Reasoning about Resource-Bounded Knowledge

Theory and Application to Security Protocol Analysis

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Riccardo Pucella

August 2004

Abstract

EXISTING approaches for analyzing security protocols, while quite successful, are limited in a number of ways. One limitation is that they often do not supply a specification language. Another limitation is that the model of the adversary is quite restricted, unable to capture protocol-specific knowledge or to support guessing. Informal specifications of security in the literature are typically phrased in terms of knowledge. It thus seems natural to use an epistemic logic as a specification language, where specifications can be written directly in terms of knowledge. However, the standard interpretation of knowledge in such logics suffers from the logical omniscience problem: agents know all logical consequences of their knowledge. This gives a notion of knowledge too strong for the purpose of reasoning about security, since the adversary knows information that no realistic adversary should know. Using a notion known as algorithmic knowledge it is possible to define a logic for reasoning about security protocol under different adversary models, where adversaries use algorithms to compute their knowledge.

The contributions of this dissertation are two-fold. Firstly, we develop the theory of algorithmic knowledge in more depth. More precisely, we investigate the properties of the logic when the knowledge algorithms implement deductions in a logical theory for the agents and when the knowledge algorithms are randomized. Dealing with specifications in the presence of randomized knowledge algorithms requires a notion of evidence, a concept heavily studied in the philosophical literature, but not so much in computer science. Secondly, we develop a logic for reasoning about security protocols based on the well-understood notions of knowledge, time, and probability, as well as algorithmic knowledge to capture the capabilities of the adversary. We show this logic is flexible enough to capture many of the adversaries considered in the literature. We finally provide evidence that this logic is sufficiently expressive to reason about security protocols: it can capture subtleties in the handling of nonces that are not captured by non-epistemic approaches to secu-

rity protocol analysis, and it can capture many operators believed to be important for security protocol analysis.

À mes parents

Acknowledgments

I first would like to thank my advisor Joe Halpern. An outstanding scholar with a keen eye for interesting questions and elegant solutions, he taught me much about research and writing. He was perfectly supportive of the process by which I attempted to define myself as a researcher; he was happy to listen to my research ideas, and to read, comment, and critique drafts of research papers. Most importantly, he always encouraged me to collaborate with other researchers.

Cornell was home to a number of wonderful people that have made these past five years fly by. First and foremost, Vicky Weissman, she of the many roles, was my friend, my officemate, my coauthor, my colleague, and my girlfriend; she gave me strength, kept me sane, and rekindled a sense of wonder I thought had died long ago. A number of fellow students played double bill as great friends and great colleagues. Matthew Fluet, Hubie Chen, and Steve Chong were more than ready to listen to my ramblings and try to extract some sense out of them. I benefited from interactions with faculty members, most notably Dexter Kozen and Greg Morrisett, and students in the Programming Language Discussion Group and beyond, especially Kevin O'Neill, Sabina Petride, Nate Nystrom, Michael Clarkson, Dan Marques, James Cheney, as well as Cornell alumni Dave Walker, Steve Zdancewic, and Stephanie Weirich. Delia Graff and Michael Fara, from the Cornell Sage School of Philosophy, have tried to instill (hammer?) in me some critical thinking, for which I am grateful. I would like to thank John Reppy for suggesting I apply to Cornell in the first place.

I would like to thank Andy Gordon, from Microsoft Research in Cambridge, for supporting me through two summer internships and allowing me to spend two summers in England. I would similarly like to thank Ron van der Meyden and the University of New South Wales for their kind invitation to spend the month of May 2002 in Sydney, working on what would become Chapter 9.

I am grateful for the support provided by the NSF, the ONR, and the AFOSR during the course of my research. I was supported at various times by NSF under

grant CTC-0208535, by ONR under grants N00014-00-1-03-41, N00014-01-10-511, and N00014-02-1-0455, by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795, and by AFOSR under grant F49620-02-1-0101.

Last but not least, I would like to thank my family. Katia and my daughter Alexandra saw me through my first years of graduate school. Alex, love of my life, may you always keep your innocence; I miss you. I am grateful to my parents and my brother for their unfaltering support and for always encouraging me to choose my own path. I am who I am and where I am today because of them.

It is fitting that, without planning to, I ended up in Ithaca. This was the home of Carl Sagan. In no small part, he stands at the genesis of it all; watching *Cosmos* on the tube at too young an age is bound to warp a child's mind, and it did. He passed away a few years before I arrived. I am sorry I missed him; I would have said thank you.

Riccardo Pucella
Ithaca, August 11, 2004.

Contents

List of Figures

1
Introduction

THIS dissertation demonstrates that a logic for reasoning about resource-bounded knowledge provides a good foundation for reasoning about security protocols. Formal theories of knowledge have proved useful in a number of fields, including artifical intelligence, economics, distributed computing, and security. However, most theories of knowledge do not take resources available to agents into account; this makes it difficult to reason about the complexity of establishing knowledge. In the first part of this dissertation, we study a logical theory of resource-bounded knowledge that captures a computational view of knowledge. In the second part of this dissertation, we apply the theory studied in the first part to the problem of reasoning about security protocols.

1.1 Knowledge and Evidence

The notion of knowledge has been a central preoccupation for western philosophy since Aristotle. While philosophers have mostly been concerned with questions such as "what is knowledge?", "what is it possible to know?", "are there facts that can be known and that are not derived from experience?", knowledge turns out to be a useful concept to formalize various situations that can be analyzed most naturally in terms of who knows what information. To illustrate this issue, consider the following well-known puzzle.

> A certain village contains a number of married couples, of which k husbands are cheating on their wives. It is a well-known fact within the community that every woman knows about all the cheating taking place, except for the infidelities of her own husband. The village is a matriarchal society which demands that husbands remain faithful to their wives. If a wife discovers irrefutable evidence that her husband has been cheating on her, she has to kill him that night and dump his body in the town square for all to see. One day, at a town meeting, the chief announces, "There is some cheating in this village, and I want it to stop." Then,

> $k-1$ nights pass uneventfully. After the kth night, however, the bodies of the k cheating husbands are found in the town square. How did this happen?

The argument for explaining the puzzle goes roughly as follows. The idea is to show that for the first $k-1$ nights, none of the cheated wives kills her husband, but after the kth night, all the cheated wives kill their husband. If $k = 1$, the result is immediate; after the first night, the single cheated wife notices that no other husband has cheated, so she knows that her own husband has cheated. If $k = 2$, the result is similar. Say that a and b are the cheated wives. After the first night, a notices that b's husband has cheated, but b did not kill him. The only reason that b did not kill her husband must be that she was not sure that he was cheating, that is, she must have witnessed another husband cheating. Since a only noticed b's husband cheating, it must be that her own husband was cheating. Wife b follows the same reasoning to establish that her own husband was cheating. Thus, both wife a and b know that their husbands cheated, and they promptly kill them the second night. A similar argument works for all values of k.

A key point for this argument to go through is the initial statement by the village chief that there is at least one husband cheating. It is interesting to ask why. Intuitively, the initial statement establishes common knowledge among the wives that there is at least one cheating husband, that is, all the wives know that there is at least one cheating husband, all the wives know that all the wives know that there is at least one cheating husband, and so on. This knowledge is needed in the above argument, for instance, for wife a to reason that b did not kill her husband because she saw another cheating.

The argument above is informal, but can be formalized in a straightforward way. A good formal model of knowledge helps explain the subtleties involved in the above puzzle, and the role of the utterance of the village chief. It also has applications to fields other than puzzle-solving.

Artificial Intelligence. One of the goals of artificial intelligence research is to design rational independent agents. To achieve this, it seems that agents should be based on notions such as belief, knowledge, intention, and so on. Thus, a formal model of knowledge is useful alongside other formal models. Knowledge in fact has been studied from at least two distinct perspectives in artificial intelligence. First, in any intelligent being, knowledge is a process that seems central to the understanding of cognition, that is, an understanding of the way the higher-level functions of the brain work. This is held by many researchers as a prerequisite to the building of intelligence in machines. For instance, in order to help machines understand natural language, it is useful to have an idea how humans understand language, from the perception of sounds to the formation of concepts. Knowledge

is an intrinsic part of this equation, so that a study of knowledge is part of a complete study of cognitive models for artificial intelligence. The second perspective from which knowledge has been studied is that of finding useful representations for the knowledge an agent has about his environment, about his internal state, and so on. Often, the representation of knowledge is very specific (a database of facts, a set of logical formulas), and research in knowledge representation involves finding efficient ways of querying for knowledge in such representations and updating the representations when new information is available to the agent.

Economics and Decision Theory. In decision theory, one often studies economic agents engaging in competitive behaviour to maximize gain. The basis of many decisions can be understood as a function of the knowledge of the agents. This is often formalized via game theory, a theory that goes back to the seminal work of John von Neumann and Oskar Morgenstern in the 1940s. Games, in this sense, are typically taken to be games of strategy, such as chess and checkers. Games such as backgammon, that involve both strategy and chance, are also considered. One of the important distinctions to make when analyzing games is whether the game is one of perfect information, where the participants have access to all the information relevant to the game, as in chess, or a game of imperfect information, where some of the information is hidden from the participants. In the latter case, the information each participant has access to, his information set, is exactly his knowledge. Game theory could be characterized theory as the theory of how to make the best decision, based on one's knowledge.

Distributed Computing. In distributed computing, one studies programs that consist of different processes running essentially independently on different machines and communicating by various means. In this context, knowledge turns out to be a useful form of specification. For example, suppose that two processes engage in a protocol to synchronize their content, such as two databases. After completion of the protocol, a desirable guarantee is that each process proceeds only if it is sure that the content was successfully synchronized, even in the presence of network failures. One way to enforce this guarantee is simply to force each process to confirm to the other that the exchange was successful, and to proceed only once that acknowledgment is received. Pending issues such as what happens if an acknowledgment is lost, there are many other ways in which such a confirmation could be implemented. What is an appropriate specification of the desired behaviour that does not force a particular number of details into the specification? A specification that is too precise would say, for example, that at the end of the execution of the protocol, the acknowledgments have been received. However, such

a specification really works only for a particular way of establishing confirmation. A more widely applicable specification is to say that at the end of the protocol, each process *knows* that the other process has successfully synchronized. Now, whether this knowledge is achieved by explicit acknowledgments or some other mechanism is immaterial to the specification. Of course, this requires a formal definition of knowledge for processes.

As the examples above show, there are many areas where a formal theory of knowledge is useful. The examples also highlight two distinct uses of such a theory. On the one hand, knowledge can be used by someone designing or analyzing a system to verify that the system behaves in a way consistent with a specification that is best expressed using knowledge. In this case, knowledge should be viewed as something ascribed to agents by an external observer. In the synchronization protocol example above, the designer of a distributed protocol wants to check that the protocol behaves as follows: "once the process knows that the synchronization succeeded, it proceeds". There is no question of the processes themselves determining whether or not they know whether the synchronization succeeded. The processes follow a particular implementation, that for instance allows them to proceed once they receive an acknowledgment from the other process. The specification, however, is given in terms of the more general notion of knowledge. This form of knowledge is known as *implicit knowledge*, because it can be understood as the knowledge implicit in the situation of the agent. This form of knowledge has been extensively studied in the literature and leads to a rather elegant theory, which we review in Chapter 2.

A different use of knowledge is as something concrete that agents can manipulate. For example, independent agents in artifical intelligence often need to base their decisions on their knowledge of the environment, such as whether there is an obstacle on their path, in the case of robotic agents. This knowledge can come directly from sensors, or from indirect reasoning such as the fact that another robotic agent had to change direction to avoid the obstacle. This requires the agent to determine whether or not he knows a fact. This form of knowledge is known as *explicit knowledge*, because it can be understood as the knowledge that the agent explicitly needs to compute in order to make decisions. Clearly, whether or not the agent explicitly knows a fact should depend on his resources. There can be time constraints restricting the amount of deduction that can be performed, or there may be limited computational abilities, such as only a small memory available on robot processors. Reasoning about knowledge in such a setting calls for a theory of resource-bounded knowledge, that can be used to model situations where agents that are resource-bounded need to explicitly compute their knowledge.

A general way of modeling explicit knowledge, known as *algorithmic knowledge*, provides every agent in the system with a "knowledge algorithm" that they

can use to answer questions about their knowledge. This framework can capture most of the approaches to modeling explicit knowledge from the literature. It is then possible to develop a logic to reason about the implicit and explicit knowledge of agents, where the explicit knowledge is modeled using algorithmic knowledge. This logic can be used as a specification language to write specifications that talk about what agents know both implicitly and explicitly, and to reason about what agents know about each other's ability to compute what they know. We review this approach in Chapter 2.

One possible interpretation of algorithmic knowledge is to view it as a way for an agent to "test" whether or not he knows a fact. Many properties of algorithmic knowledge depend on whether or not this test for knowledge is "sound", that is, if whenever the knowledge algorithm answers "Yes" when asked whether the agent knows a particular fact, then the agent indeed (implicitly) knows that fact. If a knowledge algorithm is sound, then algorithmic knowledge of a fact implies knowledge of that fact. Not all knowledge algorithms are sound in this way; many of them are "almost sound", especially those that involve some randomization. Despite the fact that these algorithms are sometimes wrong, it certainly seems that if the probability that the algorithm gives the wrong answer is low, it provides very useful information when it says "Yes" to a query. This intuition appears in the randomized algorithms literature, where a "Yes" answer from a highly reliable randomized algorithm that is, one with a low probability of being wrong, is deemed "good enough". It is certainly not the case that a "Yes" answer to a query F from a highly reliable randomized knowledge algorithm should make the probability that F is true be high. Rather, the information should be viewed as *evidence* that the fact F is true; the probability that F is true also depends in part on the prior probability of the fact.

The notion of evidence, like that of knowledge, has been widely discussed in the philosophical literature. Much of this discussion occurs in the philosophy of science, specifically *confirmation theory*, where the concern has been historically to assess the support that evidence obtained through experimentation lends to various scientific theories. Confirmation theory aims at determining and measuring the support a piece of evidence provides a hypothesis. Many different measures of confirmation have been proposed in the literature. Typically, a proposal has been judged on the degree to which it satisfies various properties that are considered appropriate for confirmation. For example, it may be required that a piece of evidence e confirms a hypothesis h if and only if e makes h more probable.

Such a notion of evidence is more generally relevant to understanding situations involving a combination of nondeterministic choices and probabilistic choices. The following example illustrates the issues involved. Suppose Alice chooses one of two coins, nondeterministically. One is fair, meaning it has probability $\frac{1}{2}$ of land-

ing heads and probability $\frac{1}{2}$ of landing tails when tossed, and the other is double-headed. Bob, who does not see which coin Alice chose, sees her toss the coin a hundred times, landing on heads on every toss. What is the probability, according to Bob, that the next coin toss lands heads as well? Since the example does not give a probability distribution on Alice's choice of coin, this question has no precise answer, beyond the unsatisfying "the probability of the next coin toss landing heads is either $\frac{1}{2}$ or 1". Yet, there is an intuition that it is more likely than not that the coin is double-headed. This can be made clearer if Bob sees an additional thousand coin tosses, all landing heads. Again, it is not possible to assign a probability to the event of the next coin toss landing heads, but now it is even more likely for the coin being double-headed. Something has changed, but what? Intuitively, there is accumulated evidence for the coin to be double-headed, and hence for the fact that the next coin toss will land heads.

The first part of this dissertation is concerned with the study of the form of resource-bounded knowledge modeled using algorithmic knowledge, and its relationship to evidence. The particular contributions of this first part are as follows.

– In Chapter 3, we study a particular form of knowledge algorithm that captures many situations of interest, where the explicit knowledge of an agent is assumed to come from a logical theory in which the agent performs his reasoning. The knowledge algorithm corresponding to such a logical theory is the algorithm that attempts to infer whether a fact is derivable from the deduction rules provided by the agent's logical theory. In contrast with the general form of algorithmic knowledge, where the algorithm is arbitrary, the highly structured presentation of the logical theory permits an axiomatization of the properties of the resulting form of algorithmic knowledge. Formally, we show that the logic interpreted over such knowledge algorithms admits sound and complete axiomatizations that can be derived directly from the rules of the logical theory. We also consider the complexity of the decision problem.
– In Chapter 4, we study what happens when the knowledge algorithm used by an agent is randomized. Handling this case requires an extension of the theory. In this context, it becomes necessary to characterize the information obtained by an agent when a randomized knowledge algorithm gives a positive answer to a query, since there is often some probability that the algorithm is wrong when it answers positively. It certainly seems that if the probability that the algorithm gives the wrong answer is low, it provides very useful information when it answers "Yes" to a query. A positive answer from a randomized algorithm that has a low probability of being wrong is often deemed "good enough". In what sense is this true? We show how to quantify this statement using an appropriate measure of evidence from the literature.

– In Chapter 5, we focus on the notion of evidence that was used in the Chapter 4 to quantify the evidence provided by randomized knowledge algorithms. We develop a logic for reasoning about evidence in a more general setting that essentially views evidence as a function from prior beliefs (before making an observation) to posterior beliefs (after making the observation). We provide a sound and complete axiomatization for the logic, and consider the complexity of the decision problem.

1.2 Security Protocol Analysis

A particularly interesting field of application for formal theories of knowledge is that of reasoning about security. An important aspect of security is confidentiality, either in the form of data that needs to be kept from falling into the wrong hands, or more generally, in the form of controlling who can know particular facts, such as passwords, keys, or identity. It therefore seems reasonable that formal theories of knowledge should shed light on security issues.

In the second part of this dissertation, we focus on a particular aspect of security, namely security protocols. Security protocols mediate the exchange of information between different agents, in order, for example, to exchange a confidential piece of information or to establish the respective identity of the agents. Security protocols often rely on cryptography to exchange messages, including mechanisms for encrypting messages and mechanisms to digitally sign messages with an unforgeable token whose validity is easy to check. As an illustration, consider the following security protocol to authenticate two agents, Alice and Bob:

$$\begin{array}{lrcl} 1. & A \to B & : & \{\!|n_A, A|\!\}_{k_B} \\ 2. & B \to A & : & \{\!|n_A, n_B|\!\}_{k_A} \\ 3. & A \to B & : & \{\!|n_B|\!\}_{k_B}. \end{array}$$

Here, k_A represents the public key of Alice and k_B the public key of Bob, while n_A and n_B are unpredictable values chosen by Alice and Bob, respectively. The notation $A \to B : m$ indicates that agent A sends a message m to agent B, and the notation $\{\!|v|\!\}_k$ indicates the (public key) encryption of value v using key k.

Assume that all agents have access to the public keys k_A and k_B which can be used to encrypt data, but only Alice has the decryption key corresponding to k_A, and only Bob has the decryption key corresponding to k_B. Thus, everyone can encrypt a message with k_A and send it to Alice, but only Alice can decrypt this message, since only she has the corresponding decryption key. The protocol above says that to authenticate two agents, Alice and Bob, Alice first begins by sending a message $\{\!|n_A, A|\!\}_{k_B}$ to Bob containing her name and a value n_A that Alice chooses so as to be unpredictable. Since only Bob has the decryption key corresponding to

k_B, only Bob can decrypt the message. Bob receives this message, decrypts it, and replies to Alice with a message $\{\!|n_A, n_B|\!\}_{k_A}$ containing the value n_A that Alice sent, along with Bob's own unpredictable n_B, all encrypted so that only Alice can decrypt it. When Alice receives that message, if she assumes that Bob is honest and does not reveal n_A to any other agent, then she sees that the message must be from Bob, since only he could have decrypted her original message, extracted n_A, and used it in the reply message. Hence, she knows she is indeed talking to Bob, and that n_B is Bob's value. She then replies to Bob by sending him back his value n_B encrypted again in such a way that only Bob can read it. A similar argument shows that Bob knows he is talking to Alice, since only Alice could have extracted his n_B from his earlier reply, and used it in the message he received. Intuitively, at the end of the exchange, Alice knows that she has been talking to Bob, and vice versa. Therefore, if Alice and Bob are communicating using a fixed channel, then Alice knows that the channel on which she is talking can be used to communicate with Bob, and vice versa. (Assuming, of course, that the agent at each end of the channel is also fixed.)

When faced with a security protocol such as the one above, there are really two tasks involved in formally reasoning about it. First, the properties of the protocol must be specified. Second, the properties must be shown to be satisfied by the protocol. While arguably the latter has received the most attention in the recent literature, the task of specifying security properties is far from having received a satisfactory solution.

To see the need for a good specification language for security properties, consider the kind of properties that arise in the example above. The goal is to establish that Alice has successfully authenticated herself to Bob, and vice versa. But what does that mean exactly? We argued that at the end of the execution of the protocol, Alice knew Bob's value n_B, and Bob knew Alice's value n_A, and moreover both Alice and Bob knew that the other knew their value. We took this to mean that Alice and Bob successfully authenticated themselves to each other. Is that a reasonable definition? Clearly, it only makes sense if the protocol being analyzed actually involves exchanging secret values. Is there a general notion of authentication that can be captured in a specification language? What about protocols that aim at preserving anonymity, or privacy? At the very least, a specification language for security protocols should be expressive enough to capture all of these notions in a natural way.

Determining the security properties that a security protocol should satisfy is an important part of the analysis. Another important part is determining the context in which the protocol is meant to be analysed. Protocols are not analyzed in a vacuum, but rather with respect to an execution context. To see the subtleties involved in taking the context of execution into account, consider the argument above. The

reasoning appears sound, and should prove that Alice is talking to Bob, and vice versa, since only the appropriate agent can decode the exchanged messages at key points of the interaction. But in what context does this reasoning hold? It turns out that Trudy, an "insider" in the system, can fool Bob into thinking he's talking to Alice while he's in fact talking to Trudy. Consider the following interaction, where Alice initiates an interaction with Trudy, and Trudy uses that interaction as an "oracle" to drive her interaction with Bob:

$$\begin{array}{rcll} A \rightarrow T & : & \{\!|n_A, A|\!\}_{k_T} & \\ T(A) \rightarrow B & : & & \{\!|n_A, A|\!\}_{k_B} \\ B \rightarrow T(A) & : & & \{\!|n_A, n_B|\!\}_{k_A} \\ T \rightarrow A & : & \{\!|n_A, n_B|\!\}_{k_A} & \\ A \rightarrow T & : & \{\!|n_B|\!\}_{k_T} & \\ T(A) \rightarrow B & : & & \{\!|n_B|\!\}_{k_B}. \end{array}$$

The first column reports the messages exchanged between Alice and Trudy, while the second column reports the messages exchanged between Trudy masquerading as Alice (written $T(A)$), and Bob. Here, Trudy has managed to convince Bob that he's talking to Alice, when he is not. It requires Trudy to be a known principal of the system, albeit a dishonest one. This attack is simple, and more importantly, does not show up in the informal analysis of the protocol given above. This does not necessarily mean that the initial protocol is flawed. It depends on the context in which the protocol is to be used. In a closed system where every agent known to the other agents is honest, the argument we gave originally holds. However, if dishonest agents are allowed in the system, then the above attack is a possibility.

The goal of security protocol analysis is to develop tools and methods to reason about protocols, highlighting problems such as the above. As the example above illustrates, there are many aspects to this: Who are the principals involved? Who is compromised? What are the capabilities of adversaries? How many instances of the protocol are running concurrently on the system? To illustrate the difficulties, consider the protocol above, corrected in such a way that the attack we just illustrated is no longer possible. The idea is to add the name of Bob to the second message:

$$\begin{array}{rrcl} 1. & A \rightarrow B & : & \{\!|n_A, A|\!\}_{k_B} \\ 2. & B \rightarrow A & : & \{\!|n_A, n_B, B|\!\}_{k_A} \\ 3. & A \rightarrow B & : & \{\!|n_B|\!\}_{k_B}. \end{array}$$

Intuitively, in the attack scenario above, if Alice sees that she gets her second message back and it contains Bob's name instead of Trudy's, with whom she is talking, then she will figure out something is wrong.

Being clear about the context of execution of a protocol is one important aspect

of reasoning about security protocols. Another important aspect is to determine the capabilities of the adversary. In the example above, what can Trudy do? Clearly, if she can "crack" arbitrary encrypted messages, then she can easily fool Alice or Bob. Indeed, the informal analysis relied on the fact that only Bob could decrypt a message encrypted with k_B. What is commonly done in the literature is to give the adversary some very restricted capabilities, such as being able to intercept and reroute messages, compose new messages by concatenation, but only allowing an adversary to decrypt a message if the decryption key is known. While such an abstract adversary is useful (it is sufficient to find the insider attack described above), it is also fairly limited, as there may be contexts where it makes sense to assume the adversary has more refined capabilities; for example, there may be properties of the encryption that the adversary can use without cracking the encryption.

There are many approaches to reasoning about security protocol analysis, which we review in Chapter 6. These approaches can be classified into broad categories, depending on how exactly they approach the protocol analysis problem. Some approaches are based on standard techniques for analyzing software systems, by focussing on the modeling of the system and the resulting properties. Other approaches are based on the theory of programming languages, and focus on compositional ways of representing the protocols so that the analysis can be done directly on the protocol text. Finally, other approaches are more in the spirit of logic-based verification, in that they offer a precise specification language that is given a formal semantics in terms of the protocols. Each class of approaches makes decisions as to the points raised above: how to model the protocol, how to specify properties, and how to verify that these properties hold. Most importantly, however, few approaches provide a way to model different capabilities of adversaries. The few that are flexible enough to provide such a facility do not provide a suitable specification language. Is it possible to develop a framework for modeling and reasoning about security protocols, where adversaries can be are defined in a flexible way, and that supports an expressive and natural specification language for security properties?

In the second part of this dissertation, we apply the framework developed in the first part of the dissertation to the problem of reasoning about security protocols, taking into account the aspects highlighted above. The goal is to derive a logic-based specification language suitable for capturing security properties, with a clear semantics ground in well-understood and intuitive models, that moreover provides enough flexibility to capture various capabilities of adversaries. The particular contributions of this second part are as follows.

– In Chapter 7, we describe a formal framework to model security protocols, a minor specialization of existing models from the distributed computing literature. Our models are simply dynamic versions of the structures studied in the first

part of the dissertation, and they allow for the clean expression of knowledge-theoretic concepts, which underlies the specification of security notions. We introduce a programming language for writing protocols, and show how it is used to generate models. We finally compare our models to a popular class of models in the literature, strand spaces, and show that they are at least as expressive as strand space models. Thus, our models can be used to model security protocols at least at the level of expressiveness of other approaches.

– In Chapter 8, we introduce a formal logic for reasoning about security properties of the models described in the previous chapter. In keeping with the observation above that most security notions are really epistemic notions, our logic is a logic of knowledge. The capabilities of adversaries can be nicely captured using a knowledge algorithm and thus the knowledge of an adversary can be expressed by the algorithmic knowledge studied in the first part of this dissertation. We show that this can be used to express in a natural way many of the adversaries studied in the literature.
– In Chapter 9, we examine in more detail some of the more interesting notions that arise in security protocol analysis, using the logic of Chapter 8. One notion central to security protocol analysis is that of nonces (the unpredictable values n_A and n_B in the protocol above). The unpredictability of nonces can best be understood and modeled using an epistemic language. Additionally, it is possible to encode many of the higher-level security operators that have been advocated in the literature using a language with well-understood and well-studied operators for knowledge, time, and probability. This provides evidence that the logic introduced in Chapter 8 supplies a reasonable foundation on which to base security protocol analysis.

1.3 Remarks

We assume the reader has a basic knowledge of logic. We assume exposure to complexity theory, including the fact that the satisfiability problem for propositional logic is NP-complete. For Chapter 4 and beyond, we assume a basic knowledge of probability theory, as well as exposure to randomized algorithms. For the second part of this dissertation, we assume a passing familiarity with basic cryptographic concepts, such as shared-key and public-key cryptography.

Keeping in mind these assumptions, every attempt has been made to make this dissertation as self-contained as possible. To avoid distracting the reader, bibliographic information and precise relationships with related work have been relegated to the end of every chapter. The proofs of the technical results appear in Appendix B.

Most of the core work in this dissertation is the result of collaborations. Specifically, Chapters 4–5 and 7–8 are joint work with Joseph Halpern. Chapter 9 is joint work with Joseph Halpern and Ron van der Meyden.

Notes

The branch of philosophy that studies the origin, structure, methods and validity of knowledge is epistemology. For moderns accounts and overviews of contemporary theories, see Hamlyn [1970] and Pollock and Cruz [1999].

The cheating husbands puzzle (variously known as the muddy children puzzle, the three blind wise men puzzle) was described, among others, by Gamow and Stern [1958]. An analysis in terms of a formal theory of knowledge as described in the next chapter is presented by Halpern and Vardi [1991].

One of the first papers to advocate associating mental qualities to machines is McCarthy [1979]. The issues involved in knowledge representations can be found in [Davis, Shrobe, and Szolovits 1993; Sowa 2000].

A discussion of the role of theories of knowledge in economics can be found in [Brandenburger 1989; Aumann 1999]. They have their source in Aumann's [1976] seminal work on the role of common knowledge in reaching agreement. Game theory originated with Von Neumann and Morgenstern [1947]. See [Fudenberg and Tirole 1991] for a modern introduction.

The use of formal theories of knowledge to specify and reason about distributed computation was advocated in a number of early papers [Halpern 1990; Dwork and Moses 1990; Moses and Tuttle 1988; Halpern and Zuck 1992]. See Fagin et al. [1995] for more references. Algorithmic knowledge was introduced by Halpern, Moses, and Vardi [1994]. Kyburg [1983] gives a good overview of the literature on evidence.

The protocol in Section 1.2 is due to Needham and Schroeder [1978]. The insider attack and the fix were discovered by Lowe [1995].

A good modern introduction to logic is Enderton [1972]. A good introduction to complexity theory is Papadimitriou [1994]. The NP-completeness of propositional logic was first proved by Cook [1971]. Good introductions to probability theory include Feller [1957] and Billingsley [1995]. Randomized algorithms are described by Motwani and Raghavan [1995]. Stinson [1995] and Schneier [1996] give excellent overviews of cryptography.

Part I

A Theory of Resource-Bounded Knowledge

2
Algorithmic Knowledge

In the first part of this dissertation, we study a formalism to reason about resource-bounded knowledge. In this chapter, we review the intuitions underlying the classical approach to knowledge, and describe various ways in which it can be extended to take into account resource bounds. We highlight the particular approach we use in the remainder of this dissertation. This chapter is mostly review of existing literature.

Note that there is a strong philosophical component to any study of logics of knowledge (also known as *epistemic logics*). While this is an intriguing topic—to develop a theory of knowledge that captures in a logic the features of what might be termed "human knowledge"—this will not be our aim in this work. The intent is to focus on epistemic logic as a specification language for systems. This leads to a particular set of desiderata, distinct from what one might expect from a logic of human knowledge. (We do consider some philosophical implications of the approach described in this chapter to the problem of human knowledge in Appendix A.)

2.1 A Model of Knowledge

What does it mean to *know* a fact? The modern approach, due to Jaakko Hintikka, goes something like this. Assume a set W of worlds. Intuitively, each world represents a possible state of affairs in the situation being studied. For example, if the situation consists of tossing a die, we might consider six worlds, one for each way the die can land. (We are implicitly assuming, therefore, that the die cannot land on its edge, or indeed not land at all.) A more complicated situation might actually involve a great many more worlds. If we are interested in a situation where we worry about the weather around the world, then we may have a world where it is cloudy in Edinburgh and raining in Ithaca, a world where it is cloudy in Edinburgh

and sunny in Ithaca, and so on. The number of worlds to consider multiply quickly. Clearly, some facts will be true at some worlds, others will be false.

Given the current (or actual) world, there are a number of worlds that may be considered as possible alternatives to the actual world. Intuitively, these are the worlds that cannot be distinguished from the actual world. Given a world w, we typically refer to the worlds that are indistinguishable from w as the worlds considered possible at w. In the weather example, if we are currently in Edinburgh and we witness the sky is cloudy, then we will only consider a world possible if indeed it says that it is cloudy in Edinburgh. Using this notion, we know a fact at a world w if that fact is true at all the worlds considered possible at w.

This can be formalized as follows. An *epistemic frame* $E = (W, \mathcal{K})$ consists of a set W of possible worlds (or states), and a binary relation $\mathcal{K}$ such that $(w, w') \in \mathcal{K}$ if the agent considers w' possible at world w. (That is, if the agent considers that w' is a possible alternative to the actual world w.) It is useful to write $\mathcal{K}$ as though it were a function, as $\mathcal{K}(w) = \{w' \in W \mid (w, w') \in \mathcal{K}\}$. A *fact* in E can be understood as a set of worlds, intuitively, the set of worlds where that fact is true. Hence, the fact "It is raining in Ithaca" can be identified with the set of worlds where it is raining in Ithaca. Following the intuition above, the agent knows a fact F at a world w, if $\mathcal{K}(w) \subseteq F$: at every world the agent considers possible at w, the fact is true.

This definition is quite general. For one, we have not put any restrictions on the worlds an agent considers possible. In general, restrictions on knowledge will amount to properties of the relation $\mathcal{K}$. Some of those are rather standard, with equally standard interpretations. For instance, one restriction could be that the actual world is always considered possible, i.e., $(w, w) \in \mathcal{K}$; in other words, $\mathcal{K}$ is reflexive. Similarly, $\mathcal{K}$ may be transitive, so that $(w_1, w_2) \in \mathcal{K}$ and $(w_2, w_3) \in \mathcal{K}$ imply that $(w_1, w_3) \in \mathcal{K}$, and so on.[1]

There is at least one feature of knowledge in epistemic frames that does *not* depend on the properties of the $\mathcal{K}$ relation. Consider a fact F along with a fact G that is a "consequence" of F. Formally, this simply means that $F \subseteq G$: whenever F holds, G holds as well. It follows directly from the definition of knowledge that an agent knowing fact F also knows fact G. In this sense, agents are very powerful reasoners, knowing all the consequences of the facts they know. We return to this observation in Section 2.3.

[1] One of these properties, reflexivity, is in fact the property that differentiates knowledge from belief, at least according to philosophers: knowledge has the property that if you know a fact, that fact is true. In contrast, it is possible to believe a fact that happens to be false.

2.2 Reasoning about Knowledge

We now define a formal logic for reasoning about the properties of epistemic frames. The starting point is propositional logic. Propositional logic can be understood as a formal system for reasoning about a particular world. Assume a set $\Phi_0 = \{p_1, p_2, \dots\}$ of primitive propositions; each primitive proposition represents a primitive "fact", such as "the door is closed", or "message m has been sent". The syntax of $\mathcal{L}^{\mathrm{K}}(\Phi_0)$ is obtained by starting with a set Φ_0 of primitive propositions, and closing off by forming conjunctions of formulas ($\varphi_1 \wedge \varphi_2$), negations of formulas ($\neg\varphi$), and knowledge formulas ($K\varphi$). The formula $K\varphi$ intuitively reads "the agent knows φ". The remaining logical operators, such as disjunction, implication, equivalence, are defined as abbreviations for more complex formulas. The disjunction $\varphi_1 \vee \varphi_2$ is taken to be an abbreviation for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, the implication $\varphi_1 \Rightarrow \varphi_2$ an abbreviation for $\neg\varphi_1 \vee \varphi_2$, and the equivalence $\varphi_1 \Leftrightarrow \varphi_2$ an abbreviation for $(\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$. The formula $true$ is taken to be an arbitrary but fixed tautology of the logic, and $false$ is an abbreviation for $\neg true$. When the set of primitive propositions is not relevant, we often simply write $\mathcal{L}^{\mathrm{K}}$ instead of $\mathcal{L}^{\mathrm{K}}(\Phi_0)$. (This convention will hold for most logics throughout this dissertation.)

The ideas of the previous section can be used to assign a semantics to this logic. More precisely, we can understand the knowledge of a fact (here understood as a formula) at a world w as the fact being true at all worlds considered possible at w. This can be turned into a semantics for formula of $\mathcal{L}^{\mathrm{K}}$ by starting with an epistemic frame and adding an interpretation that assigning a truth value to every primitive proposition at every world of the frame. More formally, an *epistemic structure* (also known as Kripke structure) is a tuple $M = (W, \mathcal{K}, \pi)$ where $(W, \mathcal{K})$ is an epistemic frame and π is an interpretation for Φ_0 in W, that is, for every world $w \in W$ and primitive proposition $p \in \Phi_0$, $\pi(w)(p) \in \{\textbf{true}, \textbf{false}\}$.

Define a formula φ of $\mathcal{L}^{\mathrm{K}}(\Phi_0)$ to be true (or satisfied) in a world w of the structure M, written $(M, w) \models \varphi$, according to the following inductive rules:

$(M, w) \models p$ if $\pi(w)(p) = \textbf{true}$
$(M, w) \models \neg\varphi$ if $(M, w) \not\models \varphi$
$(M, w) \models \varphi \wedge \psi$ if $(M, w) \models \varphi$ and $(M, w) \models \psi$
$(M, w) \models K\varphi$ if $(M, w') \models \varphi$ for all $w' \in \mathcal{K}(w)$.

A formula φ is *valid in* M, written $M \models \varphi$, if $(M, w) \models \varphi$ for all $w \in W$. A formula is *valid*, written $\models \varphi$, if $M \models \varphi$ for all models M. If φ is valid, φ is often called a tautology. Conversely, φ is *satisfiable* if there is a structure M and a state w of M such that $(M, w) \models \varphi$.

Given the above semantics for $\mathcal{L}^{\mathrm{K}}$, it is straightforward to verify that some of the properties alluded to in the previous section hold in this system. First, an agent

knows all "consequences" of his knowledge. This can be made precise via the following formula. If an agent knows a formula φ and knows that $\varphi \Rightarrow \psi$, then at each state the agent considers possible, both φ and $\varphi \Rightarrow \psi$ hold, and hence ψ holds. In other words, the agent knows ψ. It follows that

$$K\varphi \wedge K(\varphi \Rightarrow \psi) \Rightarrow K\psi$$

is valid in epistemic structures. This property is often called the *Distribution Axiom*.

This property seems to imply that agents are powerful reasoners. The following observation provides further evidence. If a formula φ is valid in an epistemic structure M, then φ holds at all the states of M, including of course all the states an agent consider possible. Therefore, at every state of M, the agent must know φ. Hence, the following *Rule of Knowledge Generalization* holds: for all epistemic structures M,

$$\text{if } M \models \varphi \text{ then } M \models K\varphi.$$

As a consequence, if φ is valid, then $K\varphi$ must be valid as well. Note that this does not say that $\varphi \Rightarrow K\varphi$ is valid. This would require that for all M and all $w \in W$, $(M, w) \models \varphi$ implies $(M, w) \models K\varphi$, which is clearly false in general. In a sense, the Rule of Knowledge Generalization says that an agent knows all the facts that are necessarily true.

It turns out that additional properties hold in epistemic structures where $\mathcal{K}$ has particular properties. The following formula is valid in any structure where $\mathcal{K}$ is transitive:

$$K\varphi \Rightarrow KK\varphi.$$

This is typically called the *Positive Introspection Axiom*: an agent knows that he knows what he knows. On the other hand, in a structure where $\mathcal{K}$ is Euclidean (that is, if $(w_1, w_2) \in \mathcal{K}$ and $(w_1, w_3) \in \mathcal{K}$, then $(w_2, w_3) \in \mathcal{K}$), the following formula is valid:

$$\neg K\varphi \Rightarrow K\neg K\varphi.$$

This is the *Negative Introspection Axiom*. In an epistemic structure where $\mathcal{K}$ is serial, that is, an agent always considers at least one world possible, then

$$\neg K\mathit{false}$$

must be valid. It is probably easier to see the converse: if $K\mathit{false}$ holds at a world of M, it must be that there is no world the agent considers possible, since there is no world such that $(M, w) \models \mathit{false}$.

Finally, in a structure where $\mathcal{K}$ is reflexive, that is, where the agent always considers the actual world possible, the formula

$$K\varphi \Rightarrow \varphi$$

is valid. This is called the *Knowledge Axiom*.

In the remainder of this dissertation, we focus, for the interpretation of knowledge, on structures where the relation $\mathcal{K}$ is an equivalence relation. Roughly speaking, this is because we assume a particular structure to the worlds, where the agent gets to "observe" part of the world, and considers two worlds possible if they yield the same observations. Henceforth, we assume that the relation $\mathcal{K}$ in an epistemic structure $M = (W, \mathcal{K}, \pi)$ is an equivalence relation. Let $\mathcal{M}^{\mathrm{K}}$ be the class of all such epistemic structures.

We can establish that the above properties of knowledge, in a sense, are all the relevant properties of knowledge in those kind of structures. Any other property can be derived from these. This can be made formal using an axiomatization. An *axiomatization* AX for a logic is a set of formulas (the axioms) and inference rules of the logic.[2] Given a set of formulas F, φ is *provable* from the set F with respect to the axiomatization AX, written $F \vdash_{\mathbf{AX}} \varphi$, if there is a derivation of φ from the axioms and inference rules of AX and the formulas in F. This is often written $F \vdash \varphi$ when the axiomatization AX is understood. Formally, a *derivation* of φ is a sequence of formulas $\varphi_1, \ldots, \varphi_n$ such that φ_n is φ, and for each i, φ_i is either an axiom of AX, a formula of F, or obtained from $\varphi_1, \ldots, \varphi_{i-1}$ via an inference rule of AX. When F is empty, φ is said to be *provable*, written $\vdash \varphi$.

An axiomatization AX is *sound* if every provable formula is valid, that is, if $\vdash \varphi$ implies $\models \varphi$ for all φ. This is a basic requirement of an axiomatization, indicating that it can only prove true facts. An axiomatization AX is *complete* if every valid formula is provable, that is, if $\models \varphi$ implies $\vdash \varphi$ for all φ. Of course, for every logic, there is a trivial sound and complete axiomatization, obtained by taking *all* the valid formulas of the logic as axioms. This is uninteresting in general, since it does not explain why formulas are valid. In order to be of interest, an axiomatization should be either finite, or at least finitely described. It is possible to relax the restrictions somewhat, and allow so-called axiom schemes as axioms, that is, templates for axioms. Given an axiom scheme, replacing the metavariables in the axiom scheme by actual formulas yields an axiom. For ease of exposition, we will keep on referring to axiom schemes simply as axioms.

Since the logic $\mathcal{L}^{\mathrm{K}}$ includes propositional reasoning, any axiomatization pur-

[2] An *inference rule* says that when particular formulas are valid, others are valid as well. By abuse of terminology, we will often refer to both axioms and inference rules as simply axioms. The form of an axiom will always make clear whether it is a *bona fide* axiom or an inference rule.

ported to be sound and complete must somehow deal with propositional reasoning. The following axiom and inference rule take care of this:

Prop. All instances of propositional tautologies.
MP. From φ and $\varphi \Rightarrow \psi$ infer ψ.

Axiom Prop stands for all instances of tautologies of propositional logic, where the instances are obtained by instantiating arbitrary $\mathcal{L}^{\mathrm{K}}(\Phi_0)$ formulas. For instance, since $f \vee \neg f$ is a propositional tautology, Prop includes $K\varphi \vee \neg K\varphi$ as an instance. Axiom MP is simply *Modus Ponens*. Axiom Prop can be replaced by the following axioms:

P1. $\varphi \Rightarrow (\psi \Rightarrow \varphi)$.
P2. $(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow \varphi_3)) \Rightarrow ((\varphi_1 \Rightarrow \varphi_2) \Rightarrow (\varphi_1 \Rightarrow \varphi_3))$.

The following axioms capture the properties of knowledge:

K1. $K\varphi \wedge K(\varphi \Rightarrow \psi) \Rightarrow K\psi$.
K2. From φ infer $K\varphi$.
K3. $K\varphi \Rightarrow \varphi$.
K4. $K\varphi \Rightarrow KK\varphi$.
K5. $\neg K\varphi \Rightarrow K\neg K\varphi$.

Let AX^{K} be the axiomatization $\{\mathrm{Prop}, \mathrm{MP}, \mathrm{K1}\text{–}\mathrm{K5}\}$. ($\mathrm{AX}^{\mathrm{K}}$ has been called S5 in the literature, from a terminology on modal logics originally going back to the philosopher Clarence Irving Lewis.)

Theorem 2.1. AX^{K} *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{K}}$ *with respect to the class of epistemic structures* $\mathcal{M}^K$.

A more general result holds when one considers epistemic structure with different properties on the relation $\mathcal{K}$. For instance, there is an axiomatization that is sound and complete for $\mathcal{L}^{\mathrm{K}}$ with respect to epistemic structures where $\mathcal{K}$ is reflexive. Not all such properties can be captured by an axiomatization, however. The classical example of this is irreflexivity, which cannot be captured by a formula of modal logic.

2.3 The Problem of Logical Omniscience

A point we have already raised in the previous two sections when describing the particular interpretation of knowledge at hand is that agents are powerful reasoners.

If one looks at the axiomatization of the previous section, the culprit appears as the axiom

$$K\varphi \wedge K(\varphi \Rightarrow \psi) \Rightarrow K\psi.$$

One reading of this axiom (in conjunction with the Knowledge Generalization axiom) is that an agent knows all the logical consequences of his knowledge. Having agreed to the intuitive notion of knowledge in Section 2.1, the formalization of that intuition forces this property. It is of course possible to debate whether the notion of knowledge in Section 2.1 is reasonable. It turns out that in some cases, it is reasonable; in other cases, it is not. The above notion of knowledge has been used successfully in the distributed systems literature to study properties of distributed protocols—for example, that a protocol correctly implement a specification of the sort: "a process repeatedly sends a message until it knows that the intended receiver has received it". Here, the knowledge is knowledge ascribed to the process by the designer of the system; the process simply executes its program, and the correctness of the implementation corresponds to the process behaving as though it "knew" the particular fact. This is a form of *implicit knowledge*: knowledge that is implicit in the situation and the environment in which the process is executing. More importantly, it is not knowledge that the process explicitly bases its actions on. In the above example, the process will repeatedly send a message until it enters a situation where it stops. It is up to the designer to ensure that this situation corresponds to the state where the intended receiver has received the message. There is no question of the process stopping to decide whether or not it knows that particular fact. (On the other hand, a reasonable way to implement an process might be for it to do exactly that.)

Implicit knowledge is a useful analysis and design tool. On the other hand, it is easy to imagine situations where it is reasonable for an agent to decide whether he knows a particular fact. It is clear that the above notion of implicit knowledge will not work in such a setting—it is simply unreasonable for agents to be able to derive all consequences of their knowledge, or indeed, for them to know all tautologies.

This problem with implicit knowledge is not new. It is called the problem of *logical omniscience*. Using implicit knowledge to reason about the knowledge of agents leads them to be logically omniscient. While in some situations this is a perfectly reasonable assumption, it ceases to be one when the agents are meant to act on their knowledge. What is needed for those cases is a notion of explicit knowledge: a way for the agent to state what he explicitly knows, knowledge on which he can base his actions. Ideally, explicit knowledge should not suffer from the logical omniscience problem.

There have been quite a few attempts at getting around the logicial omniscience problem in the literature. We will only survey the most relevant here. One way to

circumvent logical omniscience is to posit a number of *impossible worlds*, worlds where the usual "laws" of logic do not hold. For instance, an impossible world w might be such that a primitive proposition p is both true and false, so that $p \wedge \neg p$ is in fact true at world w. There are some difficulties with this approach, not the least of which how to decide, when constructing the model, which impossible worlds to add. Impossible worlds have to break the laws of logic in some way, otherwise tautologies are still known. But which laws do you break?

The impossible-worlds approach is a semantic approach: it modifies the models of epistemic logic by adding new worlds. At the other extreme, there are syntactic approaches based on *awareness*. Circumventing logical omniscience via awareness is based on the intuition that to explicitly know something is to first implicitly know something, and then to be made aware of it. How can this awareness be modeled? The simplest way is to assume that there is a set of formulas at each world, the awareness set, those formulas of which the agent is aware. Say $A\varphi$ is true if φ is a formula in the awareness set of the agent; then, take φ to be explicitly known at a world if $K\varphi \wedge A\varphi$ is true at that world. The advantage of this approach over the impossible-worlds approach is that it is often easier to qualify what an agents knows explicitly in terms of language (formulas) than it is in terms of sets of worlds.

The awareness-based approach captures a nice intuition and underlies many related systems. There is one question that is not resolved by the general framework, however, and it is the question of how to establish which formulas an agent is aware of at every world? Is the agent aware of the same formulas at all the worlds he considers possible? That would seem reasonable. If an agent is aware of a compound formula, is he aware of its subformulas? That would also seem reasonable. However, even if these questions are answered, how do you state that an agent only has limited resources for reasoning?

It turns out that there is an approach related to the awareness approach that has a particular answer to those questions, and that moreover captures nicely the intuition that agents explicitly know facts by computing that they know them, within the resources available to them. This is the approach studied in the remainder of this dissertation.

2.4 Algorithmic Knowledge

The aim is to come up with a notion of explicit knowledge that can be used by an agent to determine whether he knows a particular fact. Appealing to intuition for a moment, how do we determine whether we know a particular fact? Typically, it will either be a fact that we know "offhand", say via a direct observation of the

world around me, or it is a fact that we can somehow derive from other facts at hand. Thus, intuitively, to explicitly know a fact is to have a procedure that says whether or not that fact is true.

This can be formalized using a notion called *algorithmic knowledge*. From a semantic point of view, it requires adding to an epistemic structure (representing the possible worlds and the implicit knowledge of the agents) a *knowledge algorithm* that can answer, given a formula φ and a world, whether or not the agent explicitly knows the formula φ at that world. For the time being, we shall make no assumption on the algorithm, except that it be effectively computable, that it always terminate, and that it returns an answer of "Yes", "No", or "?". Intuitively, "Yes" indicates that, according to the knowledge algorithm, the agent explicitly knows the formula, "No" indicates that the agent does not explicitly know the formula, and "?" indicates that the algorithm does not have enough resources to determine if the agent knows the formula. (It is this last possibility that allows knowledge algorithms to model resource-boundedness.) Let $\mathcal{L}^{\mathrm{KX}}(\Phi_0)$ be the following language for reasoning about algorithmic knowledge. As before, it is obtained by starting from a set of propositions in Φ_0, and closing under negation, conjunction, knowledge formulas $K\varphi$, as well as algorithmic knowledge formulas $X\varphi$, read "the agent can compute that φ is true". As usual, define $\vee$, $\Rightarrow$, $\Leftrightarrow$, and $true$ as abbreviations. Similarly, $true$ is taken to be an arbitrary but fixed tautology of the logic, and set $false$ as $\neg true$.

In order to formalize the above, however, the structure of the worlds deserves more attention. Up until now, the worlds were taken to be abstract elements of a set. Algorithmic knowledge, on the other hand, is very concrete—there is an explicit algorithm in the model that takes as input formulas and worlds. This makes it necessary to agree on what a world is. Going back to the intuition, the procedure for determining if we explicitly know something should not be based on the world as a whole, but rather should be based on what we can observe of the world.

To give a semantics to algorithmic knowledge, start as before with a set of worlds W, and an interpretation π for the primitive propositions at the worlds. Rather than directly postulate a relation $\mathcal{K}$, define a set L of *local states*, and a function $\mathcal{V} : W \to L$ giving, for each world, the view of the agent at that world, that is, his local state. An *algorithmic knowledge structure* is a tuple $M = (W, \mathcal{V}, \pi, \mathtt{A})$ where W is a set of worlds, $\mathcal{V}$ is the view of the world for the agent, π is an interpretation function for a set Φ_0 of primitive propositions to truth values, and $\mathtt{A}$ is a knowledge algorithm. (The set L of local states is left implicit, as it can be recovered from the function $\mathcal{V}$.) The knowledge algorithm takes as input a formula φ and a local state $\ell \in L$ for the agent, and returns one of {"Yes", "No", "?"}. There is no relation $\mathcal{K}$ in M, but it is easy to derive one. Say two worlds are indistinguishable to the agent if they have the same view. Write $w_1 \sim w_2$ if and only if $\mathcal{V}(w_1) = \mathcal{V}(w_2)$. It is not

hard to see that $\sim$ is an equivalence relation on the worlds, so that $(W, \sim, \pi)$ is an epistemic structure. Let $\mathcal{M}^{\mathrm{alg}}$ be the class of all algorithmic knowledge structures.

Following this intuition of M as an epistemic structure, define a formula φ to be true at a world of M, written $(M, w) \models \varphi$, by the following inductive definition:

$(M, w) \models p$ if $\pi(w)(p) =$ **true**
$(M, w) \models \neg\varphi$ if $(M, w) \not\models \varphi$
$(M, w) \models \varphi \wedge \psi$ if $(M, w) \models \varphi$ and $(M, w) \models \psi$
$(M, w) \models K\varphi$ if $(M, w') \models \varphi$ for all $w' \in W$ such that $w' \sim w$
$(M, w) \models X\varphi$ if $\mathtt{A}(\varphi, \mathcal{V}(w)) =$ "Yes"

It is easy to verify that the indistinguishability relation $\sim$ is an equivalence relation. Therefore, according to the discussion of Section 2.2, this notion of knowledge satisfies axioms K1–5.

There is a subtlety about the logic above. By taking $\vee$ and $\Rightarrow$ as abbreviation, any formula containing $\vee$ or $\Rightarrow$ is really a formula containing $\wedge$ and $\neg$. Thus, the agent cannot explicitly distinguish between $\varphi \vee \psi$ and $\neg(\neg\varphi \wedge \neg\psi)$; they are the same formula for him. In other words, $\models X(\varphi \vee \psi) \Leftrightarrow X(\neg(\neg\varphi \wedge \neg\psi))$. This seems to go against the main motivation for explicit knowledge, to ensure that knowledge is not closed under tautologies. One way around this problem is to use a syntax that directly uses $\vee$, $\Rightarrow$, and perhaps other connectives, rather than introducing them through abbreviations. We will not follow this approach in this dissertation.

It is immediate from the above definitions that there is no necessary connection between implicit and explicit knowledge. Indeed, there is nothing constraining the knowledge algorithm. One way to see this is to look at axiomatizations. The only properties of algorithmic knowledge follow from the fact that answers of the knowledge algorithm depend only on the local state of the agent. This translates into the following axiom:

X1. $X\varphi \Rightarrow KX\varphi$.

It is easy to see that in the presence of K1–5, axiom X1 implies that $\neg X\varphi \Rightarrow K\neg X\varphi$ is provable:

1.	$\neg X\varphi \Rightarrow \neg KX\varphi$	$(\mathrm{K3}, \mathrm{Prop})$
2.	$\neg KX\varphi \Rightarrow K\neg KX\varphi$	$(\mathrm{K5})$
3.	$\neg KX\varphi \Rightarrow \neg X\varphi$	$(\mathrm{X1}, \mathrm{Prop})$
4.	$K(\neg KX\varphi \Rightarrow \neg X\varphi)$	$(3, \mathrm{K2})$
5.	$\neg KX\varphi \Rightarrow K\neg X\varphi$	$(2, 4, \mathrm{Prop}, \mathrm{K1})$
6.	$\neg X\varphi \Rightarrow K\neg X\varphi$	$(1, 5, \mathrm{Prop}, \mathrm{MP})$.

Let $\mathrm{AX}^{\mathrm{KX}}$ be the axiomatization AX^{K} augmented with X1.

Theorem 2.2. $\mathrm{AX}^{\mathrm{KX}}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KX}}$ *with respect to algorithmic knowledge structures.*

In general, the most useful knowledge algorithms are those that are "correct", in some sense of the word. There are a few notions of correctness that are relevant. First, it may be the case that the knowledge algorithm is right when it answers "Yes" or "No", dismissing "?" answers. Say a knowledge algorithm $\mathtt{A}$ is *sound* for an agent in an algorithmic knowledge structure M if for all φ in $\mathcal{L}^{\mathrm{KX}}$ and all $w \in W$, $\mathtt{A}(\varphi, \mathcal{V}(w)) =$ "Yes" implies $(M, w) \models K\varphi$, and $\mathtt{A}(\varphi, \mathcal{V}(w)) =$ "No" implies $(M, w) \models \neg K\varphi$. Say a knowledge algorithm is *complete* for an algorithmic knowledge structure M if it always answers either "Yes" or "No"—it never answers "?". Most knowledge algorithms considered in the literature that are sound are not complete. Intuitively, the soundness guarantees that the algorithm, when it returns a result, is correct, but the algorithm is not forced to return a result (which is what captures resource bounds).

The above properties can be naturally relativized to a particular set of formulas one cares about, or to a particular set of worlds. For example, a knowledge algorithm is sound with respect to $\Sigma \subseteq \mathcal{L}^{\mathrm{KX}}$ and $W' \subseteq W$ in M, if for all φ in Σ and all $w \in W'$, $\mathtt{A}(\varphi, \mathcal{V}(w)) =$ "Yes" implies $(M, w) \models K\varphi$, and $\mathtt{A}(\varphi, \mathcal{V}(w)) =$ "No" implies $(M, w) \models \neg K\varphi$.

Unfortunately, soundness and completeness properties of knowledge algorithms cannot be formalized within the logic $\mathcal{L}^{\mathrm{KX}}$. Intuitively, $\mathcal{L}^{\mathrm{KX}}$ cannot distinguish between a knowledge algorithm answering "No" and a knowledge algorithm answering "?"; both answers result in lack of algorithmic knowledge. Soundness and completeness require this distinction. This distinction can be captured by introducing a new operator in the logic. Let $\mathcal{L}^{\mathrm{KXD}}(\Phi_0)$ be the language defined just as $\mathcal{L}^{\mathrm{KX}}(\Phi_0)$, with an additional modal operator $D\varphi$ that is true if the algorithm is *definite* about φ, that is, if the algorithm answers either "Yes" or "No" (not "?") to a query φ. Semantically, this can be captured by the following rule:

$(M, w) \models D\varphi$ if $\mathtt{A}(\varphi, \mathcal{V}(w)) \in \{$"Yes", "No"$\}$.

An alternate approach to using a $D\varphi$ operator to capture the distinction between "No" and "?" is to introduce an operator $\overline{X}\varphi$, true if and only if the algorithm answers "No". The formula $D\varphi$ can then be taken as an abbreviation for $X\varphi \vee \overline{X}\varphi$.

The following axioms are the only axioms needed to account for $D\varphi$ in the axiomatization:

X2. $X\varphi \Rightarrow D\varphi$.
X3. $D\varphi \Rightarrow KD\varphi$.

As in the case of axiom X1, in the presence of K1–5, X3 implies that $\neg D\varphi \Rightarrow K\neg D\varphi$ is provable. Let $\mathrm{AX}^{\mathrm{KXD}}$ be the axiomatization $\mathrm{AX}^{\mathrm{KX}}$ augmented with axioms X2 and X3.

Theorem 2.3. $\mathrm{AX}^{\mathrm{KXD}}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures.*

To capture soundness and completeness of knowledge algorithms in this extended logic, consider the following axioms:

X4. $X\varphi \Rightarrow K\varphi$.
X5. $D\varphi \wedge \neg X\varphi \Rightarrow \neg K\varphi$.

Axiom X4 simply says that a knowledge algorithm answering "Yes" is correct, while axiom X5 says that a knowledge algorithm answering "No" (this is what $D\varphi \wedge \neg X\varphi$ expresses) is also correct.

Theorem 2.4. $\mathrm{AX}^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}\}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures with sound algorithms.*

Another axiom is needed to account for complete knowledge algorithms. A complete knowledge algorithm is characterized by the fact that it always answers either "Yes" or "No". This is exactly what the D operator is meant to capture:

X6. $D\varphi$.

This axiom simply says that for any formula φ, the algorithm says "Yes" or "No" when queried for φ. This exactly captures completeness of the algorithm.

Theorem 2.5. $\mathrm{AX}^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}, \mathrm{X6}\}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures with sound and complete algorithms.*

It is immediate that for a sound and complete algorithm, the formula $X\varphi \Leftrightarrow K\varphi$ holds, meaning that sound and complete algorithms capture implicit knowledge. This indicates that sound and complete algorithms are hard to come by, and inefficient when they do exist. We will mostly be concerned with sound algorithms, those that are correct when they return a result. In Chapter 4, we consider knowledge algorithms that are not quite sound, because they have a small probability of error. Algorithmic knowledge in the presence of sound algorithms can be seen as an instance of awareness, as defined in Section 2.3.

It is clear that the notion of algorithmic knowledge is quite general. What is not so clear is that this notion is actually interesting; the risk is that it is *too* general.

There is one sense in which this generality is useful: it allows us to capture particular forms of explicit knowledge by putting *restrictions* on either the form of knowledge algorithms or their properties. Thus, we can get insight into a particular form of explicit knowledge by examining the corresponding knowledge algorithms. In the next chapter, for instance, we will see knowledge algorithms that arise out of deductive systems.

Restricting the form of knowledge algorithms often immediately translates into axioms that are sound with respect to algorithmic knowledge structures that use such algorithms. This is hardly surprising, but this fact is used to characterize classes of knowledge algorithms in Chapter 3. By way of example, consider the behaviour of a knowledge algorithm with respect to negation. There is of course nothing in the definition of a knowledge algorithm that says that the answers of the knowledge algorithm to queries φ and $\neg\varphi$ need to be related. However, there is a natural way to define the behaviour of a knowledge algorithm on negated formulas. A strategy to evaluate $\mathtt{A}_i(\neg\varphi, \ell)$ is to evaluate $\mathtt{A}_i(\varphi, \ell)$, and return the negation of the result. There is a choice to be made in the case when the $\mathtt{A}_i$ returns "?" to the query for φ. One possibility is to return "?" to the query for $\neg\varphi$ when the query for φ returns "?"; another possibility is to return "Yes" if the query for φ returns "?". (Arguably, the former is more intuitive than the latter.) Say that a knowledge algorithm $\mathtt{A}$ *weakly respects negation* if for all local states ℓ,

$$\mathtt{A}(\neg\varphi, \ell) = \begin{cases} \text{"Yes"} & \text{if } \mathtt{A}(\varphi, \ell) = \text{"No"} \\ \text{"No"} & \text{if } \mathtt{A}(\varphi, \ell) = \text{"Yes"} \\ \text{"?"} & \text{if } \mathtt{A}(\varphi, \ell) = \text{"?"}. \end{cases}$$

Similarly, say that a knowledge algorithm $\mathtt{A}$ *strongly respects negation* if for all local states ℓ,

$$\mathtt{A}(\neg\varphi, \ell) = \begin{cases} \text{"Yes"} & \text{if } \mathtt{A}(\varphi, \ell) \neq \text{"Yes"} \\ \text{"No"} & \text{if } \mathtt{A}(\varphi, \ell) = \text{"Yes"}. \end{cases}$$

Theorem 2.6. *Let $M = (W, \mathcal{V}, \pi, \mathtt{A})$ be an algorithmic knowledge structure. If $\mathtt{A}$ weakly respects negation, then $M \models X\varphi \Rightarrow \neg X\neg\varphi$. If $\mathtt{A}$ strongly respects negation, then $M \models X\varphi \Leftrightarrow \neg X\neg\varphi$.*

Similarly, say that a knowledge algorithm $\mathtt{A}$ respects conjunction if for all local states $\ell \in L$ and all formulas $\varphi \in \mathcal{L}^{\mathrm{KX}}$, $\mathtt{A}(\varphi \wedge \psi, \ell) =$ "Yes" if and only if $\mathtt{A}(\varphi, \ell) =$ "Yes" and $\mathtt{A}(\psi, \ell) =$ "Yes". This leads to the valid formula $X(\varphi \wedge \psi) \Leftrightarrow X\varphi \wedge X\psi$.

