Tuttle [1993]. The first use of a knowledge-based program as a tool for designing a standard program was by Dwork and Moses [1990]. Their discussion, and that of Moses and Tuttle [1988], forms the basis of Section 7.4. A generalization of knowledge-based programs, called *knowledge-oriented programs*, is suggested by Moses and Kislev [1993]. They consider programs that involve, in addition to tests for knowledge, high-level actions that are defined in terms of their effect on the state of knowledge of the agents.

A knowledge-based analysis of the sequence-transmission problem in an asynchronous setting was carried out in detail by Halpern and Zuck [1992]; it serves as the basis for our treatment in Sections 7.6 and 7.7. Well-known solutions to the sequence-transmission problem include the alternating-bit protocol [Bartlett, Scantlebury, and Wilkinson 1969], Stenning's protocol [1976], and the protocols of Aho, Ullman, Wyner, and Yannakakis [1979, 1982]. Stenning's protocol is designed to work in asynchronous systems where messages can be deleted, duplicated, reordered, or detectably corrupted; it is essentially our ST'. As is shown by Halpern and Zuck, the other protocols mentioned here can be derived from ST as well. This leads to relatively straightforward proofs of their correctness. For other proofs of correctness of programs for the sequence-transmission problem, the reader should consult Bochmann and Gecsei [1977], Gouda [1985], Hailpern [1982, 1985], and Hailpern and Owicki [1983]. A knowledge-oriented program for the sequence transmission problem is presented by Moses and Kislev [1993]. That program generalizes ST and satisfies σ^{st} in a wider range of contexts.

Chapter 8

Evolving Knowledge

We may with advantage at times forget what we know.

Publilius Syrus, c. 100 B.C.

In Chapters 2 and 3 we studied the properties of knowledge in depth. The possible-worlds setting in which we conducted our study was static. There was no notion of an agent gaining or losing knowledge over time. On the other hand, our formal model of multi-agent systems described in Chapter 4 explicitly includes time, and we extended the language with temporal operators in Section 4.3. We already saw that incorporating time in our framework is quite useful. For example, we saw in Section 4.5 that we can gain useful insights into a.m.p. systems by understanding how knowledge is gained or lost over time. In the analysis of SBA in Chapter 6 we also saw a specific example of how knowledge evolves when the full-information protocol is executed in specific contexts of interest. In this chapter, we study in more generality how knowledge evolves in multi-agent systems.

8.1 Properties of Knowledge and Time

In Chapters 2 and 3 we characterized the properties of knowledge using an axiom system. We can similarly characterize the properties of time. We focus here on axioms for $\bigcirc$ and U, since as we noted, $\Box$ and $\Diamond$ are expressible in terms of U.

T1. $\bigcirc\varphi \wedge \bigcirc(\varphi \Rightarrow \psi) \Rightarrow \bigcirc\psi$

T2. $\bigcirc\neg\varphi \Leftrightarrow \neg \bigcirc \varphi$

T3. $\varphi U \psi \Leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi U \psi))$

RT1. From φ infer $\bigcirc\varphi$

RT2. From $\varphi' \Rightarrow \neg\psi \wedge \bigcirc\varphi'$ infer $\varphi' \Rightarrow \neg(\varphi U \psi)$

Axiom T1 and rule RT1 show that the modal operator $\bigcirc$ has some of the same flavor as K_i; axiom T1 is analogous to the Distribution Axiom for knowledge (A2), while RT1 is analogous to the Rule of Knowledge Generalization (R1). The difference between K_i and $\bigcirc$ is summarized by T2, which says that time is deterministic. There is only one way the world can be at the next step, so that if at the next step φ is false, then it is not the case that φ can be true at the next step. (We could also consider *branching time* operators, where time is taken to be nondeterministic. See the notes at the end of the chapter for further discussion of this point.) Axiom T3 captures the relationship between $\bigcirc$ and U. Intuitively, it says that φ holds until ψ does exactly if either ψ is true now, or φ is true now and at the next step it is still the case that φ holds until ψ does. Finally, rule RT2 gives us a way to conclude that $\neg(\varphi U \psi)$ holds. As we show below, it is an analogue to the Induction Rule for common knowledge.

As we shall see, these axioms are both sound and complete for the language of temporal logic, so they can be viewed as a complete characterization of the properties of time. Furthermore, when combined with the S5 axioms for knowledge, they form a complete axiomatization for knowledge and time in multi-agent systems. Thus, the combined set of axioms can be viewed as a complete characterization of the properties of evolving knowledge in multi-agent systems. To state this result carefully, we need some notation analogous to that defined in Chapter 3. Let $\mathcal{L}_n^U(\Phi)$ be the result of augmenting $\mathcal{L}_n(\Phi)$, the language of knowledge defined in Chapter 3, with the modal operators U and $\bigcirc$. We similarly define the language $\mathcal{L}_n^{CU}(\Phi)$ as the analogous extension of $\mathcal{L}_n^C(\Phi)$, the language of knowledge and common knowledge. Again, we typically omit mention of Φ when it is clear from context. In this section, we focus attention on knowledge and common knowledge, and how they interact with time. Adding distributed knowledge to the language does not seem to cause any difficulties; see the notes at the end of the chapter for further discussion. As we shall see in later sections, however, in other circumstances distributed knowledge does play a key role in capturing properties of the system.

We first prove soundness. Let $\mathcal{C}_n$ be the class of all interpreted systems with n agents.

Theorem 8.1.1 *For all formulas $\varphi, \psi \in \mathcal{L}_n^{CU}$ and interpreted systems $\mathcal{I} \in \mathcal{C}_n$:*

(a) $\mathcal{I} \models \bigcirc\varphi \wedge \bigcirc(\varphi \Rightarrow \psi) \Rightarrow \bigcirc\psi$,

(b) $\mathcal{I} \models \bigcirc\neg\varphi \Leftrightarrow \neg\bigcirc\varphi$,

(c) $\mathcal{I} \models \varphi U \psi \Leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi U \psi))$,

(d) if $\mathcal{I} \models \varphi$ *then* $\mathcal{I} \models \bigcirc\varphi$,

(e) if $\mathcal{I} \models \varphi' \Rightarrow (\neg\psi \wedge \bigcirc\varphi')$ *then* $\mathcal{I} \models \varphi' \Rightarrow \neg(\varphi U \psi)$.

Proof We sketch a proof of part (e) here, leaving the proofs of the other parts to the reader (Exercise 8.1). The proof of part (e) has some of the flavor of the proof of validity of the Induction Rule for common knowledge. Fix an interpreted system $\mathcal{I} \in \mathcal{C}_n$ and suppose that $\mathcal{I} \models \varphi' \Rightarrow \neg\psi \wedge \bigcirc\varphi'$. By induction on k we can show that $\mathcal{I} \models \varphi' \Rightarrow \bigcirc^k\varphi'$ (where $\bigcirc^k\varphi$ is an abbreviation for $\bigcirc \ldots \bigcirc \varphi$, with k occurrences of $\bigcirc$). It follows that $\mathcal{I} \models \varphi' \Rightarrow \Box\varphi'$. And since $\mathcal{I} \models \varphi' \Rightarrow \neg\psi$, we also get that $\mathcal{I} \models \varphi' \Rightarrow \Box\neg\psi$. Thus, ψ is never true after a point where φ' is true.

But this means that for any choice of φ, the formula $\varphi U \psi$ cannot be true either after any point where φ' is true (since if $\varphi U \psi$ were true, then at some point in the future ψ would have to be true). Thus, $\mathcal{I} \models \varphi' \Rightarrow \neg(\varphi U \psi)$. ∎

We next prove completeness. More precisely, let $\mathrm{S5}_n^U$ (resp., $\mathrm{S5}_n^{CU}$) be the axiom system that results from adding T1, T2, T3, RT1, and RT2 to $\mathrm{S5}_n$ (resp., $\mathrm{S5}_n^C$). Thus, $\mathrm{S5}_n^U$ is the result of combining the axioms and rules for knowledge and time into one system; $\mathrm{S5}_n^{CU}$ is the result of adding axioms and rules for common knowledge. There are no additional axioms for knowledge and no axioms describing the interaction between knowledge and time.

Theorem 8.1.2 $\mathrm{S5}_n^U$ *(resp.,* $\mathrm{S5}_n^{CU}$*) is a sound and complete axiomatization for the language* $\mathcal{L}_n^U$ *(resp.,* $\mathcal{L}_n^{CU}$*) with respect to* $\mathcal{C}_n$.

Proof Soundness follows immediately from Theorem 8.1.1 and the fact that the axioms of $\mathrm{S5}_n$ are valid in multi-agent systems. To prove completeness, we use ideas similar to those we have already encountered in the completeness proofs in Chapter 3. We just sketch the main ideas in the case of $\mathcal{L}_n^U$ here, leaving details (and the $\mathcal{L}_n^{CU}$ case) to the reader (Exercise 8.2).

Suppose $\varphi \in \mathcal{L}_n^U$ is consistent with $\mathrm{S5}_n^U$. We want to show that there is some interpreted system $\mathcal{I}$ and some point (r, m) in $\mathcal{I}$ such that $(\mathcal{I}, r, m) \models \varphi$. As in the proof of Theorem 3.1.3, each state corresponds to a set of formulas. The difference

is that here we have to define both the possibility relations and a temporal relation between the states. As before, we take $Sub^+(\varphi)$ to consist of all the subformulas of φ and their negations. Let $S = \{s_V \mid V \text{ is a maximal } S5_n^U\text{-consistent subset of } Sub^+(\varphi)\}$. We define equivalence relations $\mathcal{K}_i$ on S as suggested by the proof of Theorem 3.2.4: we take $(s_V, s_W) \in \mathcal{K}_i$ iff $V/K_i = W/K_i$ where, as before, for an arbitrary set X of formulas, we define $X/K_i = \{\varphi \mid K_i\varphi \in X\}$. We also define a binary relation $\mathcal{T}$ on S by taking $(s_V, s_W) \in \mathcal{T}$ iff $V/\bigcirc \subseteq W$, where

$$V/\bigcirc = \{\psi \mid \bigcirc\psi \in V\} \cup \{\neg\psi \mid \neg\bigcirc\psi \in V\}.$$

(Note that although $\neg\bigcirc\psi \Rightarrow \bigcirc\neg\psi$ is valid, we would not get the same set if we had just defined $V/\bigcirc$ as $\{\psi \mid \bigcirc\psi \in V\}$. The problem is that $\neg\bigcirc\psi \in Sub^+(\varphi)$ does not imply $\bigcirc\neg\psi \in Sub^+(\varphi)$.) We can now use techniques like those used in our earlier proofs to show that for all maximal consistent subsets V of $Sub^+(\varphi)$ and all formulas $\psi \in Sub^+(\varphi)$, we have

1. if ψ is of the form $K_i\psi'$, then $\psi \in V$ iff $\psi' \in W$ for all W such that $(s_V, s_W) \in \mathcal{K}_i$,
2. if ψ is of the form $\bigcirc\psi'$, then $\psi \in V$ iff $\psi' \in W$ for all W such that $(s_V, s_W) \in \mathcal{T}$,
3. if ψ is of the form $\psi_1 U \psi_2$, then $\psi \in V$ iff there exists a sequence $(s_{V_0}, \ldots, s_{V_k})$ of states such that $V_0 = V$, $(s_{V_l}, s_{V_{l+1}}) \in \mathcal{T}$ for $l < k$, $\psi_2 \in V_k$, and $\psi_1 \in V_l$ for $l < k$ (see Exercise 8.2).

We say an infinite sequence $(s_{V_0}, s_{V_1}, \ldots)$ of states is *acceptable* if

1. $(s_{V_k}, s_{V_{k+1}}) \in \mathcal{T}$ for all $k \geq 0$, and
2. for all k, if the formula $\psi_1 U \psi_2 \in V_k$, then there exists $l \geq k$ such that $\psi_2 \in V_l$ and $\psi_1 \in V_{l'}$ for all l' with $k \leq l' < l$.

It is not hard to show from the properties described above that for every state $s \in S$ there is an acceptable sequence that starts with s. Finally, we now define an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$. Let $A = (s_{V_0}, s_{V_1}, \ldots)$ be an arbitrary acceptable sequence of states in S. Define the run r^A by taking $r_i^A(m) = V_m/K_i$, for $i = 1, \ldots, n$, and setting $r_e^A(m) = \emptyset$. We now define $\mathcal{R}$ to be the set of all such runs r^A. Define π by setting $\pi(r^A(m))(p) = \textbf{true}$ iff $p \in V_m$. Let $\mathcal{I} = (\mathcal{R}, \pi)$. It is now straightforward to show (using the properties of $\mathcal{K}_i$ and $\mathcal{T}$ listed above) that if A is the acceptable sequence $(s_{V_0}, s_{V_1}, \ldots)$, then $(\mathcal{I}, r^A, m) \models \psi$ iff $\psi \in V_m$, for all

formulas $\psi \in Sub^+(\varphi)$. Since φ is in some maximal consistent subset of $Sub^+(\varphi)$, it must be the case that φ is satisfied in the interpreted system $\mathcal{I}$. ∎

This result shows that in the most general setting, where we consider the class of *all* interpreted systems, there is no interaction between knowledge and time. That is, we can completely characterize the properties of knowledge and time by simply combining the separate characterizations of knowledge and of time. What are the properties of knowledge and time in less general settings? In Chapter 3, we considered classes of structures determined by properties of the $\mathcal{K}_i$ relations, and saw that the properties of knowledge could vary, depending on the class of structures under consideration. Given this experience, the reader may not find it surprising that the properties of knowledge and time in multi-agent systems, and in particular the interaction between knowledge and time, depend strongly on the setting, i.e., on the class of interpreted systems under consideration. What is surprising is how subtle this dependence can be. We illustrate this situation by considering a number of the classes we have discussed in previous chapters, namely, synchronous systems, systems with perfect recall, asynchronous message passing systems, and systems of observing agents.

8.2 Synchrony and Perfect Recall

We start by considering systems that are synchronous and systems where agents have perfect recall. We use the superscript *pr* on $\mathcal{C}_n$ to indicate a restriction to those systems in $\mathcal{C}_n$ in which agents have perfect recall and the superscript *sync* to indicate a restriction to synchronous systems. For example, $\mathcal{C}_n^{pr,sync}$ represents the class of synchronous systems where agents have perfect recall.

If we add just the requirement of synchrony, it turns out that we get no additional properties at all. This shows that our language is not strong enough to capture synchrony.

Theorem 8.2.1 $S5_n^U$ *(resp.,* $S5_n^{CU}$*) is a sound and complete axiomatization for the language* $\mathcal{L}_n^U$ *(resp.,* $\mathcal{L}_n^{CU}$*) with respect to* $\mathcal{C}_n^{sync}$.

Proof The proof is a straightforward modification of the proof of Theorem 8.1.2. The only difference is that we take $r_i^A(m) = (m, V_m/K_i)$. By making the time part of agent i's state, we guarantee that the system is synchronous. The only difficulty now is in showing that it is still the case that if $K_i\psi \in Sub^+(\varphi)$, then we have $(\mathcal{I}, r^A, m) \models K_i\psi$ iff $K_i\psi \in V_m$. This is done by showing that if a state appears in an acceptable sequence, then for all k, we can find an acceptable sequence in which

it is the k^{th} member. This in turn requires two additional observations: (1) for all V, there exists W such that $(s_W, s_V) \in \mathcal{T}$ and (2) any finite sequence $(s_{V_0}, \ldots, s_{V_k})$ of states such that $(s_{V_l}, s_{V_{l+1}}) \in \mathcal{T}$ for $0 \leq l < k$ can be extended to an acceptable sequence. Again, we leave details to the reader (Exercise 8.3). ∎

If we restrict attention to systems where agents have perfect recall, then knowledge and time do interact. As we observed in Section 4.4.4, although $K_i\varphi \Rightarrow \Box K_i\varphi$ is not valid in general, it is valid if we restrict to stable formulas (formulas that, once true, remain true). It is not hard to show that a formula ψ is stable in an interpreted system $\mathcal{I}$ precisely if it is equivalent in the system to a formula of the form $\Box\varphi$ for some formula φ (Exercise 8.4). This observation suggests the following axiom:

KT1. $K_i\Box\varphi \Rightarrow \Box K_i\varphi, \quad i = 1, \ldots, n$

Axiom KT1 says, roughly, that formulas that are known to always be true are always known to be true. It is easily seen to be valid with respect to $\mathcal{C}_n^{pr}$ (Exercise 8.5). Of course, it is still valid with respect to $\mathcal{C}_n^{pr,sync}$, but once we assume synchrony, we get an additional property. Intuitively, if an agent that has perfect recall can also keep track of time, then if he knows that a formula φ will hold at the next time instant, then at the next instant he will be able to notice that the time has elapsed, and so will know φ then. This intuition is captured by the following axiom:

KT2. $K_i \bigcirc \varphi \Rightarrow \bigcirc K_i\varphi, \quad i = 1, \ldots, n$

It is easy to see that axiom KT2 is valid with respect to $\mathcal{C}_n^{pr,sync}$ (Exercise 8.5). Axiom KT2 is actually a strengthening of axiom KT1; KT1 can be deduced from KT2 and the other axioms and rules of S5_n^U, using the fact that $\Box\varphi$ is an abbreviation for $\neg(true\, U \neg\varphi)$ (Exercise 8.6).

Are there any further properties of knowledge and time in synchronous systems with perfect recall? The following result shows that there are not.

Theorem 8.2.2 $\text{S5}_n^U + \{\text{KT2}\}$ *is a sound and complete axiomatization for the language* $\mathcal{L}_n^U$ *with respect to* $\mathcal{C}_n^{pr,sync}$.

What happens when we drop the assumption of synchrony? It can be shown that $\text{S5}_n^U + \{\text{KT1}\}$ is not complete for $\mathcal{L}_n^U$ with respect to $\mathcal{C}_n^{pr}$. By adding a rather complicated axiom, a sound and complete axiomatization can be obtained; see Exercise 8.7 and the bibliographic notes for further discussion.

We next consider what happens when we add common knowledge to the language, still restricting attention to systems where agents have perfect recall. It is easy to show that $\text{S5}_n^{CU} + \{\text{KT1}\}$ is a sound axiomatization for $\mathcal{L}_n^{CU}$ with respect to $\mathcal{C}_n^{pr}$,

while $S5_n^{CU} + \{KT2\}$ is sound with respect to $\mathcal{C}_n^{pr,sync}$. Our earlier results might lead us to expect completeness, at least in the latter case. However, we can prove that we do not get completeness in either case. In fact, there can be *no* sound and complete axiomatization for the language $\mathcal{L}_n^{CU}$ with respect to either $\mathcal{C}_n^{pr}$ or $\mathcal{C}_n^{pr,sync}$ consisting of only finitely many axiom schemes and rules! Adding common knowledge to the language considerably complicates things. To make this statement precise, we need to consider the complexity of the validity problem.

We start by considering validity with respect to $\mathcal{C}_n$. How hard is it to decide if a formula in $\mathcal{L}_n^U$ (resp., $\mathcal{L}_n^{CU}$) is valid with respect to $\mathcal{C}_n$? Equivalently (thanks to Theorem 8.1.2) how hard is it to decide if a formula is provable in $S5_n^U$ (resp., $S5_n^{CU}$)? We saw in Chapter 3 that deciding validity for formulas involving knowledge alone (that is, for the language $\mathcal{L}_n$) is co-*NP*-complete in the case of one agent (i.e., if $n = 1$) and *PSPACE*-complete if there are two or more agents. Adding distributed knowledge does not affect the complexity, but with common knowledge in the language (and at least two agents in the system) the complexity of the validity problem goes up to *EXPTIME*-complete. If we consider pure temporal formulas (i.e., formulas where the only modal operators that appear are $\bigcirc$ and U), then the validity problem for that language is also known to be *PSPACE*-complete. Clearly the complexity of the validity problem for the language involving both knowledge and time must be at least as high as the complexity for knowledge or for time separately, since formulas involving only knowledge or only time are a subset of those involving both. The interaction between knowledge and time, however, might, in general, make things much worse. In the case of $\mathcal{C}_n$, we have seen that there is no interaction between knowledge and time reflected in the axioms. This also extends to complexity (of validity); the complexity of reasoning about knowledge and time together is no worse than the complexity of reasoning about each of knowledge and time separately. As can be seen from the top row of Table 8.1, the validity problem with respect to $\mathcal{C}_n$ is *PSPACE*-complete for the language $\mathcal{L}_n^U$, and *EXPTIME*-complete for the language $\mathcal{L}_n^{CU}$. Since precisely the same formulas are valid with respect to $\mathcal{C}_n^{sync}$ and $\mathcal{C}_n$, the results are identical in the case of $\mathcal{C}_n^{sync}$.

Once we assume that agents have perfect recall, the validity problem becomes dramatically harder. To make precise exactly how hard it becomes, we need to introduce some additional notation. Define $ex_m(k)$ inductively on m, by setting $ex_0(k) = k$ and $ex_{m+1}(k) = 2^{ex_m(k)}$. (Intuitively, $ex_m(k)$ is a stack of m 2's, with the top 2 having exponent k.) It turns out that the validity problem for $\mathcal{L}_n^U$ with respect to $\mathcal{C}_n^{pr}$ (resp., $\mathcal{C}_n^{pr,sync}$) is *nonelementary*. That is, the complexity of deciding validity is not bounded by ex_m for any m. For every algorithm $\mathcal{A}$ that correctly decides the

validity problem and every m, there is a formula φ such that $\mathcal{A}$ will take time at least $ex_m(|\varphi|)$ when running on φ.

In fact, we can get an even more precise characterization of the complexity of the validity problem. Let the *alternation depth* of a formula $\varphi \in \mathcal{L}_n^U$, written $ad(\varphi)$, be the number of alternations of distinct K_i's in φ; temporal operators do not count. We take the alternation depth of a formula that mentions only one of the K_i operators to be one. Thus, the alternation depth of $K_1 \Box K_1(p \wedge K_1 q)$ is one, the alternation depth of $K_1 K_2 p$ is two, the alternation depth of both $K_1 \Box \neg K_2(p U K_1 q)$ and $K_1 \neg K_2 K_1 q$ is three, and the alternation depth of $K_1 K_2 K_3 K_2 q$ is four. There exists an algorithm that decides whether a formula $\varphi \in \mathcal{L}_n^U$ is valid with respect to $\mathcal{C}_n^{pr}$ (resp., $\mathcal{C}_n^{pr,sync}$) that runs in time $ex_{ad(\varphi)+1}(c|\varphi|)$, for some constant $c > 0$. Thus, the time required looks like a stack of 2's of height one greater than the alternation depth of φ, with the top 2 having exponent $c|\varphi|$, for some constant $c > 0$, and this running time is essentially optimal.

We can define the complexity class *nonelementary time* to consist of those sets A such that for some constant $c > 0$, the question of whether $x \in A$ can be decided in time $ex_{|x|}(c|x|)$. The techniques used in the analysis above actually show that the validity problem for the language $\mathcal{L}_n^U$ with respect to $\mathcal{C}_n^{pr}$ (resp., $\mathcal{C}_n^{pr,sync}$) is *nonelementary time complete*: it is decidable in nonelementary time, and any problem decidable in nonelementary time can be reduced to this validity problem. Since the alternation depth of any formula that mentions only the knowledge of agent 1 (i.e., where the only knowledge operator that appears is K_1) is at most one, the same techniques can also be used to show that the validity problem for the language $\mathcal{L}_1^U$ is doubly-exponential time complete.

Once we add common knowledge to the language, things get significantly worse. The validity problem is not decidable at all. That is, there is no program which, given an arbitrary formula in $\mathcal{L}_n^{CU}$, can decide if it is valid with respect to $\mathcal{C}_n^{pr}$ or $\mathcal{C}_n^{pr,sync}$. In fact, we can characterize exactly how undecidable the validity problem is; technically, it is what is known as Π_1^1-complete. The exact definition of Π_1^1 is not all that relevant here, but the following observation is: if a logic has a sound and complete axiomatization with a finite collection of axioms (actually, axiom schemes) and inference rules, then the set of valid formulas must be *recursively enumerable*, that is, there must be a program that can generate all the valid formulas of the language in some systematic way. This is easy to see. Suppose we have a sound and complete axiom system AX. It is straightforward to construct a program that generates all the possible proofs in AX. In this way, it will actually generate all the provable (and, since AX is complete, all the valid) formulas. Since AX is sound, it will generate only valid formulas. However, a set of formulas that is Π_1^1-complete is not

	$\mathcal{L}_n^U, \mathcal{L}_n^{CU}, n = 1$	$\mathcal{L}_n^U, n \geq 2$	$\mathcal{L}_n^{CU}, n \geq 2$
$\mathcal{C}_n, \mathcal{C}_n^{sync}$	*PSPACE*-complete	*PSPACE*-complete	*EXPTIME*-complete
$\mathcal{C}_n^{pr}, \mathcal{C}_n^{pr,sync}$	doubly-exponential time complete	nonelementary time complete	Π_1^1-complete

Table 8.1 The complexity of the validity problem for logics of knowledge and time

recursively enumerable. Thus, the fact that the validity problem for $\mathcal{L}_n^{CU}$ with respect to $\mathcal{C}_n^{pr}$ (resp., $\mathcal{C}_n^{pr,sync}$) is Π_1^1-complete implies that there cannot be a finite complete axiomatization in this case. In fact, the same argument shows that there cannot even be a recursively enumerable set of axioms that is complete.

These results are summarized in Table 8.1.

8.3 Knowledge and Time in A.M.P. Systems

We now turn our attention to asynchronous message passing systems. Let $\mathcal{C}_n^{amp}$ be the class of interpreted a.m.p. systems with n agents. Since $\mathcal{C}_n^{amp} \subset \mathcal{C}^{pr}$, it follows from our earlier observations that $S5_n^U + \{KT1\}$ is a sound axiomatization for $\mathcal{L}_n^U$ with respect to $\mathcal{C}_n^{amp}$. It is not, however, complete. There are a number of additional properties that are a consequence of Theorem 4.5.3 and are thus sound for $\mathcal{L}_n^U$ with respect to $\mathcal{C}_n^{amp}$. For example, it is not hard to show that $\bigcirc K_i K_j \varphi \Rightarrow K_j \varphi$ is valid in $\mathcal{C}_n^{amp}$ if $i \neq j$. For suppose that $K_i K_j \varphi$ holds at the point $(r, m+1)$ in some interpreted system $\mathcal{I} \in \mathcal{C}_n^{amp}$. If $\neg K_j \varphi$ holds at (r, m) then, by Theorem 4.5.3, $\langle j, i \rangle$ must be a process chain in $(r, m \,..\, m+1)$. But this means that there must be an event e_1 in j's history that occurs at or after round $m+1$ and an event e_2 in i's history that occurs at or before round $m+1$ such that $e_1 \xrightarrow{r} e_2$. It is easy to see that this cannot happen.

A different property arises from the fact that an agent can "stutter," that is, stay in the same state for a long time. For example, because of stuttering, $K_i \bigcirc K_i \varphi \Rightarrow K_i \varphi$ is valid with respect to $\mathcal{C}_n^{amp}$. For suppose that $\mathcal{I} \in \mathcal{C}_n^{amp}$ and $(\mathcal{I}, r, m) \models K_i \bigcirc K_i \varphi$. Because of stuttering, there is a point (r', m) such that $r(m) = r'(m)$ and agent i's state does not change between (r', m) and $(r', m+1)$. It follows both that $(\mathcal{I}, r', m+1) \models K_i \varphi$ and that $(r', m+1) \sim_i (r', m) \sim_i (r, m)$. Hence, $(\mathcal{I}, r, m) \models K_i \varphi$.

A case can be made that it is inappropriate to include $\bigcirc$ in a language for reasoning about asynchronous systems. The whole point of asynchrony is that the notion of

"next time" does not make sense! But even without $\bigcirc$, we have a new property. Because of stuttering, we can show that $K_i \Diamond K_i \varphi \Rightarrow K_i \varphi$ is also valid with respect to $\mathcal{C}_n^{amp}$. (See Exercise 8.8 for further discussion of properties of knowledge in $\mathcal{C}_n^{amp}$.)

Once we add common knowledge to the language, there are further properties. As we saw in Theorem 4.5.4, common knowledge can be neither gained nor lost in a.m.p. systems. This fact can be captured axiomatically:

CG. $\neg C_G \varphi \Rightarrow \Box \neg C_G \varphi$ if $|G| \geq 2$

This axiom says only that common knowledge cannot be gained. It might seem that we need another axiom of the form $C_G \varphi \Rightarrow \Box C_G \varphi$ to ensure that common knowledge cannot be lost. In fact, this axiom already follows from axiom CG together with other properties of knowledge and common knowledge, particularly negative introspection (see Exercise 8.9).

8.4 Knowledge and Time in $\mathcal{I}_n^{oa}(\Phi)$

Finally, we consider properties of knowledge and time in the system $\mathcal{I}_n^{oa}(\Phi)$ of observing agents, as described in Section 7.3. This will be a case study illustrating how subtle the interaction between knowledge and time in a specific setting may be, and how sensitive it is to small changes in modeling the system. We continue to write the set Φ of primitive propositions in $\mathcal{I}_n^{oa}(\Phi)$, since the set Φ will figure prominently in an axiom we discuss later. Our construction of $\mathcal{I}_n^{oa}(\Phi)$ clearly assumes that the agents have perfect recall and that the system is synchronous, so KT2 is valid with respect to $\mathcal{I}_n^{oa}(\Phi)$. An additional property follows because we assumed that the truth assignment does not change over time. This fact is characterized by the following axiom CP (which stands for "constant propositions"):

CP. $\Box p \vee \Box \neg p$, for all $p \in \Phi$

In Section 4.5 we showed that common knowledge is neither gained nor lost in a.m.p. systems. In the system of observing agents, common knowledge can be gained, since the system is synchronous (although common knowledge of stable facts cannot be lost, by the results of Exercise 4.18, since the agents have perfect recall). However, as we now show, *distributed* knowledge of propositional formulas (ones which do not mention any modal operators) cannot be gained or lost among the group of all agents. More formally, let $D\varphi$ be an abbreviation of $D_G \varphi$, when G is the group consisting of all the agents in the system. Now consider the following two axioms:

DG1. $D\varphi \Rightarrow \Box D\varphi$ if φ is a propositional formula

DG2. $\neg D\varphi \Rightarrow \Box \neg D\varphi$ if φ is a propositional formula

Theorem 8.4.1 *DG1 and DG2 are both valid with respect to* $\mathcal{I}_n^{oa}(\Phi)$.

Proof Since the truth value of a primitive proposition does not change throughout a run, it immediately follows that a propositional formula φ must be *stable*; if $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \varphi$, then $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \Box\varphi$. Since $\mathcal{I}_n^{oa}(\Phi)$ is a synchronous system where agents have perfect recall, it follows that $D\varphi$ is stable too (Exercise 4.18). Thus $\mathcal{I}_n^{oa}(\Phi) \models D\varphi \Rightarrow \Box D\varphi$.

Now let r be a run in $\mathcal{I}_n^{oa}(\Phi)$, and suppose $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \neg D\varphi$. We must show that $(\mathcal{I}_n^{oa}(\Phi), r, m') \models \neg D\varphi$ for all $m' > m$. The validity of DG1 implies that $(\mathcal{I}_n^{oa}(\Phi), r, 0) \models \neg D\varphi$. Suppose $(\mathcal{T}_i, \langle\rangle)$ is the initial state of agent i in run r, for $i = 1, \ldots, n$. Since $D\varphi$ is not true at the point $(r, 0)$, there is a truth assignment α' that makes φ false such that $\alpha' \in \cap_{i=1}^{n} \mathcal{T}_i$. Define r' to be a run exactly like r (in that the same messages are sent and received at every step) except that α' is the truth assignment in the environment component of r'. We want to show that r' is a run in $\mathcal{I}_n^{oa}(\Phi)$. Recall that we obtained $\mathcal{I}_n^{oa}(\Phi)$ by applying Theorem 7.2.4. To show that r' is a run in $\mathcal{I}_n^{oa}(\Phi)$, we must show that each prefix of r' appears in the inductive construction of $\mathcal{I}_n^{oa}(\Phi)$ in the proof of Theorem 7.2.4, which in turn means that we must show that the only messages sent in r' are ones known to be true. But since by construction $(r, m') \sim_i (r', m')$ for all times m' and all agents i, it is easy to show that this is the case by induction on m' (Exercise 8.11). Since α' is the truth assignment at every point of r', it follows that $(\mathcal{I}_n^{oa}(\Phi), r', m') \models \neg\varphi$ for all $m' \geq 0$. By construction, $(r, m') \sim_i (r', m')$ for all agents i and times $m' \geq 0$. Thus, we have $(\mathcal{I}_n^{oa}(\Phi), r, m') \models \neg D\varphi$ for all $m' \geq 0$, giving us the desired result. ∎

The reader should compare this to Theorem 6.7.1, which says that when running the full-information protocol for SBA in the case of crash failures, distributed knowledge cannot be gained. Note, however, that because of the possibility of faulty processes in that context, distributed knowledge can be lost there.

It is easy to show that distributed knowledge of a stable formula cannot be *lost* among any subgroup of agents (Exercise 4.18). That is, DG1 holds if we replace $D\varphi$ by $D_G\varphi$, where G is any nonempty subgroup of agents and φ is an arbitrary stable formula. We cannot, however, strengthen DG2 in this way, as the following examples show.

Example 8.4.2 In this example, we show that the assumption that the formula φ is propositional is necessary for the validity of DG2. Let p be a primitive proposition. Neither Alice nor Bob initially knows whether p is true or false, but Charlie knows that p is true. Charlie sends Alice the message p, which is received by Alice in the first round. Let φ be the formula $K_{Alice}p$; this is a stable formula. This formula is initially false, but becomes true after Alice receives the message. Thus, initially $D\varphi$ does not hold (since φ is false), but when Alice receives the message from Charlie, both φ and $D\varphi$ become true; thus distributed knowledge is gained. ∎

Example 8.4.3 In this example, we show that considering distributed knowledge only of the set of *all* agents is necessary for the validity of DG2. The situation is just as in the previous example. Thus, p is a primitive proposition; neither Alice nor Bob initially knows whether p is true or false, but Charlie knows that p is true; and Charlie sends Alice the message p, which is received by Alice in the first round. Now let G consist of Alice and Bob. Initially $\neg D_G p$ holds, but after Alice receives Charlie's message, $D_G p$ holds. Again, distributed knowledge is gained. ∎

Theorem 8.4.1 shows that there is some interaction between knowledge and time in the system $\mathcal{I}_n^{oa}(\Phi)$. We now demonstrate that the properties of evolving knowledge also affect the properties of static knowledge. More precisely, we show that in $\mathcal{I}_n^{oa}(\Phi)$ there are extra properties of knowledge alone, not involving time, over and above the S5 axioms. We start by considering an example.

Example 8.4.4 Assume that there are precisely two agents, namely Alice and Bob, and assume that there is exactly one primitive proposition p in the language. We can think of p as a fact that characterizes nature at a given time (for example, if all we care about is whether or not it rained in San Francisco on January 1, we could take p to be "It rained in San Francisco on January 1"). Consider a situation where p is in fact true, but Alice doesn't know whether p is true or false, and Alice knows that either p is true and Bob knows that p is true, or p is false and Bob doesn't know that p is false. Alice's state of knowledge can be captured by the formula

$$\varphi_{AB} =_{\text{def}} \neg K_{Alice}p \wedge \neg K_{Alice}\neg p \wedge K_{Alice}\big((p \wedge K_{Bob}p) \vee (\neg p \wedge \neg K_{Bob}\neg p)\big).$$

It is easy to show that φ_{AB} is consistent with $S5_2$, where the two agents are Alice and Bob (Exercise 8.12). We now give an informal argument that although this is a consistent situation, Alice can never attain this state of knowledge! (We give a formal proof later.) Assume that at some point φ_{AB} holds. Then we can reason as follows:

> Suppose p is false. Then φ_{AB} implies that neither Alice nor Bob knows that p is false. But Alice could then receive a message from Bob saying "I (Bob) don't know p." Then, since Alice knows that either (a) p is true and Bob knows that p is true, or (b) p is false and Bob does not know that p is false, it follows that Alice would know that p must be false. But, by axiom DG2, it is impossible for Alice and Bob to learn a nontrivial fact about nature (in this case, that p is false) simply by communicating, if this fact was not distributed knowledge beforehand. So p must be true.

This argument shows that if φ_{AB} holds, then so does p. Now observe that if φ_{AB} holds at some point (r, m), then since φ_{AB} is a statement about Alice's knowledge, φ_{AB} holds at all points that Alice cannot distinguish from (r, m). Thus, p must hold at all points that Alice cannot distinguish from (r, m). But this means that $K_{Alice}p$ holds at (r, m), contradicting the assumption that φ_{AB} (which implies $\neg K_{Alice}p$) holds at (r, m). ∎

This informal argument shows that the formula $\neg\varphi_{AB}$, which is not a consequence of $S5_2$, is valid with respect to $\mathcal{I}_2^{oa}(\{p\})$. In fact, it is a consequence of a more general axiom.

Suppose Φ is a finite set of primitive propositions, say $\Phi = \{p_1, \ldots, p_k\}$. To state our new axiom, it is convenient to identify a truth assignment α to the primitive propositions in Φ with the formula $p_1' \wedge \ldots \wedge p_k'$, where p_i' is p_i if p_i is true under the truth assignment α, and p_i' is $\neg p_i$ if p_i is false under the truth assignment α. (The assumption that Φ is finite is crucial here; if Φ were infinite, then α could not be viewed as a formula in the language. This is the only place in the book that we find it necessary to assume that the set of primitive propositions is finite. We reconsider this assumption later in the section.) Consider the following axiom:

$\mathrm{OA}_{n,\Phi}$. $D\neg\alpha \Rightarrow (K_1\neg\alpha \vee \ldots \vee K_n\neg\alpha)$, where α is a truth assignment to the primitive propositions in Φ

Axiom $\mathrm{OA}_{n,\Phi}$ says that if it is distributed knowledge among all the agents that α does not describe the state of nature, then in fact one of the agents must already know that α does not describe the state of nature. Axiom $\mathrm{OA}_{n,\Phi}$ is *not* a consequence of $S5_n^D$ (Exercise 8.13). It describes a property of knowledge that does not hold in arbitrary systems; however, as we now show, it does hold in the system of observing agents.

Proposition 8.4.5 *$OA_{n,\Phi}$ is valid with respect to $\mathcal{I}_n^{oa}(\Phi)$.*

Proof Suppose that $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \neg K_1\neg\alpha \wedge \ldots \wedge \neg K_n\neg\alpha$, where α is a truth assignment to the primitive propositions in Φ. We must show that $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \neg D\neg\alpha$. Since the formulas $K_i\alpha$ are stable for $i = 1, \ldots, n$ (Exercise 4.18), we must

have that $(\mathcal{I}_n^{oa}(\Phi), r, 0) \models \neg K_1 \neg\alpha \wedge \ldots \wedge \neg K_n \neg\alpha$. Suppose $(\mathcal{T}_i, \langle\rangle)$ is the initial state of agent i in run r, for $i = 1, \ldots, n$. Then we must have $\alpha \in \mathcal{T}_i$ for each agent i. By definition, $\mathcal{G}_0$ includes the initial global state $s = ((\alpha, \langle\rangle), (\mathcal{T}_1, \langle\rangle), \ldots, (\mathcal{T}_n, \langle\rangle))$. By the inductive construction of $\mathcal{I}_n^{oa}(\Phi)$, there must be some run r' with $r'(0) = s$. Since all agents consider $(r', 0)$ possible at the point $(r, 0)$, we must have $(\mathcal{I}_n^{oa}(\Phi), r, 0) \models \neg D \neg\alpha$. The validity of DG2 implies that $(\mathcal{I}_n^{oa}(\Phi), r, m) \models \neg D \neg\alpha$, as desired. ∎

Again, both of the assumptions that are implicit in axiom $\mathrm{OA}_{n,\Phi}$, namely that α is a truth assignment and that we are considering the distributed knowledge of the group of all agents, are necessary. $\mathrm{OA}_{n,\Phi}$ is not valid if we replace the truth assignment α by an arbitrary propositional formula (Exercise 8.14(a)). Nor is it the case that $D_G \neg\alpha \Rightarrow \bigvee_{i \in G} K_i \neg\alpha$ is valid with respect to $\mathcal{I}_n^{oa}(\Phi)$ for an arbitrary subgroup G (Exercise 8.14(b)). Of course $D_G \neg\alpha \Rightarrow (K_1 \neg\alpha \vee \ldots \vee K_n \neg\alpha)$ is valid, since $D_G \neg\alpha \Rightarrow D \neg\alpha$ is valid for any subgroup G (Exercise 2.10(f)).

It is easy to check that $\neg\varphi_{AB}$ is provable in $S5_2^D + \{\mathrm{OA}_{2,\{p\}}\}$ (Exercise 8.15). This is the formal version of the argument given in Example 8.4.4.

Is $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$ complete with respect to $\mathcal{I}_n^{oa}(\Phi)$ for the language $\mathcal{L}_n^D(\Phi)$? That depends. If $n = 2$, so that there are exactly two agents, then it can be shown to be complete. However, if $n \geq 3$, it is not. For, assume that there are three agents, say, Alice, Bob, and Charlie, and p is a primitive proposition. Given the observing agents' protocol, it is impossible to arrive at a situation where Alice knows that Bob knows p and that Charlie does not know p: this is because Alice never knows whether Bob has just told Charlie that p is true. Thus, the formula

$$\varphi_{ABC} =_{\text{def}} K_{Alice} K_{Bob} p \wedge K_{Alice} \neg K_{Charlie} p$$

is false at every point. However, $\neg\varphi_{ABC}$ is not a consequence of $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$, where $p \in \Phi$ (Exercise 8.21). It is an open problem to characterize the formulas in $\mathcal{L}_n$ that are valid in the case $n \geq 3$. If we modify the assumptions of the system of communicating agents by requiring that each message be delivered in the same round it is sent (and in particular not allowing messages to be lost), then there are some further properties of knowledge (Exercise 8.16). This demonstrates once again how small changes in the underlying assumptions about communication can affect the properties of knowledge, even if we keep the general scenario fixed. Furthermore, under appropriate assumptions about the agents' behavior and the messages they can send, $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$ is indeed complete, even when $n \geq 3$. (See the notes at the end of the chapter for further details.)

We briefly consider one other seemingly subtle change—this time a syntactic change—that can affect the properties of knowledge. Up to now we have assumed that the language has a fixed, finite set Φ of primitive propositions. Suppose we allow Φ to be infinite. We can now construct the system $\mathcal{I}_n^{oa}(\Phi)$ as before, although it has the somewhat unpleasant property that there are uncountably many possible initial states (since there are uncountably many truth assignments). Alternatively, we can consider the class $\mathcal{C}_n^{oa}(\Phi)$ consisting of all interpreted systems $\mathcal{I}_n^{oa}(\Phi')$ such that Φ' is a finite subset of Φ. The first possibility corresponds to assuming that the agents really are communicating about an infinite set of propositions, while the second amounts to saying that the agents are communicating about a fixed finite set of propositions (some subset of Φ), but we do not know which it is. Are there any extra properties of knowledge in either case? Note that the extra axiom $OA_{n,\Phi}$ is no longer even a formula, since it is "infinitely long." It is still conceivable *a priori* that there could be other extra properties of knowledge. It turns out, however, that this does not happen in either case. More precisely, $S5_n^D$ is a sound and complete axiomatization with respect to both the interpreted system $\mathcal{I}_n^{oa}(\Phi)$ and the class $\mathcal{C}_n^{oa}(\Phi)$, of interpreted systems, if Φ is an infinite set of primitive propositions (Exercise 8.23). Thus, in this case, no additional axioms are needed; the need for an extra axiom depends crucially on the assumption that Φ is finite.

8.5 A Closer Look at Axiom $OA_{n,\Phi}$

It may seem somewhat surprising that knowledge in systems of observing agents has such a nonobvious property as that described by axiom $OA_{n,\Phi}$ (and by $\neg\varphi_{AB}$, which is a consequence in $S5_2^D + \{OA_{2,\{p\}}\}$, by Exercise 8.15). The reader may wonder at this point if axiom $OA_{n,\Phi}$ applies in other situations as well. If not, what is it about $\mathcal{I}_n^{oa}(\Phi)$ that makes this axiom valid? It turns out that the validity of $OA_{n,\Phi}$ depends on a number of subtle assumptions hidden in our construction of $\mathcal{I}_n^{oa}(\Phi)$. Before we examine the situation formally, we consider the following examples, all variants of Example 8.4.4 in the previous section, showing situations where the "impossible" formula φ_{AB} is attainable.

Example 8.5.1 Let p be the statement "The communication line between Alice and Bob is up." Suppose p is true and Alice sends Bob the message "Hello," which Bob receives (since, after all, the communication line is up). At this point, Bob knows p (since he received the message) and Alice doesn't know whether p is true or false (since she doesn't know whether Bob received her message). But Alice does know that either p is true and Bob knows that p is true, or else p is false and Bob doesn't

know that p is false (since if p is false, Bob will have no way of knowing whether he didn't receive a message because the line was down or because Alice didn't send one in the first place). Thus, we have a situation where φ_{AB} is attained. ∎

Why does φ_{AB} hold in this situation? Where does our soundness proof fail? In our construction of $\mathcal{I}_n^{oa}(\Phi)$, we implicitly assumed that the primitive propositions about which the agents are communicating cannot affect the communication process. By contrast, in this example, if p is false, then Alice and Bob cannot communicate. It is interesting to note that we also used this assumption implicitly to prove the validity of axiom DG2. DG2 fails here as well. Before communicating, there was no distributed knowledge among Alice and Bob about p, but after communicating, Bob knows that p is true.

Example 8.5.2 Another key assumption made in $\mathcal{I}_n^{oa}(\Phi)$ is that agents have perfect recall. To see the importance of this assumption, suppose we no longer require it. Now let p be the statement "Bob has perfect recall." Alice knows that if Bob knows p, then p is true, so Bob has perfect recall, and so Bob never forgets that he knows p. In addition, Alice knows that if Bob knows $\neg p$, then p is false, and so Bob does not have perfect recall, and so Bob might forget that he knows $\neg p$. Suppose in fact that p is true and Bob knows this, and Bob sends Alice two messages. The first one says "either I know p or I know $\neg p$" (i.e., $K_{Bob}p \vee K_{Bob}\neg p$), and the second says "I don't know $\neg p$" (i.e., $\neg K_{Bob}\neg p$). At this point, Alice knows that either p is true and Bob knows that p is true, or that p is false and Bob doesn't know that p is false (Bob knew that p was false and then forgot). Again, we have a situation where the "unattainable" state of knowledge is attained! ∎

Example 8.5.3 We have already mentioned that axiom $\mathrm{OA}_{n,\Phi}$ does not hold if we replace the truth assignment α by an arbitrary propositional formula, nor does it hold if we consider an arbitrary subgroup G rather than all the agents in the system (Exercise 8.14). It is not hard to modify Example 8.4.4 by adding more agents to the system or by using extra propositions to characterize nature in order to get a situation where φ_{AB} holds. For example, suppose that rather than assuming that nature is completely characterized by one primitive proposition p, we assume that nature is characterized both by whether it is rainy or dry and whether the temperature is cold or warm. Thus, there are *two* primitive propositions p and q, where p is "It rained in San Francisco on January 1," and q is "It was cold in San Francisco on January 1." Assume that Alice knows that either (a) it was rainy and cold (that is, p and q are both true), or else (b) it was dry and warm (that is, p and q are both false). Assume that Bob knows that it was rainy and cold (that is, he knows that p and q are both

true). Assume that Bob tells Alice that either (a′) he knows that it was rainy and cold (which is the actual situation), or else (b′) he knows that it was warm but doesn't know whether it was rainy (that is, he knows that q is false but doesn't know whether or not p is true). After Alice receives this information from Bob, she still doesn't know whether it was rainy or dry. She knows that if it was rainy, then case (a) occurred, so it was cold, hence case (b′) is impossible, so case (a′) occurred, and Bob knows that it was rainy. Thus, Alice knows that if p is true (it was rainy), then Bob knows that p is true. Furthermore, she knows that if it was dry, then case (b) occurred, so it was warm, hence case (a′) is impossible, so case (b′) occurred, and Bob doesn't know whether it was rainy. Thus, Alice knows that if p is false (it was dry), then Bob doesn't know that p is false. So, yet again, we have attained the "unattainable" state of knowledge. We leave it to the reader to construct a variant of the system with a third agent in the picture where, again, φ_{AB} holds (Exercise 8.18). ∎

These examples illustrate the subtleties in modeling a real-world situation. Small changes in the model can result in significant changes to the conclusions one can draw from the model. Under seemingly small perturbations of the assumptions underlying the system, we can go from a situation where $OA_{n,\Phi}$ is valid to one where it is not. Our goal in the remainder of this section is to isolate the key features of the system of observing agents that cause axiom $OA_{n,\Phi}$ to be valid. The idea will be to consider a number of abstract properties of systems, motivated by properties that hold in $\mathcal{I}_n^{oa}(\Phi)$, and show that $OA_{n,\Phi}$ is valid in all systems satisfying these abstract properties.

As we showed in Example 8.5.2, one key property of $\mathcal{I}_n^{oa}(\Phi)$ that we need is that it is a system where agents have perfect recall. What else is necessary?

It was clearly important in the discussion of the observing agents that the primitive propositions talked about the initial state of nature (so that their truth values did not change over time), and in fact completely characterized the initial state of nature. More formally, we say that *the primitive propositions characterize the initial environment* in an interpreted system if the truth values of the primitive propositions remain the same throughout each run, and the primitive propositions have the same truth values in two runs r and r' precisely when the initial environment state is identical in r and r'. It is easy to see that $\mathcal{I}_n^{oa}(\Phi)$ is a system where the primitive propositions characterize the initial environment.

The next key feature of the system of observing agents that we want to capture is that the agents start by making independent observations of nature. This "independence" intuitively means that what one agent observes does not affect what any of the other agents observe (although, of course, the observations do have to be consistent, so that we cannot have one agent observing p and another observing $\neg p$).

Since "nature" is encoded in the environment's state, this amounts to requiring that an agent's initial state depend only on the initial state of the environment. More precisely, suppose s_e is an initial environment state and it is possible for agent i to have initial state s_i when the initial state of the environment is s_e, for $i = 1, \ldots, n$. That is, assume that, for each agent i, there is a run r^i such that in the initial global state $r^i(0)$, the environment state is s_e and agent i's local state is s_i. Intuitively, this means there is some observation that agent i can make that puts it into state s_i when the environment state is s_e. We then require that each of the agents can simultaneously make the observation that puts them into them into this local state when the environment state is s_e. That is, we require that there be a run r^* such that $r^*(0) = (s_e, s_1, \ldots, s_n)$. If this requirement holds for every possible initial state s_e of the environment, then we say that *the environment determines the initial states*. It is easy to see that $\mathcal{I}_n^{oa}(\Phi)$ is a system where the environment determines the initial states.

As we can see from Example 8.5.1, to guarantee that $\neg\varphi_{AB}$ is valid, it is not enough that the agents have perfect recall, the primitive propositions characterize the initial environment, and the environment determines the initial states. Somehow we need to assume that we do not have primitive propositions corresponding to events such as "the communication line is up," whose truth may affect the transitions of the system. Formally, we say that *agent state transitions are independent of the initial environment* in a system if whenever there are two runs r and r' such that each agent has the same initial state in r as in r', then there is a run r'' with the same initial global state as in r', and where for each time m, each agent has the same state at time m in r'' as in r. Intuitively, r'' has the same initial state as r' (both for the agents and for the environment), and every agent makes the same transitions at corresponding steps of r and r''. As shown in Exercise 8.11, $\mathcal{I}_n^{oa}(\Phi)$ is a system where agent state transitions are independent of the initial environment. However, in Example 8.5.1, where the environment could describe whether the communication line between Alice and Bob is up, it is *not* the case that agent state transitions are independent of the initial environment. We can have two runs r and r' such that Alice and Bob have the same initial state in both (intuitively, they are ignorant as to whether the communication line is up), the communication line is up in r, and down in r'. Moreover, suppose Bob sends Alice a message in r, which Alice receives (since the line is up). The receipt of the message is encoded in Alice's state in $(r, 1)$. Suppose that the state of the line is part of the environment state. There can be no run r'' with the same initial global state as in r' (so that the communication line is down), but where Alice receives a message at time 1 (as she does in r).