2.5 Multiple Agents

The framework described in the previous sections extends in a straightforward way to multiple agents. The aim is to reason about the knowledge of different agents with respect to each other—for instance, we may want to express the fact that "Bob knows that Alice knows that he sent a message containing 42". This can be achieved by adding knowledge operators to the logic that are indexed by agents, and by providing each agent with a binary relation describing his possible worlds. The notion of algorithmic knowledge can also be extended to multiple agents by providing each agent with a knowledge algorithm.

Let the agents be named $1, \ldots, n$. The epistemic logic for n agents $\mathcal{L}_n^{\mathrm{K}}(\Phi_0)$ is defined just like $\mathcal{L}^{\mathrm{K}}(\Phi_0)$. The difference is that rather than having a single operator $K\varphi$, there is a family of modal operators $K_i\varphi$, read "agent i knows φ". As usual, take $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$ as abbreviations. To interpret this logic, consider *epistemic structures for n agents* $M = (W, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$, where W is a set of worlds, $\mathcal{K}_1, \ldots, \mathcal{K}_n$ are binary relations on W, one per agent, and π is an interpretation for the primitive propositions at every world. The semantics of $\mathcal{L}_n^{\mathrm{K}}$ is given by the obvious generalization of the semantics of $\mathcal{L}^{\mathrm{K}}$.

To reason about algorithmic knowledge with multiple agents, define the logic $\mathcal{L}_n^{\mathrm{KX}}(\Phi_0)$ by adding a family of modal operators $X_i\varphi$ to $\mathcal{L}_n^{\mathrm{K}}(\Phi_0)$. As expected, the formula $X_i\varphi$ is read "agent i can compute that φ is true". Similarly, the logic $\mathcal{L}_n^{\mathrm{KXD}}(\Phi_0)$ is obtained by adding the family of modal operators $D_i\varphi$. The formula $D_i\varphi$ is read "agent i is definite about φ".

Formally, define a *algorithmic knowledge structure for n agents* (simply called an algorithmic knowledge structure when there is no ambiguity) to be a tuple $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$, where W is a set of worlds, $\mathcal{V}_1, \ldots, \mathcal{V}_n$ are the views of each agents, π is an interpretation for the primitive propositions in Φ_0, and $\mathtt{A}_1, \ldots, \mathtt{A}_n$ are the knowledge algorithms of each agent. As in the single agent case, each view function induces an equivalence relation on the set of worlds; formally, $w_1 \sim_i w_2$ if and only if $\mathcal{V}_i(w_1) = \mathcal{V}_i(w_2)$. Hence, $w_1 \sim_i w_2$ if w_1 and w_2 are indistinguishable for agent i.

The satisfaction relation $(M, w) \models \varphi$ is defined in the obvious way, by analogy with the single agent case:

$(M, w) \models p$ if $\pi(w)(p) =$ **true**
$(M, w) \models \neg\varphi$ if $(M, w) \not\models \varphi$
$(M, w) \models \varphi \wedge \psi$ if $(M, w) \models \varphi$ and $(M, w) \models \psi$
$(M, w) \models K_i\varphi$ if $(M, w') \models \varphi$ for all $w' \in W$ such that $w' \sim_i w$
$(M, w) \models X_i\varphi$ if $\mathtt{A}_i(\varphi, \mathcal{V}_i(w)) =$ "Yes".

For $\mathcal{L}_n^{\mathrm{KXD}}$, the following rule is used to interpret $D\varphi$:

$(M, w) \models D_i\varphi$ if $\mathtt{A}_i(\varphi, \mathcal{V}_i(w)) \in \{\text{“Yes”}, \text{“No”}\}$.

As far as axiomatizations are concerned, it is not hard to see that since the various K_i and X_i operators do not interfere with each other, the axioms compose in a straightforward way. The results of Section 2.2 can be lifted immediately, by replacing references to $K\varphi$ in axioms K1–5 by $K_i\varphi$. For instance, K1 becomes $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$, for every agent i. In a similar way, the axiom X1 simply becomes $X_i\varphi \Rightarrow K_i X_i\varphi$. We will continue to refer to these axioms as K1–5 and X1–6, the context making it clear whether we are talking about the single agent setting or the multiple agents setting. Let $\mathrm{AX}_n^{\mathrm{KX}}$ and $\mathrm{AX}_n^{\mathrm{KXD}}$ be the axiomatizations corresponding to $\mathrm{AX}^{\mathrm{KX}}$ and $\mathrm{AX}^{\mathrm{KXD}}$ for multiple agents. The equivalent of Theorems 2.2, 2.3, 2.4 and 2.5 hold in the multiple agent setting.

Theorem 2.7.

(a) $\mathrm{AX}_n^{\mathrm{KX}}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KX}}$ *with respect to algorithmic knowledge structures for* n *agents.*

(b) $\mathrm{AX}_n^{\mathrm{KXD}}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *with respect to algorithmic knowledge structures for* n *agents.*

(c) $\mathrm{AX}_n^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}\}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *over algorithmic knowledge structures for* n *agents with sound algorithms.*

(d) $\mathrm{AX}_n^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}, \mathrm{X6}\}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *over algorithmic knowledge structures for* n *agents with sound and complete algorithms.*

It is interesting to examine some properties of knowledge in the presence of multiple agents. For instance, $K_1K_2\varphi$ implies immediately that $K_1\varphi$, that is, knowing that someone else knows something implies knowing that something oneself.

The interaction between the knowledge algorithms of the various agents is especially interesting. Since we assumed that there is a single algorithm per agent in the models, in a precise sense, the algorithms used by the agents are common knowledge. What does this common knowledge indicate? At every point, if an agent knows the input, he knows the outcome of the algorithm. More precisely, if agent i uses an algorithm that replies “Yes” to queries ψ when φ is true, then $K_j(\varphi \Rightarrow X_i\psi)$ holds for all agents j. All this says is that there is no uncertainty on the part of the agents as to the explicit knowledge of other agents once the data they have is known.

A more general framework would allow agents to have different knowledge algorithms at different worlds. This would permit the modeling of agents that learn by essentially updating their knowledge algorithms. We focus on the simpler setting, studying static structures with static algorithms. In the second part of this

dissertation, we extend the framework to dynamic systems, but still consider static knowledge algorithms.

2.6 Decision Procedures

What is the complexity of the various decision procedures for the logics described in the previous sections? In a precise sense, the complexity depends on the complexity of the corresponding decision procedures for the logic of knowledge $\mathcal{L}_n^{\mathrm{K}}$.

Consider first the model-checking problem. Since the structure is given as an input to the problem, we restrict our attention in this section to *finite* structures. Given a formula φ, let $|\varphi|$ be the the number of symbols needed to write down φ. The following result is well known:

Theorem 2.8. *There is a procedure that runs in time polynomial in $|\varphi| \cdot |W|$ for deciding, given an epistemic structure for n agents $M = (W, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$ and $\varphi \in \mathcal{L}_n^{\mathrm{K}}$, whether $(M, w) \models \varphi$.*

This results extends almost immediately to $\mathcal{L}_n^{\mathrm{KX}}$. Given a knowledge algorithm $\mathtt{A}$, let $f_{\mathtt{A}}$ be a function representing the running time of $\mathtt{A}$. More precisely, let $f_{\mathtt{A}}(n)$ be the time it takes for $\mathtt{A}(\varphi, \ell)$ to execute for any given observation ℓ and an input formula φ of size n. (Intuitively, observations are taken to be atomic and sizeless; the focus is on the complexity of determining the truth of φ.)

Theorem 2.9. *There is a procedure that runs in time polynomial in $|\varphi| \cdot |W| \cdot f(|\varphi|)$ (where $f(n) = \max\{f_{\mathtt{A}_i}(n) \mid i \in \{1, \ldots, n\}\}$) for deciding, given an algorithmic knowledge structure for n agents $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ and $\varphi \in \mathcal{L}_n^{\mathrm{KX}}$, whether $(M, w) \models \varphi$.*

A similar result holds for $\mathcal{L}_n^{\mathrm{KXD}}$.

For satisfiability, a similar phenomenon arises. The complexity of the decision problem for $\mathcal{L}_n^{\mathrm{K}}$ satisfiability is again a well known result.

Theorem 2.10. *The problem of deciding whether a formula φ of $\mathcal{L}_n^{\mathrm{K}}$ is satisfiable in an epistemic structure for n agents is NP-complete if $n = 1$ and PSPACE-complete if $n > 1$.*

Clearly, since $\mathcal{L}_n^{\mathrm{KX}}$ extends $\mathcal{L}_n^{\mathrm{K}}$, satisfiability is at least as hard to decide for $\mathcal{L}_n^{\mathrm{KX}}$ as it is for $\mathcal{L}_n^{\mathrm{K}}$. The interesting thing is that without any restriction on the knowledge algorithms, satisfiability is no harder to decide, since it is trivial to come up with an algorithm that says "Yes" or "No" for the appropriate subformulas present in the formula at hand.

Theorem 2.11. *The problem of deciding whether a formula φ of $\mathcal{L}_n^{\mathrm{KX}}$ is satisfiable in an algorithmic knowledge structure for n agents is NP-complete if $n = 1$ and PSPACE-complete if $n > 1$.*

Notes

The model of knowledge based on possible worlds presented here is originally due to Hintikka [1962]. A modern survey of the use of knowledge and epistemic logic in computer science, with specific application to distributed system is the work of Fagin et al. [1995]. See also Meyer and Hoek [1995].

Classical overviews of modal logic include [Hughes and Cresswell 1972; Goldblatt 1992]. An approachable but still thorough introduction is [Popkorn 1994]. A technical overview focusing on the proof theoretic aspects of modal logics is [Blackburn, Rijke, and Venema 2001]. The terminology S5 is introduced and discussed by Lewis and Langford [1959].

It is of course possible to extend propositional modal logic to first-order, to yield quantified modal logic (or first-order modal logic) [Garson 1984; Fitting and Mendelsohn 1998]. Modal logic has also been extended to the higher-order setting, where it is often known as intensional logic [Gallin 1975].

There are many approaches to providing a semantics for modal logic. The one we describe, due to Kripke [1963], is the most common. Other approaches include algebraic semantics [Lemmon 1966a; Lemmon 1966b] and topological semantics [McKinsey and Tarski 1944].

Axioms P1–2 are given by Popkorn [1994], who proves that, along with Modus Ponens, they form a sound and complete axiomatization for propositional logic. The axiomatization AX^{K} is well-known. A proof of Theorem 2.1 can be found in [Hughes and Cresswell 1972]. There has been a vast amount of work on studying the kind of properties that are expressible via modal logic, under the heading of correspondence theory [Benthem 1984].

The problem of logical omniscience already appears in Hintikka [1962]. The topic has generated much discussion in the philosophical literature. See Stalnaker [1991] for one view. The approaches described in Section 2.3 are attempts to circumvent the problem by modifying the semantics for knowledge. Impossible worlds are introduced by Cresswell [1973], Rantala [1982], and Hintikka [1975]. Awareness was introduced and studied by Fagin and Halpern [1988], and further investigated by Huang and Kwast [1991]. Moreno [1998] gives a good overview of the various approaches for dealing with the logical omniscience problem. The distinction between the two forms of knowledge we called *implicit knowledge* and *explicit knowledge* has long been recognized. In the classical approach in artificial

intelligence known as the *interpreted symbolic structures* approach, knowledge is based on information stored in data structures of the agent [Rosenschein 1985]; this can be seen as an instance of explicit knowledge. In contrast, the *situated automata* approach, which interprets knowledge based on information carried by the state of the machine [Rosenschein 1985], can be seen as an instance of implicit knowledge. Levesque [1984] makes a similar distinction in the context of belief.

The notion of algorithmic knowledge was defined by Halpern, Moses and Vardi [1994], although the approach in this chapter is more restricted, since we assumed a single algorithm per agent. An approach similar in spirit was introduced earlier by Parikh [1987], which he calls *linguistic knowledge*, and which essentially amounts to using sound algorithms. It makes sense to weaken the soundness condition on knowledge algorithms. Algorithmic knowledge generalizes many other approaches, such as step logics [Elgot-Drapkin and Perlis 1990], Levesque's [1984] system, Konolige's [1986] deductive model of belief, and the logic of Duc [2001]. Duc also calls his notion algorithmic knowledge, but takes algorithmic knowledge as being computed over the evolution of a system, rather than being used to examine the local state of the agents.

Berman, Garay, and Perry [1989] implicitly use a particular form of algorithmic knowledge in their analysis of Byzantine agreement. Roughly speaking they allow agents to perform limited tests based on the information they have; agents know only what follows from these limited tests.

Ramanujam [1999] investigates a particular form of knowledge algorithm, where essentially the knowledge algorithm is a model-checking procedure for a logic of implicit knowledge. More specifically, Ramanujam considers, at every world, the part of the model that a particular agent sees (for instance, an agent in a distributed system may only be aware of its immediate neighbors with whom he can communicate) and takes as knowledge algorithm the model-checking procedure for epistemic logic, applied to the submodel generated by the visible worlds.

Theorem 2.8 is straightforward; a proof can be found in Halpern and Moses [Halpern and Moses 1992]. The problem of model checking knowledge is less trivial in the context of dynamic systems; see [Meyden 1998; Meyden and Shilov 1999]. The case $n = 1$ of Theorem 2.11 is due to Ladner [1977], while the proof of the general case $n > 1$ can be found in [Halpern and Moses 1992].

3
Deductive Algorithmic Knowledge

THE generality of the algorithmic knowledge approach, which makes it ideal as a modeling framework, also means that there are no nontrivial properties of algorithmic knowledge proper, unless we consider particular classes of knowledge algorithms. This becomes important when we want to use the framework as a specification language amenable to automatic verification. In that setting, we would like a class of knowledge algorithms that can capture the properties of interest, while still having enough structure to yield a tractable, or at least analyzable, system. This structure typically reveals itself in a class of properties of the corresponding algorithmic knowledge operator, which can be used to study the structures purely deductively.

In this chapter, we study a form of algorithmic knowledge, *deductive algorithmic knowledge*, where the explicit knowledge of agents comes from a logical theory in which the agents perform their reasoning about the facts they know. Many useful forms of explicit knowledge can be formalized in such a logical theory for agents. For instance, Horn theories, which have been used to approximate more general knowledge bases, fit into this framework particularly nicely. Explicit knowledge via a logical theory can be viewed as a form of algorithmic knowledge, where the knowledge algorithm used by an agent is an algorithm that attempts to infer whether a fact is derivable from the deduction rules provided by the agent's logical theory. The highly structured presentation of an agent's logical theory lets us readily derive properties of explicit knowledge in this context. Intuitively, the deduction rules of the logical theory directly translate into logical properties of explicit knowledge.

To motivate the use of logical theories to capture explicit knowledge, consider the following example, which will be analyzed in more detail in the second part of this dissertation. As we saw in Section 1.2, security protocols are analyzed in the presence of an adversary that has a certain number of capabilities to decode the messages he intercepts. There are of course restrictions on the capabilities of

a reasonable adversary. For instance, the adversary may not explicitly know that he has a given message if that message is encrypted using a key that the adversary does not know, despite the fact that he has intercepted the message. There is a now-standard description of capabilities of adversaries that captures these restrictions, due to Danny Dolev and Andrew Yao. Roughly speaking, a Dolev-Yao adversary can decompose messages, or decipher them if he knows the right keys, but cannot otherwise "crack" encrypted messages. The adversary can also construct new messages by concatenating known messages, or encrypting them with a known encryption key. It is natural to formalize a Dolev-Yao adversary using a deductive system that describes what messages the adversary "has" based on the messages he has intercepted, and what messages the adversary can construct.

To reason about such examples, we introduce a modal logic that captures both the implicit knowledge of agents, which is useful for specifications, and the explicit knowledge of agents formalized as a logical theory. We focus in this chapter on the technical properties of the resulting logic, such as axiomatization and complexity of the decision problem. This approach shows that it is possible to combine a standard possible-worlds account of implicit knowledge with a logical theory representing the explicit knowledge of agents, and to reason about both simultaneously. Another advantage is that it is straightforward to extend the framework with probabilities, by taking, for instance, a probability measure over the possible worlds.

3.1 Deductive Systems

We start by defining the framework in which to express the logical theories of the agents, that is, their deductive or inferential powers. Following common practice, we take logical theories as acting over the terms of some term algebra. More precisely, assume a fixed finite signature $\Sigma = (f_1, \ldots, f_n)$, where each f_i is an operation symbol, with arity r_i. Operation symbols of arity 0 are called constants. Assume a countable set $Vars$ of variables. Define the *term algebra* T_Σ as the least set such that $Vars \subseteq T_\Sigma$, and for all $f \in \Sigma$ of arity n, and for all $t_1, \ldots, t_n \in T_\Sigma$, then $f(t_1, \ldots, t_n) \in T_\Sigma$. Intuitively, T_Σ contains all the terms that can be built from the variables, constants, and operations in Σ. A term is a *ground term* if it contains no variables. Let T_Σ^g be the set of ground terms in T_Σ. A ground substitution ρ is a mapping from variables in $Vars$ to ground terms. The application of a ground substitution ρ to a term t, written $\rho(t)$, essentially consists of replacing every variable in t with the ground term corresponding to t in ρ. Clearly, the application of a ground substitution to a term yields a ground term.

A *deductive system* D is a subset of $\wp_{fin}(T_\Sigma) \times T_\Sigma$. (We write $\wp(X)$ for the set

of subsets of X, and $\wp_{fin}(X)$ for the set of finite subsets of X.) A *deduction rule* $(\{t_1, \ldots, t_n\}, t)$ of D is typically written $t_1, \ldots, t_n \triangleright t$, and means that t can be immediately deduced from $t_1, \ldots, t_n$. A deduction of t from a set Γ of terms is a sequence of ground terms $t_1, \ldots, t_n$, such that $t_n = t$, and every t_i is either:

(1) A term $\rho(t')$, for some ground substitution ρ and some term $t' \in \Gamma$;
(2) A term $\rho(t')$, for some ground substitution ρ and some term t' for which there is a deduction rule $t'_{i_1}, \ldots, t'_{i_k} \triangleright t'$ in D such that $\rho(t'_{i_j}) = t_{i_j}$ for all j, and $i_1, \ldots, i_j < i$.

We write $\Gamma \vdash_D t$ if there is a deduction from Γ to t via deduction rules in D. Observe that by definition we have $t \vdash_D t$ for all terms t.

We will only be concerned with deductive systems that are *decidable*, that is, for which the problem of deciding whether a deduction of t from Γ exists is decidable, for a term t and set of terms Γ. Moreover, it should be clear from the definitions that deductive systems are monotonic. Formally, if $\Gamma \vdash_D t$, then $\Gamma' \vdash_D t$ when $\Gamma \subseteq \Gamma'$. Finally, observe that there are no restrictions on the formation of terms. It is possible to assign to each term a sort, and restrict operators to take terms of a given sort only. The resulting *sorted term algebra* can be used as the starting point of the theory in this chapter, with little changes.

Example 3.1. The following deductive system DY over the signature

$$\Sigma = (\mathsf{recv}, \mathsf{has}, \mathsf{encr}, \mathsf{conc}, \mathsf{inv})$$

captures the Dolev-Yao adversary described at the beginning of this chapter. Here, $\mathsf{recv}(m)$ represents the fact that the adversary has received the term m, $\mathsf{has}(m)$ represents the fact that the adversary understands the term m, $\mathsf{encr}(m, k)$ represents the encryption of term m with key k, $\mathsf{conc}(m_1, m_2)$ represents the concatenation of terms m_1 and m_2, and $\mathsf{inv}(k)$ represents the inverse of the key k (that is, the key needed to decrypt messages encrypted with k):

$$\begin{aligned}
\mathsf{recv}(m) &\triangleright \mathsf{has}(m) \\
\mathsf{has}(\mathsf{inv}(k)), \mathsf{has}(\mathsf{encr}(m, k)) &\triangleright \mathsf{has}(m) \\
\mathsf{has}(\mathsf{conc}(m_1, m_2)) &\triangleright \mathsf{has}(m_1) \\
\mathsf{has}(\mathsf{conc}(m_1, m_2)) &\triangleright \mathsf{has}(m_2).
\end{aligned}$$

Assume further that Σ contains constants such as $\mathsf{m}, \mathsf{k}_1, \mathsf{k}_2$. We can therefore derive:

$$\mathsf{recv}(\mathsf{encr}(\mathsf{m}, \mathsf{k}_1)), \mathsf{recv}(\mathsf{encr}(\mathsf{inv}(\mathsf{k}_1), \mathsf{k}_2)), \mathsf{recv}(\mathsf{inv}(\mathsf{k}_2)) \vdash_{\mathrm{DY}} \mathsf{has}(\mathsf{m}).$$

In other words, it is possible for a Dolev-Yao adversary to derive the message

m if he has received m encrypted under a key k_1, which inverse he has received encrypted under a key k_2, which inverse he has received.

To account for constructing new messages, consider the signature Σ' which extends Σ with a unary constructor constr, where $\mathsf{constr}(m)$ represents the fact that the adversary can construct the term m. We can account for this new constructor by adding the following deduction rules to DY:

$$\begin{array}{rcl} \mathsf{has}(m) & \triangleright & \mathsf{constr}(m) \\ \mathsf{constr}(k), \mathsf{constr}(m) & \triangleright & \mathsf{constr}(\mathsf{encr}(m,k)) \\ \mathsf{constr}(m_1), \mathsf{constr}(m_2) & \triangleright & \mathsf{constr}(\mathsf{conc}(m_1,m_2)). \end{array}$$

For instance, we have:

$$\mathsf{recv}(\mathsf{encr}(\mathsf{m},\mathsf{k}_1)), \mathsf{recv}(\mathsf{inv}(\mathsf{k}_1)), \mathsf{recv}(\mathsf{k}_2) \vdash_{\mathrm{DY}} \mathsf{constr}(\mathsf{encr}(\mathsf{m},\mathsf{k}_2)).$$

□

3.2 Deductive Algorithmic Knowledge

We now introduce a propositional modal logic for reasoning about the implicit and explicit knowledge of an agent, where the explicit knowledge is formalized as a logical theory. In this section, we focus on a single agent. The framework is extended to multiple agents in Section 3.5.

The syntax of the logic is simply that of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$, as given in chapter 2. The primitive propositions are T^g_Σ, the ground terms over signature Σ. In this setting, $X\varphi$ is read as "the agent explicitly knows φ, according to his logical theory".

Rather than taking a general deductive system over an arbitrary signature, consider a special form of deductive system. A *KD signature* Σ is a signature containing a class of constructors $\{\mathsf{ob}, \mathsf{true}, \mathsf{false}, \mathsf{not}, \mathsf{and}, \mathsf{know}, \mathsf{xknow}\} \subseteq \Sigma$ corresponding to the operators in the logic; the constructors true and false have arity 0, $\mathsf{ob}, \mathsf{not}, \mathsf{know}$ and xknow have arity 1, and and has arity 2. The terms of the form $\mathsf{ob}(t)$ in T^g_Σ are called the observations, and we let $Obs = \{\mathsf{ob}(t) \mid t \in T^g_\Sigma\} \subseteq T^g_\Sigma$ denote the set of observations. Let ob range over the observations in Obs. Note that Obs is a countably infinite set. A *KD deductive system* D is a decidable deductive system defined over a KD signature Σ, such that no observation arises as the conclusion of a deduction rule in D. Formally, for all $ob \in Obs$ and for all rules $t_1, \ldots, t_n \triangleright t$ of D, there does not exist a ground substitution ρ such that $\rho(t) = ob$. The intuition is that observations are facts that the agent has directly observed, as opposed to facts that have been derived by reasoning.

The semantics of the logic follows the standard possible-worlds presentation given in Chapter 2. A *deductive algorithmic knowledge structure* is a tuple $M =$

(W, π, D), where W is a set of worlds π is an interpretation for the primitive propositions at each world, and D is a KD deductive system over Σ, with observation set Obs. Every world w in W is of the form (e, obs), where e is a state of the environment (taken from a set E), that captures the general state of the system, and obs is a set of observations, taken from Obs, representing the observations that the agent has made at that world. Hence, $W \subseteq E \times \wp_{fin}(Obs)$.[1] We abstract away from the question of how the agent makes those observations, and any temporal relationship between the worlds. A world simply represents a snapshot of the system under consideration. The interpretation π associates with every world the set of primitive propositions that are true at that world, so that for every primitive proposition $p \in T^g_\Sigma$ and world $w \in W$, we have $\pi(w)(p) \in \{\textbf{true}, \textbf{false}\}$. The only assumption we make is that the interpretation respects the observations made at a world, that is, $\pi(e, obs)(ob) = \textbf{true}$ if and only if $ob \in obs$.

There is a distinction between a fact (represented by a term t), and an observation of that fact (represented by a term $ob(t)$). For instance, the fact that Alice holds an apple might be represented by the term $\mathsf{holds}(\mathsf{alice}, \mathsf{apple})$, which can be true or not at a world, while the fact that the agent has observed that Alice is holding an apple is represented by the term $\mathsf{ob}(\mathsf{holds}(\mathsf{alice}, \mathsf{apple}))$, which is true if and only if that observation is in the state of the agent. Of course, the observation can be in the state of the agent whether or not $\mathsf{holds}(\mathsf{alice}, \mathsf{apple})$ is true, if the agent makes unreliable observations.

Let $\mathcal{M}^{\mathrm{ded}}(\Sigma)$ be the class of all deductive algorithmic knowledge structures with KD signature Σ. For a fixed KD deductive system D over Σ, let $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ be the class of all deductive algorithmic knowledge structures with deductive system D.

Define a relation $\sim$ on the worlds that captures the worlds that the agent cannot distinguish based on the observations. Take $w \sim w'$ if $w = (e, obs)$ and $w' = (e', obs)$ for some e, e', and set of observations obs. Clearly, $\sim$ is an equivalence relation.

To define the semantics of the X operator, we need to invoke the deductive system. To do this, first define the translation of a formula φ in $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ into a term φ^T in the term algebra, in the completely obvious way: p^T is p (recall that primitive propositions are just terms in T^g_Σ), $true^T$ is true, $false^T$ is false, $(\neg\varphi)^T$ is $\mathsf{not}(\varphi^T)$, $(\varphi \wedge \psi)^T$ is $\mathsf{and}(\varphi^T, \psi^T)$, $(K\varphi)^T$ is $\mathsf{know}(\varphi^T)$, and $(X\varphi)^T$ is $\mathsf{xknow}(\varphi^T)$.

Define what it means for a formula φ to be true at a world w of M, written $(M, w) \models \varphi$, inductively as follows:

$(M, w) \models true$

[1] For simplicity, assume that the observations form a set. This implies that repetition of observations and their order is unimportant. It is easy to model the case where the observations form a sequence, at the cost of complicating the presentation.

$(M, w) \not\models \mathit{false}$
$(M, w) \models p$ if $\pi(w)(p) = \textbf{true}$
$(M, w) \models \neg\varphi$ if $(M, w) \not\models \varphi$
$(M, w) \models \varphi \wedge \psi$ if $(M, w) \models \varphi$ and $(M, w) \models \psi$
$(M, w) \models K\varphi$ if $(M, w') \models \varphi$ for all $w' \sim w$
$(M, w) \models X\varphi$ if $w = (e, obs)$ and $obs \vdash_D \varphi^T$.

This semantics is the same as that of Section 2.4, except that the $X\varphi$ operator is interpreted via the deductive system D rather than an explicit knowledge algorithm. The semantics agree if we notice that to every deductive algorithmic knowledge structure $M = (W, \pi, D)$ we can associate an algorithmic knowledge structure $M' = (W, \mathcal{V}, \pi, \mathtt{A})$ with $\mathcal{V}((e, obs)) = obs$, and $\mathtt{A}(\varphi, obs) =$ "Yes" if and only if $obs \vdash_D \varphi^T$, which is implementable since D is assumed to be decidable.

Example 3.2. Consider the deductive system DY from Example 3.1, over an augmented signature containing the constructors required to make it a KD deductive system. This deductive system can be viewed as a KD deductive system by adding a rule $\mathsf{ob}(t) \triangleright t$ to DY. Intuitively, an observation represents a message intercepted by the adversary. The subterm relation $\sqsubseteq$ on T^g_{DY} typically considered in the security literature is defined as the smallest relation subject to:

$t \sqsubseteq t$
if $t \sqsubseteq t_1$ then $t \sqsubseteq \mathsf{conc}(t_1, t_2)$
if $t \sqsubseteq t_2$ then $t \sqsubseteq \mathsf{conc}(t_1, t_2)$
if $t \sqsubseteq t_1$ then $t \sqsubseteq \mathsf{encr}(t_1, t_2)$.

Consider a structure $M = (W, \pi, \mathrm{DY})$, where we record at every world all messages intercepted by the adversary at that world. We restrict the observations at a world to be of the form $\mathsf{ob}(\mathsf{recv}(t))$, for ground terms t in which has does not occur. Let π be an interpretation that respects the observations made at a world, and such that $\pi(e, obs)(\mathsf{has}(t)) = \textbf{true}$ if and only if there exists $t' \in T^g_{\mathrm{DY}}$ such that $\mathsf{ob}(\mathsf{recv}(t')) \in obs$ and $t \sqsubseteq t'$. In other words, $\mathsf{has}(t)$ holds at a world if t is a subterm of a message intercepted by the adversary. Let w_1 be a world with observations

$$\{\mathsf{ob}(\mathsf{recv}(\mathsf{encr}(\mathsf{m}, \mathsf{k}_1))), \mathsf{ob}(\mathsf{recv}(\mathsf{encr}(\mathsf{inv}(\mathsf{k}_1), \mathsf{k}_2)))\},$$

and w_2 a world with observations

$$\{\mathsf{ob}(\mathsf{recv}(\mathsf{encr}(\mathsf{m}, \mathsf{k}_1))), \mathsf{ob}(\mathsf{recv}(\mathsf{encr}(\mathsf{inv}(\mathsf{k}_1), \mathsf{k}_2))), \mathsf{ob}(\mathsf{recv}(\mathsf{inv}(\mathsf{k}_2)))\}.$$

By definition of π, $(M, w_1) \models K(\mathsf{has}(\mathsf{m}))$ and $(M, w_2) \models K(\mathsf{has}(\mathsf{m}))$, so that at both worlds, the adversary implicitly knows he has the message m. However, from

the results of Example 3.1, we see that $(M, w_2) \models X(\mathsf{has}(\mathsf{m}))$, while $(M, w_1) \models \neg X(\mathsf{has}(\mathsf{m}))$. In other words, the adversary explicitly knows he has m at world w_2 (where he has intercepted the appropriate terms), but not at world w_1. □

Example 3.3. The following deduction rules can be added to any deductive system to obtain a deductive system that captures a subset of the inferences that can be performed in propositional logic:

$$
\begin{array}{ll}
t \triangleright \mathsf{not}(\mathsf{not}(t)) & t, t' \triangleright \mathsf{and}(t, t') \\
\mathsf{not}(\mathsf{not}(t)) \triangleright t & \mathsf{and}(t, t') \triangleright t \\
t \triangleright \mathsf{not}(\mathsf{and}(\mathsf{not}(t), \mathsf{not}(t'))) & \mathsf{and}(t, t') \triangleright t' \\
t' \triangleright \mathsf{not}(\mathsf{and}(\mathsf{not}(t), \mathsf{not}(t'))) & t, \mathsf{not}(t) \triangleright \mathsf{false} \\
\mathsf{not}(\mathsf{and}(t, \mathsf{not}(t'))), t \triangleright t' & \mathsf{false} \triangleright t \\
\mathsf{not}(\mathsf{and}(t, \mathsf{not}(t'))), t' \triangleright t. &
\end{array}
$$

One advantage of these rules, despite the fact that they are incomplete, is that they can be used to perform very efficient (linear-time) propositional inference. □

Example 3.4. We can easily let the agent explicitly reason about his deductive algorithmic knowledge by adding a rule

$$t \triangleright \mathsf{xknow}(t) \tag{3.1}$$

to his deductive system D. Thus, if M is a deductive algorithmic knowledge structure over D, and $(M, w) \models X\varphi$, then we have $w = (e, obs)$, with $obs \vdash_D \varphi^T$, and by the above rule, the deductive system D can also derive $obs \vdash_D \mathsf{xknow}(\varphi^T)$, so that $obs \vdash_D (X\varphi)^T$. Thus, $(M, w) \models X(X\varphi)$, as required. It is possible to restrict the deductive algorithmic knowledge of an agent with respect to his own deductive algorithmic knowledge by suitably modifying rule (3.1), restricting it to a subclass of terms. □

The monotonicity of the deductive systems means that for a structure M with worlds $w = (e, obs)$, $w' = (e', obs')$, and $obs \subseteq obs'$, we have $(M, w) \models X\varphi$ implies $(M, w') \models X\varphi$. Thus, explicit knowledge of facts is never lost when new observations are made.

It is natural to consider classes of signatures (and deductive systems) that capture logical theories dealing with only part of the formulas expressible in $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. For instance, it may make sense to distinguish the notion of a primitive signature, that does not provide constructors for the propositional and modal connectives. Intuitively, a deductive system based on a primitive signature only permits reasoning about the explicit knowledge of primitive propositions; $X\varphi$ is false for any φ not a primitive proposition.

3.3 Axiomatizations

Clearly, for a particular deductive system, the properties of X depend on that deductive system. Intuitively, we should be able to read off the properties of X from the deduction rules themselves. This is hardly surprising. Properties of the knowledge algorithms in the framework of algorithmic knowledge immediately translate to properties of the X operator. For instance, if a knowledge algorithm is sound, that is, if whenever it answers "Yes" in a world w for a formula φ then φ is true at that world, then $X\varphi \Rightarrow \varphi$ is valid in a structure using such a knowledge algorithm. What is interesting in the context of deductive algorithmic knowledge is that we can completely characterize the properties of X, because of the structure of the deductive systems. The remainder of this section aims at making this statement more precise.

As a first step, consider an axiomatization for reasoning about deductive systems in general, independently of the actual deduction rules of the system. For this, we need axioms capturing propositional reasoning in the logic:

Prop. All instances of propositional tautologies
MP. From φ and $\varphi \Rightarrow \psi$ infer ψ.

Axiom Prop can be replaced by an axiomatization of propositional tautologies, as in Section 2.2. The following axioms capture the properties of the knowledge operator, as in Section 2.2:

K1. $(K\varphi \wedge K(\varphi \Rightarrow \psi)) \Rightarrow K\psi$
K2. From φ infer $K\varphi$
K3. $K\varphi \Rightarrow \varphi$
K4. $K\varphi \Rightarrow KK\varphi$
K5. $\neg K\varphi \Rightarrow K\neg K\varphi$.

Since algorithmic knowledge is interpreted with respect to the observations at the current state, and that two states are indistinguishable to an agent if the same observations are made at both states, agents know whether or not they explicitly know a fact. This is captured by the following axiom:

X1. $X\varphi \Rightarrow KX\varphi$

In the presence of K1–5, we saw in Section 2.4 that $\neg X\varphi \Rightarrow K\neg X\varphi$ is provable from X1. In addition, all observations are explicitly known. This fact is expressed by the following axiom:

X2. $ob \Leftrightarrow X\,ob$.

Formally, this is a consequence of the definition of deduction in Section 3.1: recall that for all terms t of a deductive system D, we have $t \vdash_D t$. An easy consequence of X1–2 is that indistinguishable worlds have exactly the same observations. It is easy to see that the formulas $ob \Rightarrow K\,ob$ and $\neg ob \Rightarrow K \neg ob$ are provable.

Let $\mathrm{AX}^{\mathrm{ded}}$ consists of the axioms Prop, MP, K1–5, and X1–2. Without further assumptions on the deductive systems under consideration, $\mathrm{AX}^{\mathrm{ded}}$ completely characterizes reasoning about deductive algorithmic knowledge.

Theorem 3.5. *The axiomatization* $\mathrm{AX}^{\mathrm{ded}}$ *is sound and complete for* $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ *with respect to* $\mathcal{M}^{\mathrm{ded}}(\Sigma)$.

If we want to reason about deductive algorithmic knowledge structures equipped with a specific deductive system, we can say more. We can essentially capture the reasoning with respect to the specific deductive system within our logic. The basic idea is to translate deduction rules of the deductive system into formulas of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. A deduction rule of the form $t_1, \ldots, t_n \triangleright t$ in D is translated to a formula $(Xt_1^R \wedge \ldots \wedge Xt_n^R) \Rightarrow Xt^R$. Define the formula t^R corresponding to the term t by induction on the structure of t: true^R is true, false^R is false, $(\mathsf{not}(t))^R$ is $\neg(t^R)$, $(\mathsf{and}(t_1, t_2))^R$ is $t_1^R \wedge t_2^R$, $(K\varphi)^R$ is $\mathsf{know}(\varphi^R)$, $(X\varphi)^R$ is $\mathsf{xknow}(\varphi^R)$, and t^R is t for all other terms t. In fact, such a translation yields an axiom scheme, where we view the variables in $t_1, \ldots, t_n, t$ as scheme metavariables, to be replaced by appropriate elements of the term algebra.[2] It is easy to see that $(t^T)^R = t$ for all terms t. Furthermore, we do not translate KD constructors that appear under non-KD constructors within a term. (Intuitively, these constructors will never arise out of the translation of formulas given in Section 3.2.) Let Ax^D be the set of axioms derived in this way for the KD deductive system D.

Note that these axioms cannot be complete for $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$, since there are formulas of the form $X\psi$ that cannot be true in any structure in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$, namely, $X\psi$ where ψ^T is not derivable from any set of observations using the deductive system D. Thus, $\neg X\psi$ is valid for those ψ, but the axioms above clearly cannot prove $\neg X\psi$. In other words, the axioms in Ax^D capture deducibility in $\vdash_D$, rather than non-deducibility. We can however establish completeness with respect to a more general class of structures, intuitively, those structures using a deductive system containing *at least* the deduction rules in D. Let $\mathcal{M}^{\mathrm{ded}}_{D\subseteq}(\Sigma) = \{M \mid M \in \mathcal{M}^{\mathrm{ded}}_{D'}(\Sigma), D \subseteq D'\}$.

Theorem 3.6. *The axiomatization* $\mathrm{AX}^{\mathrm{ded}} + \{Ax^D\}$ *is sound and complete for* $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ *with respect to* $\mathcal{M}^{\mathrm{ded}}_{D\subseteq}(\Sigma)$.

[2] One needs to be careful when defining this kind of axiom scheme formally. Intuitively, an axiom scheme of the form above, with metavariables appearing in terms, corresponds to the set of axioms where each primitive proposition in the axiom is a ground substitution instance of the appropriate term in the axiom scheme.

3.4 Decision Procedures

In this section, we study the decision problem for $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ satisfiability, that is, the problem of determining, for a given formula, whether it is satisfiable. Since our logic extends the logic $\mathcal{L}^{\mathrm{K}}(T^g_\Sigma)$, and since the complexity of the decision problem for the latter is NP-complete (Theorem 2.10), the difficulty of deciding satisfiability for $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is at least as hard.

We measure complexity in terms of the size of the formulas, as well as the size of the models. Define the size $|t|$ of a term t to be the number of symbols required to write t, where each operation symbol is counted as a single symbol. If Γ is a set of terms, then $|\Gamma|$ is just the sum of the sizes of the terms in Γ. Similarly, the size $|\varphi|$ of a formula is defined to be the number of symbols required to write φ, where again each operation symbol is counted as a single symbol. The size $|M|$ of a model $M \in \mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ (that is, for a specific deductive system D) is taken to be the sum of the sizes of the states, where the size of a state $(e, \{ob_1, \ldots, ob_k\})$ is $1 + |ob_1| + \cdots + |ob_k|$.

It is known that the decision problem for $\mathcal{L}^{\mathrm{K}}$ satisfiability is NP-complete (Theorem 2.10). Adding deductive algorithmic knowledge does not add to the complexity if we do not require a fixed deductive system. Intuitively, for satisfiability, we can simply take as a deductive system one with specific deduction rules sufficient to satisfy the subformulas $X\varphi$ appearing in the formula.

Theorem 3.7. *The problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}(\Sigma)$ is NP-complete.*

What happens if we fix a particular deductive system, and want to establish whether a formula φ is satisfiable in a structure over that particular deductive system? The difficulty of this problem depends intrinsically on the difficulty of deciding whether a deduction $\Gamma \vdash_D t$ exists in D. Since this problem may be arbitrarily difficult for certain deductive systems D, reasoning in our logic can be arbitrarily difficult over those deductive systems. On the other hand, if the deductive system is decidable in polynomial time (i.e., if the problem of deciding whether a deduction $\Gamma \vdash_D t$ exists in D can be solved in time polynomial in $|\Gamma|$ and $|t|$), then the decision problem for our logic remains relatively easy.

Theorem 3.8. *For any given propositional deductive system D that is decidable in polynomial time, the problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ is NP-complete.*

There is a class of deductive systems that can be efficiently decided, and thus by Theorem 3.8 lead to a reasonable complexity for $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. Say a deductive

system D is *local* if whenever $\Gamma \vdash_D t$ there exists a local deduction of t from Γ. A deduction is local if every proper subterm of a term in the deduction is either a proper subterm of t, a proper subterm of a member of Γ, or appears as a subterm of a deduction rule in D. One can show that, for any deductive system D, whether a local deduction of t from Γ exists in time polynomial in $|\Gamma|$ and $|t|$. If D is local, so that the existence of a deduction ensures the existence of a local deduction, then the deduction relation $\vdash_D$ is polynomial-time decidable. The deductive system in Example 3.1 is local, while adding the deduction rules in Example 3.3 to any local deductive system yields a local deductive system.

Corollary 3.9. *For any local KD deductive system D, the problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ is NP-complete.*

3.5 Multiple Agents

The framework we have described extends to multiple agents in a straightforward way. We simply need to equip every agent with a deductive system. A priori, there is no difficulty in modeling this using what has been already done. Unfortunately, this does not let an agent explicitly reason about another agent's knowledge. In order to do this, it is necessary to modify and extend the framework. The syntax of the logic is just $\mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma)$, as expected, where we take Σ to be a KD signature for n agents. A *KD signature for n agents* is a signature containing the operation symbols true, false, not, and, as well as the operation symbols ob_i, know_i, xknow_i, for $i \in \{1, \ldots, n\}$, where $\mathsf{true}, \mathsf{false}$ have arity 0, $\mathsf{ob}_i, \mathsf{not}, \mathsf{know}_i, \mathsf{xknow}_i$ have arity 1, and and has arity 2. The terms of the form $\mathsf{ob}_i(t)$ in T^g_Σ are called the observations for agent i, and we let $Obs_i = \{\mathsf{ob}_i(t) \mid t \in T^g_\Sigma\} \subseteq T^g_\Sigma$ denote the set of observations for agent i. We typically let ob range over observations. A *KD deductive system for n agents* D is a decidable deductive system defined over a KD signature Σ for n agents, with the restriction that no observation arises as the conclusion of a deduction rule in D. Formally, for all i, for all $ob \in Obs_i$ and for all rules $t_1, \ldots, t_n \rhd t$ of D, there does not exist a ground substitution ρ such that $\rho(t) = ob$.

The models are a straightforward generalization of those used in the single-agent case. A *deductive algorithmic knowledge structure for n agents* is a tuple $M = (W, \pi, D_1, \ldots, D_n)$, where W is a set of worlds, π is an interpretation for the primitive propositions, and D_i is a KD deductive system for n agents over Σ. Every world w in W is of the form $(e, obs_1, \ldots, obs_n)$, where e is a state of the environment that captures the general state of the system, and obs_i is a finite set

of observations from Obs_i, representing the observations that agent i has made at that world. The interpretation π associates with every world the set of primitive propositions true at that world, so that for all primitive proposition $p \in T^g_\Sigma$ and all worlds $w \in W$, we have $\pi(w)(p) \in \{\textbf{true}, \textbf{false}\}$. The only assumption on the interpretation is that it respects the observations made at a world, that is, $\pi(e, obs_1, \ldots, obs_n)(\mathsf{ob}_i(t)) = \textbf{true}$ if and only if $\mathsf{ob}_i(t) \in obs_i$.

Let $\mathcal{M}^{\mathrm{ded}}_n(\Sigma)$ be the set of all deductive algorithmic knowledge structures with KD signature for n agents Σ. For fixed KD deductive systems $D_1, \ldots, D_n$ over Σ, let $\mathcal{M}^{\mathrm{ded}}_{D_1,\ldots,D_n}(\Sigma)$ be the set of all deductive algorithmic knowledge structures for n agents with deductive systems $D_1, \ldots, D_n$ (that is, where agent i uses deductive system D_i).

The remaining definitions generalize in a similar way. Define, for each agent, a relation on the worlds that captures the worlds that the agent cannot distinguish, based on his observations. More precisely, take $w \sim_i w'$ if $w = (e, obs_1, \ldots, obs_n)$ and $w' = (e', obs'_1, \ldots, obs'_n)$, for some $e, e', obs_1, \ldots, obs_n, obs'_1, \ldots, obs'_n$, with $obs_i = obs'_i$. Again, each $\sim_i$ is an equivalence relation.

The translation of a formula φ into a term φ^T of the deductive system now takes into account the name of the agents. As expected, we take p^T is p, $true^T$ is true, $false^T$ is false, $(\neg\varphi)^T$ is $\mathsf{not}(\varphi^T)$, $(\varphi \wedge \psi)^T$ is $\mathsf{and}(\varphi^T, \psi^T)$, $(K_i\varphi)^T$ is $\mathsf{know}_i(\varphi^T)$, and $(X_i\varphi)^T$ is $\mathsf{xknow}_i(\varphi^T)$.

The semantics is just like that of Section 3.2, except with the following rules for $K_i\varphi$ and $X_i\varphi$:

$(M, w) \models K_i\varphi$ if $(M, w') \models \varphi$ for all $w' \sim_i w$
$(M, w) \models X_i\varphi$ if $w = (e, obs_1, \ldots, obs_n)$ and $obs_i \vdash_{D_i} \varphi^T$.

Example 3.10. The phenomenon of *simulative inference* arises when, roughly speaking, an agent can reconstruct the reasoning of another agent. It is possible to capture this by making suitable assumptions on an agent's deductive system. A deductive system D_i for agent i *permits simulative inference of agent j with D_j* if D_i contains a rule $\mathsf{ob}_j(t) \rhd \mathsf{xknow}_j(\mathsf{ob}_j(t))$, and for every rule $t_1, \ldots, t_k \rhd t$ of D_j, there is a corresponding rule $\mathsf{xknow}_j(t_1), \ldots, \mathsf{xknow}_j(t_k) \rhd \mathsf{xknow}_j(t)$ in D_i. It is then easy to check that if we have $(M, w) \models X_j\varphi$ for some world $w = (e, obs_1, \ldots, obs_n)$ with $\{ob_1, \ldots, ob_k\} \subseteq obs_j$, and $(M, w) \models X_i(ob_1) \wedge \ldots \wedge X_i(ob_k)$, then $(M, w) \models X_iX_j\varphi$. Note that this derivation assumes that the agent i can explicitly determine that agent j has observed $ob_1, \ldots ob_k$. □

As far as axiomatizations are concerned, we can essentially lift the results of Section 3.3. It suffices to consider an axiomatization where K1–5 now refer to K_i rather than just K. For instance, K1 becomes $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$, for every agent i. In a similar way, the axiom X1 simply becomes $X_i\varphi \Rightarrow K_iX_i\varphi$. For

X2, we need to further restrict the observations to be those of the agent under consideration: $ob \Leftrightarrow X_i ob$ (if $ob \in Obs_i$). Let $\mathrm{AX}_n^{\mathrm{ded}}$ be the resulting axiomatization. The following result is obtained in a straightforward way:

Theorem 3.11. *The axiomatization* $\mathrm{AX}_n^{\mathrm{ded}}$ *is sound and complete for* $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$ *with respect to* $\mathcal{M}_n^{\mathrm{ded}}(\Sigma)$.

As in the single agent case, we can capture the reasoning with respect to specific deductive systems (one per agent) within our logic. Again, we translate deduction rules of the deductive systems into formulas of $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$. Consider the deductive system D_i for agent i. A deduction rule of the form $t_1, \ldots, t_n \triangleright t$ in D_i is translated to a formula $(X_i t_1^R \wedge \ldots \wedge X_i t_n^R) \Rightarrow X_i t^R$. Define the formula t^R corresponding to the term t by induction on the structure of t: true^R is $true$, false^R is $false$, $(\mathsf{not}(t))^R$ is $\neg(t^R)$, $(\mathsf{and}(t_1, t_2))^R$ is $t_1^R \wedge t_2^R$, $(K_j\varphi)^R$ is $\mathsf{know}_j(\varphi^R)$, $(X_j\varphi)^R$ is $\mathsf{xknow}_j(\varphi^R)$, and t^R is t for all other terms t. (As in Section 3.3, such a translation yields an axiom scheme, where we view the variables in $t_1, \ldots, t_n, t$ as scheme metavariables, to be replaced by appropriate elements of the term algebra.) Let $Ax_n^{D_i}$ be the set of axioms derived in this way for the KD deductive system D_i of agent i. As in the single agent case, we cannot capture exactly the reasoning in structures where agent i is using deductive system D_i, since we cannot capture non-deducibility within the logic. Therefore, completeness is established with respect to a wider class of structures. Let $\mathcal{M}_{D_1,\ldots,D_n\subseteq}^{\mathrm{ded}}(\Sigma) = \{M \mid M \in \mathcal{M}_{D_1',\ldots,D_n'}^{\mathrm{ded}}, D_1 \subseteq D_1', \ldots, D_n \subseteq D_n'\}$.

Theorem 3.12. *The axiomatization* $\mathrm{AX}_n^{\mathrm{ded}} + \{Ax_n^{D_1}, \ldots, Ax_n^{D_n}\}$ *is sound and complete for* $\mathcal{L}_n^{\mathrm{KX}}(\Sigma)$ *with respect to* $\mathcal{M}_{D_1,\ldots,D_n\subseteq}^{\mathrm{ded}}(\Sigma)$.

The complexity of the decision problem in the case of multiple agents reflects the complexity of the decision problem of the modal logic of knowledge for multiple agents. $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$ extends the logic $\mathcal{L}_n^{\mathrm{K}}$, and it is known that the decision problem for the latter is PSPACE-complete (Theorem 2.10). As in the single agent case, adding deductive algorithmic knowledge does not affect the complexity of the decision problem if we do not require a fixed deductive system.

Theorem 3.13. *If* $n \geq 2$*, the problem of deciding whether a formula* φ *of* $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$ *is satisfiable in a structure in* $\mathcal{M}_n^{\mathrm{ded}}(\Sigma)$ *is PSPACE-complete.*

There is no clear candidate for an equivalent of Theorem 3.8 in the multiple agents context. Assuming every agent uses a tractable deductive system yields an easy EXPTIME upper bound on the decision problem for $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$, while the best lower bound we obtain is the same as the one in Theorem 3.13, that is, the problem is PSPACE-hard.

Notes

The work in this chapter appeared in a preliminary form in [Pucella 2004].

There has been a fair amount of work on developing models for agents reasoning via logical theories; see, for instance, the approaches of Konolige [1986] and Giunchiglia et al. [1993]), which reason completely within the logical theories while assuming a global logical theory for the world. In contrast, the logic in this chapter is based on a standard possible-worlds semantics.

The use of Horn theories to approximate knowledge bases is described by Selman and Kautz [1996]. The Dolev-Yao adversary is due to Dolev and Yao [1983], and it is now standard in the field of security protocol analysis. Imposing a probability distribution on the set of possible worlds is explored by Fagin and Halpern [1994].

The notion of term algebra is usually studied in universal algebra [Burris and Sankappanavar 1981], although it is also often used in term rewriting systems [Baader and Nipkow 1998]. Sorted term algebras are described by Higgins [1963].

The incomplete inference rules for propositional reasoning used in Example 3.3 are taken from McAllester [1993]. The notion of a local deductive system is explored in detail by McAllester [1993]. It generalizes a particularly well-known fact that ensures that a deductive system can be efficiently searched for deductions, namely that one need only consider subformulas of a formula one is attempting to derive.

Simulative inference, as described in Example 3.10, was studied by Kaplan and Schubert [2000]. They work in a slightly different setting than the one in this chapter. They assume that the inference engine is explicitly told formulas. Thus, they essentially work in a setting similar to that of belief revision [Alchourrón, Gärdenfors, and Makinson 1985]. They also implicitly make the assumption that agents are aware of the observation of other agents, since they study simulative inference in a context where all the agents make the same observations.

4
Probabilistic Algorithmic Knowledge

ALL the examples of algorithm knowledge appearing in the literature use *sound* knowledge algorithms: although the algorithm may not give an answer under all circumstances, when it says "Yes" on input φ, the agent really does know φ in the standard possible-worlds sense. Although soundness is not required in the basic definition, it does seem to be useful in many applications.

Our interest in this chapter is knowledge algorithms that may use some randomization. There are numerous examples of natural randomized knowledge algorithms. With randomization, whether or not the knowledge algorithm says "Yes" may depend on the outcome of coin tosses. This poses a slight difficulty in even giving semantics to algorithmic knowledge, since the standard semantics makes sense only for deterministic algorithms. One solution to this problem is to make the algorithms deterministic by supplying them an extra argument (intuitively, the outcome of a sequence of coin tosses) to "derandomize" them. We show that this approach provides a natural extension of the deterministic case.

Having defined the framework, we try to characterize the information obtained by getting a "Yes" answer to a query for φ. If the knowledge algorithm is sound, then a "Yes" answer guarantees that φ is true. However, the randomized algorithms of most interest to us give wrong answers with positive probability, so are not sound. Nevertheless, it certainly seems that if the probability that the algorithm gives the wrong answer is low, it provides very useful information when it says "Yes" to a query φ. This intuition appears in the randomized algorithms literature, where a "Yes" answer from a highly reliable randomized algorithm, that is, one with a low probability of being wrong, is deemed "good enough". In what sense is this true? One contribution of this chapter is to provide a formal answer to that question. It may seem that a "Yes" answer to a query φ from a highly reliable randomized knowledge algorithm should make the probability that φ is true be high but, as we show, this is not necessarily true. Rather, the information should

be viewed as *evidence* that φ is true; the probability that φ is true also depends in part on the prior probability of φ.

Evidence has been widely studied in the literature on inductive logic. Evidence for a particular hypothesis can be accumulated from different sources, and it is possible to combine the evidence provided by knowledge algorithms with other kind of evidence already accumulated for a formula, obtained for example by running other kinds of tests, and look at the combined evidence. We do not consider evidence in such a general setting in this chapter; rather, we focus on the evidence contributed specifically by a randomized knowledge algorithm. Chapter 5 provides a treatment of evidence in a more general setting.

4.1 Randomized Knowledge Algorithms

Randomized knowledge algorithms arise frequently in the literature (although they have typically not been viewed as knowledge algorithms). In order to deal with randomized algorithms in the algorithmic knowledge framework, we need to address a technical question. Randomized algorithms are possibly nondeterministic; they may not yield the same result on every invocation with the same arguments. Since $X_i\varphi$ holds at a world w if the knowledge algorithm answers "Yes" at that world, this means that, with the semantics of Section 2.4, $X_i\varphi$ would not be well defined. Whether it holds at a given world depends on the outcome of random choices made by the algorithm. However, the semantics should unambiguously declare a formula either true or false.

Before we describe the chosen solution to the problem, consider another potential solution, which is to define the satisfaction relation probabilistically. That is, rather than associating a truth value with each formula at each world, we associate a probability $\Pr(w, \varphi)$ with each formula φ at each world w. The standard semantics can be viewed as a special case of this semantics, where the probabilities are always either 0 or 1. Under this approach, it seems reasonable to take $\Pr(w, p)$ to be either 0 or 1, depending on whether primitive proposition p is true at state w, and to take $\Pr(w, X_i\varphi)$ to be the probability that i's knowledge algorithm returns "Yes" given inputs φ and $\mathcal{V}_i(w)$. However, it is not then clear how to define $\Pr(w, \varphi \wedge \psi)$. Taking it to be $\Pr(w, \varphi)\Pr(w, \psi)$ implicitly treats φ and ψ as independent, which is clearly inappropriate if ψ is $\neg\varphi$. Even ignoring this problem, it is not clear how to define $\Pr(w, X_i\varphi \wedge X_i\psi)$, since again there might be correlations between the output of the knowledge algorithm on input $(\varphi, \mathcal{V}_i(w))$ and input $(\psi, \mathcal{V}_i(w))$.

Rather than use probabilistic semantics here, we deal with the problem by adding information to the semantic model to resolve the uncertainty about the truth value of formulas of the form $X_i\varphi$. Observe that if the knowledge algorithm `A` is ran-

domized, then the answer that $\mathtt{A}$ gives on input (φ, ℓ) will depend on the outcome of coin tosses (or whatever other randomizing device is used by $\mathtt{A}$). We thus turn the randomized algorithm into a deterministic algorithm by supplying it with an appropriate argument. For example, we supply an algorithm that makes random choices by tossing coins a sequence of outcomes of coin tosses. We can now interpret a knowledge algorithm answering "Yes" with probability α at a world by considering the probability of those sequences of coin tosses at the state that make the algorithm answer "Yes".

Formally, start with (possibly randomized) knowledge algorithms $\mathtt{A}_1, \ldots, \mathtt{A}_n$. For simplicity, assume that the randomness in the knowledge algorithms comes from tossing coins. A *derandomizer* is a tuple $v = (v_1, \ldots, v_n)$ such that for every agent i, v_i is a sequence of outcomes of coin tosses (heads and tails). There is a separate sequence of coin tosses for each agent rather than just a single sequence of coin tosses, since we do not want to assume that all agents use the same coin. Let V be the set of all such derandomizers. To every randomized algorithm $\mathtt{A}$, associate a derandomized algorithm $\mathtt{A}^d$ which takes as input not just the query φ and local state ℓ, but also the sequence v_i of i's coin tosses, taken from a derandomizer $(v_1, \ldots, v_n)$. A *probabilistic algorithmic knowledge structure* is a tuple $N = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$, where ν is a probability distribution on V and $\mathtt{A}_i^d$ is the derandomized version of $\mathtt{A}_i$. (Note that in a probabilistic algorithmic knowledge structure the knowledge algorithms are in fact deterministic.)

The only assumption on the distribution ν is that it does not assign zero probability to the nonempty sets of sequences of coin tosses that determine the result of the knowledge algorithm. More precisely, assume that for all agents i, formulas φ, and local states ℓ of agent i, $\{v \mid \mathtt{A}_i^d(\varphi, \ell, v_i) = \text{"Yes"}\} \neq \varnothing$ if and only if $\nu(\{v \mid \mathtt{A}_i^d(\varphi, \ell, v_i) = \text{"Yes"}\}) > 0$, and similarly for "No" and "?" answers. Note that this property is met, for instance, if ν assigns nonzero probability to every sequence of coin tosses. There are no other restrictions on ν. In particular, it is not required that the coin be fair or that the tosses be independent. We can capture correlation between the agents' coins by using an appropriate distribution ν.

The truth of a formula is now determined relative to a pair (w, v) consisting of a world w and a derandomizer v. We abuse notation and continue to call these pairs worlds. The semantics of formulas in a probabilistic algorithmic knowledge structure is a straightforward extension of their semantics in algorithmic knowledge structures. The semantics of primitive propositions is given by π; conjunctions and negations are interpreted as usual; for knowledge and algorithmic knowledge, we have

$(N, w, v) \models K_i\varphi$ if $(N, w', v') \models \varphi$ for all $v' \in V$ and all $w' \in W$ such that $w \sim_i w'$

$(N, w, v) \models X_i\varphi$ if $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "Yes", where $v = (v_1, \ldots, v_n)$.

Here, $\mathtt{A}_i^d$ gets v_i as part of its input. $\mathtt{A}_i^d(\varphi, \ell, v_i)$ is interpreted as the output of $\mathtt{A}_i^d$ given that v_i describes the outcomes of the coin tosses. Having the sequence of coin tosses as part of the input allows us to talk about the probability that i's algorithm answers "Yes" to the query φ at a local state ℓ. It is simply $\nu(\{v \mid \mathtt{A}_i^d(\varphi, \ell, v_i) =$ "Yes"$\})$. To capture this in the language, extend the language of $\mathcal{L}_n^{\mathrm{KX}}$ to allow formulas of the form $\Pr(\varphi) \geq \alpha$, read "the probability of φ is at least α". The semantics of such formulas is straightforward:

$(N, w, v) \models \Pr(\varphi) \geq \alpha$ if $\nu(\{v' \mid (N, w, v') \models \varphi\}) \geq \alpha$.

Note that the truth of $\Pr(\varphi) \geq \alpha$ at a world (w, v) is independent of v. Thus, we can abuse notation and write $(N, w) \models \Pr(\varphi) \geq \alpha$. In particular, $(N, w) \models \Pr(X_i\varphi) < \alpha$ (or, equivalently, $(N, w) \models \Pr(\neg X_i\varphi) \geq 1-\alpha$) if the probability of the knowledge algorithm returning "Yes" on a query φ is less than α, given agent i's local state at world w.

If all the knowledge algorithms used are deterministic, then this semantics agrees with the semantics given in Section 2.4. To make this precise, note that if $\mathtt{A}$ is deterministic, then $\mathtt{A}^d(\varphi, \ell, v_i) = \mathtt{A}^d(\varphi, \ell, v_i')$ for all $v, v' \in V$. In this case, we abuse notation and write $\mathtt{A}(\varphi, \ell)$.