The following theorem shows that the four conditions we have abstracted out from the system of observing agents are what cause axiom $\mathrm{OA}_{n,\Phi}$ to hold. Let $\mathcal{C}_n^{oa}$

be the subclass of $\mathcal{C}_n$ consisting of all interpreted systems for n agents that satisfy these four conditions: (a) agents have perfect recall, (b) the primitive propositions characterize the initial environment, (c) the environment determines the initial states, and (d) agent state transitions are independent of the initial environment.

Theorem 8.5.4 $S5_n^D + \{OA_{n,\Phi}\}$ *is a sound and complete axiomatization for the language* $\mathcal{L}_n^D$ *with respect to* $\mathcal{C}_n^{oa}$.

Proof See Exercises 8.19 and 8.20. ∎

Interestingly, all four conditions that characterize $\mathcal{C}_n^{oa}$ are crucial. If any one of them is dropped, then $OA_{n,\Phi}$ is not sound, and $S5_n^D$ alone remains a sound and complete axiomatization (Exercise 8.22).

Exercises

8.1 Prove parts (a)–(d) of Theorem 8.1.1, and fill in the details of the sketch of the proof provided for part (e).

* **8.2** Fill in the missing details of the proof of Theorem 8.1.2. In particular, show the following:

(a) The $\mathcal{K}_i$ and $\mathcal{T}$ relations have the three properties claimed of them. (Hint: for property (3), define φ_V, as in the proof of Theorem 3.3.1, to be the conjunction of the formulas in V. Define φ' to be the disjunction of all of the formulas φ_V such that the formula $\psi_1 U \psi_2$ is in V but there is no sequence as promised in part (3). Now apply rule RT2, where the roles of φ and ψ in RT2 are played by ψ_1 and ψ_2, respectively.)

(b) There is an acceptable sequence going through every state in S.

(c) If $\psi \in Sub^+(\varphi)$ and A is the acceptable sequence $(s_{V_0}, s_{V_1}, \ldots)$, then $(\mathcal{I}, r^A, m) \models \psi$ iff $\psi \in V_m$.

Point out what changes would have to be made to deal with the language $\mathcal{L}_n^{CU}$.

* **8.3** Fill in the missing details of the proof of Theorem 8.2.1. In particular, prove the two observations stated in the text and do the induction argument showing that if A is the acceptable sequence $(s_{V_0}, s_{V_1}, \ldots)$, then $(\mathcal{I}, r^A, m) \models \psi$ iff $\psi \in V_m$, for all formulas $\psi \in Sub^+(\varphi)$.

8.4 Show that a formula ψ is stable in an interpreted system $\mathcal{I}$ precisely if it is equivalent in $\mathcal{I}$ to a formula of the form $\Box\varphi$ for some formula φ.

8.5 In this exercise, we consider the validity of axioms KT1 and KT2.

(a) Show that axiom KT1 is valid with respect to $\mathcal{C}_n^{pr}$.

(b) Show that axiom KT2 is valid with respect to $\mathcal{C}_n^{pr,sync}$.

(c) Show that KT2 is *not* valid with respect to $\mathcal{C}_n^{pr}$. Note that this means that KT2 is not valid with respect to $\mathcal{C}_n$ either.

(d) Show that KT1 is not valid with respect to $\mathcal{C}_n$.

8.6 Show that axiom KT1 can be deduced from $\mathrm{S5}_n^U + \{\mathrm{KT2}\}$.

* **8.7** Consider the following axiom:

KT3. $\big(K_i\varphi_1 \wedge K_i\neg((K_i\varphi_1 \vee K_i\varphi_3)U\neg\varphi_2)\big) \Rightarrow \bigcirc(K_i\varphi_3 \Rightarrow K_i\varphi_2)$.

(a) Show that KT3 is valid in $\mathcal{C}_n^{pr}$.

(b) Show that KT1 is provable from $\mathrm{S5}_n^U + \{KT3\}$.

(c) Show that KT3 is not provable from $\mathrm{S5}_n^U + \{KT1\}$. (Hint: consider "nonstandard" runs where time ranges over the integers rather than the natural numbers. Construct a system consisting of two nonstandard runs where all the axioms of $\mathrm{S5}_n^U + \{KT1\}$ are valid, but KT3 is not.)

This shows that $\mathrm{S5}_n^U + \{KT1\}$ is not a complete axiomatization for $\mathcal{C}_n^{pr}$. A complete axiomatization can be obtained, but it requires an adding an axiom even more complicated that KT3. (See the notes for further discussion.)

* **8.8** In this exercise, we consider properties of $\mathcal{C}_n^{amp}$. Show that the following formulas are all valid with respect to $\mathcal{C}_n^{amp}$:

(a) $K_i \bigcirc K_i\varphi \Rightarrow K_i\varphi$,

(b) $K_i\Diamond K_i\varphi \Rightarrow K_i\varphi$,

(c) $K_i\Diamond K_j\varphi \Rightarrow K_iK_j\varphi$,

(d) $\bigcirc^{m-1} K_{i_1} \ldots K_{i_m} \varphi \Rightarrow K_{i_m} \varphi$, where $i_j \neq i_{j+1}$ for $j = 1, \ldots, m-1$,

(e) $\neg K_i K_j \varphi_1 \wedge K_i K_k \varphi_2 \Rightarrow \bigcirc(\neg K_i K_j \varphi_1 \vee K_i K_k \varphi_2)$ if $i \neq j$ and $i \neq k$. (Hint: use Exercise 4.31.)

Show that none of these formulas are provable in $\mathrm{S5}_n^U + \{KT1\}$.

8.9 Show that the fact that common knowledge cannot be lost is provable from the fact that common knowledge cannot be gained. More precisely, show

$$\mathrm{S5}_n^{CU} + \{\mathrm{CG}\} \vdash C_G \psi \Rightarrow \Box C_G \psi, \quad \text{if } |G| \geq 2.$$

(Hint: use the results of Exercise 3.11(d).)

* **8.10** Show that axiom CG characterizes systems where the initial point in a run is reachable from all later points. More precisely, let $\mathcal{C}_n^{reach}$ consist of all interpreted systems $\mathcal{I}$ of n agents where for all subgroups G with $|G| \geq 2$ and all points (r, m) in $\mathcal{I}$, we have that $(r, 0)$ is G-reachable from (r, m). Show that $\mathrm{S5}_n^{CU} + \{\mathrm{CG}\}$ is a sound and complete axiomatization for the language $\mathcal{L}_n^{CU}$ with respect to $\mathcal{C}_n^{reach}$. (Hint: extend the techniques used to prove Theorem 8.1.2.)

8.11 Fill in the details of the inductive proof showing that the run r' in the proof of Theorem 8.4.1 is indeed a run in $\mathcal{I}_n^{oa}(\Phi)$. That is, prove that if r is a run in $\mathcal{I}_n^{oa}(\Phi)$ such that $r(0) = ((\alpha, \langle\rangle), (\mathcal{T}_1, \langle\rangle), \ldots, (\mathcal{T}_n, \langle\rangle))$ and $\alpha' \in \bigcap_{i=1,\ldots,n} \mathcal{T}_i$, then the run r' which is identical to r except that the truth assignment at every point of r' is α' rather than α, is also a run in $\mathcal{I}_n^{oa}(\Phi)$. In the language of Section 8.5, this says that agent state transitions are independent of the initial environment in $\mathcal{I}_n^{oa}(\Phi)$.

8.12 Show that the formula φ_{AB} from Example 8.4.4 is consistent with $\mathrm{S5}_2$, by constructing a Kripke structure $M \in \mathcal{M}_2^{rst}$ such that $(M, s) \models \varphi_{AB}$ for some state s in M.

8.13 Show that axiom $\mathrm{OA}_{n,\Phi}$ is not a consequence of $\mathrm{S5}_n^D$ by constructing a Kripke structure $M \in \mathcal{M}_n^{rst}$ and a state s of M such that $\mathrm{OA}_{n,\Phi}$ is false at (M, s).

8.14 In this exercise, we show that the assumptions in axiom $\mathrm{OA}_{n,\Phi}$ are necessary.

(a) Show that the formula that results by allowing α in $\mathrm{OA}_{n,\Phi}$ to be a primitive proposition p, rather than a truth assignment, is false at some point if there are at least two primitive propositions.

(b) Show that $D_G\alpha \Rightarrow \bigvee_{i\in G} K_i\neg\alpha$ is not valid with respect to $\mathcal{I}_n^{oa}(\Phi)$ if G is a proper subset of the agents, even if α is a truth assignment.

8.15 Show that $\mathrm{S5}_2^D + \{\mathrm{OA}_{2,\{p\}}\} \vdash \neg\varphi_{AB}$.

* **8.16** Suppose that we modify the system of observing agents by assuming that immediate message delivery is guaranteed (that is, every message is delivered in the same round it was sent). Assume that there are only two agents. Let us say that two points (r, m) and (r', m') are *equivalent* if they satisfy precisely the same formulas in $\mathcal{L}_n^{CD}(\Phi)$.

(a) If S is a set of truth assignments, then let τ_S be the formula

$$\left(\bigwedge_{\alpha\notin S} K_1\neg\alpha\right) \wedge \left(\bigwedge_{\alpha\in S} \neg K_1\neg\alpha\right).$$

Intuitively, τ_S says that according to agent 1's knowledge, precisely the truth assignments in S are possible. Let (r, m) be a point. Show that there is a run r' in which agent 1 sends a message in round 1 to agent 2 of the form $\tau_{S_1} \vee \ldots \vee \tau_{S_x}$, and in which agent 2 sends a similar message to agent 1 in round 1, such that (r, m) and $(r', 1)$ are equivalent. Thus, to decide whether a formula is satisfiable, we can restrict attention to runs where each agent sends one message in round 1 and no other messages afterwards. (Hint: the only information that agent 2 can learn about agent 1 is what agent 1's initial information was; this can be done by considering which possible formulas τ_S about agent 1's initial information are consistent with the messages that agent 1 sent.)

(b) Show that there are only a finite number of distinct equivalence classes of points. That is, show that there is a finite set P of points such that every point is equivalent to a point in P.

(c) Show that only finitely many of the following formulas are satisfiable:

$K_1\alpha \wedge \neg K_1K_2\alpha$

$K_1K_2\alpha \wedge \neg K_1K_2K_1\alpha$

$K_1K_2K_1\alpha \wedge \neg K_1K_2K_1K_2\alpha$

...

(Hint: show that no two of these formulas are mutually satisfiable, that is, there is no point where two of these formulas can both be true. Then use part (b).)

(d) Show that there are new axioms when there are only two agents and immediate message delivery is guaranteed. (Hint: use part (c), along with the fact that each of the formulas in part (c) is satisfiable when immediate message delivery is not guaranteed.)

(e) Show that if α is a truth assignment, then the formula

$$(K_1 K_2 \alpha \wedge K_2 K_1 \alpha) \Rightarrow K_1 K_2 K_1 \alpha$$

is valid. (Hint: by part (a), we need only show that this formula is true at every point $(r', 1)$, where in run r', agent 1 sends agent 2 a message of the form $\tau_{S_1} \vee \ldots \vee \tau_{S_s}$ in round 1, agent 2 sends agent 1 a message of the form $\tau_{T_1} \vee \ldots \vee \tau_{T_t}$ in round 1, and no other messages are sent. Assume that τ_{S_1} and τ_{T_1} hold before any messages are sent. Thus, agent 1's initial knowledge before any communication takes place is that precisely the truth assignments in S_1 are possible; similarly, agent 2's initial knowledge before any communication takes place is that precisely the truth assignments in T_1 are possible. Show that $K_1 K_2 \alpha$ holds after round 1 iff for each T_i that intersects S_1, and for each S_j that intersects T_i, we have $S_j \cap T_i = \{\alpha\}$. Show that the analogous statement characterizes when $K_2 K_1 \alpha$ holds after round 1. Show that $K_1 K_2 K_1 \alpha$ holds after round 1 iff for each T_i that intersects S_1, and for each S_j that intersects T_i, and for each T_k that intersects S_j, we have $T_k \cap S_j = \{\alpha\}$.)

(f) Show, however, that if α is a truth assignment, then the formula

$$K_1 K_2 \alpha \Rightarrow K_2 K_1 \alpha$$

is *not* valid.

(g) Refine the result of part (c) by showing that of the infinite set of formulas in the list in part (c), only $K_1 \alpha \wedge \neg K_1 K_2 \alpha$ and $K_1 K_2 \alpha \wedge \neg K_1 K_2 K_1 \alpha$ are satisfiable.

(h) Show that if α is a truth assignment, then the formula

$$(K_1 K_2 \alpha \wedge K_2 K_1 \alpha) \Rightarrow C_{\{1,2\}} \alpha$$

is valid.

8.17 This exercise presents an alternative to $\mathrm{OA}_{n,\Phi}$ for the language $\mathcal{L}_n$ without distributed knowledge.

(a) As in Exercise 3.33, define a *pure knowledge formula* to be a Boolean combination of formulas of the form $K_i\varphi$, where φ is arbitrary. Consider the following axiom, where α is a truth assignment and β is a pure knowledge formula:

$$\mathrm{OA}'_{n,\Phi}.\ \ \beta \wedge K_i(\beta \Rightarrow \neg\alpha) \Rightarrow (K_1\neg\alpha \vee \ldots \vee K_n\neg\alpha)$$

Since β is a pure knowledge formula, it is a fact about the state of knowledge of the agents. So axiom $\mathrm{OA}'_{n,\Phi}$ says that if a fact about the state of knowledge of the agents holds, and if some agent knows that this state is incompatible with the truth assignment α, then some agent knows that the truth assignment α is impossible. Show that $\mathrm{OA}'_{n,\Phi}$ is a consequence of $\mathrm{S5}^D_n + \{\mathrm{OA}_{n,\Phi}\}$. (Hint: use the fact, proved in Exercise 3.33, that if β is a pure knowledge formula, then $\beta \Rightarrow D\beta$ is provable in $\mathrm{S5}^D_n$.)

(b) Assume that p is the only primitive proposition. Show that $\neg\varphi_{AB}$ is a consequence of $\mathrm{S5}_2 + \{\mathrm{OA}'_{2,\Phi}\}$, where agents 1 and 2 are Alice and Bob. (Hint: let ψ be the formula obtained from axiom $\mathrm{OA}'_{2,\Phi}$ by letting i be Alice, letting α be p, and letting β be $\neg K_{Bob}p \wedge \neg K_{Bob}\neg p$. Then show that $\neg\varphi_{AB}$ can be proved from $\mathrm{S5}_2 + \{\psi\}$.)

It can be shown that if the class of messages is sufficiently rich, then $\mathrm{S5}_n + \{\mathrm{OA}'_{n,\Phi}\}$ is a sound and complete axiomatization for the language $\mathcal{L}_n$ with respect to $\mathcal{I}^{oa}_n(\Phi)$.

8.18 This exercise presents yet another "counterexample" to the argument in Example 8.4.4. Show that if there is another agent besides Alice and Bob, then φ_{AB} is attainable in the system of observing agents. Where does the argument of Example 8.4.4 break down in this case?

* **8.19** In this exercise, and the next, we provide details of the proof of Theorem 8.5.4. This exercise proves soundness; the next proves completeness. Thus, in this exercise, we show that the four conditions that we have abstracted out from the system of observing agents are sufficient to cause every instance of axiom $\mathrm{OA}_{n,\Phi}$ to hold. We begin by considering a certain condition on Kripke structures that essentially characterizes $\mathrm{OA}_{n,\Phi}$. Let us say that a Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ satisfies the *pasting condition* if whenever

1. $s_1, \ldots, s_n, t' \in S$,
2. $\pi(s_i) = \alpha$, for $i = 1, \ldots, n$, and
3. $(t', s_i) \in \mathcal{K}_i$, for $i = 1, \ldots, n$,

then there exists $t \in S$ such that $\pi(t) = \alpha$ and $(t, t') \in \mathcal{K}_i$ for $i = 1, \ldots, n$.

This condition is called the "pasting condition," since it says that if certain states are in S, then another state, that is the result of "pasting together" information from these states, is also in S. Not surprisingly, the pasting condition implies certain properties of knowledge (as we see in part (b)).

(a) Let $\mathcal{I}$ be an interpreted system in $\mathcal{C}_n^{oa}$. Prove that the Kripke structure $M_{\mathcal{I}}$ corresponding to $\mathcal{I}$ satisfies the pasting condition. (Hint: assume that the three antecedents of the pasting condition hold for $M_{\mathcal{I}} = (S, \pi, \sim_1, \ldots, \sim_n)$. Assume that $s_i = r_i(m_i)$, for $i = 1, \ldots, n$, and $t' = r'(m')$. Show that the environment components at the points $(r_i, 0)$, for $i = 1, \ldots, n$, are all the same; call this common environment component s_e. Show that there is a run r'' where $r''(0)$ has environment component s_e and $r''(0) \sim_i r'(0)$ for all i. Show that there is a run r with $r(0) = r''(0)$ such that $r(m') \sim_i r'(m') = t'$ for all i. Show that $t = r(m')$ satisfies the consequent of the pasting condition.)

(b) Prove that the pasting condition holds for a Kripke structure $M \in \mathcal{M}_n^{rst}(\Phi)$ iff every instance of $\mathrm{OA}_{n,\Phi}$ (in the language $\mathcal{L}_n^D(\Phi)$) is valid in M. Thus, in a precise sense, the pasting condition corresponds to axiom $\mathrm{OA}_{n,\Phi}$.

(c) Show that $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$ is sound with respect to $\mathcal{C}_n^{oa}$.

* **8.20** Show that $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$ is complete with respect to $\mathcal{C}_n^{oa}$. (Hint: assume that φ is consistent with $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$. Show that φ is satisfiable in a Kripke structure $M \in \mathcal{M}_n^{rst}(\Phi)$ where every instance of $\mathrm{OA}_{n,\Phi}$ is valid in M. (Use an argument similar to that in the proof of Theorem 3.4.1 and, in particular, in Exercise 3.30, but where every maximal consistent set that is used is consistent with respect to not just $\mathrm{S5}_n^D$ but also $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$.) Use Exercise 8.19(b) to show that M satisfies the pasting condition of Exercise 8.19. Write $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$. Define a subset $S' \subseteq S$ to be a *connected component* if no state $t \notin S'$ is reachable from a state $s \in S'$ and every state in S' is reachable from every other state in S'. We say that a Kripke structure is *connected* if its state space is a connected component. Since φ is satisfiable in a Kripke structure satisfying the pasting condition, it is satisfiable in a connected Kripke structure satisfying the pasting condition. Now show that for each

connected Kripke structure $M' \in \mathcal{M}_n^{rst}$ satisfying the pasting condition, there is an interpreted system $\mathcal{I} \in \mathcal{C}_n^{oa}$ such that some connected component of $M_{\mathcal{I}}$ is identical to M'. Therefore, φ is satisfiable in a member of $\mathcal{C}_n^{oa}$.)

8.21 Show that $\neg\varphi_{ABC}$ is not a consequence of $\mathrm{S5}_n^D + \{\mathrm{OA}_{n,\Phi}\}$, where $n \geq 3$, $p \in \Phi$, and φ_{ABC} is the formula $K_{Alice}K_{Bob}p \wedge K_{Alice}\neg K_{Charlie}p$. (Hint: construct a Kripke structure $M \in \mathcal{M}_n^{rst}$ and a state s of M such that M satisfies the pasting condition of Exercise 8.19, and $(M, s) \models \varphi_{ABC}$.)

* **8.22** Show that if any of the four conditions in the definition of $\mathcal{C}_n^{oa}$ are dropped, then $\mathrm{S5}_n^D$ is a sound and complete axiomatization for the language $\mathcal{L}_n^D$ with respect to the resulting class of systems. (Hint: let M be a connected Kripke structure, as defined in Exercise 8.20. For each proper subset $\mathcal{A}$ of the four conditions, construct an interpreted system $\mathcal{I}$ that satisfies the conditions in $\mathcal{A}$ such that some connected component of $M_{\mathcal{I}}$ is identical to M. So if φ is consistent with $\mathrm{S5}_n^D$, then φ is satisfiable in an interpreted system that satisfies the conditions in $\mathcal{A}$.)

* **8.23** This exercise deals with the situation where the set Φ of primitive propositions is infinite, as discussed at the end of Section 8.4.

(a) Show that $\mathrm{S5}_n^D$ is a sound and complete axiomatization with respect to the interpreted system $\mathcal{I}_n^{oa}(\Phi)$. (Hint: given a formula φ that is consistent with $\mathrm{S5}_n^D$, show that there must be a Kripke structure $M \in \mathcal{M}_n^{rst}(\Phi')$ satisfying φ, where Φ' consists of all the finitely many primitive propositions that appear in φ. Show that there is a Kripke structure $M' \in \mathcal{M}_n^{rst}(\Phi)$ satisfying the pasting condition of Exercise 8.19, that is identical to M when restricted to Φ'. It is then easy to show that φ is satisfiable in M'. The result now follows as in the conclusion of the hint in Exercise 8.20.)

(b) Show that $\mathrm{S5}_n^D$ is a sound and complete axiomatization with respect to the class $\mathcal{C}_n^{oa}(\Phi)$ of interpreted systems. (Hint: the proof is similar to that of part (a).)

Notes

The axiom system for time alone is a variation of that given by Gabbay, Pnueli, Shelah, and Stavi [1980] (where the completeness result was first proved), along the lines of the axiom system for branching time given by Emerson and Halpern [1985].

Axiom KT1 is due to Ladner and Reif [1986], while KT2 is due to Lehmann [1984]. The completeness of $S5_n^U$ with respect to $\mathcal{C}_n$ and $\mathcal{C}_n^{sync}$, as well as completeness of $S5_n^U + \{KT2\}$, is due to Halpern and Vardi [1986]. The *PSPACE*-completeness of the logic with temporal operators only was proved by Halpern and Reif [1983] and by Sistla and Clarke [1985]. All of the complexity results in Table 8.1 are due to Halpern and Vardi [1989]. Further details are given in their papers [1988a, 1988b]; see also [Spaan 1990]. The exponential-time complexity of the validity problem for common knowledge and time with respect to $\mathcal{C}_n$ is also proved by Fischer and Immerman [1987]. Halpern and Vardi [1989] extend Table 8.1 to cover a number of other conditions, including the case where branching time operators are used.

Ron van der Meyden pointed out to us that axiom KT3 from Exercise 8.7 is not provable in $S5_n^U + \{KT1\}$; he has recently provided a sound and complete axiomatization for $\mathcal{C}_n^{pr}$ [Meyden 1994]. In Section 8.2, we saw that in the class of all systems and in the class of synchronous systems there is no interaction between knowledge, common knowledge, and time. We can similarly show that in these classes there is no interaction between knowledge, distributed knowledge, and time. More precisely, if we let $S5_n^{DU}$ be the axiom system that results from adding T1, T2, T3, RT1, and RT2 to $S5_n^D$, and we define the language $\mathcal{L}_n^{DU}$ similarly to before, then we get analogues to Theorems 8.1.2 and 8.2.1 for distributed knowledge, namely, that $S5_n^{DU}$ is a sound and complete axiomatization for the language $\mathcal{L}_n^{DU}$ with respect to both $\mathcal{C}_n$ and $\mathcal{C}_n^{sync}$.

The properties of knowledge and time for $\mathcal{C}_n^{amp}$ discussed in Exercise 8.8 were also pointed out to us by Ron van der Meyden. At this point, we do not even have a candidate for a sound and complete axiomatization of $\mathcal{C}_n^{amp}$.

Example 8.4.4 and the results of Section 8.4 are by Fagin and Vardi [1986]. They introduce the axioms $OA_{n,\Phi}$ and $OA'_{n,\Phi}$ (of Exercise 8.17). They also consider a situation where the agents can send messages that are promises about the future (such as "I will not send a message in round 17"). They then modify the system of observing agents by requiring that agents fulfill all such promises. Under these circumstances, they show that $S5_n^D + \{OA_{n,\Phi}\}$ (respectively, $S5_n + \{OA'_{n,\Phi}\}$) is a sound and complete axiomatization for $\mathcal{L}_n^D(\Phi)$ (respectively, $\mathcal{L}_n(\Phi)$), with respect to the resulting interpreted systems. The material in Section 8.5 (along with Exercise 8.23) is taken from [Fagin, Halpern, and Vardi 1992a]; this paper also considers other conditions on interpreted systems, and provides sound and complete axiomatizations for interpreted systems that satisfy various combinations of these conditions.

Chapter 9

Logical Omniscience

The animal knows, of course. But it certainly does not know that it knows.

Teilhard de Chardin

A person who knows anything, by that very fact knows that he knows and knows that he knows that he knows, and so on ad infinitum.

Baruch Spinoza, *Ethics*, II, Prop. 21, 1677

Throughout this book we have found the possible-worlds model to be a very useful tool. As we already saw in Chapter 2, however, the possible-worlds model gives rise to a notion of knowledge that seems to require that agents be very powerful reasoners, since they know all consequences of their knowledge and, in particular, they know all tautologies. Thus, the agents could be described as *logically omniscient*. This does not especially trouble us in the context of multi-agent systems, since in that context we view our notion of knowledge as *external*. That is, knowledge is *ascribed* by the system designer to the agents. There is no notion of the agents computing their knowledge, and no requirement that the agents be able to answer questions based on their knowledge.

Nevertheless, there are situations where the assumption of logical omniscience seems completely inappropriate. Perhaps the most obvious example occurs when we consider human reasoning. People are simply not logically omniscient; a person can know a set of facts without knowing all of the logical consequences of this set of facts. For example, a person can know the rules of chess without knowing whether or not White has a winning strategy.

Lack of logical omniscience may stem from many sources. One obvious source is lack of computational power; for example, an agent simply may not have the computational resources to compute whether White has a winning strategy in chess. But there are other causes of lack of logical omniscience that are quite independent of computational power. For example, people may do faulty reasoning or refuse to acknowledge some of the logical consequences of what they know, even in cases where they do not lack the computational resources to compute those logical consequences. Our goal in this chapter is to develop formal models of knowledge that do not suffer from the logical-omniscience problem to the same extent that the standard possible-worlds approach does. We consider a number of approaches that are appropriate to capture different sources of the lack of logical omniscience. In Chapter 10, we describe a computational model of knowledge, which addresses issues such as capturing what a resource-bounded agent knows. As we shall see, that model also addresses the logical-omniscience problem in a way that is closely related to some of the approaches described in this chapter.

9.1 Logical Omniscience

A careful examination of the logical-omniscience problem must begin with the notion of logical omniscience itself. Exactly what do we mean by logical omniscience?

Underlying the notion of logical omniscience is the notion of *logical implication* (or *logical consequence*). Roughly speaking, a formula ψ logically implies the formula φ if φ holds whenever ψ holds; a set Ψ of formulas logically implies the formula φ if φ holds whenever all members of Ψ hold. Clearly, a formula is valid precisely if it is a logical consequence of the empty set of formulas. Like validity, logical implication is not an absolute notion, but is relative to a class of structures (and to a notion of truth, or satisfaction). Thus, more formally, we say that Ψ *logically implies* φ *with respect to a class* $\mathcal{M}$ *of structures* if, for all $M \in \mathcal{M}$ and all states s in M, whenever $(M, s) \models \psi$ for every $\psi \in \Psi$, then $(M, s) \models \varphi$. If Ψ is finite, it follows from Theorem 3.1.3 of Chapter 3 that Ψ logically implies φ with respect to $\mathcal{M}_n$ if and only if the formula $(\bigwedge \Psi) \Rightarrow \varphi$ is provable in K_n, where $\bigwedge \Psi$ is the conjunction of all formulas in Ψ. Thus, in the standard modal logic studied so far, logical and material implication coincide (where φ *materially implies* ψ if the formula $\varphi \Rightarrow \psi$ is valid). As we shall see, logical and material implication do not coincide in some of the logics considered in this chapter.

We can define *logical equivalence* in terms of logical implication. Two formulas are logically equivalent if each logically implies the other. Thus, if two formulas are

logically equivalent, then one is true precisely when the other is. Note that logical equivalence is also a relative notion, just as logical implication is.

Logical omniscience can be viewed as a certain closure property of an agent's knowledge; it says that if an agent knows certain facts and if certain conditions hold, then the agent must also know some other facts. The term logical omniscience actually refers to a family of related closure conditions. As we shall see, some of our models will eliminate certain strong forms of logical omniscience, but not necessarily all of its forms. The strongest form is what we call full logical omniscience.

- An agent is *fully logically omniscient* with respect to a class $\mathcal{M}$ of structures if, whenever he knows all of the formulas in a set Ψ, and Ψ logically implies the formula φ with respect to $\mathcal{M}$, then the agent also knows φ.

It is easy to see that we have full logical omniscience with respect to $\mathcal{M}_n$ (see Exercise 9.1). Full logical omniscience encompasses several weaker forms of omniscience. All of these notions also depend on the class of structures under consideration; we do not state that dependence here, to avoid clutter.

- *Knowledge of valid formulas*: if φ is valid, then agent i knows φ.
- *Closure under logical implication*: if agent i knows φ and if φ logically implies ψ, then agent i knows ψ.
- *Closure under logical equivalence*: if agent i knows φ and if φ and ψ are logically equivalent, then agent i knows ψ.

Because all of the above are special cases of full logical omniscience, they automatically hold for $\mathcal{M}_n$. As we shall see, however, it is possible to define classes of structures (and notions of truth) for which we do not have full logical omniscience, but do have some of these weaker notions.

There are also forms of omniscience that do not necessarily follow from full logical omniscience:

- *Closure under material implication*: if agent i knows φ and if agent i knows $\varphi \Rightarrow \psi$, then agent i also knows ψ.
- *Closure under valid implication*: if agent i knows φ and if $\varphi \Rightarrow \psi$ is valid, then agent i knows ψ.
- *Closure under conjunction*: if agent i knows both φ and ψ, then agent i knows $\varphi \wedge \psi$.

All these forms of logical omniscience in fact do hold for $\mathcal{M}_n$. Indeed, if $\{\varphi, \varphi \Rightarrow \psi\}$ logically implies ψ, as is it does with respect to $\mathcal{M}_n$, and we give $\Rightarrow$ its standard interpretation, then closure under material implication is a special case of full logical omniscience. One of the approaches described in this chapter is based on a *nonstandard* propositional semantics. In this approach, $\Rightarrow$ does not get its standard interpretation. So logical and material implication do not coincide; we get full logical omniscience but not closure under material implication. Similarly, closure under valid implication is a special case of full logical omniscience (and in fact is equivalent to closure under logical implication) if $\varphi \Rightarrow \psi$ is valid precisely when φ logically implies ψ. Again, while this is true under standard propositional semantics, it is not true in our nonstandard approach. Finally, closure under conjunction is a special case of full logical omniscience if the set $\{\varphi, \psi\}$ logically implies $\varphi \wedge \psi$. This indeed will be the case in all the logics we consider, including the nonstandard one. It is clear that there are other relationships among the types of omniscience previously discussed. For example, closure under logical equivalence follows from closure under logical implication. See Exercise 9.2 for further examples.

Full logical omniscience seems to be unavoidable, given that knowledge is defined as truth in all possible worlds, as it is in Kripke structures. If an agent knows all of the formulas in Ψ, then every formula in Ψ is true in every world he considers possible. Therefore, if Ψ logically implies φ, then φ is true in every world he considers possible. Hence, the agent must also know φ. Thus, any line of attack on the logical-omniscience problem has to start by addressing this basic definition of knowledge as truth in all possible worlds. The most radical approach is to abandon the definition altogether. We describe two approaches that do this, one syntactic and one semantic, in Section 9.2.

It is possible, however, to attack the logical-omniscience problem without completely abandoning the idea that knowledge means truth in all possible worlds. One approach to dealing with logical omniscience is to change the notion of truth. This is the direction pursued in Section 9.3, where we consider a nonstandard notion of truth. Another approach is to change the notion of possible world. This is what we do in Section 9.4, where we consider "impossible" worlds. Yet another approach is to have truth in all possible worlds be a necessary but not sufficient condition for knowledge. For example, in Section 9.5, we consider a notion of knowledge in which an agent is said to know a formula φ if φ is true in all worlds he considers possible and if, in addition, he is "aware" of the formula φ. Finally, we consider an approach in which knowledge is defined as truth in a *subset* of the possible worlds. In Section 9.6, we describe a notion of knowledge in which an agent is said to know

a formula φ if φ is true in all worlds he considers possible in some particular "frame of mind."

As we saw earlier, the possible-worlds approach gives rise to many different notions of knowledge whose appropriateness depends on the situation under consideration. For example, at the beginning of Chapter 3 we described a situation where the $\mathcal{K}_i$ relation need not be reflexive. Similarly, our goal in this chapter is to demonstrate that the logical-omniscience problem can be attacked in a variety of ways. Rather than prescribe the "correct" way to deal with logical omniscience, we describe several ways in which the problem can be dealt with. Because the issue of dealing with logical omniscience is orthogonal to the issues of dealing with distributed knowledge and common knowledge, we do not deal with distributed and common knowledge in this chapter.

In the chapters dealing with knowledge in multi-agent systems we used the term *knowledge* with a specific sense in mind: knowledge was defined by the possible-worlds semantics. The properties of this notion of knowledge were described in Chapters 2 and 3. But it is precisely this notion of knowledge that creates the logical-omniscience problem. Thus, in this chapter we start with a notion of knowledge that involves no prior assumptions on its properties. What exactly we mean by knowledge in this chapter will depend on the approach discussed and will differ from approach to approach.

9.2 Explicit Representation of Knowledge

As we already said, the simplest and most radical approach to the logical-omniscience problem is to abandon the definition of knowledge as truth in all possible worlds. This does not mean that we also have to abandon the notion of possible worlds. After all, the notion that the world can be in any one of a number of different states is independent of the concept of knowledge, and it arises naturally in a number of contexts, in particular, as we have seen, in our model of multi-agent systems. In this section, we intend to keep the possible-worlds framework, but change the way we define knowledge. Instead of defining knowledge *in terms* of possible worlds, we let knowledge be defined directly. Intuitively, we think of each agent's knowledge as being explicitly stored in a database of formulas. We now describe two ways to capture this intuition: a syntactic approach and its semantic analogue.

9.2.1 The Syntactic Approach

As we saw in Chapter 3, a Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ consists of frame $F = (S, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ and an assignment π of truth values to the primitive propositions in each state. Our definition of the satisfaction relation $\models$ then gives truth values to all formulas in all states. Here we replace the truth assignment π by a *syntactic assignment*. A syntactic assignment simply assigns truth values to all formulas in all states. For example, a syntactic assignment σ can assign both p and $\neg p$ to be true in a state s.

Since in this section we are interested in changing the definition of knowledge but not the underlying propositional semantics, we restrict our syntactic assignments here to be standard. A *standard* syntactic assignment σ is a syntactic assignment that obeys the following constraints for all formulas φ and ψ:

$\sigma(s)(\varphi) =$ **true** if and only if $\sigma(s)(\neg\varphi) =$ **false**, and

$\sigma(s)(\varphi \wedge \psi) =$ **true** if and only if $\sigma(s)(\varphi) =$ **true** and $\sigma(s)(\psi) =$ **true**.

Thus, in syntactic structures we replace truth assignments by standard syntactic assignments. In addition, we discard the possibility relations, because these relations were needed just to define satisfaction for formulas of the form $K_i\varphi$. Formally, a *syntactic structure* M is a pair (S, σ), consisting of a set S of states and a standard syntactic assignment σ. We can now define the truth of a formula φ in a syntactic structure in a straightforward way: $(M, s) \models \varphi$ precisely when $\sigma(s)(\varphi) =$ **true**. Notice that we can identify every Kripke structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ with the syntactic structure (S, σ), where $\sigma(s)(\varphi) =$ **true** if $(M, s) \models \varphi$. Thus, syntactic structures can be viewed as a generalization of Kripke structures. In fact, syntactic structures provide the most general model of knowledge. (More precisely, they provide a most general model for knowledge among models that are based on standard propositional semantics.)

It is easy to see that no form of logical omniscience holds for syntactic structures. For example, knowledge of valid formulas fails, because there is no requirement that a standard syntactic assignment assign the truth value **true** to formulas of the form $K_i\varphi$ where φ is a valid formula. Similarly, closure under logical equivalence fails, since φ and ψ could be logically equivalent, but a standard syntactic assignment may assign the truth value **true** to $K_i\varphi$ and the truth value **false** to $K_i\psi$. In fact, knowledge in syntactic structures does not have any interesting properties; the only formulas that are valid in all syntactic structures are substitution instances of propositional tautologies. If we want to use syntactic structures to model a notion of knowledge that does

obey certain properties, then we have to impose some constraints on the allowable standard syntactic assignments. For example, if we want to capture the fact that we are modeling knowledge rather than belief, then we can enforce the Knowledge Axiom ($K_i\varphi \Rightarrow \varphi$) by requiring that $\sigma(s)(\varphi) = \textbf{true}$ whenever $\sigma(s)(K_i\varphi) = \textbf{true}$.

One particular property of knowledge that syntactic structures fail to capture is a property that played a central role in our study of multi-agent systems in Chapter 4. In that framework we assumed that every agent in the system is in some local state at any point in time. An important feature of knowledge in the framework of Chapter 4 is its *locality*. That is, if s and s' are states of the system such that agent i has the same local state in both of them, i.e., $s \sim_i s'$, and agent i knows φ in state s, then i also knows φ in state s'. In Chapter 4, this property was a consequence of the definition of knowledge as truth in all possible worlds, but it is a property that we may want to keep even if we abandon that definition. For example, a natural interpretation of $\sigma(s)(K_i\varphi) = \textbf{true}$, which we pursue in Chapter 10, is that agent i can decide, say by using some algorithm A, whether φ follows from the information in i's local state. Unfortunately, syntactic structures have no notion of local state; we cannot get locality of knowledge just by imposing further restrictions on the syntactic assignments. If we want to capture locality of knowledge, then we need to reintroduce possibility relations, because we can use them to express locality. If the possibility relation $\mathcal{K}_i$ is an equivalence relation $\sim_i$, for $i = 1, \ldots, n$, then we can say that an agent i is in the same local state in states s and t precisely when $s \sim_i t$. For knowledge to depend only on the agent's local state, we should require that if $s \sim_i t$, then $\sigma(s)(K_i\varphi) = \sigma(t)(K_i\varphi)$.

Syntactic structures can be used to model fairly realistic situations. Consider the situation where each agent has a base set of formulas from which the agent's knowledge is derived using a sound but possibly incomplete set of inference rules. Formally, we could interpret this to mean that for each agent i, there is a formal system R_i of inference rules, and for each state $s \in S$, there is a set $B_i(s)$ (the base set of formulas) such that $\sigma(s)(K_i\varphi) = \textbf{true}$ iff φ is derivable from $B_i(s)$ using R_i. Intuitively, agent i knows φ if she can deduce φ from her base formulas using her inference rules. For example, a student might know that $x + a = b$ but not conclude that $x = b - a$, because he might not know the rule allowing subtraction of equal quantities from both sides. In this case, we would have $\sigma(s)(K_i(x+a = b)) = \textbf{true}$, but $\sigma(s)(K_i(x = b-a)) = \textbf{false}$. As another example, a deduction system might be capable of certain limited reasoning about equality. For example, from $A = B$ and $B = C$, it might be able to deduce that $A = C$; however, given the information that $f(1) = 1$ and that $f(x) = x \cdot f(x-1)$, it might not be able to deduce that $f(4) = 24$. In both of these cases, agents have a base set of formulas and an incomplete set of

inference rules. Notice that, in these examples, if we view the base set $B_i(s)$ of formulas as agent i's local state, then agent i's knowledge indeed depends only on his local state, so knowledge has the locality property.

9.2.2 The Semantic Approach

In a syntactic structure $M = (S, \sigma)$, for each state s the function $\sigma(s)$ tells which formulas are true at state s. In particular, agent i knows φ at state s precisely if $\sigma(s)(K_i\varphi) = \textbf{true}$. Essentially, an agent's knowledge is *explicitly* described at a state by giving a list of the formulas that he knows. While this approach indeed avoids the logical-omniscience problem, its main weakness is its syntactic flavor. After all, the separation between syntax and semantics is one of the major strengths of modern logic. Is it possible to "semanticize" this approach? That is, is it possible to model knowledge explicitly on a semantic level?

To do that, we first need to find the semantic counterpart of formulas. We can identify the semantic "content" of a formula φ with its *intension* (see Section 2.5), i.e., the set of states in which φ holds. The motivation for this identification is as follows. Let φ and ψ be two formulas with the same intension in a structure M. Then for all $s \in S$ we have that $(M, s) \models \varphi$ if and only if $(M, s) \models \psi$. That is, if φ and ψ have the same intension in M, then they are semantically indistinguishable in M. Consequently, we can take sets of states as the semantic counterpart of formulas. Put another way, we can think of a set W of states as a "proposition" p_W that is true precisely at the states of W. Thus, we can represent an agent's semantic knowledge by simply listing the propositions that he knows, instead of representing his knowledge syntactically by listing the formulas that he knows. Since a proposition is a set of states, we can describe agent i's semantic knowledge explicitly by a set of sets of states.

The previous discussion motivates the following definition. A *Montague-Scott structure* M is a tuple $(S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ where S is a set of states, $\pi(s)$ is a truth assignment to the primitive propositions for each state $s \in S$, and $\mathcal{C}_i(s)$ is a set of subsets of S, for $i = 1, \ldots, n$. For the sake of brevity, we refer to Montague-Scott structures as *MS structures*. In an MS structure, we describe agent i's knowledge (in state s) by a set of sets of states; this is given to us by $\mathcal{C}_i(s)$. The members of $\mathcal{C}_i(s)$ are the propositions that agent i knows.

We can now define $\models$ for all formulas. The clauses for primitive propositions, conjunctions, and negations are identical to the corresponding clauses for Kripke structures. The clause for formulas $K_i\varphi$ is different:

$(M, s) \models K_i\varphi$ iff $\{t \mid (M, t) \models \varphi\} \in \mathcal{C}_i(s)$.

As in Section 2.5, we denote the intension of a formula φ in the structure M by φ^M. That is, $\varphi^M = \{s \mid (M, s) \models \varphi\}$ is the set of states in M where φ is true. The clause above says that agent i knows φ at state s if the intension of φ is one of the propositions that he knows, that is, if $\varphi^M \in \mathcal{C}_i(s)$.

Example 9.2.1 These definitions are perhaps best illustrated by a simple example. Suppose $\Phi = \{p\}$ and $n = 2$, so that our language has one primitive proposition p and there are two agents. Further suppose that $M' = (S, \pi, \mathcal{C}_1, \mathcal{C}_2)$, where $S = \{s, t, u\}$, and that the primitive proposition p is true at states s and u, but false at t (so that $\pi(s)(p) = \pi(u)(p) =$ **true** and $\pi(t)(p) =$ **false**). Suppose $\mathcal{C}_1(s) = \mathcal{C}_1(t) = \{\{s, t\}, \{s, t, u\}\}$ and $\mathcal{C}_1(u) = \{\{u\}, \{s, u\}, \{t, u\}, \{s, t, u\}\}$. Suppose also that $\mathcal{C}_2(s) = \mathcal{C}_2(u) = \{\{s, u\}, \{s, t, u\}\}$ and $\mathcal{C}_2(t) = \{\{t\}, \{s, t\}, \{t, u\}, \{s, t, u\}\}$. Consider agent 1. In states s and t agent 1 knows the propositions $\{s, t\}$ and $\{s, t, u\}$, and in state u he knows the propositions $\{u\}$, $\{s, u\}$, $\{t, u\}$, and $\{s, t, u\}$. In some sense, one could say that agent 1 cannot distinguish between the states s and t, since $\mathcal{C}_1(s) = \mathcal{C}_1(t)$, and s and t play symmetric roles in $\mathcal{C}_1(s)$, $\mathcal{C}_1(t)$, and $\mathcal{C}_1(u)$. On the other hand, agent 1 can distinguish between the state u and each of s and t, since $\mathcal{C}_1(s) = \mathcal{C}_1(t) \neq \mathcal{C}_1(u)$. The situation for agent 2 is analogous.

Consider the formulas $K_1 p$ and $K_1 \neg p$. The intension of p is $\{s, u\}$ and the intension of $\neg p$ is $\{t\}$. Since $\{s, u\} \notin \mathcal{C}_1(s)$ and $\{t\} \notin \mathcal{C}_1(s)$, in state s agent 1 does not know whether or not p holds. That is,

$$(M', s) \not\models K_1 p \vee K_1 \neg p.$$

Consider the formulas $K_2 p$ and $K_2 \neg p$. Because the intension of p is $\{s, u\}$, and since $\{s, u\} \in \mathcal{C}_2(s)$, $\{s, u\} \in \mathcal{C}_2(u)$, and $\{s, u\} \notin \mathcal{C}_2(t)$, the intension of $K_2 p$ is $\{s, u\}$. Similarly, the intension of $K_2 \neg p$ is $\{t\}$. Thus, the intension of $K_2 p \vee K_2 \neg p$ is $\{s, t, u\}$. Because $\{s, t, u\} \in \mathcal{C}_1(s)$, it follows that in state s agent 1 knows that agent 2 knows whether or not p is true. That is,

$$(M', s) \models K_1(K_2 p \vee K_2 \neg p).$$

It is instructive to compare this example with the example described by Figure 2.1. In that example, we considered a Kripke structure $M = (S, \pi, \mathcal{K}_1, \mathcal{K}_2)$ with the same state space as M'. There are a number of other more significant similarities between M and M'. Just as with M', agent 1 cannot distinguish s and t in M, although he can distinguish u from both of them. Similarly, agent 2 cannot distinguish s and u in M, although she can distinguish s and t. Notice, however, that the way we captured indistinguishability in M (using the relations $\mathcal{K}_1$ and $\mathcal{K}_2$) is very different

from the way we capture it in M' (in terms of the sets of propositions the agents know). Nevertheless, it can be shown that precisely the same formulas are true at corresponding states in the two structures. As we shall see, this similarity between the MS structure M' and the Kripke structure M is not a coincidence. ∎

We observed in Section 9.2.1 that syntactic structures generalize Kripke structures. Similarly, MS structures can also be viewed as a generalization of Kripke structures. Thus, let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ be a Kripke structure. Let $\mathcal{K}_i(s)$ be the set of all "i-neighbors" of a state s, i.e.,

$$\mathcal{K}_i(s) = \{t \mid (s, t) \in \mathcal{K}_i\}.$$

Let M' be the MS structure $(S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, where $\mathcal{C}_i(s)$ is the set of all supersets of $\mathcal{K}_i(s)$. Intuitively, in a state s of M, an agent i knows that the actual state is one of the states in $\mathcal{K}_i(s)$. Thus, i knows all the propositions that contain $\mathcal{K}_i(s)$. It can now be shown that for each formula φ we have $(M, s) \models \varphi$ iff $(M', s) \models \varphi$ (Exercise 9.4). This explains the tight correspondence between the MS structure M' and the Kripke structure M observed in Example 9.2.1. The reader can verify that in that example we indeed had that $\mathcal{C}_i(s)$ is the set of all supersets of $\mathcal{K}_i(s)$.

Earlier we observed that syntactic structures strip away all properties of knowledge. This is not quite the case for MS structures. Suppose that φ and ψ are equivalent formulas. Then, by definition, they must have the same intension in every MS structure. It follows that $K_i\varphi$ is true in a state precisely when $K_i\psi$ is true in that state. Thus, knowledge in MS structures is closed under logical equivalence. It is easy to verify, however, that all other forms of logical omniscience fail here (see Exercise 9.5). In fact, closure under logical equivalence is in some formal sense the only necessary property of knowledge in MS structures.

Theorem 9.2.2 *The following is a sound and complete axiomatization for validity with respect to MS structures:*

A1. All instances of tautologies of propositional logic

R1. From φ and $\varphi \Rightarrow \psi$ infer ψ (modus ponens)

LE. From $\varphi \Leftrightarrow \psi$ infer $K_i\varphi \Leftrightarrow K_i\psi$

Proof See Exercise 9.6. ∎

Thus, propositional reasoning and closure under logical equivalence completely characterize knowledge in MS structures. This suggests that reasoning about knowledge in Montague-Scott semantics may not be any harder than propositional reasoning. This indeed is the case, as the following result shows.

Theorem 9.2.3 *The satisfiability problem with respect to MS structures is NP-complete.*

Proof The lower bound is immediate, since the satisfiability problem for propositional logic is already *NP*-hard. For the upper bound, we proceed along similar lines to the proof of Proposition 3.6.2: we show that if a formula φ is satisfiable, then it is satisfiable in an MS structure with at most $|\varphi|^2$ states. We leave details to the reader (Exercise 9.7); however, the following example might provide some intuition. Consider the formula $\varphi = K_i\varphi_1 \wedge \ldots \wedge K_i\varphi_r \wedge \neg K_i\psi$. Clearly, for φ to be satisfiable, ψ cannot be logically equivalent to any of the φ_j, for $j = 1, \ldots, r$. In other words, not knowing ψ has to be consistent with knowing φ_j, for $j = 1, \ldots, r$. It turns out that the consistency of not knowing ψ with knowing φ_j, for $j = 1, \ldots, r$, is also a sufficient condition for the satisfiability of φ. This means that we can test if φ is satisfiable by testing if $K_i\varphi_j \wedge \neg K_i\psi$ is satisfiable, for $j = 1, \ldots, r$. Moreover, it is easy to see that $K_i\varphi_j \wedge \neg K_i\psi$ is satisfiable exactly if at least one of $\neg\varphi_j \wedge \psi$ or $\varphi_j \wedge \neg\psi$ is satisfiable. Thus, we can decompose the problem of testing if φ is satisfiable into testing a number of smaller satisfiability problems. In fact, it can be shown that there are only quadratically many such problems to test (at most two for every pair of subformulas of φ), so the problem is in *NP*. ∎

Do MS structures avoid logical omniscience? The answer is "almost." As we observed, all forms of logical omniscience fail except for closure under logical equivalence. In other words, while agents need not know all logical consequences of their knowledge, they are unable to distinguish between logically equivalent formulas. This is as much as we can expect to accomplish in a purely semantic model, since logically equivalent formulas are by definition semantically indistinguishable. Thus, just as syntactic structures provide the most general model of knowledge, MS structures provide the most general *semantic* model of knowledge. (Of course, just as in the case of syntactic structures, MS structures provide the most general semantic model of knowledge only among models that are based on standard propositional semantics, since Montague-Scott semantics is based on standard propositional semantics.)

We saw in Chapter 3 how certain properties of knowledge in Kripke structures correspond to certain conditions on the possibility relations $\mathcal{K}_i$. Similarly, certain properties of knowledge in MS structures correspond to certain conditions on the $\mathcal{C}_i$'s. Especially interesting are properties of knowledge that correspond to various forms of logical omniscience. For example, knowledge of valid formulas corresponds to the condition that $S \in \mathcal{C}_i(s)$, where S is the set of all states, since the intension of *true* is the set S. For another example, consider closure under conjunction. This

property corresponds to $\mathcal{C}_i(s)$ being closed under intersection; that is, if U and V are in $\mathcal{C}_i(s)$, then $U \cap V$ is also in $\mathcal{C}_i(s)$. The reason for this is that the intension of $\varphi \wedge \psi$ is the intersection of the intensions of φ and ψ. Exercise 9.8 provides a precise statement of the equivalence between various properties of knowledge and corresponding restrictions on the $\mathcal{C}_i$'s.

We already saw in Chapter 3 that imposing certain restrictions on Kripke structures, or, equivalently, assuming that knowledge satisfies some additional properties may sometimes (but not always) have an effect on the computational complexity of reasoning about knowledge. A similar phenomenon occurs in the context of MS structures. It turns out that we can capture several properties of knowledge without increasing the complexity of reasoning beyond that of propositional reasoning (i.e., *NP*-complete). Once, however, we capture closure under conjunction by insisting that each $\mathcal{C}_i(s)$ be closed under intersection, then the complexity of the satisfiability problem rises to *PSPACE*-complete. To understand the intuitive reason for this difference, consider again the formula $\varphi = K_i\varphi_1 \wedge \ldots \wedge K_i\varphi_r \wedge \neg K_i\psi$. As we observed in the proof of Theorem 9.2.3, if we do not assume closure under conjunction, then a necessary and sufficient condition for φ to be satisfiable is that ψ cannot be logically equivalent to any of the φ_j, for $j = 1, \ldots, r$. The situation is quite different when we do assume closure under conjunction. Now it is not sufficient that ψ not be logically equivalent to any of the φ_j's; it also cannot be equivalent to any conjunction of φ_j's. In other words, in the presence of closure under conjunction we have to show that not knowing ψ is simultaneously consistent with knowing any conjunction of φ_j's. Thus, to test whether φ is satisfiable, we have to consider sets of subformulas of φ rather than only pairs of subformulas of φ. Since there are exponentially many such sets, the problem is *PSPACE*-hard.

9.2.3 Discussion

The two approaches described in this section overcome the logical omniscience problem by explicitly modeling an agent's knowledge, either as a set of formulas (the formulas the agent knows) or as a set of sets of possible worlds (the intensions of the formulas the agent knows). These approaches are very powerful. They solve the logical-omniscience problem by giving us direct fine-grained control over an agent's knowledge. This power, however, comes at a price. One gains very little intuition about knowledge from studying syntactic structures or MS structures; in these approaches knowledge is a primitive construct (much like the primitive propositions in a Kripke structure). Arguably, these approaches give us ways of *representing* knowledge rather than *modeling* knowledge. In contrast, the semantics given to

knowledge in Kripke structures *explains* knowledge as truth in all possible worlds. Unfortunately, this "explanation" does not fit certain applications, because it forces logical omniscience.

In the following sections, we try to steer a middle course, by keeping the flavor of the possible-worlds approach, while trying to mitigate its side effects.

9.3 Nonstandard Logic

If knowledge is truth in all possible worlds, then one way to deal with logical omniscience is to change the notion of truth. The underlying idea is to weaken the "logical" aspect of the logical-omniscience problem, thus reducing the acuteness of the problem. Indeed, as we saw in Section 9.1, certain forms of logical omniscience follow from full logical omniscience only under standard propositional semantics. The nonstandard semantics for knowledge we are about to describe is based on a nonstandard propositional semantics. Knowledge is still defined to be truth in all possible worlds, so we still have logical omniscience, but this time with respect to the nonstandard logic. The hope is that the logical-omniscience problem can be alleviated somewhat by appropriately choosing the nonstandard logic.

There are many ways in which one can define a nonstandard propositional semantics. We describe here one approach that changes the treatment of negation. We do not mean to argue that this is the "right" propositional semantics to deal with knowledge, but rather we mean to demonstrate how knowledge can be modeled on the basis of a nonstandard propositional semantics.

9.3.1 Nonstandard Structures

Standard propositional logic has several undesirable and counterintuitive properties. Often people first introduced to propositional logic are somewhat uncomfortable when they learn that "$\varphi \Rightarrow \psi$" is taken to be simply an abbreviation for $\neg\varphi \vee \psi$. Why should the fact that either $\neg\varphi$ is true or ψ is true correspond to "if φ is true, then ψ is true"?

Another problem with standard propositional logic is that it is fragile: a false statement implies everything. In particular, the formula $(p \wedge \neg p) \Rightarrow q$ is valid, even when p and q are unrelated primitive propositions; for example, p could say that Alice graduated from college in 1987 and q could say that Bob's salary is \$500,000. This could be a serious problem if we have a large database of formulas, obtained from multiple sources. Such a database will often contain an inconsistency; for

example, someone may have input the datum that Alice graduated in 1987, and someone else may have input the datum that Alice graduated in 1986. If the database contains a constraint that each person's graduation year is unique, then, using standard propositional reasoning, any arbitrary fact about Bob's salary can be derived from the database. Many alternatives to the standard semantics have been proposed over the years, designed to deal with various aspects of these problems. We focus on one particular alternative here, and consider its consequences.

The idea is to allow formulas φ and $\neg\varphi$ to have "independent" truth values. Thus, rather than requiring that $\neg\varphi$ be true if and only if φ is false, we wish instead to allow the possibility that $\neg\varphi$ can be either true or false, regardless of whether φ is true or false. Intuitively, the truth of formulas can be thought of as being determined by some database of formulas. We can think of φ being true as meaning that the fact φ is in a database of true formulas, and we can think of $\neg\varphi$ being true as meaning that the fact φ is in a database of false formulas. Since it is possible for φ to be in both databases, it is possible for both φ and $\neg\varphi$ to be true. Similarly, if φ is in neither database, then neither φ nor $\neg\varphi$ would be true.

There are several ways to capture this intuition formally. We now discuss one approach; some closely related approaches are discussed in Exercises 9.9 and 9.10. For each state s, there is an *adjunct* state s^*, which is used for giving semantics to negated formulas. Rather than defining $\neg\varphi$ to hold at s iff φ does not hold at s, we instead define $\neg\varphi$ to hold at s iff φ does not hold at s^*. Note that if $s = s^*$, then this gives our usual notion of negation. Very roughly, we can think of a state s as consisting of a pair $\langle B_T, B_F\rangle$ of databases; B_T is the database of true facts, while B_F is the database of false facts. The state s^* should be thought of as the adjunct pair $\langle \overline{B_F}, \overline{B_T}\rangle$ (where, if X is a set of formulas, then $\overline{X}$ is the set consisting of all formulas not in X). Continuing this intuition, to see if φ holds at s, we check if $\varphi \in B_T$; to see if $\neg\varphi$ holds at s, we check if $\varphi \in B_F$. Notice that $\varphi \in B_F$ iff $\varphi \notin \overline{B_F}$. Since $\overline{B_F}$ is the database of true facts at s^*, we have an alternative way of checking if $\neg\varphi$ holds at s: we can check if φ does not hold at s^*. Note that if $B_T = \overline{B_F}$, then $s = s^*$ and we get the standard semantics of negation.

Under this interpretation, not only is s^* the adjunct state of s, but s is the adjunct state of s^*; i.e., $s^{**} = s$ (where $s^{**} = (s^*)^*$). To support this intuitive view of s as a pair of databases and s^* as its adjunct, we make $s^{**} = s$ a general requirement in our framework.

A *nonstandard (Kripke) structure* M is a tuple $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$, where the tuple $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is a Kripke structure, and * is a unary function from the set S of worlds to itself (where we write s^* for the result of applying the function * to the state s) such that $s^{**} = s$ for each $s \in S$. We call these structures nonstandard,

since we think of a world where φ and $\neg\varphi$ are both true or both false as nonstandard. We denote the class of nonstandard structures for n agents over Φ by $\mathcal{NM}_n(\Phi)$ (or by $\mathcal{NM}_n$ when Φ is clear from the context).

The definition of $\models$ is the same as for standard Kripke structures, except for the clause for negation. In this case, we have

$(M, s) \models \neg\varphi$ iff $(M, s^*) \not\models \varphi$.

Note that it is possible for neither φ nor $\neg\varphi$ to be true at state s (if $(M, s) \not\models \varphi$ and $(M, s^*) \models \varphi$) and for both φ and $\neg\varphi$ to be true at state s (if $(M, s) \models \varphi$ and $(M, s^*) \not\models \varphi$). We call a state s where neither φ nor $\neg\varphi$ is true *incomplete* (with respect to φ); otherwise, we call s *complete* (with respect to φ). The intuition behind an incomplete state is that there is not enough information to determine whether φ is true or whether $\neg\varphi$ is true. We call a state s where both φ and $\neg\varphi$ are true *incoherent* (with respect to φ); otherwise, s is *coherent* (with respect to φ). The intuition behind an incoherent state is that it is overdetermined; it might correspond to a situation where several people have provided mutually inconsistent information.

A state s is *standard* if $s = s^*$. Note that for a standard state, the semantics of negation is equivalent to the standard semantics. In particular, a standard state s is both complete and coherent with respect to all formulas: for each formula φ exactly one of φ or $\neg\varphi$ is true at s. (See also Exercise 9.11.)

In standard propositional logic, disjunction ($\vee$) and material implication ($\Rightarrow$) can be defined in terms of conjunction and negation, that is, $\varphi_1 \vee \varphi_2$ can be defined as $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, and $\varphi_1 \Rightarrow \varphi_2$ can be defined as $\neg\varphi_1 \vee \varphi_2$. We retain these definitions in the nonstandard framework. Since, however, the semantics of negation is now nonstandard, it is not *a priori* clear how the propositional connectives behave in our nonstandard semantics. For example, while $p \wedge q$ holds by definition precisely when p and q both hold, it is not clear that $p \vee q$ holds precisely when at least one of p or q holds. It is even less clear how negation interacts with conjunction and disjunction in our nonstandard semantics. The next proposition shows that even though we have decoupled the semantics for φ and $\neg\varphi$, the propositional connectives $\neg$, $\wedge$, and $\vee$ still behave in a fairly standard way.

Proposition 9.3.1 *Let M be a nonstandard structure. Then*

(a) $(M, s) \models \neg\neg\varphi$ *iff* $(M, s) \models \varphi$.

(b) $(M, s) \models \varphi \vee \psi$ *iff* $(M, s) \models \varphi$ *or* $(M, s) \models \psi$.

(c) $(M, s) \models \neg(\varphi \wedge \psi)$ *iff* $(M, s) \models \neg\varphi \vee \neg\psi$.

(d) $(M, s) \models \neg(\varphi \vee \psi)$ *iff* $(M, s) \models \neg\varphi \wedge \neg\psi$.

(e) $(M, s) \models \varphi \wedge (\psi_1 \vee \psi_2)$ *iff* $(M, s) \models (\varphi \wedge \psi_1) \vee (\varphi \wedge \psi_2)$.

(f) $(M, s) \models \varphi \vee (\psi_1 \wedge \psi_2)$ *iff* $(M, s) \models (\varphi \vee \psi_1) \wedge (\varphi \vee \psi_2)$.

Proof See Exercise 9.12. ∎

In contrast to $\wedge$ and $\vee$, the connective $\Rightarrow$ behaves in a nonstandard fashion. In particular, both p and $p \Rightarrow q$ can be true at a state without q being true, so $\Rightarrow$ does not capture our usual notion of logical implication (see Exercise 9.14).

What are the properties of knowledge in nonstandard structures? So far, our approach to understanding the properties of knowledge in some semantic model has been to consider all the valid formulas under that semantics. What are the valid formulas with respect to $\mathcal{NM}_n$? It is easy to verify that certain tautologies of standard propositional logic are not valid. For example, the formula $(p \wedge \neg p) \Rightarrow q$, which wreaked havoc in deriving consequences from a database, is not valid. How about even simpler tautologies of standard propositional logic, such as $p \Rightarrow p$? This formula, too, is not valid. One might think that these formulas are not valid because of the nonstandard behavior of $\Rightarrow$, but observe that $p \Rightarrow p$ is just an abbreviation for $\neg p \vee p$ (which is not valid). In fact, no formula is valid with respect to $\mathcal{NM}_n$! Furthermore, there is a single structure that simultaneously shows that no formula is valid!

Theorem 9.3.2 *No formula of $\mathcal{L}_n$ is valid with respect to $\mathcal{NM}_n$. In fact, there is a nonstandard structure M and a state s of M such that every formula of $\mathcal{L}_n$ is false at s, and a state t of M such that every formula of $\mathcal{L}_n$ is true at t.*

Proof Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, {}^*)$ be a special nonstandard structure, defined as follows. Let S contain only two states s and t, where $t = s^*$ (and so $s = t^*$). Define π by taking $\pi(s)$ be the truth assignment where $\pi(s)(p) =$ **false** for every primitive proposition p, and taking $\pi(t)$ be the truth assignment where $\pi(t)(p) =$ **true** for every primitive proposition p. Define $\mathcal{K}_i$ to be $\{(s, s), (t, t)\}$, for $i = 1, \ldots, n$. By a straightforward induction on the structure of formulas (Exercise 9.13), it follows that for every formula φ of $\mathcal{L}_n$, we have $(M, s) \not\models \varphi$ and $(M, t) \models \varphi$. In particular, every formula of $\mathcal{L}_n$ is false at s and every formula of $\mathcal{L}_n$ is true at t. Since every formula of $\mathcal{L}_n$ is false at s, no formula of $\mathcal{L}_n$ is valid with respect to $\mathcal{NM}_n$. ∎

It follows from Theorem 9.3.2 that the validity problem with respect to $\mathcal{NM}_n$ is *very* easy: the answer is always, "No, the formula is not valid!" Thus, the notion

of validity is trivially uninteresting in our logic. In particular, we cannot use valid formulas to characterize the properties of knowledge in nonstandard structures, since there are no valid formulas.

In contrast to validity, there are many nontrivial logical implications with respect to $\mathcal{NM}_n$. For example, as we see from Proposition 9.3.1, $\neg\neg\varphi$ logically implies φ and $\neg(\varphi_1 \wedge \varphi_2)$ logically implies $\neg\varphi_1 \vee \neg\varphi_2$. The reader may be puzzled why Proposition 9.3.1 does not provide us with some tautologies. For example, Proposition 9.3.1 tells us that $\neg\neg\varphi$ logically implies φ. Doesn't this mean that $\neg\neg\varphi \Rightarrow \varphi$ is a tautology? This does not follow. With *standard* Kripke structures, φ logically implies ψ iff the formula $\varphi \Rightarrow \psi$ is valid. This is not the case for *nonstandard* structures; here, logical and material implication do not coincide. For example, φ logically implies φ, yet we have already observed that $\varphi \Rightarrow \varphi$ (i.e., $\neg\varphi \vee \varphi$) is not valid with respect to $\mathcal{NM}_n$. In Section 9.3.2, we define a new connective that allows us to express logical implication *in the language*, just as $\Rightarrow$ does for standard Kripke structures.

What about logical omniscience? Full logical omniscience holds, just as with ordinary Kripke structures. For example, it follows from Proposition 9.3.1(b) that φ logically implies $\varphi \vee \psi$; hence, by full logical omniscience, $K_i\varphi$ logically implies $K_i(\varphi \vee \psi)$. Moreover, closure under conjunction holds, since $\{\varphi, \psi\}$ logically implies $\varphi \wedge \psi$. Nevertheless, since it is not the case here that $\{\varphi, \varphi \Rightarrow \psi\}$ logically implies ψ (Exercise 9.14), we might expect that closure under material implication would fail. This is indeed the case: it is possible for $K_i\varphi$ and $K_i(\varphi \Rightarrow \psi)$ to hold, without $K_i\psi$ holding (Exercise 9.16). Finally, note that although knowledge of valid formulas holds, it is completely innocuous here; there are no valid formulas!

9.3.2 Strong Implication

In the previous subsection we introduced nonstandard semantics, motivated by our discomfort with the tautology $(p \wedge \neg p) \Rightarrow q$, and, indeed, under this semantics $(p \wedge \neg p) \Rightarrow q$ is no longer valid. Unfortunately, the bad news is that other formulas, such as $\varphi \Rightarrow \varphi$, that seem as if they should be valid, are not valid either. In fact, as we saw, no formula is valid in the nonstandard approach. It seems that we have thrown out the baby with the bath water.

To get better insight into this problem, let us look more closely at why the formula $\varphi \Rightarrow \varphi$ is not valid. Our intuition about implication tells us that $\varphi_1 \Rightarrow \varphi_2$ should say "if φ_1 is true, then φ_2 is true," but $\varphi_1 \Rightarrow \varphi_2$ is defined to be $\neg\varphi_1 \vee \varphi_2$, which says "either $\neg\varphi_1$ is true or φ_2 is true." In standard propositional logic, this is the same as "if φ_1 is true, then φ_2 is true," since $\neg\varphi_1$ is false in standard logic iff φ_1 is true. In

nonstandard structures, however, these are not equivalent. In particular, $\varphi \Rightarrow \varphi$ is simply an abbreviation for $\neg\varphi \vee \varphi$. Since our semantics decouples the meaning of φ and $\neg\varphi$, the formula $\neg\varphi \vee \varphi$ *should not* be valid.

While the above explains why $\neg\varphi \vee \varphi$ is not valid, it still seems that the statement "if φ is true then φ is true" ought to be valid. Unfortunately, our definition of $\Rightarrow$ does not capture the intuition of "if ... then ..." This motivates the definition of a new propositional connective $\hookrightarrow$, which we call *strong implication*, where $\varphi_1 \hookrightarrow \varphi_2$ is defined to be true if whenever φ_1 is true, then φ_2 is true. Formally,

$(M, s) \models \varphi_1 \hookrightarrow \varphi_2$ iff $(M, s) \models \varphi_2$ holds whenever $(M, s) \models \varphi_1$ does.

That is, $(M, s) \models \varphi_1 \hookrightarrow \varphi_2$ iff either $(M, s) \not\models \varphi_1$ or $(M, s) \models \varphi_2$.

We denote by $\mathcal{L}_n^{\hookrightarrow}(\Phi)$, or $\mathcal{L}_n^{\hookrightarrow}$ when Φ is obvious from the context, the set of formulas obtained by closing off $\mathcal{L}_n(\Phi)$ under the connective $\hookrightarrow$ (i.e., if φ and ψ are formulas of $\mathcal{L}_n^{\hookrightarrow}(\Phi)$, then so is $\varphi \hookrightarrow \psi$). We call the formulas of $\mathcal{L}_n^{\hookrightarrow}$ *nonstandard formulas*, to distinguish them from the *standard formulas* of $\mathcal{L}_n$. We call the propositional fragment of $\mathcal{L}_n^{\hookrightarrow}$ and its interpretation by nonstandard structures *nonstandard propositional logic*, to distinguish it from *standard propositional logic*. We redefine *true* to be an abbreviation for some fixed nonstandard tautology such as $p \hookrightarrow p$; again, we abbreviate $\neg$*true* by *false*.

Strong implication is indeed a new connective, that is, it cannot be defined using (nonstandard) $\neg$ and $\wedge$. For example, there are no valid formulas using only $\neg$ and $\wedge$, whereas by using $\hookrightarrow$, there are valid formulas: $\varphi \hookrightarrow \varphi$ is valid, as is $\varphi_1 \hookrightarrow (\varphi_1 \vee \varphi_2)$. Strong implication is indeed stronger than implication, in the sense that if φ_1 and φ_2 are standard formulas (formulas of $\mathcal{L}_n$), and if $\varphi_1 \hookrightarrow \varphi_2$ is valid with respect to nonstandard Kripke structures, then $\varphi_1 \Rightarrow \varphi_2$ is valid with respect to standard Kripke structures (Exercise 9.17). The converse, however, is false. For example, the formula $(p \wedge \neg p) \Rightarrow q$ is valid with respect to standard propositional logic, whereas the formula $(p \wedge \neg p) \hookrightarrow q$ is not valid with respect to nonstandard propositional logic (Exercise 9.17).

As we promised in the previous section, we can now express nonstandard logical implication using $\hookrightarrow$, just as we can express standard logical implication using $\Rightarrow$.

Proposition 9.3.3 *Let φ_1 and φ_2 be formulas in $\mathcal{L}_n^{\hookrightarrow}$. Then φ_1 logically implies φ_2 with respect to $\mathcal{NM}_n$ iff $\varphi_1 \hookrightarrow \varphi_2$ is valid with respect to $\mathcal{NM}_n$.*

Proof See Exercise 9.18. ∎

It turns out that it was the lack of ability to express logical implication within the language that prevented us from getting interesting formulas valid in $\mathcal{NM}_n$. Now

that we have introduced $\hookrightarrow$, there are many valid formulas. Thus, we can try again to characterize the properties of knowledge by getting a sound and complete axiomatization for $\mathcal{L}_n^{\hookrightarrow}$. Such an axiomatization can be obtained by modifying the axiom system K_n by (a) replacing propositional reasoning by nonstandard propositional reasoning, and (b) replacing standard implication ($\Rightarrow$) in the other axioms and rules by strong implication ($\hookrightarrow$). Thus, we obtain the axiom system $\mathrm{K}_n^{\hookrightarrow}$, which consists of all instances (for the language $\mathcal{L}_n^{\hookrightarrow}$) of the following axiom scheme and inference rules:

A2$^{\hookrightarrow}$. $(K_i\varphi \wedge K_i(\varphi \hookrightarrow \psi)) \hookrightarrow K_i\psi$ (Distribution Axiom)

NPR. All sound inference rules of nonstandard propositional logic

R2. From φ infer $K_i\varphi$ (Knowledge Generalization)

Note that the way we include nonstandard propositional reasoning in our axiomatization is quite different from the way we include standard propositional reasoning in the axiomatizations of Chapter 3. In the axiomatizations of Chapter 3, we took from the underlying propositional logic only the tautologies and modus ponens, while here we include all sound inference rules of nonstandard propositional logic. An example of a sound inference rule of nonstandard propositional logic is (nonstandard) modus ponens: From φ and $\varphi \hookrightarrow \psi$ infer ψ (this is a "nonstandard" rule, since it uses $\hookrightarrow$ instead of $\Rightarrow$). Other examples of sound inference rules of nonstandard propositional logic are given in Exercise 9.20. It can be shown that the axiomatization would not be complete had we included only the valid formulas of our nonstandard propositional logic as axioms, along with nonstandard modus ponens as the sole nonstandard propositional inference rule, rather than including all the sound inference rules of nonstandard propositional logic.

Theorem 9.3.4 $\mathrm{K}_n^{\hookrightarrow}$ *is a sound and complete axiomatization with respect to* $\mathcal{NM}_n$ *for formulas in the language* $\mathcal{L}_n^{\hookrightarrow}$.

Proof See Exercise 9.19. ∎

Theorem 9.3.4 shows that we can in some sense separate the properties of knowledge from the properties of the underlying propositional semantics. Even though logical omniscience is an essential feature of the possible-worlds approach (it is easy to see that in our enlarged language $\mathcal{L}_n^{\hookrightarrow}$, all of our types of logical omniscience hold, where we use $\hookrightarrow$ instead of $\Rightarrow$; see Exercise 9.21), it can be controlled to a certain degree by varying the propositional component of the semantics. Thus, one can say that in the nonstandard approach agents are "nonstandardly" logically omniscient.

In Chapter 3, we saw that under the standard semantics we can capture additional properties of knowledge by imposing suitable restrictions on the possibility relations $\mathcal{K}_i$. For example, restricting $\mathcal{K}_i$ to be reflexive captures the property that agents know only true facts ($K_i\varphi \Rightarrow \varphi$), and restricting $\mathcal{K}_i$ to be transitive captures the property of positive introspection ($K_i\varphi \Rightarrow K_iK_i\varphi$). The same phenomenon occurs under the nonstandard semantics. For example, restricting $\mathcal{K}_i$ to be reflexive captures the property that agents know only true facts, and analogously for positive introspection. Note, however, that to express these properties by valid formulas in the nonstandard approach, we need to use strong implication instead of standard implication. That is, the property that agents know only true facts is expressed by the axiom $K_i\varphi \hookrightarrow \varphi$, and the property of positive introspection is expressed by the axiom $K_i\varphi \hookrightarrow K_iK_i\varphi$. In the standard approach we captured negative introspection ($\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$) by requiring $\mathcal{K}_i$ to be Euclidean. It turns out that this is not sufficient to capture negative introspection in the nonstandard approach (as expressed by the axiom $\neg K_i\varphi \hookrightarrow K_i\neg K_i\varphi$), because of the nonstandard behavior of negation. To capture negative introspection we have to impose further restrictions on $\mathcal{K}_i$. For example, a sufficient additional restriction is that $(s, t) \in \mathcal{K}_i$ implies that $(s^*, t^*) \in \mathcal{K}_i$ (see Exercise 9.22).

Now that we have characterized the properties of knowledge in the nonstandard approach (in Theorem 9.3.4), we can consider the computational complexity of reasoning about knowledge, i.e., the complexity of determining validity in the nonstandard approach. Clearly, without $\hookrightarrow$, determining validity is very easy, since no formula is valid. As we saw in Proposition 9.3.3, however, $\hookrightarrow$ enables us to express logical implication. It turns out that once we introduce $\hookrightarrow$, reasoning about knowledge in the nonstandard approach is just as hard as reasoning about knowledge in the standard approach. The reason for this is the ability to "emulate" the standard semantics within the nonstandard semantics. Let φ be a formula of $\mathcal{L}_n^{\hookrightarrow}$. Then $(M, s) \models \varphi \hookrightarrow \mathit{false}$ iff $(M, s) \not\models \varphi$ for all nonstandard structures M and states s (see Exercise 9.23). Thus, standard negation can be expressed using strong implication.

Theorem 9.3.5 *The validity problem for $\mathcal{L}_n^{\hookrightarrow}$-formulas with respect to $\mathcal{NM}_n$ is PSPACE-complete.*

Proof The polynomial-space upper bound holds for much the same reasons that it does in the case of the logic K_n (see Section 3.5); a proof is beyond the scope of this book. For the lower bound, see Exercise 9.24. ∎

As we observed, logical omniscience still holds in the nonstandard approach. We also observed that the computational complexity of reasoning about knowledge does

not improve. Nevertheless, our goal, which was to weaken the "logical" aspect in the logical-omniscience problem, is accomplished. For example, under the nonstandard semantics, agents do not know all *standard* tautologies. In the next section we provide an additional payoff of this approach: we show that in a certain important application we can obtain polynomial-time algorithms for reasoning about knowledge.

9.3.3 A Payoff: Querying Knowledge Bases

In Section 4.4.1, we introduced and discussed knowledge bases. An interesting application of our approach here concerns query evaluation in knowledge bases. Recall that the knowledge base (KB for short) is told facts about an external world, and is asked queries about the world. As we saw in Section 4.4.1, after the KB is told a sequence of standard propositional facts whose conjunction is κ, it then answers the propositional query ψ positively precisely when $\kappa \Rightarrow \psi$ is valid with respect to standard propositional logic, which is precisely when $K_{KB}\kappa \Rightarrow K_{KB}\psi$ is valid with respect to $\mathcal{M}_n^{rst}$. Thus, the formula κ completely characterizes the KB's knowledge in this case.

Our focus in this section is on the computational complexity of query evaluation in knowledge bases. We know that in the standard propositional approach, determining whether κ logically implies ψ or, equivalently, whether $K_i\kappa$ logically implies $K_i\psi$, is co-*NP*-complete. We show now that the nonstandard approach can yield a more efficient algorithm.

Consider the query-evaluation problem from the nonstandard perspective. Is the problem of determining the consequences of a knowledge base in the nonstandard approach any easier than in the standard approach? It turns out that, just as in the standard case, in the nonstandard approach, determining whether κ logically implies ψ is equivalent to determining whether $K_i\kappa$ logically implies $K_i\psi$ and both problems are still co-*NP*-complete (see Exercises 9.25 and 9.26). There is, however, an important special case where using the nonstandard semantics does make the problem easier.

Define a *literal* to be a primitive proposition p or its negation $\neg p$, and a *clause* to be a disjunction of literals. For example, a typical clause is $p \vee \neg q \vee r$. A formula that is a conjunction of clauses is said to be in *conjunctive normal form* (*CNF*). We assume here that κ (the formula that characterizes the KB's knowledge) is in CNF. This is not so unreasonable in practice, since once we have a knowledge base in CNF, it is easy to maintain it in CNF; before telling a new fact to the KB, we simply convert it to CNF. If φ^{CNF} is the result of converting φ to CNF, then the result of adding φ to κ is $\kappa \wedge \varphi^{CNF}$. Note that $\kappa \wedge \varphi^{CNF}$ is in CNF if κ is. Every

standard propositional formula is equivalent to a formula in CNF (this is true even in our nonstandard semantics; see Exercise 9.27), so the transformation from φ to φ^{CNF} can always be carried out. Now, in general, this transformation can result in an exponential blowup; i.e., the length of φ^{CNF} can be exponential in the length of φ. We typically expect each fact φ that the KB is told to be small relative to the size of the KB, so even this exponential blowup is not unreasonable in practice. (On the other hand, it would not be reasonable to convert to CNF a whole knowledge base that had not been maintained in CNF.) For similar reasons, we can safely assume that the query ψ has been transformed to CNF.

Let us now reconsider the query evaluation problem, where both the KB and the query are in CNF. Under the standard semantics, the problem is no easier than the general problem of logical implication in propositional logic, that is, co-*NP*-complete (Exercise 9.28). By contrast, the problem is feasible under the nonstandard semantics.

Theorem 9.3.6 *There is a polynomial-time algorithm for deciding whether κ logically implies ψ with respect to $\mathcal{NM}_n$ for CNF formulas κ and ψ.*

Proof We say that clause α_1 *includes* clause α_2 if every literal that is a disjunct of α_2 is also a disjunct of α_1. For example, the clause $p \vee \neg q \vee \neg r$ includes the clause $p \vee \neg q$. We can now characterize when κ logically implies ψ with respect to nonstandard propositional logic, for CNF formulas κ and ψ.

Let κ and ψ be propositional formulas in CNF. We claim that κ logically implies ψ with respect to nonstandard propositional logic iff every clause of ψ includes a clause of κ. (This claim is false in standard propositional logic. For example, let κ be $q \vee \neg q$, and let ψ be $p \vee \neg p$. Then $\kappa \Rightarrow \psi$ is valid, but the single clause $p \vee \neg p$ of ψ does not include the single clause of κ.)

The "if" direction, which is fairly straightforward, is left to the reader (Exercise 9.29). We now prove the other direction. Assume that some clause α of ψ includes no clause of κ. We need only show that there is a nonstandard structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, {}^*)$ and state $s \in S$ such that $(M, s) \models \kappa$ but $(M, s) \not\models \psi$. Let S contain precisely two states s and t, and let $s^* = t$. Define $\pi(s)(p) =$ **false** iff p is a disjunct of α, and $\pi(t)(p) =$ **true** iff $\neg p$ is a disjunct of α, for each primitive proposition p. The $\mathcal{K}_i$'s are arbitrary. We now show that $(M, s) \not\models \alpha'$, for each disjunct α' of α. Notice that α' is a literal. If α' is a primitive proposition p, then $\pi(s)(p) =$ **false**, so $(M, s) \not\models \alpha'$; if α' is $\neg p$, where p is a primitive proposition, then $\pi(t)(p) =$ **true**, so $(M, t) \models p$, so again $(M, s) \not\models \alpha'$. Hence, $(M, s) \not\models \alpha$. Since α is one of the conjuncts of ψ, it follows that $(M, s) \not\models \psi$. We next show that

$(M, s) \models \beta$ if α does not include β. For if α does not include β, then there is some literal β' that is a disjunct of β but not of α. It is easy to see that $(M, s) \models \beta'$, and hence that $(M, s) \models \beta$. It follows that $(M, s) \models \kappa$, since by assumption, α does not include any of the conjuncts of κ.

It is clear that this characterization of nonstandard implication gives us a polynomial-time decision procedure for deciding whether one CNF formula implies another in the nonstandard approach. ∎

Theorem 9.3.6 gives us a real payoff of the nonstandard approach. It shows that even though the nonstandard approach does not improve the complexity of reasoning about knowledge in general, there are practical applications for which reasoning about knowledge can be feasible. As the following proposition shows, this also has implications for reasoning in standard logic.

Proposition 9.3.7 *Let κ and ψ be propositional formulas in CNF. If κ logically implies ψ with respect to $\mathcal{NM}_n$, then κ logically implies ψ with respect to standard propositional logic.*

Proof See Exercise 9.30. ∎

Theorem 9.3.6 and Proposition 9.3.7 yield an efficient algorithm for the evaluation of a CNF query ψ with respect to a CNF knowledge base κ: answer "Yes" if κ logically implies ψ with respect to $\mathcal{NM}_n$. By Theorem 9.3.6, logical implication of CNF formulas with respect to $\mathcal{NM}_n$ can be checked in polynomial time. Proposition 9.3.7 implies that any positive answer we obtain from testing logical implication between CNF formulas in nonstandard semantics will provide us with a correct positive answer for standard semantics as well. This means that even if we are ultimately interested only in conclusions that are derivable from standard reasoning, we can safely use the positive conclusions we obtain using nonstandard reasoning. Thus, the nonstandard approach yields a feasible query-answering algorithm for knowledge bases. Notice that the algorithm need not be correct with respect to negative answers. It is possible that κ does not logically imply ψ with respect to $\mathcal{NM}_n$, even though κ logically implies ψ with respect to standard propositional logic (see Exercise 9.30).

9.3.4 Discussion

The goal of our approach in this section was to gain some control over logical omniscience rather than to eliminate it. To this end, we tried to decouple the knowledge part of the semantics from its propositional part by keeping the definition of knowledge as truth in all possible worlds but changing the underlying notion of truth (the

propositional semantics). With this approach, we still have closure under logical implication. Since knowledge is still defined as truth in all possible worlds, it is still the case that if φ logically implies ψ, then an agent that knows φ will also know ψ. Nevertheless, as a result of the change in the definition of truth, the notion of logical implication has changed. It may not be so unreasonable for an agent's knowledge to be closed under logical implication if we have a weaker notion of logical implication. As a particular example of this approach, we considered a nonstandard logic in which the truth values of φ and $\neg\varphi$ are independent, and logical implication is captured using $\hookrightarrow$ rather than $\Rightarrow$. While this particular nonstandard approach does not improve the complexity of reasoning about knowledge in general, we gave one application where it does yield a significant improvement.

We should stress that we considered only one particular nonstandard logic in this section. Many nonstandard propositional logics have been studied. (See the notes at the end of the chapter for references.) It would be interesting to explore how these other nonstandard propositional logics could be combined with epistemic operators, and what the consequences of doing so would be.

9.4 Impossible Worlds

Logical omniscience arises from considering knowledge as truth in all possible worlds. In the previous section, we modified logical omniscience by changing the notion of truth. In this section, we modify logical omniscience by changing the notion of possible world. The idea is to augment the possible worlds by *impossible worlds*, where the customary rules of logic do not hold. For example, we may have both φ and ψ holding in an impossible world without having $\varphi \wedge \psi$ hold in that world. Even though these worlds are logically impossible, the agents nevertheless may consider them possible. Unlike our approach in the previous section, where nonstandard worlds had the same status as standard worlds, under the current approach the impossible worlds are only a figment of the agents' imagination; they serve only as epistemic alternatives. Thus, logical implication and validity are determined solely with respect to the standard worlds.

Formally, an *impossible-worlds structure* M is a tuple $(S, W, \sigma, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is a Kripke frame, $W \subseteq S$ is the set of *possible* states or worlds, and σ is a syntactic assignment (recall that syntactic assignments assign truth values to all formulas in all states). We require that σ behaves standardly on possible states, that is, if $s \in W$, then

$\sigma(s)(\varphi \wedge \psi) = \textbf{true}$ iff $\sigma(s)(\varphi) = \textbf{true}$ and $\sigma(s)(\psi) = \textbf{true}$,

$\sigma(s)(\neg\varphi) = \textbf{true}$ iff $\sigma(s)(\varphi) = \textbf{false}$, and

$\sigma(s)(K_i\varphi) = \textbf{true}$ iff $\sigma(t)(\varphi) = \textbf{true}$ for all t such that $(s, t) \in \mathcal{K}_i$.

Note that σ can behave in an arbitrary way on the impossible states, i.e, the states in $S - W$. We use σ to define satisfaction in the obvious way: $(M, s) \models \varphi$ precisely when $\sigma(s)(\varphi) = \textbf{true}$.

As mentioned earlier, logical implication and validity are determined only with respect to possible states, that is, the states in W. Formally, a set Ψ of formulas *logically implies* the formula φ *with respect to impossible-worlds structures* if for each impossible-worlds structure $M = (S, W, \sigma, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ and possible state $s \in W$ we have that whenever $(M, s) \models \psi$ for all $\psi \in \Psi$, then $(M, s) \models \varphi$. Similarly, φ is *valid with respect to impossible-worlds structures* if for each impossible-worlds structure $M = (S, W, \sigma, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ and possible state $s \in W$ we have that $(M, s) \models \varphi$.

Since agents consider the impossible states when determining their knowledge, but impossible states are not considered when determining logical implication, logical omniscience need not hold. Consider, for example, full logical omniscience. Suppose that an agent knows all formulas in Ψ, and Ψ logically implies φ. Since the agent knows all formulas in Ψ, all formulas in Ψ must hold in all the states that the agent considers epistemically possible. But in an impossible state, φ may fail even though Ψ holds. The reason for this is that logical implication is determined by us, rational logicians, for whom impossible states are simply impossible and are therefore not taken into account. Thus, the agent need not know φ, since φ may fail to hold in some impossible state that the agent considers possible.

The impossible-worlds approach is very general; it can capture different properties of knowledge by imposing certain conditions on the behavior of syntactic assignment σ in the impossible states. For example, to capture closure under conjunction we have to demand that in an impossible state if both φ and ψ are true, then $\varphi \wedge \psi$ is also true. (See also Exercise 9.31.)

We now consider one instance of the impossible-worlds approach, which will enable us to contrast the impossible-worlds approach with the nonstandard-logic approach of Section 9.3. Essentially, the idea is to view nonstandard structures as impossible-worlds structures, where the nonstandard states are the impossible worlds. Recall that nonstandard structures are Kripke structures with a * function. This function associates with a state s an adjunct state s^*. If $s = s^*$, then s is a standard state and therefore a possible world. If $s \neq s^*$, then s and s^* are nonstandard states and therefore considered to be impossible worlds. More formally, given a nonstandard

structure $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$, we can identify it with the impossible-worlds structure $M' = (S, W, \sigma, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where W is the set of standard states, i.e., the states s such that $s^* = s$, and, for all states $s \in S$, we have that $\sigma(s)(\varphi) = \mathbf{true}$ iff $(M, s) \models \varphi$. We can therefore view a nonstandard structure M as implicitly defining the impossible-worlds structure M' obtained by this translation. We shall abuse language slightly and say that we view M as an impossible-worlds structure. When M is viewed as a nonstandard structure, the distinction between standard and nonstandard states does not play any role. In contrast, when M is viewed as an impossible-worlds structure, the standard states have a special status. Intuitively, although an agent (who is not a perfect reasoner) might consider nonstandard states possible (where, for example, $p \wedge \neg p$ or $K_i p \wedge \neg K_i p$ holds), we do not consider such states possible; surely in the real world a formula is either true or false, but not both.

Nonstandard structures can be viewed both from the perspective of the nonstandard-logic approach and from the perspective of the impossible-worlds approach. When we view a nonstandard structure as an impossible-worlds structure, we consider nonstandard states to be impossible states, and thus consider a formula φ to be valid if it is true in all of the possible states, that is, in all of the standard states. Formally, define a formula of $\mathcal{L}_n$ to be *standard-state valid* if it is true at every standard state of every nonstandard structure. The definition for *standard-state logical implication* is analogous.

We demonstrate the difference between logical implication and standard-state logical implication by reconsidering the knowledge base example discussed in Section 9.3.3, where the knowledge base is characterized by the formula κ and the query is the formula φ. We saw in Section 9.3.3 that in the nonstandard approach, φ is a consequence of κ precisely when knowledge of φ is a consequence of knowledge of κ. This is not the case in the impossible-worlds approach; it is possible to find φ_1 and φ_2 in $\mathcal{L}_n$ such that φ_1 standard-state logically implies φ_2, but $K_i\varphi_1$ does not standard-state logically imply $K_i\varphi_2$ (Exercise 9.32). The reason for this difference is that φ_1's standard-state logically implying φ_2 deals with logical implication in standard states, whereas $K_i\varphi_1$'s standard-state logically implying $K_i\varphi_2$ deals with logical implication in states agents consider possible, which can include nonstandard states. Interestingly, logical implication of knowledge formulas coincides in the nonstandard approach and the impossible-worlds approach; that is, $K_i\varphi_1$ standard-state logically implies $K_i\varphi_2$ iff $K_i\varphi_1$ logically implies $K_i\varphi_2$ with respect to $\mathcal{NM}_n$ (Exercise 9.33).

The reader may recall that under the nonstandard semantics, $\Rightarrow$ behaves in a nonstandard way. In particular, $\Rightarrow$ does not capture the notion of logical implication. In fact, that was part of the motivation for the introduction of strong implication. In

standard states, however, $\Rightarrow$ and $\hookrightarrow$ coincide; that is, $\varphi_1 \Rightarrow \varphi_2$ holds at a standard state precisely if $\varphi_1 \hookrightarrow \varphi_2$ holds. It follows that even though $\Rightarrow$ does not capture logical implication, it does capture standard-state logical implication. The following analogue to Proposition 9.3.3 is immediate.

Proposition 9.4.1 *Let φ_1 and φ_2 be formulas in $\mathcal{L}_n$. Then φ_1 standard-state logically implies φ_2 iff $\varphi_1 \Rightarrow \varphi_2$ is standard-state valid.*

The main feature of the impossible-worlds approach is the fact that knowledge is evaluated with respect to all states, while logical implication is evaluated only with respect to standard states. As a result, we avoid logical omniscience. For example, an agent does not necessarily know valid formulas of standard propositional logic. Although the classical tautology $\varphi \vee \neg\varphi$ is standard-state valid, $K_i(\varphi \vee \neg\varphi)$ may not hold at a standard state s, since agent i might consider an incomplete state possible. (Recall that in an incomplete state of a nonstandard structure both φ and $\neg\varphi$ may fail to hold.) On the other hand, as we now show, incompleteness is all that prevents an agent from knowing valid formulas. In particular, we show that if an agent knows that the state is complete, then he does know all tautologies.

What does it mean for an agent to know that a state is complete? Let φ be a propositional formula that contains precisely the primitive propositions $p_1, \ldots, p_k$. Define *complete*(φ) to be the formula

$$(p_1 \vee \neg p_1) \wedge \ldots \wedge (p_k \vee \neg p_k).$$

Thus, *complete*(φ) is true at a state s precisely if s is complete as far as all the primitive propositions in φ are concerned. In particular, if *complete*(φ) is true at s, then s is complete with respect to φ (see Exercise 9.34). Thus, if an agent knows *complete*(φ), then he knows that he is in a state that is complete with respect to φ.

The following result makes precise our earlier claim that incompleteness is all that prevents an agent from knowing tautologies.

Theorem 9.4.2 *Let φ be a tautology of standard propositional logic. Then K_i(complete(φ)) $\Rightarrow K_i\varphi$ is standard-state valid.*

Proof By Exercise 9.35 (see also Exercise 9.36), *complete*(φ) logically implies φ with respect to $\mathcal{N}\mathcal{M}_n$. From Exercise 9.25 it follows that K_i(*complete*(φ)) logically implies $K_i\varphi$ with respect to $\mathcal{N}\mathcal{M}_n$. In particular, K_i(*complete*(φ)) standard-state logically implies $K_i\varphi$. It follows by Proposition 9.4.1 that K_i(*complete*(φ)) $\Rightarrow K_i\varphi$ is standard-state valid. ∎

In addition to the failure of knowledge of valid formulas, another form of logical omniscience that fails under the impossible-worlds approach is closure under logical implication: the formula $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$ is not standard-state valid (Exercise 9.37). This lack of closure results from considering incoherent states possible: indeed, $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i(\psi \vee (\varphi \wedge \neg\varphi))$ is standard-state valid (Exercise 9.37). That is, if an agent knows that φ holds and also knows that $\varphi \Rightarrow \psi$ holds, then she knows that either ψ holds or the state is incoherent. This observation generalizes. As we now show, as long as the agent knows that the state is coherent, then her knowledge is closed under logical implication.

Recall that *true* is an abbreviation for some fixed nonstandard tautology such as $p \hookrightarrow p$, and *false* is an abbreviation for $\neg$*true*. We use the fact that the formula $\varphi \hookrightarrow$ *false* asserts the falsehood of φ (see Exercise 9.23). Let φ be a formula that contains precisely the primitive propositions $p_1, \ldots, p_k$. Define *coherent*(φ) to be the formula

$$\big((p_1 \wedge \neg p_1) \hookrightarrow \mathit{false}\big) \wedge \ldots \wedge \big((p_k \wedge \neg p_k) \hookrightarrow \mathit{false}\big).$$

Thus, *coherent*(φ) is true at a state s precisely if s is coherent as far as the primitive propositions in φ are concerned. In particular, if *coherent*(φ) holds at s, then s is coherent with respect to φ. (Note that *coherent*(φ) is not definable in $\mathcal{L}_n$ but only in $\mathcal{L}_n^{\hookrightarrow}$; see Exercise 9.38.) Knowledge of coherence implies that knowledge is closed under material implication.

Theorem 9.4.3 *Let φ and ψ be standard propositional formulas. Then $(K_i(\mathit{coherent}(\varphi)) \wedge K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$ is standard-state valid.*

Proof Denote $K_i(\mathit{coherent}(\varphi)) \wedge K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)$ by τ. By Proposition 9.4.1, it is sufficient to show that τ standard-state logically implies $K_i\psi$. We shall show the stronger fact that τ logically implies $K_i\psi$. Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, {}^*)$ be a nonstandard structure, and s a state of M. Assume that τ is true at s, and that $(s, t) \in \mathcal{K}_i$. So *coherent*(φ) is true at t. By a straightforward induction on the structure of formulas, we can show that for every propositional formula γ all of whose primitive propositions are contained in φ, it is not the case that both γ and $\neg\gamma$ are true at t. Now φ and $\varphi \Rightarrow \psi$ are both true at t, since $K_i\varphi$ and $K_i(\varphi \Rightarrow \psi)$ are true at s. Since φ is true at t, it follows from what we just showed that $\neg\varphi$ is not true at t. Since $\varphi \Rightarrow \psi$ is an abbreviation for $\neg\varphi \vee \psi$, it follows that ψ is true at t. Hence, $K_i\psi$ is true at s. ∎

Theorems 9.4.2 and 9.4.3 explain why agents are not logically omniscient: when we view nonstandard structures as impossible-worlds structures, "logically" is defined with respect to standard states, but the agents may consider nonstandard

states possible. If an agent considers only standard states possible, so that both $K_i(complete(\varphi))$ and $K_i(coherent(\varphi))$ hold, then by Theorems 9.4.2 and 9.4.3, this agent is logically omniscient (more accurately, he knows every tautology of standard propositional logic and his knowledge is closed under material implication).

9.5 Awareness

In Section 9.2, we described syntactic and semantic approaches to dealing with omniscience. In Section 9.4, we described what can be viewed as a mixed approach, i.e., an approach that has both semantic and syntactic components: in impossible-worlds structures, truth is defined semantically in the possible states and syntactically in the impossible states. We now describe another approach that has both semantic and syntactic components.

The underlying idea is that it is necessary to be *aware* of a concept before one can have beliefs about it. One cannot know something of which one is unaware. Indeed, how can someone say that he knows or doesn't know about p if p is a concept of which he is completely unaware? One can imagine the puzzled response of someone not up on the latest computer jargon when asked if he knows that the price of SIMMs is going down! (For the benefit of the reader who is not fluent in the computer-speak of the early 1990's, a SIMM is a Single In-line Memory Module, a basic component in current-day computer memories.) In fact, even a sentence such as "He doesn't even know that he doesn't know p!" is often best understood as saying "He's not even *aware* that he doesn't know p."

In this section we augment the possible-worlds approach with a syntactic notion of *awareness*. This will be reflected in the language by a new modal operator A_i for each agent i. The intended interpretation of $A_i\varphi$ is "i is aware of φ." We do not wish to attach any fixed cognitive meaning to the notion of awareness; $A_i\varphi$ may mean "i is familiar with all the propositions mentioned in φ," "i is able to figure out the truth of φ," or perhaps "i is able to compute the truth of φ within time T." (We return to a computational notion of knowledge later in this chapter and also in Chapter 10.) The power of the approach comes from the flexibility of the notion of awareness.

To represent the knowledge of agent i, we allow two modal operators K_i and X_i, standing for *implicit knowledge* and *explicit knowledge* of agent i, respectively. Implicit knowledge is the notion we have been considering up to now: truth in all worlds that the agent considers possible. On the other hand, an agent explicitly knows a formula φ if he is aware of φ and implicitly knows φ. Intuitively, an agent's implicit

knowledge includes all the logical consequences of his explicit knowledge. We denote by $\mathcal{L}_n^A(\Phi)$, or $\mathcal{L}_n^A$ for short, the set of formulas obtained by enlarging $\mathcal{L}_n(\Phi)$ to include the new modal operators A_i and X_i.

An *awareness structure* is a tuple $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$, where the tuple $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is a Kripke structure and $\mathcal{A}_i$ is a function associating a set of formulas with each state, for $i = 1, \ldots, n$. Intuitively, $\mathcal{A}_i(s)$ is the set of formulas that agent i is aware of at state s. The awareness functions $\mathcal{A}_i$ form the syntactic component of the semantics. The formulas in $\mathcal{A}_i(s)$ are those that the agent is "aware of," not necessarily those he knows. The set of formulas that the agent is aware of can be arbitrary. It is possible for both φ and $\neg\varphi$ to be in $\mathcal{A}_i(s)$, for only one of φ and $\neg\varphi$ to be in $\mathcal{A}_i(s)$, or for neither φ nor $\neg\varphi$ to be in $\mathcal{A}_i(s)$. It is also possible, for example, that $\varphi \vee \psi$ is in $\mathcal{A}_i(s)$ but $\psi \vee \varphi$ is not in $\mathcal{A}_i(s)$.

The semantics for primitive propositions, conjunctions, negations, and for formulas $K_i\varphi$ is just as for standard Kripke structures. We only need to add new clauses for formulas of the form $A_i\varphi$ and $X_i\varphi$:

$(M, s) \models A_i\varphi$ iff $\varphi \in \mathcal{A}_i(s)$

$(M, s) \models X_i\varphi$ iff $(M, s) \models A_i\varphi$ and $(M, t) \models K_i\varphi$

The first clause states that agent i is aware of φ at state s exactly if φ is in $\mathcal{A}_i(s)$. The second clause states that agent i explicitly knows φ iff (1) agent i is aware of φ, and (2) agent i implicitly knows φ (i.e., φ is true in all the worlds he considers possible). We see immediately that $X_i\varphi \Leftrightarrow A_i\varphi \wedge K_i\varphi$ is valid. You cannot have explicit knowledge about formulas of which you are not aware! If we assume that agents are aware of all formulas, then explicit knowledge reduces to implicit knowledge.

By definition, the implicit-knowledge operator K_i behaves just as it does in a Kripke structure. Thus, as in Chapter 3, implicit knowledge is closed under material implication (that is, $(K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$ is valid) and $K_i\varphi$ is valid for every valid formula φ. The explicit-knowledge operator X_i, however, may behave differently. Agents do not explicitly know all valid formulas; for example, $\neg X_i(p \vee \neg p)$ is satisfiable, because the agent might not be aware of the formula $p \vee \neg p$. Also, an agent's explicit knowledge is not necessarily closed under material implication; $X_i p \wedge X_i(p \Rightarrow q) \wedge \neg X_i q$ is satisfiable, because i might not be aware of q. Since awareness is essentially a syntactic operator, this approach shares some of the features of the syntactic approach. For example, order of presentation matters; there is no reason to suppose that the formula $X_i(\varphi \vee \psi)$ is equivalent to $X_i(\psi \vee \varphi)$, since $A_i(\varphi \vee \psi)$ might hold without $A_i(\psi \vee \varphi)$ holding. A computer program that can determine in time T whether $\varphi \vee \psi$ follows from some initial premises might not

be able to determine in time T whether $\psi \vee \varphi$ follows from those premises. (The program might work on, say, the left disjunct first, and be able to determine quickly that φ is true, but get stuck working on ψ.) And people do *not* necessarily identify formulas such as $\varphi \vee \psi$ and $\psi \vee \varphi$. The reader can validate the idea that the order matters by computing the product $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 0$.

Up to now we have placed no restrictions on the set of formulas that an agent may be aware of. Once we have a concrete interpretation in mind, we may well want to add some restrictions to the awareness function to capture certain types of "awareness." The clean separation in our framework between knowledge (captured by the binary relations $\mathcal{K}_i$) and awareness (captured by the syntactic functions $\mathcal{A}_i$) makes this easy to do. Some typical restrictions we may want to add to $\mathcal{A}_i$ include the following:

- Awareness could be closed under subformulas; i.e., if $\varphi \in \mathcal{A}_i(s)$ and ψ is a subformula of φ, then $\psi \in \mathcal{A}_i(s)$. Note that this makes sense if we are reasoning about a computer program that will never compute the truth of a formula unless it has computed the truth of all its subformulas. But it is also easy to imagine a program that knows that $\varphi \vee \neg\varphi$ is true without needing to compute the truth of φ. Perhaps a more reasonable restriction is simply to require that if $\varphi \wedge \psi \in \mathcal{A}_i(s)$ then both $\varphi, \psi \in \mathcal{A}_i(s)$ (see Exercise 9.39).

- Agent i might be aware of only a certain subset of the primitive propositions, say Ψ. In this case we could take $\mathcal{A}_i(s)$ to consist of exactly those formulas that mention only primitive propositions that appear in Ψ.

- A self-reflective agent will be aware of what he is aware of. Semantically, this means that if $\varphi \in \mathcal{A}_i(s)$, then $A_i\varphi \in \mathcal{A}_i(s)$. This corresponds to the axiom $A_i\varphi \Rightarrow A_iA_i\varphi$.

- Similarly, an agent might know of which formulas he is or is not aware. Semantically, this means that if $(s, t) \in \mathcal{K}_i$, then $\mathcal{A}_i(s) = \mathcal{A}_i(t)$. This corresponds to the axioms $A_i\varphi \Rightarrow K_iA_i\varphi$ and $\neg A_i\varphi \Rightarrow K_i\neg A_i\varphi$. This restriction holds when the set of formulas that an agent is aware of is a function of his local state. It also holds when awareness is generated by a subset of primitive propositions, as discussed previously.