Theorem 4.1. *Let $N = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge structure, where $\mathtt{A}_1, \ldots, \mathtt{A}_n$ are deterministic. Let $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$. If there are no occurrences of $\Pr$ in φ, then for all $w \in W$ and all $v \in V$, $(N, w, v) \models \varphi$ if and only if $(M, w) \models \varphi$.*

Thus, derandomizers are not needed to interpret the X_i operators if the knowledge algorithms are all deterministic. Moreover, in general, derandomizers are necessary only to interpret the $\Pr$ and X_i operators.

Theorem 4.2. *Let $N = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ be a probabilistic algorithmic knowledge structure, and let $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1', \ldots, \mathtt{A}_n')$ be an algorithmic knowledge structure where $\mathtt{A}_1', \ldots, \mathtt{A}_n'$ are arbitrary deterministic knowledge algorithms. If there are no occurrences of X_i and $\Pr$ in φ, then for all $w \in W$ and all $v \in V$, $(N, w, v) \models \varphi$ if and only if $(M, w) \models \varphi$.*

Theorems 4.1 and 4.2 justify the decision to "factor out" the randomization of the knowledge algorithms into semantic objects that are distinct from the worlds; the semantics of formulas that do not depend on the randomized choices do not in fact depend on those additional semantic objects.

4.2 Measures of Confirmation and Evidence

We are often interested in randomized knowledge algorithms that may sometimes make mistakes. For example, suppose that Alice has in her local state a number $n > 2$. Let prime be a proposition true at world w if and only if the number n in Alice's local state is prime. Clearly, Alice either (implicitly) knows prime or knows $\neg$prime. However, this is implicit knowledge. Suppose that Alice uses the following randomized primality-testing algorithm to test if n is prime. That algorithm uses a (polynomial-time computable) predicate $P(n, a)$ with the following properties, for a natural number n and $1 \leq a \leq n - 1$:

(1) $P(n, a) \in \{0, 1\}$,
(2) if n is composite, $P(n, a) = 1$ for at least $\frac{n}{2}$ choices of a,
(3) if n is prime, $P(n, a) = 0$ for all a.

Thus, Alice uses the following randomized knowledge algorithm $\mathtt{A}_{\mathrm{Alice}}$: when queried about prime, the algorithm picks a number a at random between 0 and the number n in Alice's local state; if $P(n, a) = 1$, it says "No" and if $P(n, a) = 0$, it says "Yes". (It is irrelevant for the purpose of this example what the algorithm does on other queries.)

It is not hard to check that $\mathtt{A}_{\mathrm{Alice}}$ has the following properties: If the number n in Alice's local state is prime, then $\mathtt{A}_{\mathrm{Alice}}$ answers "Yes" to a query prime with probability 1 (and hence "No" to the same query with probability 0). If n is composite, $\mathtt{A}_{\mathrm{Alice}}$ answers "Yes" to a query prime with probability $\leq \frac{1}{2}$ and "No" with probability $\geq \frac{1}{2}$. Thus, if n is composite, there is a chance that $\mathtt{A}_{\mathrm{Alice}}$ will make a mistake, although we can make the probability of error arbitrarily small by applying the algorithm repeatedly.

Randomized knowledge algorithms like this are quite common in the literature. They are not sound, but are "almost sound". The question is what we can learn from such an "almost sound" algorithm. Note that we know the probability that $\mathtt{A}_{\mathrm{Alice}}$ says "Yes" given that n is prime; what we are interested in is the probability that n is prime given that $\mathtt{A}_{\mathrm{Alice}}$ says "Yes". (Of course, n is either prime or not. However, if Alice has to make decisions based on whether n is prime, it seems reasonable for her to ascribe a subjective probability to n's being prime. It is this subjective probability that we are referring to here.)

Bayes' rule tells us that

$$\Pr(n \text{ is prime} \mid \mathtt{A}_{\mathrm{Alice}} \text{ says "Yes"}) = \frac{\Pr(\mathtt{A}_{\mathrm{Alice}} \text{ says "Yes"} \mid n \text{ is prime})\Pr(n \text{ is prime})}{\Pr(\mathtt{A}_{\mathrm{Alice}} \text{ says "Yes"})}. \quad (4.1)$$

The only information in this equation we have is $\Pr(\mathtt{A}_{\mathrm{Alice}}$ says "Yes" $\mid n$ is prime$)$. If we had $\Pr(n$ is prime$)$, we could derive $\Pr(\mathtt{A}_{\mathrm{Alice}}$ says "Yes"$)$. However, we do not have that information, since there is no probability distribution on the number in Alice's local state. Although we do not have the information needed to compute $\Pr(n$ is prime $\mid \mathtt{A}_{\mathrm{Alice}}$ says "Yes"$)$, there is still a strong intuition that if $X_i\varphi$ holds, this tells us something about whether the number is prime or not. How can this be formalized?

Intuitively, the fact that $X_i\varphi$ holds provides "evidence" that φ holds. But what is evidence? Evidence has been studied in depth in the philosophical literature, under the name of *confirmation theory*. Confirmation theory aims at determining and measuring the support a piece of evidence provides a hypothesis. As we mentioned in the introduction, many different measures of confirmation have been proposed in the literature. Typically, a proposal has been judged on the degree to which it satisfies various properties that are considered appropriate for confirmation. For example, it may be required that a piece of evidence e confirms a hypothesis h if and only if e makes h more probable. We will not enter the debate as to which class of measures of confirmation is more appropriate. For our purposes, most confirmation functions are useless: they assume that there is a prior on the space of hypotheses and observations. By marginalization, this gives a prior on hypotheses, which is exactly the information we do not have and do not want to assume. One exception is measures of evidence that use the log-likelihood ratio, provided that there are only two hypotheses. In this case, rather than a prior on the space of hypotheses and observations, it suffices that, for each hypothesis h, there is a probability μ_h on observations, where, intuitively, $\mu_h(ob)$ is the probability of observing ob when h holds. Given an observation ob, the degree of confirmation that it provides for a hypothesis h is

$$l(ob, h) = \log\left(\frac{\mu_h(ob)}{\mu_{\overline{h}}(ob)}\right),$$

where $\overline{h}$ represents the hypothesis other than h (recall that this approach applies only if there are two hypotheses). Thus, the degree of confirmation is the ratio between these two probabilities. The use of the logarithm is not critical here. Using it ensures that the likelihood is positive if and only if the observation confirms the hypothesis. [1]

One problem with the log-likelihood ratio measure l as we have defined it is that it can be used only to reason about evidence discriminating between two competing hypotheses, namely between an hypothesis h holding and the hypothesis h not holding. We would like a measure of confirmation along the lines of the log-

[1] In the literature, confirmation is usually taken with respect to some background knowledge. For ease of exposition, we ignore background knowledge here, although it can easily be incorporated into the framework.

likelihood ratio measure, but that can handle multiple competing hypotheses. One such generalization was developed in the context of the Dempster-Shafer theory of evidence based on belief functions

Start with a finite set $\mathcal{H}$ of mutually exclusive and exhaustive hypotheses; thus, exactly one hypothesis holds at any given time. Let $\mathcal{O}$ be the set of possible observations (or pieces of evidence). For simplicity, assume that $\mathcal{O}$ is finite. Just as in the case of log-likelihood, assume that, for each hypotheses $h \in \mathcal{H}$, there is a probability measure μ_h on $\mathcal{O}$ such that $\mu_h(ob)$ is the probability of ob if hypothesis h holds. Define an evidence space (over $\mathcal{H}$ and $\mathcal{O}$) to be a tuple $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_h \mid h \in \mathcal{H}\})$.

Given an evidence space $\mathcal{E}$, define the weight that the observation ob lends to hypothesis h, written $w_{\mathcal{E}}(ob, h)$, as

$$w_{\mathcal{E}}(ob, h) = \frac{\mu_h(ob)}{\sum_{h' \in \mathcal{H}} \mu_{h'}(ob)}. \tag{4.2}$$

The weight of evidence $w_{\mathcal{E}}$ is not defined by Equation (4.2) for an observation ob such that $\sum_{h \in \mathcal{H}} \mu_h(ob) = 0$. Intuitively, this means that the observation ob is impossible. In the literature on confirmation theory it is typically assumed that this case never arises. More precisely, it is assumed that all observations are possible, so that for every observation ob, there is an hypothesis h such that $\mu_h(ob) > 0$. In the present case, making this assumption is unnatural. We want to view the answers given by knowledge algorithms as observations, and it seems perfectly reasonable to have a knowledge algorithm that never returns "?", for instance. As we shall see below, the fact that the weight of evidence is undefined in the case that $\sum_{h \in \mathcal{H}} \mu_h(ob) = 0$ is not a problem, thanks to the assumption that ν does not assign zero probability to the nonempty sets of sequences of coin tosses that determine the result of the knowledge algorithm.

For a set of hypotheses H, define $w_{\mathcal{E}}(ob, H)$ as simply the sum of $w_{\mathcal{E}}(ob, h)$ for $h \in H$. This definition makes $w_{\mathcal{E}}(ob, \cdot)$ a probability measure on hypotheses, for each fixed observation ob for which $\sum_{h \in \mathcal{H}} \mu_h(ob) > 0$. Intuitively, if $w_{\mathcal{E}}(ob, h) = 1$, then ob fully confirms h (i.e., h is certainly true if ob is observed), while if $w_{\mathcal{E}}(ob, h) = 0$, then ob disconfirms h (i.e., h is certainly false if ob is observed). Intuitively, the weight $w_{\mathcal{E}}(ob, h)$ is the probability that h is the right hypothesis in the light of observation ob.[2] The advantages of $w_{\mathcal{E}}$ over other known measures of confirmation are that (a) it is applicable when there is no prior probability distribution on the hypotheses, (b) it is applicable when there are more than two competing hypotheses, and (c) it has a fairly intuitive probabilistic interpretation.

[2] We could have taken the log of the ratio to make $w_{\mathcal{E}}$ more in line with the log-likelihood ratio l defined earlier, but there are technical advantages in having the weight of evidence be a number between 0 and 1.

Note that if $\mathcal{H} = \{h_1, h_2\}$, then $w_{\mathcal{E}}$ in some sense generalizes the log-likelihood ratio measure.[3] More precisely, for a fixed observation ob, $w_{\mathcal{E}}(ob, \cdot)$ induces the same relative order on hypotheses as $l(ob, \cdot)$, and for a fixed hypothesis h, $w_{\mathcal{E}}(\cdot, h)$ induces the same relative order on observations as $l(\cdot, h)$.

Theorem 4.3. *For all ob, we have $w_{\mathcal{E}}(ob, h_i) \geq w_{\mathcal{E}}(ob, h_{2-i})$ if and only if $l(ob, h_i) \geq l(ob, h_{2-i})$, for $i = 1, 2$, and for all h, ob, and ob', we have $w_{\mathcal{E}}(ob, h) \geq w_{\mathcal{E}}(ob', h)$ if and only if $l(ob, h) \geq l(ob', h)$.*

Although $w_{\mathcal{E}}(ob, \cdot)$ behaves like a probability measure on hypotheses for every observation ob (for which $\sum_{h \in \mathcal{H}} \mu_h(ob) > 0$), it is perhaps best not to think of it as a probability. Rather, it is an encoding of evidence. (We will see an interpretation of evidence in the next chapter.)

Example 4.4. For the primality example, the set $\mathcal{H}$ of hypotheses is $\{\mathsf{prime}, \neg\mathsf{prime}\}$. The observations $\mathcal{O}$ are simply the possible outputs of the knowledge algorithm $\mathtt{A}_{\mathrm{Alice}}$ on the formula prime, namely, {"Yes", "No"}. From the discussion following the description of the example, it follows that

$$\mu_{\mathsf{prime}}(\text{"Yes"}) = 1 \qquad \mu_{\mathsf{prime}}(\text{"No"}) = 0$$
$$\mu_{\neg\mathsf{prime}}(\text{"Yes"}) \leq \frac{1}{2} \qquad \mu_{\neg\mathsf{prime}}(\text{"No"}) \geq \frac{1}{2}.$$

(There is a drastic simplifying assumption here, namely, that the probability that the knowledge algorithm answers "Yes" to whether or not the number in Alice's local state is prime is the same for all values of the number. In reality, this is not the case. All we are really given is bounds on the probabilities μ_{prime} and $\mu_{\neg\mathsf{prime}}$; the actual probabilities vary with the actual prime number to which the knowledge algorithm is applied. For the remainder of this chapter, we assume that the probability associated with the answer of the knowledge algorithm for the primality example is the same for all numbers. Most knowledge algorithms will have that property.)

Let $\mathcal{E} = (\{\mathsf{prime}, \neg\mathsf{prime}\}, \{\text{"Yes"}, \text{"No"}\}, \{\mu_{\mathsf{prime}}, \mu_{\neg\mathsf{prime}}\})$ be the evidence

[3] Another representation that has similar characteristics is the following original representation of evidence via belief functions, defined as

$$w^S_{\mathcal{E}}(ob, H) = \frac{\max_{h \in H} \mu_h(ob)}{\max_{h \in \mathcal{H}} \mu_h(ob)}.$$

This measure is known in statistical hypothesis testing as the *generalized likelihood-ratio statistic*. It is another generalization of the log-likelihood ratio measure l. At this point, one may well ask what could help decide which weight function to use. In the case of $w_{\mathcal{E}}$ and $w^S_{\mathcal{E}}$, their main difference is in how they behave when one considers the combination of evidence. It can be argued that $w_{\mathcal{E}}$ behaves better in this case.

space capturing this situation. We can compute that

$$w_{\mathcal{E}}(\text{“Yes”}, \mathsf{prime}) \geq \frac{2}{3}$$

and

$$w_{\mathcal{E}}(\text{“Yes”}, \neg\mathsf{prime}) \leq \frac{1}{3}.$$

Intuitively, a “Yes” answer to the query prime provides more evidence for the hypothesis prime than the hypothesis ¬prime. Similarly, $w(\text{“No”}, \mathsf{prime}) = 0$ and $w(\text{“No”}, \neg\mathsf{prime}) = 1$. Thus, an output of “No” to the query prime indicates that the hypothesis ¬prime must hold. □

We can extend the definition of weight of evidence to *sets* of observations. If $obs \subseteq \mathcal{O}$ is a set of observations, define

$$w_{\mathcal{E}}(obs, h) = \frac{\mu_h(obs)}{\sum_{h' \in \mathcal{H}} mu_{h'}(obs)}.$$

Roughly speaking, $w_{\mathcal{E}}(obs, h)$ can be viewed as the weight of evidence provided by an observation that is compatible with all of the observations in obs and no other observations. For example, in an evidence space $(\mathcal{H}, \mathcal{O}, \mu_{h_0}, \mu_{\overline{h_0}})$, the weight of evidence that $\mathcal{O}$ provides for both h_0 and $\overline{h_0}$ is $1/2$, meaning that simply making an observation without any indication of the actual observation made does not provide support for one hypothesis more than to the other, as expected.

4.3 Reliable Randomized Knowledge Algorithms

What does a “Yes” answer to a query φ given by an “almost sound” knowledge algorithm tell us about φ? To make this precise, we need to first characterize how reliable the knowledge algorithm is. A randomized knowledge algorithm $\mathtt{A}_i$ is (α, β)-*reliable for* φ *in* N *(for agent* i*)* if $\alpha, \beta \in [0, 1]$ and for all worlds w and derandomizers v,

– $(N, w, v) \models \varphi$ implies $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \geq \alpha$,
– $(N, w, v) \models \neg\varphi$ implies $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \leq \beta$.

These conditions are equivalent to $(N, w, v) \models \varphi$ implying $(N, w, v) \models \Pr(X_i\varphi) \geq \alpha$ and $(N, w, v) \models \neg\varphi$ implying $(N, w, v) \models \Pr(X_i\varphi) \leq \beta$. In other words, if φ is true at world w, then an (α, β)-reliable algorithm says “Yes” to φ at w with probability at least α (and hence is right when it answers “Yes” to query φ with probability at least α); on the other hand, if φ is false, it says “Yes” with probability at most β (and hence is wrong when it answer “Yes” to query φ with probability at most β). The primality testing knowledge algorithm is $(1, \frac{1}{2})$-reliable for prime.

The intuition here is that (α, β)-reliability is a way to bound the probability that the knowledge algorithm is wrong. The knowledge algorithm can be wrong in two ways: it can answer "No" or "?" to a query φ when φ is true, and it can answer "Yes" to a query φ when φ is not true. If a knowledge algorithm is (α, β)-reliable, then the probability that it answers "No" or "?" when the answer should be "Yes" is at most $1 - \alpha$: the probability that it answers "Yes" when it should not is at most β.

We now associate an evidence space over the hypotheses $\{\varphi, \neg\varphi\}$ with a knowledge algorithm $\mathtt{A}_i$ and a local state ℓ for agent i. Let

$$\mathcal{E}_{\mathtt{A}_i,\varphi,\ell} = (\{\varphi, \neg\varphi\}, \{\text{“Yes”}, \text{“No”}, \text{“?”}\}, \mu_{\varphi,\ell}, \mu_{\neg\varphi,\ell}).$$

Roughly speaking, $\mu_{\varphi,\ell}(\text{“Yes”})$ (resp., $\mu_{\varphi,\ell}(\text{“No”})$; $\mu_{\varphi,\ell}(\text{“?”})$) is the probability that $\mathtt{A}_i$ says "Yes" (resp., "No"; "?") to a query φ at worlds where agent i has local state ℓ and φ is true, while $\mu_{\neg\varphi,\ell}(\text{“Yes”})$ (resp., $\mu_{\neg\varphi,\ell}(\text{“No”})$; $\mu_{\neg\varphi,\ell}(\text{“?”})$) is the probability that $\mathtt{A}_i$ answers "Yes" (respectively, "No"; "?") to the query φ at worlds where agent i has local state ℓ and $\neg\varphi$ is true. More precisely, for the given agent i, define $W_{\varphi,\ell} = \{(w, v) \mid \mathcal{V}_i(w) = \ell, (N, w, v) \models \varphi\}$ and $W_{\neg\varphi,\ell} = \{(w, v) \mid \mathcal{V}_i(w) = \ell, (N, w, v) \models \neg\varphi\}$, and take

$$\mu_{\varphi,\ell}(\text{“Yes”}) = \begin{cases} \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, v_i') = \text{“Yes”}\}) & \text{if } W_{\varphi,\ell} \neq \varnothing \\ 1 & \text{otherwise} \end{cases}$$

$$\mu_{\varphi,\ell}(\text{“No”}) = \begin{cases} \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, v_i') = \text{“No”}\}) & \text{if } W_{\varphi,\ell} \neq \varnothing \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{\varphi,\ell}(\text{“?”}) = 1 - \mu_{\varphi,\ell}(\text{“Yes”}) - \mu_{\varphi,\ell}(\text{“No”}).$$

Similarly, take

$$\mu_{\neg\varphi,\ell}(\text{“Yes”}) = \begin{cases} \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, v_i') = \text{“Yes”}\}) & \text{if } W_{\neg\varphi,\ell} \neq \varnothing \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{\neg\varphi,\ell}(\text{“No”}) = \begin{cases} \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, v_i') = \text{“No”}\}) & \text{if } W_{\neg\varphi,\ell} \neq \varnothing \\ 1 & \text{otherwise} \end{cases}$$

$$\mu_{\neg\varphi,\ell}(\text{“?”}) = 1 - \mu_{\neg\varphi,\ell}(\text{“Yes”}) - \mu_{\neg\varphi,\ell}(\text{“No”}).$$

If $\mathtt{A}_i$ is (α, β)-reliable for φ in N, then $\mu_{\varphi,\ell}(\text{“Yes”}) \geq \alpha$, $\mu_{\varphi,\ell}(\{\text{“No”}, \text{“?”}\}) \leq 1 - \alpha$, $\mu_{\neg\varphi,\ell}(\text{“Yes”}) \leq \beta$, and $\mu_{\neg\varphi,\ell}(\{\text{“No”}, \text{“?”}\}) \geq 1 - \beta$, for all local state ℓ for agent i.

To be able to talk about evidence within the logic, we introduce an operator to capture the evidence provided by the knowledge algorithm of agent i, $\mathrm{Ev}_i(\varphi)$, read "the weight of evidence that i would get for φ if he were to query the knowledge

algorithm about φ". There is a subtlety here, due to the asymmetry in the definition of reliability. Reliability talks only about the probability that the algorithm returns "Yes", and thus does not distinguish between the algorithm returning "No" or "?". In order to establish a link between reliability and evidence, it is convenient to identify the answers "No" and "?", using the notation $\ulcorner ob \urcorner$ to represent which of the sets {"Yes"} and {"No", "?"} the observation ob belongs to. Formally, define

$$\begin{aligned} \ulcorner \text{"Yes"} \urcorner &= \{\text{"Yes"}\} \\ \ulcorner \text{"No"} \urcorner &= \{\text{"No"}, \text{"?"}\} \\ \ulcorner \text{"?"} \urcorner &= \{\text{"No"}, \text{"?"}\}. \end{aligned}$$

We can now give the semantics of the Ev_i operator:

$$(N, w, v) \models \mathrm{Ev}_i(\varphi) \geq \alpha \text{ if } w_{\mathcal{E}_{\mathtt{A}_i, \varphi, \mathcal{V}_i(w)}}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner, \varphi) \geq \alpha.$$

We can similarly define $(N, w, v) \models \mathrm{Ev}_i(\varphi) \leq \alpha$. Thus, the Ev_i operator captures the evidence for φ given by the answer of i's knowledge algorithm to a query φ. $\mathrm{Ev}_i(\varphi)$ is always defined, despite the weight of evidence not being defined in the case where $\mu_{\varphi,\ell}(ob) + \mu_{\neg\varphi,\ell}(ob) = 0$.

Theorem 4.5. *For all probabilistic algorithmic knowledge structures N, worlds w of N, and derandomizers $v \in V$, $w_{\mathcal{E}_{\mathtt{A}_i, \varphi, \mathcal{V}_i(w)}}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner, \varphi)$ is defined.*

Intuitively, since the knolwedge algorithm $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)$ returns a result ob at world w, our restriction on ν guarantees that the probability of observing ob must be nonzero.

This definition of evidence has a number of interesting properties. For instance, full evidence in support of a formula φ essentially corresponds to knowledge of φ, as the following result show.

Theorem 4.6. *For all probabilistic algorithmic knowledge structures N, we have*

$$N \models \mathrm{Ev}_i(\varphi) = 1 \Rightarrow K_i\varphi.$$

The following theorem captures the relationship between reliable knowledge algorithms and evidence.

Theorem 4.7. *If $\mathtt{A}_i$ is (α, β)-reliable for φ in N then*

(a) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta}$ *if* $(\alpha, \beta) \neq (0, 0)$*;*
(b) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0, 0)$*;*
(c) $N \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $(\alpha, \beta) \neq (1, 1)$*;*
(d) $N \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 0$ *if* $(\alpha, \beta) = (1, 1)$.

In other words, if a randomized knowledge algorithm says "Yes", then that provides evidence for φ being true. The weight of evidence depends on the reliability. Similarly, if the randomized knowledge algorithm says "No", there is less evidence in favor of φ being true.

Theorem 4.7 becomes interesting in the context of well-known classes of randomized algorithms. An **RP** (random polynomial-time) algorithm is a polynomial-time randomized algorithm that is $(\frac{1}{2}, 0)$-reliable. It thus follows from Theorem 4.7 that if $\mathtt{A}_i$ is an **RP** algorithm, then $X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 1$ and $\neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1}{3}$ are both valid in N. By Theorem 4.6, $\mathrm{Ev}_i(\varphi) \Rightarrow K_i\varphi$ is valid, and thus we have $X_i\varphi \Rightarrow K_i\varphi$, as expected. Similarly, a **BPP** (bounded-error probabilistic polynomial-time) algorithm is a polynomial-time randomized algorithm that is $(\frac{3}{4}, \frac{1}{4})$-reliable. Thus, by Theorem 4.7, if $\mathtt{A}_i$ is a **BPP** algorithm, then $X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \frac{3}{4}$ and $\neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1}{4}$ are both valid in N.

Notice that Proposition 4.7 talks about the evidence that the knowledge algorithm provides for φ. Intuitively, we might expect some kind of relationship between the evidence for φ and the evidence for $\neg\varphi$. A plausible relationship would be that high evidence for φ implies low evidence for $\neg\varphi$, and low evidence for φ implies high evidence for $\neg\varphi$. Unfortunately, given the definitions in this section, this is not the case. Evidence for φ is completely unrelated to evidence for $\neg\varphi$. Roughly speaking, this is because evidence for φ is measured by looking at the results of the knowledge algorithm when queried for φ, and evidence for $\neg\varphi$ is measured by looking at the results of the knowledge algorithm when queried for $\neg\varphi$. However, there is nothing in the definition of a knowledge algorithm that says that the answers of the knowledge algorithm to queries φ and $\neg\varphi$ need to be related in any way.

A relationship between evidence for φ and evidence for $\neg\varphi$ can be established by considering knowledge algorithms that are "well-behaved" with respect to negation. We already saw special classes of knowledge algorithms in terms of how they behave with respect to negation in Section 2.4. We can adapt these notions to the randomized knowledge algorithm case, in the obvious way. Say that a randomized knowledge algorithm $\mathtt{A}$ *weakly respects negation* if for all local states ℓ and derandomizers v:

$$\mathtt{A}^d(\neg\varphi, \ell, v_i) = \begin{cases} \text{"Yes"} & \text{if } \mathtt{A}^d(\varphi, \ell, v_i) = \text{"No"} \\ \text{"No"} & \text{if } \mathtt{A}^d(\varphi, \ell, v_i) = \text{"Yes"} \\ \text{"?"} & \text{if } \mathtt{A}^d(\varphi, \ell, v_i) = \text{"?"}. \end{cases}$$

Similarly, say that a randomized knowledge algorithm $\mathtt{A}$ *strongly respects negation* if for all local states ℓ and derandomizers v:

$$\mathtt{A}^d(\neg\varphi, \ell, v_i) = \begin{cases} \text{“Yes”} & \text{if } \mathtt{A}^d(\varphi, \ell, v_i) \neq \text{“Yes”} \\ \text{“No”} & \text{if } \mathtt{A}^d(\varphi, \ell, v_i) = \text{“Yes”}. \end{cases}$$

The following result shows that for knowledge algorithms that respect negation, reliability for φ is related to reliability for $\neg\varphi$:

Theorem 4.8. *If $\mathtt{A}_i$ weakly respects negation and $\mathtt{A}_i$ is (α, β)-reliable for φ in N, then $\mathtt{A}_i$ is $(0, 1-\alpha)$-reliable for $\neg\varphi$ in N. If $\mathtt{A}_i$ strongly respects negation, then $\mathtt{A}_i$ is (α, β)-reliable for φ in N if and only if $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N.*

It is not hard to construct examples showing that the reliability assessments in Proposition 4.8 in the case where $\mathtt{A}_i$ weakly respects negation are tight; for instance, we can exhibit a knowledge algorithm weakly respecting negation that is (α, β)-reliable for φ in N and that is not $(\epsilon_1, 1-\beta-\epsilon_2)$-reliable for $\neg\varphi$ in N, for any $\epsilon_1, \epsilon_2 > 0$. Adapting the proof of Theorem 2.6, it is easy to check that if $\mathtt{A}_i$ weakly respects negation, then $X_i\varphi \Rightarrow \neg X_i\neg\varphi$ is a valid formula. Similarly, if $\mathtt{A}_i$ strongly respects negation, then $X_i\varphi \Leftrightarrow \neg X_i\neg\varphi$ is a valid formula. Combined with Theorem 4.7, this yields the following results.

Theorem 4.9. *If $\mathtt{A}_i$ is (α, β)-reliable for φ in N and $\mathtt{A}_i$ weakly respects negation, then*

(a) $N \models X_i\varphi \Rightarrow \left(\mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \mathrm{Ev}_i(\neg\varphi) \leq \frac{1}{1+\alpha}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*
(b) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0, 0)$*;*
(c) $N \models X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $\alpha \neq 1$*;*
(d) $N \models X_i\neg\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) = 1 \wedge \mathrm{Ev}_i(\varphi) = 0)$ *if* $\alpha = 1$.

Theorem 4.10. *If $\mathtt{A}_i$ is (α, β)-reliable for φ in N and $\mathtt{A}_i$ strongly respects negation, then*

(a) $N \models X_i\varphi \Rightarrow \left(\mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \mathrm{Ev}_i(\neg\varphi) \leq \frac{\beta}{\alpha+\beta}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*
(b) $N \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) = 1 \wedge \mathrm{Ev}_i(\neg\varphi) = 0)$ *if* $(\alpha, \beta) = (0, 0)$*;*
(c) $N \models X_i\neg\varphi \Rightarrow \left(\mathrm{Ev}_i(\neg\varphi) \geq \frac{1-\beta}{2-(\alpha+\beta)} \wedge \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}\right)$ *if* $(\alpha, \beta) \neq (1, 1)$*;*
(d) $N \models X_i\neg\varphi \Rightarrow \left(\mathrm{Ev}_i(\neg\varphi) \geq \frac{1}{2} \wedge \mathrm{Ev}_i(\varphi) \leq \frac{1}{2}\right)$ *if* $(\alpha, \beta) = (1, 1)$.

One goal of this chapter was to understand what the evidence provided by a knowledge algorithm tells us. To take an example from security, consider an enforcement mechanism used to detect and react to intrusions in a system. Such an

enforcement mechanism uses algorithms that analyze the behaviour of users and attempt to recognize intruders. While the algorithms may sometimes be wrong, they are typically reliable, with some associated probabilities. Clearly the mechanism wants to make sensible decisions based on this information, How should it do this? What actions should the system take based on a report that a user is an intruder? Resolving this asks for an interpretation of evidence, which we attempt to provide in the next chapter.

Notes

The work in this chapter first appeared in [Halpern and Pucella 2003c].

Suitable introductions to probability theory include [Feller 1957; Billingsley 1995]. A good overview of the theory of randomized algorithms is [Motwani and Raghavan 1995], where the classes **RP** and **BPP** are described in details. Rabin's probabilistic procedure is developed in [Rabin 1980]

Much of the work on evidence in philosophy arises in philosophy of science, specifically *confirmation theory*, where the concern has been historically to assess the support that evidence obtained through experimentation lends to various scientific theories [Carnap 1962; Popper 1959]. Kyburg [1983] gives a good survey of the various approaches. The definition of weight of evidence we use in this chapter, $w_{\mathcal{E}}$, was introduced by Shafer [1982], in the context of the Dempster-Shafer theory of evidence based on belief functions [Shafer 1976]; it was further studied by Walley [1987]. Our description is taken mostly from [Halpern and Fagin 1992]. The alternative $w^S_{\mathcal{E}}$ is Shafer's original measure [Shafer 1976]. A comparison between $w_{\mathcal{E}}$ and $w^S_{\mathcal{E}}$ can be found in [Walley 1987; Halpern and Fagin 1992].

Other classes of confirmation measures, that do assume a prior probability, include the difference measures of Jeffrey [1992]. Another class, the log ratio measures, has been advocated, among others, by Milne [1996]. The log-likelihood ratio measures have been advocated, among others, by Good [1950, 1960]. It can be shown that those functions are *ordinally distinct*, that is, they will support different hypotheses given the same hypotheses and evidence [Fitelson 1999].

5
Reasoning about Evidence

CONSIDER the following situation. A coin is tossed, which is either fair or double-headed. The coin lands heads. How likely is it that the coin is double-headed? What if the coin is tossed 20 times and it lands heads each time? Intuitively, it is much more likely that the coin is double-headed in the latter case than in the former. But how should the likelihood be measured? A straightforward application of probability theory is not possible here. We cannot compute the probability of the coin being double-headed; assigning a probability to that event requires that we have a prior probability on the coin being double-headed. For example, if the coin was chosen at random from a barrel with one billion fair coins and one double-headed coin, it is still overwhelmingly likely that the coin is fair, and that the sequence of 20 heads is just unlucky. However, in the problem statement, there is no prior probability mentioned. We could of course posit a prior probability and see how the posterior probability behaves when we change the prior, but the point is that our intuition does not seem to rely on such a posited prior.

The main feature of this situation is that it involves a combination of probabilistic outcomes (e.g., the coin tosses) and nonprobabilistic outcomes (e.g., the choice of the coin). There has been a great deal of work on reasoning about systems that combine both probabilistic and nondeterministic choices. However, the observations above suggest that if we attempt to formally analyze this situation in one of those frameworks, which essentially permit only the modeling of probabilities, we will not be able to directly capture this intuition about increasing likelihood. To see how this plays out, consider a formal analysis of the situation in the framework due to Joseph Halpern and Mark Tuttle. Suppose that Alice nonprobabilistically chooses one of two coins: a fair coin with probability $1/2$ of landing heads, or a double-headed coin with probability 1 of landing heads. Alice tosses this coin repeatedly. Let φ_k be a formula stating: "the kth coin toss lands heads". What is the probability of φ_k according to Bob, who does not know which coin Alice chose?

According to the Halpern-Tuttle framework, this can be modeled by considering the set of runs describing the states of the system at each point in time, and partitioning this set into two subsets, one for each coin used. In the set of runs where the fair coin is used, the probability of φ_k is $1/2$; in the set of runs where the double-headed coin is used, the probability of φ_k is 1. In this setting, the only conclusion that can be drawn is $(\Pr_B(\varphi_k) = 1/2) \vee (\Pr_B(\varphi_k) = 1)$. (This is of course the probability from Bob's point of view; Alice presumably knows which coin she is using.) Intuitively, this seems reasonable: if the fair coin is chosen, the probability that the kth coin toss lands heads, according to Bob, is $1/2$; if the double-headed coin is chosen, the probability is 1. Since Bob does not know which of the coins is being used, that is all that can be said.

But now suppose that, before the 101st coin toss, Bob learns the result of the first 100 tosses. Suppose, moreover, that all of these landed heads. What is the probability that the 101st coin toss lands heads? By the same analysis, it is still either $1/2$ or 1, depending on which coin is used.

This is hardly useful. To make matters worse, no matter how many coin tosses Bob witnesses, the probability that the next toss lands heads remains unchanged. But this answer misses out on some important information. And this information is exactly the kind of information provided by reliable knowledge algorithms in the last chapter. The fact that all of the first 100 coin tosses are heads is very strong *evidence* that the coin is in fact double-headed. Indeed, a straightforward computation using Bayes' Rule shows that if the prior probability of the coin being double-headed is α, then after observing that all of the 100 tosses land heads, the probability of the coin being double-headed becomes

$$\frac{\alpha}{\alpha + 2^{-100}(1-\alpha)} = \frac{2^{100}\alpha}{2^{100}\alpha + (1-\alpha)}.$$

However, note that it is not possible to determine the posterior probability that the coin is double-headed (or that the 101st coin toss is heads) without the prior probability α. After all, if Alice chooses the double-headed coin with probability only 10^{-100}, then it is still overwhelmingly likely that the coin used is in fact fair, and that Bob was just very unlucky to see such a nonrepresentative sequence of coin tosses.

We am not aware of any logical framework for reasoning about nondeterminism and probability that takes the issue of evidence into account. On the other hand, as we saw in the last chapter, evidence has been discussed extensively in the philosophical literature.

In this chapter, we introduce a logic for reasoning about evidence that extends existing logics for reasoning about likelihood expressed as either probability or belief. The logic has first-order quantification over the reals (so includes the theory

of real closed fields), for reasons that will shortly become clear. By adding observations to the states, it is possible to refine the language to talk about both the prior probability of hypotheses and the posterior probability of hypotheses, taking into consideration the observation at the states. We provide an additional operator to talk about the evidence provided by particular observations. This lets us write formulas that talk about the relationship between the prior probabilities, the posterior probabilities, and the evidence provided by the observations.

We then provide a sound and complete axiomatization for the logic. To obtain such an axiomatization, first-order quantification appears necessary. Roughly speaking, this is because ensuring that the evidence operator has the appropriate properties requires us to assert the existence of suitable probability measures. It does not seem possible to do this without existential quantification. Finally, we consider the complexity of the satisfiability problem. The complexity problem for the full language requires exponential space, since it incorporates the theory of real closed fields, for which an exponential space lower bound is known. However, we show that the satisfiability problem for a propositional fragment of the language, which is still strong enough to allow us to express many properties of interest, is NP-complete.

5.1 Evidence and Probability Updates

In order to develop a logic for reasoning about evidence, we need to first formalize an appropriate notion of evidence. We described a particular form of evidence in Section 4.2, and this is the notion that we take as underlying our logic. But what is evidence? Evidence can be thought of as a *function* mapping a prior probability on the hypotheses to a posterior probability, based on the piece of evidence witnessed. There is a precise sense in which $w_{\mathcal{E}}$ can be viewed as a function that maps a prior probability μ_0 on the hypotheses $\mathcal{H}$ to a posterior probability μ_{ob} based on observing ob, by applying Dempster's Rule of Combination. That is,

$$\mu_{ob} = \mu_0 \oplus w_{\mathcal{E}}(ob, \cdot), \tag{5.1}$$

where $\oplus$ combines two probability distributions on $\mathcal{H}$ to get a new probability distribution on $\mathcal{H}$ defined as follows:

$$(\mu_1 \oplus \mu_2)(H) = \frac{\sum_{h \in H} \mu_1(h)\mu_2(h)}{\sum_{h \in \mathcal{H}} \mu_1(h)\mu_2(h)}.$$

Bayes' Rule is the standard way of updating a prior probability based on an observation, but it is only applicable when we have a joint probability distribution on both the hypotheses and the observations, something which we did not assume we had. Dempster's Rule of Combination essentially "simulates" the effects of

Bayes's rule. The relationship between Dempster's Rule and Bayes' Rule is made precise by the following well-known theorem.

Theorem 5.1. *Let $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_h \mid h \in \mathcal{H}\})$ be an evidence space. Suppose that P is a probability on $\mathcal{H} \times \mathcal{O}$ such that $P(\mathcal{H} \times \{ob\} | \{h\} \times \mathcal{O}) = \mu_h(ob)$ for all $h \in \mathcal{H}$ and all $ob \in \mathcal{O}$. Let μ_0 be the probability on $\mathcal{H}$ induced by marginalizing P; that is, $\mu_0(h) = P(\{h\} \times \mathcal{O})$. For $ob \in \mathcal{O}$, let $\mu_{ob} = \mu_0 \oplus w_{\mathcal{E}}(ob, \cdot)$. Then $\mu_{ob}(h) = \Pr(\{h\} \times \mathcal{O} | \mathcal{H} \times \{ob\})$.*

In other words, when we do have a joint probability on the hypotheses and observations, then Dempster's Rule of Combination gives us the same result as a straightforward application of Bayes' Rule.

Example 5.2. To get a feel for how this measure of evidence can be used, consider a variation of the two-coins example in the introduction. Assume that the coin chosen by Alice is either double-headed or fair, and consider sequences of hundred tosses of that coin. Let $\mathcal{O} = \{m \mid 0 \leq m \leq 100\}$ (the number of heads observed), and let $\mathcal{H} = \{F, D\}$, where F is "the coin is fair", and D is "the coin is double-headed". The probability spaces associated with the hypotheses are generated by the following probabilities for simple observations m:

$$\mu_F(m) = \frac{1}{2^{100}} \binom{100}{m} \qquad \mu_D(m) = \begin{cases} 1 & \text{if } m = 100 \\ 0 & \text{otherwise.} \end{cases}$$

(We extend by additivity to the whole space $\mathcal{O}$.) Take $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_F, \mu_D\})$. For any observation $m \neq 100$, the weight in favor of F is given by

$$w_{\mathcal{E}}(m, F) = \frac{\frac{1}{2^{100}} \binom{100}{m}}{0 + \frac{1}{2^{100}} \binom{100}{m}} = 1,$$

which means that the support of m is unconditionally provided to F; indeed, any such sequence of tosses cannot appear with the double-headed coin. Thus, if $m > 0$, we get that

$$w_{\mathcal{E}}(m, D) = \frac{0}{0 + \frac{1}{2^{100}} \binom{100}{m}} = 0.$$

What happens when the hundred coin tosses are all heads? It is straightforward to check that

$$w_{\mathcal{E}}(100, F) = \frac{\frac{1}{2^{100}}}{1 + \frac{1}{2^{100}}} = \frac{1}{1 + 2^{100}} \qquad w_{\mathcal{E}}(100, D) = \frac{1}{1 + \frac{1}{2^{100}}} = \frac{2^{100}}{1 + 2^{100}};$$

this time there is overwhelmingly more evidence in favor of D than F.

Note that we have not assumed any prior probability on the hypotheses. Thus,

we cannot talk about the probability that the coin is fair or double-headed. What we have is a quantitative assessment of the evidence in favor of one of the hypotheses. However, if we assume a prior probability α on the coin being fair and m heads are observed after 100 tosses, then the probability that the coin is fair is 1 if $m \neq 100$; if $m = 100$ then, applying the rule of combination, the posterior probability of the coin being fair is $\alpha/(\alpha + (1-\alpha)2^{100})$. □

Can we characterize weight functions using a small number of properties? More precisely, given sets $\mathcal{H}$ and $\mathcal{O}$, and a function f from $\mathcal{O} \times \mathcal{H}$ to $[0,1]$, are there properties of f that ensure that there are probability measures $\{\mu_h \mid h \in \mathcal{H}\}$ such that $f = w_{\mathcal{E}}$? As we saw earlier, for a fixed observation ob, f acts like a probability measure on $\mathcal{H}$. However, this is not sufficient to guarantee that f is a weight function. Consider the following example, with $\mathcal{O} = \{ob_1, ob_2\}$ and $\mathcal{H} = \{h_1, h_2, h_3\}$:

$$\begin{array}{lcllcl} f(ob_1,h_1) & = & 1/4 & f(ob_2,h_1) & = & 1/4 \\ f(ob_1,h_2) & = & 1/4 & f(ob_2,h_2) & = & 1/2 \\ f(ob_1,h_3) & = & 1/2 & f(ob_2,h_3) & = & 1/4. \end{array}$$

It is straightforward to check that $f(ob_1,\cdot)$ and $f(ob_2,\cdot)$ are probability measures on $\mathcal{H}$, but that there is no evidence space $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_{h_1}, \mu_{h_2}, \mu_{h_3}\})$ such that $f = w_{\mathcal{E}}$. Indeed, assume that we do have such $\mu_{h_1}, \mu_{h_2}, \mu_{h_3}$. By the definition of weight of evidence, and the fact that f is that weight of evidence, we get the following system of equations:

$$\begin{array}{ll} \frac{\mu_{h_1}(ob_1)}{\mu_{h_1}(ob_1)+\mu_{h_2}(ob_1)+\mu_{h_3}(ob_1)} = 1/4 & \frac{\mu_{h_1}(ob_2)}{\mu_{h_1}(ob_2)+\mu_{h_2}(ob_2)+\mu_{h_3}(ob_2)} = 1/4 \\ \frac{\mu_{h_2}(ob_1)}{\mu_{h_1}(ob_1)+\mu_{h_2}(ob_1)+\mu_{h_3}(ob_1)} = 1/4 & \frac{\mu_{h_2}(ob_2)}{\mu_{h_1}(ob_2)+\mu_{h_2}(ob_2)+\mu_{h_3}(ob_2)} = 1/2 \\ \frac{\mu_{h_3}(ob_1)}{\mu_{h_1}(ob_1)+\mu_{h_2}(ob_1)+\mu_{h_3}(ob_1)} = 1/2 & \frac{\mu_{h_3}(ob_2)}{\mu_{h_1}(ob_2)+\mu_{h_2}(ob_2)+\mu_{h_3}(ob_2)} = 1/4. \end{array}$$

It is now immediate that there exist α_1 and α_2 such that $\mu_{h_i}(ob_j) = \alpha_j f(ob_j, h_i)$, for $i = 1,2,3$. Indeed, $\alpha_j = \mu_{h_1}(ob_j) + \mu_{h_2}(ob_j) + \mu_{h_3}(ob_j)$, for $j = 1,2$. Moreover, since μ_{h_i} is a probability measure, we must have that

$$\mu_{h_i}(ob_1) + \mu_{h_1}(ob_2) = \alpha_1 f(ob_1, h_i) + \alpha_2 f(ob_2, h_i) = 1,$$

for $i = 1,2,3$. Thus,

$$\alpha_1/4 + \alpha_2/4 = \alpha_1/4 + \alpha_2/2 = \alpha_1/2 + \alpha_4/4 = 1.$$

These constraints are easily seen to be unsatisfiable.

This argument generalizes to arbitrary functions f; thus, a necessary condition for f to be a weight function is that there exists α_i for each observation ob_i such that $\mu_h(ob_i) = \alpha_i f(ob_i, h)$ for each hypothesis h is a probability measure, that is,

$\alpha_1 f(ob_1, h) + \cdots + \alpha_k f(ob_k, h) = 1$. In fact, when combined with the constraint that $f(ob, \cdot)$ is a probability measure for a fixed ob, this condition turns out to be sufficient, as the following theorem establishes.

Theorem 5.3. *Let $\mathcal{H} = \{h_1, \ldots, h_m\}$ and $\mathcal{O} = \{ob_1, \ldots, ob_n\}$, and let f be a real-valued function with domain $\mathcal{O} \times \mathcal{H}$ such that $f(ob, h) \in [0, 1]$. Then there exists an evidence space $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \mu_{h_1}, \ldots, \mu_{h_m})$ such that $f = w_{\mathcal{E}}$ if and only if f satisfies the following properties:*

WF1. *For every $ob \in \mathcal{O}$, $f(ob, \cdot)$ is a probability measure on $\mathcal{H}$.*

WF2. *There exists $x_1, \ldots, x_n > 0$ such that, for all $h \in \mathcal{H}$, $\sum_{i=1}^{n} f(ob_i, h) x_i = 1$.*

This characterization is fundamental to the completeness of the axiomatization of the logic we introduce in the next section.

5.2 Reasoning about Evidence

We introduce a logic $\mathcal{L}^{fo\text{-}ev}$ for reasoning about evidence. The logic has both propositional features and first-order features. It takes the probability of propositions, and views evidence as a proposition. On the other hand, it allows first-order quantification over numerical quantities, such as probabilities and evidence. The logic essentially considers two time periods, which can be thought of as the time before an observation is made and the time after an observation is made. For simplicity, assume that exactly one observation is made. Thus, we can talk of the probability of a formula φ before an observation is made, denoted $\Pr^0(\varphi)$, the probability of φ after the observation, denoted $\Pr(\varphi)$, and the evidence provided by the observation ob for a hypothesis h, denoted $\mathrm{w}(ob, h)$. Of course, we want to be able to use the logic to relate all these quantities.

Formally, start with two sets of primitive propositions, $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_{n_{\mathsf{h}}}\}$ representing the hypotheses, and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_{n_{\mathsf{o}}}\}$ representing the observations. Let $\mathcal{L}_{\mathsf{h}}(\Phi_{\mathsf{h}})$ be the propositional sublanguage of *hypothesis formulas* obtained by taking primitive propositions in Φ_{h} and closing off under negation and conjunction; we use ρ to range over formulas of that sublanguage.

A *basic term* has the form $\Pr^0(\rho)$, $\Pr(\rho)$, or $\mathrm{w}(ob, \rho)$, where ρ is an hypothesis formula, and ob is an observation. A *polynomial term* has the form $t_1 + \cdots + t_n$, where each term t_i is a product of integers, basic terms, and variables (which range over the reals). A *polynomial inequality formula* has the form $p \geq c$, where p is a polynomial term and c is an integer. Let $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ be the language obtained by starting out with the primitive propositions in Φ_{h} and Φ_{o} and polynomial inequality formulas, and closing off under conjunction, negation, and first-order

quantification. Let $true$ be an abbreviation for an arbitrary propositional tautology involving only hypotheses, such as $h_1 \vee \neg h_1$; let $false$ be an abbreviation for $\neg true$. With this definition, $true$ and $false$ can be considered as part of the sublanguage $\mathcal{L}_{\mathsf{h}}(\Phi_{\mathsf{h}})$.

It should be clear that while only integers coefficients are allowed to appear in polynomial terms, it is possible in fact to express polynomial terms with rational coefficients by crossmultiplying. For instance, $\frac{1}{3}\Pr(\rho) + \frac{1}{2}\Pr(\rho') \geq 1$ can be represented by the polynomial inequality formula $2\Pr(\rho) + 3\Pr(\rho') \geq 6$. While there is no difficulty in giving a semantics to polynomial terms that use arbitrary real coefficients, the restriction to integers is necessary in order to make use of results from the theory of real closed fields in both the axiomatization of Section 5.3 and the complexity results of Section 5.4.

We use obvious abbreviations where needed, such as $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \Rightarrow \psi$ for $\neg\varphi \vee \psi$, $\exists x\varphi$ for $\neg\forall x(\neg\varphi)$, $\Pr(\varphi) - \Pr(\psi) \geq c$ for $\Pr(\varphi) + (-1)\Pr(\psi) \geq c$, $\Pr(\varphi) \geq \Pr(\psi)$ for $\Pr(\varphi) - \Pr(\psi) \geq 0$, $\Pr(\varphi) \leq c$ for $-\Pr(\varphi) \geq -c$, $\Pr(\varphi) < c$ for $\neg(\Pr(\varphi) \geq c)$, and $\Pr(\varphi) = c$ for $(\Pr(\varphi) \geq c) \wedge (\Pr(\varphi) \leq c)$ (and analogous abbreviations for inequalities involving $\Pr^0$ and w).

Example 5.4. Consider again the situation given in Example 5.2. Let Φ_{o}, the observations, consist of primitive propositions of the form $\mathsf{heads}[m]$, where m is an integer with $0 \leq m \leq 100$, indicating that m heads out of 100 tosses have appeared. Let Φ_{h} consist of the two primitive propositions fair and $\mathsf{doubleheaded}$. The computations in Example 5.2 can be written as follows:

$$\mathrm{w}(\mathsf{heads}[100], \mathsf{fair}) = 1/(1 + 2^{100}) \wedge \\ \mathrm{w}(\mathsf{heads}[100], \mathsf{doubleheaded}) = 2^{100}/(1 + 2^{100}).$$

We can also capture the fact that the weight of evidence of an observation maps a prior probability into a posterior probability by Dempster's Rule of Combination. For example, the following formula captures the update of the prior probability α of the hypothesis fair upon observation of a hundred coin tosses landing heads:

$$\Pr^0(\mathsf{fair}) = \alpha \wedge \mathrm{w}(\mathsf{heads}[100], \mathsf{fair}) = 1/(1 + 2^{100}) \Rightarrow \\ \Pr(\mathsf{fair}) = \alpha/(\alpha + (1 - \alpha)2^{100}).$$

We develop a deductive system to derive such conclusions in the next section. □

Now we consider the semantics. As usual, a model is a set of possible worlds. A world describes which hypothesis is true and which observation was made (recall that we have assumed that exactly one hypothesis is true, and exactly one observation is made), together with a probability distribution describing the prior probability, which is used to interpret $\Pr^0$. Thus, a world has the form (h, ob, μ),

where h is a hypothesis, ob is an observation, and μ is a probability distribution on Φ_h. In addition, to interpret w, we need an evidence space over Φ_{h} and Φ_{o}, which gives a probability measure μ_h on Φ_{o} for each hypothesis $h \in \Phi_{\mathsf{h}}$. Thus, take an *evidential structure* M to be a tuple $(S \times \mathcal{P}, \mathcal{E})$, where $S \subseteq \Phi_{\mathsf{h}} \times \Phi_{\mathsf{o}}$, $\mathcal{P}$ is a set of probability distributions[1] on Φ_{h}, and $\mathcal{E}$ is an evidence space over Φ_{h} and Φ_{o}. Note that the states of the structure are required to be only a subset of $\Phi_{\mathsf{h}} \times \Phi_{\mathsf{o}}$. This allows us to rule out particular combinations of hypothesis and observation.

To interpret propositional formulas in $\mathcal{L}_{\mathsf{h}}(\Phi_{\mathsf{h}})$, associate with each hypothesis formula ρ a set $[\![\rho]\!]$ of hypotheses, by induction on the structure of ρ:

$$\begin{aligned} [\![h]\!] &= \{h\} \\ [\![\neg\rho]\!] &= \Phi_{\mathsf{h}} - [\![\rho]\!] \\ [\![\rho_1 \wedge \rho_2]\!] &= [\![\rho_1]\!] \cap [\![\rho_2]\!]. \end{aligned}$$

To interpret first-order formulas that may contain variables, we need a valuation v that assigns a real number to every variable. Given a valuation v, an evidential structure $M = (S \times \mathcal{P}, \mathcal{E})$, and a world $w = (h, ob, \mu)$, we can assign to a polynomial term p a real number $[p]^{M,w,v}$ in a straightforward way:

$$\begin{aligned} [x]^{M,w,v} &= v(x) \\ [a]^{M,w,v} &= a \\ [\mathrm{Pr}^0(\rho)]^{M,w,v} &= \mu([\![\rho]\!]) \\ [\mathrm{Pr}(\rho)]^{M,w,v} &= (\mu \oplus w_{\mathcal{E}}(ob, \cdot))([\![\rho]\!]) \\ [\mathrm{w}(ob', \rho)]^{M,w,v} &= w_{\mathcal{E}}(ob', [\![\rho]\!]) \\ [t_1 t_2]^{M,w,v} &= [t_1]^{M,w,v} \times [t_2]^{M,w,v} \\ [p_1 + p_2]^{M,w,v} &= [p_1]^{M,w,v} + [p_2]^{M,w,v}. \end{aligned}$$

Note that, to interpret $\mathrm{Pr}(\rho)$, the posterior probability of ρ after having observed ob (the observation at world w), we use Equation (5.1), which says that the posterior is obtained by combining the prior probability μ with $w_{\mathcal{E}}(ob, \cdot)$.

We define what it means for a formula φ to be true (or satisfied) at a world w of an evidential structure $M = (S \times \mathcal{P}, \mathcal{E})$ under valuation v, written $(M, w, v) \models \varphi$, as follows:

$(M, w, v) \models h$ if $w = (h, ob, \mu)$ for some ob, μ
$(M, w, v) \models ob$ if $w = (h, ob, \mu)$ for some h, μ
$(M, w, v) \models \neg\varphi$ if $(M, w, v) \not\models \varphi$

[1] We allow sets of probability distributions for generality, despite the fact that our logic does not have the expressive power to reason about them. It is straightforward to extend the logic with a knowledge operator $K\varphi$ as in Section 2.2, true at a world if φ is true at all worlds with the same observation, that would capture formally the uncertainty about the prior probability.

$(M, w, v) \models \varphi \wedge \psi$ if $(M, w, v) \models \varphi$ and $(M, w, v) \models \psi$
$(M, w, v) \models p \geq c$ if $[p]^{M,w,v} \geq c$
$(M, w, v) \models \forall x \varphi$ if $(M, w, v') \models \varphi$ for all v' that agree with v on all variables but x.

If $(M, w, v) \models \varphi$ is true for all v, we simply write $(M, w) \models \varphi$. It is easy to check that if φ is a closed formula (that is, one with no free variables), then $(M, w, v) \models \varphi$ if and only if $(M, w, v') \models \varphi$, for all v, v'. Therefore, given a closed formula φ, if $(M, w, v) \models \varphi$, then in fact $(M, w) \models \varphi$. We will typically be concerned only with closed formulas. If $(M, w) \models \varphi$ for all worlds w, then we write $M \models \varphi$ and say that φ is valid in M. Finally, if $M \models \varphi$ for all evidential structures M, we write $\models \varphi$ and say that φ is valid. In the next section, we characterize axiomatically all the valid formulas of the logic.

5.3 Axiomatizing Evidence

In this section we present a sound and complete axiomatization $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ for our logic.

The axioms can be divided into four parts. The first set of axioms accounts for first-order reasoning:

Taut. All instances of valid formulas of first-order logic with equality.
MP. From φ and $\varphi \Rightarrow \psi$ infer ψ.

Instances of Taut include, for example, all formulas of the form $\varphi \vee \neg\varphi$, where φ is an arbitrary formula of the logic. It also includes formulas such as $(\forall x \varphi) \Leftrightarrow \varphi$ if x is not free in φ. In particular, $(\forall x(h)) \Leftrightarrow h$ for hypotheses in Φ_{h}, and similarly for observations in Φ_{o}. Axiom Taut can be replaced by a sound and complete axiomatization for first-order logic with equality.

The second set of axioms accounts for reasoning about polynomial inequalities, by relying on the theory of real closed fields:

RCF. All instances of formulas valid in real closed fields (and, thus, true about the reals), with nonlogical symbols $+$, $\cdot$, $<$, 0, 1, -1, 2, -2, 3, -3, ...

Formulas that are valid in real closed fields include, for example, the fact that addition on the reals is associative, $\forall x \forall y \forall z((x + y) + z = x + (y + z))$, that 1 is the identity for multiplication, $\forall x(x \cdot 1 = x)$, and formulas relating the constant symbols, such as $k = 1 + \cdots + 1$ (k times) and $-1 + 1 = 0$. As for Taut, we could replace RCF by a sound and complete axiomatization for real closed fields.

The third set of axioms essentially captures the fact that there is a single hypothesis and a single observation that holds per state.

H1. $h_1 \vee \cdots \vee h_{n_{\mathsf{h}}}$.
H2. $h_i \Rightarrow \neg h_j$ if $i \neq j$.
O1. $ob_1 \vee \cdots \vee ob_{n_{\mathsf{o}}}$.
O2. $ob_i \Rightarrow \neg ob_j$ if $i \neq j$.

These axioms illustrate a subtlety of our logic. Like most propositional logics, ours is parametrized by primitive propositions, in this case, Φ_{h} and Φ_{o}. However, while axiomatizations for propositional logics typically do not depend on the exact set of primitive propositions, ours does. Clearly, axiom H1 is sound only if the hypothesis primitives are exactly $h_1, \ldots, h_{n_{\mathsf{h}}}$. Similarly, axiom O1 is sound only if the observation primitives are exactly $ob_1, \ldots, ob_{n_{\mathsf{o}}}$. It is therefore important to identify the primitive propositions when talking about the axiomatization $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$.

The last set of axioms concerns reasoning about probabilities and evidence proper.

Pr1. $\Pr^0(true) = 1$.
Pr2. $\Pr^0(\rho) \geq 0$.
Pr3. $\Pr^0(\rho_1 \wedge \rho_2) + \Pr^0(\rho_1 \wedge \neg\rho_2) = \Pr^0(\rho_1)$.
Pr4. $\Pr^0(\rho_1) = \Pr^0(\rho_2)$ if $\rho_1 \Leftrightarrow \rho_2$ is a propositional tautology.

Axiom Pr1 simply say that the event $true$ has probability 1. Axiom Pr2 says that probability is nonnegative. Axiom Pr3 captures finite additivity. It is not possible to express countable additivity in our logic. On the other hand, we do not need an axiom for countable additivity. Roughly speaking, as we establish in the next section, if a formula is satisfiable at all, it is satisfiable in a finite structure. Similar axioms capture posterior probability formulas:

Po1. $\Pr(true) = 1$.
Po2. $\Pr(\rho) \geq 0$.
Po3. $\Pr(\rho_1 \wedge \rho_2) + \Pr(\rho_1 \wedge \neg\rho_2) = \Pr(\rho_1)$.
Po4. $\Pr(\rho_1) = \Pr(\rho_2)$ if $\rho_1 \Leftrightarrow \rho_2$ is a propositional tautology.

Finally, we need axioms to account for the behaviour of the evidence operator w. What are these properties? For one thing, the weight function acts like a probability on hypotheses, for each fixed observation. This gives the following four axioms, which are the obvious analogues of Pr1–4:

E1. $\mathrm{w}(ob, true) = 1$.
E2. $\mathrm{w}(ob, \rho) \geq 0$.
E3. $\mathrm{w}(ob, \rho_1 \wedge \rho_2) + \mathrm{w}(ob, \rho_1 \wedge \neg\rho_2) = \mathrm{w}(ob, \rho_1)$.
E4. $\mathrm{w}(ob, \rho_1) = \mathrm{w}(ob, \rho_2)$ if $\rho_1 \Leftrightarrow \rho_2$ is a propositional tautology.

Second, evidence connects the prior and posterior beliefs via Dempster's Rule of Combination, as in (5.1). This is captured by the following axiom. (Note that,

since there is no division in the language, crossmultiplication is used to clear the denominator.)

E5. $ob \Rightarrow (\Pr^0(h)\mathrm{w}(ob,h) = \Pr(h)\Pr^0(h_1)\mathrm{w}(ob,h_1) + \cdots + \Pr(h)\Pr^0(h_{n_\mathsf{h}})\mathrm{w}(ob,h_{n_\mathsf{h}}))$.

This is not quite enough. As we saw in Section 5.1, property WF2 in Theorem 5.3 is required for a function to be an evidence function. The following axiom captures WF2 in our logic:

E6. $\exists x_1 \ldots \exists x_{n_\mathsf{o}}(x_1 > 0 \wedge \cdots \wedge x_{n_\mathsf{o}} > 0 \wedge \mathrm{w}(ob_1,h_1)x_1 + \cdots + \mathrm{w}(ob_{n_\mathsf{o}},h_1)x_{n_\mathsf{o}} = 1 \wedge \cdots \wedge \mathrm{w}(ob_1,h_{n_\mathsf{h}})x_1 + \cdots + \mathrm{w}(ob_{n_\mathsf{o}},h_{n_\mathsf{h}})x_{n_\mathsf{o}} = 1)$.

Note that axiom E6 is the only axiom that requires quantification. Moreover, axioms E5 and E6 both depend on Φ_h and Φ_o.

Theorem 5.5. $\mathrm{AX}^{fo\text{-}ev}(\Phi_\mathsf{h},\Phi_\mathsf{o})$ *is a sound and complete axiomatization for the logic* $\mathcal{L}^{fo\text{-}ev}(\Phi_\mathsf{h},\Phi_\mathsf{o})$ *with respect to evidential structures.*

As usual, soundness is straightforward, and to prove completeness, it suffices to show that if a formula φ is consistent with $\mathrm{AX}^{fo\text{-}ev}(\Phi_\mathsf{h},\Phi_\mathsf{o})$, it is satisfiable in an evidential structure. However, the usual approach for proving completeness in modal logic, which involves considering maximal consistent sets and canonical structures does not work. The problem is that there are maximal consistent sets of formulas that are not satisfiable. For example, there is a maximal consistent set of formulas that includes $\Pr(\rho) > 0$ and $\Pr(\rho) \leq 1/n$ for $n = 1, 2, \ldots$. This is clearly unsatisfiable.

To express axiom E6, we needed to have quantification in the logic. An interesting question is whether it is possible to give a sound and complete axiomatization to the propositional fragment of our logic (without quantification or variables). To do this, we need to give quantifier-free axioms to replace axiom E6. This amounts to asking whether there is a simpler property than WF2 in Theorem 5.3 that characterizes weight of evidence functions. This remains an open question.