We now turn to examining the properties of knowledge in this logic. It is easy to see that the axiom system K_n is sound, since the semantics of K_i has not changed. Indeed, we can obtain a sound and complete axiomatization simply by adding the

axiom $X_i\varphi \Leftrightarrow (A_i\varphi \wedge K_i\varphi)$ to $\mathbf{K}_n$ (Exercise 9.40). These axioms, however, do not give us much insight into the properties of explicit knowledge.

In fact, despite the syntactic nature of the awareness operator, explicit knowledge retains many of the same properties as implicit knowledge, once we relativize to awareness. For example, corresponding to the Distribution Axiom

$$\big(K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)\big) \Rightarrow K_i\psi$$

we have

$$\big(X_i\varphi \wedge X_i(\varphi \Rightarrow \psi) \wedge A_i\psi\big) \Rightarrow X_i\psi$$

(see Exercise 9.41). Thus, if you explicitly know φ and $\varphi \Rightarrow \psi$, then you will explicitly know ψ *provided* you are aware of ψ. Similarly, corresponding to the Knowledge Generalization Rule, we have

$$\text{from } \varphi \text{ infer } A_i\varphi \Rightarrow X_i\varphi$$

(see Exercise 9.41). That is, you explicitly know a valid formula if you are aware of it. Note the similarity between this rule and Theorem 9.4.2, which says that, in the impossible-worlds approach, knowledge of a tautology φ follows from knowledge of $complete(\varphi)$. In that setting, we can think of $K_i(p \vee \neg p)$ as saying that agent i is aware of p. If we take the view of awareness as being generated by a subset of primitive propositions, then $K_i(complete(\varphi))$ can be thought of saying that agent i is aware of φ. Thus, Theorem 9.4.2 can be viewed as saying that an agent knows a tautology if he is aware of it. In both cases, an agent must be aware of the relevant formula in order to know it explicitly.

As we saw earlier, we can capture certain properties of knowledge by imposing the appropriate conditions on the $\mathcal{K}_i$ relations. For example, if we assume that $\mathcal{K}_i$ is reflexive, then, as before, we obtain the axiom $X_i\varphi \Rightarrow \varphi$, since $X_i\varphi \Rightarrow K_i\varphi$ is valid, and reflexivity of $\mathcal{K}_i$ entails that $K_i\varphi \Rightarrow \varphi$ is valid. It may be tempting to think that if $\mathcal{K}_i$ is an equivalence relation, then we obtain the introspection axioms $X_i\varphi \Rightarrow X_iX_i\varphi$ and $\neg X_i\varphi \Rightarrow X_i\neg X_i\varphi$. It is easy to verify, however, that this is not the case. In fact, even the obvious modification of the introspection axioms, where an agent must be aware of a formula before she explicitly knows it, fails to hold:

$$(X_i\varphi \wedge A_iX_i\varphi) \Rightarrow X_iX_i\varphi$$
$$\big(\neg X_i\varphi \wedge A_i(\neg X_i\varphi)\big) \Rightarrow X_i\neg X_i\varphi$$

(see Exercise 9.42). The reason for this failure is the independence of the the awareness operator and the possibility relation; an agent may be aware of different formulas

in states that she considers to be equivalent. It may be reasonable to assume that if an agent cannot distinguish between two states, then she is aware of the same formulas in both states, i.e., $(s, t) \in \mathcal{K}_i$ implies $\mathcal{A}_i(s) = \mathcal{A}_i(t)$. Intuitively, this means that the agent knows of which formulas she is aware. If this assumption holds and if $\mathcal{K}_i$ is an equivalence relation, then the modified introspection properties mentioned earlier hold (see Exercise 9.42). The phenomenon described by these axioms is similar to the phenomenon of the previous section, where knowledge of completeness or coherence was required. The first of the two axioms suggests how, as in the quotation from de Chardin at the beginning of the chapter, an animal may know, but not know that it knows: it might not be aware of its knowledge. The second axiom suggests how someone can fail to be conscious of his ignorance. By contrast, the Spinoza quotation suggests that people are aware of their knowledge. Of course, we can construct axioms analogous to these even if we do not assume that an agent knows what formulas she is aware of, although they are not quite as elegant (see Exercise 9.42).

As we observed earlier, if we are reasoning about a computer program that will never compute the truth of a formula unless it has computed the truth of all its subformulas, then awareness is closed under subformulas: if $\varphi \in \mathcal{A}_i(s)$ and ψ is a subformula of φ, then $\psi \in \mathcal{A}_i(s)$. Taking awareness to be closed under subformulas has some interesting consequences. First note that this property can be captured axiomatically by the following axioms:

$$
\begin{aligned}
A_i(\neg\varphi) &\Rightarrow A_i\varphi \\
A_i(\varphi \wedge \psi) &\Rightarrow (A_i\varphi \wedge A_i\psi) \\
A_i(X_j\varphi) &\Rightarrow A_i\varphi \\
A_i(K_j\varphi) &\Rightarrow A_i\varphi \\
A_i(A_j\varphi) &\Rightarrow A_i\varphi.
\end{aligned}
$$

(By changing $\Rightarrow$ to $\Leftrightarrow$ in these axioms, we can capture a notion of awareness generated by a set of primitive propositions; see Exercise 9.43.)

Although agents still do not explicitly know all valid formulas if awareness is closed under subformulas, an agent's knowledge is then closed under material implication; i.e., $X_i\varphi \wedge X_i(\varphi \Rightarrow \psi) \Rightarrow X_i\psi$ is then valid (see Exercise 9.44). Thus, the seemingly innocuous assumption that awareness is closed under subformulas has a rather powerful impact on the properties of explicit knowledge. Certainly this assumption is inappropriate for resource-bounded notions of awareness, where awareness of φ corresponds to being able to compute the truth of φ. As we remarked, it may be easy to see that $\varphi \vee \neg\varphi$ is a tautology without having to compute whether

either φ or $\neg\varphi$ follows from some information. Nevertheless, this observation shows that there are some natural interpretations of awareness and explicit knowledge (for example, an interpretation of awareness that is closed under subformulas and an interpretation of explicit knowledge that is not closed under material implication) that cannot be simultaneously captured in this framework.

What about the computational complexity of the validity problem? Clearly the addition of A_i and X_i cannot decrease the complexity. It turns out that this addition does not increase the complexity; the validity problem is still *PSPACE*-complete.

In summary, the awareness approach is very flexible and general. In a natural and appealing way, it can be used to demonstrate why various types of logical omniscience fail, and to give assumptions on what the agent must be aware of for these various types of logical omniscience to hold. It gains this flexibility through the use of a syntactic awareness operator. While at first this may seem to put us right back into the syntactic approach of Section 9.2, by isolating the syntactic component, we have more structure to study, while maintaining our intuition about knowledge being truth in all possible worlds.

This observation suggests that we focus on natural notions of awareness. We considered some notions already in this section. In Chapter 10, we describe a computational model of knowledge, which can be viewed as using a computational notion of awareness.

It is interesting to relate the awareness approach to the impossible-worlds approach. Both mix syntax and semantics, but in a very different way. In the impossible-worlds approach, knowledge depends on impossible worlds, where truth is defined syntactically. In the awareness approach, knowledge depends on awareness, which is defined syntactically. It turns out that in some sense the two approaches are equivalent; every impossible-worlds structure can be represented by an awareness structure and vice versa (see Exercise 9.45); thus, both approaches can be used to model the same situations.

9.6 Local Reasoning

An important difference between an idealized model of knowledge (such as a Kripke structure) and the knowledge of people in the real world is that in the real world people have inconsistent knowledge. That is, they may believe both φ and $\neg\varphi$ for some formula φ; this may happen when an agent believes both φ and $\neg\psi$ without realizing that φ and ψ are logically equivalent. We already have tools to model inconsistent knowledge: it is possible for an agent to believe both φ and $\neg\varphi$ in

a standard Kripke structure. Standard Kripke structures, however, provide a poor model for inconsistent knowledge. It is easy to see that the only way that an agent i in a standard Kripke structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ can have inconsistent knowledge in a state s is for $\mathcal{K}_i(s) = \{t \mid (s, t) \in \mathcal{K}_i\}$ to be empty, which implies that in state s, agent i knows *every* formula. Some of the approaches described earlier in this chapter can also be used to model inconsistent knowledge. For example, in an awareness structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$, it is possible for agent i to have contradictory explicit knowledge in state s without having explicit knowledge of every formula: this can be modeled by again letting $\mathcal{K}_i(s)$ be empty and taking $\mathcal{A}_i(s)$ to consist precisely of those formulas of which agent i has explicit knowledge.

In this section we describe another approach, in which inconsistent knowledge arises in a very natural way. Since this approach seems to be especially interesting when the agents are people, we describe the results in this section in these terms.

One reason that people have inconsistent knowledge is that knowledge tends to depend on an agent's frame of mind. We can view an agent as a society of minds, each with its own knowledge. The members of the society may have contradictory knowledge (or, perhaps better, beliefs). For example, in one frame of mind, a politician might believe in the importance of a balanced budget. In another frame of mind, however, he might believe it is necessary to greatly increase spending. This phenomenon seems to occur even in science. For example, the two great theories physicists reason with are the theory of quantum phenomena and the general theory of relativity. Some physicists work with both theories, even though they believe that the two theories might well be incompatible!

In Kripke structures, agents can be said to have a single frame of mind. We viewed $\mathcal{K}_i(s)$ as the set of states that agent i thinks possible in state s. In our next approach, there is not necessarily one set of states that an agent thinks possible, but rather a number of sets, each one corresponding to the knowledge of a different member of the society of minds. We can view each of these sets as representing the worlds the agent thinks possible in a given frame of mind, when he is focusing on a certain set of issues. This models agents with many frames of mind.

More formally, a *local-reasoning structure* is a tuple $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ where S is a set of states, $\pi(s)$ is a truth assignment to the primitive propositions for each state $s \in S$, and $\mathcal{C}_i(s)$ is a nonempty set of subsets of S. Intuitively, if $\mathcal{C}_i(s) = \{T_1, \ldots, T_k\}$, then in state s agent i sometimes (depending perhaps on his frame of mind or the issues on which he is focusing) considers the set of possible states to be precisely T_1, sometimes he considers the set of possible states to be precisely T_2, etc. Or, taking a more schizophrenic point of view, we could view each

of these sets as representing precisely the worlds that some member of the society in agent i's mind thinks possible.

We now interpret $K_i\varphi$ as "agent i knows φ in *some* frame of mind"; i.e., some member of the society of minds making up agent i at s knows φ. Note that although we are using the same symbol K_i, this notion is quite different from the notions of knowledge discussed earlier in this chapter. This form of knowledge could be called *local knowledge*, since it is local to one of the members of the society. The semantics for primitive propositions, conjunctions, and negations is just as for standard Kripke structures. The semantics for knowledge, however, has changed:

$(M, s) \models K_i\varphi$ iff there is *some* $T \in \mathcal{C}_i(s)$ such that $(M, t) \models \varphi$ for all $t \in T$.

There is a stronger notion of knowledge where we would say that i knows φ if φ is known in *all* of i's frames of mind. Under the society of minds viewpoint, our notion of K_i corresponds to "some member (of agent i's society) knows," whereas this stronger notion corresponds to "all members know." We can get an even stronger notion by having i know φ only if φ is common knowledge among i's frames of mind. Going in the other direction, towards weaker notions of knowledge, there is a notion of distributed knowledge (among the frames of mind of agent i) analogous to that considered in Chapter 2. We do not pursue these directions here; Exercise 9.46 deals with the notion of distributed knowledge among the frames of mind.

Note that an agent may hold inconsistent knowledge in a local-reasoning structure: $K_i p \wedge K_i \neg p$ is satisfiable, since in one frame of mind agent i might know p, while in another he might know $\neg p$. In fact, $K_i(\mathit{false})$ is even possible: this will be true at state s if one of the sets in $\mathcal{C}_i(s)$ is the empty set. There is quite a difference between having inconsistent knowledge (that is, $K_i\varphi \wedge K_i\neg\varphi$) and knowing a contradiction (that is, $K_i(\varphi \wedge \neg\varphi)$). In the approach of this section, these are not equivalent. One can imagine a situation where contradictory statements φ and $\neg\varphi$ can both be known: this might correspond to having received contradictory information. It is harder to imagine knowing a contradictory statement $\varphi \wedge \neg\varphi$. Knowing contradictory statements can be forbidden (while still allowing the possibility of having inconsistent knowledge) by simply requiring that each set in each $\mathcal{C}_i(s)$ be nonempty.

If $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ is a local-reasoning structure, and if $\mathcal{C}_i(s)$ is a singleton set for each state s, say $\mathcal{C}_i(s) = \{T_i^s\}$, then M is equivalent to a Kripke structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where $(s, t) \in \mathcal{K}_i$ exactly if $t \in T_i^s$ (see Exercise 9.47).

Clearly, there is a formal similarity between local-reasoning structures and MS structures,though the philosophy and the semantics are quite different. It is instructive to compare the two approaches. The Montague-Scott approach is more general than

the local reasoning approach. In fact, if we were to take a local-reasoning structure $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, and let $\mathcal{C}_i'$ be the set of all supersets of members of $\mathcal{C}_i$, then the MS structure $M' = (S, \pi, \mathcal{C}_1', \ldots, \mathcal{C}_n')$ is equivalent to the local-reasoning structure M, in the sense that $(M, s) \models \varphi$ iff $(M', s) \models \varphi$ (Exercise 9.48). Despite this formal embedding of local-reasoning structures in MS structures, we do not view the former as a special case of the latter. As we said earlier, the philosophy behind them is quite different. In Montague-Scott semantics, $\mathcal{C}_i(s)$ represents a set of propositions believed by i, while in local reasoning semantics $\mathcal{C}_i(s)$ represents the knowledge of each of the members of the society of minds. Thus the former is a model that explicitly represents knowledge, while the latter is a model for local reasoning.

What about logical omniscience? We already noted that closure under conjunction fails in the local reasoning semantics, since knowing φ and knowing $\neg\varphi$ is not equivalent to knowing $\varphi \wedge \neg\varphi$. It is easy to see that knowledge is not closed under material implication, but for different reasons than for the logics of the previous sections. The formula $K_i p \wedge K_i(p \Rightarrow q) \wedge \neg K_i q$ is satisfiable simply because in one frame of mind agent i might know p, in another he might know $p \Rightarrow q$, but he might never be in a frame of mind where he puts these facts together to conclude q. (See Exercise 9.49 to see how to guarantee closure under material implication.) We do have other types of omniscience: for example, in this approach, there is knowledge of valid formulas and closure under logical implication (Exercise 9.50). In fact, these two types of omniscience form the basis for a sound and complete axiomatization:

Theorem 9.6.1 *The following is a sound and complete axiomatization for validity with respect to local-reasoning structures:*

A1. All instances of tautologies of propositional logic

R1. From φ and $\varphi \Rightarrow \psi$ infer ψ (modus ponens)

R2. From φ infer $K_i\varphi$ (Knowledge of valid formulas)

R3. From $\varphi \Rightarrow \psi$ infer $K_i\varphi \Rightarrow K_i\psi$ (Closure under valid implication)

Proof See Exercise 9.51. ∎

The computational complexity of the satisfiability problem is only *NP*-complete, even if there are many agents. This contrasts with the complexity of the satisfiability problem for K_n, which is *PSPACE*-complete. This is essentially the phenomenon that we saw in Section 9.2.2, where in the absence of closure under conjunction the complexity of the satisfiability problem in MS structures is only *NP*-complete.

Theorem 9.6.2 *The satisfiability problem with respect to local-reasoning structures is NP-complete.*

Proof See Exercise 9.52. ∎

Just as we can impose conditions on the $\mathcal{K}_i$'s to capture various properties of knowledge, we can similarly impose conditions on the $\mathcal{C}_i$'s. We already noted that knowing contradictory statements can be forbidden (while still allowing the possibility of having inconsistent knowledge) by simply requiring that each set in each $\mathcal{C}_i(s)$ be nonempty (this is the analogue of the seriality condition of Section 3.1). We also gave a condition in Exercise 9.49 that guarantees closure under material implication. We now mention some other properties of knowledge that can be guaranteed by appropriate assumptions on the $\mathcal{C}_i$'s. By assuming that s is a member of every $T \in \mathcal{C}_i(s)$, we make $K_i\varphi \Rightarrow \varphi$ valid (this is the analogue to the reflexivity condition of Section 3.1). In Kripke structures, we capture positive and negative introspection by requiring the $\mathcal{K}_i$'s to be transitive and Euclidean, respectively. Here we can capture positive introspection by requiring that if $T \in \mathcal{C}_i(s)$ and $t \in T$, then $T \in \mathcal{C}_i(t)$. Intuitively, this says that in each frame of mind an agent considers it possible that he is in that frame of mind. We can capture negative introspection by requiring that if $T \in \mathcal{C}_i(s)$ and $t \in T$, then $\mathcal{C}_i(t) \subseteq \mathcal{C}_i(s)$. Intuitively, this says that in each frame of mind, the agent considers possible only the actual frames of mind. We note that these conditions are sufficient but not necessary. Exercises 9.53 and 9.54 deal with the conditions that we need to impose on the $\mathcal{C}_i$'s to capture various properties of knowledge.

A particularly interesting special case we can capture is one where in each frame of mind, an agent refuses to admit that he may occasionally be in another frame of mind. (This phenomenon can certainly be observed with people!) Semantically, we can capture this by requiring that if $T \in \mathcal{C}_i(s)$ and $s' \in T$, then $\mathcal{C}_i(s')$ is the singleton set $\{T\}$. This says that if an agent has a frame of mind T, then in every state in this frame of mind, he thinks that his only possible frame of mind is T. We call such agents *narrow-minded agents*.

A narrow-minded agent will believe he is consistent (even if he is not), since in a given frame of mind he refuses to recognize that he may have other frames of mind. Thus, $K_i(\neg(K_i\varphi \wedge K_i\neg\varphi))$ is valid in this case, even though $K_i\varphi \wedge K_i\neg\varphi$ is consistent (Exercise 9.55). Moreover, because an agent can do perfect reasoning within a given frame of mind, a narrow-minded agent will also believe that he is a perfect reasoner. Thus $K_i\big((K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi\big)$ is a valid formula in all local-reasoning structures with narrow-minded agents (Exercise 9.55).

9.7 Concluding Remarks

The motivation behind this chapter is the observation that the semantics of knowledge in Kripke structures presented in Chapters 2 and 3, while adequate (and very useful!) for many applications, simply does not work for all applications. In particular, logical omniscience of agents, which is inherent in the standard possible-worlds approach, is in many cases inappropriate.

Just as we do not feel that there is one right, true definition of knowledge that captures all the nuances of the use of the word in English, we also do not feel that there is a single semantic approach to deal with the logical-omniscience problem. Thus, in this chapter we suggested a number of different approaches to avoiding or alleviating the logical-omniscience problem. With the exception of the explicit-representation approach (using either syntactic structures or MS structures), all of our approaches try to maintain the flavor of the possible-worlds approach, with knowledge defined as truth in all possible worlds. Nevertheless, they embody quite different intuitions. The nonstandard approach concedes that agents do not know all the logical consequences of their knowledge, at least if we consider all the logical consequences in standard logic. The hope is that by moving to a nonstandard logic, the fact that an agent's knowledge is closed under logical consequence will become more palatable. The impossible-worlds approach, while formally quite similar to the nonstandard approach, takes the point of view that, although we, the modelers, may know that the world satisfies the laws of standard logic, the agent may be confused, and consider "impossible" worlds possible. The awareness approach adds awareness as another component of knowledge, contending that one cannot explicitly know a fact unless one is aware of it. Finally, the local-reasoning approach tries to capture the intuition of a mind as a society of agents, each with its own (possibly inconsistent) beliefs.

One issue that we did not explore in this chapter is that of hybrid approaches, which combine features from several of the approaches discussed. We also did not address the interaction between knowledge and time. Combining several approaches and adding time to the models can greatly increase the complexity of the situations that can be captured. To see how the extra expressive power gained by adding time can be used, consider how people deal with inconsistencies. It has frequently been observed that people do not like inconsistencies. Yet occasionally they become aware that their beliefs are inconsistent. When this happens, people tend to modify their beliefs in order to make them consistent again. In a system with awareness, local reasoning, and time, this can be captured with the following axiom:

$$\big(X_i\varphi \wedge X_i\neg\varphi \wedge A_i(X_i\varphi \wedge X_i\neg\varphi)\big) \Rightarrow \bigcirc\big(\neg(X_i\varphi \wedge X_i\neg\varphi)\big).$$

This axiom says that if agent i has an inconsistent belief of which he is aware, then at the next step he will modify his belief so that it is no longer inconsistent. See Exercise 9.56 for a further discussion of adding time.

Ultimately, the choice of the approach used depends on the application. Certainly one criterion for an adequate approach is that it be expressive. As is shown in Exercises 9.45 and 9.48, there is a sense in which the syntactic approach, the impossible-worlds approach (in its full generality), and the awareness approach are all equally expressive, and more expressive than the other approaches we have considered. Nevertheless, while whatever approach we use must be expressive enough to describe the relevant details of the application being modeled, the "most expressive" approach is not always the one that does so in the most useful or most natural way. We expect that many applications can be usefully represented using the techniques we have presented, but this is an empirical question that deserves further study.

Exercises

9.1 Show that if $\mathcal{M}$ is any subclass of $\mathcal{M}_n$, then we have full logical omniscience with respect to $\mathcal{M}$.

9.2 This exercise considers some relations among various cases of omniscience.

(a) Assume that whenever φ logically implies ψ, then the formula $\varphi \Rightarrow \psi$ is valid. Show that closure under material implication and knowledge of valid formulas implies closure under logical implication.

(b) Assume that $\varphi \wedge \psi$ logically implies both φ and ψ, and that φ logically implies ψ iff φ is logically equivalent to $\varphi \wedge \psi$. Show that closure under logical implication is equivalent to the combination of closure under logical equivalence and the opposite direction of closure under conjunction (if agent i knows $\varphi \wedge \psi$, then agent i knows both φ and ψ).

9.3 Discuss various conditions that can be imposed on standard syntactic assignments in order to make the positive and negative introspection axioms valid in syntactic structures that satisfy these conditions.

9.4 Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ be a Kripke structure. Let M' be the MS structure $(S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, where $\mathcal{C}_i(s)$ is the set of all supersets of $\mathcal{K}_i(s)$. Show that $(M, s) \models \varphi$ iff $(M', s) \models \varphi$, for each formula φ.

9.5 Show that the only form of logical omniscience that holds in MS structures is closure under logical equivalence.

** **9.6** Prove Theorem 9.2.2. (Hint: use the maximal consistent set construction as in Chapter 3.)

** **9.7** Fill in the details of the proof of Theorem 9.2.3.

* **9.8** Consider the following possible axioms:

E1. $\neg K_i(\mathit{false})$

E2. $K_i(\mathit{true})$

E3. $K_i(\varphi \wedge \psi) \Rightarrow K_i\varphi$

E4. $K_i\varphi \wedge K_i\psi \Rightarrow K_i(\varphi \wedge \psi)$

E5. $K_i\varphi \Rightarrow K_iK_i\varphi$

E6. $\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$

E7. $K_i\varphi \Rightarrow \varphi$

Now consider the following conditions on $\mathcal{C}_i$:

C1. $\emptyset \notin \mathcal{C}_i(s)$

C2. $S \in \mathcal{C}_i(s)$

C3. If $T \in \mathcal{C}_i(s)$ and $T \subseteq U$, then $U \in \mathcal{C}_i(s)$

C4. If $T \in \mathcal{C}_i(s)$ and $U \in \mathcal{C}_i(s)$, then $T \cap U \in \mathcal{C}_i(s)$

C5. If $T \in \mathcal{C}_i(s)$ then $\{t \mid T \in \mathcal{C}_i(t)\} \in \mathcal{C}_i(s)$

C6. If $T \notin \mathcal{C}_i(s)$ then $\{t \mid T \notin \mathcal{C}_i(t)\} \in \mathcal{C}_i(s)$

C7. If $T \in \mathcal{C}_i(s)$, then $s \in T$

Define an *MS frame* to be a tuple $(S, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ where S is a set (whose members are called states), and $\mathcal{C}_i(s)$ is a set of subsets of S. We say that the MS structure $(S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ is *based on* the MS frame $(S, \mathcal{C}_1, \ldots, \mathcal{C}_n)$. Note that the conditions C1–C7 are really conditions on frames.

Prove that for $1 \leq k \leq 7$, an MS frame N satisfies Ck if and only if every MS structure based on N satisfies Ek at every state. (Hint: the "if" direction may require the use of two primitive propositions.)

9.9 In this exercise, we consider an alternate approach to the nonstandard approach based on a nonstandard notion of truth. In this approach, there is no * function. The structure, however, contains *nonstandard truth assignments* (which we describe below) rather than usual truth assignments, and a pair of possibility relations, $\mathcal{K}_i^+$ and $\mathcal{K}_i^-$, for each agent i. Intuitively, $\mathcal{K}_i^+$ is used to evaluate the truth of formulas of the form $K_i\varphi$ and $\mathcal{K}_i^-$ is used to evaluate the truth of formulas of the form $\neg K_i\varphi$.

As before, a *literal* is a primitive proposition p or its negation $\neg p$. Define a *nonstandard truth assignment* to be a function that assigns to each literal a truth value. Thus, although an ordinary truth assignment assigns a truth value to each primitive proposition p, a nonstandard truth assignment assigns a truth value to both p and $\neg p$. Under a given nonstandard truth assignment, it is possible that both p and $\neg p$ are assigned the value **true**, or that both are assigned **false**, or that one is assigned **true** and the other **false**. Intuitively, this allows the truth value of p and of its negation to be independent.

An *alternate nonstandard structure* M is a tuple

$$(S, \pi, \mathcal{K}_1^+, \ldots, \mathcal{K}_n^+, \mathcal{K}_1^-, \ldots, \mathcal{K}_n^-),$$

where S is a set of states, $\pi(s)$ is a nonstandard truth assignment for each state $s \in S$, and each $\mathcal{K}_i^+$ and $\mathcal{K}_i^-$ is a binary relation on S. Rather than define $\neg\varphi$ to be true iff φ is not true, as we do with Kripke structures, we define separately what it means for φ to be true and what it means for $\neg\varphi$ to be true, for each type of formula φ (that is, for primitive propositions, and formulas of the form $\varphi_1 \wedge \varphi_2$, $\neg\varphi$, and $K_i\varphi$). This way we can make the truth of φ and $\neg\varphi$ independent. The definition is as follows:

$(M, s) \models p$ (for a primitive proposition p) iff $\pi(s)(p) =$ **true**

$(M, s) \models \varphi_1 \wedge \varphi_2$ iff $(M, s) \models \varphi_1$ and $(M, s) \models \varphi_2$

$(M, s) \models K_i\varphi$ iff $(M, t) \models \varphi$ for all t such that $(s, t) \in \mathcal{K}_i^+$

$(M, s) \models \neg p$ (for a primitive proposition p) iff $\pi(s)(\neg p) =$ **true**

$(M, s) \models \neg(\varphi_1 \wedge \varphi_2)$ iff $(M, s) \models \neg\varphi_1$ or $(M, s) \models \neg\varphi_2$

$(M, s) \models \neg\neg\varphi$ iff $(M, s) \models \varphi$

$(M, s) \models \neg K_i\varphi$ iff $(M, t) \not\models \varphi$ for some t such that $(s, t) \in \mathcal{K}_i^-$.

Show that nonstandard semantics and alternate nonstandard semantics are equivalent:

(a) Show that for each nonstandard structure M and state s of M, there is an alternate nonstandard structure M' and state s' of M', such that for each formula φ of $\mathcal{L}_n$, we have $(M, s) \models \varphi$ iff $(M', s') \models \varphi$.

(b) Show that for each alternate nonstandard structure M and state s of M, there is a nonstandard structure M' and state s' of M', such that for each formula φ of $\mathcal{L}_n$, we have $(M, s) \models \varphi$ iff $(M', s') \models \varphi$.

9.10 In this exercise, we consider yet another alternate nonstandard semantics. We again use the alternate nonstandard structures of Exercise 9.9, but instead of explicitly defining the negation case separately (as we did in Exercise 9.9), we now have two "support relations" $\models_T$ and $\models_F$. Intuitively, $(M, s) \models_T \varphi$ (where T stands for "true") means that the truth of φ is supported at (M, s), while $(M, s) \models_F \varphi$ (where F stands for "false") means that the truth of $\neg\varphi$ is supported at (M, s). We say that $(M, s) \models \varphi$ if $(M, s) \models_T \varphi$.

$(M, s) \models_T p$ (for a primitive proposition p) iff $\pi(s)(p) = \mathbf{true}$

$(M, s) \models_F p$ (for a primitive proposition p) iff $\pi(s)(\neg p) = \mathbf{true}$

$(M, s) \models_T \neg\varphi$ iff $(M, s) \models_F \varphi$

$(M, s) \models_F \neg\varphi$ iff $(M, s) \models_T \varphi$

$(M, s) \models_T \varphi_1 \wedge \varphi_2$ iff $(M, s) \models_T \varphi_1$ and $(M, s) \models_T \varphi_2$

$(M, s) \models_F \varphi_1 \wedge \varphi_2$ iff $(M, s) \models_F \varphi_1$ or $(M, s) \models_F \varphi_2$

$(M, s) \models_T K_i\varphi$ iff $(M, t) \models_T \varphi$ for all t such that $(s, t) \in \mathcal{K}_i^+$

$(M, s) \models_F K_i\varphi$ iff $(M, t) \not\models_T \varphi$ for some t such that $(s, t) \in \mathcal{K}_i^-$

Show that for all formulas φ, we have $(M, s) \models_T \varphi$ iff $(M, s) \models \varphi$ according to the alternative semantics of Exercise 9.9, and $(M, s) \models_F \varphi$ iff $(M, s) \models \neg\varphi$ according to the alternative semantics.

9.11 Define a *nonstandard frame* F to be a tuple $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$ where $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ is a Kripke frame and where * is a unary function from S to itself such that $s^{**} = s$ for each $s \in S$. Thus, a nonstandard frame is a nonstandard structure without the π. We say that the nonstandard structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$ is *based on* the nonstandard frame $(S, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$. Prove that a state s in a frame F is standard iff the formula $p \wedge \neg p$ is false at s in all nonstandard structures based on F.

9.12 Prove Proposition 9.3.1.

9.13 Prove the induction claim in Proposition 9.3.2.

9.14 Demonstrate the nonstandard behavior of $\Rightarrow$ by showing that

(a) the following are equivalent for nonstandard structures:

(i) $(M, s) \models \varphi_1 \Rightarrow \varphi_2$

(ii) If $(M, s^*) \models \varphi_1$, then $(M, s) \models \varphi_2$.

Note that if $s = s^*$, then part (ii) gives the usual semantics for implication.

(b) $\{p, p \Rightarrow q\}$ does not logically imply q in nonstandard structures.

9.15 Let $(S, \pi, \mathcal{K}_1^+, \ldots, \mathcal{K}_n^+, \mathcal{K}_1^-, \ldots, \mathcal{K}_n^-)$ be an alternate nonstandard structure as in Exercise 9.9. Show that positive introspection ($K_i\varphi \Rightarrow K_iK_i\varphi$) holds for this structure if $\mathcal{K}_i^+$ is transitive. Show that negative introspection ($\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$) holds if $(s, t) \in \mathcal{K}_i^+$ and $(s, u) \in \mathcal{K}_i^-$ imply that $(t, u) \in \mathcal{K}_i^-$.

9.16 Construct a nonstandard structure M and state s of M where $(M, s) \models K_i p$ and $(M, s) \models K_i(p \Rightarrow q)$, but $(M, s) \not\models K_i q$.

9.17 Let φ_1 and φ_2 be standard formulas. Show that if $\varphi_1 \hookrightarrow \varphi_2$ is valid with respect to nonstandard Kripke structures, then $\varphi_1 \Rightarrow \varphi_2$ is valid with respect to standard Kripke structures. Show that the converse implication does not hold.

9.18 Prove Proposition 9.3.3.

** **9.19** Prove Theorem 9.3.4. (Hint: use the maximal consistent set construction as in Chapter 3. Show that a set V of $\mathrm{K}_n^{\hookrightarrow}$ formulas is a maximal consistent set iff for each formula φ of $\mathcal{L}_n^{\hookrightarrow}$, either φ or $(\varphi \hookrightarrow \mathit{false})$ is in V. If V is a set of $\mathrm{K}_n^{\hookrightarrow}$ formulas, define $V^* = \{\varphi \in \mathcal{L}_n^{\hookrightarrow} \mid \neg\varphi \notin V\}$. Show that V^* is a maximal $\mathrm{K}_n^{\hookrightarrow}$ consistent set, and that $V^{**} = V$. In constructing the canonical model, if s_V is the set corresponding to the maximal $\mathrm{K}_n^{\hookrightarrow}$ consistent set V, then define $(s_V)^* = s_{V^*}$.)

9.20 In this exercise, we consider the relationship between the two forms of negation definable in nonstandard structures.

(a) Show that the rule "from $\neg\varphi$ infer $\varphi \hookrightarrow \mathit{false}$" is a sound inference rule with respect to $\mathcal{NM}_n$.

(b) Show that the rule "from $\varphi \hookrightarrow \mathit{false}$ infer $\neg\varphi$" is a sound inference rule with respect to $\mathcal{NM}_n$.

(c) Show, however, that neither $\neg\varphi \hookrightarrow (\varphi \hookrightarrow \mathit{false})$ nor $(\varphi \hookrightarrow \mathit{false}) \hookrightarrow \neg\varphi$ are sound axioms with respect to $\mathcal{NM}_n$.

9.21 Show that in the nonstandard worlds approach, each of the types of logical omniscience mentioned in the introduction to this chapter hold, when we replace $\Rightarrow$ by $\hookrightarrow$.

9.22 In this exercise, we focus on the axiom $\neg K_i\varphi \hookrightarrow K_i\neg K_i\varphi$.

(a) Show that the axiom $\neg K_i\varphi \hookrightarrow K_i\neg K_i\varphi$ may fail in nonstandard structures, even if we restrict to structures where the $\mathcal{K}_i$'s are equivalence relations.

(b) Show that $\neg K_i\varphi \hookrightarrow K_i\neg K_i\varphi$ is valid in nonstandard structures where the $\mathcal{K}_i$'s are equivalence relations and where $(s^*, t^*) \in \mathcal{K}_i$ whenever $(s, t) \in \mathcal{K}_i$.

(c) Show that $\neg K_i\varphi \hookrightarrow K_i\neg K_i\varphi$ is valid in a nonstandard frame $F = (S, \mathcal{K}_1, \ldots, \mathcal{K}_n, *)$ (i.e., valid in every nonstandard structure based on F, according to the definition given in Exercise 9.11) iff for all $s, t, u \in S$, we have that $(s^*, t^*) \in \mathcal{K}_i$ and $(s, u) \in \mathcal{K}_i$ together imply that $(u^*, t^*) \in \mathcal{K}_i$.

9.23 In this exercise, we consider the problem of expressing standard negation in nonstandard structures.

(a) Prove that in $\mathcal{L}_n$ we cannot say that a formula φ is false. That is, there is no formula ψ such that for all nonstandard structures M and states s we have that $(M, s) \models \psi$ iff $(M, s) \not\models \varphi$. (Hint: use the second part of Theorem 9.3.2.)

(b) Recall that in the nonstandard semantics, we redefined *true* to be an abbreviation for some fixed nonstandard tautology such as $p \hookrightarrow p$; we still abbreviate $\neg\mathit{true}$ by *false*. Let φ be a formula of $\mathcal{L}_n^{\hookrightarrow}$. Prove that for all nonstandard structures M and states s we have that $(M, s) \models \varphi \hookrightarrow \mathit{false}$ iff $(M, s) \not\models \varphi$.

* **9.24** Prove that the validity problem for $\mathcal{L}_n^{\hookrightarrow}$-formulas with respect to $\mathcal{NM}_n$ is *PSPACE*-hard in general and co-*NP*-hard for propositional $\mathcal{L}_n^{\hookrightarrow}$-formulas. (Hint: show that standard validity can be reduced to nonstandard validity in the following manner. If φ is a standard formula, then let φ^{ns} be the nonstandard formula obtained by recursively replacing in φ each subformula of the form $\neg\varphi$ by $\varphi \hookrightarrow \mathit{false}$ and each occurrence of $\Rightarrow$ by $\hookrightarrow$. Show that φ is valid with respect to standard structures iff φ^{ns} is valid with respect to nonstandard structures.)

9.25 Let φ_1 and φ_2 be $\mathcal{L}_n^{\hookrightarrow}$-formulas. Show that φ_1 logically implies φ_2 with respect to $\mathcal{NM}_n$ iff $K_i\varphi_1$ logically implies $K_i\varphi_2$ with respect to $\mathcal{NM}_n$. (Hint: one direction is easy. For the other direction, assume that φ_1 does not logically imply φ_2 with respect to $\mathcal{NM}_n$. Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, ^*)$ be a nonstandard structure and u a state of M such that $(M, u) \models \varphi_1$ and $(M, u) \not\models \varphi_2$. Define a new nonstandard structure $M' = (S', \pi', \mathcal{K}_1', \ldots, \mathcal{K}_n', ^\dagger)$ with one additional state $t \notin S$ by taking (a) $S' = S \cup \{t\}$, (b) $\pi'(s) = \pi(s)$ for $s \in S$ and $\pi'(t)$ is arbitrary, (c) $\mathcal{K}_j' = \mathcal{K}_j$ for $j \neq i$, and $\mathcal{K}_i' = \mathcal{K}_i \cup \{(t, u)\}$, and (d) $s^\dagger = s^*$ for $s \in S$, and $t^\dagger = t$. Show that since $(M, u) \models \varphi_1$ and $(M, u) \not\models \varphi_2$, we also have $(M', u) \models \varphi_1$ and $(M', u) \not\models \varphi_2$. But then $(M', t) \models K_i\varphi_1$ and $(M', t) \not\models K_i\varphi_2$, and hence $K_i\varphi_1$ does not logically imply $K_i\varphi_2$ with respect to $\mathcal{NM}_n$.)

9.26 Show that the problem of determining whether κ logically implies φ in nonstandard propositional logic is co-*NP*-hard. (Hint: use Exercise 9.24 and Proposition 9.3.3).

9.27 Show that even in our nonstandard propositional semantics, every standard propositional formula is equivalent to a formula in CNF. (Hint: make use of Proposition 9.3.1, and mimic the usual textbook proof that every propositional formula is equivalent to a formula in CNF.)

9.28 Show that in standard propositional logic, the problem of deciding whether a CNF formula κ logically implies a clause φ is no easier than the general problem of logical implication in propositional logic, that is, co-*NP*-complete. (Hint: let p be a primitive proposition that does not appear in the propositional formula φ. Show that $\varphi \Rightarrow p$ is valid with respect to standard propositional logic iff φ is unsatisfiable in standard propositional logic.)

9.29 Prove the "if" direction in the proof of Theorem 9.3.6.

9.30 Prove Proposition 9.3.7. Show that the converse claim fails.

* **9.31** Show how the various forms of logical omniscience can be captured by imposing appropriate conditions on the syntactic assignment in impossible-worlds structures.

9.32 Show that there are formulas φ_1 and φ_2 in $\mathcal{L}_n$ such that φ_1 standard-state logically implies φ_2, but $K_i\varphi_1$ does not standard-state logically imply $K_i\varphi_2$.

9.33 Let φ_1 and φ_2 be $\mathcal{L}_n^{\hookrightarrow}$-formulas. Show that $K_i\varphi_1$ standard-state logically implies $K_i\varphi_2$ iff $K_i\varphi_1$ logically implies $K_i\varphi_2$ with respect to $\mathcal{NM}_n$. (Hint: the proof is very similar to that of Exercise 9.25.)

9.34 Let φ be a standard propositional formula. Show that if $complete(\varphi)$ is true at a state s of a nonstandard structure, then s is complete with respect to φ.

9.35 Let φ be a standard propositional formula. Show that φ is a tautology of standard propositional logic if and only if $complete(\varphi)$ logically implies φ with respect to $\mathcal{NM}_n$. (Hint: one direction is easy. For the other direction: assume that φ is a tautology of standard propositional logic, and $(M, s) \models complete(\varphi)$. Let Ψ be the set of primitive propositions that appear in φ. Thus, $(M, s) \models p \vee \neg p$ for each $p \in \Psi$. Hence, either $(M, s) \models p$ or $(M, s) \models \neg p$, for each $p \in \Psi$. Define the truth assignment v by letting $v(p) =$ **true** if $(M, s) \models p$, and $v(p) =$ **false** otherwise. By an induction on the structure of formulas, show that for each propositional formula ψ all of whose primitive propositions are in Ψ, (a) if ψ is true under v, then $(M, s) \models \psi$, and (b) if ψ is false under v, then $(M, s) \models \neg\psi$. Since φ is a tautology of standard propositional logic, φ is true under v. It follows that $(M, s) \models \varphi$.)

* **9.36** Let φ be a formula in $\mathcal{L}_n$ of depth d (see Exercise 3.23). Let τ be the formula

$$complete(\varphi) \wedge E(complete(\varphi)) \wedge \ldots \wedge E^d(complete(\varphi)).$$

Show that φ is a valid formula of K_n iff τ logically implies φ with respect to $\mathcal{NM}_n$. (Hint: the proof is similar to the proof of Exercise 9.35. Let φ be a valid formula of K_n. Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, {}^*)$ be a nonstandard structure and s a state of M such that $(M, s) \models \tau$. Define π' so that $\pi'(t)(p) =$ **true** if $(M, t) \models p$, and $\pi'(t)(p) =$ **false** otherwise, for each state t of M and each primitive proposition p. Let $M' = (S, \pi', \mathcal{K}_1, \ldots, \mathcal{K}_n)$. Thus, M' is a standard Kripke structure. Show that if ψ is of depth $d' \leq d$, if every primitive proposition that appears in ψ also appears in φ, and if t is of distance at most $d - d'$ from s, then $(M, t) \models \psi$ iff $(M', t) \models \psi$. In particular, since $(M', s) \models \varphi$, we also have $(M, s) \models \varphi$.)

9.37 Show that the following formulas are satisfiable at a standard state of a nonstandard structure:

(a) $K_i p \wedge K_i(p \Rightarrow q) \wedge \neg K_i q$,

(b) $K_i p \wedge \neg K_i\big(p \wedge (q \vee \neg q)\big)$.

Show that the following formulas are standard-state valid:

(c) $\varphi \Leftrightarrow \big(\varphi \wedge (\psi \vee \neg\psi)\big)$,

(d) $\big(K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)\big) \Rightarrow K_i\big(\psi \vee (\varphi \wedge \neg\varphi)\big)$.

9.38 Let p be a primitive proposition. Show that there is no formula φ in $\mathcal{L}_n$ such that if φ holds in a state s of a nonstandard structure M, then at most one of p or $\neg p$ is true in s. (Hint: use Theorem 9.3.2.)

9.39 Show that without the restriction that if $\varphi \wedge \psi \in \mathcal{A}_i(s)$, then both $\varphi, \psi \in \mathcal{A}_i(s)$, the formula $X_i(p \wedge \neg X_i p)$ ("agent i explicitly knows both that p is true and that he doesn't explicitly know it") is satisfiable in an awareness structure, even if $\mathcal{K}_i$ is an equivalence relation. As shown in Exercise 3.10, this is not true if we replace the X_i by K_i.

* **9.40** In this exercise, we examine some properties of the logic of awareness.

(a) Show that a sound and complete axiomatization of the logic of awareness is given by $\mathrm{K}_n + \{X_i\varphi \Leftrightarrow (A_i\varphi \wedge K_i\varphi)\}$.

(b) Show that a sound and complete axiomatization of the logic of awareness when we restrict attention to awareness structures where the $\mathcal{K}_i$'s are equivalence relations is given by $\mathrm{S5}_n + \{X_i\varphi \Leftrightarrow (A_i\varphi \wedge K_i\varphi)\}$.

9.41 Show that the following are valid with respect to awareness structures:

(a) $\big(X_i\varphi \wedge X_i(\varphi \Rightarrow \psi) \wedge A_i\psi\big) \Rightarrow X_i\psi$,

(b) From φ infer $A_i\varphi \Rightarrow X_i\varphi$.

9.42 This exercise deals with awareness structures where the $\mathcal{K}_i$'s are equivalence relations.

(a) Show that there is an awareness structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$ where each $\mathcal{K}_i$ is an equivalence relation, but the introspection axioms $X_i\varphi \Rightarrow X_iX_i\varphi$ and $\neg X_i\varphi \Rightarrow X_i\neg X_i\varphi$ fail.

(b) Show that there is an awareness structure $(S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$ where each $\mathcal{K}_i$ is an equivalence relation, but the following modified introspection axioms fail:

(i) $(X_i\varphi \wedge A_i X_i\varphi) \Rightarrow X_i X_i\varphi$.

(ii) $\big(\neg X_i\varphi \wedge A_i(\neg X_i\varphi)\big) \Rightarrow X_i \neg X_i\varphi$.

(c) Show that if $\mathcal{K}_i$ is an equivalence relation, and if $(s, t) \in \mathcal{K}_i$ implies $\mathcal{A}_i(s) = \mathcal{A}_i(t)$, then the modified introspection axioms in part (b) are valid with respect to awareness structures.

(d) Modify the axioms in (b) so that they stay valid even if we no longer assume that $(s, t) \in \mathcal{K}_i$ implies $\mathcal{A}_i(s) = \mathcal{A}_i(t)$.

9.43 Let $\mathcal{A}_i(s)$ consist precisely of the formulas where the only primitive propositions that appear are those in some fixed subset of the primitive propositions. Show that a sound and complete axiomatization of the logic of awareness is then given by the axiomatization $\mathrm{K}_n + \{X_i\varphi \Leftrightarrow (A_i\varphi \wedge K_i\varphi)\}$ of Exercise 9.40(a), along with the following axioms:

$$\begin{aligned} A_i(\neg\varphi) &\Leftrightarrow A_i\varphi \\ A_i(\varphi \wedge \psi) &\Leftrightarrow (A_i\varphi \wedge A_i\psi) \\ A_i(X_j\varphi) &\Leftrightarrow A_i\varphi \\ A_i(K_j\varphi) &\Leftrightarrow A_i\varphi \\ A_i(A_j\varphi) &\Leftrightarrow A_i\varphi. \end{aligned}$$

Show a similar result for the case of Exercise 9.40(b).

9.44 Show that if awareness is closed under subformulas, then

$$\big(X_i\varphi \wedge X_i(\varphi \Rightarrow \psi)\big) \Rightarrow X_i\psi$$

is valid with respect to awareness structures.

9.45 This exercise relates impossible-worlds structures, awareness structures, and syntactic structures.

(a) Let $M = (S, W, \sigma, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ be an impossible-worlds structure. Construct an awareness structure $M' = (W, \pi, \mathcal{K}'_1, \ldots, \mathcal{K}'_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$ such that for any $\mathcal{L}_n$-formula φ and state $s \in W$ we have that $(M, s) \models \varphi$ iff $(M', s) \models \varphi'$, where φ' is obtained from φ by replacing each K_i by X_i.

(b) Let $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n)$ be an awareness structure. Construct a syntactic structure $M' = (S, \sigma)$ such that for any $\mathcal{L}_n$-formula φ and state $s \in S$ we have that $(M, s) \models \varphi'$ iff $(M', s) \models \varphi$, where φ' is obtained from φ by replacing each K_i by X_i.

(c) Let $M = (S, \sigma)$ be a syntactic structure. Construct an impossible-worlds structure $M' = (S', S, \sigma', \mathcal{K}_1, \ldots, \mathcal{K}_n)$ such that for any $\mathcal{L}_n$-formula φ and state $s \in S$ we have that $(M, s) \models \varphi$ iff $(M', s) \models \varphi$.

This exercise shows that, in a certain sense, impossible-worlds structures, awareness structures, and syntactic structures are equi-expressive. Any situation that can be described in one framework can also be described in the other two.

9.46 In the local-reasoning approach, let $D_i\varphi$ mean that agent i would know φ as a result of pooling together the information in his various frames of mind.

(a) Give a formal definition of the semantics of $D_i\varphi$.

(b) Show that $D_i\varphi$ holds if φ is true in every world in all of agent i's frames of mind.

(c) Show that if an agent holds inconsistent knowledge in different frames of mind, then $D_i(\mathit{false})$ holds.

9.47 Let $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a local-reasoning structure. Show that if $\mathcal{C}_i(s)$ is just a singleton set, say $\mathcal{C}_i(s) = \{T_i^s\}$, for each state s and agent i, then this structure is equivalent to a Kripke structure $M' = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$, where $(s, t) \in \mathcal{K}_i$ exactly if $t \in T_i^s$. That is, show that for every formula φ and a state s we have that $(M, s) \models \varphi$ if and only if $(M', s) \models \varphi$.

9.48 In this exercise, we show that local-reasoning structures are less expressive than MS structures, in that any situation that can be described by a local-reasoning structure can also be described by an MS structure, while the converse does not hold. We also show that MS structures are less expressive than syntactic structures. Since, by Exercise 9.45, syntactic structures, awareness structures, and impossible-worlds structures are all equally expressive, we have a complete characterization of the expressive power of these frameworks.

(a) Let $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a local-reasoning structure, and let $\mathcal{C}_i'$ be the set of all supersets of members of $\mathcal{C}_i$. Let M' be the MS structure $(S, \pi, \mathcal{C}_1', \ldots, \mathcal{C}_n')$. Show that $(M, s) \models \varphi$ iff $(M', s) \models \varphi$, for each formula φ.

(b) Show that there are situations describable by MS structures that are not describable by local-reasoning structures in the sense of part (a). (Hint: recall that $K_i(true)$ is valid in local-reasoning structures but not in MS structures.)

(c) Given an MS structure $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, construct a syntactic structure $M' = (S, \sigma)$ such that $(M, s) \models \varphi$ iff $(M', s) \models \varphi$, for each formula φ.

(d) Show that there are situations describable by syntactic structures that are not describable by MS structures in this sense. (Hint: $K_i p \equiv K_i(p \vee p)$ is valid with respect to MS structures, but not with respect to syntactic structures.)

9.49 Show that we can guarantee closure under material implication in a local-reasoning structure by requiring that there always be a frame of mind where an agent puts together information that he knows in other frames.

9.50 Show that in the local-reasoning approach we have knowledge of valid formulas and closure under logical implication.

** **9.51** Prove Theorem 9.6.1.

** **9.52** Prove Theorem 9.6.2. (Hint: the lower bound is immediate by considering propositional formulas. For the upper bound prove that if a formula is satisfiable, then it has a "small" model. Use techniques analogous to those used in Section 3.6.)

9.53 Let $M = (S, \pi, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a local-reasoning structure and let φ be an arbitrary formula of $\mathcal{L}_n$.

(a) Assume that s is a member of T whenever $T \in \mathcal{C}_i(s)$. Show that $K_i\varphi \Rightarrow \varphi$ is valid in M.

(b) Assume that whenever $T \in \mathcal{C}_i(s)$ and $s' \in T$, then $T \in \mathcal{C}_i(s')$. Show that $K_i\varphi \Rightarrow K_i K_i\varphi$ is valid in M.

(c) Assume that for some $T \in \mathcal{C}_i(s)$ and all $t \in T$, we have $\mathcal{C}_i(t) \subseteq \mathcal{C}_i(s)$. Show that $\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$ is valid in M.

* **9.54** The conditions in clauses (b) and (c) of Exercise 9.53 are sufficient but not necessary. By considering local-reasoning structures as MS structures (Exercise 9.48), find closure conditions on the $\mathcal{C}_i$'s that precisely capture, in the sense of Exercise 9.8, positive and negative introspection.

9.55 Show that the following formulas are valid with respect to local-reasoning structures with narrow-minded agents:

(a) $K_i(\neg(K_i\varphi \wedge K_i\neg\varphi))$,

(b) $K_i\big((K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi\big)$.

9.56 We can add awareness to our model of knowledge in multi-agent systems by having awareness of agent i be a function $\mathcal{A}_i(r, m)$ of the point (r, m).

(a) Describe conditions on $\mathcal{A}_i$ that express the assumptions that agent i's awareness increases over time and that agent i eventually becomes aware of all formulas.

(b) Consider a system with perfect recall, with agents whose awareness increases over time and who eventually become aware of every formula. Such a system satisfies a notion of *eventual* logical omniscience. Describe an axiom and an inference rule that express this form of logical omniscience.

Notes

The problem of logical omniscience was first observed (and named) by Hintikka [1962]. The syntactic approach to modeling explicit knowledge was studied by Eberle [1974] and Moore and Hendrix [1979]. Its formalization within the possible-worlds approach is due to Halpern and Moses [1990]. Related ideas were studied by Georgeff [1985]. Konolige [1986] considers the case where each agent has a base set of formulas he knows, and uses a sound but possibly incomplete set of inference rules to derive his other knowledge. Montague-Scott structures were introduced by Montague [1968, 1970] and Scott [1970]. Theorem 9.2.2 was proven in the single-agent case by Segerberg [1971]. (Segerberg called the logic satisfying the axioms described in Theorem 9.2.2 *classical logic*, and denoted it E.) Theorem 9.2.3 is due to Vardi [1989]. Exercise 9.8 is due to Chellas [1980] and Vardi [1989].

The nonstandard logic, as well as the approach and results in Section 9.3, where nonstandard worlds are treated the same as standard worlds, were introduced by Fagin, Halpern, and Vardi [1990]; they also introduced the notion of strong implication. Numerous other nonstandard logics are known in the literature; perhaps the two best known are *intuitionistic logic* [Heyting 1956] and (various notions of) *relevance logic* [Anderson and Belnap 1975]. Strong implication is closely related to various notions of "relevant implications" in relevance logic. First-order logics based on a nonstandard propositional semantics closely related that given here are described by

Kifer and Lozinski [1992] and Patel-Schneider [1985]. Our strong implication is essentially the ontological implication of Kifer and Lozinski. The idea of using * to define negation is due to Routley and Routley [1972] (see also [Białynicki-Birula and Rasiowa 1957]), as is a propositional version of Theorem 9.3.2. The axiomatization $K_n^{\hookrightarrow}$ was studied by Fagin, Halpern, and Vardi [1990]. They observe that Theorem 9.3.4 would fail if $K_n^{\hookrightarrow}$ included only the tautologies and nonstandard modus ponens of nonstandard propositional logic instead of including NPR. Fagin, Halpern, and Vardi [1992b] also study the inference rules of nonstandard propositional logic.

The nonstandard-worlds approach of Exercise 9.10 is due to Levesque [1984b] and Lakemeyer [1987]. Levesque considers the case where there is no nesting of K_i's, and makes use of standard-state validity. He considers *situations* (as Barwise and Perry [1983] do), where φ can be true, false, neither, or both. Lakemeyer extends Levesque's approach to allow nesting of K_i's, by using what we call $\mathcal{K}_i^+$ and $\mathcal{K}_i^-$, as in Exercise 9.9. Levesque shows that if κ and φ are propositional formulas in CNF, then $K_i\kappa \Rightarrow K_i\varphi$ is standard-state valid iff every clause of φ includes a clause of κ. He thereby obtains a polynomial-time procedure for deciding whether knowledge of a CNF formula implies knowledge of another, as in Theorem 9.3.6. Both Levesque and Lakemeyer allow only one agent, but the extension to multiple agents is straightforward. Exercise 9.15 is due to Lakemeyer. The use of nonstandard truth assignments, as in Exercise 9.9, is due to Levesque [1988]; it is derived from a four-valued semantics developed by Dunn in the late 1960's and published in 1976 [Dunn 1976], and discussed further by Belnap [1977] and Dunn [1986]. Dunn [1986] also discusses the equivalence between the four-valued semantics and the Routleys' semantics using the * operator. Patel-Schneider [1989] gives another application of this semantics. The application of the nonstandard-logic approach to query evaluation in knowledge bases was suggested by Levesque [1981].

The idea of using nonstandard worlds, where not all valid formulas need be true, or where inconsistent formulas may be true, in order to deal with logical omniscience, is due to Cresswell [1970, 1972, 1973], who called them *non-classical worlds*. (Kripke's notion of *non-normal worlds* [1965] can be viewed as a precursor.) Such worlds were called *impossible* by Hintikka [1975] and Rantala [1982], and *nonstandard* by Rescher and Brandom [1979]. Wansing [1990] showed how different properties of knowledge can be captured by imposing certain conditions on the impossible worlds (Exercise 9.31). He also showed that this approach is at least as powerful as the awareness approach, thereby providing the inspiration for Exercise 9.45. Lipman [1992a, 1992b, 1994] has recently used the impossible-worlds approach as a tool for dealing with lack of logical omniscience in the context of game theory.

The awareness approach (Section 9.5) and the local-reasoning approach (Section 9.6) were introduced by Fagin and Halpern [1988a]. All of the exercises about these approaches, Theorem 9.6.1, and a version of Theorem 9.4.2 also appear in their paper. The notion of awareness was further studied and developed by Huang and Kwast [1991] and Thijsse [1992, 1993]. Further discussion regarding the issues raised in Exercises 9.45, 9.48, and 9.54 can also be found in Thijsse's paper and thesis. The term *explicit knowledge* in this context is due to Levesque [1984b]. Logics similar to the logic of local reasoning are given by Levesque [1985], Stalnaker [1985], and Zadrozny [1985].

Chapter 10

Knowledge and Computation

Is knowledge knowable? If not, how do we know this?

Woody Allen

10.1 Knowledge and Action Revisited

As we discussed in Chapter 4, the interpretation of knowledge in the possible-world approach is an *external* one, ascribed by, say, the system designer to the agents. Agents are not assumed to compute their knowledge in any way, nor is it assumed that they can necessarily answer questions based on their knowledge. In Chapter 9 we referred to such knowledge as *implicit*, since, while it could be inferred from the information available to the agents, it was not necessarily *explicitly* available to them. Despite the fact that no notion of computation is involved, we gave many examples to show that this notion of implicit knowledge is useful in a number of applications, such as the analysis of distributed systems.

We are often, however, interested in applications in which agents need to *act* on their knowledge. In such applications, implicit knowledge may be insufficient; in many cases an agent can base his actions only on what he *explicitly* knows. (Note that the term "explicit" here is used in the same spirit but not necessarily the same technical sense as in Chapter 9.) In these situations, an agent that acts on his knowledge must be able to compute this knowledge; we need to take into account the algorithms available to the agent, as well as the "effort" required to compute knowledge. Computing knowledge demands that the agents have access to appropriate algorithms and to the computational resources required by these algorithms. Indeed, the fact that

economic agents need to compute their knowledge is well-recognized nowadays on Wall Street, where the trend towards using more sophisticated algorithms and more extensive computational resources in economic decision-making is quite evident.

Knowledge bases provide an example of an application in which agents have to act on their knowledge. In Sections 4.4.1 and 7.3 we provided a model for a knowledge-based system in which the KB is told facts about an external world, and has to answer queries about that world. The focus in Section 7.3 was on the knowledge-based program followed by the Teller. If we want to analyze the query-answering aspect of this application, then we can think of the KB as following a knowledge-based program of the following form:

case of
  if *asked "does* φ *hold?"* $\wedge K_{KB}\varphi$ **do** say "Yes"
  if *asked "does* φ *hold?"* $\wedge K_{KB}\neg\varphi$ **do** say "No"
  if *asked "does* φ *hold?"* $\wedge \neg K_{KB}\varphi \wedge \neg K_{KB}\neg\varphi$
    do say "I don't know"
end case

Clearly, to actually run the preceding program, the knowledge base must be able to determine *explicitly* whether it knows φ. A notion of implicit knowledge would not suffice here.

In Section 7.1, we defined what it means for a standard program to implement a knowledge-based program (in a given context). Recall that the idea is that instead of running a knowledge-based program, an agent runs a standard program that implements the knowledge-based program. In our discussion of implementation, we did not take into account the complexity of explicitly computing the knowledge in knowledge tests. Intuitively, however, implementing a knowledge-based program requires the agent to evaluate the knowledge tests. In the case of the sequence-transmission problem in Section 7.6, it is fairly simple to evaluate the knowledge tests of the knowledge-based program ST in the relevant contexts, which is why we could replace them by the standard tests of the standard program ST′. In contrast, evaluating the knowledge tests in the program SBA for simultaneous Byzantine agreement in Section 7.4 is far from trivial. In fact, as we saw in Theorem 6.6.10, computing whether a given fact is common knowledge may not be possible for a process that can perform only polynomial-time computations (assuming $P \neq NP$). As a consequence, processes that can perform only polynomial-time computations may not be able to perform the computations necessary to run an optimum program for SBA if we allow generalized-omission failures.

The previous discussion demonstrates that our information-based notion of knowledge in multi-agent systems is not sensitive enough to capture the distinction between the program ST for sequence transmission and the program SBA for simultaneous Byzantine agreement, since it does not distinguish between knowledge that is easy to compute and knowledge that is hard to compute. A knowledge-based analysis of such situations requires a finer notion of knowledge, one that is sensitive to the difficulty of computing knowledge. Our goal in this chapter is to provide such a computational notion of knowledge.

The problem of being able to compute knowledge is related to the logical omniscience problem. As we observed in Chapter 9, one reason that we do not expect agents to be logically omniscient is that they lack the computational resources to compute the logical consequences of their knowledge (although there are other reasons as well). Thus, we would expect that any model that captures the difficulty of computing knowledge is one in which the agents are not logically omniscient. Indeed, this is the case for the solution we propose in this section. Not surprisingly, our solution is related to some of the solutions to the logical omniscience problem proposed in Chapter 9.

It is worth noting that even a resource-bounded agent may at times be able to exhibit some of the properties we associate with logical omniscience. Consider an agent Alice who knows that, given her current epistemic state (i.e., the information she has obtained thus far), the world could be in one of three possible states. In the possible-worlds approach, this situation is modeled by a Kripke structure with a state s where Alice considers three states possible. Even if Alice is resource-bounded, she might well be able to figure out whether a formula φ is true in all three states she considers possible. In this case, Alice would be logically omniscient in state s. In particular, she would explicitly know all logical tautologies, since all logical tautologies are true at all three worlds Alice considers possible. But this does not mean that Alice can solve co-*NP*-complete problems! Even though Alice knows all logical tautologies (among other facts she knows given her epistemic state), she does not know *which* of the facts she knows are tautologies, and she probably does not care!

Alice is much more likely to be interested, not in whether the formula φ is valid, but whether she knows φ in state s. This question is closely related to the model-checking problem (cf. Section 3.2), since an agent i knows a formula φ in a state s of a Kripke structure M precisely when $(M, s) \models K_i\varphi$. We saw earlier (Proposition 3.2.1) that the model-checking problem has an efficient algorithmic solution: it can be solved in time linear in the product of the size of M and the length of φ. In general, if Alice has at her disposal the model-checking algorithm

and she can afford the computational resources required by that algorithm, then she indeed can compute her knowledge (modulo some of the subtleties regarding the application of Proposition 3.2.1 that were discussed in Section 3.2). Of course, Alice may not have the necessary computational resources required by the algorithm if the model has too many states, or if the formula she is trying to check is too large. For example, in systems for simultaneous Byzantine agreement, the number of global states is exponential in the number of processes, which explains the lower bounds of Theorem 6.6.10.

But even in situations where an agent has the required computational resources, the agent may not be able to explicitly compute what she knows if she does not have the model-checking algorithm at her disposal. This last point is crucial. It is not enough that an algorithm exists that would enable the agent to compute what she knows; the agent must have access to that algorithm. As we are about to see, explicitly representing the algorithms that the agents have at their disposal is the key ingredient of our approach.

10.2 Algorithmic Knowledge

10.2.1 Algorithmic Systems

Intuitively, we would like to say that an agent (algorithmically) knows a fact φ if she can compute that she knows φ. As we hinted earlier, we intend to model this by saying that the agent has an algorithm for deciding if she knows φ. To make this precise, we need to describe what it means for an algorithm to decide if agent i knows φ, and also what it means for the agent to "have" such an algorithm.

An agent's knowledge clearly depends on her local state. Thus, we might expect that an algorithm to decide if agent i knows φ is one that, given as input a local state ℓ and a formula φ, returns either "Yes" or "No", depending on whether agent i knows φ when she is in local state ℓ. Note that a "No" is not taken to mean that agent i knows that φ is false, but that she does not know that it is true. In general, it is too much to expect an agent to have an algorithm that can compute whether she knows φ in local state ℓ for every formula φ and every local state ℓ. We may be happy with an algorithm that works in the prescribed manner only for certain formulas φ or only on a subset of the points of the system. To deal with this, we allow an output of "?", in addition to "Yes" and "No"; a "?" output means that the agent is unable to compute whether she knows φ. Thus, we focus on algorithms that take as input a local state and a formula, and return as an answer either "Yes", "No", or "?". (We are considering only algorithms that always terminate.)

What does it mean to say that agent i "has" an algorithm for deciding φ? It is certainly not enough to say that there is some algorithm for i that gives the right answer on input φ. At a given point (r, m), either $K_i\varphi$ holds or it does not. Consider two trivial algorithms: the first is the algorithm that always says "Yes," and the second is the algorithm that always says "No." Clearly, one of them gives the right answer as to whether $K_i\varphi$ holds at (r, m). Although agent i can certainly execute both of these algorithms, we would not want to say that she "has" an algorithm to compute whether she knows φ in this case unless she also knows which of the two algorithms to use when asked φ. Part of "having" an algorithm is knowing when to use it.

We deal with this problem by assuming that the algorithm that agent i uses to compute her knowledge at a point (r, m) is part of her local state at (r, m). Thus, agent i "has" the algorithm in her local state. We do not mean to imply that the agent necessarily has the same algorithm at every point in the system. An agent may have different algorithms at her disposal at different points. In general, the algorithm the agent uses may depend on the information she receives. Indeed, the information she receives may allow her to come up with a new algorithm.

Formally, we now model agent i's local state as a pair $\langle \mathtt{A}, \ell \rangle$, where $\mathtt{A}$ is an algorithm, and ℓ is the rest of her local state. We call $\mathtt{A}$ the agent's *local algorithm* and ℓ the agent's *local data*. We call local states of this form *algorithmic states*. An interpreted system in which the local states of the agents are algorithmic is called an *algorithmic system*. For $r_i(m) = \langle \mathtt{A}, \ell \rangle$, we use $\mathtt{alg}_i(r, m)$ to denote the algorithm $\mathtt{A}$, and $\mathtt{data}_i(r, m)$ to denote i's local data ℓ. Intuitively, when presented with a formula φ in local state $\langle \mathtt{A}, \ell \rangle$, agent i runs $\mathtt{A}$ on input (φ, ℓ), to determine if $K_i\varphi$ holds. (By $K_i\varphi$ here we refer to the notion of implicit, information-based, knowledge that we have been using all along.) Thus, the output of $\mathtt{A}$ on (φ, ℓ) is "Yes", which means that $K_i\varphi$ holds, "No", which means that $K_i\neg\varphi$ holds, or "?", which means that neither $K_i\varphi$ nor $K_i\neg\varphi$ hold.

The separation between an agent's local algorithm and her local data in algorithmic systems allows us to distinguish the agent's operational knowledge ("knowing how") from her factual knowledge about the world ("knowing that"). The operational knowledge is captured by the agent's local algorithm, while the factual knowledge is captured by her local data. Note that it is not always obvious how to partition a local state into local algorithm and local data (just as it is not always obvious how to divide a system into agents and environment; see Section 4.1). For example, as we discuss later, a cryptographic *key*, which is a secret piece of data used by a decryption procedure, can be viewed either as local data or as part of the decryption procedure, and therefore as part of the local algorithm. In general, there is more than one way to "cut the cake"; the appropriate choice is application dependent.

Algorithmic systems can model both our earlier notion of implicit, externally ascribed knowledge, and a notion of explicit knowledge that we call *algorithmic* knowledge. As before, we denote implicit knowledge by the modal operator K_i, while algorithmic knowledge is denoted by the modal operator X_i. Implicit knowledge is defined as before:

$(\mathcal{I}, r, m) \models K_i\varphi$ iff $(\mathcal{I}, r', m') \models \varphi$ for all (r', m') such that $(r, m) \sim_i (r', m')$.

For explicit knowledge, we have

$(\mathcal{I}, r, m) \models X_i\varphi$ iff $\mathtt{A}(\varphi, \ell) =$ "Yes", for $\mathtt{A} = \mathtt{alg}_i(r, m)$ and $\ell = \mathtt{data}_i(r, m)$.

Thus, agent i has algorithmic knowledge of φ at a given point if the agent's algorithm at that point answers "Yes" when presented with φ and with the agent's local data. (Note that both the answer "No" and the answer "?" result in lack of algorithmic knowledge.) We say that a local algorithm *claims* (that agent i knows) a formula φ at a given point if it outputs "Yes" when presented with φ and i's local data at that point.

Notice that our definition makes clear that computing whether an agent knows φ has essentially nothing to do with computing whether φ is valid. An agent may certainly know a formula φ that is not valid. Thus, the fact that checking validity in $S5_n$ is *PSPACE*-complete (see Section 3.5) does not indicate that computing knowledge in any particular situation will necessarily be hard. In contrast, as we shall see, there is a connection between computing knowledge and the *model-checking* problem, that is, the problem of checking whether a formula is true at a particular point in the system.

While the definition of algorithmic knowledge draws a direct connection between an agent's explicit knowledge and the need to compute this knowledge, $X_i\varphi$ as defined is a notion of belief, since $X_i\varphi$ may hold even if φ does not hold. An algorithm could very well claim that agent i knows φ (i.e., output "Yes") whenever it chooses to, including at points where φ does not hold. This is not so unreasonable in practice; agents do make mistakes!

Although algorithms that make mistakes are common, we are often interested in local algorithms that are correct, or "sound." Formally, a local algorithm $\mathtt{A}$ is called *sound* for agent i in the algorithmic system $\mathcal{I}$ if for all points (r, m) of $\mathcal{I}$ and formulas φ, if $\mathtt{alg}_i(r, m) = \mathtt{A}$ and $\mathtt{data}_i(r, m) = \ell$, then (a) $\mathtt{A}(\varphi, \ell) =$ "Yes" implies $(\mathcal{I}, r, m) \models K_i\varphi$, and (b) $\mathtt{A}(\varphi, \ell) =$ "No" implies $(\mathcal{I}, r, m) \models \neg K_i\varphi$. It follows from condition (a) that at a point where agent i uses a sound local algorithm, $X_i\varphi \Rightarrow K_i\varphi$ holds. Thus, if all the local algorithms used by agent i in a given system $\mathcal{I}$ are sound, then X_i satisfies the Knowledge Axiom: $\mathcal{I} \models X_i\varphi \Rightarrow \varphi$.

Soundness of local algorithms is clearly a desirable property, since acting based on knowledge is, in general, better than acting based on beliefs that may turn out to be wrong.

In a precise sense, when $X_i\varphi$ holds at a point where agent i uses a sound local algorithm, it is appropriate to say that i is able to compute that she knows φ. Soundness is a *safety* property; it is a guarantee that wrong answers are not given. Notice that an algorithm does not have to be very sophisticated in order to be sound. In fact, an algorithm that never outputs "Yes" and never outputs "No" (i.e., always outputs "?") is clearly sound. We are often interested, in addition to soundness, in a guarantee that the algorithm always yields a definite ("Yes" or "No") answer. This motivates the following definition. A local algorithm $\mathtt{A}$ is called *complete* for agent i in the algorithmic system $\mathcal{I}$ if for all points (r, m) of $\mathcal{I}$ and formulas φ, if $\mathtt{alg}_i(r, m) = \mathtt{A}$ and $\mathtt{data}_i(r, m) = \ell$, then $\mathtt{A}(\varphi, \ell) \in$ {"Yes", "No"}. It follows that at a point where agent i uses a sound and complete local algorithm, $X_i\varphi \Leftrightarrow K_i\varphi$ holds.

For many applications of interest, completeness is too strong a requirement. For example consider the knowledge-based program SBA_i for the simultaneous Byzantine agreement problem described in Section 7.4. This program has two tests: $\neg decided_i \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0)$ and $\neg decided_i \wedge \neg B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 0) \wedge B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists 1)$. Recall that a formula of the form $B_i^{\mathcal{N}}\varphi$ is an abbreviation for $K_i(i \in \mathcal{N} \Rightarrow \varphi)$. Thus, to implement this knowledge-based program, it suffices to find an algorithm that can compute the truth of knowledge tests of the form $B_i^{\mathcal{N}} C_{\mathcal{N}}(\exists y)$. Moreover, the algorithm does not have to be able to compute the truth of these tests at all points; only at ones where $\neg decided_i$ holds. This leads us to the following weakening of the completeness requirement: Given a set Σ of formulas and a set L of local data for i in an algorithmic system $\mathcal{I}$, a local algorithm $\mathtt{A}$ is *complete with respect to* Σ and L if for all points (r, m) in $\mathcal{I}$ such that $\mathtt{alg}_i(r, m) = \mathtt{A}$ and $\mathtt{data}_i(r, m) \in L$, and for all formulas $\psi \in \Sigma$, it is the case that $\mathtt{A}(\psi, \mathtt{data}_i(r, m)) \in$ {"Yes", "No"}. Thus, on the formulas and local data of interest (as specified by Σ and L), a complete algorithm always gives a definite answer. (A restricted version of soundness can be defined in an analogous manner.) To implement the knowledge-based program SBA_i, we need an algorithm that is sound and complete only with respect to Σ and L, where Σ consists of the two formulas $i \in \mathcal{N} \Rightarrow C_{\mathcal{N}}(\exists 0)$ and $i \in \mathcal{N} \Rightarrow C_{\mathcal{N}}(\exists 1)$, and L consists of the local data in which i has not performed the decide_i action (recall that we assume that i records in its local state whether it has decided).

The soundness of an agent's local algorithm is related to the agent's rationality. One interpretation of an agent's being rational is that he would not act based on incorrect information. Similarly, the completeness of agent's local algorithm is related to the agent's expertise. Intuitively, the less often an agent's local algorithm

gives the answer "?," the more expert the agent is. Notice that if an agent has a sound algorithm for computing whether he knows φ at a particular local state, then he essentially has an algorithm for checking if $K_i\varphi$ holds at a point where he is in that local state; thus, he can do a limited form of model checking.

10.2.2 Properties of Algorithmic Knowledge

As we already observed, algorithmic knowledge and implicit knowledge essentially coincide as long as the agents are using sound and complete algorithms. In particular, $X_i\varphi \Rightarrow \varphi$ holds at all points where agent i's local algorithm is sound, as does $X_i\varphi \Rightarrow K_i\varphi$, while $K_i\varphi \Leftrightarrow X_i\varphi$ holds at all points where i's local algorithm is sound and complete.

It is easy to see that, in general, algorithmic knowledge does not suffer from any of the forms of logical omniscience discussed in Chapter 9 (see Exercise 10.1). For example, there is no need for an agent to algorithmically know any tautologies, nor to know the logical consequences of his algorithmic knowledge. Indeed, if we put no constraints on the algorithms, then there are no interesting properties of algorithmic knowledge.

Since the behavior of the algorithm in algorithmic systems may clearly depend on the syntax of φ, algorithmic knowledge shares many of the properties of knowledge in syntactic structures, as discussed in Section 9.2.1 (see Exercise 10.1). In fact, there is a close connection between algorithmic systems and syntactic structures; see Exercise 10.2.

The syntactic nature of algorithmic knowledge also yields a close connection between algorithmic knowledge and the notion of awareness from Section 9.5. Recall that in the logic of awareness, we have a modal operator A_i, and the intended meaning of the formula $A_i\varphi$ is "agent i is aware of φ." The agent may be aware of some formulas at a given state, and not be aware of other formulas. In the context of algorithmic systems, we can define agent i to be aware of φ at a point (r, m) of system $\mathcal{I}$, denoted $(\mathcal{I}, r, m) \models A_i\varphi$, if $\mathtt{A}(\varphi, \ell) \neq$ "?" where $r_i(m) = \langle \mathtt{A}, \ell \rangle$. Thus, we think of i as being aware of φ if his local algorithm answers either "Yes" or "No" when presented with φ. If $\mathtt{alg}_i(r, m)$ is a sound algorithm, then we have that $(\mathcal{I}, r, m) \models X_i\varphi \Leftrightarrow K_i\varphi \wedge A_i\varphi$. In fact, for every algorithmic system $\mathcal{I}$ in which the local algorithms used by the agents are sound, there is an awareness structure M, whose states are the points in $\mathcal{I}$, such that the same formulas are true at corresponding points in M and $\mathcal{I}$ (see Exercise 10.3).

Explicit knowledge in algorithmic structures has a property that it does not have in general in awareness structures. Since a local algorithm takes only the local, and

not the global, state as an argument, algorithmic knowledge depends only on the local state. Thus, an agent (implicitly) knows whether or not he has algorithmic knowledge. That is, $X_i\varphi \Rightarrow K_i X_i\varphi$ and $\neg X_i\varphi \Rightarrow K_i \neg X_i\varphi$ are both valid in algorithmic systems (Exercise 10.4). This means that algorithmic systems have the locality property discussed in Section 9.2: knowledge depends only on agents' local states.

10.3 Examples

A framework is useful only if it can capture a wide variety of problems in a natural way. As we now show by example, algorithmic knowledge can indeed capture many situations of interest in which agents need to act on their knowledge.

Players' skill level Consider the card game of blackjack. In this game, players of different skills vary in both their local data and local algorithms. The local data of a naive player is often just the information about the cards in his hand and the exposed cards on the table. Such a player would typically also use a rather simple local algorithm to decide whether or not to draw more cards. The local data of a more sophisticated player may also include information about cards that were exposed in previous rounds of the game (a practice called card counting). Such a player would typically also have a more sophisticated local algorithm at his disposal. It is well known that while the odds are against a naive player of blackjack, they favor a sufficiently sophisticated card counter. Thus, a sophisticated player gets an advantage both from having more detailed local data and from using a more sophisticated local algorithm.

In the *ten-count system*, the player keeps track of the number of tens (where a "ten" is a ten, jack, queen, or king) and the number of non-tens that are left in the deck; the player's decision as to what to do next depends on the ratio of the number of remaining non-tens to the number of remaining tens. There are other card-counting systems, that keep track of other information. We can view card-counters who use different card-counting systems as following the same knowledge-based program: "Take an action (such as drawing another card) that you know gives you the largest expected profit." Nevertheless, even though they follow the same knowledge-based program, card counters who use different card-counting schemes would take different actions, since they differ in their local data and their local algorithms.

Moving beyond games, these ideas can also be applied to trading activity in financial markets. Traders vary both in the raw information available to them (the

local data) as well as in the way they interpret this information (the local algorithm). Traders gain advantage by getting more information (perhaps even "inside information") and by using more sophisticated algorithms for estimating, for example, what the future value of a stock or commodity is likely to be. Indeed, it is well-known that professional investors have the advantage in the futures and options markets; the vast majority of non-professional investors in those markets typically lose a significant portion of their investment.

Cryptography Another application that can be captured in the framework of algorithmic systems is modern-day cryptography. A recent trend in cryptography is to use computational difficulty to guarantee the security of encryption. In *public-key cryptography*, messages are encrypted by applying a publicized function to the original text of the message. These functions are chosen in such a way that decrypting an encrypted message is easy for an agent who possesses a secret key (such as the factorization of a particular large number), while an agent with no access to such a key would have to use a great deal of computation power to decrypt the message. Thus, while all of the information about the original content of the message is encoded in the encrypted message (given the publicized encryption function), the content is inaccessible without the secret key. Suppose that i possesses the secret key, while j does not. Agent i then has at his disposal an efficient decryption algorithm, and hence will explicitly know the content of encrypted messages he receives. Agent j, not possessing the secret, will not explicitly know the content of such encrypted messages. On the other hand, j would have implicit knowledge of the content of each message she receives, although, because it is implicit, this knowledge would probably be of no practical use to her.

Notice that the main source of i's uncertainty regarding j here is not what j knows: i knows that j knows the encrypted message and does not know the secret. (Actually, the latter is probably belief rather than knowledge, but that is not the issue here.) Rather, it is uncertainty regarding j's algorithm. Such uncertainty is precisely the type of uncertainty we need to model in cryptographic settings, where the question of whether an opponent has an algorithm allowing him to break a code is crucial.

We discussed earlier the choice involved in partitioning a local state into the local algorithm and the local data. To demonstrate the subtlety of such a choice, consider what happens in the cryptography example when agent j, who did not have the secret key originally, somehow obtains this key. Clearly, this can have a drastic effect on j's algorithmic knowledge. We have a choice regarding how to view the event of j's obtaining the secret key: we can view it either as changing j's local

algorithm or as modifying j's local data. The most appropriate way of modeling this situation depends in part on what we expect. If j is expecting to learn the key at some point, then her local algorithm most likely has a provision for what to do once the key is learnt. In this case, obtaining the key would result in a change in the local data. On the other hand, if j never expected to obtain the key, and then learns it serendipitously, this would most likely result in a change in j's local algorithm.

Knowledge bases Knowledge bases provide another example of algorithmic systems. As we saw in Section 4.4.1, after the KB is told the propositional facts $\varphi_1, \ldots, \varphi_k$ whose conjunction is κ, its local data consists essentially of κ. The KB can then claim a propositional fact φ precisely when $K_{KB}\kappa \Rightarrow K_{KB}\varphi$ is valid in $\mathcal{M}_n$. Thus, in general, query answering requires the KB to decide validity of propositional implications, which is co-*NP*-complete (see Section 3.5). In practice, the KB answers queries by evaluating algorithmic knowledge rather than implicit knowledge. That is, the KB applies some local algorithm `alg` to its local data κ and the query φ, answers "Yes" if $\mathtt{alg}(\varphi, \kappa)$ = "Yes" and "I don't know" otherwise. Since we expect the KB to be truthful, the local algorithm of the KB should be sound. Furthermore, we expect the algorithm to be efficient; in fact, the answer "I don't know" may indicate that the algorithm failed to reach an answer within a prescribed time limit.

The results of Section 9.3.3 can be viewed as the description of an efficient sound local algorithm. We saw in Section 9.3.3 that, in the nonstandard setting, queries can be answered efficiently under certain assumptions. More precisely, Theorem 9.3.6 asserts that if κ and φ are propositional formulas in conjunctive normal form (CNF), then determining whether $K_i\kappa$ logically implies $K_i\varphi$ in the nonstandard semantics of Section 9.3 can be carried out in polynomial time (in the size of κ and φ). We now consider what this means in terms of algorithmic knowledge.

Proposition 9.3.7 implies that any positive answer we obtain from testing logical implication between CNF formulas in nonstandard semantics provides us with a positive answer for standard semantics as well. Thus, a local algorithm of the knowledge base could very well limit itself to testing logical implication between κ and a query φ with respect to nonstandard semantics. If the logical implication holds, it should output "Yes"; otherwise, it should output "I don't know." If the KB is told only formulas in CNF, then such a local algorithm is clearly sound with respect to all formulas in CNF, although it is not in general complete. By Theorem 9.3.6, this local algorithm is polynomial in terms of the size of the KB's local data and the size of the query.

10.4 Algorithmic Programs

10.4.1 Algorithmic Programming

We argued in Chapter 7 that knowledge-based programs can be used to describe the relationship between knowledge and action. Intuitively, knowledge-based programs prescribe what actions the agents should take as a function of their local state and their knowledge. We explained, however, that knowledge-based programs are not directly "runnable," since knowledge is implicit rather than explicit. Motivated by this, we introduced and studied the notion of implementing knowledge-based programs by standard programs. The concept of algorithmic knowledge suggests another solution to the "non-runnability" of knowledge-based programs; we can consider programs that prescribe what actions the agents should take as a function of their local state and their *explicit* knowledge. We call such programs *algorithmic* programs.

Formally, an algorithmic program for agent i has the following form:

case of
  if $t_1 \wedge x_1$ **do** a_1
  if $t_2 \wedge x_2$ **do** a_2
  ...
end case

where the t_j's are standard tests, the x_j's are *algorithmic knowledge tests* for agent i, and the a_j's are actions of agent i. An algorithmic knowledge test for agent i is a Boolean combination of formulas of the form $X_i\varphi$. Intuitively, the agent selects an action based on the result of applying the standard test to her local state and applying the knowledge test to her "explicit knowledge state." In any given clause, we can omit either the standard test or the knowledge test; thus, a standard program is a special case of an algorithmic program.

We now describe the formal semantics of algorithmic programs. The semantics is very close to the semantics we described in Section 5.3 for standard programs. We assume that all local states are algorithmic. We also assume that we are given an interpretation π over the set $\mathcal{G}$ of global states that is compatible with the agents' programs, i.e., every proposition that appears in a standard test of the program Pg_i is local to i.

Let $\langle \mathtt{A}, \ell \rangle$ be an algorithmic state, t_j a standard test in Pg_i, and x_j an algorithmic knowledge test in Pg_i. Since π is compatible with Pg_i, satisfaction of t_j in $\langle \mathtt{A}, \ell \rangle$ with respect to π, denoted $(\pi, \mathtt{A}, \ell) \models t_j$, is well defined. Satisfaction of x_j in $\langle \mathtt{A}, \ell \rangle$, denoted $\langle \mathtt{A}, \ell \rangle \models x_j$, is also well defined: $\langle \mathtt{A}, \ell \rangle \models X_i\varphi$ if $\mathtt{A}(\ell, \varphi)$="Yes";

this is extended to Boolean combinations in the standard way. We use the notation $(\pi, \mathtt{A}, \ell) \models t_j \wedge x_j$ to denote the fact that $(\pi, \mathtt{A}, \ell) \models t_j$ and $\langle \mathtt{A}, \ell \rangle \models x_j$. We can now define the protocol Pg_i^π by setting

$$\mathsf{Pg}_i^\pi(\mathtt{A}, \ell) = \begin{cases} \{\mathsf{a}_j : (\pi, \mathtt{A}, \ell) \models t_j \wedge x_j\} & \text{if } \{j : (\pi, \mathtt{A}, \ell) \models t_j \wedge x_j\} \neq \emptyset \\ \{\Lambda\} & \text{if } \{j : (\pi, \mathtt{A}, \ell) \models t_j \wedge x_j\} = \emptyset. \end{cases}$$

As with standard programs, and in contrast to knowledge-based programs, the actions selected here depend only on the agent's local state, and not on the interpreted system.

So far we have focused on programs for individual agents. The notions of joint algorithmic programs and the interpreted system that represents a joint algorithmic program in a given context can now be defined as we did for standard programs in Section 5.3. We leave the details to the reader.

In many applications, algorithmic programs capture our intuition of "action based on knowledge" better than knowledge-based programs do. Recall that we motivated applying the knowledge terminology to distributed systems in Chapter 1 by pointing out that designers of distributed programs often say things like: "once A knows B has received the message μ, then A will stop sending μ to B." A program based on such an intuition will typically involve having B send A an acknowledgment when B receives μ, and having A stop sending μ once it receives such an acknowledgment. It may happen that before receiving this acknowledgment A already has enough information to infer that B has received μ. For example, this is the case if A received a message from a third agent C that could not have been sent unless B had already received μ. In practice, A would usually not try to detect such knowledge (unless stopping to send μ as early as possible is absolutely essential); rather, it would continue to send μ until the particular test for the acknowledgment succeeds. We may conclude that the reference to knowledge in the designers' intuition above does not quite refer to implicit knowledge. Rather, it can be thought of as assuming that there are particular tests (in our example, checking whether an acknowledgment has arrived) on which the knowledge is based. Clearly, these tests give rise to local algorithms for the processes to use in testing for knowledge about the relevant facts of interest. Thus, what the designers often have in mind is an algorithmic notion of knowledge rather than an information-based one. This is exactly what the notion of algorithmic programs is intended to capture.

This discussion suggests that reasoning about implicit knowledge may not suffice in the analysis of many complex algorithms; reasoning in terms of algorithmic knowledge may be necessary as well. One such complex situation that can be found in the literature is in the design and description of programs for Byzantine agreement in the Byzantine-failure mode, where faulty processes can behave arbitrarily

and lie outright. In this case, computing whether a process i implicitly knows that another process j is faulty may be a very difficult task in general. Done naively, it involves performing exponential computations, and even in the most efficient way it often requires solving co-*NP*-hard problems (see Theorem 6.6.10). The information that j is faulty, however, may be very useful for i. Moreover, there are various simple computational tests that i can employ which, when successful, imply that j is faulty. This can be viewed as giving a sound local algorithm for testing whether j is faulty. In developing a program for Byzantine agreement in this case, it is possible to use such a sound local algorithm and then reason about statements such as "*if i knows that j is faulty then it should ...*." Again, the knowledge referred to in such a statement is actually the algorithmic knowledge i has about whether j is faulty.

10.4.2 Algorithmic Knowledge and Complexity

The computational resources that an agent can actually use when running a local algorithm are typically quite limited. One reason may be the cost and availability of the computational resources. Another is the need of the agent to act within a given time frame. The result of a computation that takes a year to complete may well be useless by the time it is obtained. It is therefore essential to consider not only the final output of a computation but also the complexity of this computation. One of our motivations for introducing algorithmic knowledge was the fact that the notion of implicit knowledge is not sensitive enough to take into account the limited computational power of the agents.

The algorithmic framework can model the limited abilities of resource-bounded agents. Indeed, we can model the fact that an agent i knows that j is restricted to, say, polynomial-time computations by having j's local algorithms at all points that agent i considers possible be polynomial-time algorithms. Under such circumstances, i would (implicitly) know that j is limited to polynomial-time computations. As we saw above, much finer distinctions regarding j's computational abilities can be made, such as assuming that j cannot perform certain specific computations (decrypting a particular message, for example). This is much stronger than assuming j's algorithm is from a particular complexity class. Of course, it may be rather difficult to actually obtain such strong knowledge about the algorithms used by another agent.

There are subtleties in using the algorithmic framework to model the limited abilities of resource-bounded agents. To start with, it is far from obvious how the complexity of computing knowledge ought to be measured. As we observed in Section 3.5, we typically measure the complexity of an algorithm as a function of the size of its input. For example, in Section 3.5 we measured the complexity of

determining the validity of a formula φ in various modal logics as a function of $|\varphi|$, the number of symbols in the syntactic representation of φ. In algorithmic systems, the algorithm takes as input a pair (φ, ℓ) consisting of a formula φ and local data ℓ. Thus, the complexity of claiming φ should be a function of $|\varphi|$ and $|\ell|$, which is essentially how we did it in Section 9.3.3.

There are times, however, when it may be appropriate to suppress the dependence of the complexity on the formula φ or on the local data ℓ. For example, if we ask many queries of a knowledge base in a fixed state, then it may be appropriate to measure the complexity simply as a function of the size of the query φ, and treat the local data (the state of the knowledge base) as a constant in our calculations. By way of contrast, an agent running a fixed knowledge-based program needs to compute only whether she knows one of a small number of formulas, possibly at many different points. Thus, in this case, it is appropriate to ignore the size of the formula (i.e., treat it as a constant), and compute the complexity as a function of the size of the local data.

The issue is in fact more subtle than these examples suggest. Consider the knowledge-based program SBA for simultaneous Byzantine agreement given in Section 7.4. In that program, there are only two formulas we care about, namely $C_{\mathcal{N}}(\exists 0)$ and $C_{\mathcal{N}}(\exists 1)$. But we measured the complexity, not as a function of the size of the local data, but as a function of n and t, the total number of processes and the bound on the number of failures. These were the appropriate parameters, since we were interested in understanding how the difficulty of implementing the algorithm depended on the number of participating processes and the potential number of failures. The upshot of this discussion is that there is not a single "right" way of defining the complexity of computing knowledge of a formula; the "right" definition will depend on the application.

Another subtlety in using the algorithmic framework to model the limited abilities of resource-bounded agents has to do with the notion of time. We often classify computational complexity by analyzing the time requirement of the computation. For example, we mentioned earlier that the resource-boundedness of an agent could be modeled by allowing her access only to polynomial-time algorithms. Recall that a run is a function from time to global states. But what is the relationship between the notion of time when we speak of polynomial-time algorithms and the notion of time modeled explicitly in multi-agent systems? Our semantics for algorithmic programs suggests that algorithmic knowledge tests are always evaluated in one round. But evaluating an algorithmic knowledge test $X_i\varphi$ in a state $\langle \mathtt{A}, \ell \rangle$ requires the application of $\mathtt{A}$ to (φ, ℓ), which means that the algorithm has to conclude its possibly polynomially long computation within one round.

The reason for this apparent inconsistency is that we really have two notions of time in mind. Time in runs serves as a convenient way to partition the system behavior into rounds, where a round is a basic unit of activity. In the coordinated-attack problem of Section 6.1, a round was (implicitly) taken to be a length of time sufficient for a messenger to carry a message from one general to the other. Similarly, in the SBA problem of Section 6.3, a round was taken to be sufficiently long for a process to send a message to all other processes. In both cases, our choice of the granularity of time is motivated by convenience of modeling, and time should not be thought of as "real time."

When we speak of time in the context of (polynomial-time) computations, we are not speaking of "real time" either. Rather, the notion of time in complexity is meant to measure the number of machine instructions performed by a computational device. We argued above that there is no unique way to measure the complexity of computing knowledge. Similarly, the precise relationship between the two notions of time is also application dependent. For example, when considering SBA, a machine instruction could be "transmit one bit" and a round could include up to some polynomial number of such instructions. Thus, when we model an algorithmic system, the local algorithms have to be such that they can be run in one round. If one wishes to model an algorithm with a longer running time, then it must be split up among successive states.

To understand this issue better, let us return to our example of the KB again. Suppose that we have a KB that has been told the propositional facts $\varphi_1, \ldots, \varphi_k$ whose conjunction is κ, and the KB wants to respond to a query φ. Moreover, suppose it does so by checking if $\kappa \Rightarrow \varphi$ is valid using a theorem prover. Rather than assuming that the theorem prover returns its answer in one step (or gives up, again in one step), we now take a time step to be just long enough for the theorem prover to make one inference. In this setting, it might be appropriate to take the KB's local state to be the set of consequences of κ which it has managed to deduce thus far, and say that the KB explicitly knows φ only if φ is one of the formulas it has deduced. The KB's local algorithm is now quite trivial: The KB explicitly knows φ only if φ is one of the formulas that has been deduced thus far.

10.4.3 Implementing Knowledge-Based Programs

As we emphasized in Chapter 7, one of the fundamental questions when dealing with knowledge-based programs is how to implement them by standard programs. While a knowledge-based program is stated in terms of implicit knowledge, an implementation that replaces knowledge tests by standard tests can make use only of knowledge

that is explicitly computable. Thus, it is natural to consider the implementation of knowledge-based programs by algorithmic programs.

Recall from Section 7.1 that a standard program Pg_s is an implementation of a knowledge-based program Pg_{kb} in an interpreted context (γ, π) if the protocol Pg_s^{π} coincides with the protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$, where $\mathcal{I} = \mathbf{I}^{rep}(\mathsf{Pg}_s, \gamma, \pi)$ is the interpreted system that represents Pg_s in (γ, π). Similarly, we say that an algorithmic program Pg_a *implements* Pg_{kb} in (γ, π) if Pg_a^{π} coincides with the protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$, where $\mathcal{I} = \mathbf{I}^{rep}(\mathsf{Pg}_a, \gamma, \pi)$. Intuitively, Pg_a implements Pg_{kb} if the two programs prescribe the same actions in the interpreted system that represents Pg_a.

We mentioned earlier that we do not have a general methodology for deriving implementations of knowledge-based programs. In the algorithmic setting, a reasonable approach is to replace all knowledge tests by algorithmic knowledge tests. Given a knowledge-based program Pg, we denote by Pg^X the program obtained by replacing each knowledge test $K_i\varphi$ in Pg by the algorithmic knowledge test $X_i\varphi$. (Note that we are replacing K_i by X_i only at the outermost level; for example, the test $K_iK_j\varphi$ is replaced by $X_iK_j\varphi$.) Thus, rather than testing for implicit knowledge as in Pg, in Pg^X we test for algorithmic knowledge. Does Pg^X implement Pg? We now describe some conditions under which it does.

We say that a knowledge-based program $\mathsf{Pg} = (\mathsf{Pg}_1, \ldots, \mathsf{Pg}_n)$ is in *normal form* if every program Pg_i in Pg is a finite program of the form

case of
  if $K_i\varphi_1$ **do** a_1
  ...
  if $K_i\varphi_m$ **do** a_m
end case

where the actions a_j are pairwise distinct (i.e., $\mathsf{a}_j \neq \mathsf{a}_k$ for $j \neq k$). Intuitively, in this representation, all the tests that enable the choice of a given action have been combined into one test. Normal-form programs have the property that there is a one-to-one correspondence between actions and knowledge tests. We may restrict attention to programs in normal form because, in a precise sense, every finite knowledge-based program is equivalent to one in normal form (Exercise 10.5).

Let $\mathbf{A} = \langle \mathtt{A}_1, \ldots, \mathtt{A}_n \rangle$ be a sequence of local algorithms, which we view as giving the local algorithms for the agents. For a global state $s = (s_e, s_1, \ldots, s_n)$, let $s^{\mathbf{A}}$ be the global state $(s_e, \langle \mathtt{A}_1, s_1 \rangle, \ldots, \langle \mathtt{A}_n, s_n \rangle)$. Thus, $s^{\mathbf{A}}$ is obtained from s by taking the algorithms in $\mathbf{A}$ to be the local algorithms of the agents, and making what originally were the local states of the agents be their local data. We can now

use $\mathbf{A}$ to "annotate" sets of global states, runs, systems, and interpreted systems. For a set $\mathcal{G}$ of global states, let $\mathcal{G}^{\mathbf{A}} = \{s^{\mathbf{A}} \mid s \in \mathcal{G}\}$. For a run r over $\mathcal{G}$, we define a run $r^{\mathbf{A}}$ so that if $r(m) = s$ then $r^{\mathbf{A}}(m) = s^{\mathbf{A}}$. For a system $\mathcal{R}$, let $\mathcal{R}^{\mathbf{A}} = \{r^{\mathbf{A}} \mid r \in \mathcal{R}\}$. The algorithmic system $\mathcal{R}^{\mathbf{A}}$ is said to be *uniform*, since an agent has the same local algorithm at all points of the system. For an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$, let $\mathcal{I}^{\mathbf{A}} = (\mathcal{R}^{\mathbf{A}}, \pi)$.

In analogy to associating with a "standard" global state s an annotated algorithmic global state $s^{\mathbf{A}}$, we can associate with a context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ an annotated context $\gamma^{\mathbf{A}} = (P_e, \mathcal{G}_0^{\mathbf{A}}, \tau^{\mathbf{A}}, \Psi^{\mathbf{A}})$, where $\mathcal{G}_0^{\mathbf{A}} = \{s^{\mathbf{A}} \mid s \in \mathcal{G}_0\}$, $\Psi^{\mathbf{A}} = \{r^{\mathbf{A}} \mid r \in \Psi\}$, and $\tau^{\mathbf{A}}$ is the natural extension of the transition function τ to $\mathcal{G}^{\mathbf{A}}$: the actions simply have no effect on the local algorithms, and have the same effect on the local data as in γ. Similarly, we can associate with an interpretation π for a context γ a corresponding interpretation $\pi^{\mathbf{A}}$ for $\gamma^{\mathbf{A}}$.

We can now describe conditions under which the strategy of going from Pg to Pg^X works. Recall from Section 5.2 that a computable protocol is a protocol where there is an algorithm that takes a local state as input and returns the set of actions prescribed by the protocol in that state.

Theorem 10.4.1 *Let* Pg *be a knowledge-based program in normal form and let* (γ, π) *be an interpreted context such that* π *is compatible with* Pg. *Then there is a computable protocol* P *that implements* Pg *in the context* (γ, π) *if and only if there is a sequence* $\mathbf{A}$ *of local algorithms such that* Pg^X *implements* Pg *in the context* $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$.

Proof Suppose that Pg^X implements Pg in $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$, and let $\mathcal{I}^{\mathbf{A}} = \mathbf{I}^{rep}(\mathsf{Pg}^X, \gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$. Since $\mathcal{I}^{\mathbf{A}}$ is uniform, the actions selected by Pg^X in $\mathcal{I}^{\mathbf{A}}$ depend only on the local data and the local algorithms in $\mathbf{A}$. Thus, we can derive from Pg^X and $\mathcal{I}^{\mathbf{A}}$ a computable protocol that implements Pg in (γ, π) (Exercise 10.6).

Conversely, suppose that there is a computable protocol $P_s = (P_e, P_1, \ldots, P_n)$ that implements Pg in $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$. Let the knowledge tests appearing in Pg_i be $K_i\varphi_1, \ldots, K_i\varphi_m$. (Here we use the one-to-one correspondence between actions and knowledge tests in normal form programs.) We define $\mathtt{A}_i$ as follows: $\mathtt{A}_i(\varphi_j, \ell) =$ "Yes" if $\mathsf{a}_j \in P_i(\ell)$ and "No" otherwise. For $\psi \notin \{\varphi_1, \ldots, \varphi_m\}$, we assume arbitrarily that $\mathtt{A}_i(\psi, \ell) =$ "No" for all ℓ. It is easy to see that Pg^X implements Pg in $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$ (Exercise 10.6). ∎

Intuitively, Theorem 10.4.1 captures the fact that implementing a knowledge-based program by a computable protocol amounts to replacing the tests for "external" implicit knowledge by tests for explicit algorithmic knowledge. Notice that the

assumptions of Theorem 10.4.1 impose a strong requirement on the local algorithms. Suppose that Pg^X implements Pg in $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$, and let $\mathcal{I}^{\mathbf{A}} = \mathbf{I}^{rep}(\mathsf{Pg}^X, \gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$. This means that at every point of $\mathcal{I}^{\mathbf{A}}$, $\mathtt{A}_i(\varphi_j, \ell)$ ="Yes" if and only if $(\mathcal{I}, \ell) \models K_i\varphi_j$. The algorithms must therefore be sound and complete with respect to the knowledge tests for the formulas $\Sigma = \{\varphi_1, \ldots, \varphi_m\}$ appearing in the program Pg and with respect to all the instances of local data at which the program is evaluated, which often means at all points of the system.

Exercises

10.1 Give an example of an algorithmic system with sound local algorithms in which algorithmic knowledge fails to satisfy the following types of logical omniscience from Chapter 9:

(a) knowledge of valid formulas,

(b) closure under logical implication,

(c) closure under logical equivalence.

10.2 In this exercise, we relate algorithmic systems and syntactic structures. If φ is an $\mathcal{L}_n$-formula φ (i.e., one not involving the explicit knowledge operators X_i), define φ' to be the result of replacing each K_i in φ by X_i. Let $\mathcal{I}$ be an algorithmic system. Construct a syntactic structure $M = (S, \sigma)$, where S consists of the points in $\mathcal{I}$, such that for each $\mathcal{L}_n$-formula φ and each state $s \in S$, we have that $(\mathcal{I}, s) \models \varphi'$ iff $(M, s) \models \varphi$.

10.3 In this exercise, we relate algorithmic systems and awareness structures. Show that for every algorithmic system $\mathcal{I}$ in which the local algorithms used by all the agents are sound, there is an awareness structure M, whose states are the points in $\mathcal{I}$, such that the same formulas are true at corresponding points in M and $\mathcal{I}$.

10.4 Prove that both $X_i\varphi \Rightarrow K_iX_i\varphi$ and $\neg X_i\varphi \Rightarrow K_i\neg X_i\varphi$ are valid in algorithmic systems.

10.5 We say that two knowledge-based programs Pg_1 and Pg_2 are *equivalent* if the following two conditions hold:

(a) an interpretation π is compatible with Pg_1 iff it is compatible with Pg_2, and

(b) if an interpretation π is compatible with Pg_1 and Pg_2, then for each interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ we have that $\mathsf{Pg}_1^{\mathcal{I}} = \mathsf{Pg}_2^{\mathcal{I}}$.

That is, the two programs are equivalent if they behave identically with respect to every interpreted system.

Prove that for every finite knowledge-based program there exists an equivalent knowledge-based program in normal form.

10.6 Complete the proof of Theorem 10.4.1:

(a) Suppose that Pg^X implements Pg in (γ, π). Derive from Pg^X and $\mathcal{I}^{\mathbf{A}}$ a computable protocol that implements Pg in (γ, π).

(b) Suppose there is a computable protocol $P_s = (P_e, P_1, \ldots, P_n)$ that implements Pg in (γ, π). Define $\mathbf{A}$ as in the proof of the theorem. Show that Pg^X implements Pg in $(\gamma^{\mathbf{A}}, \pi^{\mathbf{A}})$.