5.4 Decision Procedures

In this section, we consider the decision problem for $\mathcal{L}^{fo\text{-}ev}(\Phi_\mathsf{h},\Phi_\mathsf{o})$, that is, the problem of deciding whether a given formula φ is satisfiable. In order to state the problem precisely, however, we need to deal carefully with the fact that the logic is parameterized by the sets Φ_h and Φ_o of primitive propositions representing hypotheses and observations. In most logics, the choice of underlying primitive

propositions is essentially irrelevant. For example, if a propositional formula φ that contains only primitive propositions in some set Φ is true with respect to all truth assignments to Φ, then it remains true with respect to all truth assignments to any set $\Phi' \supseteq \Phi$. This monotonicity property does not hold here. For example, as we have already observed, axiom H1 clearly depends on the set of hypotheses and observations; it is no longer valid if the set is changed. The same is true for O1, E5, and E6.

This means that we have to be careful, when stating decision problems, about the role of Φ_{h} and Φ_{o} in the algorithm. A straightforward way to deal with this is to assume that the satisfiability algorithm gets as input Φ_{h}, Φ_{o}, and a formula $\varphi \in \mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. Because $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ contains the full theory of real closed field, it is unsurprisingly difficult to decide. For our decision procedure, we can use an existing exponential space algorithm to decide the satisfiability of real closed field formulas. Define the length $|\varphi|$ of φ to be the number of symbols required to write φ, where we count the length of each coefficient as 1. Similarly, define $\|\varphi\|$ to be the length of the longest coefficient appearing in f, when written in binary.

Theorem 5.6. *There is a procedure that runs in space exponential in $|\varphi| \cdot \|\varphi\|$ for deciding, given Φ_{h} and Φ_{o}, whether a formula φ of $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is satisfiable in an evidential structure.*

This is essentially the best we can do, as the decision problem for the real closed fields is complete for exponential space, and $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ contains the full language of real closed fields.

While Theorem 5.6 assumed that the algorithm takes as input the set of primitive propositions Φ_{h} and Φ_{o}, this does not really affect the complexity of the algorithm. More precisely, if we are given a formula φ in $\mathcal{L}^{fo\text{-}ev}$ over some set of hypotheses and observations, we can still decide whether φ is satisfiable, that is, whether there are sets Φ_{h} and Φ_{o} of primitive propositions containing all the primitive propositions in φ and an evidential structure M that satisfies φ.

Theorem 5.7. *There is a procedure that runs in space exponential in $|\varphi| \cdot \|\varphi\|$ for deciding whether there exists sets of primitive propositions Φ_{h} and Φ_{o} such that $\varphi \in \mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ and φ is satisfiable in an evidential structure.*

The main culprit for the exponential-space complexity is the theory of real closed fields, which we had to add to the logic to be able to even write down axiom E6 of the axiomatization $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$.[2] However, if we are not interested in

[2] Recall that axiom E6 requires quantification. Observe however that it requires a single quantifier alternation; thus, we can restrict to the sublanguage consisting of formulas with at most one quantifier alternation. The satisfiability problem may be easier for this fragment.

axiomatizations, but simply in verifying properties of probabilities and weights of evidence, we can consider the following propositional (quantifier-free) fragment of our logic. As before, start with sets Φ_{h} and Φ_{o} of hypothesis and observation primitives, and form the sublanguage $\mathcal{L}_{\mathsf{h}}$ of hypothesis formulas. Basic terms have the form $\Pr^0(\rho)$, $\Pr(\rho)$, and $\mathrm{w}(ob, \rho)$, where ρ is an hypothesis formula and ob is an observation. A quantifier-free linear term has the form $a_1 t_1 + \cdots + a_n t_n$, where each a_i is an integer, and each t_i is a basic term. A quantifier-free linear inequality formula has the form $p \geq c$, where p is a quantifier-free linear term, and c is an integer. For instance, a quantifier-free linear inequality formula takes the form $\Pr^0(\rho) + 3\mathrm{w}(ob, \rho) + 5\Pr(\rho') \geq 7$.

Let $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ be the language obtained by starting out with the primitive propositions in Φ_{h} and Φ_{o} and quantifier-free linear inequality formulas, and closing off under conjunction and negation. Since quantifier-free linear inequality formulas are polynomial inequality formulas, $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is a sublanguage of $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. The logic $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is sufficiently expressive to express many properties of interest; for instance, it can certainly express the relationship between prior probability and posterior probability through the weight of evidence of a particular observation, as shown in Example 5.4. Reasoning about the propositional fragment of our logic $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is easier than the full language:

Theorem 5.8. *The problem of deciding, given Φ_{h} and Φ_{o}, whether a formula φ of $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is satisfiable in an evidential structure is NP-complete.*

As in the general case, the complexity is unaffected by whether or not the decision problem takes as input the sets Φ_{h} and Φ_{o} of primitive propositions.

Theorem 5.9. *The problem of deciding, for a formula φ, whether there exists sets of primitive propositions Φ_{h} and Φ_{o} such that $\varphi \in \mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ and φ is satisfiable in an evidential structure is NP-complete.*

Since $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ allows only quantifier-free linear inequalities, it cannot express the general connection between priors, posteriors, and evidence captured by axiom E5. It is possible to extend $\mathcal{L}^{ev}$ to allow multiplication of probability terms. Let $\mathcal{L}^{ev,\times}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ be defined as $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$, except that instead of using quantifier-free linear terms, we allow quantifier-free polynomial terms, of the form $a_1 t_1 + \cdots + a_n t_n$, where each a_i is an integer, and each t_i is a *product* of basic terms. Clearly, E5 can be expressed in $\mathcal{L}^{ev,\times}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. Furthermore, this sublanguage of $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ can be decided in polynomial space, using an existing procedure for deciding the validity of quantifier-free formulas in the theory of real closed fields.

5.5 Evidence in Dynamic Systems

The evidential structures we have considered until now are essentially static, in that they model only the situation where a single observation is made. Even in Example 5.2, where we consider sequences of coin tosses, these are viewed as a single observations. Doing this lets us focus on the relationship between the prior and posterior probabilities on hypotheses and the weight of evidence of a single observation. In the last chapter, we studied evidence in the context of randomized algorithms; evidence was used to characterize the information provided by, for example, a randomized algorithm for primality when it says that a number is prime. The framework in that chapter is dynamic; sequences of observations are made over time. In this section, we discuss combining evidence from sequences of observation, and extend our logic to reason about such combination of evidence.

5.5.1 Combining Evidence

In Section 4.2, we considered the weight of evidence of a single observation. This generalizes in a straightforward way to the case where there are sequences of observations. Let $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \{\mu_h \mid h \in \mathcal{H}\})$ be an evidence space. Let $\mathcal{O}^*$ be the set of sequences $\langle ob_1, \ldots, ob_n\rangle$ over $\mathcal{O}$. Assume that the observations are independent, that is, for each basic hypothesis h, assume that $\mu_h^*(\langle ob_1, \ldots, ob_n\rangle)$, the probability of observing a particular sequence of observations given hypothesis h, is $\mu_h(ob_1) \cdots \mu_h(ob_n)$, the product of the probability of making each observation in the sequence. Let $\mathcal{E}^* = (\mathcal{H}, \mathcal{O}^*, \{\mu_h^* \mid \mathcal{H}\})$. With this assumption, it is well known that Dempster's Rule of Combination can be used to combine evidence; that is,

$$w_{\mathcal{E}^*}(\langle ob_1, \ldots, ob_k\rangle, \cdot) = w_{\mathcal{E}}(ob_1, \cdot) \oplus \cdots \oplus w_{\mathcal{E}}(ob_k, \cdot).$$

It is an easy exercise to check that the weight provided by the sequence of observations $\langle ob_1, \ldots, ob_n\rangle$ can be expressed in terms of the weight of the individual observations:

$$w_{\mathcal{E}^*}(\langle ob_1, \ldots, ob_n\rangle, h) = \frac{w_{\mathcal{E}^*}(ob_1, h) \cdots w_{\mathcal{E}^*}(ob_n, h)}{\sum_{h' \in \mathcal{H}} w_{\mathcal{E}^*}(ob_1, h') \cdots w_{\mathcal{E}^*}(ob_n, h')}. \tag{5.2}$$

If we let μ_0 be a prior probability on the hypotheses, and $\mu_{\langle ob_1, \ldots, ob_k\rangle}$ be the probability on the hypotheses after observing $ob_1, \ldots, ob_k$, we can verify that

$$\mu_{\langle ob_1, \ldots, ob_k\rangle} = \mu_0 \oplus w_{\mathcal{E}^*}(\langle ob_1, \ldots, ob_k\rangle, \cdot).$$

Example 5.10. Consider a variant of Example 5.2, where we take the coin tosses as individual observations, rather than the number of heads that turn up in one

hundred coin tosses. As before, assume that the coin chosen by Alice is either double-headed or fair. Let $\mathcal{O} = \{H, T\}$, the result of an individual coin toss, where H is "the coin landed heads" and T is "the coin landed tails". Let $\mathcal{H} = \{F, D\}$, where F is "the coin is fair", and D is "the coin is double-headed". Let $\mathcal{E}^* = (\mathcal{H}, \mathcal{O}^*, \{\mu_h \mid h \in \mathcal{H}\})$. The probability spaces associated with the hypotheses are generated by the following probabilities for simple observations:

$$\mu_F(H) = \frac{1}{2} \qquad \mu_D(H) = 1.$$

For example, we have $\mu_F(\langle H, H, T, H\rangle) = 1/16$, $\mu_D(\langle H, H, H\rangle) = 1$, and $\mu_H(\langle H, H, T, H\rangle) = 0$.

We can now easily verify results similar to those that were obtained in Example 5.2. For instance, the weight of observing T in favor of F is given by

$$w_{\mathcal{E}^*}(T, F) = \frac{\frac{1}{2}}{0 + \frac{1}{2}} = 1,$$

which again indicates that observing T provides unconditional support to F; a double-headed coin cannot land tails.

How about sequences of observations? The weight provided by the sequence $\langle ob_1, \ldots, ob_n\rangle$ for hypothesis h is given by Equation (5.2). If $\underline{H} = \langle H, \ldots, H\rangle$, a sequence of a hundred coin tosses, we can check that

$$w_{\mathcal{E}^*}(\underline{H}, F) = \frac{\frac{1}{2^{100}}}{1 + \frac{1}{2^{100}}} = \frac{1}{1 + 2^{100}} \qquad w_{\mathcal{E}^*}(\underline{H}, D) = \frac{1}{1 + \frac{1}{2^{100}}} = \frac{2^{100}}{1 + 2^{100}}.$$

Unsurprisingly, this is the same result as in Example 5.2. □

There are subtleties involved in trying to find an appropriate logic for reasoning about situations like that in Example 5.10. The most important one is the relationship between observations and time. By way of illustration, consider the following example. Bob is expecting an email from Alice stating where a rendezvous is to take place. Calm under pressure, Bob is reading while he waits. Assume that Bob is not concerned with the time. For the purposes of this example, one of three things can occur at any given point in time:

(1) Bob does not check if he has received email;
(2) Bob checks if he has received email, and notices he has not received an email from Alice;
(3) Bob checks if he has received email, and notices he has received an email from Alice.

How is Bob's view of the world affected by these events? In (1), it should be clear that, all things being equal, Bob's view of the world does not change: no observation is made. Contrast this with (2) and (3). In (2), Bob does make an observation, namely that he has not yet received Alice's email. The fact that he checks indicates that he wants to observe a result. In (3), he also makes an observation, namely that he received an email from Alice. In both of these cases, the check yields an observation, that he can use to update his view of the world. In case (2), he essentially observed that "nothing" happened, but we emphasize again that this is an observation, to be distinguished from the case where Bob does not even check whether email has arrived. This "nothing" observation should be taken into account in the evidence space under consideration.

This discussion motivates the models that we introduce in the next section. We characterize an agent's state by the observations that she has made, including possibly the "nothing" observation. Although we do not explicitly model time, it is easy to incorporate time in the framework, since the agent can observe times or clock ticks.

5.5.2 Reasoning about the Evidence of Sequences of Observations

We extend the framework of Section 5.2 to the dynamic setting. Rather than just considering worlds, we now consider sequences of worlds (which we call *runs*), representing the evolution of the system over time. The models are now sets of runs, with a set of prior probabilities on the hypotheses that hold in the runs. It is straightforward to modify our logic to express properties of evidence in this more dynamic setting.

In some ways, considering a dynamic setting simplifies things. Rather than talking about the prior and posterior probability using different operators, we need only a single probability operator that represents the probability of an hypothesis at the current time. To express the analogue of axiom E5 in this logic, we need to be able to talk about the probability at the next time step. This can be done by adding the "next-time" operator $\bigcirc$ to the logic, where $\bigcirc\varphi$ holds at the current time if φ holds at the next time step.[3] The logic is further extended to talk about the weight of evidence of a sequence of observations.

Define the logic $\mathcal{L}_{dyn}^{fo\text{-}ev}$ as follows. As in Section 5.2, start with a set of primitive propositions Φ_{h} and Φ_{o}, respectively representing the hypotheses and the observations. Again, let $\mathcal{L}_{\mathsf{h}}(\Phi_{\mathsf{h}})$ be the propositional sublanguage of hypotheses formulas

[3] Following the discussion at the end of Section 5.5.1, time steps are associated with new observations. Thus, $\bigcirc\varphi$ means that φ is true at the next time step, that is, after the next observation. This simplifies the presentation of the logic.

obtained by taking primitive propositions in Φ_{h} and closing off under negation and conjunction; we use ρ to range over formulas of that sublanguage.

A basic term now has the form $\Pr(\rho)$, or $\mathrm{w}(\underline{ob}, \rho)$, where ρ is an hypothesis formula and $\underline{ob} = \langle ob_1, \ldots, ob_n \rangle$ is a nonempty sequence of observations. (If $\underline{ob} = \langle ob_1 \rangle$, we write $\mathrm{w}(ob_1, \rho)$ rather than $\mathrm{w}(\langle ob_1 \rangle, \rho)$.) As before, a polynomial term has the form $t_1 + \cdots + t_n$, where each term t_i is a product of integers, basic terms, and variables (which intuitively range over the reals). A polynomial inequality formula has the form $p \geq c$, where p is a polynomial term and c is an integer. Let $\mathcal{L}_{dyn}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ be the language obtained by starting out with the primitive propositions in Φ_{h} and Φ_{o} and polynomial inequality formulas, and closing off under conjunction, negation, first-order quantification, and application of the $\bigcirc$ operator. The same abbreviations as in Section 5.2 are used.

The semantics of this logic now involves models that have dynamic behaviour. More precisely, a model is a set of infinite runs, where each run describes a possible dynamic evolution of the system. As before, each run records the observations being made, as well as the hypothesis that is true and a probability distribution describing the prior probability of the hypothesis at the initial state of the system. An evidence space over Φ_{h} and Φ_{o} is necessary to interpret w, as in Section 5.2. Define an *evidential system* I to be a tuple $(\mathcal{R} \times \mathcal{P}, \mathcal{E}^*)$ where $\mathcal{R}$ is a set of runs, $\mathcal{P}$ is a set of probability measures on Φ_{h}, and $\mathcal{E}^*$ is an evidence space over Φ_{h} and Φ_{o}^*. A run r is a map from the natural numbers (representing time) to histories of the system up to that time. The history records, at time 0, the hypothesis that is true in that run, and at subsequent times, the observation made at each time step. Hence, a history has the form $\langle h, ob_1, \ldots, ob_n \rangle$. Assume that $r(0) = \langle h \rangle$ for some h, while $r(m) = \langle h, ob_1, \ldots, ob_m \rangle$ for $m > 0$. Define a point of the system to be a triple (r, m, μ) consisting of a run r, time m, and probability distribution μ.

Associate with each propositional formula ρ in $\mathcal{L}_{\mathsf{h}}(\Phi_{\mathsf{h}})$ a set $[\![\rho]\!]$ of hypotheses, just as was done in Section 5.2.

In order to ascribe a semantics to first-order formulas that may contain variables, we need a valuation v that assigns a real number to every variable. Given a valuation v, an evidential system $I = (\mathcal{R} \times \mathcal{P}, \mathcal{E}^*)$, and a point (r, m, μ), where $r(m) = \langle h, ob_1, \ldots, ob_m \rangle$, we can assign to a polynomial term p a real number $[p]^{I,r,m,\mu,v}$ using essentially the same approach as in Section 5.2:

$$
\begin{aligned}
[x]^{I,r,m,\mu,v} &= v(x) \\
[a]^{I,r,m,\mu,v} &= a \\
[\Pr(\rho)]^{I,r,m,\mu,v} &= (\mu \oplus w_{\mathcal{E}^*}(\langle ob_1, \ldots, ob_m \rangle, \cdot)))([\![\rho]\!]) \\
&\qquad \text{where } r(m) = \langle h, ob_1, \ldots, ob_m \rangle \\
[\mathrm{w}(\underline{ob}, \rho)]^{I,r,m,\mu,v} &= w_{\mathcal{E}^*}(\underline{ob}, [\![\rho]\!])
\end{aligned}
$$

$$\begin{aligned}[t_1 t_2]^{I,r,m,\mu,v} &= [t_1]^{I,r,m,\mu,v} \times [t_2]^{I,r,m,\mu,v} \\ [p_1 + p_2]^{I,r,m,\mu,v} &= [p_1]^{I,r,m,\mu,v} + [p_2]^{I,r,m,\mu,v}.\end{aligned}$$

Define what it means for a formula φ to be true (or satisfied) at a point (r, m, μ) of an evidential system $I = (\mathcal{R} \times \mathcal{P}, \mathcal{E}^*)$ under valuation v, written $(I, r, m, \mu, v) \models \varphi$, again using essentially the same approach as in Section 5.2:

$(I, r, m, \mu, v) \models h$ if $r(m) = \langle h, \ldots \rangle$
$(I, r, m, \mu, v) \models ob$ if $r(m) = \langle h, \ldots, ob \rangle$
$(I, r, m, \mu, v) \models \neg\varphi$ if $(I, r, m, \mu, v) \not\models \varphi$
$(I, r, m, \mu, v) \models \varphi \wedge \psi$ if $(I, r, m, \mu, v) \models \varphi$ and $(I, r, m, \mu, v) \models \psi$
$(I, r, m, \mu, v) \models p \geq c$ if $[p]^{I,r,m,\mu,v} \geq c$
$(I, r, m, \mu, v) \models \bigcirc\varphi$ if $(I, r, m + 1, \mu, v) \models \varphi$
$(I, r, m, \mu, v) \models \forall x\varphi$ if $(I, r, m, \mu, v') \models \varphi$ for all valuations v' that agree with v on all variables but x.

If $(I, r, m, \mu, v) \models \varphi$ is true for all v, we simply write $(I, r, m, \mu) \models \varphi$. If $(I, r, m, \mu) \models \varphi$ for all points (r, m, μ), then we write $I \models \varphi$ and say that φ is valid in I. Finally, if $I \models \varphi$ for all evidential systems I, we write $\models \varphi$ and say that φ is valid.

5.5.3 Axiomatization

It is rather straightforward to axiomatize this new logic. This axiomatization shows that we can capture the combination of evidence directly in the logic, a pleasant property. Most of the axioms from Section 5.2 carry over immediately. Let the axiomatization $\mathrm{AX}_{dyn}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ consists of the following axioms: first-order reasoning (Taut, MP), reasoning about polynomial inequalities (RCF), reasoning about hypotheses and observations (H1,H2,O1,O2), reasoning about probabilities (Po1–4 only, since we do not have Pr^0 in the language), and reasoning about weights of evidence (E1–4, E6), as well as new axioms we now present.

Basically, the only axiom that needs replacing is E5, which links prior and posterior probabilities, since this now needs to be expressed using the $\bigcirc$ operator. Moreover, we need an axiom to relate the weight of evidence of a sequence of observation to the weight of evidence of the individual observations, as given by Equation (5.2).

E7. $$\begin{aligned} ob \Rightarrow \forall x(\bigcirc(\mathrm{Pr}(h) = x) \Rightarrow & \\ \mathrm{Pr}(h)\mathrm{w}(ob, h) = & \; xPr(h_1)\mathrm{w}(ob, h_1) + \\ & \cdots + \\ & \; x\mathrm{Pr}(h_{n_{\mathsf{h}}})\mathrm{w}(ob, h_{n_{\mathsf{h}}})). \end{aligned}$$

E8. $\mathrm{w}(ob_1, h) \cdots \mathrm{w}(ob_n, h) =$
$\mathrm{w}(\langle ob_1, \ldots, ob_n \rangle, h)\mathrm{w}(ob_1, h_1) \cdots \mathrm{w}(ob_n, h_1) +$
$\cdots +$
$\mathrm{w}(\langle ob_1, \ldots, ob_n \rangle, h)\mathrm{w}(ob_1, h_{n_\mathsf{h}}) \cdots \mathrm{w}(ob_n, h_{n_\mathsf{h}})$.

To get a complete axiomatization, we also need axioms that capture the properties of the temporal operator $\bigcirc$. These axioms also capture the fact that the truth of hypotheses as well as the value of polynomial terms not containing occurrences of Pr is time-independent.

T1. $\bigcirc\varphi \wedge \bigcirc(\varphi \Rightarrow \psi) \Rightarrow \bigcirc\psi$.
T2. $\bigcirc\neg\varphi \Leftrightarrow \neg\bigcirc\varphi$.
T3. From φ infer $\bigcirc\varphi$.
T4. $\bigcirc\rho \Leftrightarrow \rho$.
T5. $\bigcirc(p \geq c) \Leftrightarrow p \geq c$ if p does not contain an occurrence of Pr.
T6. $\bigcirc(\forall x\varphi) \Leftrightarrow \forall x(\bigcirc\varphi)$.

Theorem 5.11. $\mathrm{AX}^{fo\text{-}ev}_{dyn}(\Phi_\mathsf{h}, \Phi_\mathsf{o})$ *is a sound and complete axiomatization for* $\mathcal{L}^{fo\text{-}ev}_{dyn}(\Phi_\mathsf{h}, \Phi_\mathsf{o})$ *with respect to evidential systems.*

Notes

The work in this chapter first appeared in [Halpern and Pucella 2003a].

The situation at the beginning of the chapter is essentially taken from [Halpern and Tuttle 1993; Fagin and Halpern 1994]. As we mentioned, there has been a great deal of work on reasoning about systems that combine both probabilistic and nondeterministic choices [Vardi 1985; Fischer and Zuck 1988; Halpern, Moses, and Tuttle 1988; Halpern and Tuttle 1993; de Alfaro 1998; He, Seidel, and McIver 1997]. The framework described at the beginning of the chapter is due to Halpern and Tuttle [1993].

The view of evidence as a function mapping a prior probability on hypotheses to a posterior probability is developed by Halpern and Fagin [1992]. Dempster's Rule of Combination arises in Shafer's [1976] theory of belief functions. Theorem 5.1 is proved in [Halpern and Fagin 1992]

The logic $\mathcal{L}^{fo\text{-}ev}$ was inspired by a logic introduced by Fagin, Halpern, and Megiddo [1990] for reasoning about likelihood expressed as either probability of belief. Sound and complete axiomatizations for first-order logic with equality are given, for instance, in Shoenfield [1967] or Enderton [1972]. The sound and complete axiomatization for RCF is due to Tarski [1951] and can also be found in [Fagin, Halpern, and Megiddo 1990; Shoenfield 1967]. The axioms for probability (Pr1–4, Po1–4) are taken from [Fagin, Halpern, and Megiddo 1990]. The

techniques for proving Theorem 5.5 and 5.11 follow those developed by Fagin, Halpern, and Megiddo [1990].

The exponential space algorithm used in Section 5.4 to decide the satisfiability of real closed field formulas is due to Ben-Or, Kozen, and Reif [1986]. They also give corresponding lower bound for deciding the satisfiability of real closed field formulas. In the literature, a number of different lower bounds are given for problems that are related, which makes reading a bit confusing. For instance, a doubly-exponential algorithm (and corresponding lower bound) is known for the problem of quantifier elimination in real closed fields [Renegar 1992; Basu 1999; Weispfenning 1988], which implies a satisfaction algorithm that solves the satisifiability problem. However, there are satisfaction algorithms that are not based on quantifier elimination that do not fall within the restrictions of the bound. The polynomial space algorithm to decide the validity of quantifier-free formulas in the theory of real closed fields is due to Canny [1988].

The fact that Dempster's Rule of Combination can be used to combine evidence under the assumption that the observations are independent can be found, for example, in [Halpern and Fagin 1992, Theorem 4.3].

Part II
Application to Security Protocol Analysis

6
Security Protocols

In the second part of this dissertation, we consider the problem of analyzing security protocols, using the framework developed in the first part. Roughly speaking, a security protocol is simply a communication protocol between multiple agents that guarantees some security properties of the communication. These properties can include:

– message confidentiality: only the authorized recipient should be able to extract the contents of the message, possibly including statistical information;
– message integrity: the recipient should be able to determine if the message has been altered during transmission;
– sender authentication: the recipient should be able to identify the sender, and verify that the purported sender actually did send the message;
– sender non-repudiation: the sender should not be able to deny sending the message;
– sender anonymity: the recipient should not be able to identify the sender.

More properties are possible, and many are used in conjunction with others. A security protocol achieving authentication, for instance, is a protocol for two or more agents to communicate in such a way that they can be given guarantees as to whom they are communicating with, often by relying on the confidentiality of a piece of data, such as a shared cryptographic key. In this dissertation, we mostly concentrate on authentication protocols, and consider confidentiality and authentication properties.

This chapter is a review of the basic concepts of security protocol analysis, including symbolic cryptography, adversary models, and security properties. We also review existing approaches to analyzing security protocols, and point out their limitations.

6.1 Protocols and Cryptography

A protocol describes the behaviour of multiple agents that seek to achieve a common goal, for example, exchanging a value. Security protocols additionally impose some security guarantees, for instance, that the value exchanged by the agents cannot be "read" by any other agent in the system. Protocols can involve an arbitrary number of agents, and also a number of trusted servers. For simplicity, we focus on two-agent protocols, possibly with a single trusted server; everything generalizes to multiple parties in the obvious way.

To enforce the security guarantees, security protocols generally rely on some form of cryptography. Intuitively, cryptography permits the encoding of information in such a way that only a select and controllable few can decode it. Much of the current research in security concerns the development of new cryptographic techniques, and new encryption schemes. In this dissertation, we are interested in establishing properties of protocols *independently* of the details of the encryption scheme. The typical way to do this in the literature is to analyze protocols in the presence of perfect cryptography. This leads to a form of protocol analysis often called *symbolic protocol analysis*. (In constrast to *computational protocol analysis*, where the computational properties of the cryptography are taken into account; see Chapter 10 for more detail.)

To model perfect cryptography, define the *symbolic encryption scheme* generated by the set $\mathcal{P}$ of plaintexts (the original values to be encrypted, such as English strings) and the set $\mathcal{K}$ of keys to be the set $\mathcal{M}$ of messages given by the grammar

$$m ::= p \mid k \mid \{m\}_k \mid (m_1, m_2)$$

where m, m_1, m_2 are generic elements of $\mathcal{M}$, p is a generic element of $\mathcal{P}$, and k is a generic element of $\mathcal{K}$. The notation $\{m\}_k$ represents the encryption of message m with the key k, while (m_1, m_2) represents the pairing (or concatenation) of m_1 and m_2. We omit parentheses for pairing when there is no confusion, so that $\{(m_1, m_2)\}_k$ can be written as $\{m_1, m_2\}_k$. We also write $(m_1, \ldots, m_n)$ for $(m_1, (\ldots, (m_{n-1}, m_n)\ldots))$.

It is useful to assume that the set $\mathcal{P}$ of plaintexts contains a representation of the names of the agents in the system (so that names of agents can be sent encrypted as part of messages), as well as a distinguished set $\mathcal{N}$ of values to be used as nonces. Assume that the sets $\mathcal{N}$, $\mathcal{K}$, and $\mathcal{P} - \mathcal{N}$ are pairwise disjoint, and that values in each set can be distinguished. Assume further that encrypted messages can be distinguished from unencrypted ones.

Assume the set $\mathcal{K}$ of keys is closed under inverses, that is, if $k \in \mathcal{K}$, then $k^{-1} \in \mathcal{K}$. If $k^{-1} = k$, the key is a *symmetric key*, otherwise, it is an *asymmetric key*. We sometimes write $\{\!|m|\!\}_k$ when wanting to emphasize the fact that message

m is encrypted using an asymmetric key k. An asymmetric key is used to model public-key cryptography, while a symmetric key is used to model shared-key cryptography. The decryption of a message $\{m\}_k$ requires the key k^{-1}, the inverse of key k. We make the standard assumption that there is enough redundancy in the encrypted message that it is possible to recognize when decrypting whether the decryption was successful. If a decryption is unsuccessful, the special plaintext $null$ is returned. Projections π_1 and π_2 are used to decompose a pairing (m_1, m_2). Again, applying π_1 and π_2 to messages that are not pairs results in the value $null$.

Having cryptography, unfortunately, is often not sufficient for two or more agents to communicate securely. For example, to communicate using symmetric keys, these keys need to be distributed and agreed upon before an interaction. Moreover, it is not always clear how to exchange messages so that secrecy is preserved across interactions between the agents. To get this right, one needs to develop communication protocols.

Historically, and rather unfortunately, protocols are presented by giving a sequence of messages exchanged between the agents in a good execution of the protocol. As a simple example, consider the following two-agent protocol SEND-SHARED, which is the simplest possible protocol for sending a message m "securely" from one agent to another:

$$1. \qquad A \rightarrow B \quad : \quad A, \{m\}_{k_{AB}}.$$

This protocol represents the sending of message m from A to B, where the message is encrypted using a symmetric key k_{AB} shared between A and B. Agent A needs to send her name along with the message, unless the receiver knows who will be sending the message. (Whether or not the receiver knows *a priori* who will be sending the message is part of the assumptions made when designing or analyzing the protocol.) Intuitively, upon receiving the message and decrypting it, B knows that the message m came from A, since only A could have encrypted it, assuming only A and B share the key k_{AB}. Moreover, m is kept secret throughout the exchange. Intuitively, the fact that the message is encrypted using k_{AB} is evidence to B that the message is from A; the fact that m itself is encrypted is only useful if m is meant to be kept secret.

The notation above, which at first seems quite reasonable, leaves a number of very important issues completely unspecified. This is a consequence of the notation essentially representing a trace of the protocol, rather than the protocol itself. For instance, the notation does not specify what actions the agents perform on the values they received. To illustrate this, consider the following exchange between

Alice, Bob, and Charlie:

$$\begin{array}{lrcl} 1. & A \to B & : & \{m\}_{k_{AB}} \\ 2. & B \to C & : & \{\{m\}_{k_{AB}}\}_{k_{BC}} \end{array}$$

where k_{AB} is a key shared between Alice and Bob and k_{BC} is a key shared between Bob and Charlie. When receive message 1, does Bob attempt to decrypt $\{m\}_{k_{AB}}$ to make sure that the value is really a value that was sent by Alice? The notation simply does not carry this information. This could be important if the protocol is not meant to proceed unless Bob is convinced that message 1 was encrypted using a key shared with Alice. A more precise notation is to actually consider a program that each principal runs to execute the protocol. Two-agent protocols are typically made up of two programs, one describing the behaviour of the *initiator* of the interaction, and one describing the behaviour of the *responder* of the interaction. For instance, the program for the initiator part of the SENDSHARED protocol is straightforward:

$$\text{SENDSHARED}_I(i, j, m, k) = \mathbf{send}\ j\ (i, \{m\}_{k(j)}).$$

This program is parametrized by variables i (the name of the agent running this program), j (the name of the agent to whom to send a message), m (the message to send), and k (a map from agents to keys shared between i and those agents). The map k could be replaced by a list of keys. (We remain informal as far as interpreting programs in this chapter, relying on the reader's intuition. we give a formal syntax and semantics to this little programming language in Chapter 7.)

The program for the responder is equally simple:

$$\begin{aligned} \text{SENDSHARED}_R(i, k) = \mathbf{recv}\ m; \\ \mathit{message} \leftarrow \mathit{decrypt}(\pi_2(m), k(\pi_1(m))). \end{aligned}$$

This program is parametrized by k, a map from agents to keys shared between the responder and those agents. Roughly, it awaits to receive a message m, and attempts to decrypt it using the key corresponding to the agent named as the first component of the received message.

The protocol above is suitable if two agents share a key, but what if they do not? This becomes an issue when there are a lot of agents in the system: if there are n agents in the system, then we need n^2 shared keys to allow every agent to communicate with every other agent. A standard approach to lessen the key burden is to use a central trusted server. Roughly, every agent need only share a key with the server, which mediates the communication by decrypting data on demand. As an example of such a protocol, consider the following two-agent protocol SENDSERVER. Here, S is a trusted server, and agents A and B both share a key with S,

denoted k_{AS} and k_{BS}, respectively:

$$\begin{array}{lrcl} 1. & A \rightarrow B & : & A, \{m\}_{k_{AS}} \\ 2. & B \rightarrow S & : & B, \{A, \{m\}_{k_{AS}}\}_{k_{BS}} \\ 3. & S \rightarrow B & : & \{m\}_{k_{BS}}. \end{array}$$

Intuitively, this protocol is meant to send an authenticated message m from A to B. Agent A start the interaction with B by sending the message m encrypted with the key she shares with the trusted server. Upon receiving this message, B forwards it (encrypted with the key he shares with the trusted server) to S, stating that it comes from A. The server S receives this message, decrypts it twice (using k_{BS} and k_{AS}) to extract m, and encrypts this value again with k_{BS} before sending it back to B. The idea is that when B receives this last message, he knows that S was able to decrypt the message that was originally sent by the agent who claimed to be A, and thus value m was indeed from A.

Here are the programs corresponding the above protocol. Since this is a two-agent protocol with a trusted server, not only is there a program for the initiator and a program for the responder, but there is also a program for the trusted server. The initiator program is essentially the same as that for SENDSHARED:

$$\text{SENDSERVER}_I(i, j, m, k) = \mathbf{send}\ j\ (i, \{m\}_k).$$

SENDSERVER_I is parametrized like SENDSHARED_I, except that k here holds the key that the initiator shares with the trusted server. The responder program communicates twice with the server:

$$\begin{array}{rl} \text{SENDSERVER}_R(i, k) = & \mathbf{recv}\ m; \\ & \mathbf{send}\ s\ (i, \{m\}_k); \\ & \mathbf{recv}\ m; \\ & \mathit{message} \leftarrow \mathit{decrypt}(m, k). \end{array}$$

This program is parametrized by i, holding the name of the agent running this program, and k, holding the key that agent i shares with the trusted server. Finally, the server program is a loop that repeatedly decrypts data formatted in the appropriate way, and sends back the results:

$$\begin{array}{rl} \text{SENDSERVER}_S(k) = & \mathbf{while}\ \mathit{true}\ \mathbf{do} \\ & \quad \mathbf{recv}\ m; \\ & \quad \mathit{receiver} \leftarrow \pi_1(m); \\ & \quad \mathit{message1} \leftarrow \mathit{decrypt}(\pi_2(m), k(\mathit{receiver})); \\ & \quad \mathit{sender} \leftarrow \pi_1(\mathit{message1}); \\ & \quad \mathit{message2} \leftarrow \mathit{decrypt}(\pi_2(\mathit{message2}), k(\mathit{sender})); \\ & \quad \mathbf{send}\ \mathit{receiver}\ \{\mathit{message2}\}_{k(\mathit{receiver})}. \end{array}$$

The server program is parametrized by k, mapping agents to the key shared between the server and the agents.

We will exclusively consider protocols that aim at exchanging messages securely between two agents over an untrusted network. Most of the issues concerning the formalization of properties arise in that simple setting. Some protocols are designed to work in a more restricted environment, with more assumptions on the communication network. For instance, we may have a secure connection between machines, and the question is to ascertain the identity of the party at the other end of the connection. We return to this in Section 6.3, where we consider the problem of specifying security properties. Other assumptions on the network considered in the literature include whether the communication is broadcast, or channel-based, and whether the system is closed or open, with new agents joining in and out.

6.2 Adversaries

One of the most interesting aspects of security protocol analysis is that the analysis is performed in an adversarial context, that is, under the assumption that there is one or more adversaries that attempt to subvert the protocol. Deciding whether or not a protocol is secure depends partly on the kind of adversary we assume.

From the point of view of the analysis methods described later in the chapter, there are two important aspects about adversaries:

(1) the capabilities of the adversary, that is, the information an adversary can extract from the messages he intercepts, as well as the messages he can construct;
(2) the status of the adversary, that is, the kind of presence the adversary has in the system, if he is known to the other agents or not, if other agents can start protocol interactions with him.

For the capabilities, the standard assumptions made in the literature are due to Danny Dolev and Andrew Yao, and go hand in hand with the symbolic encryption scheme described in Section 6.1. Roughly speaking, a Dolev-Yao adversary can compose messages, replay them, or decipher them if he knows the right keys, but cannot otherwise "crack" encrypted messages. These capabilities can be formalized using the following inference rules. The idea is to define a relation $\vdash_{DY}$, where $H \vdash_{DY} m$ is interpreted as saying that the adversary can infer message m from a set H of message (intuitively, messages he has intercepted). This relation is defined using the inference rules

$$\frac{m \in H}{H \vdash_{DY} m} \qquad \frac{H \vdash_{DY} \{\!|m|\!\}_k \quad H \vdash_{DY} k^{-1}}{H \vdash_{DY} m}$$

$$\frac{H \vdash_{DY} (m_1, m_2)}{H \vdash_{DY} m_1} \quad \frac{H \vdash_{DY} (m_1, m_2)}{H \vdash_{DY} m_2}.$$

(This is essentially the system described in Example 3.1.) Thus, for instance, if an adversary intercepts the messages $\{m\}_{k_1}$, $\{k_1^{-1}\}_{k_2}$, and k_2^{-1}, he can derive m using these inference rules, since

$$\{\{m\}_{k_1}, \{k_1^{-1}\}_{k_2}, k_2^{-1}\} \vdash_{DY} m.$$

There are, of course, limitations to the Dolev-Yao adversary model. Among other things, the model does not consider the information the adversary may infer from properties of messages and knowledge about the protocol that is being used. To give an extreme example, due to John Mitchell, consider what we will refer to as the *Duck-Duck-Goose* protocol: an agent has an n-bit key and, according to her protocol, sends the bits that make up her key one by one. Of course, after intercepting these messages, an adversary will know the key. However, there is no way for a Dolev-Yao adversary to recognize that, at this point, he actually has the key. Much more can be said about adversary capabilities. We return to this topic in Chapter 8.

The second important aspect about adversaries is their role in the context of the protocol. At first blush, an important characteristic is whether the adversary is a *passive* adversary, also known as an *eavesdropping* adversary, that only gets to listen in on the messages exchanged between agents. Passive adversaries can breach confidentiality, but do not send messages (and so cannot pretend to be some other agent).

In contrast, an *active* adversary can attempt to reroute traffic, initiate interactions with other agents, and so on. We can distinguish two forms of active adversaries. An active adversary can be *insider*. Otherwise, it is an *outsider*. An insider is recognized as an agent by the other agents, who will accept communications from him, share keys with him, attempt to initiate sessions with him, and so on. Insider adversaries are sometimes called *corrupt*, *dishonest*, or *subverted* agents.

Establishing security in the presence of active adversaries is complicated by the fact that it is sometimes difficult to determine if a particular scenario is an attack or not. For the remainder of this section, we illustrate some of the issues that arise, and motivate the use of nonces in security protocols. To start off, consider the SENDSHARED protocol of the last section. Despite its simplicity, it is subject to the following *replay attack*, where an active adversary, having intercepted the first message sent from A to B, will simply replay this message at his leisure:

$$\begin{aligned} A \to B &: A, \{m\}_{k_{AB}} \\ X(A) \to B &: A, \{m\}_{m_{AB}}. \end{aligned}$$

The notation $X(A)$ indicates that the adversary X is masquerading as A. One

question that is worth asking is if this is really an attack. Clearly, this depends on the message, and on the context in which this protocol is used. For example, if the timeliness of the message m is important, or if the message should only be received once, or if the message is meant to indicate that agent A was in fact on the system at the time when m was received, then a replay attack is problematic. Note that the confidentiality of m is not compromised in these scenarios.

If replays are indeed a problem, we can amend the protocol to make them impossible. The idea is to use a challenge to allow B to recognize whether a message is a replay or not. Before sending her message to B, A first asks B for a challenge, a unique message n_B (typically called a *nonce*). She then encrypts not only her message m, but also the nonce n_B before sending it. Upon receiving the message, B can check that the nonce correspond to the one that he sent before, and that it has not been used before. If the nonce is recognized as having been used before, then this particular interaction may be a replay attack, and the message can simply be dropped. Here is the amended protocol SENDSHAREDNONCE:

$$\begin{array}{lrcl} 1. & A \rightarrow B & : & A \\ 2. & B \rightarrow A & : & n_B \\ 3. & A \rightarrow B & : & A, \{m, n_B\}_{k_{AB}}. \end{array}$$

The program for the initiator protocol is straightforward:

$$\begin{aligned} \text{SENDSHAREDNONCE}_I(i, j, m, k) = \ & \textbf{send}\ j\ i; \\ & \textbf{recv}\ n; \\ & \textbf{send}\ j\ (i, \{m, n\}_{k(j)}). \end{aligned}$$

For the responder protocol, we need a nonce to send to the agent. For simplicity, assume that the nonce is a parameter of the program:

$$\begin{aligned} \text{SENDSHAREDNONCE}_R(i, k, n) = \ & \textbf{recv}\ i; \\ & \textbf{send}\ j\ n; \\ & \textbf{recv}\ m; \\ & \mathit{content} \leftarrow \mathit{decrypt}(\pi_2(m), k(\pi_1(m))); \\ & \mathit{nb} \leftarrow \pi_2(\mathit{content}); \\ & \textbf{if}\ n = \mathit{nb}\ \textbf{then} \\ & \quad \mathit{message} \leftarrow \pi_1(\mathit{content}) \\ & \textbf{else} \\ & \quad \mathit{message} \leftarrow \mathit{null}. \end{aligned}$$

Note the explicit check to see if the nonce received with the message is the one that was sent out. Having the nonce as a parameter to the protocol is of course not ideal, although we can simply imagine the principal running a number of copies of this program, one per nonce. Another way of doing this would be to introduce a

new function $newnonce()$ that creates a new nonce. Dealing with nonces created in this way, although central to many of the symbolic approaches we discuss later in this section, is problematic. We return to this topic in Chapter 9.

One nice property of the SENDSHAREDNONCE protocol is that B, upon reception of the last message, knows that it is meant for him, since A used a key she shares with him to send the message. That property does not hold in general, even with symmetric cryptography. Consider the following "problems" with SENDSERVER. First off, SENDSERVER is vulnerable to the same replay attack as SENDSHARED, so we can amend the protocol as we did SENDSHARED, by adding an initial nonce handshake, and add the nonce to the message A sends to B. This yields the following protocol:

$$
\begin{array}{lrcl}
1. & A \rightarrow B & : & A \\
2. & B \rightarrow A & : & n_B \\
3. & A \rightarrow B & : & A, \{m, n_B\}_{k_{AS}} \\
4. & B \rightarrow S & : & B, \{A, \{m, n_B\}_{k_{AS}}\}_{k_{BS}} \\
5. & S \rightarrow B & : & \{m, n_B\}_{k_{BS}}.
\end{array}
$$

But even with this correction, there are two potential problems if the adversary is an insider; first, B cannot be certain that the message he receives actually came from A, and second, B cannot be sure that the message he receives was sent to him. As in the case of replay attacks, whether these are security problems or not depends on the context of use of the protocol. We now simply indicate what goes wrong, and how to correct it.

Assume that X, the active adversary, is an insider in the system, so that he shares a key k_{XS} with the trusted server. To illustrate the first problem, X can send a message m to B and make B believe that the message m was sent by A, another agent known to B, but possibly offline at the moment of the attack. This scenario involves X starting two interactions with B, one as himself, and one masquerading as A. A trace of the attack is presented in Figure 6.1, where the first column reports the messages exchanged between X and B, while the second column reports the messages exchanged between X masquerading as A, and B. In that interaction, m' is the result of decrypting $\{m, n'_B\}_{k_{XS}}$ with key k_{AS}. (Thus, this attack relies on the decryption operation to return a meaningful value, even when decrypting a message with the wrong key. This is a property common to many encryption schemes.) While two protocol interactions are initiated by X, only one of them succeeds, but B believes that he has been sent m by A, since the message bundles the nonce n_B, which was sent to A, as far as B is concerned.

Similarly, the second problem higlights that there is no assurance to B that the message he receives presumably from A was actually meant for him, even if it was actually sent by A. A trace of this attack is presented in Figure 6.2. Whether or not

$$
\begin{array}{rcll}
X \to B & : & X & \\
B \to X & : & n_B & \\
X(A) \to B & : & & A \\
B \to X(A) & : & & n'_B \\
X \to B & : & X, \{m, n_B\}_{k_{XS}} & \\
B \to S & : & B, \{X, \{m, n_B\}_{k_{XS}}\}_{k_{BS}} & \\
S \to X(B) & : & \{m, n_B\}_{k_{BS}} & \\
X(A) \to B & : & & A, \{m, n'_B\}_{k_{XS}} \\
B \to S & : & & B, \{A, \{m, n'_B\}_{k_{XS}}\}_{k_{BS}} \\
S \to X(B) & : & & \{m'\}_{k_{BS}} \\
X(S) \to B & : & \{m, n_B\}_{k_{BS}}. &
\end{array}
$$

Figure 6.1. First attack on amended SENDSERVER protocol

$$
\begin{array}{rcll}
A \to X & : & A & \\
X(A) \to B & : & & A \\
B \to X(A) & : & & n_B \\
X \to A & : & n_B & \\
A \to X & : & A, \{m, n_B\}_{k_{AS}} & \\
X(A) \to B & : & & A, \{m, n_B\}_{k_{AS}} \\
B \to S & : & & B, \{A, \{m, n_B\}_{k_{AS}}\}_{k_{BS}} \\
S \to B & : & & \{m, n_B\}_{k_{BS}}.
\end{array}
$$

Figure 6.2. Second attack on amended SENDSERVER protocol

this is actually an attack is debatable. The key point appears in the first message sent, where A initiate an interaction with X, not X masquerading as B. If the fact that the message is meant for X (and not B) is important to the context in which this protocol is used, then the above can meaningfully be called an attack. It is not a question of the origin of the message, since the message originally came from A in this trace.

One way to handle these (potential) problems is to add to every message the identity of the sender and that of the intended receiver; this information should be checked by the appropriate agents. Here is the amended protocol SENDSERVER-NONCE:

$$
\begin{array}{rrcl}
1. & A \to B & : & A \\
2. & B \to A & : & n_B \\
3. & A \to B & : & A, \{A, B, m, n_B\}_{k_{AS}} \\
4. & B \to S & : & B, \{A, \{A, B, m, n_B\}_{k_{AS}}\}_{k_{BS}} \\
5. & S \to B & : & \{A, B, m, n_B\}_{k_{BS}}.
\end{array}
$$

Note that the identity of the sender (A) as well as the intended receiver (B) are bundled with m and n_B in message (3). The programs for the protocol correspond to those for SENDSERVER, updated in the obvious way:

$$\begin{aligned}\text{SENDSERVERNONCE}_I(i,j,m,k) = \ & \mathbf{send}\ j\ i;\\ & \mathbf{recv}\ n;\\ & \mathbf{send}\ j\ (i,\{i,j,m,n\}_k).\end{aligned}$$

The program for the responder now needs to check that he is indeed the intended target of the final message:

$$\begin{aligned}\text{SENDSERVERNONCE}_R(i,k) = \ & \mathbf{recv}\ j;\\ & \mathbf{send}\ j\ n;\\ & \mathbf{recv}\ m;\\ & \mathbf{send}\ s\ (i,\{j,m\}_k);\\ & \mathbf{recv}\ m;\\ & mi \leftarrow \pi_1(\pi_2(\mathit{decrypt}(m,k)));\\ & mm \leftarrow \pi_1(\pi_2(\pi_2(\mathit{decrypt}(m,k))));\\ & mn \leftarrow \pi_2(\pi_2(\pi_2(\mathit{decrypt}(m,k))));\\ & \mathbf{if}\ mn = n \wedge mi = i\ \wedge mj = j\ \mathbf{then}\\ & \quad \mathit{message} \leftarrow mm\\ & \mathbf{else}\\ & \quad \mathit{message} \leftarrow \mathit{null}.\end{aligned}$$

Upon receiving the last message, the responder checks that the nonces agree, that he is the intended receiver ($mi = i$) and that the sender is the claimed sender. (This is where the trace notation is not specific enough. Clearly, it is not sufficient to have the information present in the messages; we also need to verify that the information is correct.)

The program for the server is unchanged:

$$\begin{aligned}\text{SENDSERVERNONCE}_S(k) = \ & \mathbf{while}\ \mathit{true}\ \mathbf{do}\\ & \mathbf{recv}\ m;\\ & \mathit{receiver} \leftarrow \pi_1(m);\\ & \mathit{m1} \leftarrow \mathit{decrypt}(\pi_2(m), k(\mathit{receiver}));\\ & \mathit{sender} \leftarrow \pi_1(\mathit{m1});\\ & \mathit{m2} \leftarrow \mathit{decrypt}(\pi_2(\mathit{m1}), k(\mathit{sender}));\\ & \mathbf{send}\ \mathit{receiver}\ \{\mathit{m2}\}_{k(\mathit{receiver})}.\end{aligned}$$

It is straightforward to check that the attack traces above are not valid protocol traces for the above programs. Of course, that these attacks cannot be carried out as above does not mean that no such attach can be carried out at all, perhaps in a more convoluted way; hence the need for formal verification of protocols.

6.3 Security Properties

We saw a number of scenarios in the last section that could be seen as being attacks or not, depending on the context of use of the protocol. The context of use can be captured by specifying *properties* that we would like the protocols to satisfy. There are a number of security properties that have been identified in the literature, and that we now review.

The most basic security property studied in the literature is that of *confidentiality*, or *secrecy*. It is also in some sense the easiest to define, at least at the level of symbolic approaches to analyze protocols. Intuitively, m is a secret between A and B if only A and B know m. (This generalizes in the obvious way to more than two agents.) We can recast many properties of protocols as preserving or exchanging a secret between parties. Depending on the framework, and the assumptions made about the system, the interpretation of the term "know" in the definition of confidentiality vary. The typical definition found in the literature on symbolic approaches to security protocol analysis takes the statement "the adversary knows m" as meaning that the adversary can, based on the messages he has intercepted and his capabilities, derive the message m. Thus, confidentiality of m is taken to mean "the adversary cannot derive m".

Confidentiality is relatively easy to define. Most other properties do not have such a clear definition. A typical property found in the literature is that of *authentication*. We often talk about authentication protocols, that is, protocols whose purpose it is to convince one agent of the identity of another agent. The literature is fairly divided as to what authentication actually means, and the general consensus on this point is that authentication as it is commonly used refers to at least two different notions. We consider the two main ones in this section: *message authentication*, where an agent is attempting to establish the source of a particular message, and *agent authentication*, where an agent is attempting to determine the identity of another agent on the network.

The intuition for message authentication is simple. An agent receives a message m, and wants to ascertain that it originated from agent B. This property can often be reduced to the confidentiality of a piece of data. For instance, m may be cryptographically bound to some secret known only to A and B. (The key itself is often the secret, in a symmetric-key setting.) In this case, m must have come from A and B, assuming the protocol did not leak the secret. Thus, authentication and confidentiality are not completely independent properties.

Agent authentication is a more difficult property to characterize. Roughly, it concerns the identity of a peer in a connection, or during a particular interaction. Formalizing this property depends intrinsically on what we take to be the definition of a connection, or an interaction. Often, this property makes the most sense when

there is an implicit notion of connection in the system being studied, for instance, if communication is via secure private channels. Another setting in which this property arises is when a protocol is first run to establish a secret key used to define what may best be called a "virtual connection" between two agents, after which this property can be used to actually identify the agent on either end of the virtual connection.

A common way to establish both kind of authentications is to consider *authenticating events*. Say that an event E in a system authenticates another event E' if the occurrence of E guarantees that event E' occurred. These events are often events of the protocol. Consider for example a way to establish that the message m received by B in protocol SENDSHAREDNONCE actually came from A. The reception of $\{m, n_B\}_{k_{AB}}$ should authenticate the sending of $\{m, n_B\}_{k_{AB}}$ by A. Intuitively, if B receives $\{m, n_B\}_{k_{AB}}$ from A, then B should know that A was the sender, since the sending event must have happened.

Artificial events can also be introduced for the purpose of analysis. For instance, it is common to introduce events $begin$ and end (perhaps parametrized) performed by agents to define points of interest in the protocol. We can modify SENDSHAREDNONCE, for instance, as follows:

$$\begin{array}{lrcl} 1. & A & : & begin(m, A, B) \\ 2. & A \rightarrow B & : & A \\ 3. & B \rightarrow A & : & n_B \\ 4. & A \rightarrow B & : & A, \{m, n_B\}_{k_{AB}} \\ 5. & B & : & end(m, A, B). \end{array}$$

The property can be recast as: event $end(m, A, B)$ authenticates $begin(m, A, B)$, that is, if B performs an end event with values m, A, B, then A must have performed a begin event with values m, A, B. This separates the specification of the protocol from the actual messages being exchanged in the protocol. The events $begin$ and end are typically called assertions, and verifying that these assertions match involve checking the correspondence of these assertions; hence the name *correspondence assertions*.

6.4 Symbolic Approaches to Security Protocol Analysis

The study of security protocols is not a recent trend, and many methods have been developed. In this section, we give a quick overview of classes of commonly used approaches. (The classification is somewhat arbitrary, but still useful.) We describe symbolic approaches exclusively; these are characterized by reasoning about the "combinatorial" properties of protocols, that is, properties that depend on which

messages can be derived and constructed from existing messages. In contrast, computational approaches put a greater emphasis on the computational aspects of the encryption, such as the probability of extracting bits of information from encrypted messages. We return to this view briefly in Chapter 10.

6.4.1 Model-Based Approaches

The class of model-based approaches to security protocol analysis is broad, and includes all the approaches that rely on modeling a protocol using techniques often used in the verification community. The idea is simply that we can think of a protocol executing in an environment as a software system like any other, characterized by a set of states, and where evolution of the system corresponds to transitions between states, triggered by actions performed by the agents, or the system itself.

To construct a model of a protocol, we first need to specify the set of states that arise during a protocol interaction. A state captures all the information relevant to the analysis of the system, for instance, the keys in possession of all the agents, the messages in transit between agents, the messages intercepted by the adversary, and so on. Transitions between states correspond to actions taken by the agents, such as an agent sending a message, or the network delivering a message to an agent. Adversary actions can also give rise to transitions between states, such as the adversary constructing a new message before sending it. Generally, a model is generated implicitly from a set of initial states and a transition relation between states; the model of the protocol corresponds to all the states reachable from the initial states following transitions.

Specifications in model-based approaches generally take the form of asserting that certain "bad states" cannot be reached from the initial states. A bad state can be, for example, a state where the adversary knows a message that was meant to remain secret, or a state where authentication has failed, perhaps because the adversary managed to end an interaction with an agent convincing him or her that some other agent has initiated the interaction. Note that these approaches force the verifier to specify the bad things that should not happen, rather than the good things that must. The task of verifying that a protocol satifisties the specification is therefore the task of checking that bad states cannot be reached,

As a representative of this class of approaches, we consider MSR, a framework based on a form of term rewriting called *multiset rewriting*. In MSR, the state of the system is represented as a multiset of facts, where each fact is an atomic formula of the form $P(t_1, \ldots, t_k)$ for variable-free terms $t_1, \ldots, t_k$ taken from a term algebra over a given signature. (See Section 3.1 for an overview of term algebras.) The signature provides at least encryption and pairing constructors, of the kind used in Example 3.1. The notation $P(\vec{t})$ is often used for $P(t_1, \ldots, t_k)$. A

multiset is just like a set, except that it allows repetition of elements; for example, the multiset a, b, b, c is different from the multiset a, a, b, b, c, since the number of as in each multiset differ. Predicates are used to record the state of each agent in the system, the messages on the network that have not yet been delivered, the messages intercepted and derived by the adversary, and so on.

Transitions between states are specified using rewrite rules. A rewrite rule has the form

$$P_1(\vec{t_1}), \ldots, P_m(\vec{t_m}) \longrightarrow \exists x_1 \ldots \exists x_k . Q_1(\vec{t'_1}), \ldots, Q_n(\vec{t'_n})$$

where the atomic formulas $P_1(\vec{t_1}), \ldots, P_m(\vec{t_m})$ on the left-hand side of the rule may have free variables, while all the variables in $Q_1(\vec{t'_1}), \ldots, Q_n(\vec{t'_n})$ must either appear on the left-hand side, or be one of $x_1, \ldots, x_k$. A rule is applicable in a state if each atomic formula on the left-hand side of the rule matches a distinct atomic formula in the state. (Two atomic formulas match if they are the same predicates, applied to terms that match; two terms match is one is a substitution instance of the other.) Moreover, applying a rule in a state involves identifying the matched atomic formulas $P_1(\vec{t_1}), \ldots, P_m(\vec{t_m})$ in the state, keeping track of the term matched by each variable in the matched formulas, removing the matched atomic formulas from the state, and forming the new state by adding $Q_1(\vec{t''_1}), \ldots, Q_n(\vec{t''_n})$, where $Q_i(\vec{t''_i})$ is obtained from $Q_i(\vec{t'_i})$ by replacing every variable appearing on the left-hand side of the rule by the term it matched, and replacing the variables $x_1, \ldots, x_k$ by new constants that do not occur in the state or in any of the rules. It is this existential quantification that makes this framework usable for protocol analysis, as it can be used to model nonce generation.

Representing protocols in MSR is relatively straightforward. As an example, consider the following implementation of the SENDSHAREDNONCE protocol. The atomic formula $N(x)$ indicates that the message x is on the network, that is, it has been sent by an agent, but has not been received by any agent. Clearly, there can be many messages in the network at the same time. Protocols are described by giving rewrite rules corresponding to the initiator and the responder. The rules for the initiator, in this case A, are straightforward:

$$A_0(m, k_{AB}) \longrightarrow N(A), A_1(m, k_{AB})$$
$$A_1(m, k_{AB}), N(n_B) \longrightarrow A_2(m, k_{AB}), N(A, \{m, n_B\}_{k_{AB}}).$$

(As this example shows, it is often necessary to have predicates that take a variable number of arguments. Rather than using a family of predicates, each of a different arity, we use a single predicate, overloaded at all appropriate arities.) Executing a step of the protocol corresponds to applying one of these rewrite rules. The predicates A_0, A_1, and A_2 capture the states of the agent: the initial state, after the

first message has been sent, after the second message has been sent, respectively. The first transition rule says that in the initial state, represented by the atomic formula $A_0(m, k_{AB})$ indicating that the message he wants to send is m and the key to use is k_{AB}, the protocol can execute by putting the message A on the network, and moving the agent to state $A_1(m, k_{AB})$. The second transition rule says that when the agent is in state $A_1(m, k_{AB})$, and if a value n_B appears on the network, then the protocol can execute by receiving the valure n_B, putting the message $(A, \{m, n_B\}_{k_{AB}})$ on the network, and moving the agent to state $A_2(m, k_{AB})$.

The transition rules for the responder are similar:

$$B_0(k_{AB}), N(A) \longrightarrow \exists n_B.B_1(k_{AB}, n_B), N(n_B)$$
$$B_1(k_{KAB}, n_B), N(A, \{m, n_B\}_{k_{AB}}) \longrightarrow B_2(k_{AB}, n_B, m).$$

The first transition rule uses existential quantification to create a fresh nonce. More precisely, the first rule says that if the responder is in his initial state $B_0(k_{AB})$ and a message A appears on the network, then the protocol can execute by creating a fresh nonce n_B, putting it on the network via $N(n_B)$, and moving the agent to state $B_1(k_{AB}, n_B)$, which also records the nonce. The second transition rule is straightforward.

To specify the initial states of the system, an initialization rule is necessary. The following rule creates a shared key between A and B, and initializes A to the initiator's first state, and B to the responder's first state, both states recording the created shared key.

$$\longrightarrow \exists k_{AB}.A_0(k_{AB}), B_0(k_{AB}).$$

How do we represent the adversary? There are a number of ways of defining an adversary. Consider the following simple definitions, which represents a Dolev-Yao adversary using symmetric keys. The idea is to model the knowledge of the adversary using a predicate M (for *memory*); the atomic formula $M(m)$ represents the fact that the adversary was able to derive the message m. The following transition rules describing the adversary reflect the inference rules defining the $\vdash_{DY}$ relation in Section 6.2:

$$N(x) \longrightarrow M(x)$$
$$N(x, y) \longrightarrow M(x, y)$$
$$M(x, y) \longrightarrow M(x), M(y)$$
$$M(\{x\}_k), M(k) \longrightarrow M(x), M(k).$$

(The first two rules say that the adversary can intercept any message on the network.) In order to send messages to other agents, we need to describe the messages that the adversary can construct. The predicate C is used to denote those messages

that the adversary can construct. The following translation rules describe them:

$$\begin{aligned} M(x) &\longrightarrow C(x), M(x) \\ C(x), C(y) &\longrightarrow C(x, y) \\ C(x), M(k) &\longrightarrow C(\{x\}_k), M(k) \\ C(x) &\longrightarrow N(x) \\ C(x, y) &\longrightarrow N(x, y) \\ &\longrightarrow \exists x.C(x). \end{aligned}$$

Intuitively, any message that the adversary was able to derive can be constructed, as well as encryptions thereof, and pairings. Moreover, the last transition rule says that the adversary can generate fresh messages.

Confidentiality properties are straightforward to express in this framework. Following the discussion in Section 6.3, the confidentiality of a message m amounts to ensuring that the adversary is not able to derive m. Since the predicate M represents the messages derivable by the adversary, m remains confidential if the atomic formula $M(m)$ does not occur in any state reachable by transitions from the initial state of the system.

Authentication can be captured using the notion of authenticating events, as we saw in Section 6.3. Here, the events are the states of the various agents. Consider specifying that the sending of A's first message authenticates the reception of B's last message. Formally, this means that for any sequence of transitions from the initial state that leads to a state containing an atomic formula $B_2(k_{AB}, n, m)$ for some n, then the transitions must first go through a state containing an atomic formula $A_0(k_{AB}, m)$.

6.4.2 Process-Based Approaches

Process calculi are formal systems developped in the context of concurrency theory as a way to provide a foundation for concurrent computation, in much the same way that the λ calculus can be viewed as a foundation for sequential computation. While the main primitive feature of the λ calculus is function application, the main primitive for process calculi is that of the "interaction" of concurrent processes. Process calculi share the view that a process is a set of concurrently executing sequential processes that communicate (or interact) via shared information.

We model the system to be analyzed as a process in the syntax of the calculus. This process represents the protocols executed by all the agents, as well as the system behaviour as a whole, and possibly the adversary as well. This yields a syntactic representation of the system. The semantics of the calculus provides a description of the behaviour of the system. One advantage of using a calculus is

that we need not always compute this behaviour, but can often simply reason at the level of the syntactic process that represents the system. For instance, there may be a set of proof rules available to derive properties of processes by induction on the structure of the processes.

Specifying properties in process calculi is done in one of two ways. One possibility is to analyze the transitions given by the operational semantics of the process calculus, and specify properties of these transition in terms of reachability, as in the model-based approaches. The other possibility, more in the spirit of the process calculus tradition, is to use processes themselves as specifications. The intuition is straightforward: we write down a description in the calculus of an ideal system that clearly has the desired behaviour, and prove either that the original process "behaves the same" as the ideal process, or that every behaviour of the original process is a behaviour of the specification process; which approach to take depends on the kind of specification one is trying to establish.

The process calculus we now describe is CSP, the calculus of *communicating sequential processes*. CSP is a notation for describing sequences of events happening in a distributed system. Roughly speaking, an event in CSP is an occurrence in the system that can be caused by agents, and on which occurrence other agents can synchronize. An example of an event is an agent sending a message. Another agent can synchronize on the message being sent, and thus basically receive it. The basic CSP processes are $Stop$, which is the process that immediately terminates, processes of the form $event \rightarrow P$ (a process that performs $event$, and then behaves as process P), and processes of the form $?event \rightarrow P$ (a process that awaits for an event, and once matched, behaves as process P). More complicated processes can be constructed from choices, so that $P \,\Box\, Q$ represents a process that behaves either as P or as Q. Finally, processes can execute in parallel, where $P|[R]|Q$ represents P and Q executing in parallel, synchronizing on the events in the set R (while every other event happens independently).[1] CSP allows parametrized events, of the form $x.y.z$, which is an event x parametrized by values y and z. The matching notation is extended to allow for matching part of the event exactly, and allowing variables. The common use of such a notation is to have events of the form $c.5$, which can be understood as send value 5 on channel c, which can be matched by an event of the form $c?5$, which matches c exactly, and binds x to the received value, here 5. Here is the process corresponding to the initiator in the protocol SENDSHAREDNONCE given above, for agent i sending to agent j, with message m:

$$INIT(i, j, m, k) = trans.i!j!i \rightarrow$$

[1] Full CSP allows other constructs. For instance, parallel execution can synchronize on events, and there is a nondeterministic choice, representing the system choosing rather than the process. Processes are also definable by recursion, or by mutually recursive process definitions.

$$
\begin{aligned}
&rec.i.j?n_B \rightarrow \\
&trans.i!j!i!\{m, n_B\}_{k(i)} \rightarrow \\
&Stop.
\end{aligned}
$$

The parameterization of the process is similar to that in Section 6.2.

The responder process is similar:

$$
\begin{aligned}
RESP(i,k,n) = \; & rec.i?j?i \rightarrow \\
& trans.i.j.n \rightarrow \\
& rec.i.j?j?\{m, n\}_{k(j)} \rightarrow \\
& Stop.
\end{aligned}
$$

The semantics of a process P can be given in a number of ways, depending on what one wants to observe about a process. The simplest semantics, and the one used by most security protocol analysis work in CSP, is the trace model. Roughly speaking, the semantics associate to every process P a set of traces $traces(P)$, representing the sequence of events performed by the process. In the case of parallel processes, $traces(P)$ is an interleaving of the traces of the subprocesses of P.

How do we use CSP to reason about security protocols? The first step is to model the system. This is done by creating a process representing the system. This process includes a process describing the program of each agent in the system (including trusted servers) and a process describing the behaviour of the adversary. Thus, a system has the form:

$$SYS = (|||_{j \in AG} AGENT_j) \; |[trans, rec]| \; ADV,$$

where $trans$ and rec are events corresponding to the sending and reception of messages. As an example, consider the system SYS for the SENDSHAREDNONCE protocol. Assume a set of agents AG. We can represent the agents' behaviour by the processes:

$$AGENT_i(m,n,k) = \Box_{j \in AG} \left(INIT(i,j,m,k) \,\Box\, RESP(i,k,n) \right).$$

The adversary is specified by encoding a Dolev-Yao adversary, using the $\vdash_{DY}$ relation of Section 6.2. The process is recursive, and parameterized by a set S of intercepted messages.

$$
\begin{aligned}
ADV(S) = \; & trans?i?j?m \rightarrow ADV(S \cup \{m\}) \\
& \Box \left(\Box_{j \in AG, S \vdash_{DY} m} rec.i!j!m \rightarrow ADV(S) \right).
\end{aligned}
$$

(This could be written up completely in CSP, rather than relying on the $\vdash_{DY}$ relation.)

How do we specify properties of processes? The basic way, given the semantics above, is to specify a property of the set $traces(P)$. First, consider confidentiality.

Intuitively, a term t is secret if the adversary cannot deduce it. How do we specify this in CSP? One approach is to change the code of the adversary to add a new channel $knows$ and have the adversary nondeterministically send any message he can derive to that channel. Thus, $ADV(S)$ can be written:

$$\begin{aligned} ADV(S) = \; & trans?i?j?m \rightarrow ADV(S \cup \{m\}) \\ & \Box\, (\Box_{j \in AG, S \vdash_{DY} m} rec.i!j!m \rightarrow ADV(S)) \\ & \Box\, (\Box_{S \vdash_{DY} m} knows!m \rightarrow ADV(S))\,. \end{aligned}$$

Following the discussion in Section 6.3, a message m is confidential if the adversary cannot derive m. Since every message that the adversary can derive can be sent to the channel $knows$, m is confidential if the adversary cannot send m on channel $knows$. This can be formalized by saying that for every trace $tr \in traces(SYS)$, $knows.m$ does not appear in tr. Using the notation $tr \upharpoonright H$ (where H is a set of events) to represent the subtrace of tr consisting only of events from H, and using $\langle\rangle$ to represent the empty trace, confidentiality of M can be written $tr \upharpoonright knows.m = \langle\rangle$ for all $tr \in traces(SYS)$.

What about authentication? Again, we can use authenticating events. For SENDSHAREDNONCE, one authentication properties says that the sending of the initiator's first message authenticates the reception of the responder's last message. Formally, this means that for all agents A, B, and any trace tr in $traces(SYS)$, if $trans.A.B.A.\{m, n\}_{k_{AB}}$ appears in tr, then $trans.A.B.A$ appears in tr. Formally, this can be written as: $tr \upharpoonright trans.A.B.A.\{m, n\}_{k_{AB}} \neq \langle\rangle$ implies $tr \upharpoonright trans.A.B.A \neq \langle\rangle$, for all A, B, k_{AB}, n, and for all $tr \in traces(SYS)$.

There is another approach to specification in CSP. A process P *refines* a process Q, written $P \sqsubseteq Q$, if $traces(P) \subseteq traces(Q)$. If we view Q as a specification denoting a set of "good" traces, and if $P \sqsubseteq Q$, then P can be seen as an implementation that satisfies Q, in the sense that every trace of P is a good trace. (This of course relies on being able to characterize the set of good traces as a process Q, and on being able to characterize goodness as a property of traces.) Refinement permits the use of the process language itself as a specification language. Moreover, writing specifications in the form of a refinement property enables the use of tools that have been developed to automatically establish refinement relation between CSP processes.

6.4.3 Logic-Based Approaches

What distinguishes the last class of approaches is the emphasis on the specification of properties. More precisely, they focus on developing a formal language in which to write down the properties of the protocols that one intends to verify.

The main formalisms for logic-based verification split across two distinct lines.

On one side, there are first-order and higher-logic logic approaches, and on the other there are modal logic approaches (often propositional). An expressive language such as higher-order logic allows us to model the protocol directly in the logic (in the form of formulas that characterize the behaviour of the model). We can then establish that a protocol satisfies a specification by showing that the specification logically follows from the logical description of the protocol. These approaches naturally lead to theorem-proving systems, often partially automated. One advantage of this approach is that infinite systems can be proved correct, by using inductive techniques. Alternatively, augmented with the appropriate support for cryptography and adversaries, modal-logic approaches lead to specification languages in many ways compatible with the model-based approaches to protocol analysis. While less expressive than approaches based higher-order logic, modal logics often are decidable, and often support efficient model-checking procedures.

As an illustration of logic-based approach to reason about protocols, we describe what is historically the most important modal logic for security protocol analysis. BAN logic (named after the researchers who introduced it, Mike Burrows, Roger Needham, and Martín Abadi) is a modal logic of belief that provides primitives for reasoning about protocols in a cryptographic setting. Contrary to the model-theoretic approach to logic that we have considered until now in this dissertation, BAN logic was essentially developed axiomatically, by giving axioms and inference rules for deriving new beliefs from old beliefs. BAN logic has formulas that say that k is a good key for communicating between agents A and B (a shared key known only to A and B), written $A \overset{k}{\leftrightarrow} B$, that m is a secret between A and B, written $A \overset{m}{\rightleftharpoons} B$, that agent A believes the formula F, written $A\ \mathbf{believes}\ F$, that agent A controls the truth of formula F, written $A\ \mathbf{controls}\ F$, that agent A sent a message meaning F, written $A\ \mathbf{said}\ F$, that agent A received a message (and was able to read it, perhaps by decrypting it if it was encrypted and she had the right decryption key) meaning F, written $A\ \mathbf{sees}\ F$, that the message m is fresh, that is, has never been used before, written $\mathbf{fresh}(m)$. A sample BAN logic inference rule is

$$\frac{A\ \mathbf{believes}\ A \overset{k}{\leftrightarrow} B \quad A\ \mathbf{sees}\ \{F^l\}_k \quad l \neq A}{A\ \mathbf{believes}\ B\ \mathbf{said}\ F},$$

which intuitively says that if agent A believes that k is a good key between herself and B, and she receives a message encrypted with key k that did not originate with her, then she believes that B sent the original message. More inference rules are given in Figure 9.1. We will have much more to say about BAN logic in Chapter 9, including a more careful interpretation of formulas. In the remainder of this section, we illustrate how BAN logic can be used to prove properties of protocols.

One difference between BAN logic and the other approaches we described ear-

lier in this section is that BAN logic does not attempt to model the protocol directly. Approaches such as MSR and CSP model the protocol using essentially a state-based transition system, either explicitly in the case of MSR, or implicitly through the semantics in the case of CSP. In BAN logic, the reasoning occurs directly on the trace of the protocol. However, because the trace of the protocol does not quite carry enough information to permit the kind of reasoning advocated by BAN logic, a process known as *idealization* needs to be performed on the protocol. Roughly speaking, idealization consists of replacing the messages in the protocol by formulas of BAN logic that capture the "meaning" of the message exchanged by the agents. For instance, if an agent A sends a key k to an agent B, with the intention of sharing a key that A considers good, then a suitable idealization of this protocol step is to have A send the formula $A \overset{k}{\leftrightarrow} B$ to B. Much of the difficulties in reasoning about protocols using BAN logic reside in this idealization phase. Here is a possible idealizaiton of the protocol SENDSHAREDNONCE:

$$\begin{array}{lll} 2'. & B \to A & : \quad n_B \\ 3'. & A \to B & : \quad \{A \overset{m}{\rightleftharpoons} B, n_B\}_{k_{AB}}. \end{array}$$

Message (1) in the original protocol carries no information that BAN logic can use, so it has been removed from the idealized protocol. Message (2') is unchanged from the original. Message (3') is the idealization of agent A sending m packages with n_B to B; this idealization consists of A sending the formula $A \overset{m}{\rightleftharpoons} B$ to B, indicating that A considers m to be a secret at that point.

Reasoning about the idealized protocol consists of laying out the initial belief of the agents, and deriving new beliefs from those and from the messages exchanged between the agents using the inference rules of the logic. For SENDSHARED-NONCE, consider the following reasonable initial beliefs. First, both agents should believe that the key they share (k_{AB}) is a key that has not been compromised. These beliefs are captured by the BAN logic formulas

$$A \textbf{ believes } A \overset{k_{AB}}{\leftrightarrow} B$$

$$B \textbf{ believes } A \overset{k_{AB}}{\leftrightarrow} B.$$

Another assumption is that the nonce that B uses has not already been used, that is, the nonce is fresh. This belief is captured by the BAN logic formula

$$B \textbf{ believes fresh}(n_B).$$

Finally, the message that A wants to send to B should initially be secret (in order to prove that this message remains secret after a protocol run). This belief is captured by the BAN logic formula

$$A \textbf{ believes } A \overset{m}{\rightleftharpoons} B.$$

We can now derive new beliefs from these initial beliefs, and the messages exchanged by the agents. After an idealized protocol step of the form $A \rightarrow B : F$, B receives a message meaning F, and thus the formula B **sees** F can be used to update B's beliefs. In the idealized protocol for SENDSHAREDNONCE, after having received message (3'), the BAN formula

$$B \textbf{ sees } \{A \overset{m}{\rightleftharpoons} B, n_B\}_{k_{AB}}$$

holds. Combined with the initial belief B **believes** $A \overset{k_{AB}}{\leftrightarrow} B$, one of the BAN logic inference rules (rule R1) allows us to derive

$$B \textbf{ believes } A \textbf{ said } (A \overset{m}{\rightleftharpoons} B, n_B). \tag{6.1}$$

From the initial belief B **believes fresh**(n_B), it is clear that B should believe any message combined with n_B to be fresh as well. This is captured by one of the BAN logic inference rules (rule R9), which lets us derive

$$B \textbf{ believes fresh}(A \overset{m}{\rightleftharpoons} B, n_B).$$

Finally, from this last formula and formula (6.1), applying one of the BAN logic inference rules (rule R3) lets us derive

$$B \textbf{ believes } A \textbf{ believes } A \overset{m}{\rightleftharpoons} B. \tag{6.2}$$

Thus, after protocol interaction, B believes that A believes that m is a secret between A and B. We can say something stronger if we make the additional assumption that B initially believes that the secrecy of m is completely up to agent A, that is, if B believes that A controls the truth of $A \overset{m}{\rightleftharpoons} B$. From this initial belief B **believes** A **controls** $A \overset{m}{\rightleftharpoons} B$ and formula (6.2), we can apply one of the BAN logic inference rules (rule R4) to derive

$$B \textbf{ believes } A \overset{m}{\rightleftharpoons} B.$$

This simple example illustrates the kind of axiomatic reasoning that can be performed using BAN logic.

One consequence of the decision to reason directly about the protocol text is that the adversary is not modeled directly within the BAN logic framework, but is rather implicit in the valid inference rules of the logic. This makes it difficult to see what is happening operationally, or change the system to accomodate different adversaries.

BAN logics seems a reasonable specification language, as far as attempting to capture the right concepts for security protocol analysis. However, BAN logic has been the subject of many criticisms. The two main criticisms affect BAN logic both as a specification language and as an approach to protocol verification. First,

the logic has a multitude of operators, but no semantics to speak of.[2] The main consequence of not having an independently motivated semantics is that it is not clear exactly what one is proving when a "proof" of security is exhibited for a protocol. In general, when a BAN-style analysis manages to exhibit a bug in a protocol, chances are good there is indeed a bug, but a proof of security does not guarantee much. Attempts have been made to supply a more adequate semantics for BAN-style logics, but without simplifying the logic. Second, the *verification method* associated with the logic, that is, how to use the logic to verify protocols, itself relies on a dubious idealization process that, among other things, is insensitive to the order of protocol steps.

6.4.4 Discussion

The above classes of approaches represent the most common ones. (A number of approaches do not fit so cleanly in that classification; we examine one such, strand spaces, in more detail in Chapter 7.) They each have their advantages and disadvantages with respect to the three aspects of interest, namely, how protocols are modeled, how security properties are specified, and how security properties are verified.

– For model-based approaches, the emphasis is on the models:
  (a) protocols are modeled explicitly via state-based transition systems: a set of states, with distinguished initial states, and a transition function from states to states indicating the evolution of the system;
  (b) a specification is a predicate on states, often indicating which states are bad states;
  (c) verification is performed by proving that no bad state is reachable from the initial states.

  Thus, the specifications are restricted, often a simple language of description of states, with perhaps some extensions ot deal with the temporal evolution of the states.
– For process-based calculi, compositionality of the modeling process is central:
  (a) protocols are described programmatically; this description is compositional, and generally higher-level than other approaches;
  (b) specifications are often described using either the properties of the programs, or as other programs (in the former case, they resemble specifications of the form for model-based approaches, in the latrer, they are "idealized" programs);

[2] The original semantics merely encodes the inference rules of the logic. Arguably, the Hoare-style presentation of the logic does provide some hints as to the meaning of the BAN logic operators, but this is far from being a satisfactory semantics.

(c) the verification is either performed by hand, or by proof systems.

– Finally, logic-based approaches put the focus on the specifications:
  (a) protocols are generally modeled as in the model-based approaches, by specifying transition systems;
  (b) specifications are written in a logical language, perhaps with security-specific primitives;
  (c) verification varies, from model-checking for approaches based on modal logic, to theorem-proving for more expressive logics.

From the point of view of specification, the most flexible approach is certainly the logic-based approaches, especially those approaches like BAN logic that are explicitly geared towards expressing security properties. (There are some questions pertaining to the foundation of those primitives; we will come back to these in Chapter 9. For the time being, they can be taken for granted.) However, the verification procedure for BAN-style logics, based on idealization, is very problematic. There are logics with more straightforward verification mechanisms, such as temporal logic, epistemic logic, and higher-order logic, but they tend to be more generic and often do not support the kind of primitives that permit the natural expression of security properties.

Moreover, the adversary model in logic-based approaches generally cannot be significantly altered, and thus does not really provide the flexibility needed to address some of the issues discussed in Section 6.2. The process-based approaches can support different adversary models, as witnessed by the CSP examples in Section 6.4.2. However, the specification language for process-based approaches does not seem easily expressible as a standard specification language augmented with security primitives.

In the next chapters, we develop a framework to try and get the best of all these worlds, from the point of view of modeling and specifications. Specifically, our goal is to get a framework that:

– models protocols using a general framework that can capture the knowledge of agents;
– supports deriving models from a protocol notation that is compositional;
– represents adversary capabilities in a natural and flexible way;
– supports a logical specification language that is expressive enough to capture useful security primitives;
– relates specifications of properties to the models of protocols in an intuitive way.

While we will not talk about automatic verification in this dissertation, it should be clear that a further desideratum is to have the framework support effective verification procedures.

Notes

Cryptography is a field with a long history. Stinson [1995] and Schneier [1996] give excellent overviews of the practice of cryptography. Goldreich [2001] gives an excellent account of the theoretical side of modern cryptography.

The symbolic approach to security protocol analysis goes back to Needham and Schroeder [1978], who were among the first to point out the fact that some attacks on protocols are essentially independent of the encryption scheme. Dolev and Yao [1983] first formalized such attacks using their now standard model of the adversary.

Protocol SENDSERVER is a value-passing variant of a protocol by Woo and Lam [1992]. The first attack presented in Section 6.2 is an adaptation of the attack due to Abadi and Needham [1996] on the original Woo-Lam protocol. The second attack is an adaptation of the attack due to Anderson and Needham [1995]. Clark and Jacob [1997] offer an excellent literature review on security protocols.

Abadi [2000] gives a general overview of security properties for protocols. Confidentiality is the cornerstone of security properties. A more general notion of confidentiality is studied in the context of information flow. See McLean [1994] for an overview. Such a general notion of confidentiality is studied by Halpern and O'Neill [2002].

Gollmann [1996] gives a typical analysis of the various notions of authentication. He describes peer-entity authentication (what we called agent authentication) versus data-origin authentication (what we called message authentication). Authenticating events were used by Schneider [1996] in CSP. Begin and end assertions were first studied by Woo and Lam [1993]. Lowe [1997] gives a taxonomy of authentication properties in CSP using correspondence assertions. Most current frameworks for analyzing protocols use correspondence assertions in one form or another to specify authentication properties. Gollmann [2003] provides an overview and critique of this technique.

Roscoe [1996] advocates intensional specifications, which essentially say that a protocol works exactly as intended. They are meant to be protocol-dependent specifications. In contrast, extensional properties are meant to be protocol-independent. Boyd [1997] attempts a classification of security properties along the intensional and extensional specification lines. In a sense, correspondence assertions with begin and end events are extensional versions of authenticating events.

The multiset rewriting approach MSR is described in [Cervesato, Durgin, Lincoln, Mitchell, and Scedrov 1999]. It is strongly related to linear logic [Girard 1987]. MSR has been used to analyze, among others, version 5 of the Kerberos protocol [Butler, Cervesato, Jaggard, and Scedrov 2002]. It was also used to establish some of the first decidability and undecidability results for the symbolic

analysis of security protocols [Cervesato, Durgin, Lincoln, Mitchell, and Scedrov 1999]. Other model-based approaches include the NRL protocol analyzer [Meadows 1996], derived from an earlier tool, the Interrogator [Millen, Clark, and Freedman 1987]. NRL is based on a logic-programming engine, and works by specifying an insecure state and attempting to construct a path to that state from an initial state, or proving that the state is unreachable. Meadows [1990] describes an approach that can accomodate partial knowledge of messages in NRL. Another logic programming tool is the protocol verifier of Blanchet [2001, 2002]. It has been used to verify an email protocol [Abadi and Blanchet 2003a], and forms the basis of a verification tool for Web Services [Bhargavan, Fournet, Gordon, and Pucella 2004].

The process calculus CSP is due to Hoare [1985]; Roscoe [1997] gives a modern account. An overview of the use of CSP for security protocol analysis is given by Ryan and Schneider [2000]. Rank functions [Schneider 1998] can be used in CSP to prove properties of infinite systems. FDR[3] [Roscoe 1994] can be used to automatically establish $P \sqsubseteq Q$. FDR works by explicitly enumerating and then exploring the state space of the system. Thus, it can only deal with finite state systems. Casper [Lowe 1998] is a compiler that takes a message-passing style representation of protocols and produces a CSP process representing the protocol; the compiler invokes FDR for checking the built-in properties that can be specified along with the protocol.

Another popular process calculus framework for analyzing security protocols is the spi calculus of Abadi and Gordon [1999]. The spi calculus is based on the π calculus of Milner [1999] (see also Sangiorgi and Walker [2001]), an extension of CCS [Milner 1989] with communication channels. The main feature here is name hiding, where the name of a channel can be hidden from other processes. The spi calculus further extends the π calculus with cryptographic primitives. A specification in the spi calculus is simply a spi process that is "obviously" correct, perhaps because it relies on a private channel for communication. A process satisfies a specification if it "behaves the same" in all contexts. Making this precise requires a notion of observational equivalence. Intuitively, two processes p_1 and p_2 are observationally equivalent if no context $C[\cdot]$, that is, a process calculus term with a hole, can distinguish p_1 and p_2 in terms of what can be observed. Thus, if $C[p_1]$, that is, the context C where the hole is "plugged" by p_1 deadlocks, but $C[p_2]$ does not, then $C[\cdot]$ can distinguish p_1 and p_2. Different notions of observational equivalence for the spi calculus can be defined [Abadi and Gordon 1999; Abadi and Gordon 1998; Boreale, de Nicola, and Pugliese 2001]. Focardi, Gorrieri, and Mar-

[3] Failure-Divergence Refinement, a product of Formal Systems (Europe) Ltd.

tinelli [2003] study various notions of authentication properties via observational equivalence.

The key feature of the spi calculus is that it does not require an explicit description of the adversary. Rather, by consider arbitrary contexts, the idea is that we get to reason about any adversary that can be expressed as a process in the spi calculus. Thus, we get automatic quantification over all attackers expressible in the system. Other process calculi can be extended with cryptography, in the spirit of the spi calculus. For instance, the sjoin calculus extends the join calculus [Fournet and Gonthier 1996], and was used by Abadi, Fournet and Gonthier [2002] to analyze secure channels.

It is sometimes possible to recast process-based specifications into a more conventional form of specification. Behavioural equivalence, for instance, can sometimes be characterized logically, in the sense that there exists a modal logic over process such that P and Q are behaviourally equivalent if and only if they satisfy the same formulas of the logic [Hennessy and Milner 1985; Milner, Parrow, and Walker 1993]. CSP refinement can be similarly characterized [Stirling 2001]. Frendrup et al. [2002] discuss such a logic for the spi calculus.

Another approach to reasoning about protocols using the spi calculus is to introduce a type system. Abadi [1999] and Abadi and Blanchet [2003b] develop type systems for the spi calculus that captures a form of secrecy. Recently, Gordon and Jeffrey [2001, 2002a, 2002b] have extended the type system of Abadi to essentially prove within a type system correspondence assertions between different entities in a communication protocol. This type system is a form of effects system [Tofte and Talpin 1997].

One of the most successful modern approaches to logic-based analysis of protocols is the *inductive assertions* method [Paulson 1998], formalized in higher-order logic [Andrews 1986], and proved using the higher-oder logic interactive theorem prover Isabelle [Paulson 1994]. TAPS [Cohen 2000; Cohen 2002] is a verifier based on first-order logic, and uses similar ideas. There has been a fair amount of work on applying temporal logic to the problem of reasoning about security protocol analysis; see for example [Gray and McLean 1995; Mitchell, Mitchell, and Stern 1997; Clarke, Jha, and Marrero 1998]. Some approaches reminiscent of Dynamic Logic [Harel, Kozen, and Tiuryn 2000] have also recently emerged [Durgin, Mitchell, and Pavlovic 2001].

BAN logic was introduced by Burrows, Abadi, and Needham [1990a]. A vast literature, starting with Abadi and Tuttle [1991], has emerged to follow up on their work, extend the logic, and attempt to supply it with a more adequate semantics [Gong, Needham, and Yahalom 1990; Syverson and Oorschot 1994; Stubblebine and Wright 1996; Wedel and Kessler 1996]. Syverson and Cervesato [2001] provide a good overview. The main problem with some of those approaches is that

semantics of the logic (to the extent that one is provided) is typically not tied to protocol executions or attacks. As a result, protocols are analyzed in an idealized form, and this idealization is itself error-prone and difficult to formalize [Mao 1995]. Snekkenes [1991] showed that because BAN idealization was insensitive to the order of protocol steps, some protocols are deemed correct by BAN when they are in fact flawed. Syverson [1990] and Bieber [1990] consider approaches based on logics of knowledge that do not suffer from those particular problems.

7
Modeling Security Protocols

To analyze protocols, we need a way to represent them and model the aspects that are relevant for the properties that we want to prove. As we saw in the previous chapter, there are many frameworks for representing and reasoning about security protocols. In this chapter, We present a general framework for modeling security protocols that is amenable to the kind of knowledge-based logical analysis performed later in the dissertation. The models used are adapted from models typically used in distributed computing.

After presenting the general framework and defining security systems, we present a few ways to generate such security systems from descriptions of protocols. The first is to derive systems from programs of the kind used in Chapter 6 to describe protocols. In essence, the semantics of the programs are given in terms of security systems. Since the language is related to process calculi, this highlights the relationship between security systems and the models underlying process calculi. The second way of generating systems is to derive them from other representations of protocols. By way of illustration, we show how to do this starting from a popular representation based on *strand spaces*.

Note that the framework in this chapter does not deal with the adversary. Modeling adversaries is discussed in Chapter 8.

7.1 Security Systems

The multiagent systems framework provides a model for knowledge that has the advantage of also providing a discipline for modeling executions of protocols. A multiagent system consists of n agents, each of which is in some local state at a given point in time. Assume that an agent's local state encapsulates all the information to which the agent has access. In the security setting, the local state of an agent might include some initial information regarding keys, the messages she has sent

and received, and perhaps the reading of a clock. In a poker game, a player's local state might consist of the cards he currently holds, the bets made by 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). The basic framework makes no assumptions about the precise nature of the local state.

We can then view the whole system as being in some global state, which is a tuple consisting of each agent's local state, together with the state of the environment, where the environment consists of everything that is relevant to the system that is not contained in the state of the agents. Thus, a global state has the form $(s_e, s_1, \ldots, s_n)$, where s_e is the state of the environment and s_i is agent i's state, for $i = 1, \ldots, n$. The actual form of the agents' local states and the environment's state depends on the application. For definiteness, let Loc_i be the set of local state for agent i, including Loc_e for the environment.

A system is not a static entity. To capture its dynamic aspects, define a run to be a function from time to global states. Intuitively, a run is a complete description of what happens over time in one possible execution of the system. For future reference, note that we have a great deal of flexibility regarding what counts as a "time step". We could, for example, take a "time step" to correspond of a tick on a global clock, a step in a protocol, or the amount of time needed to perform a computation. We could also, as BAN logic does, consider only two time instants: the first representing the state of the world before the protocol is run and the second time represents the state of the world after the protocol is run. The local state of each agent after the protocol is run will contain all the events the agent has participated in. It is up to the modeler to decide which notion of time is most appropriate for an analysis.

A *point* is a pair (r, t) consisting of a run r and a time t. For simplicity, take time to range over the natural numbers in the remainder of this discussion. At a point (r, t), the system is in some global state $r(t)$. If $r(t) = (s_e, s_1, \ldots, s_n)$, then take $r_i(t)$ to be s_i, agent i's local state at the point (r, t). Formally, define a system $\mathcal{R}$ to consist of a set of runs (or executions). Therefore, a system is just a trace model, except that there is much more flexibility in representing the states, as well as allowing for simultaneous events for different agents. It is compatible with the trace models used in process calculi, as well as the transition systems used by the model-based approaches.

It is relatively straightforward to model security protocols as systems. Since security protocols are essentially protocols based on messages exchanged between distributed participants, a natural class of systems to use is that of *message-passing systems*. Consider a fixed set M of messages. A *history* for agent i (over M) is a sequence of elements of the form $send(j, u)$, $recv(u)$, and $int(a)$, where $u \in M$

and a is some internal action. We think of $send(j, u)$ as representing the event "message u is sent to agent j", $recv(u)$ as representing the event "message u is received", and $int(a)$ as representing the event "internal action a was performed". Intuitively, i's history at (r, t) consists of i's initial state, taken to be the empty sequence, followed by the sequence describing i's actions up to time t. If i performs no actions in round t, then its history at (r, t) is the same as its history at $(r, t-1)$. For an agent i, let $r_i(t)$ be agent i's history in (r, t). An event e *occurs in* i*'s history in round* $t+1$ *of run* r if e is in (the sequence) $r_i(t+1)$ but not in $r_i(t)$.

In a message-passing system, the agent's local state at any point is its history. Of course, if h is the history of agent i at the point (r, t), then we want it to be the case that h describes what happened in r up to time t 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 do not shrink over time, and that every message received in round t corresponds to a message that was sent at some earlier round.

Given a set M of messages, define a *message-passing system* (over M) to be a system such that for each point (r, t) and each agent i, the following constraints are satisfied:

MP1. $r_i(t)$ is a history over M;
MP2. for every event $recv(u)$ in $r_i(t)$ there exists a corresponding $send(i, u)$ in $r_j(t)$, for some j;
MP3. $r_i(0)$ is the empty sequence and $r_i(t+1)$ is either identical to $r_i(t)$ or the result of appending one event to $r_i(t)$.

MP1 says that an agent's local state is its history, MP2 guarantees that every message received at round t corresponds to one that was sent earlier, and MP3 guarantees that histories do not shrink.

An *asynchronous message-passing system system* is a message-passing system that does not place any constraints on the relative order of events in different agents' histories beyond those imposed by MP1 and MP2. Such asynchrony can be captured by considering systems that consist of *all* runs satisfying MP1–3 for some set of histories. Formally, $\mathcal{R}$ is an *asynchronous message-passing system* if there exists a sequence $V_1, \ldots, V_n$, where V_i is a set of histories over some set M of messages, such that $\mathcal{R}$ consists of all runs satisfying MP1–3 where agent i's local state is a history in V_i at every point. The system $\mathcal{R}$ is the system *generated by* $V_1, \ldots, V_n$. Informally, the set V_i specifies the possible histories agent i could have. The system generated by $V_1, \ldots, V_n$ consists of all runs satisfying MP1–3 such that agent i's histories are in V_i for all i.

For the purposes of analyzing security protocols, define the class of *security systems*. The messages exchanged by the agents are taken from the symbolic encryp-

tion scheme $\mathcal{M}$ generated by a set $\mathcal{P}$ of plaintexts and a set $\mathcal{K}$ of keys. A security system is an asynchronous message-passing systems over $\mathcal{M}$ where the local state of an agent consists of the agent's initial information followed by the sequence of events that the agent has been involved in. An event is either the reception $recv(m)$ of a message m, the sending $send(i, m)$ of a message m to another agent i, or the update $update(var, m)$ of variable var to value m. We write $\langle evt_1, \ldots, evt_n \rangle$ for sequences of events, where $\langle\rangle$ is the empty sequence, and write $evts \cdot evt$ for the result of appending event evt to the sequence of events $evts$.

Assume a distinguished value $null \in \mathcal{M}$ used for undefined values, such as the initial value of variables. As we shall see shortly, the cryptographic operations all return $null$ if one of their argument is $null$; moreover, assume that if decryption fails, it returns $null$.

Define $\sqsubseteq$ on $\mathcal{M}$ as the smallest relation satisfying the following constraints:

(1) $m \sqsubseteq m$
(2) if $m \sqsubseteq m_1$, then $m \sqsubseteq (m_1, m_2)$
(3) if $m \sqsubseteq m_2$, then $m \sqsubseteq (m_1, m_2)$
(4) if $m \sqsubseteq m_1$, then $m \sqsubseteq \{m_1\}_k$.

Intuitively, $m_1 \sqsubseteq m_2$ if m_1 *could* be used in the construction of m_2. For example, if $m = \{\!|m_1|\!\}_k = \{\!|m_2|\!\}_k$, then both $m_1 \sqsubseteq m$ and $m_2 \sqsubseteq m$. Therefore, if we want to establish that $m_1 \sqsubseteq m_2$ for a given m_1 and m_2, then we have to look at all the possible ways in which m_2 can be taken apart, either by pairing or encryption, to finally decide if m_1 can be derived from m_2.

There are a number of useufl operations that can be performed on local states. The function $\theta(\ell)$ gives a mapping from variables to values as recorded in local state ℓ, by looking up the last binding for the variable (returning $null$ if none is found). Let θ_0 be the mapping that assigns $null$ to every variables.

$$\theta(\langle\rangle) = \theta_0$$

$$\theta(evts \cdot evt) = \begin{cases} \theta(evts)[var \mapsto m] & \text{if } evt = update(var, m) \\ \theta(evts) & \text{otherwise,} \end{cases}$$

where the notation $f[x \mapsto y]$ for a function f represents the function f' defined as

$$f'(z) = \begin{cases} y & \text{if } z = x \\ f(z) & \text{otherwise.} \end{cases}$$

The function $\rho(\ell)$ returns the value of the last received message in local state ℓ, or $null$ if no message has been received yet.

$$\rho(\langle\rangle) = null$$

$$\rho(evts \cdot evt) = \begin{cases} m & \text{if } evt = recv(m) \\ \rho(evts) & \text{otherwise.} \end{cases}$$

An important concept that arises in both the next section and the next chapter is that of a term representing a message in a local state. Roughly speaking, a term is just an expression that describes how to construct a message, by applying encryptions and decryptions, pairings and projections. A term can refer to variables, and when a message is constructed from the term, the values of those variables is taken from the bindings in the local state of an agent. The syntax of terms is as follows:

$$\begin{aligned} term ::= \; & m \\ & \mid var \\ & \mid \{term_1\}_{term_2} \\ & \mid decrypt(term_1, term_2) \\ & \mid (term_1, term_2) \\ & \mid \pi_1(term) \\ & \mid \pi_2(term) \\ & \mid received() \end{aligned}$$

where m is an arbitrary element of $\mathcal{M}$, and var is an arbitrary variable. We write $(term_1, \ldots, term_n)$ for $(term_1, (\ldots, (term_{n-1}, term_n)\ldots))$. Given a term $term$, we can *evaluate* the term in a given local state ℓ to get the message it represents, denoted $[\![term]\!]_\ell$:

$$\begin{aligned} [\![m]\!]_\ell &= m \\ [\![var]\!]_\ell &= \theta(\ell)(var) \\ [\![\{term_1\}_{term_2}]\!]_\ell &= \begin{cases} null & \text{if } m_1 = null \text{ or } m_2 = null \\ \{m_1\}_{m_2} & \text{otherwise} \end{cases} \\ &\quad \text{where } m_1 = [\![term_1]\!]_\ell \\ &\qquad\; m_2 = [\![term_2]\!]_\ell \\ [\![decrypt(term_1, term_2)]\!]_\ell &= \begin{cases} null & \text{if } m_1 = null \text{ or } m_2 = null \\ null & \text{if } m_1 = \{m_1'\}_k \text{ and } m_2 \neq k^{-1} \\ m_1' & \text{if } m_1 = \{m_1'\}_k \text{ and } m_2 = k^{-1} \end{cases} \\ &\quad \text{where } m_1 = [\![term_1]\!]_\ell \\ &\qquad\; m_2 = [\![term_2]\!]_\ell \end{aligned}$$

$$[\![(term_1, term_2)]\!]_\ell = \begin{cases} null & \text{if } m_1 = null \text{ or } m_2 = null \\ (m_1, m_2) & \text{otherwise} \end{cases}$$
$$\text{where } m_1 = [\![term_1]\!]_\ell$$
$$m_2 = [\![term_2]\!]_\ell$$

$$[\![\pi_1(term)]\!]_\ell = \begin{cases} null & \text{if } m = null \\ null & \text{if } m \neq (m_1, m_2) \text{ for some } m_1, m_2 \\ m_1 & \text{if } m = (m_1, m_2) \end{cases}$$
$$\text{where } m = [\![term]\!]_\ell$$

$$[\![\pi_2(term)]\!]_\ell = \begin{cases} null & \text{if } m = null \\ null & \text{if } m \neq (m_1, m_2) \text{ for some } m_1, m_2 \\ m_2 & \text{if } m = (m_1, m_2) \end{cases}$$
$$\text{where } m = [\![term]\!]_\ell$$

$$[\![received()]\!]_\ell = \rho(\ell).$$

For the purposes of this chapter, assume that the adversary in a security protocol can be modeled as just another agent. The adversary's information at a point in a run can be modeled by his local state. We return to adversaries and the intricacies of modeling them in Chapter 8.

7.2 A Language for Security Protocols

We now present a programming language IMPSEC that can be used to program an agent's actions in a system. This language is a formalization of the informal language we used in the previous chapter. The idea is to provide a program for each agent that, along with an execution context for the program, gives rise to a particular security system. In a precise sense, programs are given a semantics using security systems.

A program for agent i can rely on Boolean tests performed on the local state of the agent. The syntax of Boolean tests is as follows:

$$bool ::= term_1 = term_2 \mid bool_1 \wedge bool_2 \mid \neg bool.$$

As usual, take $bool_1 \vee bool_2$ as an abbreviation for $\neg(\neg bool_1 \wedge \neg bool_2)$. To every test and agent i, associate the set $[\![bool]\!]_i$ of local states of agent i at which the test is true:

$$[\![term_1 = term_2]\!]_i = \{\ell \in Loc_i \mid [\![term_1]\!]_\ell = [\![term_2]\!]_\ell\}$$
$$[\![\neg bool]\!]_i = Loc_i - [\![bool]\!]_i$$
$$[\![bool_1 \wedge bool_2]\!] = [\![bool_1]\!]_i \cap [\![bool_2]\!]_i.$$

A test $bool$ is true in ℓ (for agent i) if $\ell \in [\![bool]\!]_i$.

A *program* for agent i specifies what action the agent should perform next. What do we take as actions? Primitive actions include nil, the null action, $\mathsf{snd}(i, m)$, the sending of message m to agent j, rcv, the reading of a message from the input buffer, and $\mathsf{upd}(var, m)$, the update of variable var to value m. The language in which we write the programs is essentially a language of while loops. It also corresponds to the sequential fragment of a process calculus such as the one described in Section 6.4.2. However, there is no attempt to capitalize on the process-calculus ability to represent the structure of the network as a whole; networking assumptions can be captured by the environment. Mostly, the focus is on communication between agents in as simple a network topology as possible.

A program is a statement, given by the following grammar:

$$\begin{aligned} stmt ::= \ & \mathbf{send}\ term_1\ term_2 \\ | \ & \mathbf{skip} \\ | \ & \mathbf{recv} \\ | \ & var \leftarrow term \\ | \ & stmt_1; stmt_2 \\ | \ & \mathbf{if}\ bool\ \mathbf{then}\ stmt_1\ \mathbf{else}\ stmt_2 \\ | \ & \mathbf{while}\ bool\ \mathbf{do}\ stmt. \end{aligned}$$

Informally, $\mathbf{send}\ term_1\ term_2$ is the sending of the message denoted by $term_2$ to the agent denoted by $term_1$; $\mathbf{skip}$ is the null statement with no effect; $\mathbf{recv}$ awaits for a message to arrive in the agent's buffer; $var \leftarrow term$ updates the variable var to the value denoted by $term$ in the local state of the agent. The remaining constructs are the standard sequencing, conditional, and looping constructs. In Chapter 6, we used the notation

$$\mathbf{recv}\ var$$

which in fact is simply an abbreviation for

$$\begin{aligned} & \mathbf{recv}; \\ & var \leftarrow received(). \end{aligned}$$

The semantics of IMPSEC, given in Figure 7.1, is defined in terms of a transition relation, where the notation $\ell \Vdash_i stmt \xrightarrow{\mathsf{a}} stmt'$ says that $stmt$ rewrites into program $stmt'$, performing action a, at local state ℓ.

An IMPSEC program describes the behaviour of a single agent. To describe the behaviour of a whole system requires at least a program for each agent. Accordingly, define a *joint* IMPSEC *program* to be a tuple $S = (stmt_1, \ldots, stmt_n)$

$$\frac{}{\ell \Vdash_i \mathbf{send}(term_1, term_2) \xrightarrow{\mathsf{snd}([\![term_1]\!]_\ell, [\![term_2]\!]_\ell)} \mathbf{skip}}$$

$$\frac{}{\ell \Vdash_i \mathbf{send}(term_1, term_2) \xrightarrow{\mathsf{nil}} \mathbf{send}(term_1, term_2)}$$

$$\frac{}{\ell \Vdash_i \mathbf{skip} \xrightarrow{\mathsf{nil}} \mathbf{skip}}$$

$$\frac{}{\ell \Vdash_i \mathbf{recv} \xrightarrow{\mathsf{rcv}} \mathbf{skip}} \qquad \frac{}{\ell \Vdash_i \mathbf{recv} \xrightarrow{\mathsf{nil}} \mathbf{recv}}$$

$$\frac{}{\ell \Vdash_i var \leftarrow term \xrightarrow{\mathsf{upd}(var, [\![term]\!]_\ell)} \mathbf{skip}}$$

$$\frac{}{\ell \Vdash_i var \leftarrow term \xrightarrow{\mathsf{nil}} var \leftarrow term}$$

$$\frac{\ell \Vdash_i stmt \xrightarrow{\mathsf{a}} stmt'}{\ell \Vdash_i \mathbf{skip}; stmt \xrightarrow{\mathsf{a}} stmt'} \qquad \frac{\ell \Vdash_i stmt_1 \xrightarrow{\mathsf{a}} stmt'}{\ell \Vdash_i stmt_1; stmt_2 \xrightarrow{\mathsf{a}} stmt'; stmt_2}$$

$$\frac{\ell \in [\![bool]\!]_i \quad \ell \Vdash_i stmt_1 \xrightarrow{\mathsf{a}} stmt_1'}{\ell \Vdash_i \mathbf{if}\ bool\ \mathbf{then}\ stmt_1\ \mathbf{else}\ stmt_2 \xrightarrow{\mathsf{a}} stmt_1'}$$

$$\frac{\ell \notin [\![bool]\!]_i \quad \ell \Vdash_i stmt_2 \xrightarrow{\mathsf{a}} stmt_2'}{\ell \Vdash_i \mathbf{if}\ bool\ \mathbf{then}\ stmt_1\ \mathbf{else}\ stmt_2 \xrightarrow{\mathsf{a}} stmt_2'}$$

$$\frac{\ell \in [\![bool]\!]_i \quad \ell \Vdash_i stmt \xrightarrow{\mathsf{a}} stmt'}{\ell \Vdash_i \mathbf{while}\ bool\ \mathbf{do}\ stmt \xrightarrow{\mathsf{a}} stmt'; \mathbf{while}\ bool\ \mathbf{do}\ stmt}$$

$$\frac{\ell \notin [\![bool]\!]_i}{\ell \Vdash_i \mathbf{while}\ bool\ \mathbf{do}\ stmt \xrightarrow{\mathsf{nil}} \mathbf{skip}}$$

Figure 7.1. Semantics of IMPSEC

of programs, one per agent. Given an initial global state $s = (s_e, s_1, \ldots, s_n)$, define $\mathcal{R}[\![S]\!](s)$ to be the set of runs consistent with S from state s. A run r is *consistent with the joint program S from state s* if $r(0) = s$, and if the sequence of global states in r corresponds to a possible execution of the joint program S, that is, if there exists a sequence of joint programs $S^0, S^1, \ldots$ (with $S^0 = S$) and joint actions $\mathsf{a}^0, \mathsf{a}^1, \ldots$ such that $r_i(t) \Vdash_i S_i^t \xrightarrow{\mathsf{a}_i^t} S_i^{t+1}$ and the

states $r(t) = (s_e, s_1, \ldots, s_n)$ and $r(t+1) = (s'_e, s'_1, \ldots, s'_n)$ satisfy the following constraints, for all agents i:

- if $\mathsf{a}_i^t = \mathsf{upd}(var, m)$, then $s'_i = s_i \cdot update(var, m)$;
- if $\mathsf{a}_i^t = \mathsf{snd}(j, m)$, then $s'_i = s_i \cdot send(j, m)$;
- if $\mathsf{a}_i^t = \mathsf{rcv}$, then either there exists j such that $\mathsf{a}_j^t = \mathsf{snd}(i, m)$ and $s'_i = s_i \cdot recv(m)$, or $\{\mathsf{a} \mid \mathsf{a} \in s_e, \mathsf{a} = \mathsf{snd}(i, m)\} \neq \varnothing$ and $s'_i = s_i \cdot recv(m)$ for some m such that $\mathsf{snd}(i, m) \in s_e$.

Furthermore, the messages that are not delivered are buffered by the environment. Formally, if

$$S = \{\mathsf{snd}(j, m) \mid \exists i.\mathsf{a}_i^t = \mathsf{snd}(j, m)\}$$
$$R = \{\mathsf{snd}(j, m) \mid \exists j.\mathsf{a}_j^t = \mathsf{rcv}, s'_j = s_j \cdot recv(m)\},$$

then $s'_e = (s_e \cup S) - R$.

If Σ is a set of initial states, define $\mathcal{R}[\![S]\!](\Sigma)$ to be the set of all runs r consistent with S from some state s in Σ. Thus, given a joint program S and a set of initial global state Σ, the system $\mathcal{R}[\![S]\!](\Sigma)$ models the protocol represented by the joint program S.

Note that there is a clear relationship between IMPSEC programs and process calculi of the kind we described in Section 6.4.2. Intuitively, we can think of IMPSEC programs as written in a sequential fragment of a process calculus. (While IMPSEC does not have a concurrency operator, it is straightforward to add one.) One difference is that the semantics of IMPSEC is more concrete, in that there is an explicit scheduler in the form of the environment. The semantics above does not take advantage of this flexibility, since the environment is simply used as a buffer to hold messages in transit. Moreover, connectivity assumptions (for instance, network topology) can also be added to the environment, rather than encoded in the process describing the system as a whole. It remains to compare the respective advantages and disadvantages of these approaches to modeling protocols.

7.3 Strand Spaces and Multiagent Systems

The strand-spaces framework is a recent popular framework for the analysis of security protocols. Roughly speaking, the strand space corresponding to a protocol is the set of the traces of the various interactions between the agents under consideration. Strand spaces are meant to capture the "causality" between the various events of a protocol; According to the strand-space theory, an event causes an other event if the presence of the latter implies the presence of the former. Thus, a causality relation in this sense is simply an authenticating relation, of the kind described in

Section 6.3. There are clear similarities between strand spaces and multiagent systems, as introduced above. We examine the relation between these two frameworks more carefully, focussing on strand spaces as a tool for modeling protocols.

The key issue in relating the two frameworks is the handling of agents. Most importantly, an agent has a state that is shared across all the interactions that the agent performs. In multiagent systems, there is a clear notion of an agent participating in an interaction. In strand spaces, there is not. Each protocol interaction (described by a strand) is viewed as independent from all others. In fact, each strand can be viewed as representing a different agent. This approach to modeling agents is deliberate in the definition of strand spaces, and gives a theory that yields general results. Strand spaces do treat agents, in a fashion, by essentially assigning to every strand a name representing the "agent" executing the strand. However, it is still the case that strands corresponding to the same "agent" can exchange values only through explicit communication, i.e. there is no shared state across the strands corresponding to the same "agent" name. For all intents and purposes, these strands may as well be assigned to different actual agents.

To highlight the role of agents, we provide a family of translations from strand spaces to *strand systems*, a subclass of multiagent systems related to security systems that seem to capture the intuition underlying strand spaces. The translations are parameterized by an assignment from strands to agents. This assignment associates with a strand the agent performing the protocol interaction described by the strand. Such an assignment captures the intuition that different strands can potentially be executed by the same agent.

Why is the role of the agents so significant? For the protocols typically considered in the literature it is not. On the other hand, it should be clear that belief and knowledge are useful concepts when reasoning about security protocols. There are a number of ways that an adversary can gain knowledge in a system. Certainly when an adversary intercepts a message, he learns the contents of the message. But he may learn much more if he knows the protocol being run. In addition, different agents representing the same adversary may be able to pool the information they have acquired. In any case, as soon as one talks about belief or knowledge, there must be agents in the picture to which belief or knowledge is ascribed. One advantage of a multiagent system is that it explicitly identifies agents and provides an easy way to ascribe knowledge to agents In the context of security, that means we are forced to reason about, for example, which names represent the same agent or which ones may represent the same agent.

Significantly, the translations in this section are not surjective. Some strand systems are not the image of any strand space, regardless of the assignment of agents to strands. This is not just an artifact of our particular translation. Any translation from strand spaces to strand systems that preserves the message history

of the agents, in a precise sense, cannot be surjective. Intuitively, this is because in a strand space we cannot say "either this sequence of events happens or that one does, but not both". This indicates a fundamental lack of expressiveness in the current formulation of strand spaces.

Let M be the set of possible messages that can be exchanged by the agents in a protocol.[1] A *signed term* is a pair $\langle \sigma, u \rangle$ with $\sigma \in \{+, -\}$ and $u \in M$. A signed term $\langle +, u \rangle$ represents the sending of message u and is typically written $+u$, and a signed term $\langle -, u \rangle$ represents the reception of message u and is typically written $-u$. We write $(\pm M)^*$ for the set of finite sequences of signed terms. A strand space over M signed terms. A *strand space* over M consists of a set Σ, whose elements are called *strands*, together with a trace mapping $\mathrm{tr} : \Sigma \to (\pm M)^*$, associating each strand in Σ with a sequence of signed terms. A strand space is typically represented by the underlying set Σ, leaving the trace mapping implicit.

In a strand space Σ, a *node* is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and an integer i with $1 \leq i \leq |tr(s)|$. The set of nodes of Σ is represented by $\mathcal{N}$. The node $\langle s, i \rangle$ is said to *belong to* the strand s, written $\langle s, i \rangle \in s$ by abuse of notation. Given a node $n = \langle s, i \rangle$, where $\mathrm{tr}(s) = \langle \sigma_1, u_1 \rangle \ldots \langle \sigma_k, u_k \rangle$, define $\mathrm{term}(n) = \langle \sigma_i, u_i \rangle$. If n_1 and n_2 are nodes, the notation $n_1 \to n_2$ indicates that $\mathrm{term}(n_1) = +u$ and $\mathrm{term}(n_2) = -u$; the notation $n_1 \Rightarrow n_2$ indicates that both n_1 and n_2 occur on the same strand s and $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i + 1 \rangle$. Note that the set $\mathcal{N}$ of nodes together with both sets of edges $n_1 \to n_2$ and $n_1 \Rightarrow n_2$ forms a directed graph $(\mathcal{N}, (\to \cup \Rightarrow))$.

A bundle represents a snapshot of a possible protocol execution. For a given strand space Σ, let $\mathcal{C} = (\mathcal{N}_\mathcal{C}, (\to_\mathcal{C} \cup \Rightarrow_\mathcal{C}))$ be a subgraph of $(\mathcal{N}, (\to \cup \Rightarrow))$. The graph $\mathcal{C}$ is a *bundle* if

B1. $\mathcal{C}$ is finite,

B2. if $n_2 \in \mathcal{N}_\mathcal{C}$ and $\mathrm{term}(n_2)$ is negative, then there is a unique n_1 such that $n_1 \to_\mathcal{C} n_2$,

B3. if $n_2 \in \mathcal{N}_\mathcal{C}$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow_\mathcal{C} n_2$,

B4. $\mathcal{C}$ is acyclic.

In B2 and B3, because $\mathcal{C}$ is a graph, it follows that $n_1 \in \mathcal{N}_\mathcal{C}$. A node n is in the bundle $\mathcal{C}$ if it is in $\mathcal{N}_\mathcal{C}$.

It will be useful in this section to allow infinite bundles. An *infinite bundle* is just a subgraph of $(\mathcal{N}, (\to \cup \Rightarrow))$ that satisfies B2–B4 (that is, we no longer require the finiteness condition B1). The *height* of an infinite bundle is the length of the longest finite sequence of nodes $n_1, n_2, n_3, \ldots, n_k$ in $\mathcal{C}$ such that $n_1 \rightsquigarrow n_2 \rightsquigarrow \ldots \rightsquigarrow n_k$, where $\rightsquigarrow$ is either $\to$ or $\Rightarrow$. (A bundle can have infinite height if there

[1] The actual contents of the message and the structure of M are not important for the purpose of this section.

is no bound on the length of the longest sequence of this type.) Of course, all finite bundles have finite height. It is easy, however, to construct infinite bundles of infinite height (even if all individual strands have length at most 2). For example, consider the strand space $\Sigma = \{s_i \mid i \in \mathbb{Z}\}$, with a trace mapping $\mathrm{tr}(s_i) = \langle -u_i, +u_{i+1} \rangle$. The strand space Σ itself in this case is an infinite bundle of infinite height. All the arguments pertaining to strand spaces that are applied to finite bundles go through without change for infinite bundles of finite height. (Indeed, they go through for infinite bundles that are *well-founded*, in the sense of having no infinite "descending" sequences of the form $\ldots \rightsquigarrow n_3 \rightsquigarrow n_2 \rightsquigarrow n_1$, although we end up using only bundles of finite height in our arguments.)

The multiagent systems that will be constructed from the strand spaces representation of a protocol are a class of systems we call *strand systems*, related to the asynchronous message-passing systems of Section 7.1, that provide the foundation for security systems. Due to the assumptions made by the strand-space approach, namely that events in strands consist of sending and receiving messages, we consider only systems where the local state of an agent is the sequences of messages that the agent has sent and received. Thus, we deliberately ignore internal actions such as variable updates (or, more accurately, treat them as irrelevant). There are other minor differences. For instance, messages do not specify a receiver, so that send events are of the form $send(u)$, instead of $send(a, u)$, for an agent a. Strand systems also allow for an infinite number of agents, whereas in the systems we describe above, there are only finitely many agents. (In this section, we use a for agent names rather than i, to emphasize this fact.) The definitions earlier in the chapter generalize to infinitely many agents in a straightforward way. Moreover, agents are allowed in security systems to have a nontrivial initial state, while for strand systems, the initial state is always the empty sequence.

7.3.1 Translating Strand Spaces to Systems

We now turn to the problem of translating strand spaces into systems. This is done by formalizing the strand space intuition that bundles represent snapshots of possible executions. Our construction derives the possible execution traces in terms of sequences of bundles, which are then used to construct the runs of the system.

A multiagent system requires an explicit set of agents; a strand space does not. To perform the translation, specify a set $\mathcal{A}$ of agents and a particular *agent assignment* $A : \Sigma \to \mathcal{A}$, which intuitively associates with each strand $s \in \Sigma$ the agent $A(s)$ executing s. In the generated strand system, an agent behaves as if it were concurrently executing the various strands assigned to it. The motivation behind this approach is that if the same agent is in reality executing many strands, then it should share its knowledge across all the strands it is executing.

The choice of agents and the agent assignment for a given strand space is left to the model designer. Different choices lead to different multiagent systems. As we show at the end of this section, associating a different agent with each strand enforces the basic strand space tenet that information is exchanged only through explicit messages, that is, there is no shared state between different strands.

The translation takes as arguments a strand space Σ, a set $\mathcal{A}$ of agents, and an agent assignment A from strands in Σ to agents. To define the translation, first define a relation on bundles that represents the actions that the agents in the strand space can perform. Given a strand $s \in \Sigma$ and a bundle $\mathcal{C}$, let B-height(s) be the largest i such that $\langle s, i\rangle \in \mathcal{N}_{\mathcal{C}}$. (We take B-height$(s) = 0$ if no node in s appears in $\mathcal{C}$.)[2] A function $f : \Sigma \to \Sigma$ *respects* A if $A(s) = A(f(s))$, that is, the same agent is associated with both strands s and $f(s)$ for all strands $s \in \Sigma$. If B_1, B_2 are (possibly infinite) bundles of Σ, and $f : \Sigma \to \Sigma$ is a bijection that respects A, we write $B_1 \sqsubseteq_f B_2$ if the following two conditions hold:

(1) if $\langle s, i\rangle$ is in B_1, then $\langle f(s), i\rangle$ is in B_2 and term$(\langle s, i\rangle) =$ term$(\langle f(s), i\rangle)$,
(2) if $\langle s, i\rangle \to \langle s', j\rangle$ is an edge in B_1, then $\langle f(s), i\rangle \to \langle f(s'), j\rangle$ is an edge in B_2.

These clauses guarantee that the prefix of s that is in B_1 is a prefix of the prefix of $f(s)$ that is in B_2. For example, if B_1 consists of the single node $\langle s, 1\rangle$ and B_2 consists of $\langle s', 1\rangle$ and $\langle s', 2\rangle$, where term$(\langle s, 1\rangle) =$ term$(\langle s', 1\rangle)$, then $B_1 \sqsubseteq_f B_2$, where f is the bijection that permutes s and s', while acting as the identity on all other strands.

For many cases of interest, we can simply take the bijection f to be the identity; in that case, $B_1 \sqsubseteq_f B_2$ if and only if B_1 is a subgraph of B_2. We discuss the reason for allowing arbitrary bijections and the role of the bijection at the end of this section.

We write $B_1 \mapsto B_2$ if there is a bijection $f : \Sigma \to \Sigma$ that respects A such that

(1) $B_1 \sqsubseteq_f B_2$, and
(2) $\sum_{s \in A^{-1}(a)} B_2$-height$(f(s)) - B_1$-height$(s) \leq 1$ for all agents $a \in \mathcal{A}$.

Informally, $B_1 \mapsto B_2$ if, for each agent $a \in \mathcal{A}$, B_2 extends the prefix of at most one strand in B_1 corresponding to a, and extends it by at most one node. (Note that the strand $f(s)$ in B_2 extending the prefix of strand s in B_1 may be different from s, depending on the definition of f.) If B_2 does extend the prefix of one of the strands in B_1 corresponding to agent a by one node, let $e_{a,B_1 \mapsto B_2}$ denote the event corresponding to that node: if the node is n and term$(n) = +u$, then $e_{a,B_1 \mapsto B_2}$ is $send(u)$, and if term$(n) = -u$, then $e_{a,B_1 \mapsto B_2}$ is $recv(u)$. Define a $\mapsto$-*chain* (or

[2] This notion of height of a strand in a bundle should not be confused with the notion of height of a bundle we defined earlier.

simply a chain) to be an infinite sequence of bundles $B_0, B_1, \ldots$ such that B_0 is the empty bundle and $B_0 \mapsto B_1 \mapsto \ldots$.

Let $Chains(\Sigma, \mathcal{A}, A)$ be the set of all chains in Σ. Associate with every chain in $Chains(\Sigma, \mathcal{A}, A)$ a run as follows: Given a chain $C = B_0 \mapsto B_1 \mapsto \ldots$ and an agent $a \in \mathcal{A}$, define $hist_a^t(C)$ inductively. Let $hist_a^0(C) = \langle\,\rangle$; let $hist_a^{n+1}(C) = hist_a^n(C)$ if no strand corresponding to agent a in B_n is extended in B_{n+1}; otherwise, let $hist_a^{n+1}(C) = hist_a^n(C) \cdot e_{a,B_n \mapsto B_{n+1}}$. (Informally, $hist_a^{n+1}(C)$ is the result of appending to $hist_a^n(C)$ the unique event performed by agent a in going from B_n to B_{n+1}.) Thus, $hist_a^n(C)$ consists of all the events that a has performed in B_n. Let r^C be the run such that $r_a^C(t) = hist_a^t(C)$ and let $\mathcal{R}(\Sigma, \mathcal{A}, A) = \{r^C \mid C \in Chains(\Sigma, \mathcal{A}, A)\}$.

Theorem 7.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system.*

In light of Theorem 7.1, define the map T_A from strand spaces to strand systems by taking $T_A(\Sigma) = \mathcal{R}(\Sigma, \mathcal{A}, A)$.

As we mentioned at the beginning of this section, strand spaces as originally described can be modeled by taking the set of agents of a strand space Σ to be Σ, and taking the identity function id as the agent assignment. This captures explicitly the intuition that strands are independent protocol executions, that for all intents and purposes may be assumed to be executed by different agents. This is the case since there is no state shared between strands, and every communication is made explicit. In other words, there is no conceptual difference between two strands s_1 and s_2 executed by different processes of an agent or by two distinct agents if there cannot be any shared state between s_1 and s_2.