Notes

One of the first works relating knowledge and resource-bounded computation is [Konolige 1986]. There have also been many attempts in the game-theoretical literature to model resource-bounded agents [Anderlini 1989; Binmore 1987; Binmore and Shin 1993; Canning 1992; Megiddo 1986; Megiddo 1989; Megiddo and Wigderson 1986; Neyman 1985; Rubinstein 1985; Shin and Williamson 1993], although the approaches taken there have been quite different from that used here. See [Binmore 1990, Chapters 5–6] for a foundational discussion. Binmore and Shin [1993] define a player's algorithmic knowledge as whatever the player can deduce using a sound recursive algorithm. They study the properties of this definition, and relate these properties to the modal logic G for provability presented by Boolos [1979].

Our framework here is due to Halpern, Moses, and Vardi [1994]. This framework was inspired by the work of Moses [1988], who was the first to define an algorithmic notion of knowledge. Our notions of soundness and completeness are based on his definitions and on those of Halpern, Moses, and Tuttle [1988]. Parikh [1987] presents a definition of a notion he calls l-knowledge (for *linguistic* knowledge) that is similar in spirit to the notion of algorithmic knowledge defined here. He considers an agent to have l-knowledge of φ if the agent says "Yes" when asked about φ and if, in addition, the agent's answers are sound in our sense.

Elgot-Drapkin and Perlis developed what they call *step-logics* to model the reasoning of agents over time [Elgot-Drapkin 1991; Elgot-Drapkin and Perlis 1990].

In their approach, the agent can carry out one step of reasoning at each time step. This is much in the spirit of the example given at the end of Section 10.4.2 of the KB that uses a theorem prover to check if $\kappa \Rightarrow \varphi$ is valid. Thus, we can embed their approach in our framework.

Extensions of the framework of Moses [1988] for applications to cryptography were made by Halpern, Moses and Tuttle [1988]. They show that the interactive and zero-knowledge proof systems of Goldwasser, Micali and Rackoff [1989] can be cast in a formalism in the spirit of the one described here, extended to allow for probabilistic computations and probabilistic notions of soundness and completeness. A related notion was described by Fischer and Zuck [1987].

The ten-count system for blackjack was developed by Thorp, who wrote the classic works on winning strategies for blackjack [1961, 1966]. The idea of public-key cryptography was introduced by Diffie and Hellman [1976]; a well-known example of a public-key cryptosystem is the RSA system of Rivest, Shamir, and Adleman [1978].

The analysis of Byzantine agreement in the Byzantine-failure mode in terms of algorithmic knowledge, referred to in Section 10.4.1, is due to Berman, Garay, and Perry [1989].

Chapter 11

Common Knowledge Revisited

Mankind, why do ye set your hearts on things
That, of necessity, may not be shared?

Dante Alighieri, "Purgatorio," 14, *The Divine Comedy*, c. 1300

The strong link between common knowledge and goals such as agreement and coordinated action was discussed in Chapter 6, where we showed that coordinated attack requires common knowledge, as does reaching simultaneous Byzantine agreement. Moreover, examples such as the muddy children puzzle demonstrate how public announcements can cause facts to become common knowledge. On the other hand, we have seen that attaining common knowledge can be difficult at times. In this chapter we take a look at the tension between the desirability of common knowledge and the difficulty in attaining it.

A critical observation that we shall make here is that common knowledge is intimately related to simultaneity: the onset of common knowledge must involve a simultaneous change at all sites. This observation will provide us with new insight into the nature of common knowledge. In particular, it strongly affects the attainability of common knowledge, since true simultaneity cannot be attained in realistic scenarios. This presents us with a dilemma. We argue for the importance of common knowledge for coordinated actions, while at the same time we show that common knowledge is not attainable in practical situations. This dilemma is the focus of this chapter.

11.1 Common Knowledge as a Conjunction

Theorems 4.5.4 and 6.1.1 show that there are a number of situations in which nontrivial common knowledge is not attainable. Much of our discussion in this chapter will be motivated by the quest to understand what makes common knowledge attainable in some situations and unattainable in others. Recall that our definition of common knowledge is in terms of an infinite conjunction of facts of the form $E^k\varphi$. This definition suggests that common knowledge has an "inherently infinite" nature. Indeed, for a fact that is not common knowledge to become common knowledge, each participating agent must come to know an infinite collection of new facts. How can common knowledge ever be attained if it requires the agents to learn an infinite collection of different facts? Could this be one of the reasons that common knowledge is hard to attain in some cases of interest? Indeed, in the coordinated attack scenario, the agents are able to satisfy any finite collection of facts of the form $E^k\varphi$, while, as we saw, they cannot attain common knowledge.

Consider an unrealistic version of the coordinated-attack problem, in which the generals can exchange an infinite number of messages in a finite amount of time. For the purpose of the argument, we can imagine that the messenger is able to double his speed every time around; while his first journey may take an hour, his second journey takes a half hour, the third a quarter hour, and so on. Thus, he will visit both camps an infinite number of times within two hours, unless he is caught by the enemy. If the messenger is not caught by the enemy, then the generals are able to attack after the messenger has completed his task. This is because, intuitively, the generals attain common knowledge after two hours. This example suggests that attaining common knowledge may require an infinite number of (communication) steps in some cases.

That this is not always the case should be clear from our discussion of the muddy children puzzle and of simultaneous Byzantine agreement (SBA). Here nontrivial common knowledge is attained after a finite amount of time and communication. The fact that each of the infinitely many facts $E^k\varphi$ holds in these cases follows from a straightforward proof by induction on k. Thus, attaining common knowledge need not require an infinite amount of communication.

If common knowledge is equivalent to E^k for some sufficiently large k, then, intuitively, k rounds of communications should suffice to attain common knowledge. The coordinated-attack scenario shows that this equivalence does not hold in general. We now discuss an important class of multi-agent systems where common knowledge is equivalent to E^k for some sufficiently large k.

In many systems, each agent's set of possible local states is finite. For example, in the bit-transmission problem of Example 4.1.1, the sender S has only four possible

local states and the receiver R has only three. In fact, in practice there is always a finite bound on the number of possible local states of an agent in a real-world system. For example, a computer has only finite (albeit typically large) memory capacity, and thus has a finite state space. We call a system where each agent's set of possible local states is finite a *finite-state system*. We now show that in a finite-state system, common knowledge is always equivalent to E^k for a sufficiently large k.

Theorem 11.1.1 *Let $\mathcal{I}$ be an interpreted finite-state system, and let G be a fixed nonempty set of agents. Then there is a constant k such that for every formula φ we have $\mathcal{I} \models C_G\varphi \Leftrightarrow E_G^k\varphi$.*

Proof It is enough to show that $\mathcal{I} \models E_G^k\varphi \Rightarrow C_G\varphi$ for some fixed k. For all agents i, let L_i be the set of agent i's local states in $\mathcal{I}$. Define $l = \min\{|L_i| : i \in G\}$. Thus, l is the number of local states of the member of G with the fewest local states. Now set $k = 2l - 1$. Assume that $(\mathcal{I}, r, m) \not\models C_G\varphi$. This means that there must be a point (r', m') that is G-reachable from (r, m) and satisfies $(\mathcal{I}, r', m') \not\models \varphi$. This, in turn, implies that there is a sequence $c_0, c_1, \ldots, c_t$ of points such that $c_0 = (r, m)$, $c_t = (r', m')$, and for all $0 \leq j < t$ there is an agent $i_j \in G$ with the same local state at c_j and c_{j+1}. Without loss of generality, assume that this path is of minimal length among all such paths connecting (r, m) and (r', m'). No agent can have the same local state at any two nonconsecutive points in the sequence, since otherwise there would be a connecting path of smaller length. It follows that $t \leq 2l - 1 = k$. As a result, $(\mathcal{I}, r, m) \not\models E_G^k\varphi$. We thus obtain that $\mathcal{I} \models E_G^k\varphi \Rightarrow C_G\varphi$. ∎

Notice that the proof still goes through if, instead of a finite-state system, we have a system in which even one member of G has a finite number of local states. Therefore, intuitively, if any member of G is bounded, then common knowledge among the agents in G is equivalent to E_G^k for some k.

One additional property of the proof of Theorem 11.1.1 is worth mentioning. Fix a finite-state system $\mathcal{I}$, and let l be as in the proof of the theorem. Moreover, let $i \in G$ be an agent with exactly l local states. Clearly, having only l different local states, agent i is not able to count up to $l + 1$. But the proof does not show that $\mathcal{I} \models E_G^{l+1}\varphi \Leftrightarrow C_G\varphi$. Rather, the proof shows that $\mathcal{I} \models E_G^k\varphi \Leftrightarrow C_G\varphi$ for $k = 2l - 1$. One might wonder whether this equivalence holds for a smaller k. The answer, in general, is no. There are examples in which $E_G^{2l-2}\varphi$ holds without $C_G\varphi$ holding (Exercise 11.1).

It would seem that Theorem 11.1.1 spells good news for common knowledge, at least for the class of finite-state systems. One might, for example, expect to attain common knowledge rapidly in a finite-state protocol in which the receiver has only

four different local states. On the other hand, Theorem 6.1.1 already shows that common knowledge of facts such as "the message has been delivered" cannot be attained if communication is not reliable. Moreover, the proof of this result did not rely on any assumptions that disallow finite-state systems; the inability to attain nontrivial common knowledge applies equally well to finite-state protocols.

There is no paradox here. In finite-state systems where communication is unreliable, it is true both that common knowledge is unattainable and that $E^k\varphi$ is equivalent to $C\varphi$ for an appropriate choice of k. We can thus conclude that in every finite-state system with unreliable communication, $E^k\varphi$ is unattainable for some sufficiently large k. One way to understand this result is to observe that the choice of l and k in the proof is such that at least one agent $i \in G$ cannot count up to $l+1$, let alone to k. For this agent, k is essentially tantamount to infinity. (We are not assuming here that in order to attain $E^k\varphi$ each agent in G needs to be able to count to k. This is false, as is shown in Exercise 11.1(a). Nevertheless, the naive method for attaining common knowledge by k rounds of information exchange essentially requires the agents to be able to count to k.) We can, of course, add a few bits to the local states of the members of G in $\mathcal{I}$, thereby enabling every agent, in principle, to count up to k. But by doing so we will have created a new finite-state system $\mathcal{I}'$ in which the number of local states these agents have has increased. As a result, we obtain a new value $l' > l$, and a corresponding value $k' > k$. In the resulting (extended) system $\mathcal{I}'$, it will no longer be the case that $\mathcal{I}' \models C_G\varphi \Leftrightarrow E_G^k\varphi$. In this system at least one process will not be able to count to k', and we are back to the original problem.

The previous discussion shows that the fact that in finite-state systems common knowledge is equivalent to E^k for some finite k does not make common knowledge any easier to attain. On the other hand, we saw in Chapter 6 that common knowledge can be attained within a finite amount of time even in some situations where common knowledge is not equivalent to E^k for any finite k. It seems that viewing common knowledge in terms of an infinite conjunction does not provide insight into the way common knowledge arises. The question of what are the basic principles underlying the attainability or unattainability of common knowledge motivates the rest of this chapter.

11.2 Common Knowledge and Simultaneity

11.2.1 Common Knowledge and Uncertainty

Theorem 6.1.1 shows that, in a strong sense, nontrivial common knowledge is not attainable in a system in which communication is not guaranteed or, for that matter,

in a system in which communication *is* guaranteed, but where there is no bound on the time it takes for messages to be delivered. It would seem that when all messages are guaranteed to be delivered within a fixed amount of time, say one nanosecond, attaining common knowledge should be a simple matter. But things are not always as simple as they seem; even in this case, uncertainty causes major difficulties. Consider the following example.

Assume that two agents, Alice and Bob, communicate over a channel in which (it is common knowledge that) message delivery is guaranteed. Moreover, suppose that there is only slight uncertainty concerning message delivery times. It is commonly known that any message sent from Alice to Bob reaches Bob either immediately or after exactly ε time units. Now suppose that at some point Alice sends Bob a message μ that does not specify the sending time in any way. Let m_S and m_D be variables denoting the time μ is sent and the time it is delivered, respectively. Thus, having sent μ, Alice knows the value of m_S, while once μ is delivered, Bob knows the value of m_D. Let $\mathcal{R}$ denote the system of runs that correspond to this example. There are many runs in $\mathcal{R}$, since we assume that the sending time m_S may be arbitrary. Given our assumption about message delivery, it is common knowledge in the system that $(m_D = m_S + \varepsilon) \vee (m_D = m_S)$. Because the events $m_D = m_S + \varepsilon$ and $m_D = m_S$ are mutually exclusive, they partition the runs of $\mathcal{R}$ into two classes, depicted by scenario (a) and scenario (b) of Figure 11.1. Let $sent(\mu)$ be the fact "*the message μ has been sent.*" Bob does not know $sent(\mu)$ initially. How does the state of knowledge of $sent(\mu)$ in the group consisting of Alice and Bob change with time?

At time m_D, Bob receives μ and hence knows $sent(\mu)$. Since it may take ε time units for μ to be delivered, as in scenario (a), Alice cannot be sure that Bob knows $sent(\mu)$ before $m_S + \varepsilon$. In particular, there is always a run indistinguishable by Alice from the current run in which Bob receives the message only at time $m_S + \varepsilon$. Thus, $K_A K_B sent(\mu)$ holds at time $m_S + \varepsilon$ and does not hold beforehand. Bob, on the other hand, knows that Alice will not know that Bob knows $sent(\mu)$ until $m_S + \varepsilon$. Since, for all Bob knows, μ may have been delivered immediately (scenario (b)), Bob does not know that $m_S + \varepsilon$ has taken place until time $m_D + \varepsilon$. Thus, $K_B K_A K_B sent(\mu)$ does not hold until time $m_D + \varepsilon$. When does $K_A K_B K_A K_B sent(\mu)$ hold? Alice knows that $K_B K_A K_B sent(\mu)$ holds at $m_D + \varepsilon$. Since, for all Alice knows, it takes time ε for the message to be delivered (scenario (a) again), m_D could be $m_S + \varepsilon$. Thus, $K_A K_B K_A K_B sent(\mu)$ does not hold until $m_S + 2\varepsilon$.

This line of reasoning can be continued indefinitely, and an easy proof by induction shows that before time $m_S + k\varepsilon$, the formula $(K_A K_B)^k sent(\mu)$ does not hold, while at $m_S + k\varepsilon$ it does. Thus, every ε time units Alice and Bob acquire an additional level of "Alice knows that Bob knows," and each such level actually requires

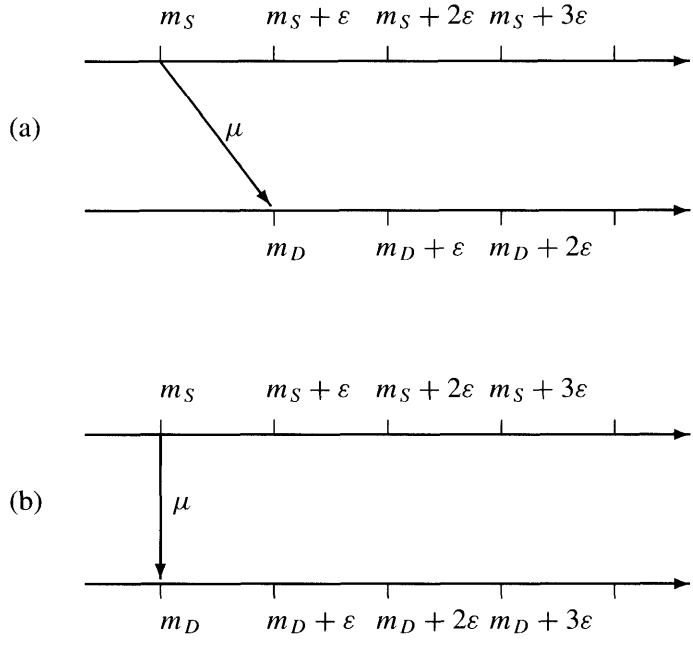

Figure 11.1 The two basic scenarios

these ε units of time. Since $C_{\{A,B\}}sent(\mu)$ implies $(K_A K_B)^k sent(\mu)$ for every k, it follows that if $\varepsilon > 0$, then $C_{\{A,B\}}sent(\mu)$ is never attained (Exercise 11.2). This may not seem too striking when we think of ε that is relatively large, say a day, or an hour. The argument, however, is independent of the magnitude of ε, and remains true even for small values of ε. Even if Alice and Bob are guaranteed that the message μ either arrives instantaneously or within one nanosecond, they still never attain common knowledge that μ was sent!

Now let us consider what happens if, instead of sending μ, Alice sends at time m a message μ' that specifies the sending time, such as the following:

"This message is being sent at time m; μ."

Recall that it is common knowledge that every message sent by Alice is received by Bob within ε time units. Assume that in this case both Alice and Bob use the same (global) clock. When Bob receives μ', he knows that μ' was sent at time m. Moreover, Bob's receipt of μ' is guaranteed to happen no later than time $m + \varepsilon$. Since

Alice and Bob use the same clock, it is common knowledge at time $m + \varepsilon$ that it is $m + \varepsilon$. It is also common knowledge that any message sent at time m is received by time $m + \varepsilon$. Thus, at time $m + \varepsilon$, we obtain that the fact that Alice sent μ' to Bob is common knowledge.

Note that in the first example common knowledge will never hold regardless of whether ε is a day, an hour, or a nanosecond. The slight uncertainty about the sending time and the message transmission time prevents common knowledge of μ from ever being attained in this scenario. What makes the second example so dramatically different? Recall that $\models C_G\varphi \Rightarrow E_G C_G\varphi$, which implies that when a fact φ is common knowledge, everybody must know that it is. It is impossible for agent i to know that φ is common knowledge without agent j knowing it as well. This means that the transition from φ not being common knowledge to its being common knowledge must involve a *simultaneous* change in all relevant agents' knowledge. In the first example, the uncertainty makes such a simultaneous transition impossible, while in the second example this transition occurs at time $m + \varepsilon$. These two examples help illustrate the connection between simultaneity and common knowledge and the effect this can have on the attainability of common knowledge. We now formalize and further explore this connection.

11.2.2 Simultaneous Events

The Alice and Bob examples presented previously illustrate how the transition from a situation in which a fact is not common knowledge to one where it is common knowledge requires simultaneous events to take place at all sites of the system. The relationship between simultaneity and common knowledge, however, is even more fundamental than that. In Chapter 6, we saw that actions that must be performed simultaneously by all parties, such as attacking in the coordinated attack problem, become common knowledge as soon as they are performed. We used this to show that common knowledge is a prerequisite for simultaneous actions. We now prove that a fact's becoming common knowledge requires the occurrence of simultaneous events at different sites of the system. Moreover, we show that, in a certain technical sense, the occurrence of simultaneous events is necessarily common knowledge. This will demonstrate the strong link between common knowledge and simultaneous events.

To prove this claim, we need to formalize the notion of simultaneous events. We start with a few definitions, all relative to a fixed interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$. Let S denote the set of points of the system $\mathcal{R}$. In this chapter and in previous chapters we spoke informally about "events" in systems. In the spirit of Section 2.5, we now take an *event* in $\mathcal{R}$ to be a subset of S; intuitively, these are the points

where the event e holds. As before, an event e is said to *hold* at a point (r, m) if $(r, m) \in e$. Of special interest are events whose occurrence is reflected in an agent's local state. More formally, an event e is *local to* i (in interpreted system $\mathcal{I}$) if there is a set L_i^e of i's local states such that for all points (r, m) we have $(r, m) \in e$ iff $r_i(m) \in L_i^e$. The events $send(\mu, j, i)$, $receive(\mu, j, i)$, and $int(\mathsf{a}, i)$ of Section 4.4.5, which correspond respectively to agent i sending a message, receiving a message, and performing an internal action, are clearly local events for agent i. We remark that the definition of a local event does not imply that an event that is local to i cannot also be local to j. In order to be local to both agents, it only needs to be reflected in the local states of the agents.

Certain events depend only on the global state. An event e is a *state event* if there is a set $\mathcal{G}^e$ of global states such that for all points (r, m) we have $(r, m) \in e$ iff $r(m) \in \mathcal{G}^e$. It is easy to see that local events are state events. We associate with every state event e a primitive proposition ψ_e such that $\pi(r(m))(\psi_e) = \mathbf{true}$ iff $(r, m) \in e$. This is well-defined because it follows easily from the definition of state events that if (r, m) and (r', m') are points where $r(m) = r'(m')$, then $(r, m) \in e$ if and only if $(r', m') \in e$.

We can similarly associate with every formula φ an event

$$\mathtt{ev}_{\mathcal{I}}(\varphi) = \{(r, m) \mid (\mathcal{I}, r, m) \models \varphi\}.$$

(Note that $\mathtt{ev}_{\mathcal{I}}(\varphi)$ is defined like $\mathtt{ev}_M(\varphi)$ in Section 2.5, where M is a Kripke structure.) The event $\mathtt{ev}_{\mathcal{I}}(\varphi)$ thus holds exactly when φ holds. We call $\mathtt{ev}_{\mathcal{I}}(\varphi)$ *the event of* φ *holding (in* $\mathcal{I}$*)*. It is easy to check that a local event e holds if and only if the corresponding agent knows that e is holding; moreover, the event of $K_i\varphi$ holding is always a local event for i. More formally, we have the following result.

Lemma 11.2.1 *An event e is local to a process i in a system $\mathcal{I}$ if and only if $\mathcal{I} \models \psi_e \Rightarrow K_i\psi_e$. Similarly, $\mathcal{I} \models \psi \Rightarrow K_i\psi$ if and only if the event $\mathtt{ev}_{\mathcal{I}}(\psi)$ is local to i in $\mathcal{I}$.*

Proof See Exercise 11.3. ∎

We are now ready to address the issue of simultaneous events. Intuitively, two events are simultaneous if they occur at the same points. Our interest in simultaneity, however, is primarily in the context of coordination. Namely, we are interested in events that are local to different agents and are coordinated in time. Thus, we concentrate on events whose occurrence is simultaneously reflected in the local state of the agents. More formally, we define an *event ensemble for* G (or just *ensemble* for short) to be a mapping $\mathbf{e} : i \mapsto e_i$, assigning to every agent $i \in G$ an event e_i local

to i. In this chapter, we investigate certain types of ensembles of particular interest. An ensemble $\mathbf{e}$ for G is said to be *perfectly coordinated* if the local events in $\mathbf{e}$ hold simultaneously; formally, if $(r, m) \in \mathbf{e}(i)$ for some $i \in G$, then $(r, m) \in \mathbf{e}(j)$ for all $j \in G$. Thus, the ensemble $\mathbf{e}$ for G is perfectly coordinated precisely if $\mathbf{e}(i) = \mathbf{e}(j)$ for all $i, j \in G$. Since an event e_i that is local to agent i is defined in terms of a set $L_i^{e_i}$ of states local to agent i, the ensemble $\mathbf{e}$ for G is perfectly coordinated if all the agents in G enter their respective sets $L_i^{e_i}$ simultaneously. Thus, the events in a perfectly coordinated ensemble are simultaneous.

An example of a perfectly coordinated ensemble is the set of local events that correspond to the ticking of a global clock, if the ticking is guaranteed to be reflected simultaneously at all sites of a system. Another example is the event of shaking hands: being a mutual action, the handshakes of the parties are perfectly coordinated.

Given an ensemble $\mathbf{e}$ for G, the proposition $\psi_{\mathbf{e}(i)}$ corresponds to the state event $\mathbf{e}(i)$ holding. We also define $\psi_{\mathbf{e}} = \bigvee_{i \in G} \psi_{\mathbf{e}(i)}$. Thus, $\psi_{\mathbf{e}}$ is true whenever one of the state events $\mathbf{e}(i)$ holds. We can now show the following:

Proposition 11.2.2 *Let $\mathcal{I}$ be an interpreted system and G a set of agents.*

(a) For every formula φ, the ensemble $\mathbf{e}$ for G defined by $\mathbf{e}(i) = \mathrm{ev}_{\mathcal{I}}(K_i C_G \varphi)$ is perfectly coordinated.

(b) If $\mathbf{e}$ is a perfectly coordinated ensemble for G, then for every $i \in G$ we have $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow K_i C_G \psi_{\mathbf{e}(i)}$.

(c) If $\mathbf{e}$ is a perfectly coordinated ensemble for G, then $\mathcal{I} \models \psi_{\mathbf{e}} \Rightarrow C_G \psi_{\mathbf{e}}$.

Proof For part (a), choose $i, j \in G$. The fact that

$$\models C_G \varphi \Leftrightarrow K_i C_G \varphi \quad \text{and} \quad \models C_G \varphi \Leftrightarrow K_j C_G \varphi$$

implies that

$$\mathrm{ev}_{\mathcal{I}}(C_G \varphi) = \mathrm{ev}_{\mathcal{I}}(K_i C_G \varphi) = \mathrm{ev}_{\mathcal{I}}(K_j C_G \varphi),$$

which are local events for i and j, respectively. It follows that if $\mathbf{e}$ is the ensemble for G defined by $\mathbf{e}(i) = \mathrm{ev}_{\mathcal{I}}(K_i C_G \varphi)$, then $\mathbf{e}$ is perfectly coordinated.

For part (b), recall that $\mathbf{e}(i)$ is a local event for every $i \in G$. We thus have by Lemma 11.2.1 that $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow K_i \psi_{\mathbf{e}(i)}$ holds for every $i \in G$. Since $\mathbf{e}$ is perfectly coordinated, we have that $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Leftrightarrow \psi_{\mathbf{e}(j)}$ holds for every $i, j \in G$. It follows that $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow E_G \psi_{\mathbf{e}(i)}$, and by the Induction Rule we obtain that $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow C_G \psi_{\mathbf{e}(i)}$. From the Fixed-Point Axiom it follows that $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow K_i C_G \psi_{\mathbf{e}(i)}$.

Finally, part (c) follows from (b): Assume $(\mathcal{I}, r, m) \models \psi_{\mathbf{e}}$. By definition of $\psi_{\mathbf{e}}$, we have that $(\mathcal{I}, r, m) \models \psi_{\mathbf{e}(i)}$ for some $i \in G$. By part (b) we obtain that $(\mathcal{I}, r, m) \models K_i C_G \psi_{\mathbf{e}(i)}$. Since $\mathcal{I} \models K_i C_G \psi_{\mathbf{e}(i)} \Rightarrow C_G \psi_{\mathbf{e}(i)}$ and $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow \psi_{\mathbf{e}}$, it thus follows that $(\mathcal{I}, r, m) \models C_G \psi_{\mathbf{e}}$ and we are done. ∎

Proposition 11.2.2 precisely captures the close correspondence between common knowledge and simultaneous events. It asserts that the local events that correspond to common knowledge are perfectly coordinated, and the local events in a perfectly coordinated ensemble are common knowledge when they hold. Notice that part (a) implies in particular that the transitions from $\neg K_i C_G \varphi$ to $K_i C_G \varphi$, for $i \in G$, must be simultaneous. Among other things, this helps clarify the difference between the two examples considered in Section 11.2.1: In the first example, Alice and Bob cannot attain common knowledge of $sent(\mu)$ because they are unable to make such a simultaneous transition, while in the second example they can (and do).

In systems where the global state records the actions performed in the latest round (such as in recording contexts, where the environment's state records the joint actions), the description of a run determines what actions are performed by the various agents and when the actions are performed. In such systems, we can talk about *simultaneous actions*; these are actions that are performed simultaneously by all members of a group whenever they are performed. Simultaneous coordinated attack, simultaneous Byzantine agreement, and the act of shaking hands are all examples of simultaneous actions. Given a system in which actions are recorded in this fashion, we can identify, for every agent i and action a, the *event of i being about to perform* a with the set of points (r, m) of the system where i performs a in round $m + 1$. Since we view actions as being performed based on a program or protocol the agent is following, the fact that i is about to perform an action is typically reflected in i's local state. When this is the case, the event of i being about to perform a is local to i. This happens, for example, when the agent is following a deterministic protocol. Proposition 11.2.2(b) immediately implies that whenever actions are guaranteed to be performed simultaneously and the event of being about to perform them is guaranteed to be local to the agents involved, these agents have common knowledge that the actions are about to be performed. This generalizes Proposition 6.1.2 and Theorem 6.4.2, where this is proved formally for coordinated attack and SBA. Moreover, it captures the sense in which common knowledge is a prerequisite for simultaneous action.

The close relationship between common knowledge and simultaneous actions is what makes common knowledge such a useful tool for analyzing tasks involving coordination and agreement. It also gives us some insight into how common knowledge arises. For example, the fact that a public announcement has been made is

common knowledge, since the announcement is heard simultaneously by everyone. (Strictly speaking, of course, this is not quite true; we return to this issue in Section 11.4.) More generally, simultaneity is inherent in the notion of *copresence*. As a consequence, when people sit around a table, the existence of the table, as well as the nature of the objects on the table, are common knowledge.

Proposition 11.2.2 formally captures the role of simultaneous actions in making agreements and conventions common knowledge. Recall that we argued in Chapter 6 that common knowledge is inherent in agreements and conventions. Hand shaking, face-to-face or telephone conversation, and a simultaneous signing of a contract are standard ways of reaching agreements. They all involve simultaneous actions and have the effect of making the agreement common knowledge.

11.3 Temporal Imprecision

As we illustrated previously and formalized in Proposition 11.2.2, simultaneity is inherent in the notion of common knowledge (and vice versa). It follows that simultaneity is a prerequisite for attaining common knowledge. Alice and Bob's failure to reach common knowledge in the first example above can therefore be blamed on their inability to perform a simultaneous state transition. As might be expected, the fact that simultaneity is a prerequisite for attaining common knowledge has additional consequences. For example, in many distributed systems each process possesses a clock. In practice, in any distributed system there is always some uncertainty regarding the relative synchrony of the clocks and regarding the precise message transmission times. This results in what is called the *temporal imprecision* of the system. The amount of temporal imprecision in different systems varies, but it can be argued that every practical system will have some (possibly very small) degree of imprecision. Formally, a given system $\mathcal{R}$ is said to have *temporal imprecision* if for all runs $r \in \mathcal{R}$, times m, and sets G of processes with $|G| \geq 2$, there exist processes $i, j \in G$ with $i \neq j$, a run $r' \in \mathcal{R}$, and a time m' such that $r'_i(m') = r_i(m)$ while $r'_j(m') = r_j(m+1)$. Intuitively, in a system with temporal imprecision, i is uncertain about j's clock reading; at the point (r, m), process i cannot tell whether j's clock is characterized by j's local state at (r, m) or j's local state at $(r, m+1)$. Techniques from the distributed-systems literature can be used to show that any system in which, roughly speaking, there is some initial uncertainty regarding relative clock readings and uncertainty regarding exact message transmission times must have temporal imprecision (provided we model time in sufficiently small units).

Systems with temporal imprecision turn out to have the property that no protocol can guarantee to synchronize the processes' clocks perfectly. As we now show,

events cannot be perfectly coordinated in systems with temporal imprecision either. These two facts are closely related.

We define an ensemble $\mathbf{e}$ for G in $\mathcal{I}$ to be *nontrivial* if there exist a run r in $\mathcal{I}$ and times m, m' such that $(r, m) \in \cup_{i \in G} \mathbf{e}(i)$ while $(r, m') \notin \cup_{i \in G} \mathbf{e}(i)$. Thus, if $\mathbf{e}$ is a perfectly coordinated ensemble for G, it is *trivial* if for each run r of the system and for each agent $i \in G$, the events in $\mathbf{e}(i)$ hold either at all points of r or at no point of r. The definition of systems with temporal imprecision implies the following:

Proposition 11.3.1 *In a system with temporal imprecision there are no nontrivial perfectly coordinated ensembles for G, if $|G| \geq 2$.*

Proof Suppose, by way of contradiction, that $\mathbf{e}$ is a nontrivial perfectly coordinated ensemble for G in a system $\mathcal{I}$ with temporal imprecision, and $|G| \geq 2$. Because $\mathbf{e}$ is nontrivial for G, there must exist a run r in $\mathcal{I}$ and times m, m' such that $(r, m) \in \cup_{i \in G} \mathbf{e}(i)$ while $(r, m') \notin \cup_{i \in G} \mathbf{e}(i)$. Suppose that $m' > m$. (The case where $m > m'$ is similar.) Then there must be a time k such that $(r, k) \in \cup_{i \in G} \mathbf{e}(i)$ while $(r, k+1) \notin \cup_{i \in G} \mathbf{e}(i)$. From the definition of temporal imprecision, it follows that there are agents $j, j' \in G$ and a point (r', k') such that $r_j(k) = r'_j(k')$ and $r_{j'}(k+1) = r'_{j'}(k')$. Since $\mathbf{e}$ is an ensemble for G, for each $i \in G$ there is a set L_i of states local to agent i such that $\mathbf{e}(i)$ holds precisely when i is in some state of L_i. Because $\mathbf{e}$ is perfectly coordinated, the sets $\mathbf{e}(i)$ are the same for all $i \in G$. Since $(r, k) \in \cup_{i \in G} \mathbf{e}(i)$ and each set $\mathbf{e}(i)$ is the same, we must have $(r, k) \in \mathbf{e}(j)$. It follows that $r_j(k) = r'_j(k') \in L_j$. Thus, $(r', k') \in \mathbf{e}(j) = \mathbf{e}(j')$. But this means that $r'_{j'}(k') = r_{j'}(k+1) \in L_{j'}$, and so $(r, k+1) \in \mathbf{e}(j')$. But this contradicts the assumption that $(r, k+1) \notin \cup_{i \in G} \mathbf{e}(i)$. ∎

We thus have the following corollary.

Corollary 11.3.2 *Let $\mathcal{I}$ be a system with temporal imprecision, let φ be a formula, and let $|G| \geq 2$. Then for all runs r and times m we have $(\mathcal{I}, r, m) \models C_G\varphi$ iff $(\mathcal{I}, r, 0) \models C_G\varphi$.*

Proof Proposition 11.2.2 says that the ensemble $\mathbf{e}$ for G defined by $\mathbf{e}(i) = \mathsf{ev}_{\mathcal{I}}(K_i C_G \varphi)$ is perfectly coordinated. By Proposition 11.3.1, this perfectly coordinated ensemble is trivial, and so $(r, m) \in \cup_{i \in G} \mathbf{e}(i)$ iff $(r, 0) \in \cup_{i \in G} \mathbf{e}(i)$. The corollary follows. ∎

In simple terms, Corollary 11.3.2 states that no fact can become common knowledge during a run of a system with temporal imprecision. This sharpens Theorem 4.5.4 of Chapter 4, where we showed that common knowledge cannot be gained in a.m.p. systems. If the units by which time is measured in our model are sufficiently small, then all practical distributed systems have temporal imprecision. As a

result, Corollary 11.3.2 implies that no fact can ever become common knowledge in practical distributed systems. Carrying this argument even further, we can view essentially all real-world scenarios as situations in which true simultaneity cannot be guaranteed. For example, the children in the muddy children puzzle neither hear nor comprehend the father simultaneously. There is bound to be some uncertainty about how long it takes each of them to process the information. Thus, according to our earlier discussion, the children in fact do not attain common knowledge of the father's statement.

We now seem to have a paradox. On the one hand, we have argued that common knowledge is unattainable in practical contexts. On the other hand, given our claim that common knowledge is a prerequisite for agreements and conventions and the observation that we do reach agreements and conventions are maintained, it seems that common knowledge *is* attained in practice.

Where is the catch? How can we explain this discrepancy between our practical experience and our technical results? In the remainder of this chapter, we present two resolutions to this paradox. The first rests on the observation that if we model time at a sufficiently coarse level, we can and do attain common knowledge. The question then becomes when and whether it is appropriate to model time in this way. The second says that, although we indeed cannot attain common knowledge, we can attain closely related variants of it, and this suffices for our purposes. In the next section, we explore the first approach.

11.4 The Granularity of Time

Given the complexity of the real world, any mathematical model of a situation must abstract away many details. A useful model is typically one that abstracts away as much of the irrelevant detail as possible, leaving all and only the relevant aspects of a situation. When modeling a particular situation, it can often be quite difficult to decide the level of granularity at which to model time. As we already observed in Section 10.4.2, the notion of time in a run rarely corresponds to real time. Rather, our choice of the granularity of time is motivated by convenience of modeling. Thus, in a distributed application, it may be perfectly appropriate to take a round to be sufficiently long for a process to send a message to all other processes, and perhaps do some local computation as well.

The argument that every practical system has some degree of temporal imprecision holds only relative to a sufficiently fine-grained model of time. For Proposition 11.3.1 and Corollary 11.3.2 to apply, time must be represented in sufficiently fine detail for temporal imprecision to be reflected in the model.

If a model has a coarse notion of time, then simultaneity, and hence common knowledge, are often attainable. In synchronous systems, for example, there is no temporal imprecision. As a result, in our simplified model of the muddy children puzzle, the children do attain common knowledge of the father's statement. As we argued above, however, if we "enhance" the model to take into consideration the minute details of the neural activity in the children's brains, and considered time on, say, a millisecond scale, the children would not be modeled as hearing the father simultaneously. Moreover, the children would not attain common knowledge of the father's statement. We conclude that whether a given fact becomes common knowledge at a certain point, or in fact whether it *ever* becomes common knowledge, depends in a crucial way on the model being used. While common knowledge may be attainable in a certain model of a given real world situation, it becomes unattainable once we consider a more detailed model of *the same situation*.

We are thus led to the question of when we are justified in reasoning and acting as if common knowledge is attainable. This reduces to the question of when we can argue that one model—in our case a coarser or less detailed model—is "as good" as another, finer, model. The answer, of course, is "it depends on the intended application." We now present one way of making this precise.

Let (γ_c, π_c) and (γ_f, π_f) be interpreted contexts. We think of the former as the coarse context and the latter as the fine context. Intuitively, (γ_f, π_f) is a more accurate model of reality than (γ_c, π_c). For example, in the muddy children puzzle, (γ_c, π_c) could be the interpreted context in which the children hear the father simultaneously, while (γ_f, π_f) could be the interpreted context where the children hear the father at very close but somewhat different times.

Suppose we are interested in designing a program that meets a certain specification σ. Initially, we may feel that the coarse context is a reasonable model of "reality," so we start, as advocated in Chapter 7, by designing a knowledge-based program Pg_{kb} and proving that it indeed satisfies σ in (γ_c, π_c). Our ultimate goal is to design a standard program, one that can be directly executed. Suppose we can indeed find a standard program Pg_s that implements Pg_{kb} in (γ_c, π_c). Recall that this means that the protocol Pg_s^{π} coincides with the protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$, for $\mathcal{I} = \mathbf{I}^{rep}(\mathsf{Pg}_s, \gamma, \pi)$. Of course, it follows that Pg_s also satisfies the specification σ in the coarse context (γ_c, π_c).

Now suppose that (γ_f, π_f) is a more accurate model of reality than (γ_c, π_c). Notice that Pg_{kb} may no longer satisfy σ in (γ_f, π_f). For example, the knowledge-based program for SBA given in Section 7.4 essentially says that process i should decide as soon as it knows either that it is common knowledge that 0 was an initial value or that it is common knowledge that 1 was an initial value. Clearly, if common

knowledge is not attainable in the fine context, then the process i will never make a decision in that context, and thus the knowledge-based program will not attain SBA. (This is less surprising than it might seem; in a model in which simultaneity cannot be guaranteed, SBA cannot be attained.) Nevertheless, we might still often want to argue that thinking in terms of a coarse context (γ_c, π_c) is good enough. This would be the case if the standard program Pg_s (which implements Pg_{kb} in the coarse context) also satisfies the specification σ in the fine context. More formally, assume that we have a coarse context (γ_c, π_c) and a standard program Pg_s satisfying the specification σ in this context. We say that the coarse context is an *adequate model* for the fine context (γ_f, π_f) with respect to Pg_s and σ, if Pg_s satisfies σ in the fine context as well. Intuitively, if the coarse context is an adequate model for the fine context, then it is safe to think of the agent following the standard program in the fine context as behaving as if the world is described by the coarse context (at least as far as satisfying σ is concerned).

This definition, involving as it does programs and specifications, may seem far removed from our concerns regarding how humans feel that they do attain common knowledge, and somehow manage to reach agreements and carry out conventions. The gap is not that large. Suppose an agent uses a coarse context to model a situation and develops a plan based on his knowledge for carrying out a course of action. We can think of such a plan as a knowledge-based program. Implementing the plan amounts to constructing a standard program that implements the knowledge-based program. Now if the fine context really is a better model of reality than the coarse context, then we end up with a situation where the standard program is, roughly speaking, executed in the fine context. If our plan is to have its intended effect (as captured by the specification), then it had better be the case that the coarse context is an adequate model for the fine context with respect to the given specification and the standard program implementing the plan.

Example 11.4.1 Recall that in Example 7.2.5 we described an interpreted context (γ^{mc}, π^{mc}) for the muddy children puzzle. In this context, the children all heard the father's initial statement and later questions simultaneously. We can think of this as a coarse context where, indeed, the children attain common knowledge. Suppose instead we assume that every time the father speaks, it takes somewhere between 8 and 10 milliseconds for each child to hear and process what the father says, but the exact time may be different for each child, and may even be different for a given child every time the father speaks. Similarly, after a given child speaks, it takes between 8 and 10 milliseconds for the other children and the father to hear and process what he says. (While there is nothing particularly significant in our choice of 8 and 10

milliseconds, it is important that a child does not hear any other child's response to the father's question before he utters his own response.) The father does not ask his k^{th} question until he has received the responses from all children to his $(k-1)^{\text{st}}$ question.

We now sketch a fine interpreted context $(\gamma_f^{mc}, \pi_f^{mc})$ that captures the scenario described previously. Just as with the context γ^{mc}, we take γ_f^{mc} to be a recording message-passing context. Again, the set $\mathcal{G}_0$ of initial states consists of the 2^n tuples of the form $(X, X^{-1}, \ldots, X^{-n})$, where $X = (x_1, \ldots, x_n)$ is a tuple of 0's and 1's. The father follows the same program as described in Example 7.2.5. The main difference between γ^{mc} and γ_f^{mc} is in the environment's protocol. Just like γ^{mc}, the environment γ_f^{mc} records all messages sent by the children and the father. It then nondeterministically decides when the message will be delivered, with the constraint that it must be delivered within 8 to 10 rounds. (We are implicitly assuming that a round corresponds to a millisecond here. Of course, it is unreasonable to assume that the father's action of sending a long message is performed in one round; more realistically, it should be performed over an interval of rounds. We ignore this issue here.) This completes the description of γ_f^{mc}. The interpretation function π_f^{mc} acts just like π^{mc}: the proposition *childheard*$_i$, which appears in MC_i is true precisely at those states where child i has heard a question from the father and has not answered it yet, and p_i is true if child i's forehead is muddy (which is encoded in the environment's state).

In (γ^{mc}, π^{mc}), the program MC satisfies the specification σ^{mc}: A child says "Yes" if he knows whether he is muddy and says "No" otherwise. Moreover, it is implemented by the standard program MC_s, where a child that sees k muddy children responds "No" to the father's first k questions and "Yes" to all the questions after that. Now consider what happens when we run MC_s in $(\gamma_f^{mc}, \pi_f^{mc})$. Let r' be a run in $\mathcal{I}' = \mathbf{I}^{rep}(\mathsf{MC}_s, \gamma_f^{mc}, \pi_f^{mc})$ and let r be the run with the same initial state as r' in $\mathcal{I} = \mathbf{I}^{rep}(\mathsf{MC}_s, \gamma^{mc}, \pi^{mc})$. For each child i, we can divide run r' into *phases* from i's point of view. Phase 1 starts when the father sends his initial message and ends when child i hears it. Phase 2 starts on the following round when child i responds to the father's initial question and ends when child i has heard all the children's responses to the father's initial question. In general, phase $2k+1$ starts the round after phase $2k$ ends, and ends when child i has heard the father's $(k+1)^{\text{st}}$ question; phase $2k+2$ starts on the following round when child i responds to the father's k^{th} question, and ends when child i has heard all the children's responses to the father's $(k+1)^{\text{st}}$ question. Note that phase k for child i and phase k for child j need not coincide; nevertheless, the phases stay fairly "close" to each other (see Exercise 11.4). It is easy to show that child i receives the same messages in phase k of run r' as he does

in round k of run r (see Exercise 11.4). Roughly speaking, the situation at the end of phase k in run r' looks the same to child i as the end of round k does in run r. More precisely, suppose phase k for child i ends at round k_i. It is not hard to show that if φ is either p_i or $\neg p_i$, then $(\mathcal{I}, r, k) \models K_i\varphi$ if and only if $(\mathcal{I}', r', k_i) \models K_i\varphi$ (Exercise 11.4). (In fact, this is true for any formula φ determined by the initial state, as defined in Section 6.5.) As a consequence, MC_s satisfies σ^{mc} in the interpreted context $(\gamma_f^{mc}, \pi_f^{mc})$.

We have just shown that (γ^{mc}, π^{mc}) is an adequate model for $(\gamma_f^{mc}, \pi_f^{mc})$ with respect to MC_s and σ^{mc}. Interestingly, the system $\mathbf{R}^{rep}(\mathsf{MC}_s, \gamma_f^{mc})$ is one with temporal imprecision (see Exercise 11.5), so the results of Proposition 11.3.1 and Corollary 11.3.2 apply: the children cannot attain common knowledge in the fine context. ∎

Example 11.4.2 Suppose we modify the interpreted context (γ^{cr}, π^{sba}) that we considered in Section 6.5 for SBA in the crash failure mode along the same lines as in the previous example. That is, rather than assuming that all messages from nonfaulty processes are received simultaneously, we assume that they take somewhere between 8 and 10 milliseconds to arrive. Let $(\gamma_f^{cr}, \pi_f^{sba})$ be the corresponding fine context. In Section 7.4, we described a knowledge-based program SBA that satisfied σ^{sba} in (γ^{cr}, π^{sba}), and sketched how to find a standard program that implemented SBA in this context. Will this standard program still satisfy σ^{sba} in the fine context $(\gamma_f^{cr}, \pi_f^{sba})$? The answer is no. One of the requirements in σ^{sba} is that the processes must decide simultaneously. Just as in the previous example, there is temporal imprecision in the fine context, so it follows from Proposition 11.3.1 that we cannot have such perfect coordination. On the other hand, all the requirements of σ^{sba} other than simultaneity are satisfied in the fine context. Moreover, if we divide a run into phases just as we did in the previous example, then we can show that all processes do decide in the same phase, even though they may not decide at the same time. Indeed, if the uncertainty in message delivery time is small, we can show that the processes decide within a very small window. In practice, "within a small window" is often tantamount to "simultaneous." Indeed, the window may be so small that the processes cannot distinguish decisions taken within such a small window from decisions taken simultaneously. ∎

These examples show that the question of when a coarse context is an adequate model is a delicate one, which is highly dependent on the specification. For example, a synchronous coarse context will typically not be an adequate model if the specification requires simultaneous actions and the fine context does not allow for

simultaneity. On the other hand, the discussion at the end of Example 11.4.2 shows that at times simultaneity may be too strong a requirement, and something very close to it, which is attainable in the fine context, may be more appropriate. We return to this point in Section 11.6.1, in the process of considering our second solution to the paradox.

11.5 Common Knowledge as a Fixed Point

11.5.1 Fixed Points

In Section 11.6, we consider another approach to the paradox of common knowledge. We intend to argue that even when common knowledge is not attainable, certain approximations of common knowledge are attainable and can be substituted for common knowledge in practical situations. To prepare for this development, we study here an alternative (but equivalent) characterization of common knowledge, which is more directly related to the way common knowledge comes about and the way it is used. This characterization is based on the notion of *fixed point*.

Recall that we argued in the muddy children puzzle that once the father makes his statement, this statement becomes common knowledge. The reason why the statement is common knowledge can be intuitively described as follows: After the father speaks, the children are in a situation that can be characterized by the fact that they have all heard the father's statement, and all know that they are in the situation. Denoting the situation by s, and the statement by φ, we might say that $s \Leftrightarrow E_G(\varphi \wedge s)$ is valid in the corresponding interpreted system. Indeed, in all cases that we have seen thus far, common knowledge arises precisely because a situation can be characterized in this way. This suggests that $C_G\varphi$ can be viewed as a solution of an "equation" of the form $x = E_G(\varphi \wedge x)$.

To make sense out of this equation, we need a way of viewing $E_G(\varphi \wedge x)$ as a function which takes an argument x. We do this by adding a *propositional variable* x to the language and providing a technique for associating with each formula in the extended language a function from sets of points in an interpreted system to sets of points. The idea is that the function corresponding to $E_G(\varphi \wedge x)$ is given as "input" the set of points where x is true, and it returns as "output" the set of points where $E_G(\varphi \wedge x)$ is true.

For the remainder of this discussion, fix an interpreted system $\mathcal{I}$. All the functions defined below take as arguments subsets of points in $\mathcal{I}$. As a first step towards our goal, we associate a function with each logical (Boolean and modal) operator. Let A

and B be sets of points. Given a function f and a subset A, we take $f^0(A) = A$ and $f^{l+1}(A) = f(f^l(A))$. Define

$$\begin{aligned} f_{\neg}(A) &= \overline{A} \quad \text{(the complement of } A\text{)} \\ f_{\wedge}(A, B) &= A \cap B \\ f_{K_i}(A) &= \{(r, m) : (r', m') \in A \text{ whenever } r_i'(m') = r_i(m)\} \\ f_{E_G}(A) &= \bigcap_{i \in G} f_{K_i}(A) \\ f_{C_G}(A) &= \bigcap_{l>0} f_{E_G}^l(A). \end{aligned}$$

These definitions are intended to capture the meaning of these operators. Intuitively, for example, $f_{K_i}(A)$ consists of all points where agent i "knows" A; that is, all points (r, m) such that all points (r', m') that i cannot distinguish from (r, m) are in A. To make this intuition precise, recall that we can associate with a formula φ its intension $\varphi^{\mathcal{I}}$ (in the language of Section 2.5);

$$\varphi^{\mathcal{I}} = \{(r, m) \mid (\mathcal{I}, r, m) \models \varphi\}.$$

(Note that $\varphi^{\mathcal{I}}$ also coincides in the system $\mathcal{I}$ with the event $\mathrm{ev}_{\mathcal{I}}(\varphi)$ of φ holding.) Notice that $f_{\neg}(\varphi^{\mathcal{I}}) = (\neg\varphi)^{\mathcal{I}}$; thus, the function $f_{\neg}$ maps the intension of φ to the intension of its negation. Similarly, $f_{K_i}(\varphi^{\mathcal{I}}) = (K_i\varphi)^{\mathcal{I}}$. More generally, we have the following result.

Lemma 11.5.1 *For every formula $\varphi \in \mathcal{L}_n^C$, we have that $(\neg\varphi)^{\mathcal{I}} = f_{\neg}(\varphi^{\mathcal{I}})$, $(\varphi \wedge \psi)^{\mathcal{I}} = f_{\wedge}(\varphi^{\mathcal{I}}, \psi^{\mathcal{I}})$, $(K_i\varphi)^{\mathcal{I}} = f_{K_i}(\varphi^{\mathcal{I}})$, $(E_G\varphi)^{\mathcal{I}} = f_{E_G}(\varphi^{\mathcal{I}})$, and $(C_G\varphi)^{\mathcal{I}} = f_{C_G}(\varphi^{\mathcal{I}})$.*

Proof See Exercise 11.6. ∎

We extend $\mathcal{L}_n^C$ by adding a propositional variable x. Syntactically, this variable is treated just like a primitive proposition, so that, for example, $K_i(p \wedge x)$ is a well-formed formula. Once we add this propositional variable to the language, we can consider formulas in the extended language as functions as follows. We associate with the variable x the identity function. Thus, for all sets A of points in $\mathcal{I}$

$$f_x(A) = A.$$

We now use the functions we defined above for the logical operators to associate a function with every formula of the extended language. We define $f_\varphi(A)$ for every

set A by induction on the structure of φ as follows:

$$\begin{aligned}
f_p(A) &= p^{\mathcal{I}} \text{ for a primitive proposition } p \\
f_{\neg\varphi}(A) &= f_{\neg}(f_\varphi(A)) \\
f_{\varphi\wedge\psi}(A) &= f_{\wedge}(f_\varphi(A), f_\psi(A)) \\
f_{K_i\varphi}(A) &= f_{K_i}(f_\varphi(A)) \\
f_{E_G\varphi}(A) &= f_{E_G}(f_\varphi(A)) \\
f_{C_G\varphi}(A) &= f_{C_G}(f_\varphi(A)).
\end{aligned}$$

Notice that, for a formula φ that does not contain the variable x, the function $f_\varphi(A)$ does not depend on the set A in any way. It follows that, for formulas $\varphi \in \mathcal{L}_n^C$, the functions f_φ are constant functions. As the next lemma shows, the function f_φ is a constant function, always returning the intension of φ.