There is a small amount of information that is lost in the translation from strand spaces to strand systems, which will become evident in Theorem 7.2 below. This loss stems from the fact that messages in strand systems are completely anonymous. For example, if agent 2 and agent 3 both send a message u and later agent 1 receives it, there is no way in a strand system to tell if agent 1 received u from agent 2 or agent 3. By way of contrast, in a strand space, there is an edge indicating who agent 1 received the message from. The multiagent system framework can in fact keep track of who an agent received a message from by adding an additional component to the global state; this is the state of the *environment*, which intuitively describes everything relevant to the system not included in the local states of the processes.[3] We will not bother going into the details of the environment in this section, as the issue does not affect our results. We can characterize the information loss resulting from our translation by defining a relation between globals states of

[3] In this particular case, the environment could record the sender of each message that is received at any given round.

$\mathcal{R}(\Sigma, \Sigma, id)$ and bundles of Σ. A global state $\langle \sigma_s \mid s \in \Sigma \rangle$ (recall that here $\mathcal{A} = \Sigma$) is *message-equivalent* to a bundle B if for each $s \in \Sigma$, if $\sigma_s = \langle e_1, \ldots, e_k \rangle$ then B-height$(s) = k$ and, for each i such that $1 \leq i \leq k$, if term$(\langle s, i \rangle) = +u$ then e_i is $send(u)$, and if term$(\langle s, i \rangle) = -u$ then e_i is $recv(u)$. Intuitively, a global state is message-equivalent to any bundle that has the same nodes. This captures the intuition that an agent receiving a message is not aware of the sender. The following theorem shows that, except for this loss of information, our translation from strand spaces to strand systems essentially identifies bundles and global states (if all strands are treated as being associated with a different agent).

Theorem 7.2. *Every global state of $\mathcal{R}(\Sigma, \Sigma, id)$ is message-equivalent to a bundle of Σ of finite height, and every bundle of Σ of finite height is message-equivalent to a global state of $\mathcal{R}(\Sigma, \Sigma, id)$.*

If the environment state is used to record the sender of each received message, Theorem 7.2 can be strengthened to a 1-1 correspondence between global states of $\mathcal{R}(\Sigma, \Sigma, id)$ and bundles of Σ of finite height.

With these results in hand, we now discuss some of the choices made, in particular, why we allowed infinitely many agents, infinite bundles, and an arbitrary bijection f in the definition of $\mapsto$. It turns out that these choices are somewhat related. First observe that, in Theorem 7.2, each strand was identified with an agent. Thus, if there are infinitely many strands in the strand space, the corresponding strand system requires infinitely many agents. Naturally, if we restrict our analysis to strand spaces with only finitely many strands, then we can take the corresponding strand systems to have only finitely many agents. Infinite bundles are needed in order to prove Theorem 7.1 when there are infinitely many agents. To understand why, consider a strand space Σ, where $\Sigma = \{s_1, s_2, \ldots\}$ and tr$(s_n) = \langle +u_n \rangle$. In other words, strand s_n has exactly one node, at which a send action is performed. If a different agent is associated with each strand, then in the corresponding strand system, the set of histories for agent n will consist of the empty history and the history $\langle send(u_n) \rangle$. The system based on this set of histories has a run where all the agents send their message simultaneously at round 1. This history corresponds to the infinite bundle consisting of all the strands in Σ. Intuitively, if all the agents can send a message, there is no reason that they should not all send it in the first round.

Why do strand spaces allow infinitely many strands? Often, security protocols rely on *nonces*, which are values guaranteed to be unique within a run of the system. Strand spaces model nonces by specifying a different strand for each possible value of a nonce. Since, theoretically, there can be infinitely many nonces (as a consequence of uniqueness), it is necessary to consider infinitely many strands for

a given protocol. Note that these strands do not necessarily represent computations of *different* agents. Indeed, it probably makes sense to consider them all as being performed by the same agent (but at most one of them being performed in a given execution of the protocol).

The bijection f in $\sqsubseteq_f$ is not needed if a different agent is associated with each strand. (That is, in this case it suffices to take f to be the identity.) Similarly, f is not needed if there is a bound k on the length of all strands in Σ. Indeed, it is needed only to take care of the possibility that there is an infinite sequence of strands, each intuitively a prefix of the next, and all associated with the same agent. For example, consider the strand space Σ where, again, $\Sigma = \{s_1, s_2, \ldots\}$ but now $\mathrm{tr}(s_n) = \langle +u_1, \ldots, +u_n \rangle$. Intuitively, in this strand space, s_n is a substrand of s_{n+1} (although, formally, there is no notion of substrand in strand spaces). Suppose that the mapping is such that $\mathcal{A}$ consists of one agent a_1 and A associates all the strands in Σ with a_1. If such a map f (or, equivalently, required f to be the identity) were not allowed, then the only chains would be those of the form $B_0 \mapsto B_1 \mapsto \ldots \mapsto B_k \mapsto B_k \mapsto B_k \mapsto \ldots$ (for some finite k), where, for some strand s, each B_i is a prefix of s. Applying the mapping to this collection of strands gives a single set of histories

$$V_{a_1} = \{\langle send(u_1)\rangle, \langle send(u_1), send(u_2)\rangle, \langle send(u_1), send(u_2), send(u_3)\rangle, \ldots\}$$

in the resulting system, where each history in V_{a_1} is finite. However, the system generated by this set of histories contains an infinite run, which sends message u_i at time i. Unfortunately, there is no chain corresponding to this run. On the other hand, once nontrivial bijections f are allowed, there is no problem. Abusing notation somewhat, there is a chain of the form $s_1 \mapsto s_2 \mapsto s_3 \mapsto \ldots$ where a_1's history is unbounded, since $s_k \sqsubseteq_{f_k} s_{k+1}$, where f_k permutes s_k and s_{k+1} and is the identity on all other strands.

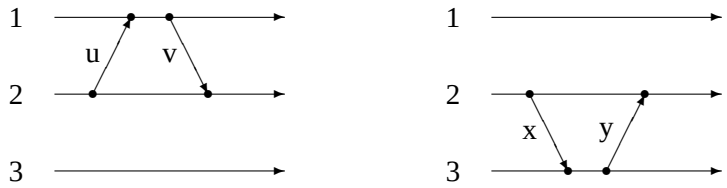
Intuitively, if f must be the identity, then every chain must "choose" the strand it is executing, which implicitly corresponds to choosing how many messages to send in that particular run. Providing a function f that permits "jumping" to strands with the same prefix between any consecutive bundles of a chain essentially models an agent that does not choose the length of the strand up front, but rather just performs the actions (and thus, if one strand is a prefix of another, it cannot tell which of the two strands it is performing).

While it is important to recognize these subtleties, they do not arise in most protocols. For instance, strands for specific protocols will typically be of bounded length, and therefore the bijection f is not needed to define chains in the corresponding strand space.

7.3.2 Translating Systems to Strand Spaces

What about the other direction, that is, the the translation of strand systems into strand spaces. Specifically, given a strand system $\mathcal{R}$, is there a strand space which maps to $\mathcal{R}$ under a suitable agent assignment? In general, there is not. This result is not an artifact of our translation, but reflects a fundamental difference between strand spaces and strand systems. In particular, it does not depend on any of the subtleties that were pointed out at the end of last section.

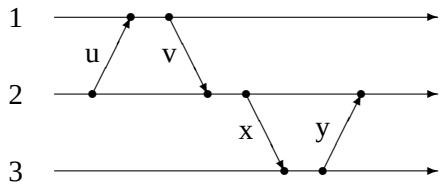
To understand the difficulties, consider the following simple system $\mathcal{R}_1$. It essentially contains two runs r_1 and r_2, with distinct messages x, y, u, v:

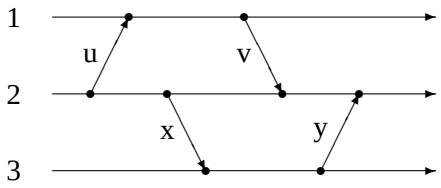


Because the MP1–3 assumptions on strand systems allow arbitrary delays between the events, there are more than two runs in the system; the essential fact is that, in any given run, agent 2 communicates only with agent 1 or only with agent 3. Formally, $\mathcal{R}_1$ is the strand system generated by taking:

$$V_1 = \{\langle\rangle, \langle recv(u)\rangle, \langle recv(u), send(v)\rangle\}$$
$$V_2 = \{\langle\rangle, \langle send(u)\rangle, \langle send(x)\rangle, \langle send(u), recv(v)\rangle, \langle send(x), recv(y)\rangle\}$$
$$V_3 = \{\langle\rangle, \langle recv(x)\rangle, \langle recv(x), send(y)\rangle\}.$$

Under the mapping presented in the previous section, there does not exist a strand space that maps to this system, for any agent assignment. Intuitively, any strand space modeling the system $\mathcal{R}_1$ will need at least strands corresponding to runs r_1 and strands corresponding to runs r_2. Since these sets of strands do not interact (that is, they do not exchange any message), the translation of Section 7.3.1 will produce a system that contains runs that amount to all possible interleaving of the strands corresponding to r_1 and r_2. This results in a system that is strictly larger than $\mathcal{R}_1$. For example, it must contain runs with the following histories for agents 1, 2, and 3:





Roughly speaking, what is happening in the strand system is that agent 2 nondeterministically decides whether to send message u to agent 1 or message x to agent 3. In any run of the system, it sends one or the other, but not both. The problem here is that, in the strand-space framework, we cannot say "one or the other, but not both".

To make this precise, given an agent assignment A, define a translation T from strand spaces to strand systems to be *A-history preserving* if, given a strand space Σ,

- for each agent $a \in \mathcal{A}$, run $r \in T(\Sigma)$, and time t, there exists a bundle $\mathcal{C}$ in Σ such that the events in agent a's history $r_a(t)$ are exactly those that appear in nodes $\langle s, i\rangle$ in $\mathcal{C}$ such that $A(s) = a$;
- conversely, for each agent $a \in \mathcal{A}$ and bundle $\mathcal{C}$ of finite height in Σ, there exists a run $r \in T(\Sigma)$ and time t such that the events in agent a's history $r_a(t)$ are exactly those that appear in nodes $\langle s, i\rangle$ in $\mathcal{C}$ such that $A(s) = a$.

Notice that the translation T_A defined in the previous section is A-history preserving.

Theorem 7.3. *There is no agent assignment A and A-history preserving translation T from strand spaces to strand systems such that the strand system $\mathcal{R}_1$ is in the image of T.*

The example above suggests that in general, systems arising from an agent running a nondeterministic protocol may not be the image of a strand space under the particular translation used. The problem in fact is more profound. Even if the agents are running deterministic protocols, the nondeterminism inherent in the delay of messages delivery may prevent a system from being the image of a strand space. Consider the following system with two agents. Agent 1 sends a message u to agent 2. If agent 2 hasn't received it yet, and hasn't sent a $nack$ message yet, she sends a $nack$. When she gets message u, she sends an ack. Here, the strand space intuitively corresponding to this situation will include a strand for agent 1 where he sends u. For agent 2, we can consider at least the following two strands, $\langle -u, +ack\rangle$ and $\langle +nack, -u, +ack\rangle$. One can check that there exists a chain leading to the bundle made up of the following strand prefixes: $\langle +u\rangle$, $\langle -u, +ack\rangle$, and $\langle +nack\rangle$, leading, through our translation, to a possible history for agent agent of the form $\langle recv(u), send(ack), send(nack)\rangle$, which does not arise in the original

system. In this example, the problem does not occur because the agent makes a choice, but, intuitively, because the "environment" is making a choice when delivering messages.

7.3.3 Discussion

In this section, we have investigated the relationship between strand spaces and multi-agent systems. Our results show that strand spaces are strictly less expressive than strand systems, a subclass of multiagent systems that seems to capture the assumptions underlying strand spaces, in two quite distinct respects. The first is that strand spaces cannot express choice, the fact that exactly one of two possible behaviours is chosen. The second is that strand spaces have no notion of agents.

How serious are these two issues? That depends, of course, on what we are trying to prove. Consider first the inability of strand spaces to express choice. Typical properties properties proved using strand spaces have the form "for all bundles in the strand space, X happens". One way to interpret Theorem 7.3 is that when a strand space is used to model a system, some of the bundles may not correspond to situations that actually arise in the system—those bundles can be seen as "impossible" bundles. This is not a problem, of course, if the property of interest in fact holds in the larger system. However, this may not always be the case. For example, we may well want to prove that a property like "agent 2 sends at most one message" holds in all executions of a protocol. If the protocol also has the property that agent 2 can send messages to either 1 or 3 (as is the case in the protocol described by the system $\mathcal{R}_1$ in Section 7.3.2), then the fact that agent 2 sends at most one message in every execution of the protocol will simply not be provable in the strand-space framework.

The runs of a strand system can be viewed as a linearization of bundles, that is, an explicit ordering of the actions performed by agents in different bundles. This suggests that results about strands can be imported to runs. The results in this section point to subtleties in doing this. More precisely, while results about strands can be imported to results about runs (the runs that arise from translating the strand space to a system), the converse may not be true, depending on the expressiveness of the language.

Turning to the issue of agents, the strand-space framework assumes that messages relayed between strands form the only means of exchanging information between strands. In other words, there is no shared state between strands. Therefore, for all intents and purposes, we can imagine that every strand is executed by a different agent. On the other hand, if the same agent is executing two strands then, intuitively, it should know whatever is happening on both strands, without requiring communication between them. Furthermore, as soon as one wants to analyze

the properties of strand spaces using belief and knowledge, agents to which the knowledge can be ascribed are needed. But even without bringing in knowledge, we need to be careful in interpreting security results proved under the assumption that different agents perform different strands. Clearly this assumption is not, in general, true. Ideally, security protocols should be proved correct under any "reasonable" assignment of agents to roles in the security protocol. At the very least it should be clear under which assignments the result holds. For instance, it is known that methods for the analysis of cryptographic protocols that fail to handle multiple roles for the same agent do not yield dependable results, as they may not reveal *multi-role flaws*. Multi-role flaws commonly arise when a cryptographic protocol logic implicitly assumes that if an agent a takes on a role A in some session, then he will not also take on another role B in some different session. This assumption is often a consequence of the identification of the notions of role and agent. It is possible to show that reasonable protocols that can be proved correct under the assumption that an agent takes on the same role in all sessions are flawed if this assumption is dropped.

The set of runs in the system and the agent assignment are particularly significant when considering specifications that are not *run-based*. A run-based specification is checked on a per-run basis. For example, "agent 2 sends at most 1 message" is a run-based specification: given a run, one can check whether the property holds for that run. A run-based specification holds for a set of runs if it holds for all runs in the set. In contrast, a *knowledge-based specification* such as "after running the protocol, agent 2 knows X" cannot be checked on a per-run basis, as it relies on the set of runs *as a whole* to verify the property. It holds if, in all runs in the system that agent 2 considers possible after running the protocol, X holds. Clearly it does not suffice to look at an individual run to determine whether such a property holds. Similarly, probabilistic specifications like "X holds in at most 3% of the runs" also depend on the whole system and cannot be checked simply by examining individual runs.

Typical specifications in the security literature are safety properties, which are often paraphrased as "bad things don't happen", and hence are run-based. Run-based specifications have the property that if they hold in a system, they hold in any subset of the runs of the system. It is "safe" to prove that a run-based specification holds of a strand space which translates to a superset of the intended system. Proving that the property holds for "impossible" runs does not hurt. This is not the case for properties that are not run-based. We believe that knowledge-based specifications, as well as probabilistic ones, will play a significant role in the design and analysis of security protocols. Fairness is a good example. A protocol is *fair* if intuitively no protocol participant can gain an advantage over other participants by misbehaving. In the context of fair exchange protocols where two agents exchange

one item for another, fairness ensures that either each agent receives the item it expects, or neither receives any information about the other's item. This notion of "not receiving any information" can be interpreted as meaning that no knowledge is gained. The results in this section suggest that strand spaces, as currently defined, will have difficulty handling such specifications.

Notes

The work in Section 7.3 first appeared in [Halpern and Pucella 2003b].

Multiagent systems are a popular model of protocols from the distributed computing literature. Our presentation in Section 7.1 is based on that of Fagin, Halpern, Moses, and Vardi [1995, Chapter 4]. Arguments for capturing asynchrony by considering all runs consistent with the constraints MP1–3 are given by Fagin, Halpern, Moses, and Vardi [1995, Section 4.4.6]. Variants of multiagent systems have been previously considered in the security literature [Halpern, Moses, and Tuttle 1988; Bieber 1990; Abadi and Tuttle 1991; Syverson and Oorschot 1994; Stubblebine and Wright 1996]. There are minor differences between the asynchronous message-passing systems described in this chapter and those from Fagin, Halpern, Moses, and Vardi [1995]. The main one is that messages in this chapter are essentially anonymous. A message does not specify a sender or an actual receiver. Messages in the asynchronous message-passing systems of [Fagin, Halpern, Moses, and Vardi 1995], on the other hand, are usually not anonymous: the events are of the form $send(u, a, b)$ (u is send from a to b) and $recv(u, a, b)$ (u is received by b from a).

The generation of multiagent systems from programs is adapted from the general approach of Fagin, Halpern, Moses, and Vardi [1995, Chapter 5]. They define the notion of a protocol, which is just a function from local states to actions, and show how protocols in that sense give rise to multiagent systems by being executing in a context, which includes a protocol for the environment. They introduce a programming language based on state machines, and show how the semantics of such a language is a protocol, that can then be used to generate a system. They mention that it is possible to extend the language with higher-level features, but that it is not necessary for the purpose of expressiveness. Since we are not aiming for generality here, we dispensed with the general notion of a context and environment protocol, essentially hardwiring it in into the semantics $\mathcal{R}[\![S]\!]$. Moreover, we also dispensed with the intermediate notion of protocol, and gave a semantics directly to programs. Of course, it is possible to express the work in Section 7.2 using the framework of Fagin, Halpern, Moses, and Vardi [1995], but doing so might lose some of the intuition of the underlying semantics, since it would amount to

compiling IMPSEC into a lower-level language. The semantics of IMPSEC is essentially a form of small-step operational semantics [Winskel 1993], in the tradition of Plotkin's [1981] structural operational semantics.

Strand spaces were introduced by Thayer, Herzog, and Guttman [1999b]. Athena [Song 1999] performs automated checking of security protocols expressed in the strand-space framework using a mixture of model checking and theorem proving.

It is possible to augment strand spaces in such a way that restores the expressiveness. One way to do this is by adding a notion of conflicting strands that essentially say that either one of two strands can appear in a bundle, but not both. See Halpern and Pucella [2003b] for details. Independently, Crazzolara and Winskel [2001] reached the same conclusions, and came up with a similar fix.

Snekkenes [1992] studies multi-role flaws in the context of various cryptographic protocol logics. Recent work on analyzing mixed protocols using strand spaces [Thayer, Herzog, and Guttman 1999a] shows that strand spaces can be extended to deal with what essentially amount to multi-role flaws. However, the approach often requires *phantom messages* (messages that are not actually exchanged during runs of the protocols) to carry state information between the different protocol strands corresponding to the same agent. Logics for reasoning explicitly about names of agents, the kind of notions that arise in multi-role flaws, have been described by Halpern and Grove [1993] and Grove [1995].

Some of the topics explored in Section 7.3 appear in various forms in other work. For example, Cervesato et al. [2000] define a notion of *parametric strand*, essentially a strand where messages may contain variables. Parametric strands correspond to roles, which are implicit in the original work on strand spaces. The work of Cervesato et al. also deals with the evolution of the system described by a strand space; they define a one-step transition between bundles. The transition is reminiscent of the one we describe in Section 7.3.1, but is restricted to extending a single strand at a time. (They also allow actions specific to their formalization, such as the instantiation of a strand from a parametric strand.) Parametric strands also appear as trace-types in Athena, a model-checker based on the strand space approach [Song 1999].

The formal definition of a safety property is due to Alpern and Schneider [1985]. The distinction between run-based specification and knowledge-based specification is clarified by Halpern [2000].

Fair exchange protocols are introduced and described, for instance, by Ben-Or, Goldreich, Micali, and Rivest [1990], Asokan, Shoup, and Waidner [1998], and Shmatikov and Mitchell [2000].

8
A Logic for Reasoning about Security Protocols

ARGUABLY, the problem of verifying that a protocol satisfies particular security properties (for instance, confidentiality) has received the most attention in the recent literature on security protocol analysis. However, the task of specifying security properties themselves is far from having received a satisfactory solution. Many formal methods for the analysis of security protocols rely on specialized logics to rigorously state and prove properties of the protocols they study. Here, we take a very general view of logic, to encompass formal methods where the specification language is implicit, or where the properties to be checked are fixed. Those logics provide constructs for expressing the basic notions involved in security protocols, such as secrecy, recency, and message composition, as well as providing means (either implicitly or explicitly) for describing the evolution of the knowledge or belief of the agents as the protocol progresses. Indeed, informal specifications of security in the literature are typically phrased in terms of knowledge. It thus seems natural to use a specification language where specifications are written directly in terms of knowledge. Knowledge specifications also have other advantages. They tend to be more abstract, since they need not specify exactly how the agents obtain knowledge. For instance, it is easy to write a property that says that an adversary never knows a key, instead of writing a property saying that the adversary never receives the key unencrypted, or never finds the key sitting in a database, and so on.

There is of course another aspect to security protocols analysis. Every logic for reasoning about security protocols aims at proving security in the presence of malicious adversaries. To analyze the effect of adversaries, a security logic specifies (again, either implicitly or explicitly) an *adversary model*, that is, a description of the capabilities of adversaries. Almost all existing logics are based the Dolev-Yao adversary model already described in Section 6.2. Recall that a Dolev-Yao adversary can compose messages, replay them, or decipher them if he knows the right keys, but cannot otherwise "crack" encrypted messages.

The Dolev-Yao adversary is a useful abstraction, in that it allows reasoning about protocols without worrying about the actual encryption scheme being used. It also has the advantage of being restricted enough that interesting theorems can be proved with respect to security. However, in many ways, the Dolev-Yao model is too restrictive. For example, it does not consider the information an adversary may infer from properties of messages and knowledge about the protocol that is being used. Recall the Duck-Duck-Goose protocol of Section 6.2: an agent has an n-bit key and, according to her protocol, sends the bits that make up her key one by one. Of course, after intercepting these messages, an adversary will know the key. However, there is no way for security logics based on a Dolev-Yao adversary to argue that, at this point, the adversary knows the key. Another limitation of the Dolev-Yao adversary is that it does not easily capture probabilistic arguments. After all, the adversary can always be lucky and just *guess* the appropriate key to use, irrespective of the strength of the encryption scheme.

The importance of being able to reason about adversaries with capabilities beyond those of a Dolev-Yao adversary is made clear when looking at the sometimes subtle interactions between the cryptographic protocol and the encryption scheme. It is known that various protocols that appear secure under a symbolic encryption scheme can be insecure when implemented using encryption schemes with specific properties. A more refined logic for reasoning about security protocols will have to be able to handle adversaries more general than the Dolev-Yao adversary. Because they effectively build in the adversary model, existing formal methods for analyzing protocols are not able to reason directly about the effect of running a protocol against adversaries with properties other than those built in.

In this chapter, we describe a logic for reasoning about security protocols that allows an explicit and natural modeling of adversaries. The idea, which should be unsurprising at this point, is to model the adversary in terms of what the adversary knows. This approach has some significant advantages. Logics of knowledge have been shown to provide powerful methods for reasoning about trace-based executions of protocols. They can be given semantics that is tied directly to protocol execution, using the models introduced in the last chapter, thus avoiding problems of having to analyze an idealized form of the protocol. A straightforward application of logics of knowledge leads to the conclusion that in the Duck-Duck-Goose protocol, the adversary knows the key. Logics of knowledge can also be extended with probabilities so as to be able to deal with probabilistic phenomena. Unfortunately, as we saw in Chapter 2, traditional logics of knowledge suffer from a well-known problem known as the *logical omniscience* problem: an agent knows all tautologies and all the logical consequences of her knowledge. The reasoning that allows an agent to infer properties of the protocol also allows an adversary to

deduce properties that cannot be computed by realistic adversaries in any reasonable amount of time.

This is exactly the motivation for explicit knowledge, and the particular approach described in Chapter 2, algorithmic knowledge: assume that agents (including adversaries) have "knowledge algorithms" that they use to compute what they know. The capabilities of the adversary can be captured by his algorithm. Hence, Dolev-Yao capabilities can be provided by using a knowledge algorithm that can only compose messages or attempt to decipher them using known keys. By changing the algorithm, the capabilities of the adversary can be extended, so that he can attempt to crack encrypted messages by factoring (in the case of RSA), using cryptanalysis, or just by guessing keys. Moreover, the algorithmic knowledge framework can also handle the case of an agent sending the bits of his key, by providing the adversary's algorithm with a way to check whether this is indeed what is happening. By explicitly using algorithms, it is possible to analyze the effect of bounding the resources of the adversary, and thus make progress toward bridging the gap between the symbolic analysis of cryptographic protocols and more computational accounts of cryptography. (See Chapter 10.) Note that both traditional knowledge and algorithmic knowledge are necessary in the analysis. Traditional knowledge is used to model an agent's beliefs about what can happen in the protocol; algorithmic knowledge is used to model the adversary's computational limitations (for example, the fact that he cannot factor).

The focus of this work is on developing a general and expressive framework for modeling and reasoning about security protocols, in which a wide class of adversaries can be represented naturally. Therefore, we emphasize the expressiveness and representability aspects of the framework, rather than studying the kind of security properties that are useful in such a setting, or developing techniques for proving that properties hold in the framework. These are all relevant questions that need to be pursued once the framework proves useful as a specification language. We will return to this point in Chapter 10.

8.1 The Logic

The goal is to be able to reason about properties of security systems as defined in Section 7.1, including properties involving the knowledge of agents in the system. To formalize this type of reasoning, we first need a language. Take as a base language the logic $\mathcal{L}_n^{\mathrm{KX}}$ of Chapter 2. Recall that starting with a set Φ_0 of primitive propositions, which we can think of as describing basic facts about the system, such as "the key is k" or "agent A sent the message m to B", formulas of $\mathcal{L}_n^{\mathrm{KX}}(\Phi_0)$

are formed by closing off under negation, conjunction, and the modal operators $K_1, \ldots, K_n$ and $X_1, \ldots, X_n$.

Recall that $\mathcal{L}_n^{\mathrm{KX}}$ is given a semantics in terms of algorithmic knowledge structures. A system can be viewed as an algorithmic knowledge structure, once we add a function π telling us how to assign truth values to the primitive propositions, and add knowledge algorithms for the agents. An *interpreted algorithmic knowledge system* $\mathcal{I}$ is a tuple $(\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$, where $\mathcal{R}$ is a system, π is an interpretation for the propositions in Φ_0, and $\mathtt{A}_i$ is the knowledge algorithm of agent i. The interpretation π assigns truth values to the primitive propositions at the global states. Thus, for every $p \in \Phi_0$ and global state s that arises in $\mathcal{R}$, we have $\pi(s)(p) \in \{\textbf{true}, \textbf{false}\}$. Of course, π also induces an interpretation over the points of $\mathcal{R}$; simply take $\pi(r,t)$ to be $\pi(r(t))$. We refer to the points of the system $\mathcal{R}$ as points of the interpreted algorithmic knowledge system $\mathcal{I}$. The knowledge algorithms are used to compute the explicit knowledge of the agents. In local state ℓ, the agent computes whether he knows φ by applying his knowledge algorithm to input (φ, ℓ).

The interpreted algorithmic knowledge system $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ can be made into an algorithmic knowledge structure by taking the possible worlds to be the points of $\mathcal{R}$, and by defining $\mathcal{V}_i$ so that $\mathcal{V}_i(r,t) = r_i(t)$. Thus, agent i considers a point (r', t') possible at a point (r, t) if i has the same local state at both points; thus, the agents' knowledge is completely determined by their local states. Define $(r,t) \sim_i (r', t')$ if and only if $r_i(t) = r'_i(t')$.

Define what it means for a formula φ to be true (or satisfied) at a point (r,t) in an interpreted algorithmic knowledge system $\mathcal{I}$, written $(\mathcal{I}, r, t) \models \varphi$, inductively as follows:

$(\mathcal{I}, r, t) \models p$ if $\pi(r,t)(p) = \textbf{true}$
$(\mathcal{I}, r, t) \models \neg\varphi$ if $(\mathcal{I}, r, t) \not\models \varphi$
$(\mathcal{I}, r, t) \models \varphi \wedge \psi$ if $(\mathcal{I}, r, t) \models \varphi$ and $(\mathcal{I}, r, t) \models \psi$
$(\mathcal{I}, r, t) \models K_i\varphi$ if $(\mathcal{I}, r', t') \models \varphi$ for all (r', t') such that $r_i(t) = r'_i(t')$
$(\mathcal{I}, r, t) \models X_i\varphi$ if $\mathtt{A}_i(\varphi, r_i(t)) =$ "Yes".

As before, the first clause shows how to use the interpretation π to define the semantics of the primitive propositions. The next two clauses, which define the semantics of $\neg$ and $\wedge$, are the standard clauses from propositional logic. The fourth clause is designed to capture the intuition that agent i knows φ exactly if φ is true in all the worlds that i thinks are possible. The last clause captures the fact that explicit knowledge is determined using the knowledge algorithm of the agent.

To reason about security protocols, consider a specific set of primitive propositions $\Phi_0^{\mathrm{S}} \subseteq \Phi_0$:

– $send_i(m)$: agent i sent message m
– $recv_i(m)$: agent i received message m
– $has_i(m)$: agent i has message m.

Intuitively, $send_i(m)$ is true when agent i has sent a message containing m at some point, $recv_i(m)$ is true when agent i has received message m at some point, and $has_i(m_1)$ is true if agent i has received a message m_2 such that $m_1 \sqsubseteq m_2$. Note that the has_i predicate is not restricted by issues of encryption: since $m \sqsubseteq \{m\}_k$, the has_i predicate characterizes the messages that agent i has implicitly in his possession, given the messages that he has received.

An *interpreted algorithmic knowledge security system* is simply an interpreted algorithmic knowledge system $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ where $\mathcal{R}$ is a security system, and π is an *acceptable* interpretation, that is, it gives the following fixed interpretation for the primitive propositions in Φ_0^{S}:

– $\pi(r,t)(send_i(m)) =$ **true** if and only if there exists j and m' such that $m \sqsubseteq m'$ and $send(j, m') \in r_i(t)$
– $\pi(r,t)(recv_i(m)) =$ **true** if and only if $recv(m) \in r_i(t)$
– $\pi(r,t)(has_i(m)) =$ **true** if and only if there exists m' such that $m \sqsubseteq m'$ and $recv(m') \in r_i(t)$.

What properties can we express using the above language? The property that will be the focus of this chapter is that of confidentiality of messages, as described in Section 6.3. Intuitively, confidentiality guarantees that throughout a protocol interaction, the adversary does not come to know a particular message. Confidentiality properties are stated naturally in terms of knowledge, for example, "agent 1 knows that the key k is a key known only to agent 2 and himself". Confidentiality properties are well studied, and central to most of the approaches to symbolic reasoning about security protocols. Higher-level security properties, such as authentication properties, can often be derived from confidentiality properties.

To illustrate some of the issues involved, consider the Needham-Schroeder public key authentication protocol, which was presented in Section 1.2:

$$\begin{array}{lrcl} 1. & A \to B & : & \{\!|n_A, A|\!\}_{k_B} \\ 2. & B \to A & : & \{\!|n_A, n_B, B|\!\}_{k_A} \\ 3. & A \to B & : & \{\!|n_B|\!\}_{k_B}. \end{array}$$

This protocol uses asymmetric cryptography, and k_A and k_B are agent A and B's respective public encryption keys. The values n_A and n_B are nonces, which are assumed to be unpredictable. The confidentiality property of this protocol can be expressed informally as follow: under suitable assumptions on the keys known to the adversary and the fact that B is running his part of the protocol, A knows that

n_A and n_B are kept confidential between her and B.[1] From this, it is possible to derive an authentication property, namely that A knows that she is interacting with B, because she has received a message containing n_A, which only B could have produced. Similarly, A also knows that when B receives her message, B will know that he is interacting with A, because only A knows the nonce n_B which is part of the last message. Similar reasoning can be applied to B. This argument relies on the confidentiality of the nonces n_a and n_b. Using knowledge, this is simply the fact that no agent but A and B knows $has_i(n_A)$ or $has_i(n_B)$.

However, the knowledge operator suffers from the drawback of logical omniscience. More specifically, at every point where an adversary i intercepts a message $\{\!|n_A, n_B, B|\!\}_{k_A}$, then $K_i(has_i(n_A))$ is true (since $n_A \sqsubseteq \{\!|n_A, n_B, B|\!\}_{k_A}$), and hence the adversary knows that he has seen the nonce n_A, irrespectively of whether or not he knows the decryption key corresponding to k_A. This is clearly not a desirable result. The adversary having the implicit knowledge that n_A is part of the message does not suffice, in general, for the adversary to *explicitly* know that n_A is part of the message. Intuitively, the adversary may not have the capabilities to realize he has seen n_A.

A more reasonable interpretation of confidentiality in this particular setting is expressed by $\neg X_i(has_i(n_A))$, that is, the adversary does not explicitly know (cannot compute) whether he has seen the nonce n_A. Most logics of security introduce special primitives to capture the fact that the adversary can see a message m encrypted with key k only if he has access to the key k. Doing this hardwires the capabilities of the adversary into the semantics. Changing these capabilities requires changing the semantics. With algorithmic knowledge, we simply need to supply the appropriate knowledge algorithm to the adversary, capturing his capabilities. In the following section, we examine in more detail the kind of knowledge algorithms that correspond to interesting capabilities.

8.2 Passive Adversaries

As we outlined in Sections 7.1 and 8.1, interpreted algorithmic knowledge security systems provide a foundation for representing security protocols, and support a logic for writing properties based on knowledge, both traditional and algorithmic. For the purposes of analyzing security protocols, traditional knowledge models an agent's beliefs about what can happen in the protocol, while algorithmic knowledge models the adversary's capabilities, possibly resource-bounded. We have not said anything yet as to what kind of algorithms are useful, short of the fact that we

[1] Strictly speaking, it may make more sense to talk about belief in the confidentiality of n_A and n_B rather than belief. For simplicitly, we will talk about knowledge in this chapter, but most of what we say about knowledge can be said about belief.

typically care about sound knowledge algorithms. For the purpose of security, the knowledge algorithms given to adversaries are the most important, as they capture the facts that adversaries can compute given what they have seen. In this section, we show how we can capture different capabilities for the adversary rather naturally in this framework. We first show how to capture the standard Dolev-Yao model of adversary. We then show how to account for adversaries in the Duck-Duck-Goose protocol, as well as adversaries that can perform self-validating guesses (also known as offline-dictionary attacks).

For this section, assume a passive (or eavesdropping) adversary, that simply records every message exchanged by the agents. For simplicity, assume a single adversary per system; our results extend to the general case immediately, but the notation starts becoming cumbersome. Passive adversaries can be modeled formally as follows. An *interpreted algorithmic knowledge security system with passive adversary a (for $a \in \{1, \ldots, n\}$)* is an interpreted algorithmic knowledge security system $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ such that for all points (r, t), the following constraints are satisfied:

P1. $r_a(t)$ consists only of $recv(u)$ events;
P2. there exists an event $recv(u)$ in $r_a(t)$ for every event $send(j, u)$ in $r_j(t)$, for all j.

P1 captures the passivity of the adversary—he can only receive messages, not send any; P2 says that every message sent by an agent is copied to the adversary's local state.

The only thing that remains to be done now is to define the capabilities of the adversary to derive information from those messages in his local state. This is done by defining suitable knowledge algorithms.

8.2.1 The Dolev-Yao Adversary

Recall that the Dolev-Yao adversary model is a combination of assumptions on the encryption scheme used and the capabilities of the adversaries. Specifically, the encryption scheme is taken to be the symbolic encryption scheme generated by $\mathcal{P}$ and $\mathcal{K}$, while the capabilities are given by the derivation $H \vdash_{DY} m$ indicating that message m can be derived from the set of messages H using the inference rules described in 6.2.

To capture the capabilities of a Dolev-Yao adversary, we specify how the adversary can tell if he in fact *has* a message, by defining a knowledge algorithm $\mathtt{A}_i^{\mathrm{DY}}$ for adversary i. Recall that a knowledge algorithm for agent i takes as input a formula and agent i's local state (which by assumption contains the messages received by i). The most interesting case in the definition of $\mathtt{A}_i^{\mathrm{DY}}$ is when the for-

$submsg(m, m', K) =$ if $m = m'$ then
    return $true$
if m' is $\{m_1\}_k$ and $k^{-1} \in K$ then
    return $submsg(m, m_1, K)$
if m' is (m_1, m_2) then
    return $submsg(m, m_1, K) \vee submsg(m, m_2, K)$
return $false$

$getkeys(m, K) =$ if $m \in \mathcal{K}$ then
    return $\{m\}$
if m' is $\{m_1\}_k$ and $k^{-1} \in K$ then
    return $getkeys(m_1, K)$
if m' is (m_1, m_2) then
    return $getkeys(m_1, K) \cup getkeys(m_2, K)$
return $\{\}$

$keysof(\ell) = K \leftarrow initkeys(\ell)$
loop until no change in K
    $K \leftarrow \bigcup_{recv(m) \in \ell} getkeys(m, K)$
return K

Figure 8.1. Dolev-Yao knowledge algorithm auxiliary functions

mula is $has_i(m)$. To compute $\mathtt{A}_i^{\mathrm{DY}}(has_i(m), \ell)$, the algorithm simply checks, for every message m' received by the adversary, whether m is a submessage of m', according to the keys that are known to the adversary. Assume that the adversary's initial state consists of the set of keys initially known by the adversary. This will typically contain, in a public-key cryptography setting, the public keys of all the agents. The function $initkeys(\ell)$ denotes the set of initial keys known by agent i in local state ℓ. (Recall that a local state for agent i is the sequence of events pertaining to agent i, including any initial information in the run, in this case, the keys initially known.) Checking whether m is a submessage of m' is performed by a function $submsg$, which can take apart messages created by concatenation, or decrypt messages as long as the adversary knows the decryption key.

$\mathtt{A}_i^{\mathrm{DY}}(has_i(m), \ell) =$ if $m \in initkeys(\ell)$ then return "Yes"
    $K = keysof(\ell)$
    for each $recv(m')$ in ℓ
        if $submsg(m, m', K)$ then
            return "Yes"
    return "No".

The auxiliary functions used by the algorithm are given in Figure 8.1.

According to the Dolev-Yao model, the adversary cannot explicitly compute anything interesting about what other messages agents have. Hence, for other primitives, including $has_j(m)$ for $j \neq i$, $\mathtt{A}_i^{\mathrm{DY}}$ returns "?". For formulas of the form $K_j\varphi$ and $X_j\varphi$, $\mathtt{A}_i^{\mathrm{DY}}$ also returns "?". For Boolean combinations of formulas, $\mathtt{A}_i^{\mathrm{DY}}$ returns the corresponding Boolean combination (where the negation of "?" is "?", the conjunction of "No" and "?" is "No", and the conjunction of "Yes" and "?" is "?") of the answer for each $has_i(m)$ query.

The following result shows that an adversary using $\mathtt{A}_i^{\mathrm{DY}}$ recognizes (i.e., returns "Yes" to) $has_i(m)$ in state ℓ if and only if m exactly the messages determined to be in the set of messages that can be derived (according to $\vdash_{DY}$) from the messages received in that state together with the keys initially known, Moreover, if a $has_i(m)$ formula is derived at the point (r, t), the $has_i(m)$ is actually true at (r, t) (so that $\mathtt{A}_i^{\mathrm{DY}}$ is sound).

Theorem 8.1. *Let $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ be an interpreted algorithmic knowledge security system where $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY}}$. Then $(\mathcal{I}, r, t) \models X_i(has_i(m))$ if and only if $\{m \mid recv(m) \in r_i(t)\} \cup initkeys(r_i(t)) \vdash_{DY} m$. Moreover, if $(\mathcal{I}, r, t) \models X_i(has_i(m))$ then $(\mathcal{I}, r, t) \models has_i(m)$.*

In particular, for an interpreted algorithmic knowledge security system with a passive adversary a with $\mathtt{A}_a = \mathtt{A}_a^{\mathrm{DY}}$, Proposition 8.1 captures the knowledge of a passive Dolev-Yao adversary.

8.2.2 The Duck-Duck-Goose Adversary

The key advantage of our framework is that it is easy to change the capabilities of the adversary beyond those prescribed by the Dolev-Yao model, to incorporate protocol-specific knowledge, for instance. In the Duck-Duck-Goose example, assume that the adversary maintains in his local state a list of all the bits received corresponding to the key of the other agent. It is easy to write the algorithm so that if the adversary's local state contains all the bits of the key of the other agent, then the adversary can decode messages that have been encrypted with that key. Specifically, assume that key k is being sent in the Duck-Duck-Goose example. Then for an adversary i, $has_i(k)$ will be false until all the bits of the key have been received. This translates immediately into the following algorithm $\mathtt{A}_i^{\mathrm{DDG}}$:

$$\mathtt{A}_i^{\mathrm{DDG}}(has_i(k), \ell) = \text{if all the bits recorded in } \ell \text{ form } k \text{ then return "Yes" else return "No".}$$

$\mathtt{A}_i^{\mathrm{DDG}}$ handles other formulas in the same way as $\mathtt{A}_i^{\mathrm{DY}}$.

Of course, nothing keeps us from combining algorithms, so that we can imagine

an adversary intercepting both messages and key bits, and using an algorithm $\mathtt{A}_i$ which is a combination of the Dolev-Yao algorithm and the Duck-Duck-Goose algorithm, such as:

$$\begin{aligned}\mathtt{A}_i(\varphi,\ell) \quad = \quad & \text{if } \mathtt{A}_i^{\mathrm{DY}}(\varphi,\ell) = \text{“Yes” then} \\ & \text{return “Yes”} \\ & \text{else return } \mathtt{A}_i^{\mathrm{DDG}}(\varphi,\ell).\end{aligned}$$

This assumes that the adversary knows the protocol, and hence knows when the key bits are being sent. The algorithm above captures this protocol-specific knowledge.

8.2.3 The Lowe Adversary

For a more realistic example of an adversary model that goes beyond Dolev-Yao, consider the following adversary model, due to Gavin Lowe, that permits the analysis of protocols subject to offline guessing attacks. The intuition is that some protocols provide for a way to “validate” the guesses of an adversary. For a simple example of this, here is a simple challenge-based authentication protocol:

$$\begin{array}{lrcl} 1. & A \rightarrow S & : & A \\ 2. & S \rightarrow A & : & n_s \\ 3. & A \rightarrow S & : & \{n_s\}_{p_a}. \end{array}$$

Intuitively, A tells the server S that she wants to authenticate herself. S replies with a challenge n_s. A sends back to S the challenge encrypted with her password p_a. Presumably, S knows the password, and can verify that she gets $\{n_s\}_{p_a}$. Unfortunately, an adversary can overhear both n_s and $\{n_s\}_{p_a}$, and can “guess” a value g for p_a and verify his guess by checking if $\{n_s\}_g = \{n_s\}_{p_a}$. The key feature of this kind of attack is that the guessing (and the validation) can be performed offline, based only on the intercepted messages. A well known variant of this problem is the problem of weak passwords, that is, passwords that can be verified offline using a dictionary. Dictionary attacks can be modeled using offline guessing, by assuming that the dictionary is part of the initial state of the adversary.

To account for this capability of adversaries is actually fairly complicated. We present a slight variation of Lowe’s description, mostly to make it notationally consistent with the rest of the section.

Lowe’s model relies on a basic one-step reduction function, $S \rhd_l m$, saying that the messages in S can be used to derive the message m. This is essentially the same as $\vdash_{DY}$, except that it represents a single step of derivation. Moreover, the relation is “tagged” by the kind of derivation performed (l):

$$\{m, k\} \rhd_{\mathsf{enc}} \{m\}_k$$

$$\{\{m\}_k, k^{-1}\} \rhd_{\mathsf{dec}} m$$
$$\{(m_1, m_2)\} \rhd_{\mathsf{fst}} m_1$$
$$\{(m_1, m_2)\} \rhd_{\mathsf{snd}} m_2.$$

Lowe also includes a reduction to derive (m_1, m_2) from m_1 and m_2. We do not add this reduction to simplify the presentation. It is straightforward to extend the work in this section to account for this augmented derivation.

Given a set H of message, and a sequence t of one-step reductions, define inductively the set $[H]_t$ of messages obtained from the one-step reductions given in t:

$$[H]_{\langle\rangle} = H$$
$$[H]_{\langle S \rhd_l m\rangle \cdot t} = \begin{cases} [H \cup \{m\}]_t & \text{if } S \subseteq H \\ \mathit{undefined} & \text{otherwise.} \end{cases}$$

Here, $\langle\rangle$ denotes the empty trace, and $t_1 \cdot t_2$ denotes trace concatenation. A trace t is said to be *monotone* if, intuitively, it does not perform any one-step reduction that "undoes" a previous one-step reduction. For example, the reduction $\{m, k\} \rhd \{m\}_k$ undoes the reduction $\{\{m\}_k, k^{-1}\} \rhd m$.

A set H of messages *validates* a guess m if, intuitively, H contains enough information to verify that m is indeed a good guess. Intuitively, this happens if a value v (called a validator) can be derived from the messages in $H \cup \{m\}$ in a way that uses the guess m, and either that (a) validator v can be derived in a different way from $H \cup \{m\}$, (b) the validator v is already in $H \cup \{m\}$, or (c) the validator v is a key whose inverse is derivable from $H \cup \{m\}$. For example, in the protocol exchange at the beginning of this section, the adversary sees the messages $H = \{n_s, \{n_s\}_{p_a}\}$, and we can check that H validates the guess $m = p_a$: clearly, $\{n_s, m\} \rhd_{\mathsf{enc}} \{n_s\}_{p_a}$, and $\{n_s\}_{p_a} \in H \cup \{m\}$. In this case, the validator $\{n_s\}_{p_a}$ is already present in $H \cup \{m\}$.

We can now define the relation $H \vdash_{\mathrm{L}} m$ that says that m can be derived from H by a Lowe adversary. Intuitively, $H \vdash_{\mathrm{L}} m$ if m can be derived by Dolev-Yao reductions, or m can be guessed and validated by the adversary, and hence susceptible to an attack. Formally, $H \vdash_{\mathrm{L}} m$ if and only if $H \vdash_{DY} m$ or there exists a monotone trace t, a set S, and a validator v such that

(1) $[H \cup \{m\}]_t$ is defined,
(2) $S \rhd_l v$ is in t,
(3) there is no trace t' such that $S \subseteq [H]_{t'}$, and
(4) either:
   (a) there exists $(S', l') \neq (S, l)$ with $S' \rhd_{l'} v$ in t,

$guess(m, \ell) =$ $H \leftarrow reduce(\{m \mid recv(m) \text{ in } \ell\} \cup initkeys(\ell)) \cup \{m\}$
$reds \leftarrow \{\}$
loop until $reductions(H) - reds$ is empty
$(S, l, v) \leftarrow$ pick an element of $reductions(H) - reds$
if $\exists(S', l', v) \in reds$ s.t. $S' \neq S$ and $l' \neq l$ then return "Yes"
if $v \in H$ then return "Yes"
if $v \in \mathcal{K}$ and $v^{-1} \in H$ then return "Yes"
$reds \leftarrow reds \cup \{(S, l, v)\}$
$H \leftarrow H \cup \{v\}$
return "No"

$reduce(H) =$ loop until no change in H
$r \leftarrow reductions(H)$
for each (S, l, v) in r
$H \leftarrow H \cup \{v\}$
return H

$reductions(H) =$ $reds \leftarrow \{\}$
for each (m_1, m_2) in H
$reds \leftarrow \{(\{m\}, \mathsf{fst}, m_1), (\{m\}, \mathsf{snd}, m_2)\}$
for each m_1, m_2 in H
if $m_2 \in \mathcal{K}$ and $sub(\{m_1\}_{m_2}, H)$ then
$reds \leftarrow \{(\{m_1, m_2\}, \mathsf{enc}, \{m_1\}_{m_2})\}$
if m_1 is $\{m'\}_k$ and m_2 is k^{-1} then
$reds \leftarrow \{(\{m_1, m_2\}, \mathsf{dec}, m')\}$
return $reds$

$sub(m, H) =$ if $H = \{m\}$ then return $true$
if $H = \{(m_1, m_2)\}$ then
return $sub(m, \{m_1\}) \vee sub(m, \{m_2\})$
if $H = \{\{m'\}_k\}$ then return $sub(m, \{m'\})$
if $|H| > 1$ and $H = \{m'\} \cup H'$ then
return $sub(m, \{m'\}) \vee sub(m, H')$
return $false$

Figure 8.2. Lowe knowledge algorithm auxiliary functions

(b) $v \in H \cup \{m\}$, or
(c) $v \in \mathcal{K}$ and $v^{-1} \in [H \cup \{m\}]_t$.

We can verify that the above formalization captures the intuition about validation given earlier. Specifically, condition (1) says that the trace t is well-formed, condition (2) says that the validator v is derived from $H \cup \{m\}$, condition (3) says that deriving the validator v depends on the guess m, and condition (4) specifies when a validator v validates a guess m, as given earlier.

It is straightforward to define a knowledge algorithm $\mathtt{A}_i^{\mathrm{L}}$ to capture the capabilities of the Lowe adversary. Again, the only case of real interest is what $\mathtt{A}_i^{\mathrm{L}}$ does on

input $has_i(m)$.

$$\begin{aligned}\mathtt{A}_i^{\mathrm{L}}(has_i(m),\ell) = \text{if } \mathtt{A}_i^{\mathrm{DY}}(has_i(m),\ell) = \text{“Yes” then}\\ \text{return “Yes”}\\ \text{if } guess(m,\ell) \text{ then}\\ \text{return “Yes”}\\ \text{return “No”.}\end{aligned}$$

The auxiliary functions used by the algorithm are given in Figure 8.2.

As before, we can check the correctness and soundness of the algorithm:

Theorem 8.2. *Let $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ be an interpreted algorithmic knowledge security system where $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{L}}$. Then $(\mathcal{I}, r, t) \models X_i(has_i(m))$ if and only if $\{m \mid recv(m) \in r_i(t)\} \cup initkeys(r_i(t)) \vdash_{\mathrm{L}} m$. Moreover, if $(\mathcal{I}, r, t) \models X_i(has_i(m))$ then $(\mathcal{I}, r, t) \models has_i(m)$.*

8.3 Probabilistic Adversaries

The Lowe adversary model described in Section 8.2.3 does not involve randomization, since the adversary needs to explicitly specify which value to guess and validate. Another guessing model consists in extending a Dolev-Yao adversary by allowing him to try to guess keys before determining if he has a given message. But how should these keys be chosen? One obvious way is that they should be chosen from the space of keys according to some probability distribution (perhaps chosen at random).

As expected, probabilistic algorithmic knowledge can deal well with an adversary who guesses keys randomly in an effort to crack an encrypted message. Consider the Dolev-Yao knowledge algorithm $\mathtt{A}_i^{\mathrm{DY}}$, modified so as to allow for key guesses. Assume that the key space is finite, and let $guesskeys(n)$ return n of these, chosen uniformly at random. Let $\mathtt{A}_i^{\mathrm{DY+rg}(n)}$ be the result of modifying the second line of $\mathtt{A}_i^{\mathrm{DY}}$ to take random guessing into account (the rg stands for *random guess*).

$$\begin{aligned}\mathtt{A}_i^{\mathrm{DY+rg}(n)}(has_i(m),\ell) = \text{if } m \in initkeys(\ell) \text{ then return “Yes”}\\ K = keysof(\ell) \cup guesskeys(n)\\ \text{for each } recv(m') \in \ell \text{ do}\\ \text{if } submsg(m, m', K) \text{ then}\\ \text{return “Yes”}\\ \text{return “No”.}\end{aligned}$$

Essentially, using $\mathtt{A}_i^{\mathrm{DY}+\mathrm{rg}(n)}$, the adversary gets to work with whatever keys he already had available, all the keys he can obtain using the standard Dolev-Yao algorithm, and an additional n randomly chosen keys. Of course, if the total number $|\mathcal{K}|$ of keys is large relative to n, making n random guesses should not help much. The algorithmic knowledge framework lets us make this precise.

An interpreted probabilistic algorithmic knowledge security system is the obvious generalization of an interpreted algorithmic knowledge security system, along the lines developed in Chapter 4 to give a semantics to randomized knowledge algorithms.

Theorem 8.3. *Suppose that $\mathcal{J} = (\mathcal{R}, \pi, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ is an interpreted probabilistic algorithmic knowledge security system with an adversary as agent i and that $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY}+\mathrm{rg}(n)}$. Let K be the number of distinct keys used in the messages in the adversary's local state $r_i(t)$ (that is, the number of keys used in the messages that the adversary has intercepted at a point (r,t)). Suppose that $K/|\mathcal{K}| < 1/2$ and that ν is the uniform distribution on sequences of coin tosses. If $(\mathcal{J}, r, t, v) \models \neg K_i X_i(has_i(m))$, then $(\mathcal{J}, r, t, v) \models \Pr(X_i(has_i(m))) < 1 - e^{-2nK/|\mathcal{K}|}$. Moreover, if $(\mathcal{J}, r, t, v) \models X_i(has_i(m))$ then $(\mathcal{J}, r, t, v) \models has_i(m)$.*

Theorem 8.3 says that what we expect to be true is in fact true: random guessing of keys does not help much (at least, if the number of keys guessed is a small fraction of the total numbers of keys). In other words, if it is possible that the adversary does not have algorithmic knowledge of m, then the probability that it has algorithmic knowledge is low. While this result just formalizes our intuitions, it does show that the probabilistic algorithmic knowledge framework has the resources to formalize these intuitions naturally. More importantly, it allows us to characterize an adversary without being limited to Dolev-Yao style adversaries.

8.4 Active Adversaries

In the last section, we were concerned with describing the capabilities of adversaries in term of deriving explicit information from messages stored in their local state. The adversaries were assumed passive, intercepting every message exchanged by the protocol participants. Eavesdropping adversaries can breach confidentiality of messages.

There are many attacks on security protocols that do not necessarily involve a breach of confidentiality. For instance, some authentication properties aim at ensuring that no adversary can pass himself off as another agent. This presumes that the adversary is able to interact with other agents. Even as as far as confidential-

ity is concerned, an active adversary can attempt to manipulate other agents into revealing secrets.

Active adversaries are more complex to reason about, since they interact with other agents. Moreover, there is the question of exactly what messages they can send. The answer depends of course on their capabilities, which are already captured using knowledge algorithms. Formally, at a local state ℓ, an adversary using knowledge algorithm $\mathtt{A}_i$ can construct the messages $Cons_i(\ell)$, defined to be the closure under $\{\cdot\}$ and $(\cdot,\cdot)$ of the set $\{m \mid \mathtt{A}_i(has_i(m), \ell) = \text{“Yes”}\}$ of messages the adversary has. Thus, we can use the knowledge algorithm describing the capabilities of the adversary not only in the specification language in which we write down the security properties, but also at the level of modeling the protocol.

When modeling, it is necessary to decide whether adversaries are outsiders to the system, or insiders. Recall from Section 6.2 that an insider is an adversary that other agents know about, and with whom they can initiate interactions.

Consider the case where there is a single active adversary. (The definitions generalize to the multiple adversaries case immediately.) An *interpreted algorithmic knowledge security system with active (insider) adversary a (for $a \in \{1,\ldots,n\}$)* is an interpreted algorithmic knowledge security system $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ such that for all points (r,t), the following constraints are satisfied:

A1. for every $recv(m) \in r_a(t)$, there is a corresponding $send(j,u)$ in $r_i(t)$ for some i;

A2. for every $send(j,m) \in r_a(t)$, we have $m \in Cons_a(r_a(t))$.

A1 says that every message sent by the agents can be intercepted by the adversary, and end up in the adversary's local state, rather than reaching its destination. A2 says that every message sent by the adversary must have been constructed out of the messages in his local state according to his capabilities. (Note that the adversary can forge the “send” field of the messages.)

To accomodate outsider adversaries, it suffices to add the restriction that no message is sent directly to the adversary. Formally, an *interpreted algorithmic knowledge security system with active (outsider) adversary a (for $a \in \{1,\ldots,n\}$)* is an interpreted algorithmic knowledge security system with an active insider adversary a $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ such that for all points (r,t) and for all agents i, the following additional constraint is satisfied:

A3. for every $send(j,m) \in r_i(t)$, $j \neq a$.

8.5 The Logical Approach

We have presented a framework for security analysis using algorithmic knowledge. The knowledge algorithm can be tailored to account for both the capabilities of the adversary and the specifics of the protocol under consideration. Of course, it is always possible to take a security logic and extend it in an *ad hoc* way to reason about adversary with different capabilities. Our approach has many advantages over *ad hoc* approaches: it is a general framework (changing the algorithm used by the adversary changes his capabilities), and it permits reasoning about protocol-specific issues (such as the agent sending the bits of its key). Another advantage of our approach is that it naturally extends to the probabilistic setting. For instance, probabilistic protocols are easily handled, by considering multiagent systems with an associated probability distribution on the runs. Randomized knowledge algorithms can also be handled, using the techniques developed in Chapter 4.

One natural question that remains is whether there is in fact a need for formal logics for reasoning about security. After all, many of the approaches described in Chapter 6 are not based on a logical specification language, and are certainly successful. In the remainder of this section, let me attempt to motivate the logical methodology.

First, what do we mean by logic for reasoning about a phenomenon? There are essentially two (not incompatible) views on the role of logics. The first view is that a logic captures valid patterns of reasoning about a particular phenomenon, where the models simply "justify" the reasoning process. The second view starts with a structure, which is used to capture abstractly a situation. The logic provides a language in which to write down properties of the structure. Of course, these two views are intimately related, but they have distinct methodologies. In this dissertation, we have mostly subscribed to the second view, and structured my arguments accordingly.

Of course, the structure-centric view begs a question: why should we bother introducing a formal language to reason about the structure, since we could do our reasoning directly on the structure, using standard mathematical techniques? This is the approach used, for instance, in probability theory and in economics. There are at least two reasons for advocating a logic. First, it provides a formal language for capturing certain notions independently of a particular structure. For instance, authentication can sometimes be established by proving that two agents share secrets initially known to the individuals only. In a different context, authentication corresponds to a different property of the abstract model representing the situation. Having a formal language gives us the possibility of writing down a formula that captures authentication in general, meaningful across different models. Secondly,

a formal language provides the structure for deriving proofs, either via the proof theory of the logic, or by induction on the structure of formulas in the language.

Therefore, there is a distinction between reasoning directly about a phenomenon over a particular structure, and using a formal language capturing the phenomenon in question. While we advocate the use of a formal language in this dissertation, there is the second question of the appropriate language to use. There are at least two choices. The first, which is the one we follow in this dissertation, is to introduce a special-purpose logic for reasoning about the phenomenon of interest. This involves carefully choosing the structures with which to capture the particulars of the real-world phenomenon being studied, and then carefully choosing the logical operators with a suitable and natural semantics. The language also has to be expressive enough to express the specifications of interest. The second alternative is to use a generic logic, such as first-order logic or higher-order logic. The advantage of using, say, higher-order logic is that most of mathematics is directly available, since much of it can be formalized in higher-order logic. Moreover, there exists theorem proving environments for higher-order logic that provide tools for proving theorems semi-automatically. Such an approach is therefore useful for proving hard (or tedious) theorems about particular situations. However, this is just highly formalized reasoning about a particular structure. Most of the criticisms laid out about reasoning directly about structures apply here, in that it may become difficult to highlight the commonality between different structures when reasoning at the level of the structures themselves—even if this reasoning is done via a formalized notation rather than informal mathematics. Finally, expressive logics such as higher-order logic are typically highly undecidable, as opposed to hand-crafted logics, which can often be tailored to be decidable.

Notes

Most of the work in this chapter first appeared in [Halpern and Pucella 2002], except for Section 8.3, which first appeared in [Halpern and Pucella 2003c].

While we focus on confidentiality and authentication properties in this dissertation, we should point out that the models described in Chapter 7 and the epistemic core of the logic in this chapter have been used to reason about other security properties, such as information flow and anonymity [Halpern and O'Neill 2002; Halpern and O'Neill 2003]. Halpern and O'Neill [2002], in particular, present a knowledge-based definition of secrecy that goes beyond the kind of confidentiality property studied here. (They do not take cryptography into account, however.) Higher-level security properties, such as authentication properties, can often be

established via confidentiality properties; Syverson and Cervesato [2001] discuss some of these issues.

The adversary in Section 8.2.3 was introduced by Lowe [2002]. We refer the reader to the original paper for a discussion of the design choices, for details on undoing reductions, and for a discussion of implementation issues.

The Dolev-Yao adversary is the most widespread adversary in the literature. Part of its attraction is its tractability, making it possible to develop formal systems to automatically check for safety with respect to such adversaries [Millen, Clark, and Freedman 1987; Mitchell, Mitchell, and Stern 1997; Paulson 1998; Lowe 1998; Meadows 1996]. The idea of moving beyond the Dolev-Yao adversary is not new. Other approaches offer some possibility of extending the adversary model. For instance, the framework of Paulson [1998], Clarke, Jha and Morrero [1998], and Lowe [1998] describe the adversary via a set of derivation rules, which could be modified by adding new derivation rules.

There are other approaches that weaken the Dolev-Yao adversary assumptions either by taking concrete encryption schemes into account, or at least by adding new algebraic identities to the algebra of messages. Bieber [1990] does not assume that the encryption scheme is a free algebra, following an idea due to Merritt and Wolper [1985]. Even et al. [1985] analyze ping-pong protocols under RSA, taking the actual encryption scheme into account. The applied π-calculus of Abadi and Fournet [2001] permits the definition of an equational theory over the messages exchanged between processes, weakening some of the symbolic encryption scheme assumptions when the applied π-calculus is used to analyze security protocols.

Most frameworks for reasoning about security protocols using knowledge or belief have had to circumvent the logical omniscience problem. In the context of security, this has taken the form of using different semantics for knowledge, either by introducing hiding operators that hide part of the local state for the purpose of indistinguishability (as done, for example, in [Abadi and Tuttle 1991]), or by using notions such as *awareness* [Fagin and Halpern 1988] to capture an intruder's inability to decrypt [Accorsi, Basin, and Viganò 2001]. The use of awareness by Accorsi et al. [2001] is not motivated by the desire to model more general adversaries, but by the desire to restrict the number of states one needs to consider in models. Halpern, Moses and Tuttle [1988] analyze zero-knowledge protocols using a notion of resource-bounded knowledge defined by Moses [1988].

9
Epistemic Foundations of Security Protocols

THE logic we described in the last chapter was based on a formal notion of knowledge. Of course, this is not the first attempt at basing a logic for security protocol analysis on epistemic notions. For the past fifteen years, there has been an intuition in the world of security that formal theories of knowledge and belief should have something interesting to say about security protocols, and cryptographic protocols in particular. One of the earliest and the most discussed is BAN logic, which we already described in Section 6.4.3. As we pointed out, BAN has been the subject of many criticisms, mostly in connection with its verification method based on idealization.

Other approaches to the formal analysis of security protocols have emerged in recent years that seem to provide stronger correctness guarantees than BAN logic. These approaches, many of which were described in Chapter 6, work by reasoning directly about the behaviour of adversaries, which cannot be done in BAN. These more recent approaches do not make explicit use of logics of knowledge and belief, and yet have been quite successful, arguably more so than approaches based on epistemic notions. One might therefore ask whether the intuitions about the utility of using knowledge and belief to reason about security protocols were misplaced.

We argue in this chapter that epistemic notions are in fact needed in any analysis of security protocols. We focus on a particular problem: in order to analyze protocols that use nonces, we need to make precise the properties of these nonces. We proceed by analyzing the well-known Needham-Schroeder public key authentication protocol, already introduced in Section 1.2, under the standard Dolev-Yao adversary model. This shows that the Dolev-Yao model does not capture some of the key intuitions underlying the Needham-Schroeder protocol—in particular, it leads to unreasonable conclusions about who must have sent certain messages. Since these conclusions follow from the Dolev-Yao model, they must be reached by any approach based on this adversary model (the majority of the approaches described in Section 6.4). This raises concerns about the interpretation of what has

been proved. To address these concerns, we formulate a variant of the Dolev-Yao model in which the adversary can guess nonces. We show that, in such a model, it does not suffice that a nonce be *fresh* (that is, that a nonce be a new message, different from any other previously seen message). If this were enough, nonces could be taken to be sequence numbers. A nonce must also be *unpredictable*. Unpredictability is an inherently epistemic notion; we show how it can be modeled in the logic of Chapter 8, extended with probability and time.

An additional argument for the suitability of such a logic to the purpose of security protocol specification is also provided; it is straightforward to capture higher-level security notions that seem useful when reasoning about protocols. The BAN logic resulted from a careful analysis of notions useful for protocol analysis. We show how to recover BAN logic by defining a translation from BAN forulas to formulas of our logic. This establishes, among other things, the expressive power of our framework. It also helps illuminate the assumptions underlying BAN, and more importantly provides insight into notions that are useful for protocol analysis.

How can we justify that we have captured the meaning of the BAN operators? The original presentation of BAN gave axioms characterizing some properties of the operators, supplemented with informal descriptions of their meaning. We would certainly expect that our interpretation of each BAN operator satisfies the same axioms as the original BAN operator. We show that the (translated versions of) the BAN axioms are indeed sound with respect to a restricted class of semantic models that capture our proposed guessing Dolev-Yao adversary behaviour. The need to consider this restricted class of models is not an artifact of our interpretation, but is a consequence of assumptions built into BAN.

Of course, it is not enough to show that the BAN axioms are validated. We also show that the "meaning" of the operators is similar. This cannot be established formally, since BAN did not provide a formal meaning for their operators. We provide some evidence for the reasonableness of our translation by showing how our logic handles the Needham-Schroeder protocol alreeady presented in Section 1.2. We show that the Needham-Schroeder protocol generates a model that satisfies the (translated version of the) specifications for the protocol given in the original BAN analysis of this protocol. Significantly, the reasoning makes no use of the idealization step that has frequently been argued to be one of the weakest points of BAN.

We emphasize that we are not attempting to give a full reconstruction of BAN. And we certainly do not claim that BAN's choice of primitives was the "right" set. There has been a debate in the literature about what the appropriate set of primitives for a security logic should be. Our view is that, where possible, one should minimize the number of primitives, and use constructs that are well understood and have utility in a broader domain. One advantage of expressing higher-level

operators in terms of simpler primitives, such as those provided by the logic of the previous chapter (extended with time and probability), is that it provides a basis for extending the higher-level operators to more general situations.

Why focus on BAN at all? The BAN operators are an attempt to clarify the intuitive notions that arise when reasoning about security protocols: belief, trust, freshness, jurisdiction. Therefore, they seem like reasonable candidates for the higher-level security notions that we claim to be able to capture. However, we believe that the epistemic approach and the insights that we obtain from it go well beyond BAN, and should be relevant for any formal attempt to reason about security.

9.1 Nonces, Uniqueness, and Unpredictability

Recall from Section 1.2, the following version of the Needham-Schroeder public key authentication protocol, without a server, which we call the SNS protocol (for "simplified Needham-Schroeder"):

$$\begin{array}{lllll} 1. & A \to B & : & \{\!|n_A, A|\!\}_{k_B} \\ 2. & B \to A & : & \{\!|n_A, n_B|\!\}_{k_A} \\ 3. & A \to B & : & \{\!|n_B|\!\}_{k_B}. \end{array}$$

It presumes that agents A and B know each other's public keys, denoted k_A and k_B, respectively. Here, n_A and n_B are nonces, and k_A and k_B are the public keys of agents A and B, respectively. The claim is that, at the end of this protocol, A and B know that they have communicated with each other. The protocol also establishes n_A and n_B as secrets known only to A and B.

Assume a context where the adversary X is not an insider. In other words, the initiator A of the protocol will will never try to authenticate herself to X. The adversary can of course intercept messages, but no message is addressed explicitly to him. (We saw in Section 1.2 that the original SNS protocol is vulnerable to a man-in-the-middle attack when the adversary is an insider. We do not focus on this attack in this chapter, although it can certainly be captured in our framework.)

We concentrate on the Dolev-Yao adversary model described in Section 6.2, since it essentially underlies the BAN approach and most other approaches to symbolic protocol analysis. Recall that this model is a combination of assumptions on the encryption scheme and the capabilities of the adversaries. The encryption scheme is the symbolic encryption scheme of Section 6.1, that is, the free algebra generated by the plaintext and the keys, using the operations of pairing, encryption, and decryption. It follows that it is always possible in this model to determine whether a message is a plaintext, a pairing, or an encrypted text. There are also no collisions; messages always have a unique decomposition. The adversary is

allowed to *extract* messages (from messages he has intercepted) by taking apart pairings and decrypting encrypted messages if he knows the decryption key, and is allowed to *construct* new messages (from messages he has been able to extract) by pairing and by encryption using known encryption keys. The adversary, however, is not allowed to guess keys, guess nonces, or attempt to "crack" encrypted messages using cryptanalysis.

There have been many analyses of the SNS protocol under Dolev-Yao adversaries. Most analyses establish a result that can be paraphrased as follows, concerning the first two messages of the protocol: "The guarantees for A are that her nonce remains secret from the adversary and that B is present. The latter follows from the former, for if the adversary does not know n_A then he could not have sent message 2". This is based on the interpretation of confidentiality from the adversary given in Section 6.3—the adversary cannot derive the secret. In MSR, the model-based approach described in Section 6.4.1, this amounts to saying that the atomic formula $M(n_A)$ does not appear in any state derivable from the initial state, where M is the predicate representing the adversary's knowledge. In order to prove such a result, most approaches assume that nonces are uniquely generated. In other words, if a protocol step calls for the generation of a nonce, then the nonce produced is generated just once in the history. Any other instance of nonce generation, by the same agent, or by another, produces a different nonce. In MSR, this is modeled by using the existential quantifier $\exists$ to produce the nonce.

These assumptions lead to somewhat odd conclusions. Consider, for example, what can be proved about the first message of the protocol. Intuitively, what the responder B should learn from the first message of the protocol is that some agent, possibly the adversary, is attempting to authenticate itself as A. The remaining two messages are designed to convince the responder that this agent is in fact A. The intuitive reason that the first message does not suffice is that it could have been constructed by the adversary. However, according to the (pure) Dolev-Yao adversary model, the adversary can send only messages constructed from what he has been able to extract. If only SNS protocol messages are sent, the adversary is not an insider, and the keys of the agents are not divulged, then the adversary is never able to extract any nonce that he can use to forge a message. Thus, agent B knows that the first message he receives was not constructed by the adversary. (The message could still be a replay of a previous attempt by A to authenticate herself to B.) This seems to be a much stronger conclusion than is warranted by intuitions about the protocol.

Clearly, this limitation of the Dolev-Yao model can be overcome by allowing the adversary to use nonces that it has not been able to extract in order to construct messages. And indeed, most approaches to security protocol analysis allow the adversary to generate nonces that it has not extracted before. But which nonces do

you allow the adversary to generate? MSR (and many other approaches) only allow the adversary to generate nonces that are unique, thereby enforcing the uniqueness assumption on nonces. This is reflected by the rule for nonce generation for the adversary given in Section 6.4.1:

$$\longrightarrow \exists x.C(x).$$

How reasonable is this? The choice has a number of consequences. First, the agents may as well use *sequence numbers* or *timestamps* rather than nonces, since the adversary will never be able to generate a value that already exists. It seems counterintuitive that a given protocol using nonces and the same protocol using sequence numbers would behave similarly in all contexts. The nonce-uniqueness assumption is not sufficient to prove that A generated the message. For example, if the current sequence number is taken to be the value of the nonce, then the nonce is unique, but completely predictable. Thus, a real adversary should be able to forge the message. However, a Dolev-Yao adversary, even augmented with unique nonces, would never be able to do this. To analyze the protocol correctly, it is necessary to be able to assume that the nonces in question cannot be predicted. Unpredictability is an *epistemic* notion. Roughly speaking, it means that the probability that an adversary will know the nonce is low.

As this discussion shows, we cannot sweep these epistemic concerns under the rug if we want to get an appropriate analysis of the SNS protocol. The logic introduced in Chapter 8, extended with probability and time, allows us to capture these notions in what we believe is a reasonable way.

9.2 Temporal and Probabilistic Extensions

There are standard techniques for extending the logic of Section 8.1 with the ability to reason about time and probability. In fact, systems already incorporate time, since they are sets of runs, and runs are functions from time to global states. To reason about probabilities, we consider *probabilistic systems*, where there are probability distributions on the set $\mathcal{R}$ of runs of the system. We do not assume a single probability distribution on $\mathcal{R}$, since that would require a probability on the possible protocols that the adversary is using. Rather, the idea is to partition $\mathcal{R}$ into subsets, called cells; intuitively, each cell corresponds to factoring out all the choices that are best viewed as nondeterministic (such as the choice of protocol, or the keys used by the participants in a protocol). Assume that there is a probability μ_C on the runs $\mathcal{R}_C$ of each cell C.

The logic itself is a straightforward extension of that presented in Chapter 8. Again start with a symbolic encryption scheme. To be able to model BAN, how-

ever, there must be sufficient structure to the space of plaintexts, namely that it includes (representations of) formulas. Thus, assume a new set of plaintext messages $string(\varphi)$, where φ is a formula in the logic. (These propositions are needed because BAN allows formulas to be sent as messages. There is no *a priori* interpretation associated with these representations. They are just strings of characters.)

The syntax of the logic is extended with the formulas $\bigcirc\varphi$, $\ominus\varphi$, $\Box\varphi$, $\boxminus\varphi$, and $\Pr_i(\varphi) \geq \alpha$. The temporal operator $\bigcirc\varphi$ states that φ is true at the next time step, while $\ominus\varphi$ states that φ was true at the previous time step. We use the abbreviations $\bigcirc^l\varphi$ and $\ominus^l\varphi$ (for $l \in \mathbb{N}$) for the l-fold application of $\bigcirc$ and $\ominus$, respectively, to φ. The temporal operator $\Box\varphi$ states that φ is true at the current time and all subsequent times. Similarly, $\boxminus\varphi$ states that φ is true at the current time and all previous times. Finally, $\Pr_i(\varphi) \geq \alpha$ says that the formula φ holds with probability at least α, according to agent i. Define the usual abbreviations, such as $\Pr_i(\varphi) < \alpha$ for $\neg(\Pr_i(\varphi) \geq \alpha)$, and so on.

To give semantics to formulas of the form $\Pr_i(\varphi) \geq \alpha$ at a point (r, t), proceed as follow. Let $\mathcal{K}_i(r, t)$ be the set of points that agent i cannot distinguish from (r, t), that is, the set $\{(r', t') \mid (r, t) \sim_i (r', t')\}$. For every run r, there is a unique cell C_r with $r \in C_r$. Let $\mathcal{C}(r)$ be the set of points from the runs in $\mathcal{R}_{C_r}$, that is, the points in the runs in the same cell as r. The prior probability μ_C on the runs $\mathcal{R}_C$ of cell C induces a probability $\mu_{r,t,i}$ on the points in $\mathcal{K}_i(r, t) \cap \mathcal{C}(r)$ in a straightforward way. If $U \subseteq \mathcal{K}_i(r, t) \cap C(r)$, define

$$\mu_{r,t,i}(U) = \frac{\mu_C(\{r' \mid (r', t) \in U\})}{\mu_C(\{r' \mid (r', t) \in \mathcal{K}_i(r, t) \cap \mathcal{C}(r)\})}.$$

The satisfaction relation of a formula φ in an interpreted algorithmic knowledge security system with probabilities $\mathcal{I} = (\mathcal{R}, \pi, \mu, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ at point (r, t) is extended by the following rules:

$(\mathcal{I}, r, t) \models \bigcirc\varphi$ iff $(\mathcal{I}, r, t + 1) \models \varphi$
$(\mathcal{I}, r, t) \models \ominus\varphi$ iff $t \geq 1$ and $(\mathcal{I}, r, t - 1) \models \varphi$
$(\mathcal{I}, r, t) \models \Box\varphi$ iff for all $t' \geq t$, $(\mathcal{I}, r, t') \models \varphi$
$(\mathcal{I}, r, t) \models \boxminus\varphi$ iff for all $t' \leq t$, $(\mathcal{I}, r, t') \models \varphi$
$(\mathcal{I}, r, t) \models \Pr_i(\varphi) \geq \alpha$ iff $\mu_{r,t,i}(\{(r', t') \mid (\mathcal{I}, r', t') \models \varphi\} \cap \mathcal{K}_i(r, t) \cap \mathcal{C}(r)) \geq \alpha$.

Define the probabilistic knowledge operator $K_i^\alpha\varphi$ as an abbreviation for the formula $K_i(\Pr_i(\varphi) \geq 1 - \alpha)$. This operator simply means that, essentially, no matter which cell C the agent thinks the current point is in, the probability of φ in that cell is at least $1 - \alpha$. It is easy to check that

$$\mathcal{I} \models (K_i^\alpha\varphi \wedge K_i^\beta\psi) \Rightarrow K_i^{\alpha+\beta}(\varphi \wedge \psi).$$

Moreover, if the system $\mathcal{I}$ does not assign probability 0 to any subset of the runs,

then

$$\mathcal{I} \models K_i^0 \varphi \Leftrightarrow K_i \varphi.$$

In other words, if no nonempty subset of runs has probability 0, then knowing with probability 1 collapses to actual knowledge.

9.3 An Interpretation of BAN

One of our claims is that the logic introduced in Section 8.1 and extended in Section 9.2 is a good foundation for security protocol logics. To provide evidence for this claim, we show how we can interpret the constructs of BAN logic by translating them into the simpler primitives of our logic. Doing so exposes many of the assumptions underlying these constructs. Our translation provides evidence that the intuitions underlying authentication logics can be given a rigorous semantic foundation using well-understood modal operators. Although we focus here on BAN, we believe that other related logics could be similarly reconstructed.

The set of formulas F and set of messages m are defined by mutual induction, since formulas are actually a subset of messages, and can include messages in them. Messages are defined essentially as in Section 8.1:

$$m ::= t \mid k \mid n \mid i \mid (m_1, m_2) \mid \{m^i\}_k \mid F.$$

The superscript i in $\{m^i\}_k$ represents a "from"-field, intended to indicate the original sender of the message. Note that BAN logic has an extra message construct, $\langle m^i \rangle_{m'}$, representing a message m combined with a *secret* m'. This is essentially a pairing (m, m'), except for the need to account for the "from"-field i, indicating the original sender of the message. Since we do not use this construct in the remainder of this chapter, and since accounting for it complicates the presentation without adding insight, we do not deal with it here. Formulas are defined by the following grammar:

$$\begin{aligned} F ::= \; & i \; \mathbf{believes} \; F \\ \mid \; & i \; \mathbf{controls} \; F \\ \mid \; & i \; \mathbf{sees} \; m \\ \mid \; & i \; \mathbf{said} \; F \\ \mid \; & i \overset{k}{\leftrightarrow} j \\ \mid \; & \overset{k}{\mapsto} j \\ \mid \; & i \overset{m}{\rightleftharpoons} j \\ \mid \; & \mathbf{fresh}(m). \end{aligned}$$

R1. $$\frac{i\ \mathbf{believes}\ j \overset{k}{\leftrightarrow} i \quad i\ \mathbf{sees}\ \{F^l\}_k \quad l \neq i}{i\ \mathbf{believes}\ j\ \mathbf{said}\ F}$$

R2. $$\frac{i\ \mathbf{believes}\ j \overset{k}{\leftrightarrow} i \quad i\ \mathbf{sees}\ \{F^l\}_{k^{-1}} \quad l \neq i}{i\ \mathbf{believes}\ j\ \mathbf{said}\ F}$$

R3. $$\frac{i\ \mathbf{believes}\ \mathbf{fresh}(F) \quad i\ \mathbf{believes}\ (j\ \mathbf{said}\ F)}{i\ \mathbf{believes}\ j\ \mathbf{believes}\ F}$$

R4. $$\frac{i\ \mathbf{believes}\ j\ \mathbf{controls}\ F \quad i\ \mathbf{believes}\ j\ \mathbf{believes}\ F}{i\ \mathbf{believes}\ F}$$

R5. $$\frac{i\ \mathbf{sees}\ (F, F')}{i\ \mathbf{sees}\ F}$$

R6. $$\frac{i\ \mathbf{believes}\ j \overset{k}{\leftrightarrow} i \quad i\ \mathbf{sees}\ \{F\}_k}{i\ \mathbf{sees}\ F}$$

R7. $$\frac{i\ \mathbf{believes}\ \overset{k}{\mapsto} i \quad i\ \mathbf{sees}\ \{F\}_k}{i\ \mathbf{sees}\ F}$$

R8. $$\frac{i\ \mathbf{believes}\ \overset{k}{\mapsto} j \quad i\ \mathbf{sees}\ \{F\}_{k^{-1}}}{i\ \mathbf{sees}\ F}$$

R9. $$\frac{i\ \mathbf{believes}\ \mathbf{fresh}(F)}{i\ \mathbf{believes}\ \mathbf{fresh}(F, F')}$$

Figure 9.1. BAN inference rules

The intuitive reading of the formulas is as follows. The formula $i\ \mathbf{believes}\ F$ holds if agent i believes formula F. The formula $i\ \mathbf{controls}\ F$ means that agent i is an authority on or has authority or jurisdiction over F. The formula $i\ \mathbf{sees}\ m$ indicates that i has received a message containing m. The formula $i\ \mathbf{said}\ F$ indicates that agent i at some time sent a message containing F and (if it was sent recently) that i believes F. The formula $\mathbf{fresh}(m)$ indicates that the message m is fresh, that is, it was sent recently. The formula $i \overset{k}{\leftrightarrow} j$ means that agents i and j can use key k to communicate (and that k is a good key). The formula $\overset{k}{\mapsto} j$ means that the key k is j's public key (and that k is a good key). Finally, $i \overset{m}{\rightleftharpoons} j$ means that message m is a secret between agents i and j. Most logics in the BAN tradition extend the logic, adding formulas such as $i\ \mathbf{says}\ F$ (i recently said F), and $i\ \mathbf{has}\ k$ (i is in possession of key k). For ease of exposition, we do not consider these modifications here.

BAN uses inference rules to derive new formulas from others. A representative fragment of these rules is given in Figure 9.1. For instance, rule R3 says that if i believes both that F is fresh and that j said it, then i also believes that j believes F. R3 encapsulates the assumptions that statements do not change their truth value in short intervals of time and that agents say only things that they believe to be true.

The BAN operators seem to capture the intuitive notions that arise when reasoning about security protocols: belief, trust, freshness, jurisdiction. The criticisms

aimed at BAN do not argue this point, but rather focus on the lack of semantics and on the protocol verification method. We have an additional criticism, namely, the choice of primitive notions. We believe that there should be relatively few primitive notions, with all the rest defined in terms of them. We believe that the logic presented in the second part of this dissertation overcomes these deficiencies, while still being able to express the key features of the BAN operators.

We now define a translation from BAN formulas to formulas in the logic of Section 9.2. We emphasize, however, that this is not the only translation that captures a useful interpretation of BAN; we discuss variants where appropriate. The translation takes a BAN formula and produces a family of formulas. The formulas in the family differ only in the probability used to determine belief. A formula F^T is a *possible translation of* F if it can be produced by the translation rules below.

Because BAN formulas include messages and are messages, messages also need to be translated; write m^M for a possible translation of message m. The translation of messages that are not formulas is the obvious one: for a primitive message m, $m^M = m$, and $(m_1, m_2)^M = ({m_1}^M, {m_2}^M)$, where ${m_1}^M$ and ${m_2}^M$ are possible translations of m_1 and m_2. Translate encryptions $\{m^i\}_k$ by treating the "from"-field as concatenated to the end of the encrypted message: $\{m^M, i\}_k$, where m^M is a possible translation of m. A possible translation of a formula F (when viewed as a message) is $string(F^T)$, where F^T is a possible translation of F, when viewed as a formula.

Here is the translation of formulas of BAN logic.

– The translation for $\mathbf{believes}$ is based on the assumption that an agent operates with a set of default assumptions, expressed as a formula A. An agent's belief in φ, relative to assumptions A, can then be captured by the formula $K_i(A \Rightarrow \varphi)$. That is, the agent believes φ relative to assumptions A if it knows that φ holds under assumptions A. The translation for the BAN logic expression i $\mathbf{believes}$ F uses this idea, but adds probabilities. The agent's default assumptions are characterized by a set of "good" runs. Intuitively, these are the runs in which undesirable events like the adversary guessing a nonce do not occur. Any set of runs can be taken as the good runs, but one should expect the prior probability of the set of good runs to be high (a fact that can be expressed in the logic). The particular choice of good runs used in proving that a protocol satisfies a BAN logic specification will depend on the details of the protocol and the system used to model the behaviour of the adversary. Let $good$ be a primitive proposition that expresses "the run is good". The possible translations of i $\mathbf{believes}$ F have the form $K_i^\alpha(good \Rightarrow F^T)$, where $0 \leq \alpha \leq 1$ and F^T is a possible translation of F. (Note that $K_i^1\varphi$ is vacuously true for all φ.) As we shall see, the soundness of the translation does not depend on the particular

choice of $good$. In many cases of interest, we can take $\alpha = 0$ (in particular, this is true for our analysis of SNS in Section 9.4); in that case, many details of the translation below can be simplified. If all runs have positive probability (as is as the case in the analysis of SNS), K_i^0 reduces to K_i; that is, we are back to the original definition of defeasible belief.

– The possible translations of $(i \, \mathbf{sees} \, m)^T$ have the form $X_i(has_i(m^M))$, where m^M is a possible translation of m. Here, the knowledge algorithm for agent i is is doing all the work of deciding what information can be extracted from the set of messages received.

– The translation of i **said** F is somewhat complicated, since the **said** operator is conflating a number of distinct notions that must be untangled to express it in our logic. For one thing, it means that a message is sent. However, an agent should not be interpreted as saying F when it sends a message containing F encrypted under a key that the agent does not possess (this situation arises in some protocols that rely on agents to forward server generated tickets.) To capture this intuition, the translation of **said** also incorporates the notion of extraction; nothing is said that was not extracted at the time when the message was sent. Accordingly, take a possible translation of i **said** F to have the form $send_i(F^M) \wedge \boxminus(\neg send_i(F^M) \wedge \bigcirc send_i(F^M) \Rightarrow X_i(has_i(F^M)))$, where F^M is a possible translation of F. The BAN reading of **said** also involves claims about belief; BAN assumes that all formulas said recently by i are believed by i. This assumption is not part of the translation, but is captured in the systems for which the translation is proved sound, by imposing an "honesty" requirement.[1]

– Capturing that k is a good key between i and j depends on what is meant by "good key". There are at least two interpretations. One interpretation is that "no one but i and j sends messages encrypted with k" for the length of the protocol interaction. Another possible interpretation is that no one other than possibly i and j has extracted the key. Both interpretations can be encoded, but the second, while stronger, seems more in keeping with the intuitive reading of k being a good key. Roughly, if a key is leaked to an adversary that never uses it to encrypt messages, BAN would consider the key a good key, despite the adversary being able to read all traffic encrypted with that key. While BAN's precise intention could be captured, it would again complicate things, and it runs somewhat counter to the intuitive reading of "goodness". Accordingly, take

[1] While it suffices for the purposes of this chapter, this translation does not quite capture all the subtleties of BAN's reading of **said**. According to their reading, an agent i should not be considered to have **said** m if it sent $\{m\}_k$, did not possess k at the time, but was nevertheless able to extract m from some other message. According to the interpretation above, i **said** m does hold in this case. While it is possible to come up with an interpretation that was even closer to BAN's (at the cost of complicating the logic), it does not seem worthwhile. Our translation satisfies BAN's axioms and seems to capture the essence of their notion.

$(i \overset{k}{\leftrightarrow} j)^T$ to be

$$X_i(has_i(k)) \wedge X_j(has_j(k)) \wedge \left(\bigwedge_{i' \neq i,j} \neg X_{i'}(has_{i'}(k)) \right).$$

Interpreting "good key" in general is not that simple, unfortunately. Many protocols studied in the literature assume the existence of a key server in charge of distributing session keys to agents. In such a context, a good key is not only known to the agents exchanging messages, but also of course to the server that initially distributed the key. It is easy to accommodate such an interpretation of "good key" by assuming that the server, as well as i and j, can extract the key. For simplicity, however, we will consider only the interpretation of "good key" given above. Note that BAN interprets k being a good key as a statement that also talks about the future; in essence, if k is a good key, it remains so throughout a protocol interaction. Such an interpretation can be captured by prefixing the translated formula by a $\Box$ operator. Of course, this interpretation precludes the analysis of protocols that leak the key value. Ideally, the analysis should reveal such leaks, rather than presupposing that they do not happen.

– $\overset{k}{\mapsto} j$ says that k is j's public key, and that the key is a good key. The formula is intended to mean that only j knows the key k^{-1}. Thus, its translation is similar in spirit to that of the formula for shared keys, and the same comments apply; take $(\overset{k}{\mapsto} j)^T$ to be $X_j(has_j(k^{-1})) \wedge \left(\bigwedge_{i' \neq j} \neg X_{i'}(has_{i'}(k^{-1})) \right)$.
– $i \overset{m}{\rightleftharpoons} j$ says that m is a secret shared by i and j. This is intended to mean that only i and j know m, that is, that only they can extract it. Hence, take a possible translation of $(i \overset{m}{\rightleftharpoons} j)$ to have the form $X_i(has_i(m^M)) \wedge X_j(has_j(m^M)) \wedge \left(\bigwedge_{i' \neq i,j} \neg X_{i'}(has_{i'}(m^M)) \right)$, where m^M is a possible translation of m.
– A message is fresh if it could not have been sent, except possibly recently. It is up to the user to decide what counts as "recently", by choosing a suitable l. Thus, a possible translation of $\mathbf{fresh}(m)$ has the form $\ominus^l \bigwedge_i (\boxminus \neg send_i(m^M))$, where m^M is a possible translation of m. Of course, this translation does not address the issue of what makes a nonce fresh, or how to prove that a nonce is fresh. Intuitively, this is where the unpredictability of the nonce comes in; we return to this issue in Section 9.4.
– We interpret i **controls** F as "i knows F if and only if F is true". Thus, a possible translation of i **controls** F has the form $K_i(F^T) \Leftrightarrow F^T$, where F^T is a possible translation of F. This captures, to some extent, the intuition that i is an authority on F. There is no way for F to change without agent i knowing it, so that F is in some sense "local" to agent i. Our translation, however, while capturing a reasonable consequence of **controls**, does not fully capture the in-

tent of the operator. For instance, the translation seems inappropriate when F^T is a formula for which neither F^T nor $K_i(F^T)$ holds; in this case, i controls F vacuously. A better translation might be: i necessarily knows F^T if and only if F^T is true. While it is straightforward to add a "necessarily" operator to the logic, the overhead of doing so does not seem justified.

To what extent does the translation above capture BAN? The minimum we can ask is that the translation validates the axioms of BAN. This ensures that we capture at least the reasoning underlying BAN. As the following theorem shows, it does provided that we use the appropriate knowledge algorithm, and make appropriate assumptions about agents. Capturing the reasoning is not quite enough however, since a formula and its translation should also have the same meaning. This cannot be made precise, however, since BAN does not provide meanings for its formulas. In Section 9.4, we address this issue by examining the extent to which our translation above validates the conclusions of BAN analyses.

In order to validate the axioms of BAN, we need to make assumptions on the system. Intuitively, these are assumptions that are made implicitly by BAN logic, and which must be made explicit in order to prove the soundness of the translation. In particular, it is necessary to assume that agents have essentially no prior information, that agents tell the truth (since BAN assumes that when a "good" agent sents a formula, it believes it), and adversaries' capabilities are characterized by the Dolev-Yao model. Agents *have no additional prior information beyond guesses* in an interpreted algorithmic knowledge security system $\mathcal{I}$ if the initial states of all agents include only public keys, their own private keys, the nonces required by their protocol (in the case of nonadversary agents), and a finite set of other keys or nonces they have guessed. It is also necessary to make precise the intuition that agents tell the truth, since BAN assumes that when a (nonadversary) agent sends a formula, it believes the formula. Without this requirement, the validity of R3 cannot be ensured. Implicit in the notion of honesty is the idea that an agent does not forge "from"-fields in messages. Furthermore, BAN assumes that agents' capabilities of creating and decomposing messages are those characterized by the Dolev-Yao model. These capabilities, together with the assumption that agents not forge "from"-fields, are captured by providing a suitable knowledge algorithm for the agent. (This is similar to what was done in Section 8.2, except that a knowledge algorithm is not given explicitly; rather, the properties it should have are specified.) Assume there is a function $init(s)$ that, given a local state s, returns the set of messages contained in the initial state. If s is a local state for agent i, define $can\_compute_i(s)$ to be the smallest set M of messages such that $m \in M$ if one of the following conditions hold:

(1) $recv(m) \in s$ or $m \in init(s)$;

(2) there exists m' with $(m, m') \in M$ or $(m', m) \in M$;
(3) there exists a key k (symmetric or asymmetric) with $\{m\}_k \in M$ and $k^{-1} \in M$;
(4) $m = (m_1, m_2)$, with $m_1 \in M$ and $m_2 \in M$;
(5) $m = \{m_1, j\}_k$, with $m_1 \in M$, $k \in M$, and $1 \leq j \leq N$.

Similarly, we can define the "nonforging" version, $can\_compute_i^{\mathrm{NF}}(s)$, by replacing rule (5) by the following:

(5') $m = \{m_1, i\}_k$, with $m_1 \in M$, $k \in M$.

Rule (5') ensures ensures that when the agent constructs an encrypted message, he includes a "from"-field set to its own name. The interpreted algorithmic knowledge security system $\mathcal{I}$ *models agent* i *as a Dolev-Yao agent* if for all runs r in $\mathcal{I}$, for all $t \geq 0$, and for all messages m,

(1) $(\mathcal{I}, r, t) \models X_i(has_i(m))$ if and only if $m \in can\_compute_i(r_i(t))$;
(2) if $r_i(t+1) = r_i(t) \cdot send(i, m)$, then $m \in can\_compute_i(r_i(t))$.

The definition of a *nonforging Dolev-Yao agent* is similar, but uses $can\_compute_i^{\mathrm{NF}}$ for $can\_compute_i$. An agent i is a *γ-honest Dolev-Yao agent* (for $0 \leq \gamma < 1$) in an interpreted algorithmic knowledge security system $\mathcal{I}$ if

(1) i is a nonforging Dolev-Yao agent in $\mathcal{I}$, and
(2) for all φ and all points (r, t), if i sends a message m at round t (i.e., between times $t-1$ and t), $string(\varphi) \sqsubseteq m$ (i.e., φ is a submessage of m), and $(\mathcal{I}, r, t-1) \models X_i(has_i(\varphi))$, then $(\mathcal{I}, r, t-1) \models K_i^{\gamma}(\bigwedge_{l' \leq l} \bigcirc^{l'} \varphi)$.

The intuition for the last condition is that an agent will say only things that he believes will still be true some time in the near future after its message is received. Again, this is parameterized by a time l, which should be taken as the same time parameter used to interpret freshness.

Observe that while the restriction to Dolev-Yao agents is hardwired into the definitions of **said** and **sees** by BAN and later logics, it is modeled using knowledge algorithms in our framework. This means that our framework can be used to deal with other adversaries besides those that satisfy the Dolev-Yao properties, without changing the underlying syntax and semantics. Similarly, rather than hardwiring honesty into the definition of **said**, it is built into the class of structures. It is therefore possible to model the kind of operators BAN advocates without being tied to the particular choices made by BAN and its successors.

The next step is to show that this translation preserves the validity of the BAN inference rules. Making this statement precise requires care, since translations of formulas are not unique. Note that an instance of a BAN inference rule has the

form "from F_1 [and F_2] infer F_3". This instance translates into a set of formulas of the form $F_1^T[\wedge F_2^T] \Rightarrow F_3^T$, where F_i^T is a possible translation of F_i, subject to some consistency conditions:

- The possible translations of rule R6 have the form $K_i^\alpha(good \Rightarrow (i \overset{k}{\leftrightarrow} j)^T) \wedge X_i(has_i(\{F^M\}_k)) \Rightarrow X_i(has_i(F^M))$, where F^M is a possible translation of F. (Of course, the same possible translation of F appears in both the antecedent and the conclusion.) The rules R5, R7, and R8 are translated similarly.
- Rules R1, R2, and R9 all have a $\mathbf{believes}$ formula in their antecedent. The same α is required in the translation of $\mathbf{believes}$ in both the antecedent and the conclusion. For example, the possible translations of rule R9 have the form $K_i^\alpha(good \Rightarrow \mathbf{fresh}(F)^T) \Rightarrow K_i^\alpha(good \Rightarrow \mathbf{fresh}(F, F')^T)$, where the possible translations of $\mathbf{fresh}(F)$ and $\mathbf{fresh}(F, F')$ use the same possible translation F^M.
- For R3, there are two $\mathbf{believes}$ formulas in the antecedent, and one in the conclusion. The possible translations of rule R3 have the form $K_i^\alpha(good \Rightarrow (\mathbf{fresh}(F))^T) \wedge K_i^\beta(good \Rightarrow (j \ \mathbf{said}\ F)^T) \Rightarrow K_i^{\alpha+\beta}(good \Rightarrow K_j^\gamma(good \Rightarrow F^T))$, where the γ is taken from the γ-honesty assumption.
- Finally, for R4, the possible translations have the form $K_i^\alpha(good \Rightarrow (K_j F^T \Leftrightarrow F^T)) \wedge K_i^\beta(good \Rightarrow K_j^\delta(good \Rightarrow F^T)) \Rightarrow K_i^{\alpha+\beta}(good \Rightarrow F^T)$.

Note that if belief is translated using K_i^0 and if $\gamma = 0$ in the definition of γ-honesty (as is natural in many applications), then there is a unique translation where all the superscripts to K_i are 0.

The following theorem, where the notation r_{ij}^T is used to emphasize that the formulas in the translation of r refer to agents i and j, states that the translation preserves soundness.

Theorem 9.1. *Every translation r_{ij}^T of an instance r_{ij} of the BAN inference rule Rn, for $n = 1, 2$, is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses and where agents i and j are nonforging Dolev-Yao agents. Every translation r_{ij}^T of an instance r_{ij} of the BAN inference rule R3 is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses and where agents i and j are γ-honest. Finally, every translation r^T of an instance r of Rn for $n \geq 4$ is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses.*

It follows from the theorem that if $\mathbf{believes}$ is interpreted as "holds with probability 1" (so that i $\mathbf{believes}$ F is translated as $K_i^0(good \Rightarrow F^T)$), then this translation also preserves validity. With this translation, if each run has positive probability, the translation of BAN belief as knowledge follows as a special case.

Thus, our translation generalizes two of the standard interpretations of BAN belief in the literature.

The soundness of our translation is independent of how the primitive proposition $good$ is interpreted. However, it should be the case that the initial probability of $good$ is high. In other words, it should be the case that $(\mathcal{I}, r, 0) \models K_i^{\alpha}(good)$ holds for all runs r, for a small α. Otherwise, the conclusions drawn about a protocol are unlikely to be of great interest. The analysis in the next section illustrates this point.

9.4 An Analysis of the SNS Protocol

We now show how to perform an analysis of the SNS protocol using the logic of Sections 8.1 and 9.2, and illustrate how to justify the conclusions of the BAN analysis of that protocol.

It is straightforward to construct a system $\mathcal{I}^{GDY}$ modeling executions of the SNS protocol. Consider the case of two agents, A and B, and an adversary X. Suppose that A runs the protocol just once. This means that the analysis does not cover replay attacks, but it suffices to illustrate the role of nonces and probability. Intuitively, for every choice of n_A and n_B (the nonces for A and B), k_A and k_B (the public keys of A and B), and for every choice of a deterministic protocol P (compatible with the Dolev-Yao capabilities) for the adversary, $\mathcal{I}^{GDY}$ contains a run where agent A uses nonce n_A and private key k_A^{-1}, agent B uses nonce n_B and private key k_B^{-1}, and where the adversary starts with the knowledge of the public keys, runs protocol P, and guesses n_A^X and n_B^X for the nonces that A and B use. For simplicity, assume that the key space is such that $k_A \neq k_B^{-1}$ if $k_A \neq k_B$ (that is, one key cannot be the inverse of another) and the space $\mathcal{N}$ of nonces is finite. Let $\mathcal{I}^{GDY}_{P,n_A^X,n_B^X}$ be the subsystem consisting of all runs in which the adversary uses protocol P and nonces n_A^X and n_B^X. (To simplify the presentation, assume that the adversary only attempts to guess a single nonce per agent. It is easy to extend our results to the more general case of a finite number of initial guesses.)

Take the adversary's choice of protocol and nonces to be nondeterministic. In other words, assume that the adversary has complete freedom of choice (unconstrained by any distribution) concerning which attack he is going to mount. In order to ensure that nonces are unpredictable, we take the agents' choice of nonce to be random. The agents use these random choices to protect themselves against the unknown but fixed adversary. Formally, define a distribution on the runs of each subsystem $\mathcal{I}^{GDY}_{P,n_A^X,n_B^X}$, by taking the values n_A and n_B to be uniformly distributed.

Consider the specifications of the SNS protocol given in the original BAN analysis of this protocol. (Given our simplification of the protocol, only the specifi-

cations that do not involve the server are relevant.) Let $F(n_A, n_B, k_A, k_B)$ be the conjunction of the following formulas, where n_A and n_B are nonces, and k_A and k_B are keys:[2]

$$A \textbf{ believes } \stackrel{k_B}{\mapsto} B$$
$$B \textbf{ believes } \stackrel{k_A}{\mapsto} A$$
$$A \textbf{ believes } A \stackrel{n_A}{\rightleftharpoons} B$$
$$B \textbf{ believes } A \stackrel{n_B}{\rightleftharpoons} B$$
$$A \textbf{ believes } B \textbf{ believes } A \stackrel{n_B}{\rightleftharpoons} B$$
$$B \textbf{ believes } A \textbf{ believes } A \stackrel{n_A}{\rightleftharpoons} B$$
$$B \textbf{ believes } A \textbf{ believes } B \textbf{ believes } A \stackrel{n_B}{\rightleftharpoons} B.$$

$F(n_A, n_B, k_A, k_B)$ is the conclusion that BAN would like to reach for a run of the protocol where k_A is A's key, k_B is B's key, n_A is A's nonce, and n_B is B's nonce.

The goal is to show that the (translated) conclusions hold at points in $\mathcal{I}^{GDY}$ where the protocol has successfully run to completion. In order to do this, it is necessary to define the notion of a good run. Intuitively, a run is good if the adversary has not guessed the correct nonce n_A or n_B for A or B respectively, and if $n_A \neq n_B$, that is, if A and B choose different nonces. Define $good$ to be true at a point (r, t) if and only if the adversary has not correctly guessed n_A and n_B, and $n_A \neq n_B$ on run r. (Thus, $good$ has the same truth value at all the points in a given run.) Because the agents choose their nonces at random, and because the adversary uses a deterministic protocol, which is independent of the choice of nonce by the agents, the prior probability of the adversary being able to guess the right nonces is low. In other words, the unpredictability of the nonces ensures that $good$ is likely to be true. Formally, for all runs r, $(\mathcal{I}^{GDY}, r, 0) \models K_A^{\alpha}(good) \wedge K_B^{\alpha}(good)$, where $\alpha = 3/|\mathcal{N}|$. Moreover, on good runs, messages with nonces in them must be fresh. Since nonces are unpredictable, the only way that an adversary could have generated a message with a nonce is by guessing it, and this is precisely what does not happen in good runs.

As this discussion suggests, on good runs, all the conclusions of interest hold, so we can take belief to hold with probability 1 in the translation of the conclusions of the BAN formulas. Let $F^{T,0}$ be the possible translation F^T where every occurrence of $i \textbf{ believes } G$ is interpreted as $K_i^0(good \Rightarrow G^{T,0})$. Since every run in $\mathcal{I}^{GDY}$ has positive probability. we can replace K_i^0 by K_i; that is, we can interpret BAN belief

[2] BAN logic does not have conjunction, but it is convenient here to assume it does.

as knowledge. The following result shows that our translation of the last section validates the goals of authentication for SNS as given by BAN.

Theorem 9.2. *If r is a run where A's key is k_A, B's key is k_B, A's nonce is n_A, and B's nonce is n_B, then*

$$(\mathcal{I}^{GDY}, r, 0) \models \Box(recv_B(\{n_B\}_{k_B}) \Rightarrow (F(n_A, n_B, k_A, k_B))^{T,0}).$$

Theorem 9.2 helps elucidate the role of unpredictability for nonces in SNS. Intuitively, the BAN beliefs are justified whenever a run is good. Unpredictability of the nonces translates into the fact that the probability of a run being good is high. Hence, the BAN beliefs are justified with high probability, assuming unpredictability of the nonces. By way of contrast, suppose that nonces are chosen as sequence numbers. The good runs are still the ones where the adversary does not guess the nonce. However, now there will be a protocol for the adversary where he takes the nonce to be the sequence number; with this protocol, the adversary guesses the nonce correctly. As a consequence, there is no $\alpha < 1$ such that $(\mathcal{I}^{GDY}, r, 0) \models K_A^\alpha(good) \wedge K_B^\alpha(good)$, since $K_i^\alpha(good)$ holds initially only if i believes that, no matter what protocol the adversary uses, the set of good runs with that protocol has probability at least $1 - \alpha$. While Theorem 9.2 still holds in this setting, it is no longer that interesting, since the set of good runs is not guaranteed to have high probability.

We stress that our result shows the original version of SNS (without the change described in Section 1.2 to prevent the insider attack) to be correct in certain settings, rather than showing that it does not work in other settings. These results are established by checking that the appropriate formulas hold at appropriate points of the model; There was no use of protocol idealization. Rather, a system was obtained directly from the original description of the protocol.

Notes

The work in this chapter, joint with Halpern and Meyden, is as yet unpublished.

The issue of nonce freshness and unpredictability is somewhat related to whether or not nonces should be kept secret. In some protocols, the nonce need not be kept secret, in which case it may as well be a timestamp. Guttman and Thayer [2002] discuss some of these issues. Different notions of freshness are discussed by Syverson and Meadows [1996].

Temporal logic has its roots in the philosophy of language [Prior 1957]. For the past few decades, it has been used to reason about temporal properties of sequences of events, or sequences of states in computer systems and software [Pnueli 1977; Gabbay, Pnueli, Shelah, and Stavi 1980]. One of the first logics for reasoning

about probabilities is due to Nilsson [1986]. The approach taken here, to view probability as a modal operator, is due to Fagin, Halpern, and Megiddo [1990]. The semantics for $\Pr_i$ is due to Halpern and Tuttle [1993]. While all subsets of points are measurable given the definitions in Section 9.2, the framework of Halpern and Tuttle [1993] makes certain set of points unmeasurable.

The syntax of BAN logic in this chapter is along the lines of the reformulation given by Abadi and Tuttle [1991].

The definition of defeasible belief on which our interpretation of BAN belief is based is due to Moses and Shoham [1993]. Abadi and Tuttle [1991] also define belief based on a set of good runs, but we differ from them in the way that the set of good runs is obtained. They define the good runs by a complicated fixpoint construction based on the original set of beliefs ascribed to the agents by the BAN analysis.

The interpretation of $\mathbf{sees}$ by providing a way to extract information from a message has been advocated before [Wedel and Kessler 1996; Dekker 2000], however, without an algorithmic interpretation.

The interpretation of good key as "no one but i and j sends messages encrypted with k" is advocated by Abadi and Tuttle [1991]. The fact that interpreting good key as a statement that talks about the future means that BAN cannot capture key leaks is discussed by Nessett [1990] and Burrows, Adadi and Needham [1990b]

Abadi and Tuttle [1991] interpret $\mathbf{controls}$ by taking only the forward direction. More precisely, they define i $\mathbf{controls}$ F as: i says F^T implies F^T. The fact that they focus on i saying F in order to control F, as opposed to simply knowing F, is orthogonal to the main concerns of $\mathbf{controls}$.

To account for the fact that an agent may forward a formula without necessarily believing it, Abadi and Tuttle [1991] introduce a special notation (quotation) for forwarded message; this does not remove the need for honesty, in that the original sender of the formula still must have believed the formula before sending it. For ease of exposition, We did not deal with forwarded messages here; they do not cause any difficulties. The fact that BAN requires honesty was pointed out by Abadi and Tuttle [1991]; they avoid the need for honesty by replacing R3, the only rule for which honesty is essential, by other rules.

The Needham-Schroeder protocol was analyzed in the original paper on BAN logic [Burrows, Abadi, and Needham 1990a].

10
Conclusion

THIS dissertation initiated a study of the applicability of theories of resource-bounded knowledge to the problem of reasoning about security protocols. Our approach consisted of investigating a theory of resource-bounded knowledge in some generality and examining its applicability to the particular domain of security protocol analysis. By way of conclusion, we now review what has been achieved, and point out some of the most interesting directions in which this work can be extended.

10.1 Algorithmic Knowledge and Evidence

The first part of this dissertation aimed at establishing that it was possible to develop a logic for resource-bounded knowledge. Has this goal been reached? Starting with the existing framework of algorithmic knowledge, it is fair to ask what it actually means to develop a reasonable logic of resource-bounded knowledge. The first desideratum is expressive power. We would like a framework that allows us to model and reason about all the relevant features of a security protocol. The general framework of algorithmic knowledge was show to be quite expressive, especially once extended to deal with randomized knowledge algorithms.

A theme that emerges from this development is that, aside from giving us a specification language that can talk about both implicit and explicit knowledge, the framework lets us naturally describe the epistemic properties of knowledge algorithms. If we view knowledge algorithms as computing approximations of knowledge, the logic can be used to make precise the conditions under which algorithmic knowledge yields knowledge. For instance, in Chapter 3, we described deductive knowledge algorithms. Studying the use of such algorithms to approximate knowledge amounts to studying the epistemic content of answers to queries to deductive databases. Another example is given in Chapter 4, where we examined reliable

knowledge algorithms. Evidence helps formalize what can be learned from such knowledge algorithms, that is, what the epistemic content of the answers given by reliable knowledge algorithms is. An interesting question is whether this approach could help shed light on probabilistic deductive databases. These databases are deductive databases with probability weights on their inference rules. Most of the issues surrounding probabilistic deductive databases are not settled, partly because it is not clear how to interpret the probability weights. The algorithmic knowledge framework could help clarify what one learns from a probabilistic deduction.

Another theme that emerges is the importance of evidential reasoning. As we already pointed out in Chapter 5, evidence arises naturally in the presence of probabilistic and nondeterministic choices. There has been a lot of work on formal frameworks for modeling stochastic concurrent systems; these systems exhibit both nondeterministic and probabilistic behaviour. We believe that evidence can help write down more precise specifications in those frameworks. For example, consider a system with a number of possible executions, and in every execution, there is an associated probability of event A occurring. Most frameworks can only handle specifications of the form "the probability of event A is at least $1/2$", meaning that for every execution, the probability of event A on the execution is at least $1/2$. A notion of evidence can help tie the occurrence of event A with an associated hypothesis.

It seems that a deeper study of the notion of evidence is required. The notion of evidence space and weight of evidence developed in Chapter 4 is not quite sufficient in general. For instance, the probability of an observation often does not depend only on whether or not an hypothesis is true, but of other aspects of the world, which the evidence space cannot take into consideration. An example of this was already given in Chapter 4, where the probabilistic behaviour of the Rabin algorithm depends on the actual natural number tested for primality, and not just whether or not the number is prime. An obvious approach would be to use a *set of probability measures* rather than a single probability measure for the likelihood of an observation given any particular hypothesis. Such a set of probability measures represents the uncertainty as to the actual probability of observation. As far as we can tell, no measure of evidence has been defined in the literature based on sets of probability measures. There are obvious questions with such a generalization. For instance, is it the case that the notion of weight of evidence that arises can be interpreted as a function from priors to posteriors, and if so, what are the properties of such a function? It is also possible to generalize the notion of evidence to represent the uncertainty of an observation using one of the many existing representations of uncertainty. Again, how to interpret evidence is an intriguing question.

10.2 Security Protocol Analysis

In the second part of the dissertation, we examined in detail a particularly appropriate area of application of algorithmic knowledge. In security protocol analysis, one studies protocols in the presence of an adversary that uses the knowledge he gains through intercepting messages and possibly interacting with other agents to learn secrets and perform other actions. To get meaningful results, we should consider the knowledge that the adversary can compute. This knowledge can be captured in a natural way using algorithmic knowledge, where the knowledge algorithms capture restrictions on the capabilities of an adversary.

It is fair to ask at this point what can be gained by using this framework. For one thing, we believe that the ability of the framework to describe the capabilities of the adversary will make it possible to specify the properties of security protocols more precisely. We also believe that an epistemic language is a natural language for the specification of security protocols. We argued in Chapter 9 that many notions underlying protocol analysis are essentially epistemic. Moreover, many security properties are intrinsically epistemic: confidentiality specifies information that agents should not know, while authentication involves an agent knowing the origin of a message, or the identity of another agent. This is not a new observation—many informal arguments are based on such an epistemic formulation. Having an epistemic language lets us make this formulation explicit.

An obvious question that arises is whether other security properties can be cast naturally in this epistemic framework. More general forms of confidentiality can be captured, including anonymity, which can be which viewed as a form of confidentiality with respect to the identity of the agents. *Fairness*, the property that no agent can gain an advantage over other agents by misbehaving in a protocol, can sometimes be expressed epistemically, when this advantage amounts to obtaining information about another agent. (This is the case in fair exchange protocols, where two agents exchange one item for another, and where fairness ensures that either each agent receives the item it expects, or neither receives any information about the other's item.) *Non-repudiation*, the property that the sender of a message should not be able to deny sending the message, can also be given an epistemic reading: the receiver of a message knows that the sender of the message actually sent it (and is able to convince other agents of this fact). One could also view non-repudiation as stating that the receiver of a message has enough evidence to establish that the sender in fact sent the message. It is interesting to speculate whether it is possible to quantify this kind of evidence using techniques described in Chapter 5.

The approach to protocol analysis presented in the second part of this dissertation is a general framework for handling different adversary models in a natural way. With this framework, it should be possible to provide a formal foundation for

new attacks that are introduced by the community. We gave concrete examples of this in Chapter 8. This can be pushed much further, while remaining in a purely symbolic setting. Many protocols require operations on the "message space" beyond the simple encryption and pairing. For instance, it is straightforward to model operations such as xor ($\oplus$). This requires reasoning under the assumption that $x \oplus y \oplus y = x$. Other operations can be similarly handled.

There are at least two topics that especially deserve further investigation. The first topic is *verification*. While we have focussed on modeling protocols and developing a logic in which it is possible to specify the capabilities of adversaries and reason about the knowledge of agents, we have yet to address the problem of automatic verification of properties expressed using the language. Two approaches are worth pursuing. The first is model-checking. While we do not expect that general model-checking techniques for arbitrary knowledge algorithms can be developed, it may well be possible to extend current techniques to handle more restricted adversaries (for example, Dolev-Yao extended with random guessing). Another approach is to reason about protocols at the language-level, that is, at the level of IMPSEC (or any other language for protocols that can be given a semantics in terms of security systems.) It may be possible to borrow techniques from the study of programming languages, process calculi in particular, to analyze a class of properties expressible in the logic. For example, a type-based analysis on IMPSEC programs could be used to establish confidentiality properties.

The second topic deserving further investigation is to move beyond symbolic analysis, and reason about *computational properties of the encryption schemes*. We suspect that the framework in the second part of this dissertation can be useful for capturing more computational approaches to security protocol analysis. Computational approaches are characterized by reasoning about encryption schemes with a probability of distinguishing distinct encrypted messages (among other things). This probability is defined in terms of a *security parameter* (for instance, the key length) that captures the "hardness" of the encryption scheme. For example, a encryption scheme is *semantically secure* if, intuitively, given two equal length messages m_1 and m_2, an adversary given an encryption of a random one of them cannot tell which it was with a probability significantly better than that of guessing. The adversary, in this context, can perform any probabilistic polynomial-time computation on the messages in order to try to distinguish the two messages. Thus, rather than having a fixed number of capabilities corresponding to abstractions of the operations of the underlying encryption scheme, the capabilities are arbitrary, but restricted by computational limitations. To account for this, it is necessary to reason about the probability of events parameterized by a security parameter and about classes of knowledge algorithms, for instance, the class of all probabilistic polynomial-time knowledge algorithms. In a computational setting, confidential-

ity is taken to mean not that the adversary does not derive any information from the encrypted messages. Thus, the specifications still have an epistemic flavour. Moreover, since this notion of confidentiality refers to information that can be explicitly derived, it should be possible to capture it using algorithmic knowledge. Of course, it may be the case that to prove correctness of a security protocol with respect to certain types of adversaries (for example, polynomial-time bounded adversaries), we will not be able to do much within the logic—we will need to appeal to techniques developed in the cryptography community. However, modeling computational approaches using algorithmic knowledge provides the hope for a truly general specification language that bridges both the symbolic and computational approaches to security protocol analysis.

Notes

Representative samples of recent work in the literature on probabilistic deductive database literature include [Lukasiewicz 1999; Lakshmanan and Sadri 2001]. Using sets of probability distributions is a common approach to capture uncertainty [Huber 1981; Kyburg 1974; Levi 1980]. A good overview of different measures of uncertainty and associated properties is given by Halpern [2003].

The importance of reasoning about xor was made clear by Ryan and Schneider [1998], who showed that a prior analysis by Paulson [1997] on a recursive authentication protocol was flawed when these properties of the protocol related to the use of xor were completely abstracted away. Related extensions to the basic symbolic approach include multiplication and Diffie-Hellman exponentiation; see [Millen and Shmatikov 2003] for more detail.

Meyden and Su [2004] have developed a tool to model-check some forms of security protocols using epistemic logic, although not security protocols involving encryption.

Goldreich [1998, 2001] provides excellent overviews and pointers to the literature of computational approaches to security protocol analysis and cryptography. Semantic security is defined by Goldwasser and Micali [1982]. A standard model for protocol analysis in a computational framework is due to Bellare and Rogaway [1993]. Impagliazzo and Kapron [2003] have developed a logic for reasoning about cryptographic notions by capturing the notion of indistinguishability of probability distributions parameterized by a security parameter.

Bridging the gap between symbolic and computational approaches has been the focus of recent work in security protocol analysis. Many of the approaches described in Chapter 6 have been extended (in a somewhat *ad hoc* manner) to deal with cryptographic assumptions. For instance, Lincoln et al. [1998] introduce a

probabilistic process calculus along the lines of the spi calculus. They establish formal relationships between security expressed using bisimulations in this calculus and more computational definitions of security based on computational indistinguishability. Similarly, Guttman et al. [2001] introduce a quantitative version of strand spaces to model cryptographic assumptions.

A distinct thread of recent work, originating with Abadi and Rogaway [2002], tries to justify the abstractions used by symbolic approaches with respect to more computational assumptions, by examining properties of the encryption scheme sufficient to ensure that a formal analysis in terms of symbolic encryption schemes yields correct results.

Appendix A
On the Problem of Human Knowledge

In the 1960's, Hintikka gave the now standard possible-worlds interpretation of knowledge.[1] In this interpretation, there are different possible worlds (or state of affairs), some of which an agent considers as possible alternatives to the actual world. The agent then knows P if P is true at all the worlds the agent considers as possible alternatives to the actual world.

However, the usefulness of this interpretation is somewhat limited by what Hintikka called the *logical omniscience problem*. Roughly speaking, logical omniscience is a 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. Generally, this takes the following form: if an agent knows P, and P logically implies Q, then the agent also knows Q. A consequence of this is that if an agent knows P, then he knows all Q that are logically equivalent to P.[2]

In many fields, the above notion of knowledge has proved useful despite the phenomenon of logical omniscience. For instance, it has been shown to be appropriate for reasoning about the knowledge of processes in a distributed system, or as a way to capture the notion of "information set" available to an agent in game theory.[3] In all of these setting, knowledge is ascribed by an external observer of the system under consideration.

On the other hand, how do we reconcile such a theory of knowledge, that implies

[1] See Hintikka [1962].

[2] There are in fact many closure conditions that go under the heading of logical omniscience. The general form is as follows: an agent is *fully logical omniscient* if whenever he knows all the formulas in a set Ψ, and Ψ logically implies φ, then the agent also knows φ. Logical implication is always defined with respect to a class of models for a logic under consideration. Roughly, φ logically implies ψ if whenever φ is true, ψ is true. In classical logic, this is equivalent to the validity of material implication: $\varphi \Rightarrow \psi$ is valid. In different logics, however, these notions may differ. See Fagin *et al.* [1990] for more details.

[3] Uses of knowledge in the distributed computing literature include Dwork and Moses [1990] and Moses and Tuttle [1988]. For a good overview, see Fagin *et al.* [1995]. The game-theoretic use of knowledge goes back to the seminal work of Aumann [1976]. Interestingly, the problem of logical omniscience also appears in game theory under the guise of rationality assumptions that are difficult to justify. See Rubinstein [1998] for more details on rationality and bounded rationality.

that agents are logically omniscient, with the fact that humans are clearly not? As Stalnaker puts it,

> It is obvious that if belief and knowledge are understood in their ordinary sense, then no nonsupernatural agent, real or artificial, will be logically omniscient. Despite this obvious fact, many formal representations of states of knowledge and belief, and some explanations of what it is to know or believe, have the consequence that agents are logically omniscient. [Stalnaker 1991]

There are two common and contrasting ways to account for this reconciliation. The first is to take that notion of knowledge as one for ideal knowers, or represent an idealized notion of knowledge. The other is to view this notion of knowledge as a normative notion, describing the ideal to be attained.[4]

Most of the formal representations of knowledge, including the one presented here, make assumptions on the nature of propositions, or on the objects of knowledge. A particularly simple view is simply to take propositions as sets of possible worlds (a proposition represents a way a world might be, and we identify a proposition with the set of worlds that are that way), and take objects of knowledge to be propositions. This is the so-called *coarse-grained* view of content.[5] This view has the advantage of being natural. To know P is to know that the world is a certain way, which way is given by the proposition P. Since the proposition P is a set of possible worlds, an agent knows P if the set of worlds he considers as possible alternatives to the actual world is a subset of the worlds in P. The problem is that such a notion of knowledge intrinsically suffers from logical omniscience.

To illustrate why, consider the following example. Suppose P is a proposition, or set of possible worlds. Suppose moreover that P logically implies Q, that is, at every world that is a way represented by P, the world is also a way represented by Q. Then, when viewed as sets of possible worlds, $P \subseteq Q$. Now, if the agent knows P, then the set of worlds he consider as possible alternatives to the actual world is a subset of P, hence a subset of Q, and thus the agent knows Q. This holds irrespectively of how complex the relationship between P and Q is.

Observe that a form of logical omniscience arises for most propositional attitudes. Say that Oscar convinced Alice to make a bet so that Alice gets money if P is true, but loses money if Q is false. Unbeknownst to Alice, P and Q are logically equivalent, maybe through a complex logical manipulation. Just like Alice could know that P without knowing that Q, Alice can certainly hope that P without hoping that Q. In decision theories that are based on a coarse-grained view of propositions, the fact that one could accept such a bet would make one

[4] There is a sense in which Hintikka's [1962] view is normative; he interprets $K\varphi$ to mean that φ is knowledge (or belief) that is *defensible*. Cresswell [1973, p. 47, footnote 61] is more forward, calling this notion of knowledge *rational*, indicating it is the kind of knowledge that a rational agent should possess.

[5] See Stalnaker [1984]. It is commonly adopted by philosophers of a logical persuasion, such as Cresswell [1973].

irrational, but just as logical omniscience seems unreasonable when representing human knowledge, it seems unreasonable in the above example as well.

There is a general way to understand this. In most cases of interest, propositional attitudes describe the attitude of the agent towards the world being a certain way (belief that the world is a certain way, knowledge, hope, fear, and so on). But there can be many ways of describing such a world, all equivalent, and the agent will realistically only be aware of some such ways. With this view, it is clear that any propositional attitude is subject to a form of logical omniscience if propositions are to be the object of propositional attitudes, and propositions are sets of possible worlds.

Hence, it appears that taking propositions as sets of possible worlds cannot avoid the logical omniscience problem. A possible response is to take a more fine-grained view of the content of propositional attitudes, such as sentences in some language of thought.[6]. Stalnaker argues that this does not solve the problem, by examining a particular proposal, the *sentence storage* model of belief.[7] (While this model applies to belief, it can be straightforwardly be used for knowledge.) This model is particularly simple: every agent has a "belief box" that holds the sentences that the agent believes. Here, the objects of belief are sentences, not propositions. This allows us to make a direct distinction between what we might term implicit and explicit belief: a belief is explicit if its sentence is stored in the box, while it is implicit if it is not. The claim here is that the notion of explicit belief does not suffer from logical omniscience, and hence more accurately captures the notion of belief that real agents have. For instance, while the sentence P might logically imply Q, if P is in the belief box and Q is not, then the agent explicitly believes P, but does not explicitly believe Q.