Lemma 11.5.2 *For every formula $\varphi \in \mathcal{L}_n^C$ and set A of points of $\mathcal{I}$, we have that $f_\varphi(A) = \varphi^{\mathcal{I}}$.*

Proof See Exercise 11.7. ∎

Among other things, we can view the associated functions f_φ as a reformulation of the semantics of formulas in $\mathcal{L}_n^C$. In particular, we could have defined the satisfaction relation '$\models$' by having $(\mathcal{I}, r, m) \models \varphi$ iff $(r, m) \in f_\varphi(\emptyset)$. It follows from Lemma 11.5.2 that this definition is equivalent to our earlier one.

Of course, the function f_x is not a constant function. Indeed, the function corresponding to a formula that mentions x will typically be nonconstant. To understand the motivation for defining f_x to be the identity function, consider the formula $\varphi \wedge x$. Notice that $f_{\varphi\wedge x}(\psi^{\mathcal{I}}) = (\varphi \wedge \psi)^{\mathcal{I}}$. Similarly, it can be shown that $f_{E_G(\varphi\wedge x)}(\psi^{\mathcal{I}}) = (E_G(\varphi \wedge \psi))^{\mathcal{I}}$. More generally, let $\varphi[x/\psi]$ be the result of replacing all occurrences of x in φ by ψ, so that, for example, $E_G(\varphi \wedge x)[x/\psi]$ is $E_G(\varphi \wedge \psi)$.

Lemma 11.5.3 $f_\varphi(\psi^{\mathcal{I}}) = (\varphi[x/\psi])^{\mathcal{I}}$.

Proof We proceed by induction on the structure of φ. If φ is the propositional variable x, then $\varphi[x/\psi]$ is just the formula ψ. Since $f_x(\psi^{\mathcal{I}}) = \psi^{\mathcal{I}}$, the result follows immediately in this case. If φ is the primitive proposition p, then $p[x/\psi] = p$ and $f_p(\psi^{\mathcal{I}}) = p^{\mathcal{I}}$, and the result again follows. We consider one more case here. Assume that φ is of the form $K_i\varphi'$. Using the inductive hypothesis and Lemma 11.5.1, we have $f_{K_i\varphi'}(\psi^{\mathcal{I}}) = f_{K_i}(f_{\varphi'}(\psi^{\mathcal{I}})) = f_{K_i}((\varphi'[x/\psi])^{\mathcal{I}}) = (K_i(\varphi'[x/\psi]))^{\mathcal{I}} =$

$((K_i\varphi')[x/\psi])^{\mathcal{I}}$. The remaining cases, which are similar in spirit, are left to the reader. ∎

With this extension, we can formalize our earlier statement that $C_G\varphi$ is a solution to an equation of the form $x = E_G(\varphi \wedge x)$. What we really want to say is that if we apply the function $f_{E_G(\varphi \wedge x)}$ to $(C_G\varphi)^{\mathcal{I}}$, then the result is $(C_G\varphi)^{\mathcal{I}}$. To see this, recall that the Fixed-Point Axiom tells us that

$$\mathcal{I} \models C_G\varphi \Leftrightarrow E_G(\varphi \wedge C_G\varphi)$$

or, equivalently, that

$$(C_G\varphi)^{\mathcal{I}} = (E_G(\varphi \wedge C_G\varphi))^{\mathcal{I}}.$$

From Lemma 11.5.3, it now follows that

$$f_{E_G(\varphi \wedge x)}((C_G\varphi)^{\mathcal{I}}) = (C_G\varphi)^{\mathcal{I}}.$$

This last observation can be captured succinctly by saying that $(C_G\varphi)^{\mathcal{I}}$ is a *fixed point* of the function $f_{E_G(\varphi \wedge x)}$. The notion of fixed point turns out to be crucial to understanding common knowledge. Fortunately, there is a well developed theory of fixed points; we now briefly review the basic definitions of that theory.

Fix a set S (for us, the set S will be the points in the interpreted system $\mathcal{I}$). Consider a function f that maps subsets of S to subsets of S. A set $A \subseteq S$ is said to be a *fixed point* of f if $f(A) = A$. A *greatest* fixed point of f is a set B such that (1) B is a fixed point of f (i.e., $f(B) = B$), and (2) if A is a fixed point of f, then $A \subseteq B$. Similarly, a *least* fixed point of f is a fixed point C of f such that if A is a fixed point of f, then $C \subseteq A$. We denote the greatest fixed point of f by $gfp(f)$ and its least fixed point by $lfp(f)$. It follows that if f has a greatest fixed point, then

$$gfp(f) = \bigcup\{A \mid f(A) = A\}.$$

We now have all of the machinery necessary to state and formally prove that $C_G\varphi$ is the greatest fixed point of $E_G(\varphi \wedge x)$. (More accurately, we prove that $(C_G\varphi)^{\mathcal{I}}$ is the greatest fixed point of $f_{E_G(\varphi \wedge x)}$, but we continue to abuse notation here and below and identify a formula with both its associated function and its intension; the intended meaning should be clear from context.) Before we do so, however, we would like to add an explicit greatest fixed-point operator νx to the language. Thus, if ψ is a formula satisfying a certain restriction that we describe below, then so is $\nu x[\psi]$,, which is read "the greatest fixed point of ψ." We ascribe semantics to such

formulas using our association of formulas with functions. As a first step, we extend our association of formulas with functions by defining

$$f_{\nu x[\psi]}(A) = gfp(f_\psi).$$

Notice that $f_{\nu x[\psi]}$ is again a constant function, independent of A.

There is one problem with our definition of $f_{\nu x[\psi]}$: it is not always well defined. Some functions do not have greatest fixed points. Indeed, it is easy to see that $f_{\neg x}$ does not have a fixed point at all, let alone a greatest fixed point (Exercise 11.8). We therefore need to restrict the ψ's that can occur in the scope of νx so that f_ψ is guaranteed to have a fixed point. Conveniently, there is a simple syntactic condition on ψ that ensures this. Before stating it, let us review a fundamental result from the theory of fixed points. The function f is said to be *monotone* if $f(A) \subseteq f(B)$ whenever $A \subseteq B$.

Lemma 11.5.4 *Every monotone function has a greatest (and a least) fixed point.*

We call a formula φ *monotone* if its associated function f_φ is monotone. As we shall see, many formulas of interest are monotone. Of course, the function $f_{\neg x}$ is not monotone, and has no fixed point. Indeed, negation is the only logical operator that causes problems; all the other operators in $\mathcal{L}_n^C$ can be shown to preserve monotonicity. Nevertheless, negation does not always cause problems. For example, the formula $\neg\neg x$ is monotone. Note that in the formula $\neg\neg x$, the variable x is in the scope of an even number of negations. As we now show, this is a special case of a more general phenomenon.

Lemma 11.5.5 *If every occurrence of x in ψ appears in the scope of an even number of negation symbols, then the function f_ψ is monotone.*

Proof See Exercise 11.9. ∎

We thus restrict $\nu x[\psi]$ so that it is a well-formed formula only if every occurrence of x in ψ appears in the scope of an even number of negation symbols. Let $\mathcal{L}_n^{C\nu}$ be the language that results from extending $\mathcal{L}_n^C$ by adding the propositional variable x and the greatest fixed-point operator νx, subject to the restriction above. For a formula φ in $\mathcal{L}_n^{C\nu}$, we now define

$$(\mathcal{I}, r, m) \models \varphi \text{ iff } (r, m) \in f_\varphi(\emptyset).$$

As we noted earlier, by Lemma 11.5.2, this definition of '$\models$' agrees with our earlier definition in Chapter 2 for formulas in $\mathcal{L}_n^C$.

What have we gained by extending the language? We would like to prove that $C_G\varphi$ is the greatest fixed point of $E_G(\varphi \wedge x)$. Recall that we have already shown that $C_G\varphi$ is a fixed point of $E_G(\varphi \wedge x)$. The formula $E_G(\varphi \wedge x)$ also has many other fixed points. For example, *false* is a fixed point (clearly the *least* fixed point since $false^{\mathcal{I}} = \emptyset$) of $E_G(\varphi \wedge x)$. Moreover, for every formula φ', the formula $C_G(\varphi \wedge \varphi')$ is also a fixed point of $E_G(\varphi \wedge x)$ (Exercise 11.10). We now show that common knowledge is the greatest fixed point of $E_G(\varphi \wedge x)$. We say that a formula φ in $\mathcal{L}_n^{C\nu}$ is *closed* if every occurrence of the propositional variable x in φ is in the scope of a greatest fixed-point operator νx. Just as with first-order logic, a formula is closed if there are no "free" occurrences of x.

Theorem 11.5.6 *For every closed formula $\varphi \in \mathcal{L}_n^{C\nu}$ and nonempty set G of processes,*

$$\models C_G\varphi \Leftrightarrow \nu x[E_G(\varphi \wedge x)].$$

Proof Let $\mathcal{I}$ be an arbitrary interpreted system, and let $\psi = E_G(\varphi \wedge x)$. We have already shown that $(C_G\varphi)^{\mathcal{I}}$ is a fixed point of f_ψ, so it only remains to show that $(C_G\varphi)^{\mathcal{I}}$ is the greatest fixed point of f_ψ. Let B be an arbitrary fixed point of the function f_ψ. We must have

$$B = f_\psi(B) = f_{E_G(\varphi \wedge x)}(B) = f_{E_G}(f_{\varphi \wedge x}(B)) = f_{E_G}(\varphi^{\mathcal{I}} \cap B).$$

We now prove that $B \subseteq (f_{E_G})^k(\varphi^{\mathcal{I}})$ for all $k > 0$. As a result, we then obtain that

$$B \subseteq \bigcap_{k>0} (f_{E_G})^k(\varphi^{\mathcal{I}}) = f_{C_G}(\varphi^{\mathcal{I}}) = (C_G\varphi)^{\mathcal{I}}.$$

Because this is true for any fixed point B of f_ψ, we conclude that $C_G\varphi$ is the greatest fixed point. We proceed by induction on k. First note that the definition of f_{K_i} implies that

$$f_{K_i}(A \cap A') = f_{K_i}(A) \cap f_{K_i}(A')$$

for all sets A and A'. The same is therefore also true of f_{E_G}. For the case $k = 1$, we have

$$B = f_{E_G}(\varphi^{\mathcal{I}} \cap B) = f_{E_G}(\varphi^{\mathcal{I}}) \cap f_{E_G}(B) \subseteq f_{E_G}(\varphi^{\mathcal{I}}) = (f_{E_G})^1(\varphi^{\mathcal{I}}).$$

Assume the claim holds for k, so that $B \subseteq (f_{E_G})^k(\varphi^{\mathcal{I}})$. This implies, by the monotonicity of f_{E_G}, that

$$f_{E_G}(B) \subseteq f_{E_G}((f_{E_G})^k(\varphi^{\mathcal{I}})) = (f_{E_G})^{k+1}(\varphi^{\mathcal{I}}).$$

As in the base case, we have

$$B = f_{E_G}(\varphi^{\mathcal{I}} \cap B) = f_{E_G}(\varphi^{\mathcal{I}}) \cap f_{E_G}(B) \subseteq f_{E_G}(B).$$

Combining $B \subseteq f_{E_G}(B)$ with $f_{E_G}(B) \subseteq (f_{E_G})^{k+1}(\varphi^{\mathcal{I}})$, we obtain that $B \subseteq (f_{E_G})^{k+1}(\varphi^{\mathcal{I}})$, and we are done. ∎

Theorem 11.5.6 provides us with an alternative definition of common knowledge, as the greatest fixed point of $E_G(\varphi \wedge x)$. While formally equivalent to the definition of common knowledge as an infinite conjunction, this fixed-point definition has an entirely different flavor to it. As discussed at the beginning of this section, the view of common knowledge as a fixed point closely corresponds to the way it seems to arise in practice. Arguably, the fixed-point definition is the more natural definition of common knowledge. Another advantage of the fixed-point definition of common knowledge, which we study in some detail in the next section, is the fact that slight modifications of this definition give rise to states of knowledge corresponding to useful forms of (non-simultaneous) coordination.

As we now show, viewing common knowledge as a fixed point helps explain two of the most important properties of common knowledge—the Fixed-Point Axiom and the Induction Rule. Indeed, these properties turn out to be instances of two more general properties of greatest fixed points that are stated in the following lemma. The first says essentially that $\nu x[\psi]$ is a fixed point of the formula ψ, while the second gives a generalized version of the induction rule for formulas of the form $\nu x[\psi]$.

Lemma 11.5.7 *Let $\mathcal{I}$ be an interpreted system and let ψ be a monotone formula. Then*

(a) $\models \psi[x/\nu x[\psi]] \Leftrightarrow \nu x[\psi]$; *and*

(b) if $\mathcal{I} \models \varphi \Rightarrow \psi[x/\varphi]$ *then* $\mathcal{I} \models \varphi \Rightarrow \nu x[\psi]$.

Proof See Exercise 11.14. ∎

If we replace ψ by $E_G(\varphi \wedge x)$ in Lemma 11.5.7(a), then by the fact that $C_G\varphi = \nu x[E_G(\varphi \wedge x)]$, which we have from Theorem 11.5.6, we get the Fixed-Point Axiom:

$$\models E_G(\varphi \wedge C_G\varphi) \Leftrightarrow C_G\varphi.$$

Similarly, if we replace ψ by $E_G(\psi \wedge x)$ in Lemma 11.5.7(b), we get the Induction Rule for common knowledge as a special case:

$$\text{From } \psi \Rightarrow E_G(\psi \wedge \varphi) \text{ infer } \varphi \Rightarrow C_G\psi.$$

We have shown how to extend the language $\mathcal{L}_n^C$ by adding a greatest fixed-point operator. Similar extensions can be performed for a large class of languages. Given a language $\mathcal{L}$ with a modal operator L, we can typically define a monotone function f_L corresponding to L so that an analogue to Lemma 11.5.1 holds, i.e., so that $f_L(\varphi^{\mathcal{I}}) = (L\varphi)^{\mathcal{I}}$. We can then add a greatest (and least) fixed-point operator to $\mathcal{L}$ as outlined above. Once we do this, analogous proofs show that Lemmas 11.5.2, 11.5.3, 11.5.5, and 11.5.7 hold with respect to $\mathcal{L}$. Thus, in Section 11.6, we use fixed points to define various approximations to common knowledge. The fact that proofs essentially identical to those given above can be used to show that analogues to the Fixed-Point Axiom and the Induction Rule hold for all the variants of common knowledge that we define in Section 11.6 suggests that viewing common knowledge in terms of the fixed-point definition can be quite useful.

11.5.2 Downward Continuity and Infinite Conjunctions

Theorem 11.5.6 proves that defining common knowledge as a fixed point is formally equivalent to our original definition of common knowledge as an infinite conjunction. It turns out that this equivalence is a special case of a more general phenomenon in the theory of fixed points. We study the relationship between fixed-point definitions and infinite conjunctions in the rest of this section.

A function f is *downward continuous* if

$$f\left(\bigcap_{l=0}^{\infty} A_l\right) = \bigcap_{l=0}^{\infty} f(A_l)$$

for all sequences $A_0, A_1, A_2, \ldots$ with $A_0 \supseteq A_1 \supseteq A_2 \supseteq \cdots$. It is easy to see that a downward continuous function must be monotone. For suppose that f is downward continuous and $A \subseteq B$. Then we have $B \supseteq A \supseteq A \supseteq A \supseteq \cdots$. From the definition of downward continuity, we get that $f(A) \cap f(B) = f(A \cap B) = f(A)$, from which it immediately follows that $f(A) \subseteq f(B)$, proving that f is monotone.

Since a downward continuous function is monotone, we know it has a greatest fixed point. Downward continuous functions are interesting because for them we can provide an explicit description of the greatest fixed point.

Lemma 11.5.8 *If f is a downward continuous function mapping subsets of S to subsets of S, then*

$$gfp(f) = \bigcap_{l=0}^{\infty} f^l(S).$$

Proof We first show that

$$f^0(S) \supseteq f^1(S) \supseteq f^2(S) \supseteq \cdots.$$

Clearly $f^0(S) \supseteq f^1(S)$, since $f^0(S) = S$. We now proceed by induction. Suppose that $f^l(S) \supseteq f^{l+1}(S)$. Since f is monotone, it follows that

$$f^{l+1}(S) = f\big(f^l(S)\big) \supseteq f\big(f^{l+1}S\big)\big) = f^{l+2}(S),$$

completing the inductive step of the argument.

Let $f^\infty(S) = \bigcap_{l=0}^{\infty} f^l(S)$. We next show that $f^\infty(S)$ is a fixed point of f. Because f is downward continuous, we know that

$$f(f^\infty(S)) = \bigcap_{l=0}^{\infty} f(f^l(S)) = \bigcap_{l=0}^{\infty} f^{l+1}(S) = f^\infty(S).$$

Thus, $f^\infty(S)$ is indeed a fixed point.

Finally, we must show that $f^\infty(S)$ is a greatest fixed point of f. Let T be any other fixed point of f. We show by induction on l that $T \subseteq f^l(S)$ for all l. Clearly $T \subseteq f^0(S)$, since $f^0(S) = S$. Suppose $T \subseteq f^l(S)$. Because f is monotone, $f(T) \subseteq f(f^l(S))$. But $f(T) = T$, since T is a fixed point of f. Thus, $T \subseteq f^{l+1}(S)$, completing the inductive step. It follows that

$$T \subseteq \bigcap_{l=0}^{\infty} f^l(S) = f^\infty(S). \quad \blacksquare$$

It is easy to check that $f_{E_G(\varphi \wedge x)}$ is downward continuous (Exercise 11.11). Since $true^{\mathcal{I}}$ consists of the set of all points in the interpreted system $\mathcal{I}$, it follows from Lemma 11.5.8 that we have

$$gfp(f_{E_G(\varphi \wedge x)}) = \bigcap_{l=0}^{\infty} f^l_{E_G(\varphi \wedge x)}(true^{\mathcal{I}}).$$

Define a sequence of formulas $\varphi_0, \varphi_1, \ldots$ inductively by taking φ_0 to be *true* and φ_{l+1} to be $E_G(\varphi \wedge \varphi_l)$, i.e., φ_{l+1} is $E_G(\varphi \wedge x)[x/\varphi_l]$. Thus, φ_1 is $E_G(\varphi \wedge true)$, φ_2 is $E_G(\varphi \wedge E_G(\varphi \wedge true))$, and so on. It follows immediately from Lemma 11.5.3 that $f^l_{E_G(\varphi \wedge x)}(true^{\mathcal{I}}) = \varphi_l^{\mathcal{I}}$. Therefore

$$\big(\nu x[E_G(\varphi \wedge x)]\big)^{\mathcal{I}} = gfp(f_{E_G(\varphi \wedge x)}) = \bigcap_{l=0}^{\infty} \varphi_l^{\mathcal{I}}.$$

Converting to the level of formulas, we have

$$\mathcal{I} \models \nu x[E_G(\varphi \wedge x)] \Leftrightarrow \bigwedge_{l=0}^{\infty} \varphi_l.$$

(Technically, the infinite conjunction is not a well-formed formula in $\mathcal{L}_n^{C\nu}$. Thus, more formally, we have shown $(\mathcal{I}, r, m) \models \nu x[E_G(\varphi \wedge x)]$ iff $(\mathcal{I}, r, m) \models \varphi_l$ for all $l \geq 0$.) Since E_G distributes over conjunctions (i.e., $E_G(\psi_1 \wedge \psi_2)$ is equivalent to $E_G\psi_1 \wedge E_G\psi_2$), and $E_G(true)$ is equivalent to *true*, it is easy to see that φ_2 is equivalent to $E_G\varphi \wedge E_G^2\varphi$, φ_3 is equivalent to $E_G\varphi \wedge E_G^2\varphi \wedge E_G^3\varphi$, etc. Thus $\bigwedge_{l=0}^{\infty} \varphi_l$ is equivalent to $\bigwedge_{l=1}^{\infty} E_G^l(\varphi)$. This gives an alternate proof of Theorem 11.5.6.

The arguments of the preceding paragraph generalize to arbitrary downward continuous functions. We formalize them in the following proposition.

Proposition 11.5.9 *Suppose f_ψ is downward continuous. Define $\psi_0, \psi_1, \ldots$ inductively by taking ψ_0 to be true and ψ_{l+1} to be $\psi[x/\psi_l]$. Then*

$$(\mathcal{I}, r, m) \models \nu x[\psi] \quad \textit{iff} \quad (\mathcal{I}, r, m) \models \psi_l \textit{ for all } l \geq 0.$$

Proof See Exercise 11.12. ∎

11.6 Approximations of Common Knowledge

Section 11.2 shows that common knowledge captures the state of knowledge resulting from simultaneous events. It also shows, however, that in the absence of events that are guaranteed to hold simultaneously, common knowledge is not attained. As we have argued, true simultaneity is irrelevant in most practical cases. In Section 11.4, we tried to answer the question of when we can reason and act as if certain events were simultaneous. But there is another point of view we can take. There are situations where events holding at different sites need not happen simultaneously; the level of coordination required is weaker than absolute simultaneity. For example, we may want the events to hold at most a certain amount of time apart. Alternatively, we may want them to hold at the same *local times*, in a situation where each process has an individual clock, and the clocks are closely (but not necessarily perfectly) synchronized. It turns out that just as common knowledge is the state of knowledge corresponding to perfect coordination, there are states of shared knowledge corresponding to other forms of coordination. We can view these states of knowledge as approximations of true common knowledge. Moreover, many of these variants of

common knowledge can be defined as fixed points in a manner similar to the fixed-point definition for common knowledge. Fortunately, while perfect coordination is hard to attain in practice, weaker forms of coordination are often attainable. This is one explanation as to why the unattainability of common knowledge might not spell as great a disaster as we might have originally expected. This section considers a few of these weaker forms of coordination, and their corresponding states of knowledge.

11.6.1 $\mathcal{E}$- and Eventual Common Knowledge

Let us return to the first Alice and Bob example from Section 11.2. Notice that if $\varepsilon = 0$, then Alice and Bob attain common knowledge of $sent(\mu)$ immediately after the message is sent. In this case, it is guaranteed that once the message is sent, both agents immediately know the contents of the message, as well as the fact that it has been sent. Intuitively, it seems that the closer ε is to 0, the closer Alice and Bob's state of knowledge should be to common knowledge. Let us try to compare the situation when $\varepsilon > 0$ with $\varepsilon = 0$. As we saw in Section 11.2, if $\varepsilon > 0$ then Alice does not know that Bob received her message immediately after she sends the message. She does, however, know that *within ε time units* Bob will receive the message and know both the contents of the message and that the message has been sent. The sending of the message results in a situation where, within ε time units, everyone knows that the situation holds. This is analogous to the fact that common knowledge corresponds to a situation where everyone knows that the situation holds. This suggests that the state of knowledge resulting in the Alice and Bob scenario should again involve a fixed point of some sort. We now formalize a notion of coordination related to the Alice and Bob example, and define an approximation of common knowledge corresponding to this type of coordination.

An ensemble $\mathbf{e}$ for G is said to be *ε-coordinated* (in a given system $\mathcal{I}$) if the local events in $\mathbf{e}$ never hold more than ε time units apart; formally, if $(r, m) \in \mathbf{e}(i)$ for some $i \in G$, then there exists an interval $I = [m', m' + \varepsilon]$ such that $m \in I$ and for all $j \in G$ there exists $m_j \in I$ for which $(r, m_j) \in \mathbf{e}(j)$. While it is essentially infeasible in practice to coordinate events so that they hold simultaneously at different sites of a distributed system, ε-coordination is often attainable in practice, even in systems where there is uncertainty in message delivery time. Moreover, when ε is sufficiently small, there are many applications for which ε-coordination is practically as good as perfect coordination. This may be the case for many instances of agreements and conventions. There is no more than an initial period of ε time units during which there may be an incompatibility between the partners to the agreement. (Of

course, it will not necessarily be common knowledge when this period is over; there is no way to get around the fact that ε-coordination, on its own, is incapable of bringing about common knowledge.) Note that an ε-coordination with $\varepsilon = 0$ is perfect coordination. One example of ε-coordination results from a message being broadcast to all members of a group G, with the guarantee that it will reach all of the members within ε time units of one another. In this case it is easy to see that when an agent receives the message, she knows the message has been broadcast, and knows that within ε time units each of the members of G will have received the message, and will know that within ε ...

Let ε be arbitrary. We say that *within an ε interval everyone in G knows φ*, denoted $E_G^\varepsilon\varphi$, if there is an interval of ε time units containing the current time such that each process comes to know φ at some point in this interval. Formally, we have $(\mathcal{I}, r, m) \models E_G^\varepsilon\varphi$ if there exists an interval $I = [m', m' + \varepsilon]$ such that $m \in I$ and for all $i \in G$ there exists $m_i \in I$ for which $(\mathcal{I}, r, m_i) \models K_i\varphi$. Thus, in the case of Alice and Bob we have $\mathcal{I} \models sent(\mu) \Rightarrow E_{\{A,B\}}^\varepsilon sent(\mu)$. We now define ε-common knowledge, denoted by C_G^ε, as follows, for a closed formula φ:

$$C_G^\varepsilon\varphi =_{\text{def}} \nu x[E_G^\varepsilon(\varphi \wedge x)].$$

Notice how similar this definition is to the fixed-point definition of common knowledge. The only change is in replacing E_G by E_G^ε. Since $f_{E_G^\varepsilon}$ is monotone (Exercise 11.15) and, if φ is a closed formula, every occurrence of x in $E_G^\varepsilon(\varphi \wedge x)$ is in the scope of an even number of negation signs, the formula $C_G^\varepsilon\varphi$ is well-defined for all closed φ. Furthermore, the arguments at the end of the previous section show that C_G^ε satisfies the obvious analogues of the Fixed-Point Axiom and Induction Rule, with E_G^ε replacing E_G. Although C_G satisfies all the S5 properties, this is not the case for C_G^ε; the only S5 property that it satisfies is the Positive Introspection Axiom (Exercise 11.16).

Just as common knowledge is closely related to perfect coordination, ε-common knowledge is related to ε-coordination. We now make this claim precise. A proof similar to that of Proposition 11.2.2 can be used to prove the following proposition.

Proposition 11.6.1 *Let $\mathcal{I}$ be an interpreted system and G a set of agents.*

(a) For every formula φ, the ensemble $\mathbf{e}$ for G defined by $\mathbf{e}(i) = \mathrm{ev}_{\mathcal{I}}(K_i C_G^\varepsilon\varphi)$ is ε-coordinated.

(b) If $\mathbf{e}$ is an ε-coordinated ensemble for G, then for every $i \in G$ we have $\mathcal{I} \models \psi_{\mathbf{e}(i)} \Rightarrow K_i C_G^\varepsilon \psi_{\mathbf{e}(i)}$.

(c) If $\mathbf{e}$ *is an* ε*-coordinated ensemble for* G*, then* $\mathcal{I} \models \psi_{\mathbf{e}} \Rightarrow C_G^{\varepsilon}\psi_{\mathbf{e}}$.

Proof See Exercise 11.18. ∎

Although ε-common knowledge is useful for the analysis of systems where the uncertainty in message communication time is small, it is not quite as useful in the analysis of a.m.p. systems, since, as we observed in Section 6.1, such systems display unbounded message delivery (umd). In a.m.p. systems, rather than perfect or ε-coordination, what can often be achieved is *eventual* coordination. An ensemble $\mathbf{e}$ for G is *eventually coordinated* (in a given system $\mathcal{I}$) if, for every run of the system, if some event in $\mathbf{e}$ holds during the run, then all events in $\mathbf{e}$ do. More formally, if $(r, m) \in \mathbf{e}(i)$ for some $i \in G$, then for all $j \in G$ there exists some m_j for which $(r, m_j) \in \mathbf{e}(j)$. An example of an eventual coordination of G consists of the delivery of (copies of) a message broadcast to every member of G in an a.m.p. system. An agent receiving this message knows the contents of the message, as well as the fact that each other member of G must receive the message at some point in time, either past, present, or future.

Eventual coordination gives rise to *eventual* common knowledge, denoted by $C_G^{\diamond}$, and defined by

$$C_G^{\diamond}\varphi =_{\text{def}} \nu x[E_G^{\diamond}(\varphi \wedge x)].$$

Here we define $E_G^{\diamond}\varphi$ to hold at $(\mathcal{I}, r, m)$ if for each $i \in G$ there is some time m_i such that $(\mathcal{I}, r, m_i) \models K_i\varphi$. Thus, $E_G^{\diamond}$ can be viewed as the limit of E_G^{ε} as ε approaches infinity. It is straightforward to show that $C_G^{\diamond}$ is related to eventual coordination just as C_G is related to simultaneous events, and C_G^{ε} to ε-coordination (Exercise 11.19). We make use of this connection in Section 11.6.2 below.

We conclude with some further comments about the properties of the modal operators C_G^{ε} and $C_G^{\diamond}$. It can be shown that E_G^{ε} is downward continuous (although it turns out this depends heavily on the fact that we are treating time as discrete rather than continuous; see Exercise 11.15). It follows from our discussion in Section 11.5 that C_G^{ε} is equivalent to an infinite conjunction; in fact, it immediately follows from that discussion that $C_G^{\varepsilon}\varphi$ is equivalent to

$$E_G^{\varepsilon}\varphi \wedge E_G^{\varepsilon}(\varphi \wedge E_G^{\varepsilon}\varphi) \wedge E_G^{\varepsilon}(\varphi \wedge E_G^{\varepsilon}\varphi \wedge E_G^{\varepsilon}(\varphi \wedge E_G^{\varepsilon}\varphi)) \wedge \ldots \qquad (*)$$

It is *not*, however, equivalent to the infinite conjunction

$$E_G^{\varepsilon}\varphi \wedge E_G^{\varepsilon}E_G^{\varepsilon}\varphi \wedge E_G^{\varepsilon}E_G^{\varepsilon}E_G^{\varepsilon}\varphi \wedge \ldots \qquad (**)$$

The reason that $C_G^{\varepsilon}\varphi$ is not equivalent to the conjunction in $(**)$ is that E_G^{ε}, unlike E_G, does not distribute over conjunction: $E_G^{\varepsilon}\varphi \wedge E_G^{\varepsilon}\psi$ does not necessarily

imply $E_G^{\varepsilon}(\varphi \wedge \psi)$. There is, however, one important special case where (**) is equivalent to $C_G^{\varepsilon}\varphi$, namely, in systems with perfect recall, where φ is a stable formula. These issues are dealt with in more detail in Exercise 11.17.

By way of contrast, $E_G^{\diamond}$ is *not* downward continuous; thus, $C_G^{\diamond}$ is not equivalent to $\bigwedge_{k=1}^{\infty}(E^{\diamond})^k\varphi$, even in systems with perfect recall where φ is a stable formula (see Exercise 11.19).

11.6.2 Applications to Coordinated Attack

Proposition 11.6.1 and the analogous Exercise 11.19(a) provide us with a way of characterizing the state of knowledge corresponding to ε- and eventual coordination. We can use this proposition (and the exercise) to decide whether ε-common knowledge and eventual common knowledge, and the corresponding types of coordination, can be attained in particular circumstances. We now apply this kind of reasoning to the analysis of weak forms of coordinated attack.

In Section 6.1 we showed that a simultaneous attack can never be coordinated in the absence of sufficiently strong guarantees on the reliability of communication. In particular, we showed that such an attack cannot be coordinated if communication displays unbounded message delivery (i.e., the system satisfies the condition umd). In fact, our discussion in Section 11.2 implies that a simultaneous attack cannot be coordinated if there is any uncertainty about message delivery. Suppose we weaken the simultaneity requirement. When is it possible to coordinate an attack if we require only that the two divisions attack within a certain ε-time bound of each other? How about if we require even less, namely, that if one of them ever attacks, then the other will eventually also attack? We call the first situation an *ε-coordinated attack* and the second an *eventually coordinated attack*. Can the generals achieve one of these weaker notions of coordinated attack?

A positive answer to that question is fairly clear. If messages are guaranteed to be delivered within ε units of time, then ε-coordinated attack can be accomplished. General A simply sends General B a message saying "*attack*" and attacks immediately; General B attacks upon receipt of the message. Similarly, if messages are guaranteed to be delivered eventually, then even if there is no bound on message delivery time, an eventually coordinated attack can be carried out.

Intuitively, these guarantees on message delivery are necessary conditions for the attainment of ε-coordinated or eventually coordinated attack. To make this precise, we first need to define the specifications more carefully. The three components of the the original specification σ^{ca} for coordinated attack in Chapter 6 required that the attacks of the generals be sufficiently coordinated, that an attack by both of the

generals be possible only if some communication has been successful, and finally that an attack does take place in at least one instance. These will be the components of the specifications of ε-coordinated attack and of eventually coordinated attack. Nevertheless, the fact that we no longer require the generals to attack in the same round makes the specifications of these problems differ slightly from σ^{ca}. In particular, *attack*—which denotes that the generals are both about to attack—may never hold in a system that satisfies ε-coordinated attack, since the generals may not attack simultaneously in any run. Thus, we want to modify the specification σ^{ca} so that it is not defined in terms of *attack*.

As in Exercise 6.4, we define *attacked* to be an abbreviation for $attacked_A \wedge attacked_B$. Thus, *attacked* is true once both generals have attacked. We define $\sigma^{\varepsilon ca}$, the specification of ε-coordinated attack, to consist of all ca-compatible interpreted systems $\mathcal{I}$ such that

1^{ε}. $\mathcal{I} \models (attacked_A \Rightarrow \bigcirc^{\varepsilon} attacked_B) \wedge (attacked_B \Rightarrow \bigcirc^{\varepsilon} attacked_A)$,

$2'$. $\mathcal{I} \models \neg delivered \Rightarrow \neg \bigcirc attacked$, and

$3'$. $(\mathcal{I}, r, m) \models attacked$ for at least one point (r, m) of $\mathcal{I}$.

The first condition says that the generals attack at most ε time units apart, the second says that least one message must be delivered before both generals can attack, and the third says that there is at least one run in which they do attack. Note that in a system where both generals attack simultaneously, the conditions $2'$ and $3'$ of $\sigma^{\varepsilon ca}$ are equivalent to conditions 2 and 3 of the specification σ^{ca} given in Chapter 6. Thus, we could have equally well defined σ^{ca} using $2'$ and $3'$.

The specification $\sigma^{\diamond ca}$ of eventually coordinated attack is defined analogously, except that we replace the first condition by:

$1^{\diamond}$. $\mathcal{I} \models (attacked_A \Rightarrow \diamond attacked_B) \wedge (attacked_B \Rightarrow \diamond attacked_A)$.

Notice that condition 1^{ε} implies $1^{\diamond}$, so that any system satisfying $\sigma^{\varepsilon ca}$ automatically satisfies $\sigma^{\diamond ca}$. As a result, if $\sigma^{\diamond ca}$ cannot be satisfied in a certain context, then neither can $\sigma^{\varepsilon ca}$. We will see an example of this later on.

Just as in the case of (perfectly) coordinated attack, an attack by both parties in these weaker problems still requires that at least one message be delivered. Since now, however, we no longer assume that the generals necessarily attack simultaneously, it is possible for one general to attack without a message having been delivered.

We now want to prove an analogue to Corollary 6.1.3, connecting ε-coordinated attack with ε-common knowledge, and eventual coordinated attack with eventual

common knowledge. Let $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$ be a ca-compatible interpreted system, where P is a deterministic protocol. Consider the event ensemble $\mathbf{e}_{attack}$, where for each $i \in \{A, B\}$, the event $\mathbf{e}_{attack}(i)$ consists of all those points in $\mathcal{I}$ where General i is attacking. (Notice that the fact that P is deterministic makes $\mathbf{e}_{attack}(i)$ be a local event, and hence $\mathbf{e}_{attack}$ is an event ensemble.) It is almost immediate from the definitions that $\mathbf{e}_{attack}$ is a perfectly coordinated ensemble if $\mathcal{I}$ satisfies σ^{ca}, an ε-coordinated ensemble if $\mathcal{I}$ satisfies $\sigma^{\varepsilon ca}$, and an eventually coordinated ensemble if $\mathcal{I}$ satisfies $\sigma^{\diamond ca}$. Let *attack'* be the formula $attacking_A \vee attacking_B$. Notice that *attack'* is the proposition $\psi_{\mathbf{e}_{attack}}$ corresponding to the event ensemble $\mathbf{e}_{attack}$ with respect to the group $\{A, B\}$. It immediately follows from part (c) of Propositions 11.2.2 and 11.6.1 and from Exercise 11.19 that $\mathcal{I} \models attack' \Rightarrow C(attack')$ if $\mathcal{I}$ satisfies σ^{ca}, $\mathcal{I} \models attack' \Rightarrow C^{\varepsilon}(attack')$ if $\mathcal{I}$ satisfies $\sigma^{\varepsilon ca}$, and $\mathcal{I} \models attack' \Rightarrow C^{\diamond}(attack')$ if $\mathcal{I}$ satisfies $\sigma^{\diamond ca}$. (As in Section 6.1, we omit the subscript $\{A, B\}$ on C, C^{ε}, and $C^{\diamond}$.) If $\mathcal{I}$ satisfies σ^{ca}, then *attack* is equivalent to *attack'*, since, according to σ^{ca}, if the generals attack, they must attack simultaneously. Thus, if $\mathcal{I}$ satisfies σ^{ca}, then $\mathcal{I} \models attack \Rightarrow C(attack)$, which is the content of Proposition 6.1.2. Once we move to ε-common knowledge and eventual common knowledge, we can no longer replace *attack* by *attack'*. There is no reason to expect that *attack* will ever hold in a system satisfying $\sigma^{\varepsilon ca}$, for example, since even in a run in which the generals attack there may be no point where both generals attack simultaneously.

Notice that to go from Proposition 6.1.2, which says that $attack \Rightarrow C(attack)$ is valid, to Corollary 6.1.3, which says that $attack \Rightarrow C(delivered)$ is valid, we used the fact that $attack \Rightarrow delivered$ is valid in systems satisfying σ^{ca}. It is not the case that $attack' \Rightarrow delivered$ is valid in systems satisfying $\sigma^{\varepsilon ca}$ or $\sigma^{\diamond ca}$, so we can no longer apply this argument. It is true, however, that $attack' \Rightarrow \bigcirc^{\varepsilon} delivered$ is valid in systems satisfying $\sigma^{\varepsilon ca}$; once one general attacks, the other will do so within time ε, and once they have both attacked, *delivered* must hold. Similarly, $attack' \Rightarrow \diamond delivered$ is valid in systems satisfying $\sigma^{\diamond ca}$. In fact, an argument essentially identical to the proof of Corollary 6.1.3 gives the following theorem.

Theorem 11.6.2 *Let $\mathcal{I} = \mathbf{I}^{rep}(P, \gamma, \pi)$, where (γ, π) is a ca-compatible interpreted context and P is a deterministic protocol. If $\mathcal{I}$ satisfies $\sigma^{\varepsilon ca}$ then $\mathcal{I} \models attack' \Rightarrow C^{\varepsilon}(\bigcirc^{\varepsilon} delivered)$. Similarly, if $\mathcal{I}$ satisfies $\sigma^{\diamond ca}$ then $\mathcal{I} \models attack' \Rightarrow C^{\diamond}(\diamond delivered)$.*

We now want to show that the guarantees on message delivery time discussed above are really necessary. To prove this formally, we use a condition that is somewhat stronger than umd. As in the case of umd, let $\mathcal{R}$ be a system such that, for an appropriately chosen π, the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is a message-delivery system. Again we write $d(r, m) = k$ if exactly k messages have been delivered in

the first m rounds of r. We say that such a system $\mathcal{R}$ *displays aml* (aml stands for *arbitrary message loss*) if for all points (r, m) in $\mathcal{R}$, there exists an agent i and a run $r' \in \mathcal{R}$ such that (1) for all $j \neq i$ and times $m' \leq m$ we have $r'_j(m') = r_j(m')$, (2) if $d(r, m) > 0$, then $d(r', m) < d(r, m)$; otherwise, $d(r', m) = d(r, m) = 0$, and (3) $d(r', m'') = d(r', m)$ for $m'' \geq m$. Notice that the first two clauses of aml are the same as in umd, except that we also allow the possibility that $d(r, m) = 0$. Thus, it is trivially the case that a system that displays aml displays umd. But aml also has the third clause, which is not part of the definition of umd. Because of this clause, the run r' is one where not only is one fewer message delivered at (r', m) than at (r, m) (if some messages were delivered by (r, m)), but no further messages are delivered after time m. Since the setting in which the coordinated-attack problem was originally presented is one in which messengers may fail to deliver messages from any arbitrary point on, it satisfies aml. As in the case of umd, we say that a context γ *displays aml* if all systems described by it display aml, that is, if $\mathbf{R}^{rep}(P, \gamma)$ displays aml for every protocol P that can be run in the context γ.

We can now prove a result analogous to Theorem 6.1.1, showing that even eventual common knowledge cannot be attained in systems that display aml.

Theorem 11.6.3 *Let $\mathcal{I} = (\mathcal{R}, \pi)$ be a message-delivery system such that $\mathcal{R}$ displays aml, and let G be a set of two or more agents. Then*

$$\mathcal{I} \models \neg C_G^{\diamond}(\diamond delivered).$$

Proof See Exercise 11.20. ∎

As in Section 6.1, an immediate corollary of Theorem 11.6.3 is

Corollary 11.6.4 *If (γ, π) is a ca-compatible interpreted context such that γ displays aml, then there is no deterministic protocol P that satisfies $\sigma^{\diamond ca}$ (and hence also none satisfying $\sigma^{\varepsilon ca}$) in (γ, π).*

Thus, sufficiently unreliable communication prevents even weak forms of coordination.

11.6.3 Timestamped Common Knowledge

The notion of perfect coordination (as well as the other notions of coordination that we have considered so far) is defined with respect to a notion of *global* time. Global time is an external notion; it is not part of the processes' local states. One could argue that the reason that perfect coordination and common knowledge are not attainable

in real systems is precisely because the agents do not have access to global time. Indeed, perfect coordination and common knowledge are attainable in synchronous systems, where agents essentially have access to global time.

Another possible answer to the common knowledge paradox is that in real-life situations we are interested not in coordination with respect to global time, but rather in coordination with respect to an "internal" notion of time, that is, a notion of time that is accessible to the agents. For example, there are many protocols that operate in phases. Each process may start and end phase k at a different time. Events are coordinated not with respect to real time, but with respect to the phases in which they hold. Simultaneity in this case can often be substituted by "occurrence in the same phase." More generally, in many systems processes each have their own clocks, which are typically kept reasonably close together by some sort of clock-synchronization algorithm. We are then not so much interested in simultaneity as in ensuring a strong form of consistency: namely, that certain actions are performed at the same local times. For example, in the coordinated-attack problem General A could send General B the message "*attack at 6 A.M.*" in an attempt to coordinate their attack to 6 A.M. on their respective watches. A different type of common knowledge is appropriate for analyzing this form of coordination.

To formalize these intuitions, we must assume that each agent has a local clock, whose reading is part of its local state. For simplicity, we assume the reading is a natural number. Moreover, we assume that with every change in the agent's local state, the time on its clock increases. Formally, we say that $\mathcal{R}$ is a *system with local clocks* if for each agent i there is a function *clock* from local states to natural numbers such that if $r_i(m+1) \neq r_i(m)$, then $clock(r_i(m+1)) > clock(r_i(m))$. Notice that since we make clock time a function of the local state, we are implicitly assuming that agents know the time on their clocks. If $\mathcal{I} = (\mathcal{R}, \pi)$ and $\mathcal{R}$ is a system with local clocks, then we say that $\mathcal{I}$ is an *interpreted system with local clocks*.

In systems with local clocks, we can define the notion of timestamped coordination as follows. Let T be a fixed time. We say that an ensemble $\mathbf{e}$ for G is *T-coordinated* (in a given system $\mathcal{I}$), if whenever the event $\mathbf{e}(i)$ holds at local time T for some $i \in G$, then $\mathbf{e}(j)$ holds at T for for all $j \in G$, i.e., for all $i, j \in G$, if $(r, m) \in \mathbf{e}(i)$ and $clock(r_i(m)) = T$, then $(r, m') \in \mathbf{e}(j)$ for all m' such that $clock(r_j(m')) = T$.

We are now ready to define timestamped common knowledge. We denote "*at time T on its clock, i knows φ*" by $K_i^T\varphi$. T is said to be the *timestamp* associated with this knowledge. Formally,

$(\mathcal{I}, r, m) \models K_i^T\varphi$ iff $(\mathcal{I}, r, m') \models K_i\varphi$ holds for all m' such that $clock(r_i(m')) = T$.

Notice that $(\mathcal{I}, r, m) \models K_i^T \varphi$ holds vacuously if i's clock never reads time T in run r. We define $E_G^T \varphi =_{\text{def}} \bigwedge_{i \in G} K_i^T \varphi$. $E_G^T \varphi$ thus corresponds to every process in G knowing φ at time T on its own clock. We now define *(time T) timestamped common knowledge*, denoted C_G^T, by

$$C_G^T \varphi =_{\text{def}} \nu x\big[E_G^T(\varphi \wedge x)\big].$$

As one might suspect, timestamped common knowledge corresponds precisely to timestamped coordination (Exercise 11.22).

The relationship between timestamped common knowledge and the other approximate notions of common knowledge depends on guarantees on the state of synchronization of the different clocks. Let $\mathcal{I}$ be an interpreted system with local clocks. We say that the clocks in $\mathcal{I}$ are *perfectly synchronized* if the processes' clocks always show identical times, i.e., for every point (r, m) of $\mathcal{I}$ and processes $i, j \in G$, we have that $clock(r_i(m)) = clock(r_j(m))$. We say that the clocks in $\mathcal{I}$ are *ε-synchronized* if the processes' clocks are always within ε time units from each other, i.e., for every point (r, m) of $\mathcal{I}$ and process $i \in G$ there exists an interval $I = [m', m' + \varepsilon]$ such that $m \in I$ and for all $j \in G$ there exists some $m_j \in I$ such that such that $clock(r_i(m)) = clock(r_j(m_j))$. Finally, we say that the clocks in $\mathcal{I}$ are *eventually synchronized* if the processes' clocks go through the same readings though perhaps at different times, i.e., for every point (r, m) of $\mathcal{I}$ and processes $i, j \in G$ there exists some m_j such that $clock(r_i(m)) = clock(r_j(m_j))$. Note that in all three cases (perfect synchronization, ε-synchronization, and eventual synchronization), the local clocks go through the same sequence of readings (up to repetition). This is important since we are concerned here with coordinating actions to specific local times. If General B's watch never reads exactly 6 A.M., the general is not able to perform the action "attack at 6 A.M."

It is not hard to check that the following holds:

Theorem 11.6.5 *For any interpreted system $\mathcal{I}$ with local clocks, group G, formula φ, and time T,*

(a) if the clocks in $\mathcal{I}$ are perfectly synchronized and $clock(r_i(m)) = T$ for some process i, then $(\mathcal{I}, r, m) \models C_G^T \varphi \Leftrightarrow C_G \varphi$.

(b) if the clocks in $\mathcal{I}$ are ε-synchronized and $clock(r_i(m)) = T$ for some process i, then $(\mathcal{I}, r, m) \models C_G^T \varphi \Rightarrow C_G^\varepsilon \varphi$.

(c) if the clocks in $\mathcal{I}$ are eventually synchronized and $clock(r_i(m)) = T$ for some process i, then $\mathcal{I} \models C_G^T \varphi \Rightarrow C_G^\diamond \varphi$.

Proof See Exercise 11.23. ∎

Note that only in part (a) of Theorem 11.6.5 do we get equivalence of timestamped common knowledge and standard common knowledge; in parts (b) and (c) we have implication in only one direction. A weak converse to parts (b) and (c) does hold. Suppose the agents are able to set their clocks to a commonly agreed upon time T when they come to know $C_G^\varepsilon\varphi$ (resp., $C_G^\diamond\varphi$). Then it is easy to see that whenever $C_G^\varepsilon\varphi$ (resp., $C_G^\diamond\varphi$) is attainable, so is $C_G^T\varphi$.

In summary, timestamped common knowledge can be viewed as an "internal" notion, since it can be defined purely in terms of processes' local states. By way of contrast, notions such as common knowledge and ε-common knowledgemake essential use of the notion of global state, which in turns makes use of global time (since a global state corresponds to a set of local events that hold at the same time), and hence can be viewed as "external" notions. As Theorem 11.6.5 shows, the precise relationship between the internal and the external notions depends on the relationship between global time and local time.

11.6.4 Other Approximations of Common Knowledge

The approximations of common knowledge that we considered thus far correspond to forms of coordination that are weaker than perfect coordination in a temporal sense. It is possible to weaken simultaneous coordination in a different way. In the coordinated attack problem, for example, the generals might be quite happy with a protocol that would guarantee a simultaneous attack with probability 0.999999999. This suggests that another approximation of common knowledge that may in some cases provide a suitable substitute for common knowledge is a probabilistic variant of common knowledge. To define such a notion, we would need to enrich our language and semantic models with appropriate mechanisms to account for and reason about probability. We could then define a notion of "probability $1-\varepsilon$ common knowledge," which essentially says that with probability $1-\varepsilon$, each agent knows that, with probability $1-\varepsilon$, each agent knows ... that with probability $1-\varepsilon$, the formula φ holds. Other variants of common knowledge can also be defined, and have been found useful in analyzing protocols. See the bibliographic notes for further discussion.

11.7 Discussion

One of the central themes of this chapter is an attempt to resolve the paradox of common knowledge: Although it can be shown to be a prerequisite for day-to-day

activities of coordination and agreement, it can also be shown to be unattainable in practice. The resolution of this paradox leads to a deeper understanding of the nature of common knowledge and simultaneity, and shows once again the importance of the modeling process. In particular, it brings out the importance of the granularity at which we model time, and stresses yet again the need to consider the applications for which these notions are being used.

The themes of modeling and the relation to applications are perhaps appropriate ones with which to close this book. We have considered a number of formal models of knowledge and a number of formal definitions of notions such as knowledge, common knowledge, and algorithmic knowledge in these models. Our original models and definitions were simple, perhaps overly so, but we were still able to show that they could help us understand and analyze systems, even ones that were quite complicated. Our understanding was aided by having a formal model of a system, in which the notion of knowledge is defined quite naturally.

Not surprisingly, our simple definitions do not capture all the subtleties of knowledge and common knowledge in the many settings that these notions are used. As we have seen, when particular problems arise in the application of one of our definitions, a deeper study of the situation typically gives us a semantic understanding of the relevant issues that suggests how these subtleties can be captured. There is usually more than one way to do this, and each of the ways adds some complication to our original definition. We tend to be somewhat reluctant to label any particular definition as providing the "right" way of capturing a notion in general. Our claim is that the right definition depends on the application and how it is being modeled. This is perhaps as it should be. The real world is complicated, and it is difficult to suggest a general prescription for the best way of modeling a situation and the right abstractions to use. The best advice one can offer is to make things as simple as possible, but no simpler.

Exercises

* **11.1** This exercise deals with common knowledge in finite-state systems.

(a) Let $l \geq 1$ be arbitrary. Design a two-agent scenario in which one of the agents has exactly l states, and for some fact φ we have $E^{2l-2}\varphi \wedge \neg C\varphi$. (For the case of $l = 1$, recall from Section 2.2 that $E^0\varphi$ is simply φ.)

(b) Prove that in an a.m.p. system, if some agent $a \in G$ has only l different local states, then $E_G^{l+1}\varphi \Leftrightarrow C_G\varphi$ for all facts φ.

11.2 Describe the interpreted system corresponding to the first Alice and Bob example in Section 11.2, and prove that at time $m_S + k\varepsilon$, the formula $(K_A K_B)^k sent(\mu)$ holds, while $(K_A K_B)^{k+1} sent(\mu)$ does not hold.

11.3 Prove Lemma 11.2.1.

11.4 Using the notation defined in the second paragraph of Example 11.4.1, prove the following:

(a) for each $k \geq 1$ there exists some $m_k \geq 1$ such that $m_k \leq k_i \leq m_k + 2$ for each child i. Thus, the k^{th} phases for all children end within an interval of three rounds.

(b) child i receives the same messages in phase k of run r' as in round k of the run r, (Hint: the standard program MC_s determines what messages are sent in each phase.)

(c) if φ is a formula of the form p_i or $\neg p_i$, then $(\mathcal{I}, r, k) \models K_i\varphi$ if and only if $(\mathcal{I}', r', k_i) \models K_i\varphi$. (Hint: prove, for all pairs of runs $\hat{r}, \hat{r}'$ of $\mathcal{I}$ and $\mathcal{I}'$, respectively, that have the same initial states, that child i considers the same set of initial states possible in the corresponding points $(\hat{r}, k)$ and $(\hat{r}', \hat{k}_i)$ of the two systems, where $\hat{k}_i$ is the round of $\hat{r}$ at which the k^{th} phase ends for child i.)

11.5 Prove that the system $\mathbf{R}^{rep}(\mathsf{MC}_s, \gamma_f^{mc})$ of Example 11.4.1 is a system with temporal imprecision.

11.6 Prove Lemma 11.5.1.

11.7 Prove Lemma 11.5.2.

11.8 Show that $f_{\neg x}$ does not have a fixed point.

* **11.9** Prove Lemma 11.5.5.

11.10 Prove that for every formula φ', the formula $C_G(\varphi \wedge \varphi')$ is a fixed point of $E_G(\varphi \wedge x)$.

11.11 Prove that $f_{E_G(\varphi \wedge x)}$ is downward continuous.

11.12 Prove Proposition 11.5.9.

11.13 Let $f(X)$ be a monotone function and let B be an arbitrary set. Prove that if $B \subseteq f(B)$ then $B \subseteq gfp(f)$.

11.14 Prove Lemma 11.5.7. (Hint: for part (a), first show that $f_\psi(gfp(f_\psi)) = gfp(f_\psi)$, and hence that $f_\psi(f_{\nu x[\psi]}(\emptyset)) = f_{\nu x[\psi]}(\emptyset)$; now use Lemma 11.5.3 to derive the claim. For (b), notice that $\mathcal{I} \models \varphi \Rightarrow \tau$ iff $\varphi^{\mathcal{I}} \subseteq \tau^{\mathcal{I}}$. Set $\tau = \psi[x/\varphi]$, and use Exercise 11.13 to prove the claim.)

* **11.15** In this exercise, we consider some properties of E^ε_G.

(a) Give a formal definition of $f_{E^\varepsilon_G}$ which satisfies an analogue of Lemma 11.5.1.

(b) Prove that $f_{E^\varepsilon_G}$ is a monotone function.

(c) Prove that $f_{E^\varepsilon_G}$ is downward continuous. (Hint: fix an interpreted system $\mathcal{I}$. Suppose that $A_0 \supseteq A_1 \supseteq A_2 \supseteq \cdots$ are subsets of points in $\mathcal{I}$. We must prove that $f_{E^\varepsilon_G}(\bigcap_{k=0}^{\infty}(A_k)) = \bigcap_{k=0}^{\infty} f_{E^\varepsilon_G}(A_k)$. $\subseteq$ is easy to show since $f_{E^\varepsilon_G}$ is monotone. To show $\supseteq$, suppose that $(r, m) \in \bigcap_{k=0}^{\infty} f_{E^\varepsilon_G}(A_k)$. Thus, for each k, there exists an interval I_k of length ε such that $m \in I_k$ and for each agent $j \in G$, there exists $m_j \in I_k$ such that $(r, m_j) \in f_{K_j}(A_k)$. Since we are dealing with discrete-time domains, there are only finitely many possible intervals of the form I_k. Thus, there must exist one interval I which satisfies the above properties for infinitely many (and hence all) k. Since I has finite length and we are dealing with finitely many points, for each agent j, there exists $m_j \in I$ such that $(r, m_j) \in f_{E^\varepsilon_G} K_j(A_k)$ for infinitely many (and hence for all) k.)

(d) The previous proof depended heavily on the fact that time ranges over the natural numbers, and hence is discrete. Give a natural definition for $f_{E^\varepsilon_G}$ in interpreted systems where time ranges over the (nonnegative) reals, and show that $f_{E^\varepsilon_G}$ is *not* downward continuous in this case.

11.16 Which of the axioms and inference rules that C_G satisfies (S5 + Fixed-Point Axiom + Induction Rule) does C^ε_G satisfy? Prove your claims.

* **11.17** In this exercise, we consider the interaction between E^ε_G and infinite conjunctions.

(a) Let $\varepsilon = 1$. Show that if $(\mathcal{I}, r, m + 2) \models C_G\varphi$, then $(\mathcal{I}, r, m + 1) \models C^\varepsilon_G\varphi$.

(b) Again, let $\varepsilon = 1$. Construct an interpreted system $\mathcal{I}$ and run r such that $(\mathcal{I}, r, 2) \models C_G p$ and $(\mathcal{I}, r, 0) \models (E^\varepsilon_G)^k p$ if $k \geq 2$, but $(\mathcal{I}, r, 0) \models \neg E^\varepsilon_G p$.

(c) Show that $C_G^\varepsilon \varphi$ is not equivalent to $\bigwedge_{k=1}^{\infty}(E_G^\varepsilon \varphi)$. (Hint: let ψ be the infinite conjunction. It suffices to show that $\psi \Rightarrow E_G^\varepsilon \psi$ is not valid. Use the construction of part (b) to help show this.)

(d) Let $\mathcal{I}$ be a system with perfect recall and let φ be a stable formula. Define $\bigcirc^\varepsilon \varphi$ to mean "ε time units from now." Prove

(i) $\mathcal{I} \models E_G^\varepsilon \varphi \Leftrightarrow \bigcirc^\varepsilon E_G \varphi$.

(ii) $\mathcal{I} \models C_G^\varepsilon \varphi \Leftrightarrow \bigwedge_{k=1}^{\infty}(\bigcirc^\varepsilon E_G)^k \varphi$.

Conclude that in systems with perfect recall where φ is a stable fact, $C_G^\varepsilon \varphi$ is equivalent to $\bigwedge_{k=1}^{\infty}(E_G^\varepsilon \varphi)$.

11.18 Prove Proposition 11.6.1.

* **11.19** In this exercise, we consider some properties of eventual common knowledge.

(a) State and prove an analogue of Proposition 11.6.1 for $C^\diamond$ and eventual coordination.

(b) Prove that $f_{C^\diamond}$ is not downward continuous.

(c) Prove that $C^\diamond \varphi$ is not equivalent to $\bigwedge_{k=1}^{\infty}(E^\diamond)^k \varphi$, even in systems with perfect recall where φ is a stable formula.

* **11.20** Prove Theorem 11.6.3.

11.21 Prove Corollary 11.6.4.

11.22 Prove an analogue of Proposition 11.2.2 for timestamped common knowledge and timestamped coordination.

11.23 Prove Theorem 11.6.5.

Notes

Most of the material in this chapter is based on [Halpern and Moses 1990]. Theorem 11.1.1 and Exercise 11.1 were stated and proved by Fischer and Immerman [1986]. Interestingly, while the definition of common knowledge as an infinite conjunction, which we started with in Chapter 2, is the one most frequently used in the literature, the original definition of common knowledge by Lewis [1969] was in terms of a fixed point, or a self-referential pair of statements. Clark and Marshall [1981] consider the role of common knowledge in natural language communication. They prove that common knowledge is essential for correct parsing of sentences involving direct reference, and emphasize the role of copresence as a cause of common knowledge. Barwise [1988] discusses three definitions of common knowledge, as an infinite conjunction, a fixed point, and an instance of copresence. His work suggests an interpretation of knowledge for which the three definitions are not equivalent. The relationship between common knowledge and simultaneous events is pointed out and formalized in [Dwork and Moses 1990], [Halpern and Moses 1990], and [Moses and Tuttle 1988]. The fact that common knowledge corresponds to an event that is local to all relevant processes was independently observed by Geanakoplos [1992].

A precise statement and proof of the result that in systems where there is uncertainty regarding message transmission times there is always some temporal imprecision were given by Dolev, Halpern, and Strong [1986] and by Halpern, Megiddo, and Munshi [1985]. Further discussion regarding simultaneity and granularity of time is given by Fischer and Immerman [1986] and by Neiger [1988]. The approach advocated in Section 11.4, according to which we think of common knowledge as being attained at a coarse level of abstraction and will be safe so long as this level of abstraction can be implemented at lower levels of abstraction, seems to have first been proposed by Kurki-Suonio [1986]. Neiger and Toueg [1993] present an example of a coarse context, in which clocks are perfectly synchronized, with a powerful *publication* action, which is guaranteed to make a message common knowledge. This context is relatively easy to design programs for. They then describe a finer context with publications, in which clocks are not perfectly synchronized. In that context publications guarantee timestamped common knowledge. They show that the coarse context is an adequate model for the fine context with respect to all programs and a large class of specifications.

Recall that Example 11.4.1 considers a fine context for modeling the muddy children puzzle, and shows that the coarse synchronous context that we have been considering throughout the book is an adequate model for it. Moses, Dolev, and

Halpern [1986] consider the cheating husbands puzzle, a variant of the muddy children puzzle, but focus on contexts that are not synchronous: messages are not necessarily delivered in the same round that they are sent. They show that in these contexts, the outcome (i.e., whether the children eventually come to know the state of their own foreheads, and which of them do) can differ substantially from that in the synchronous context. Not surprisingly, the standard programs that implement the (analogue of the) knowledge-based program MC in these contexts may also differ substantially from MC_s. Among other things, this shows that, in contrast to the context $(\gamma_f^{mc}, \pi_f^{mc})$ of Example 11.4.1, such asynchronous contexts are not adequately modeled by (γ^{mc}, π^{mc}).

The fixed-point semantics introduced here is a simplified version of the framework of [Halpern and Moses 1990], and the earlier and more general framework of [Kozen 1983]. The general logical theory of fixed points dates back to [Tarski 1955]. In particular, the fact that a monotone modal operator is guaranteed to have a greatest (and least) fixed point is an immediate corollary of the Knaster-Tarski Theorem. (See [Tarski 1955] for a historical discussion of this theorem.)

We remark that a characterization similar to that of Lemma 11.5.8 can be given for greatest fixed points of arbitrary (not necessarily downward continuous) functions, but this involves the additional mathematical sophistication of an inductive construction that proceeds through all the *ordinal numbers*. A characterization similar to that of Lemma 11.5.8 holds for least fixed points of *upward continuous* functions (i.e., functions f such that $f(\bigcup_{l=0}^{\infty} A_l) = \bigcup_{l=0}^{\infty} f(A_l)$ for increasing sequences $A_0 \subseteq A_1 \subseteq \cdots$).

All variants of common knowledge introduced in this chapter are due to Halpern and Moses [1990]. As discussed in Section 11.6.4, other variants of common knowledge, including probabilistic approximations, have also been found to be useful. In particular, notions of probabilistic common knowledge have been considered by Brandenburger and Dekel [1987], Fagin and Halpern [1994], Halpern and Tuttle [1993], Krasucki, Parikh and Ndjatou [1990], and Monderer and Samet [1989]. Such notions have been used to analyze probabilistic Byzantine agreement by Halpern and Tuttle [1993] and probabilistic coordination by Monderer and Samet [1989] and by Rubinstein [1989].

Koo and Toueg [1988] prove a generalization of Corollary 11.6.4. Roughly speaking, they define a run-based specification to be *nontrivial* if it is not satisfied by runs in which no message is delivered. They prove that, in a message-passing context in which message delivery is asynchronous and unreliable but *fair* (so that any message sent repeatedly is eventually delivered), no program solving a nontrivial specification can guarantee that all participants will be able to reach a terminating

state. This result has far-reaching implications in the design of communication protocols in current-day computer networks. For a formulation of this result in the terminology of Theorem 11.6.3, see the overview by Moses [1992].

Panangaden and Taylor [1992] define the notion of *concurrent common knowledge*, which is a variant of common knowledge that is achievable in reliable asynchronous systems. They discuss the extent to which it can be used in such systems as a substitute for common knowledge. A variant of common knowledge called *continual common knowledge* is defined by Halpern, Moses, and Waarts [1990] and used to analyze and find round-optimal algorithms for the *eventual* Byzantine agreement problem, which is defined just like simultaneous Byzantine agreement except that processes are not required to decide simultaneously (see Exercise 6.39). Continual common knowledge is the one variant of common knowledge that is typically strictly stronger than common knowledge, while the other variants of common knowledge discussed in the literature can be viewed as approximations of common knowledge.

Bibliography

Afrati, F., C. H. Papadimitriou, and G. Papageorgiou (1988). The synthesis of communication protocols. *Algorithmica 3*(3), 451–472.

Aho, A. V., J. D. Ullman, A. D. Wyner, and M. Yannakakis (1982). Bounds on the size and transmission rate of communication protocols. *Computers and Mathematics with Applications 8*(3), 205–214. This is a later version of [Aho, Ullman, and Yannakakis 1979].

Aho, A. V., J. D. Ullman, and M. Yannakakis (1979). Modeling communication protocols by automata. In *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 267–273.

Anderlini, L. (1989). Computability and communication in common interest games. mimeo, Cambridge University.

Anderson, A. and N. D. Belnap (1975). *Entailment: The Logic of Relevance and Necessity*. Princeton, N.J.: Princeton University Press.

Aumann, R. J. (1976). Agreeing to disagree. *Annals of Statistics 4*(6), 1236–1239.

Bacharach, M. (1985). Some extensions of a claim of Aumann in an axiomatic model of knowledge. *Journal of Economic Theory 37*, 167–190.

Barcan, R. C. (1946). A functional calculus of first order based on strict implication. *Journal of Symbolic Logic 11*, 1–16.

Barcan, R. C. (1947). The identity of individuals in a strict functional calculus of second order. *Journal of Symbolic Logic 12*, 12–15.

Bartlett, K. A., R. A. Scantlebury, and P. T. Wilkinson (1969). A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM 12*, 260–261.

Barwise, J. (1981). Scenes and other situations. *Journal of Philosophy 78*(7), 369–397.

Barwise, J. (1988). Three views of common knowledge. In M. Y. Vardi (Ed.), *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 365–379. San Francisco, Calif.: Morgan Kaufmann.

Barwise, J. and J. Perry (1983). *Situations and Attitudes*. Cambridge, Mass.: Bradford Books.

Bayart, A. (1958). La correction de la logique modale du premier et second ordre S5. *Logique et Analyse 1*, 28–44.

Bazzi, R. and G. Neiger (1992). The complexity and impossibility of achieving fault-tolerant coordination. In *Proc. 11th ACM Symp. on Principles of Distributed Computing*, pp. 203–214.

Belnap, N. D. (1977). A useful four-valued logic. In G. Epstein and J. M. Dunn (Eds.), *Modern Uses of Multiple-Valued Logic*, pp. 5–37. Dordrecht, Netherlands: Reidel.

Benthem, J. F. A. K. van (1974). Some correspondence results in modal logic. Report 74–05, University of Amsterdam.

Benthem, J. F. A. K. van (1983). *Modal Logic and Classical Logic*. Naples: Bibliopolis.

Berman, P., J. Garay, and K. J. Perry (1989). Towards optimal distributed consensus. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pp. 410–415.

Białynicki-Birula, A. and H. Rasiowa (1957). On the representation of quasi-Boolean algebras. *Bull. de l'académie polonaise des sciences 5*, 259–261.

Binmore, K. (1987). Modeling rational players I. *Economics and philosophy 3*, 179–214. Part II appeared ibid., *4*, 9–55.

Binmore, K. (1990). *Essays on the Foundations of Game Theory*. Oxford, U.K.: Basil Blackwell.

Binmore, K. and H. S. Shin (1993). Algorithmic knowledge and game theory. In C. Bicchieri and M. L. D. Chiara (Eds.), *Knowledge, Belief, and Strategic Interaction*, Chapter 9. Cambridge, U.K.: Cambridge University Press.

Bochmann, G. V. and J. Gecsei (1977). A unified method for the specification and verification of protocols. In B. Gilchrist (Ed.), *Information Processing 77*, pp. 229–234. Amsterdam: North-Holland.

Boolos, G. (1979). *The Unprovability of Consistency*. Cambridge: Cambridge University Press.

Brafman, R., J.-C. Latombe, Y. Moses, and Y. Shoham (1994). Knowledge as a tool in motion planning under uncertainty. In R. Fagin (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pp. 208–224. San Francisco, Calif.: Morgan Kaufmann.

Brandenburger, A. and E. Dekel (1987). Common knowledge with probability 1. *Journal of Mathematical Economics 16*, 237–245.

Canning, D. (1992). Rationality, computability and Nash equilibrium. *Econometrica 60*, 877–888.

Carnap, R. (1946). Modalities and quantification. *Journal of Symbolic Logic 11*, 33–64.

Carnap, R. (1947). *Meaning and Necessity*. Chicago: University of Chicago Press.

Carver, M. (1989). Aces and eights. *Discover* (February), 88.

Cave, J. (1983). Learning to agree. *Economics Letters 12*, 147–152.

Chandy, K. M. and J. Misra (1986). How processes learn. *Distributed Computing 1*(1), 40–52.

Chandy, K. M. and J. Misra (1988). *Parallel Program Design: A Foundation*. Reading, Mass.: Addison-Wesley.

Chang, C. C. and H. J. Keisler (1990). *Model Theory* (3rd ed.). Amsterdam: North-Holland.

Chellas, B. F. (1980). *Modal Logic*. Cambridge, U.K.: Cambridge University Press.

Clark, H. H. and C. R. Marshall (1981). Definite reference and mutual knowledge. In A. K. Joshi, B. L. Webber, and I. A. Sag (Eds.), *Elements of discourse understanding*. Cambridge, U.K.: Cambridge University Press.

Clarke, E. M., E. A. Emerson, and A. P. Sistla (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems 8*(2), 244–263. An early version appeared in *Proc. 10th ACM Symposium on Principles of Programming Languages*, 1983.

Conway, J. H., M. S. Paterson, and U. S. S. R. Moscow (1977). A headache-causing problem. In J. K. Lenstra et al. (Eds.), *Een pak met een korte broek: Papers presented to H. W. Lenstra on the occasion of the publication of his "Euclidische Getallenlichamen"*. Private publication.

Cook, S. A. (1971). The complexity of theorem proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pp. 151–158.

Cresswell, M. J. (1970). Classical intensional logics. *Theoria 36*, 347–372.

Cresswell, M. J. (1972). Intensional logics and logical truth. *Journal of Philosophical Logic 1*, 2–15.

Cresswell, M. J. (1973). *Logics and Languages*. London: Methuen and Co.

DeMillo, R. A., N. A. Lynch, and M. J. Merritt (1982). Cryptographic protocols. In *Proc. 14th ACM Symp. on Theory of Computing*, pp. 383–400.

Diffie, W. and M. E. Hellman (1976). New directions in cryptography. *IEEE Transactions on Information Theory 22*(5), 644–654.

Dolev, D., J. Y. Halpern, and H. R. Strong (1986). On the possibility and impossibility of achieving clock synchronization. *Journal of Computer and System Sciences 32*(2), 230–250.

Dolev, D., R. Reischuk, and H. R. Strong (1990). Early stopping in Byzantine agreement. *Journal of the ACM 34*(7), 720–741.

Dolev, D. and H. R. Strong (1982). Polynomial algorithms for multiple processor agreement. In *Proc. 14th ACM Symp. on Theory of Computing*, pp. 401–407.

Dunn, J. M. (1976). Intuitive semantics for first-degree entailment and "coupled trees". *Philosophical Studies 29*, 149–168.

Dunn, J. M. (1986). Relevance logic and entailment. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic, Vol. III*, pp. 117–224. Dordrecht, Netherlands: Reidel.

Dwork, C. and Y. Moses (1990). Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation 88*(2), 156–186.

Eberle, R. A. (1974). A logic of believing, knowing and inferring. *Synthese 26*, 356–382.

Elgot-Drapkin, J. J. (1991). Step-logic and the three-wise-men problem. In *Proc. National Conference on Artificial Intelligence (AAAI '91)*, pp. 412–417.

Elgot-Drapkin, J. J. and D. Perlis (1990). Reasoning situation in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence 2*(1), 75–98.

Emde Boas, P. van, J. Groenendijk, and M. Stokhof (1980). The Conway paradox: its solution in an epistemic framework. In *Proc. 3rd Amsterdam Montague Symposium*, pp. 87–111. Math. Centrum, Amsterdam. Reprinted by Foris Publications, 1984.

Emerson, E. A. and J. Y. Halpern (1985). Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences 30*(1), 1–24.

Emerson, E. A. and J. Y. Halpern (1986). "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM 33*(1), 151–178.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. New York: Academic Press.

Fagin, R. and J. Y. Halpern (1988a). Belief, awareness, and limited reasoning. *Artificial Intelligence 34*, 39–76.

Fagin, R. and J. Y. Halpern (1988b). I'm OK if you're OK: on the notion of trusting communication. *Journal of Philosophical Logic 17*(4), 329–354. Reprinted in: R. H. Thomason (Ed.) (1989), *Philosophical Logic and Artificial Intelligence*, pp. 9–34, Dordrecht, Netherlands: Kluwer.

Fagin, R. and J. Y. Halpern (1994). Reasoning about knowledge and probability. *Journal of the ACM 41*(2), 340–367.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1994). Knowledge-based programming. Research Report RJ 9711, IBM.

Fagin, R., J. Y. Halpern, and M. Y. Vardi (1990). A nonstandard approach to the logical omniscience problem. In R. Parikh (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, pp. 41–55. San Francisco, Calif.: Morgan Kaufmann. To appear in *Artificial Intelligence*.

Fagin, R., J. Y. Halpern, and M. Y. Vardi (1992a). What can machines know? On the properties of knowledge in distributed systems. *Journal of the ACM 39*(2), 328–376.

Fagin, R., J. Y. Halpern, and M. Y. Vardi (1992b). What is an inference rule? *Journal of Symbolic Logic 57*(3), 1018–1045.

Fagin, R. and M. Y. Vardi (1986). Knowledge and implicit knowledge in a distributed environment: preliminary report. In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 187–206. San Francisco, Calif.: Morgan Kaufmann.

Fischer, M. J. (1983). The consensus problem in unreliable distributed systems. Technical Report RR-273, Yale University.

Fischer, M. J. and N. Immerman (1986). Foundations of knowledge for distributed systems. In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 171–186. San Francisco, Calif.: Morgan Kaufmann.

Fischer, M. J. and N. Immerman (1987). Interpreting logics of knowledge in propositional dynamic logic with converse. *Information Processing Letters 25*(3), 175–182.

Fischer, M. J. and R. E. Ladner (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences 18*(2), 194–211.

Fischer, M. J. and N. A. Lynch (1982). A lower bound on the time to assure interactive consistency. *Information Processing Letters 14*(4), 183–186.

Fischer, M. J. and L. D. Zuck (1987). Relative knowledge and belief (extended abstract). Technical Report YALEU/DCS/TR-589, Yale University.

Frege, G. (1892). Über sinn und bedeutung. *Zeitschrift für Philosophie und Philosophische Kritik 100*, 25–50. English translation in P. Geach and M. Black (1952), On sense and reference, *Translations From the Writings of Gottlob Frege*, Oxford, U.K.: Basil Blackwell.

Friedman, N. and J. Y. Halpern (1994). A knowledge-based framework for belief change. Part I: Foundations. In R. Fagin (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pp. 44–64. San Francisco, Calif.: Morgan Kaufmann.

Fudenberg, D. and J. Tirole (1991). *Game Theory*. Cambridge, Mass.: MIT Press.

Gabbay, D., A. Pnueli, S. Shelah, and J. Stavi (1980). On the temporal analysis of fairness. In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pp. 163–173.

Gamow, G. and M. Stern (1958). *Puzzle Math*. New York: Viking Press.

Gardner, M. (1977). The "jump proof" and its similarity to the toppling of a row of dominoes. *Scientific American 236*, 128–135.

Gardner, M. (1984). *Puzzles From Other Worlds*. New York: Viking Press.

Garson, J. W. (1977). Quantification in modal logic. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic, Vol. II*, pp. 249–307. Dordrecht, Netherlands: Reidel.

Geanakopolos, J. (1989). Game theory without partitions, and applications to speculation and consensus. Cowles Foundation Discussion Paper #914, Yale University.

Geanakopolos, J. (1992). Common knowledge. In Y. Moses (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fourth Conference*, pp. 255–315. San Francisco, Calif.: Morgan Kaufmann.

Geanaokoplos, J. and H. Polemarchakis (1982). We can't disagree forever. *Journal of Economic Theory 28*(1), 192–200.

Georgeff, M. P. (1985). Attributing attitudes to machines. Manuscript.

Gettier, E. (1963). Is justified true belief knowledge? *Analysis 23*, 121–123.

Goldblatt, R. (1992). *Logics of Time and Computation* (2nd ed.). CSLI Lecture Notes Number 7. Center for Studies in Language and Information, Stanford University.

Goldwasser, S., S. Micali, and C. Rackoff (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*(1), 186–208.

Gouda, M. (1985). On "A simple protocol whose proof isn't". *IEEE Transactions on Communications COM-33*(4), 382–384.

Gray, J. (1978). Notes on database operating systems. In R. Bayer, R. M. Graham, and G. Seegmuller (Eds.), *Operating Systems: An Advanced Course*, Lecture Notes in Computer Science, Vol. 66. Berlin/New York: Springer-Verlag. Also appears as IBM Research Report RJ 2188, 1978.

Grove, A. J. and J. Y. Halpern (1991). Naming and identity in a multi-agent epistemic logic. In J. A. Allen, R. Fikes, and E. Sandewall (Eds.), *Principles of Knowledge Representation and Reasoning: Proc. Second International Conference (KR '91)*, pp. 301–312. San Francisco, Calif.: Morgan Kaufmann.

Hadzilacos, V. (1987). A knowledge-theoretic analysis of atomic commitment protocols. In *Proc. 6th ACM Symp. on Principles of Database Systems*, pp. 129–134. A revised version has been submitted for publication.

Hailpern, B. T. (1982). *Verifying Concurrent Processes using Temporal Logic*. Lecture Notes in Computer Science, Vol. 129. Berlin/New York: Springer-Verlag.

Hailpern, B. T. (1985). A simple protocol whose proof isn't. *IEEE Transactions on Communications COM-33*(4), 330–337.

Hailpern, B. T. and S. S. Owicki (1983). Modular verification of communication protocols. *IEEE Transactions on Communications COM-31*(1), 56–68.

Halpern, J. Y. (1986). Reasoning about knowledge: an overview. In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 1–17. San Francisco, Calif.: Morgan Kaufmann. Reprinted in *Proc. National Computer Conference*, 1986, pp. 219–228.

Halpern, J. Y. (1987). Using reasoning about knowledge to analyze distributed systems. In J. F. Traub, B. J. Grosz, B. W. Lampson, and N. J. Nilsson (Eds.), *Annual Review of Computer Science, Vol. 2*, pp. 37–68. Palo Alto, Calif.: Annual Reviews Inc.

Halpern, J. Y. (1991). A note on knowledge-based protocols and specifications. Research Report RJ 8454, IBM.

Halpern, J. Y. (1993a). Reasoning about knowledge: a survey circa 1991. In A. Kent and J. G. Williams (Eds.), *Encyclopedia of Computer Science and Technology, Volume 27 (Supplement 12)*. New York: Marcel Dekker.

Halpern, J. Y. (1993b). Reasoning about only knowing with many agents. In *Proc. National Conference on Artificial Intelligence (AAAI '93)*, pp. 655–661.

Halpern, J. Y. and R. Fagin (1985). A formal model of knowledge, action, and communication in distributed systems: preliminary report. In *Proc. 4th ACM Symp. on Principles of Distributed Computing*, pp. 224–236.

Halpern, J. Y. and R. Fagin (1989). Modelling knowledge and action in distributed systems. *Distributed Computing 3*(4), 159–179. A preliminary version appeared in *Proc. 4th ACM Symposium on Principles of Distributed Computing*, 1985, with the title "A formal model of knowledge, action, and communication in distributed systems: Preliminary report".

Halpern, J. Y., N. Megiddo, and A. Munshi (1985). Optimal precision in the presence of uncertainty. *Journal of Complexity 1*, 170–196.

Halpern, J. Y. and Y. Moses (1984). Towards a theory of knowledge and ignorance. In *Proc. AAAI Workshop on Non-monotonic Logic*, pp. 125–143. Reprinted in K. Apt (Ed.), *Logics and Models of Concurrent Systems*, Springer-Verlag, Berlin/New York, pp. 459–476, 1985.

Halpern, J. Y. and Y. Moses (1990). Knowledge and common knowledge in a distributed environment. *Journal of the ACM 37*(3), 549–587. A preliminary version appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

Halpern, J. Y. and Y. Moses (1992). A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence 54*, 319–379.

Halpern, J. Y., Y. Moses, and M. R. Tuttle (1988). A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symp. on Theory of Computing*, pp. 132–147.

Halpern, J. Y., Y. Moses, and M. Y. Vardi (1994). Algorithmic knowledge. In R. Fagin (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pp. 255–266. San Francisco, Calif.: Morgan Kaufmann.

Halpern, J. Y., Y. Moses, and O. Waarts (1990). A characterization of eventual Byzantine agreement. In *Proc. 9th ACM Symp. on Principles of Distributed Computing*, pp. 333–346.

Halpern, J. Y. and J. H. Reif (1983). The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science 27*, 127–165.

Halpern, J. Y. and M. R. Tuttle (1993). Knowledge, probability, and adversaries. *Journal of the ACM 40*(4), 917–962.

Halpern, J. Y. and M. Y. Vardi (1986). The complexity of reasoning about knowledge and time. In *Proc. 18th ACM Symp. on Theory of Computing*, pp. 304–315.

Halpern, J. Y. and M. Y. Vardi (1988a). The complexity of reasoning about knowledge and time in asynchronous systems. In *Proc. 20th ACM Symp. on Theory of Computing*, pp. 53–65.

Halpern, J. Y. and M. Y. Vardi (1988b). The complexity of reasoning about knowledge and time: synchronous systems. Research Report RJ 6097, IBM.

Halpern, J. Y. and M. Y. Vardi (1989). The complexity of reasoning about knowledge and time, I: lower bounds. *Journal of Computer and System Sciences 38*(1), 195–237.

Halpern, J. Y. and M. Y. Vardi (1991a). Model checking vs. theorem proving: a manifesto. In V. Lifschitz (Ed.), *Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy)*, pp. 151–176. San Diego, Calif.: Academic Press.

Halpern, J. Y. and M. Y. Vardi (1991b). Model checking vs. theorem proving: a manifesto. In J. A. Allen, R. Fikes, and E. Sandewall (Eds.), *Principles of Knowledge Representation and Reasoning: Proc. Second International Conference (KR '91)*, pp. 325–334. San Francisco, Calif.: Morgan Kaufmann.

Halpern, J. Y. and L. D. Zuck (1992). A little knowledge goes a long way: knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM 39*(3), 449–478.

Heyting, A. (1956). *Intuitionism: An Introduction*. Amsterdam: North-Holland.

Hintikka, J. (1957). Quantifiers in deontic logic. *Societas Scientiarumi Fennica, Commentationes Humanarum Literarum 23*(4), 1–23.

Hintikka, J. (1961). Modalities and quantification. *Theoria 27*(61), 119–128.

Hintikka, J. (1962). *Knowledge and Belief*. Ithaca, N.Y.: Cornell University Press.

Hintikka, J. (1975). Impossible possible worlds vindicated. *Journal of Philosophical Logic 4*, 475–484.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Englewood Cliffs, N.J.: Prentice-Hall.

Hoek, W. van der and J.-J. C. Meyer (1992). Making some issues of implicit knowledge explicit. *Int'l. J. of Foundations of Computer Science 3*(2), 193–223.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages and Computation*. New York: Addison-Wesley.

Huang, Z. and K. Kwast (1991). Awareness, negation and logical omniscience. In J. van Eijck (Ed.), *Logics in AI, Proceedings JELIA'90*, Lecture Notes in Computer Science, Vol. 478, pp. 282–300. Berlin/New York: Springer-Verlag.

Hughes, G. E. and M. J. Cresswell (1968). *An Introduction to Modal Logic*. London: Methuen.

Hughes, G. E. and M. J. Cresswell (1984). *A Companion to Modal Logic*. London: Methuen.

Kanger, S. (1957a). The morning star paradox. *Theoria 23*, 1–11.

Kanger, S. (1957b). *Provability in Logic*. Stockholm Studies in Philosophy I.

Kaplan, D. (1966). Review of "A semantical analysis of modal logic I: normal modal propositional calculi". *Journal of Symbolic Logic 31*, 120–122.

Kaplan, D. (1969). Quantifying in. *Synthese 19*, 178–214.

Katz, S. and G. Taubenfeld (1986). What processes know: definitions and proof methods. In *Proc. 5th ACM Symp. on Principles of Distributed Computing*, pp. 249–262.

Kifer, M. and E. L. Lozinski (1992). A logic for reasoning with inconsistency. *J. Automated Deduction 9*, 179–215.

Konolige, K. (1986). *A Deduction Model of Belief.* San Francisco, Calif.: Morgan Kaufmann.

Koo, R. and S. Toueg (1988). Effects of message loss on the termination of distributed programs. *Information Processing Letters 27*, 181–188.

Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science 27*(1), 333–354.

Kozen, D. and R. Parikh (1981). An elementary proof of the completeness of PDL. *Theoretical Computer Science 14*(1), 113–118.

Krasucki, P., R. Parikh, and G. Ndjatou (1990). Probabilistic knowledge and probabilistic common knowledge (preliminary report). In Z. W. Ras, M. Zemankova, and M. L. Emrich (Eds.), *Methodologies for Intelligent Systems*, Volume 5, pp. 1–8. The Hague: Elsevier Science Publishing Co., Inc.

Kraus, S. and D. J. Lehmann (1988). Knowledge, belief, and time. *Theoretical Computer Science 58*, 155–174.

Kripke, S. (1959). A completeness theorem in modal logic. *Journal of Symbolic Logic 24*, 1–14.

Kripke, S. (1963a). A semantical analysis of modal logic I: normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9*, 67–96. Announced in (1959) *Journal of Symbolic Logic 24*, 323.

Kripke, S. (1963b). Semantical considerations on modal logic. *Acta Philosophica Fennica 24*, 83–94.

Kripke, S. (1965). A semantical analysis of modal logic II: non-normal propositional calculi. In L. Henkin and A. Tarski (Eds.), *The Theory of Models*, pp. 206–220. Amsterdam: North-Holland.

Kurki-Suonio, R. (1986). Towards programming with knowledge expressions. In *Proc. 13th ACM Symposium on Principles of Programming Languages*, pp. 140–149.

Ladner, R. E. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing 6*(3), 467–480.

Ladner, R. E. and J. H. Reif (1986). The logic of distributed protocols (preliminary report). In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 207–222. San Francisco, Calif.: Morgan Kaufmann.

Lakemeyer, G. (1987). Tractable meta-reasoning in propositional logics of belief. In *Proc. Tenth International Joint Conference on Artificial Intelligence (IJCAI '87)*, pp. 402–408.

Lakemeyer, G. (1993). All they know: a study in multi-agent autoepestemic reasoning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93)*, pp. 376–381.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM 21*(7), 558–565.

Lamport, L. (1980). "Sometimes" is sometimes "not never": On the temporal logic of programs. In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pp. 164–185.

Lamport, L. (1985). Paradigms for distributed computing. In M. Paul and H. J. Siegert (Eds.), *Methods and tools for specification, an advanced course*, Lecture Notes in Computer Science, Vol. 190, pp. 19–30, 454–468. Berlin/New York: Springer-Verlag.

Lamport, L. (1986). On interprocess communication, Part I: basic formalism. *Distributed Computing 1*(2), 77–85.

Lamport, L. and M. J. Fischer (1982). Byzantine generals and transactions commit protocols. Technical Report Opus 62, SRI International, Menlo Park, Calif.

Lehmann, D. J. (1984). Knowledge, common knowledge, and related puzzles. In *Proc. 3rd ACM Symp. on Principles of Distributed Computing*, pp. 62–67.

Lemmon, E. J. (1977). *The "Lemmon Notes": An Introduction to Modal Logic*. Oxford, U.K.: Basil Blackwell. Written in collaboration with Dana Scott; edited by Krister Segerberg. American Philosophical Quarterly Monograph Series. Monograph No. 11.

Lenzen, W. (1978). Recent work in epistemic logic. *Acta Philosophica Fennica 30*, 1–219.

Levesque, H. J. (1981). The interaction with incomplete knowledge bases: a formal treatment. In *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 240–245.

Levesque, H. J. (1984a). Foundations of a functional approach to knowledge representation. *Artificial Intelligence 23*, 155–212.

Levesque, H. J. (1984b). A logic of implicit and explicit belief. In *Proc. National Conference on Artificial Intelligence (AAAI '84)*, pp. 198–202.

Levesque, H. J. (1985). Global and local consistency and completeness of beliefs. Manuscript.

Levesque, H. J. (1988). Logic and the complexity of reasoning. *Journal of Philosophical Logic 17*(4), 355–389.

Levesque, H. J. (1990). All I know: A study in autoepistemic logic. *Artificial Intelligence 42*(3), 263–309.

Lewis, C. I. and C. H. Langford (1959). *Symbolic Logic* (2nd ed.). New York: Dover.

Lewis, D. (1969). *Convention, A Philosophical Study*. Cambridge, Mass.: Harvard University Press.

Lichtenstein, O., A. Pnueli, and L. Zuck (1985). The glory of the past. In R. Parikh (Ed.), *Proc. Workshop on Logics of Programs*, Lecture Notes in Computer Science, Vol. 193, pp. 196–218. Berlin/New York: Springer-Verlag.

Lipman, B. L. (1992a). Decision theory with impossible possible worlds. Technical Report Working paper, Queen's University.

Lipman, B. L. (1992b). Logics for nonomniscient agents: An axiomatic approach. Technical Report Working paper 874, Queen's University. Forthcoming in *Proceedings of the Second Castiglioncello Conference*, C. Bicchieri and B. Skyrms (editors), Cambridge University Press.

Lipman, B. L. (1994). An axiomatic approach to the logical omniscience problem. In R. Fagin (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pp. 182–196. San Francisco, Calif.: Morgan Kaufmann.

Lynch, N. A. and M. J. Fischer (1981). On describing the behavior and implementation of distributed systems. *Theoretical Computer Science 13*, 17–43.

Lynch, N. A. and M. R. Tuttle (1989). An introduction to input/output automata. *CWI Quarterly 2*(3), 219–246. Also available as MIT Technical Memo MIT/LCS/TM-373.

Makinson, D. (1966). On some completeness theorems in modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 12*, 379–384.

Manna, Z. and A. Pnueli (1992). *The Temporal Logic of Reactive and Concurrent Systems*, Volume 1. Berlin/New York: Springer-Verlag.

Mazer, M. S. (1990). A link between knowledge and communication in faulty distributed systems. In R. Parikh (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, pp. 289–304. San Francisco, Calif.: Morgan Kaufmann.

Mazer, M. S. (1991). Implementing distributed knowledge-based protocols. Submitted for publication.

Mazer, M. S. and F. H. Lochovsky (1990). Analyzing distributed commitment by reasoning about knowledge. Technical Report CRL 90/10, DEC-CRL.

McCarthy, J. (1979). Ascribing mental qualities to machines. Technical Report STAN-CS-79-725, Stanford University.

McCarthy, J., M. Sato, T. Hayashi, and S. Igarishi (1979). On the model theory of knowledge. Technical Report STAN-CS-78-657, Stanford University.

Megiddo, N. (1986). Remarks on bounded rationality. Research Report RJ 5270, IBM.

Megiddo, N. (1989). On computable beliefs of rational machines. *Games and Economical Behavior 1*, 144–169.

Megiddo, N. and A. Wigderson (1986). On play by means of computing machines. In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 259–274. San Francisco, Calif.: Morgan Kaufmann.

Meredith, C. A. (1956). Interpretations of different modal logics in the "property calculus", mimeographed manuscript, Philosophy Department, Canterbury University College (recorded and expanded by A. N. Prior).

Merritt, M. J. (1984). Unpublished notes on the Dolev-Strong lower bound for Byzantine agreement.

Merritt, M. J. and G. Taubenfeld (1991). Knowledge in shared memory systems. In *Proc. 10th ACM Symp. on Principles of Distributed Computing*, pp. 189–200.

Meyden, R. van der (1994). Axioms for knowledge and time in distributed systems with perfect recall. In *Proc. 9th IEEE Symp. on Logic in Computer Science*, pp. 448–457.

Meyer, J.-J. C., W. van der Hoek, and G. A. W. Vreeswijk (1991a). Epistemic logic for computer science: a tutorial (Part I). *EATCS Bulletin 44*, 242–270.

Meyer, J.-J. C., W. van der Hoek, and G. A. W. Vreeswijk (1991b). Epistemic logic for computer science: a tutorial (Part II). *EATCS Bulletin 45*, 256–287.

Michel, R. (1989a). A categorical approach to distributed systems, expressibility and knowledge. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pp. 129–143.

Michel, R. (1989b). *Knowledge in Distributed Byzantine Environments*. Ph. D. thesis, Yale University.

Milgrom, P. (1981). An axiomatic characterization of common knowledge. *Econometrica 49*(1), 219–222.

Milgrom, P. and N. Stokey (1982). Information, trade, and common knowledge. *Journal of Economic Theory 26*, 17–27.

Milner, R. (1980). *A Calculus of Communicating Systems*. Lecture Notes in Computer Science, Vol. 92. Berlin/New York: Springer-Verlag.

Monderer, D. and D. Samet (1989). Approximating common knowledge with common beliefs. *Games and Economic Behavior 1*, 170–190.

Montague, R. (1960). Logical necessity, physical necessity, ethics, and quantifiers. *Inquiry 4*, 259–269.

Montague, R. (1968). Pragmatics. In R. Kalibansky (Ed.), *Contemporary Philosophy*, pp. 101–121. Florence, Italy: La Nuova Italia Editrice.

Montague, R. (1970). Universal grammar. *Theoria 36*, 373–398.

Moore, R. C. (1985). A formal theory of knowledge and action. In J. Hobbs and R. C. Moore (Eds.), *Formal Theories of the Commonsense World*, pp. 319–358. Norwood, N.J.: Ablex Publishing Corp.

Moore, R. C. and G. Hendrix (1979). Computational models of beliefs and the semantics of belief sentences. Technical Note 187, SRI International, Menlo Park, Calif.

Moses, Y. (1988). Resource-bounded knowledge. In M. Y. Vardi (Ed.), *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 261–276. San Francisco, Calif.: Morgan Kaufmann.

Moses, Y. (1992). Knowledge and communication (a tutorial). In Y. Moses (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fourth Conference*, pp. 1–14. San Francisco, Calif.: Morgan Kaufmann.

Moses, Y. and B. Bloom (1994). Knowledge, timed precedence and clocks. In *Proc. 13th ACM Symp. on Principles of Distributed Computing*, pp. 294–303.

Moses, Y., D. Dolev, and J. Y. Halpern (1986). Cheating husbands and other stories: a case study of knowledge, action, and communication. *Distributed Computing 1*(3), 167–176.

Moses, Y. and O. Kislev (1993). Knowledge-oriented programming. In *Proc. 12th ACM Symp. on Principles of Distributed Computing*, pp. 261–270.

Moses, Y. and G. Nachum (1990). Agreeing to disagree after all. In R. Parikh (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, San Francisco, Calif., pp. 151–168. Morgan Kaufmann.

Moses, Y. and G. Roth (1989). On reliable message diffusion. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pp. 119–128.

Moses, Y. and Y. Shoham (1993). Belief as defeasible knowledge. *Artificial Intelligence 64*(2), 299–322.

Moses, Y. and M. R. Tuttle (1988). Programming simultaneous actions using common knowledge. *Algorithmica 3*, 121–169.

Neiger, G. (1988). Knowledge consistency: a useful suspension of disbelief. In M. Y. Vardi (Ed.), *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 295–308. San Francisco, Calif.: Morgan Kaufmann.

Neiger, G. and R. Bazzi (1992). Using knowledge to optimally achieve coordination in distributed systems. In Y. Moses (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fourth Conference*, pp. 43–59. San Francisco, Calif.: Morgan Kaufmann.

Neiger, G. and S. Toueg (1990). Automatically increasing the fault-tolerance of distributed algorithms. *Journal of Algorithms 11*(3), 374–419.

Neiger, G. and S. Toueg (1993). Simulating real-time clocks and common knowledge in distributed systems. *Journal of the ACM 40*(2), 334–367.

Neiger, G. and M. R. Tuttle (1993). Common knowledge and consistent simultaneous coordination. *Distributed Computing 6*(3), 334–352.

Newell, A. (1982). The knowledge level. *Artificial Intelligence 18*, 87–127.

Neyman, A. (1985). Bounded complexity justifies cooperation in finitely repated prisoner's dilemma. *Economic Letters*, 227–229.

Orlowska, E. (1989). Logic for reasoning about knowledge. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 35*, 559–572.

Panangaden, P. and S. Taylor (1992). Concurrent common knowledge: Defining agreement for asynchronous systems. *Distributed Computing 6*(2), 73–94.

Parikh, R. (1987). Knowledge and the problem of logical omniscience. In Z. W. Ras and M. Zemankova (Eds.), *Methodologies of Intelligent Systems*, pp. 432–439. The Hague: Elsevier Science Publishing Co., Inc.

Parikh, R. (1990). Recent issues in reasoning about knowledge. In R. Parikh (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, pp. 3–10. San Francisco, Calif.: Morgan Kaufmann.

Parikh, R. (1991). Monotonic and nonmonotonic logics of knowledge. *Fundamenta Informaticae 15*(3,4), 255–274.

Parikh, R. (1992). Finite and infinite dialogues. In Y. N. Moshovakis (Ed.), *Logic from Computer Science, MSRI Publication No. 21*, pp. 481–497. Berlin/New York: Springer-Verlag.

Parikh, R. and P. Krasucki (1990). Communication, consensus, and knowledge. *Journal of Economic Theory 52*(1), 178–189.

Parikh, R. and P. Krasucki (1992). Levels of knowledge in distributed computing. *Sādhanā 17*(1), 167–191.

Parikh, R. and R. Ramanujam (1985). Distributed processing and the logic of knowledge. In R. Parikh (Ed.), *Proc. Workshop on Logics of Programs*, pp. 256–268.

Patel-Schneider, P. F. (1985). A decidable first-order logic for knoweledge representation. In *Proc. Ninth International Joint Conference on Artificial Intelligence (IJCAI '85)*, pp. 455–458.

Patel-Schneider, P. F. (1989). A four-valued semantics for terminological logics. *Artificial Intelligence 38*, 319–351.

Pease, M., R. Shostak, and L. Lamport (1980). Reaching agreement in the presence of faults. *Journal of the ACM 27*(2), 228–234.

Perrault, C. R. and P. R. Cohen (1981). It's for your own good: a note on inaccurate reference. In A. K. Johsi, B. L. Webber, and I. A. Sag (Eds.), *Elements of discourse understanding*. Cambridge, U.K.: Cambridge University Press.

Plantinga, A. (1974). *The Nature of Necessity*. Oxford, U.K.: Oxford University Press.

Pratt, V. R. (1979). Models of program logics. In *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 115–122.

Pratt, V. R. (1982). On the composition of processes. In *Proc. 9th ACM Symp. on Principles of Programming Languages*, pp. 213–223.

Pratt, V. R. (1985). Modelling concurrency with partial orders. *International Journal of Parallel Programming 15*(1), 33–71.

Prior, A. N. (1956). Modality and quantification in S5. *Journal of Symbolic Logic 21*, 60–62.

Prior, A. N. (1957). *Time and Modality*. Oxford, U.K.: Oxford University Press.

Prior, A. N. (1962). Possible worlds (idea attributed to P. T. Geach). *Philosophical Quarterly 12*, 36–43.

Quine, W. V. O. (1947). The problem of interpreting modal logic. *Journal of Symbolic Logic 12*, 43–48.

Rantala, V. (1982). Impossible worlds semantics and logical omniscience. *Acta Philosophica Fennica 35*, 18–24.

Rasmusen, E. (1989). *Games and Information: An Introduction to Game Theory*. Oxford, U.K. and Cambridge, Mass.: Basil Blackwell.

Rescher, N. and R. Brandom (1979). *The Logic of Inconsistency*. Totowa, N.J.: Rowman and Littlefield.

Rivest, R. L., A. Shamir, and L. Adelman (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*(2), 120–126.

Rogers, Jr., H. (1967). *Theory of Recursive Functions and Effective Computability*. New York: McGraw-Hill.

Rosenschein, S. J. (1985). Formal theories of AI in knowledge and robotics. *New Generation Computing 3*, 345–357.

Rosenschein, S. J. and L. P. Kaelbling (1986). The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pp. 83–97. San Francisco, Calif.: Morgan Kaufmann.

Roth, G. (1989). Message diffusion in anonymous distributed systems. Master's thesis, Weizmann Institute of Science.

Routley, R. and V. Routley (1972). Semantics of first degree entailment. *Noûs 6*, 335–359.

Rubinstein, A. (1985). Finite automata play the repeated prisoner's dilemma. ST/ICERD Discussion Paper 85/109, London School of Economics.

Rubinstein, A. (1989). The electronic mail game: strategic behavior under "almost common knowledge". *American Economic Review 79*, 385–391.

Rubinstein, A. and A. Wolinsky (1990). On the logic of "agreeing to disagree" type results. *Journal of Economic Theory 51*, 184–193.

Samet, D. (1987). Ignoring ignorance and agreeing to disagree. MEDS discussion paper, Northwestern University.

Sanders, B. (1991). A predicate transformer approach to knowledge and knowledge-based protocols. In *Proc. 10th ACM Symp. on Principles of Distributed Computing*, pp. 217–230. A revised report appears as ETH Informatik Technical Report 181, 1992.

Savage, L. J. (1954). *Foundations of Statistics*. New York: John Wiley & Sons.

Scott, D. (1970). Advice on modal logic. In K. Lambert (Ed.), *Philosophical Problems in Logic*, pp. 143–173. Dordrecht, Netherlands: Reidel.

Segerberg, K. (1968). *Results in Nonclassical Logic*. Lund, Sweden: Berlingska Boktryckeriet.

Segerberg, K. (1971). *An Essay in Classical Modal Logic*. Philosophical Studies 13. Uppsala University.

Shalpey, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences 39*, 1095–1100.

Shin, H. S. and T. Williamson (1993). Representing the knowledge of Turing machines. Manuscript, University College, Oxford.

Shoham, Y. (1993). Agent oriented programming. *Artificial Intelligence 60*(1), 51–92.

Sistla, A. P. and E. M. Clarke (1985). The complexity of propositional linear temporal logics. *Journal of the ACM 32*(3), 733–749.

Spaan, E. (1990). Nexttime is not necessary. In R. J. Parikh (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, pp. 241–256. San Francisco, Calif.: Morgan Kaufmann.

Stalnaker, R. (1985). *Inquiry*. Cambridge, Mass.: MIT Press.

Stark, W. R. (1981). A logic of knowledge. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 27*, 371–374.

Stenning, M. V. (1976). A data transfer protocol. *Comput. Networks 1*, 99–110.

Tarski, A. (1955). A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics 5*, 285–309.

Thijsse, E. (1992). *On Partial Logic and Knowledge Representation*. Netherlands: Eburon, Delft. (Ph.D. dissertation, Tilburg University.).

Thijsse, E. (1993). On total awareness logics. In M. de Rijke (Ed.), *Diamonds and Defaults*, pp. 309–347. Dordrecht, Netherlands: Kluwer.

Thomason, R. H. (1984). Combinations of tense and modality. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic, Vol. II*, pp. 135–165. Dordrecht, Netherlands: Reidel.

Thorp, E. O. (1961). A favorable strategy for twenty-one. *Proc. Natl. Acad. Sci. 47*(1), 110–112.

Thorp, E. O. (1966). *Beat the dealer* (2nd ed.). New York: Vintage.

Vardi, M. Y. (1985). A model-theoretic analysis of monotonic knowledge. In *Proc. Ninth International Joint Conference on Artificial Intelligence (IJCAI '85)*, pp. 509–512.

Vardi, M. Y. (1989). On the complexity of epistemic reasoning. In *Proc. 4th IEEE Symp. on Logic in Computer Science*, pp. 243–252.

Voorbraak, F. (1992). Generalized Kripke models for epistemic logic. In Y. O. Moses (Ed.), *Theoretical Aspects of Reasoning about Knowledge: Proc. Fourth Conference*, pp. 214–228. San Francisco, Calif.: Morgan Kaufmann.

Wansing, H. (1990). A general possible worlds framework for reasoning about knowledge and belief. *Studia Logica 49*(4), 523–539.

Wright, G. H. von (1951). *An Essay in Modal Logic*. Amsterdam: North-Holland.

Yemini, Y. and D. Cohen (1979). Some issues in distributed processes communication. In *Proc. of the 1st International Conf. on Distributed Computing Systems*, pp. 199–203.

Zadrozny, W. (1985). Explicit and implicit beliefs, a solution of a problem of H. Levesque. Manuscript.