However, argues Stalnaker, equating real belief with explicit belief is not quite the identification we want to make. On the one hand, we do not want trivial consequences of beliefs in our belief box to clutter the belief box, yet these trivial consequences ought to be as explicit as the sentences actually stored in the belief box. What one really wants here is a notion of a belief being *accessible* from the beliefs stored in the belief box. Accounting for such a notion of accessibility of belief is essentially the same problem as accounting for a notion of belief that does not suffer from logical omniscience.[8] Clearly, if we allow for any logical consequence of the content of the belief box to be accessible, we simply recover the

[6] See Fodor [1976].

[7] See Harman [1973] or Cherniak [1986].

[8] Stalnaker [1984, p.73] admits that in the case of mathematical propositions, the objects of knowledge may have to take the presentation of the proposition into account, in an attempt to circumvent the fact that mathematical propositions are necessarily true or false. In Stalnaker [1987], he examines the view that objects of knowledge and belief are always propositions, by allowing the *that*-clause of a knowledge or belief attribution to not always represent the same proposition all contexts, via his notion of propositional concept.

standard notion of belief, closed under logical consequence, and thus subject to logical omniscience. Therefore, we need to define a notion of accessibility that is more restricted, yet can capture obvious (or reasonable) consequences of the content of the belief box.

In this chapter, we will try to argue for a way to define knowledge to retain as much as possible the coarse-view of content, while avoiding logical omniscience. The intuition is in fact taken from the belief box approach, and relies on taking a more explicitly cognitive view of the knowledge attribution process. We describe a notion of knowledge by taking into account the "mechanisms" in which agents can obtain their knowledge. More precisely, we will assume a representation of propositions in the mind of the agents. We then describe mechanisms in which a sentence of the language can be, for an agent, associated with a particular representation of the proposition that the sentence expresses. We maintain the view that the object of knowledge is a proposition, but allow for the agent not to be able to actually derive a representation of the proposition corresponding to that particular sentence.[9] This difficulty on the part of the agent in deriving a representation of the proposition expressed by a sentence is the reason why this approach does not *a priori* suffer from the logical omniscience problem.

The above account requires some precisions. First, what do we mean by a representation of propositions in the mind? A proposition is a set of possible worlds. Note that there are uncountably many propositions, at least, in any reasonable account of possible worlds. We will restrict representations to be effective, and hence will take the set of representations to be countable (in fact, recursive). So, we seem to be limited in the number of propositions we can represent. We shall argue that this is a reasonable restriction for any natural agent. Second, going back to Stalnaker's quote, accounting for a reasonable notion of knowledge for nonsupernatural agents presupposes that we can specify what make an agent nonsupernatural for the purposes of attributing knowledge. As we shall argue, this can be understood computationally, that is, what differentiates a supernatural agent from a nonsupernatural agent is that the latter is subject to inherent computational limitations.

A.1 Impossible Worlds

Let us first examine an idea that has been advocated to deal with the logical omniscience problem, while remaining within a possible-worlds framework. This account has proved popular in some communities, mostly because it allows for a technically flexible solution to the logical omniscience problem. However, the philosophical underpinnings seem shaky.

[9] Such an approach is alluded to in Stalnaker [1984, p.72], but not developed.

The idea is to consider, along with the familiar possible worlds, so-called *impossible worlds*, worlds where the laws of logic do not hold, where contradictions occur.[10] Perhaps the clearest account of impossible worlds is given by Cresswell, who calls them *non-classical worlds*. The idea is easily summarized:

> A way out is to divide the worlds into those we may call 'classical' and those we can call 'non-classical'. Two propositions are logically equivalent if and only if they contain the same classical worlds, though they may differ in the presence or absence of certain non-classical worlds. [Cresswell 1973, p.40]

The intuition behind non-classical worlds is that, for instance, contradictions may hold at a non-classical world. As Cresswell points out, it is certainly possible to make sense of non-classical worlds in a purely formal setting, without questioning what those worlds are. Many such formal semantics of knowledge have been proposed based on non-classical worlds, without addressing the metaphysical status of those worlds.

A standard argument against such a presentation is that it leads to a contradiction.[11] Let us say that a proposition P is true at a world if the world is a way represented by P. Suppose a non-classical world w in which both P and $\neg P$ are true. Consider a modal operator 'in w', such that 'in w, P' is true at a world if and only if P is true in world w. Hence, 'in w, P' and 'in w, $\neg P$' are both true in the actual world. But 'in w, $\neg P$' is logically equivalent to '$\neg$ in w, P'. So a contradiction is true at the actual world, which it is fair to presuppose, is a classical world, not non-classical.

Cresswell's response to this argument is to actually take non-classical worlds as those where the logical operators (negation, conjunction, and the likes) do not have their classical truth-tables. The idea is to say that if $P \wedge \neg P$ is true at a non-classical world, it is not because P and $\neg P$ are truly contradictory, but simply that $\wedge$ and $\neg$ do not have their usual interpretation. An important point of Cresswell's view is that while his aim is to define propositions in such a way that they can describe, for instance, contradictory states of affairs (or states of affairs where $P \vee \neg P$ is not true), sets of possible worlds cannot in fact describe contradictions.

To achieve this, he defines a proto-proposition to be a set of possible worlds. Accordingly, proto-propositions are essentially what we called 'propositions' until now. These cannot be contradictory, and satisfy the usual rules of classical logic. A set of proto-position forms a heaven, which plays the role of a world, except that it should be thought of as a state of affairs that need not be consistent with the laws of logic. A proposition is finally defined as a set of heavens.

[10] Proponents of this approach include Cresswell [1972], Hintikka [1975], Rantala [1982], Rescher and Brandom [1979], and Wansing [1990]. Kripke [1965] talks about non-normal worlds, which can be considered a precursor to impossible worlds.

[11] See Stalnaker [1996], for a discussion of this argument.

One attractive feature of the impossible-worlds approach, from a formal point of view, is its flexibility. Indeed, it is possible, formally, to add impossible worlds that provide a counterexample to any specific logical equivalence. Returning to our initial example, suppose P and Q are propositional formulas such that P logically implies Q. This means that at every (classical) world, if P is true, then Q is true as well. On the standard account of knowledge, this yields that if an agent knows P, then he knows Q. Consider adding an impossible world that the agent considers as a possible alternative to the actual world to the model, a world where P is true, but Q is not. This is an impossible world, since it does not obey the logical law that if P implies Q, then Q is true whenever P is true. In this extended model, the agent knows P, since P is still true at all worlds he considers as possible alternatives to the actual world, but does not know Q, since there is a world, namely the impossible one, that he considers as a possible alternative to the actual world and where Q is not true.

This showcases the flexibility of the approach. However, it also raises the question of how one chooses the impossible worlds to add to any particular model. This question has rarely been addressed in the literature.[12] Adding impossible worlds to break specific equivalence seems *ad hoc*, and does not help explain which equivalences should be broken. Adding all impossible worlds breaks all equivalences, and does not begin to explain why some equivalences are indeed known. Finally, the flexibility of the framework comes at the cost of the intuitive view of propositions as sets of possible worlds. Even Cresswell admits:

> It would perhaps have been nicer to remain with propositions as sets of possible worlds. ... The more complex analysis seems needed only when we have functors which represent 'propositional attitudes', ... A full justification of our procedure would need a far deeper semantical analysis of these notions that has yet been given. [Cresswell 1973, p.47]

As we advocate in the remainder of this chapter, we can recover the intuition underlying impossible worlds (that there are situations that are impossible but are not recognized as such by agents) by taking a more cognitive view of the knowledge attribution process.

A.2 Awareness and Mental Representations

While the notion of impossible worlds allows for a formally elegant solution to the logical omniscience problem, it remains difficult to account for the metaphysical status of those worlds. In this section, we describe a different approach to the prob-

[12] Rantala [1975] gives an approach based on urn models. Impossible worlds have been used in decision theory to model bounded rationality by Lipman [1999], who advocates a particular construction of impossible worlds.

lem of logical omniscience that remains compatible with the view of propositions as sets of possible worlds.

The idea is to consider the knowledge attribution process as a cognitive one. Roughly speaking, to know a fact is to be able to derive a representation of the proposition capturing that fact, and to have that proposition true at all worlds that the agent considers as possible alternatives to the actual world. Thus, we retain the view that propositions are sets of possible worlds, but now capture the limited reasoning process of the agent by restricting what representations the agent can handle.

We can illustrate the issue as follows. Recall the problematic consequence of logical omniscience: if Alice knows P, and P is logically equivalent to Q, then Alice knows Q. As many people have noticed, this is particularly problematic in the context of mathematical expressions. To avoid dragging in issues of mathematical foundations, consider a simple class of sentences, that suffices to exhibit the problem of interest:

Alice knows that φ is a valid formula of classical first-order logic.

Here, φ ranges over formulas of classical first-order logic. Clearly, for any φ, the sentence $\ulcorner\varphi$ is a valid formula of classical first-order logic$\urcorner$ is either the necessarily true proposition, or the necessarily false proposition. The standard account of knowledge simply says that Alice knows all such sentences where φ is indeed a valid formula of classical first-order logic. This includes such uncontroversial formulas such as *true*.

What happens, however, if Alice does not know any classical first-order logic? Even if we assume that Alice is a competent English speaker, nothing warrants that she knows what a valid formula of classical first-order logic is! So she is certainly in no position to agree that 'that *true* is a valid formula of classical first-order logic', however uncontroversial, represents the necessarily true proposition. What is going on here is that Alice, who does not know any classical first-order logic, cannot assent to the fact that she knows that *true* is a valid formula of first-order logic, despite the fact that it is the necessarily true proposition; she doesn't have enough information, so to speak. One way to capture this is to assume that Alice has an internal representation of the meaning of sentences, which in this case we take to be propositions. We can understand that Alice doesn't have information to assent to the fact that *true* is a valid formula of classical first-order logic by saying that she cannot derive that the meaning of '*true* is a valid formula of classical first-order logic' is a representation of the necessarily true proposition.

With this view in mind, let us say that an agent is *aware* of a sentence if he can derive a representation of the proposition corresponding to the sentence. The actual form of the representation is not relevant for our purposes, as long as there is

a well-defined way of relating the representation of a proposition to the proposition it represents. Note further that this presentation is agnostic as to which proposition we take to be the meaning of the sentence.[13] A reasonable notion of knowledge can be defined by saying that an agent knows S (where S is a sentence) if the agent is aware of P, the representation of the proposition P expressed by S, and P is true at all worlds he considers as possible alternatives to the actual world.[14] Observe first that knowledge applies to sentences, even though in a formal sense the final objects of knowledge really are proposition (via their representation in the agent's internal language). Second, this requires us to talk about representation issues when dealing with knowledge.[15]

What should an agent be aware of? More to the point, if an agent is aware of a particular sentence, does this imply that he is aware of other sentences? Intuitively, this should be the case. For instance, it seems reasonable to say that if an agent is aware of the sentence P *and* Q, then he is in fact aware of both P and Q. More generally, it appears plausible that awareness is closed under subsentences; if Q is a subsentence of P, then if the agent is aware of P, then he is aware of Q. This seems to be the content of standard linguistic compositionality principles, that among other things say that to determine the meaning of P, we need the meaning of its subsentences, including Q.[16] Hence, deriving the representation of the meaning of P should require a representation of the meaning of Q, that is, this requires that we are aware of Q. However, one can check that if awareness is closed under subsentences, then awareness is closed under material implication: if an agent is aware of P and is aware of $(P \Rightarrow Q)$, then the agent is aware of Q. This is not quite logical omniscience, but still a strong principle nonetheless.

Is this really such a strong principle? Consider the Alice example from earlier in this section. Assume Alice is in fact a logician, and is thus aware of a sound and complete deductive system for first-order logic. The deduction rules are of the form $S_1 \Rightarrow S_2$, for instance, ⌜if φ is a valid formula of classical first-order logic and ψ is a valid formula of classical first-order logic, then $\varphi \wedge \psi$ is a valid formula of classical first-order logic⌝. Alice is aware of all such rules, which simply means

[13] For instance, this is compatible with Stalnaker's [1987] propositional-concept approach to belief attribution.

[14] A general notion of awareness is formally defined Fagin and Halpern [1988], and studied further by Huang and Kwast [1991]. Interestingly, awareness can be shown to be equivalent in expressive power to impossible worlds. In other words, from a purely formal point of view, any situation that can be described or analyzed in terms of impossible worlds can also be described in terms of awareness. Wansing [1990] shows how impossible worlds can capture what awareness can capture. The converse result is an exercise in Fagin *et al.* [1995, Ex.9.45].

[15] Dienes and Perner [1999] attempt to make a distinction between implicit and explicit knowledge via cognitive notions, although in a sense different than what we try to do here. They are much close to the kind of distinctions that the sentence storage model of belief attempts to make.

[16] On the other hand, there are situations where this is not so clear. The best candidate along those lines would be a sentence S *or not* S, which we can reasonbly expect to be aware of as representing the necessarily true proposition, without being aware of the sentence S (that is, without having a representation for the proposition expressed by S).

that she can derive a representation of the proposition expressed by all such rules. Part of the assumption that Alice is a logician includes that she is aware of the sentence '*true* is a valid formula of classical first-order logic'. Let φ be an arbitrary complex and valid formula of classical first-order logic. Since the deductive system is complete, there is a deduction in the deductive system that proves φ. Because Alice is aware of '*true* is a valid formula of classical first-order logic', and she is aware of all the deduction rules of the complete deductive system, she is aware of $\ulcorner\varphi$ is a valid formula of classical first-order logic$\urcorner$. In other words, she can derive a representation for this sentence, which must be a representation of the necessarily true proposition: she can derive that φ is valid, for all valid φ. Thus, it seems that closure under awareness is still too strong a principle to capture a reasonable notion of knowledge.

Closure under material implication (and thus closure under subsentences) essentially misses out on a particular feature of real reasoning agents, the same feature that appears to be the problem with logical omniscience. Intuitively, it does not capture that fact that deriving a logical conclusion from a set of hypotheses is a *process* that requires mental resources, perhaps more resources than the agent possesses. This suggest that while we need in some sense a "theory" of what sentences an agent is aware of, this theory should not only take the form of a set of rules describing what an agent is aware of, based on sentences he is already aware of, but also take into account the process of deriving sentences he is aware of, keeping track of the mental resources needed to perform the derivation.

A.3 Towards Computational Mental Representations

How can we characterize, then, awareness of sentences by taking into account the process of deriving the representation of a proposition expressing the meaning of a sentence? The answer is implicit in the use of the term "process" above. More precisely, we will view the process of deriving the representation of a proposition expressing the meaning of a sentence to be an effectively computable process performed by the agent.

The notion of effectively computable process (or effectively computable function) goes back to Church and Turing.[17] Roughly speaking, a function is effectively computable if there is a "mechanical" procedure (mechanical in the sense that it does not require intuition) that produces the results of the function given its inputs. (We must also assume that the inputs are, in some sense, themselves computable.) A Turing machine is an abstract device that captures a particular form effective computability. The *Church-Turing thesis* (CT) is the widely believed thesis that

[17] See Church [1936] and Turing [1936].

Turing machines in fact capture all forms of effective computability. (This thesis is supported by the fact that every other alternative to Turing machines that captures a form of effective computability can be shown to be equivalent to Turing machines.) In other words, if a function can be effectively computed by any conceivable mechanism, then it can be computed by a Turing machine. This may not have been the belief of Church and Turing, but it has become the common interpretation of CT.[18]

We can posit that a nonsupernatural being, for our purposes, is one for which the process of deriving the representation of the proposition expressed by a sentence is an effectively computable process. (As opposed, say, to a being that could magically derive the representation of the proposition expressed by the sentence.) Assuming CT, and assuming that the representation of the proposition is of a suitable form, this means that this derivation process can be simulated by a Turing machine.

This view that the activity of the brain might be somehow related to the notion of effective computability is certainly not original. The general assumption that the brain, or in fact any biological system, is an effectively computable process, and thereby can be simulated by a Turing machine is shared by many philosophers. Searle, for instance, writes:

> Can the operations of the brain be simulated on a digital computer? ... The answer seems to me ... demonstrably 'Yes' ... That is, naturally interpreted, the question means: Is there some description of the brain such that under that description you could do a computational simulation of the operations of the brain. But given Church's thesis that anything that can be given a precise enough characterization as a set of steps can be simulated on a digital computer, it follows trivially that the question has an affirmative answer. [Searle 1992, p.200]

Similarly, Johnson-Laird and the Churchlands write:

> If you assume that [consciousness] is scientifically explicable ... [and] [g]ranted that the [Church-Turing] thesis is correct, then ... [i]f you believe [functionalism] to be false ... then ... you [should] hold that consciousness could be modeled in a computer program in the same way that, say, the weather can be modeled ... [and if] you accept functionalism ... you should believe that consciousness is a computational process. [Johnson-Laird 1987, p.252]

> Church's Thesis says that whatever is computable is Turing computable. Assuming, with some safety, that what the mind-brain does is computable, then it can in principle be simulated by a computer. [Churchland and Churchland 1983, p.6]

While these are all interesting statements which, if true, have deep consequences for the study of cognition, we must remark that for the purposes of this chapter, one

[18] See Copeland [2002] for a discussion of these views. There are also dissenting opinions, holding that Turing machines do not fully capture effective computability; see Penrose [1989, 1994] and Steiglitz [1988] for typical arguments along those lines.

need not go so far as assume that the whole process of cognition is captured by Turing machines. Specifically, it is sufficient that there is a notion of a representation of a proposition, and a notion of effectively computing this representation from a sentence.

Perhaps the most vocal advocate of the computational view of propositional attitudes is Fodor, who holds a view quite close to the one presented here, at least superficially. Fodor argues[19] that a propositional attitude such a *believing that* P should be understood as a computational relation between an organism and a mental representation expressing the proposition P. In a way, this chapter describes a particular formalization of Fodor's ideas, and in the next section provides a suitable logic for reasoning about knowledge in such a setting.

Our thesis is that computational issues are relevant to a notion of knowledge that pertains to address human-like knowledge. As we argued in the last section, it seems required to consider the process under which representations of propositions are derived from sentences, in order to capture the fact that the mental capacities of nonsupernatural beings are resource-limited. An interesting consequence of this view is that if we believe that the Church-Turing thesis applies to this derivation process, that it is somehow effective, then there are fundamental limits to the representations of propositions we can derive from sentences. What kind of limits? One of the main results related to Turing machines, and in fact the motivation behind Turing's research, was to show that there exists problems that are not computable using Turing machines, and hence, via CT, that are not effectively computable. These are the so-called *undecidable* problems.

A classical problem that is undecidable is the problem of deciding the validity of an arbitrary classical first-order logic formula. More specifically, consider the problem of deciding, given a formula of classical first-order logic, whether or not it is valid, that is, true in all models. It can be proved that there is no Turing machine that can answer such a question uniformly for all formulas. Assuming CT, this means that there is no effectively computable process that can reliably answer the question of whether a formula of first-order logic is valid or not. The existence of undecidable problems seems to imply that it is not possible for a nonsupernatural entity to always be able to derive a representation of the proposition corresponding to a sentence. Returning to the example of Alice in the last section, consider again the knowledge attribution:

Alice knows that φ is a valid formula of classical first-order logic.

If the process of deriving a representation of the proposition expressed by $\ulcorner\varphi$ is a valid formula of classical first-order logic$\urcorner$ for any given φ is indeed subject

[19] See Fodor [1976, 1981].

to the Church-Turing thesis, then there is no effectively computable process that can derive for each φ whether that representation is the necessarily true or the necessarily false proposition. In other words, Alice cannot know $\ulcorner\varphi$ is a valid formula of classical first-order logic$\urcorner$ for *all* φ, even if she has complete mastery of classical first-order logic.

As we mentioned, there is some debate as to the question of whether our brains can be simulated by Turing machines. We also pointed out that all we need is the somewhat more restricted assumption that only our ability to derive a representation of propositions corresponding to the meaning of sentences need to be so simulated. In fact, the argument goes through relatively unchanged if we allow the computation of the representation of propositions to be simulated by more powerful forms of Turing machines, such as relativized Turing machines. (Roughly speaking, a relativized Turing machine is a Turing machine with access to an oracle that can itself be noncomputable.)

A.4 Computational Knowledge

The approach of Hintikka, to explain knowledge in terms of epistemically possible worlds, has the advantage of immediately giving rise to a logic for knowledge. Indeed, a possible-worlds account of knowledge, where the objects of knowledge are propositions, gives us a semantics for a normal modal logic of knowledge.[20]

Why is this interesting? Essentially, it gives a way to argue, in an abstract setting, for various properties of knowledge, such as positive and negative introspection, and argue whether they are reasonable or not. For instance, some properties of knowledge that come out of the possible-words formalization translate directly into axioms for knowledge such as the knowledge distribution axiom $K\varphi \wedge K(\varphi \Rightarrow \psi) \Rightarrow K\psi$ or the knowledge introspection axiom $K\varphi \Rightarrow KK\varphi$. Of course, there is some debate as to what properties knowledge ought to have even in Hintikka's definition, and these debates translate into debates as to the correct axioms that knowledge satisfies in such a logic. But the point is that Hintikka's definition of knowledge has an associated logic which is amenable to interesting axiomatization about which one can debate.

In this section, we examine to what extent we can talk about a logic for the account of knowledge that we have been talking about, and that we will call *computational knowledge*. Clearly, such an account intrinsically depends on the computational process of going from sentences to a representation of propositions.

The idea is simple: we model explicitly the process of deriving a representation of the proposition expressed by a sentence, or perhaps simply an abstraction of

[20] See Hughes and Cresswell [1972].

this process. To this end, we assume for every agent a *meaning algorithm* taking a sentence of the language and a representation of the actual world for the agent, and returning a representation of the proposition corresponding to the sentence (or perhaps an indication that the derivation cannot happen). We will represent the latter event by the token $\boxed{?}$, assumed not to correspond to the representation of any existing proposition. Since this algorithm is computational in nature, it falls under the discussion of the previous section, and hence is subject to the CT limitations that it be simulated by a Turing machine, assuming that our representations themselves effective, which we will also assume. An agent is then said to computationally know a sentence if the agent's meaning algorithm says that the sentence, given the agent's representation of the actual world, corresponds to the representation of a proposition P, and P holds at all the worlds that the agent considers as possible alternatives to the actual world. We assume that the meaning algorithm always terminates, and returns $\boxed{?}$ when a representation cannot be derived (perhaps because it requires too much mental resources).[21]

The above refers to a representation of the actual world. Intuitively, this is meant to encompass everything that the agent needs to take into consideration in order to derive a representation of the proposition of the sentence. Among other things, the "meaning" of the words (or lexical entities) in the sentence need to be known. Not every competent English speaker will know all the words in all the grammatical English sentences, or even all the synonyms for a given word.

Let me now give a logical formalization of the above, in a way done for instance by Hintikka. The models of the logic still use possible worlds, so that we can make sense of the notion of a proposition, and implicit knowledge is still defined as truth at all possible worlds. We furthermore equip the agent about whose knowledge we are ostensibly reasoning with a meaning algorithm that gives, for every sentence, a representation of the proposition corresponding to the sentence.

We have to be a little bit careful here. We need to distinguish the primitive facts about the world, that make us the propositions, from the primitive vocabulary. Under any reasonable interpretation of propositions, there are uncountably many primitive facts. On the other hand, the primitive vocabulary in which we express ourselves (or more accurately in which the agents express themselves) has at most countably many words. They can be enumerated in a dictionary, for instance. Note that only countably many sentences can be written, given this countable primitive vocabulary. We certainly do not have a name for every real number, say, although

[21] This approach is inspired by the work of Parikh [1987], who introduced a form of *linguistic knowledge*, and Halpern, Moses and Vardi [1994], who introduced a form of *algorithmic knowledge*. In the latter case, the algorithms associated with the agents directly return "Yes", "No", or "?" to the question of whether the agent knows the given formula in the agent's current state.

we can describe the "naturally" occurring ones using more and more complex sentences.[22]

The logic itself needs to distinguish sentences, which are technically objects of knowledge in this setting, from representations of propositions. For simplicity, we only consider a propositional logic in this chapter. The ideas extend straightforwardly to predicate logic. We assume a primitive vocabulary Φ_v, containing the basic elements of the vocabulary of the language. We assume that this set is countable, following the above discussion. We also assume a set Φ_f of primitive facts about the world, and we assume that $\Phi_v \subseteq \Phi_f$.

We define a sentence σ to be either a primitive vocabulary element $v \in \Phi_v$, a conjunction σ_1 **and** σ_2 (for sentences σ_1 and σ_2), a disjunction σ_1 **or** σ_2 (for sentences σ_1 and σ_2), an implication σ_1 **implies** σ_2 (for sentences σ_1 and σ_2), or a negation **not** σ_1 (for a sentence σ_1). This defines a very simply sentential language, and clearly could be extended to cover more of English. The above lets us retain the flavour of propositional logic. Note that we do not define implication and disjunction by abbreviation as it is often done in many accounts of propositional logic.

We define a representation for proposition as a formula of a propositional logic. (Recall that a proposition is a set of possible worlds; we will have the equivalence that a set of possible worlds is equivalent to the representation of a proposition P that is true at all of the worlds in P) The representation is a formula of a propositional language that we presently introduce. A formula is either a primitive fact $p \in \Phi_f$, a conjunction $\varphi_1 \wedge \varphi_2$ (for formulas φ_1 and φ_2), a negation $\neg\varphi_1$ (for a formula φ_1), and a knowledge attribution $K\sigma$ (for a sentence σ). Note that syntactically, the object of knowledge in this setting is a sentence. In this propositional logic, we do define $\varphi \vee \psi$ as an abbreviation for $\neg(\neg\varphi \wedge \neg\psi)$, and $\varphi \Rightarrow \psi$ as an abbreviation for $\neg\varphi \vee \psi$.

To give a semantics to this logic, we start with augmented Kripke structures $M = (W, \mathcal{K}, \pi, \mathcal{A})$, where W is a set of worlds, and $\mathcal{K}$ is a binary relation on worlds that represents the worlds that an agent considers as possible alternatives to the actual world. We typically write $w' \in \mathcal{K}(w)$ if $(w', w) \in \mathcal{K}$. The function π associate with every world a set of primitive facts, essentially the primitive facts that are true at that world. We also have a meaning algorithm $\mathcal{A}$ that takes as input the sentence whose representation of the proposition we want to establish, as well as a representation of the world that the agent takes into consideration when deriving the propositional meaning. We do not go into the details of this representa-

[22] This statement does not have as much content as one may think. What is a "naturally" occurring real number? Say this is a real number that occurs during the course of mathematical investigation. This mathematical investigation that gives rise to this real number can be considered as a description of that real number, which seems expressible using a countable language, in this case, English supplemented by the apparatus of formal mathematics.

tion, but simply posit a function $\mathsf{rep}(w)$ that takes a world and somehow hands out representation of that world that the agent can take into account. We assume this representation to be effective. This representation should also be compatible with the $\mathcal{K}$ relation, so that if $w' \in \mathcal{K}(w)$, we have $\mathsf{rep}(w') = \mathsf{rep}(w)$. In other words, if the agent considers both w and w' as possible alternatives to the actual world, then the agent should not have different representations corresponding to these worlds. As we discussed above, we assume that the algorithm always terminates, perhaps with $\boxed{\textbf{?}}$ if no meaning can be derived for the sentence.

We can now define the truth of a formula $(M, w) \models \varphi$, as follows:

$(M, w) \models p$ if $p \in \pi(w)$
$(M, w) \models \varphi_1 \wedge \varphi_2$ if $(M, w) \models \varphi_1$ and $(M, w) \models \varphi_2$
$(M, w) \models \neg\varphi_1$ if $(M, w) \not\models \varphi_1$
$(M, w) \models K\sigma$ if $\mathcal{A}(\sigma, \mathsf{rep}(w)) \neq \boxed{\textbf{?}}$ and for all $w' \in \mathcal{K}(w)$, $(M, w') \models \mathcal{A}(\sigma, \mathsf{rep}(w))$.

As usual, we write $M \models \varphi$ if $(M, w) \models \varphi$ for all $w \in W$.

Given the discussion in the previous sections, it should be clear that the above logic does not suffer from the logical omniscience problem. Depending on the meaning algorithm $\mathcal{A}$, it certainly can be the case that $(M, w) \models K\sigma$ and $(M, w) \models K(\sigma \textbf{ implies } \sigma')$, but that $(M, w) \not\models K\sigma'$, if the meaning algorithm cannot derive the appropriate representation for σ', given his representation of the world w.

As a simple example, consider once again Alice, the logician. This example can be modeled using a model M with a single world w, the actual world. Let Φ_v be the set of sentences $\ulcorner \varphi$ is a valid formula of classical first-order logic$\urcorner$ for all formulas φ, and let $\Phi_f = \Phi_v$. Let the interpretation π at w be all the sentences in Φ_v where the formula φ appearing in the sentence is indeed a valid formula of classical first-order logic. The representation $\mathsf{rep}(w)$ of the actual world is inconsequential for this example, so we simply take it to be null. As Alice's algorithm $\mathcal{A}$, take a variant of the resolution procedure[23] restricted to a fixed number of resolvents, say ten. The algorithm will return one of the following tokens: **T** as a representation of the necessarily true proposition, **F** as a representation of the necessarily false proposition, and $\boxed{\textbf{?}}$ if the procedure does not return a result within ten resolvents. It is easy to see that if a sentence σ in Φ_v talks about a formula of first-order logic φ that is valid but too complicated for the restricted resultion procedure to handle, then $\mathcal{A}(\sigma, \mathsf{rep}(w)) = \boxed{\textbf{?}}$, and thus $(M, w) \models \neg K\sigma$. In other words, there is a sentence that expressses the necessarily true proposition, but that Alice does not know. If σ talks about a formula of first-order logic that is simple enough and is valid, then $\mathcal{A}(\sigma, \mathsf{rep}(w)) = \mathbf{T}$, the necessarily true proposition, and thus $(M, w) \models K\sigma$, since **T** is true at w.

[23] See Nerode and Shore [1994], for instance.

This is a very generic account of computational knowledge, and by itself does not give rise to any logical theory of knowledge. In some sense, if we do not put restrictions on the meaning algorithms we consider, there are no interesting valid formulas involving knowledge. What kind of restrictions could we impose on meaning algorithms to yield classes of meaning algorithms such that with respect to models over algorithms in that class, we have interesting and relevant properties of knowledge? Let us focus on one here. First, define the true meaning of a sentence σ, written σ^T, to be the proposition representing the true meaning of σ. This can defined inductively as follows: v^T is just v (recall that we assumed that $\Phi_v \subseteq \Phi_f$), $(\sigma_1 \textbf{ and } \sigma_2)^T$ is $\sigma_1^T \wedge \sigma_2^T$, $(\sigma_1 \textbf{ or } \Sigma_2)^T$ is $\sigma_1^T \vee \sigma_2^T$, $(\sigma_1 \textbf{ implies } \sigma_2)^T$ is $\sigma_1^T \Rightarrow \sigma_2^T$, and $(\textbf{not } \sigma)^T$ as $\neg\sigma^T$. We say an algorithm $\mathcal{A}$ is sound in $M = (W, \mathcal{K}, \pi, \mathcal{A})$ if for all w such that $\mathcal{A}(\sigma, \mathsf{rep}(w)) \neq \boxed{?}$, we have $\mathcal{A}(\sigma, \mathsf{rep}(w))$ logically equivalent to σ^T. In other words, if the algorithm returns a nontrivial result, it is the correct result (as far as the meaning of the sentence is concerned). Let $\mathcal{M}^s$ be the class of augmented Kripke structures over sound meaning algorithms. It is easy to see that in all models M in $\mathcal{M}^s$, $M \models K\sigma \Rightarrow \sigma^T$ is valid, which is essentially the truth axiom for knowledge.

Other general principles of knowledge could potentially be extracted if we assume a universality to the meaning algorithms used by, say, humans. For instance, the kind of universal principle that Chomsky believed was common to all human languages.[24]

A.5 Conclusions

In this chapter, we have investigated a framework for knowledge that retains the flavour of knowledge as truth in all possible worlds, while not suffering from the problem of logical omniscience. This is achieved by essentially capturing the mental process of deriving from a sentence of which knowledge is claimed the mental representation of the proposition corresponding to the meaning of the sentence.

This approach to understanding knowledge pushes the problem of determining the properties of knowledge into the mechanisms used by agents to derive the representation of propositions expressed by sentences. We can view the notion of knowledge as defined by Hintikka as a limit case of the kind of knowledge we have here. The standard possible-worlds approach to knowledge truly represents the knowledge that an ideal agent (not limited by computational issues) would have. In that sense, it is idealized. The strong statement that we can make, if we assume the Church-Turing thesis, and the effectiveness of the representation of proposi-

[24] See, for instance, Chomsky [1968].

tions and the view of the actual world, is that this ideal simply cannot be realized by any human, however much good will one puts into it.

We observe that there appears to be a relationship between our notion of knowledge and Ryle's notion of knowing-how.[25] Ryle describes a dichotomy between knowing-that and knowing-how. That is, there is a distinction of sorts between knowing that a fact is true, and knowing how to perform an action. In some sense, we take the view that knowing a fact is knowing how to derive this fact using an internal mental process. It would be interesting to see if our approach is subject to the same criticisms as the Ryle account of knowing-how. Of course, it may simply be the case that knowing a fact, even on our view, is not equivalent to knowing how to derive this fact using an internal mental process, since the internal process is not explicit to the agent.

It is our belief that we cannot have a uniform notion of knowledge, a theory that encompasses in the abstract what can be known by any individual, without going into some specifics of how that individual is thinking, so to speak. The current approach provides a reasonable hope that some kind of logic can be developed to account for realistic knowledge.

[25] See Ryle [1949]. This relationship was first pointed out by Parikh [1987].

Appendix B
Proofs

THIS appendix gives the proofs of the technical results in the body of the text. For ease of reference, we repeat the statement of the results proved.

B.1 Proofs for Chapter 2

Theorem 2.2. $\mathrm{AX}^{\mathrm{KX}}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KX}}$ *with respect to algorithmic knowledge structures.*

Proof. Proving soundness is straightforward. For completeness, we prove the equivalent statement that if φ is consistent (i.e., if $\neg\varphi$ is not provable from the axioms in $\mathrm{AX}^{\mathrm{KX}}$) then φ is satisfiable in some algorithmic knowledge structure. We can do this by adapting the canonical model constructions typically found in the modal logic literature [Hughes and Cresswell 1972]; we assume knowledge of constructions based on maximal consistent sets of formulas throughout this section.

First, given a set V of formulas, let $V/K = \{\varphi \mid K\varphi \in V\}$. Let $\mathcal{C}$ be the set of all maximal consistent sets of formulas of $\mathcal{L}^{\mathrm{KX}}$. Define the relation $\approx$ over $\mathcal{C}$ by taking $V \approx U$ if and only if $V/K \subseteq U$. We first check that this is an equivalence relation, assuming the axioms K1–K5. For reflexivity, we show $V/K \subseteq V$. Assume $\varphi \in V/K$. Then $K\varphi \in V$, by definition of V/K. By axiom K3, $\varphi \in V$, as desired. For symmetry, we show that $V/K \subseteq U$ implies $U/K \subseteq V$. Let $\varphi \in U/K$. By definition, $K\varphi \in U$. Assume, by way of contradiction, that $\varphi \notin V$. Then by maximality of V, $\neg\varphi \in V$. By K3, we have $\neg K\varphi \in V$. By K5, $K\neg K\varphi \in V$, so that $\neg K\varphi \in V/K \subseteq U$, so that $\neg K\varphi \in U$, but this contradicts $K\varphi \in U$ and U consistent. Thus, $\varphi \in V$, so that $U/K \subseteq V$, as desired. Finally, for transitivity, we show that $T/K \subseteq U$, assuming $T/K \subseteq V$ and $V/K \subseteq U$. Let $\varphi \in T/K$. By definition, $K\varphi \in T$. By K4, $KK\varphi \in T$. Thus, $K\varphi \in T/K \subseteq V$.

Therefore, $\varphi \in V/K \subseteq U$, and $T/K \subseteq U$, as desired. So $\approx$ is an equivalence relation.

A property of interest is that for all ψ in $\mathcal{L}^{\mathrm{KX}}$, if $V \approx U$, then $X\psi \in V$ if and only if $X\psi \in U$. This follows easily from X1. Assume $X\psi \in V$. Then $KX\psi \in V$ by X1, and thus $X\psi \in V/K$, and since $V \approx U$, $X\psi \in U$. The converse direction follows from the fact that $\approx$ is symmetric.

Let φ be a consistent formula of $\mathcal{L}^{\mathrm{KX}}$, and let $Sub(\varphi)$ be the set of subformulas of φ (including φ itself). Since φ is consistent, there is a set $V^\varphi \in \mathcal{C}$ with $\varphi \in V^\varphi$. Let $[V^\varphi]_\approx$ be the $\approx$-equivalence class that contains V^φ. We will use $[V^\varphi]_\approx$ to define the states of our canonical structure. More specifically, define the canonical algorithmic knowledge structure $M^\varphi = (W^\varphi, \mathcal{V}^\varphi, \pi^\varphi, \mathtt{A}^\varphi)$ by taking:

$$
\begin{aligned}
W^\varphi &= \{w_V \mid V \in [V^\varphi]_\approx\} \\
\mathcal{V}^\varphi(w_V) &= \bot \\
\pi^\varphi(w_V)(p) &= \begin{cases} \textbf{true} & \text{if } p \in V \\ \textbf{false} & \text{if } p \notin V \end{cases} \\
\mathtt{A}^\varphi(\psi, \bot) &= \begin{cases} \text{“Yes”} & \text{if } \psi \in Sub(\varphi),\ X\psi \in V^\varphi \\ \text{“No”} & \text{if } \psi \in Sub(\varphi),\ X\psi \notin V^\varphi \\ \text{“?”} & \text{otherwise.} \end{cases}
\end{aligned}
$$

Since $Sub(\varphi)$ is finite, it is easy to see that $\mathtt{A}^\varphi$ is an algorithm that simple searches a given finite list.

We now show that for all $w_V \in W^\varphi$ and all subformulas $\psi \in Sub(\varphi)$, we have $(M^\varphi, w_V) \models \psi$ if and only if $\psi \in V$, by induction on the structure of formulas.

For $true$ and $false$, the result is immediate, since $true$ is in every maximally consistent set, and $false$ is in none. For a primitive proposition p, which is recall a term in T^g_Σ, the result follows immediately. $(M^\varphi, s) \models p$ if and only $\pi^\varphi(s)(p) =$ **true** (by definition) if and only if $p \in V$ (by definition of π^φ). For a conjunction $\psi_1 \wedge \psi_2$, we have $(M^\varphi, s) \models \psi_1 \wedge \psi_2$ if and only if $(M^\varphi, s) \models \psi_1$ and $(M^\varphi, s) \models \psi_2$ if and only (by the induction hypothesis) $\psi_1 \in V$ and $\psi_2 \in V$ if and only $\psi_1 \wedge \psi_2 \in V$ (by maximal consistency of V). For a negation $\neg\psi$, we have $(M^\varphi, s) \models \neg\psi$ if and only if $(M^\varphi, s) \not\models \psi$ if and only if $\psi \notin V$ (by the induction hypothesis) if and only if $\neg\psi \in V$ (by maximal consistency of V).

For a knowledge formula $K\psi$, we have the result following from essentially the same proof as that of Halpern and Moses [1992]. First, assume $(M^\varphi, w_V) \models K\psi$. It follows that $(V/K) \cup \{\neg\psi\}$ is not consistent. (Otherwise, it would be contained in some maximal consistent set U in $\mathcal{C}$, and by construction, we would have $V/K \subseteq U$, and thus $V \approx U$, and since $V \approx V^\varphi$, we have $U \approx V^\varphi$, and $w_U \in W^\varphi$; but since we have $\neg\psi \in U$, we have $\psi \notin U$, and by the induction hy-

pothesis, $(M^\varphi, w_U) \not\models \psi$, contradicting $(M^\varphi, w_V) \models K\psi$.) Since $(V/K) \cup \{\neg\psi\}$ is not consistent, there must be some finite subset $\{\varphi_1, \ldots, \varphi_k, \neg\psi\}$ which is not consistent. By propositional reasoning, we can derive that $\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots))$ is provable, and thus $K(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots)))$ is provable by K2. It is straightforward to derive from this by induction, propositional reasoning, and K1, that $K\varphi_1 \Rightarrow (K\varphi_2 \Rightarrow (\ldots \Rightarrow (K\varphi_k \Rightarrow K\psi)\ldots))$ is provable. Thus, $K\varphi_1 \Rightarrow (K\varphi_2 \Rightarrow (\ldots \Rightarrow (K\varphi_k \Rightarrow K\psi)\ldots)) \in V$. Because $\varphi_1, \ldots, \varphi_k \in V$, we have $K\varphi_1, \ldots, K\varphi_k \in V$, and by MP, we have $K\psi \in V$, as desired. Conversely, if we assume $K\psi \in V$, then $\psi \in V/K$. Let w_U be an arbitrary state of W^φ. By construction of M^φ, $V \approx U$ and thus $V/K \subseteq U$. Therefore, we have $\psi \in U$, and by the induction hypothesis, $(M^\varphi, w_U) \models \psi$. Since w_U was arbitrary, and $\mathcal{V}(w_U) = \mathcal{V}(w_V)$, this means that $(M^\varphi, w_V) \models K\psi$.

Now, consider an algorithmic knowledge formula $X\psi$. Assume that $(M^\varphi, w_V) \models X\psi$. By definition, $\mathtt{A}^\varphi(\psi, \mathcal{V}(w_V)) =$ "Yes", which by the properties of $\mathtt{A}^\varphi$ means that $X\psi \in V^\varphi$. Since $V \approx V^\varphi$, by the property of $\approx$ given above, we have $X\psi \in V$, as required. Conversely, assume that $X\psi \in V$. Since ψ is a subformula of φ (since $X\psi$ is), we have by definition that $\mathtt{A}^\varphi(\psi, \bot) =$ "Yes". Thus, $(M^\varphi, w_V) \models X\psi$.

Completeness of $\mathrm{AX}^{\mathrm{KX}}$ now follows immediately. Since $\varphi \in V^\varphi$ and $\varphi \in Sub(\varphi)$, we have $(M^\varphi, w_{V^\varphi}) \models \varphi$, and thus φ is satisfiable. □

Theorem 2.3. $\mathrm{AX}^{\mathrm{KXD}}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures.*

Proof. Soundness is straightforward. The proof of completeness is just like that of Theorem 2.2, except that now we must also account for the operator $D\varphi$.

First, we verify the following property of $D\varphi$, namely that all ψ in $\mathcal{L}^{\mathrm{KXD}}$, if $V \approx U$, then $D\psi \in V$ if and only if $D\psi \in U$. This follows easily from X3. Assume $D\psi \in V$. Then $KD\psi \in V$ by X3, and thus $D\psi \in V/K$, and since $V \approx U$, $D\psi \in U$. The converse direction follows from the fact that $\approx$ is symmetric.

We construct a canonical structure $M^\varphi = (W^\varphi, \mathcal{V}^\varphi, \pi^\varphi, \mathtt{A}^\varphi)$, as before, except that we take the following definition for $\mathtt{A}^\varphi$:

$$\mathtt{A}^\varphi(\psi, \bot) = \begin{cases} \text{"Yes"} & \text{if } \psi \in Sub(\varphi),\ X\psi \in V^\varphi \\ \text{"No"} & \text{if } \psi \in Sub(\varphi),\ X\psi \notin V^\varphi,\ D\psi \in V^\varphi \\ \text{"?"} & \text{otherwise} \end{cases}$$

We can again show that for all $w_V \in W^\varphi$ and all subformulas $\psi \in Sub(\varphi)$, we have $(M^\varphi, w_V) \models \psi$ if and only if $\psi \in V$, by induction on the structure of formulas. The only case that we need to add is the one for $D\psi$. Assume first that

$(M^\varphi, w_V) \models D\varphi$. Thus, we have $\mathtt{A}^\varphi(\psi, \bot) \in \{\text{“Yes”}, \text{“No”}\}$. By definition, this means either $X\psi \in V^\varphi$, or $X\psi \notin V^\varphi$ and $D\psi \in V^\varphi$. In the first case, we have $X\psi \in V$ (since $V \approx V^\varphi$), and thus $D\psi \in V$ (by axiom X2). In the second case, we have $D\psi \in V^\varphi$, so that $D\psi \in V$ (because $V \approx V^\varphi$). Thus, in all cases, $D\psi \in V$, as required. Conversely, assume $D\psi \in V$. Because $V \approx V^\varphi$, we have $D\psi \in V^\varphi$. By maximality of V^φ, either $X\psi \in V^\varphi$ (in which case $A^\varphi(\psi, \bot) =$ “Yes”), or $\neg X\psi \in V^\varphi$, so that $X\psi \notin V^\varphi$, and thus $\mathtt{A}^\varphi(\psi, \bot) =$ “No”. Thus, in either case, $\mathtt{A}^\varphi(\psi, \bot) \in \{\text{“Yes”}, \text{“No”}\}$, so that $(M^\varphi, w_V) \models D\psi$, as required. The resulting completeness of $\mathrm{AX}^{\mathrm{KXD}}$ now follows as before. □

Theorem 2.4. $\mathrm{AX}^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}\}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures with sound algorithms.*

Proof. Soundness is straightforward. The proof of completeness is exactly as that of Theorem 2.3. The only thing is that we need to verify that the knowledge algorithm $\mathtt{A}^\varphi$ is in fact sound.

Given $w_V \in W^\varphi$, assume $\mathtt{A}^\varphi(\psi, \mathcal{V}(w_V)) =$ “Yes”. By the definition of $\mathtt{A}^\varphi$, we have $X\psi \in \mathit{Sub}(\varphi)$ and $X\psi \in V^\varphi$. Since $V^\varphi \approx V$, $X\psi \in V$. By Axiom X4, $K\psi \in V$. Hence, $\psi \in V/K$. Since $X\psi \in \mathit{Sub}(\varphi)$, $\psi \in \mathit{Sub}(\varphi)$. Let w_U be any state of M^φ. By construction, we have $V \approx U$, and thus $V/K \subseteq U$, so that $\psi \in U$. By the result in the proof of Theorem 2.3, $(M, w_U) \models \psi$. Since w_U was arbitrary, $(M, w_V) \models K\psi$, as required.

Similarly, given $w_V \in W^\varphi$, assume $\mathtt{A}^\varphi(\psi, \mathcal{V}(w_V)) =$ “No”. Then by the definition of $\mathtt{A}^\varphi$, we have $X\psi \in \mathit{Sub}(\varphi)$, $X\psi \notin V^\varphi$, and $D\psi \in V^\varphi$. Since $V^\varphi \approx V$, $X\psi \notin V$ and $D\psi \in V$. By maximality of V, $\neg X\psi \in V$. By Axiom X5, $\neg K\psi \in V$. Since $X\psi \in \mathit{Sub}(\varphi)$, $\psi \in \mathit{Sub}(\varphi)$. We can apply the appropriate part of the proof of Theorem 2.3, to get that if we have $(M, w_V) \models K\psi$, then $K\psi \in V$, contradicting the consistency of V; therefore, $(M, w_V) \models \neg K\psi$, as required.

□

The following known result about the satisfiability of $\mathcal{L}^{\mathrm{K}}$ formulas is central to many proofs of this section. We write $(M, w) \models_K \varphi$ for the satisfaction relation of $\mathcal{L}^{\mathrm{K}}$. Recall that, following Section 2.2, $\mathcal{L}^{\mathrm{K}}$ is interpreted over epistemic structures $M = (W, \mathcal{K}, \pi)$, where we assume that $\mathcal{K}$ is an equivalence relation on W.

Lemma B.1. [Ladner 1977] *Given f an $\mathcal{L}^{\mathrm{K}}$ formula, if f is satisfiable in an epistemic structure, then f is satisfiable in an epistemic structure $M = (W, \mathcal{K}, \pi)$ where $|W| \leq |f|$, and $\mathcal{K}$ is the universal relation, that is, $\mathcal{K} = W \times W$.*

Theorem 2.5. $\mathrm{AX}^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}, \mathrm{X6}\}$ *is a sound and complete axiomatization for* $\mathcal{L}^{\mathrm{KXD}}$ *over algorithmic knowledge structures with sound and complete algorithms.*

Proof. Soundness is straightforward. For completeness, we cannot use the same technique as used in Theorem 2.4. Certainly, we cannot define the knowledge algorithm as we did there. Intuitively, we cannot have the algorithm $\mathtt{A}^\varphi$ return "?", because we want it to be complete. In the original algorithms, we replied "?" when $\psi \notin Sub(\varphi)$. This allowed us to only have to consider finitely many formulas, those in $Sub(\varphi)$. Because we want the algorithm to be both sound, we cannot simply reply "No" when $\psi \notin Sub(\varphi)$. In fact, the only sensical algorithm is the following:

$$\mathtt{A}^\varphi(\psi, \bot) = \begin{cases} \text{"Yes"} & \text{if } X\psi \in V^\varphi \\ \text{"No"} & \text{if } X\psi \notin V^\varphi. \end{cases}$$

However, in general, this does not work, as we have no general way to check whether $X\psi \in V^\varphi$ algorithmically.

Instead, we take a different approach. We again prove the equivalent statement that if a formula φ is consistent, then it is satisfiable. We rely on the intuition that is $\mathtt{A}$ is a sound and complete knowledge algorithm, then $X\varphi$ behaves like $K\varphi$, and $D\varphi$ behaves like $true$. Formally, define a translation from a formula φ of $\mathcal{L}^{\mathrm{KXD}}$ into a formula $\widehat{\varphi}$ of $\mathcal{L}^{\mathrm{K}}$ by taking $\widehat{p} = p$, $\widehat{\neg\varphi} = \neg\widehat{\varphi}$, $\widehat{\varphi_1 \wedge \varphi_2} = \widehat{\varphi_1} \wedge \widehat{\varphi_2}$, $\widehat{K\varphi} = K\widehat{\varphi}$, $\widehat{X\varphi} = K\widehat{\varphi}$, and $\widehat{D\varphi} = true$. It is easy to prove that for all φ, $\vdash \varphi \Leftrightarrow \widehat{\varphi}$, using axioms X4–6. We first establish that if φ is consistent, then $\widehat{\varphi}$ is AX^{K}-consistent. This follows rather immediately from the fact that for all φ, if $\vdash_{\mathrm{AX}^{\mathrm{K}}} \widehat{\varphi}$, then $\vdash \varphi$. (Since every axiom in AX^{K} is an axiom of $\mathrm{AX}^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}, \mathrm{X6}\}$, $\vdash_{\mathrm{AX}^{\mathrm{K}}} \widehat{\varphi}$ implies that $\vdash \widehat{\varphi}$, and applying $\vdash \varphi \Leftrightarrow \widehat{\varphi}$ yields $\vdash \varphi$.) By completeness of AX^{K} for $\mathcal{L}^{\mathrm{K}}$, $\widehat{\varphi}$ AX^{K}-consistent means that $\widehat{\varphi}$ is satisfiable in an epistemic structure. By Lemma B.1, $(M, w) \models_K \widehat{\varphi}$ for some epistemic structure $M = (W, \mathcal{K}, \pi)$ where $|W| \leq |\widehat{\varphi}|$ and $\mathcal{K} = W \times W$. We derive from M an algorithmic knowledge structure $M' = (W, \mathcal{V}, \pi, \mathtt{A})$ by taking $\mathcal{V}(w) = \bot$ and $\mathtt{A}(\varphi, \bot)$ be the algorithm for checking that $(M, w) \models \varphi$ for all $w \in W$. (For instance, we can take the algorithm from Theorem 2.8.) It is straightforward to check that $(M', w) \models \varphi$, and that $\mathtt{A}$ is a sound and complete knowledge algorithm, establishing our result. □

Theorem 2.6. *Let $M = (W, \mathcal{V}, \pi, \mathtt{A})$ be an algorithmic knowledge structure. If $\mathtt{A}$ weakly respects negation, then $M \models X\varphi \Rightarrow \neg X\neg\varphi$. If $\mathtt{A}$ strongly respects negation, then $M \models X\varphi \Leftrightarrow \neg X\neg\varphi$.*

Proof. We prove the result when $\mathtt{A}$ weakly respects negation. (The result when $\mathtt{A}$ strongly respects negation is similar and left to the reader.) Let $w \in W$. If $(M, w) \models X_i\varphi$, then $\mathtt{A}(\varphi, \mathcal{V}_i(w)) =$ "Yes". Since $\mathtt{A}$ weakly respects negation, this implies that $\mathtt{A}(\neg\varphi, \mathcal{V}_i(w)) =$ "No" and hence that $(M, w) \not\models X_i\neg\varphi$, so $(M, w) \models$

$\neg X_i \neg\varphi$. Thus, $(M, w) \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. Since w was arbitrary, we have that $M \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. □

Theorem 2.7.

(a) $\mathrm{AX}_n^{\mathrm{KX}}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KX}}$ *with respect to algorithmic knowledge structures for* n *agents.*

(b) $\mathrm{AX}_n^{\mathrm{KXD}}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *with respect to algorithmic knowledge structures for* n *agents.*

(c) $\mathrm{AX}_n^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}\}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *over algorithmic knowledge structures for* n *agents with sound algorithms.*

(d) $\mathrm{AX}_n^{\mathrm{KXD}} + \{\mathrm{X4}, \mathrm{X5}, \mathrm{X6}\}$ *is a sound and complete axiomatization for* $\mathcal{L}_n^{\mathrm{KXD}}$ *over algorithmic knowledge structures for* n *agents with sound and complete algorithms.*

Proof. (a) This is a straightforward generalization of the proof of Theorem 2.2. Soundness is easy to check. For completeness, we again show that if φ is consistent, then φ is satisfiable. We give the definitions here, leaving the details of the proof to the reader. Given a set V of formulas, let $V/K_i = \{\varphi \mid K_i\varphi \in V\}$. Let $\mathcal{C}$ be the set of all maximal consistent sets of formulas of $\mathcal{L}_n^{\mathrm{KX}}$. We define $\approx_i$ over $\mathcal{C}$, for every i, by taking $V \approx_i U$ if and only if $V/K_i \subseteq U$. We can check that $\approx_i$ is an equivalence relation for every i, assuming the axioms K1–K5, just like in the proof of Theorem 2.2. We can also check that for all ψ, if $V \approx_i U$, then $X_i\psi \in V$ if and only if $X_i\psi \in U$.

Let φ be a consistent formula of $\mathcal{L}^{\mathrm{KX}}$ and let $Sub(\varphi)$ be the set of subformulas of φ (including φ itself). Since φ is consistent, there is a set $V^\varphi \in \mathcal{C}$ with $\varphi \in V^\varphi$. For every i, let $[V^\varphi]_{\approx_i}$ be the $\approx_i$-equivalence class that contains V^φ. We will use $[V^\varphi]_{\approx_1} \cap \cdots \cap [V^\varphi]_{\approx_n}$ to define the states of our canonical structure. More specifically, define the canonical algorithmic knowledge structure $M^\varphi = (W^\varphi, \mathcal{V}_1^\varphi, \ldots, \mathcal{V}_n^\varphi, \pi^\varphi, \mathtt{A}_1^\varphi, \ldots, \mathtt{A}_n^\varphi)$ by taking

$$W^\varphi = \{w_V \mid V \in [V^\varphi]_{\approx_1 \cap \cdots \cap \approx_n}\}$$

$$\mathcal{V}_i^\varphi(w_V) = \bot$$

$$\pi^\varphi(w_V)(p) = \begin{cases} \textbf{true} & \text{if } p \in V \\ \textbf{false} & \text{if } p \notin V \end{cases}$$

$$\mathtt{A}_i^\varphi(\psi, \bot) = \begin{cases} \text{“Yes”} & \text{if } \psi \in Sub(\varphi),\ X_i\psi \in V^\varphi \\ \text{“No”} & \text{if } \psi \in Sub(\varphi),\ X_i\psi \notin V^\varphi \\ \text{“?”} & \text{otherwise.} \end{cases}$$

We can check that M^φ is a deductive algorithmic knowledge structure with n

agents. We can prove, adapting the proof of Theorem 2.2, that for all $w_V \in W^\varphi$ and all subformulas $\psi \in Sub(\varphi)$, $(M^\varphi, w_V) \models \psi$ if and only if $\psi \in V$. Completeness follows from the fact that $\varphi \in V^\varphi$ and $\varphi \in Sub(\varphi)$, so that $(M^\varphi, w_V) \models \varphi$, and thus φ is satisfiable.

(b) This is a straightforward generalization of the proof of Theorem 2.3, along the lines of part (a). We can verify that for all ψ in $\mathcal{L}_n^{\mathrm{KXD}}$, if $V \approx_i U$, then $D_i\psi \in V$ if and only if $D_i\psi \in U$. We construct the canonical algorithmic knowledge structure $M^\varphi = (W^\varphi, \mathcal{V}_1^\varphi, \ldots, \mathcal{V}_n^\varphi, \pi^\varphi, \mathtt{A}_1^\varphi, \ldots, \mathtt{A}_n^\varphi)$ as in part (b), except that we take the following definition for $\mathtt{A}_i^\varphi$:

$$\mathtt{A}_i^\varphi(\psi, \bot) = \begin{cases} \text{“Yes”} & \text{if } \psi \in Sub(\varphi),\ X_i\psi \in V^\varphi \\ \text{“No”} & \text{if } \psi \in Sub(\varphi),\ X_i\psi \notin V^\varphi,\ D_i\psi \in V^\varphi \\ \text{“?”} & \text{otherwise.} \end{cases}$$

(c) This is a straightforward generalization of the proof of Theorem 2.4, along the lines of part (a). Soundness of $\mathtt{A}_i^\varphi$ is proved in exactly the same way.

(d) This is a straightforward generalization of the proof of Theorem 2.5, along the lines of part (a). One (slight) difficulty is that the proof of Theorem 2.5 relies on Lemma B.1, which does not hold for $\mathcal{L}_n^{\mathrm{K}}$. However, a weaker but sufficient result is a *small model theorem* for $\mathcal{L}_n^{\mathrm{K}}$, namely, that if a formula f of $\mathcal{L}_n^{\mathrm{K}}$ is satisfiable in an epistemic structure for n agents, then it is satisfiable in an epistemic structure $M = (W, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$, where $|W|$ is finite [Halpern and Moses 1992]. □

Theorem 2.9. *There is a procedure that runs in time polynomial in $|\varphi| \cdot |W| \cdot f(|\varphi|)$ (where $f(n) = \max\{f_{\mathtt{A}_i}(n) \mid i \in \{1, \ldots, n\}\}$) for deciding, given an algorithmic knowledge structure for n agents $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ and $\varphi \in \mathcal{L}_n^{\mathrm{KX}}$, whether $(M, w) \models \varphi$.*

Proof. Let $\varphi_1, \ldots, \varphi_k$ be the subformulas of φ listed in order of length, with ties broken arbitrarily. Thus, we have $\varphi_k = \varphi$, and if φ_i is a subformula of φ_j, then $i < j$. There are at most $|\varphi|$ subformulas of φ, so we must have $k < |\varphi|$. An easy induction on k' shows that we can klabel each world w in M with φ_j or $\neg\varphi_j$, for $j = 1, \ldots, k'$, depending on whether or not φ_j is true at w, in time $O(k' \cdot |W| \cdot f(|\varphi|))$. In the case where φ_j is of the form $K_i\varphi_{j'}$, where $j' < j$, we label a world w with $K_i\varphi_{j'}$ if and only if each world w' such that $\mathcal{V}_i(w) = \mathcal{V}_i(w')$ is labeled with $\varphi_{j'}$. Assuming inductively that each world has already been labeled with $\varphi_{j'}$ or $\neg\varphi_{j'}$, this step can be carried out in time $O(|W|)$, as desired. In the case where φ_j is of the form $X_i\varphi_{j'}$, where $j' < j$, we label a world w with $X_i\varphi_{j'}$ if and only $A_i(\varphi_{j'}, \mathcal{V}_i(w)) =$ “Yes”. By assumption, this can be done in time $O(f_{\mathtt{A}_i}(|\varphi_{j'}|))$, which is $O(f(|\varphi|))$, since $|\varphi_{j'}| < |\varphi|$. □

Lemma B.1, about the satisfiability of $\mathcal{L}^{\mathrm{K}}$ formulas, is central to the decision procedures for $\mathcal{L}^{\mathrm{KX}}$. For one thing, we can easily reduce the decision problem for $\mathcal{L}^{\mathrm{K}}$ to our logic, by simply ignoring the $X\varphi$ formulas.

Lemma B.2. *If $f \in \mathcal{L}^{\mathrm{K}}$, then f is satisfiable in an epistemic structure if and only if f is satisfiable in $\mathcal{M}^{\mathrm{alg}}$.*

Proof. For the forward direction, assume f is satisfiable in an epistemic structure. $M = (W, \mathcal{K}, \pi)$. Construct the algorithmic knowledge structure $M' = (W, \mathcal{V}, \pi, \mathtt{A})$ by taking $\mathcal{V}(w) = [w]_{\mathcal{K}}$, the equivalence class of w with respect to $\mathcal{K}$. Thus, $(w, w') \in \mathcal{K}$ if and only if $\mathcal{V}(w) = \mathcal{V}(w')$, and thus $\mathcal{K} = \sim$. It is immediate to check by induction on the structure of f that if $(M, w) \models_K f$, then $(M', w) \models f$. For the backwards direction, assume f is satisfiable in an algorithmic knowledge structure $M = (W, \mathcal{V}, \pi, \mathtt{A})$. Construct the epistemic structure $M' = (W, \mathcal{K}, \pi)$ by taking $\mathcal{K} = \sim$. It is immediate to check by induction on the structure of f that if $(M, w) \models f$, then $(M', w) \models_K f$, from which the result is immediate. □

There is a similar relationship between satisfiability of a formula φ in $\mathcal{L}^{\mathrm{KX}}$, and satisfiability in $\mathcal{L}^{\mathrm{K}}$. More precisely, given $\varphi \in \mathcal{L}^{\mathrm{KX}}(\Phi_0)$, let $\tilde{\varphi}$ be defined as follows. The set of formulas $\{X\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)\}$ is countable, so let $\Phi'_0 = \{q_\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}(\Phi_0)\}$ be a countable set of primitive propositions disjoint from Φ_0, where q_ψ corresponds to the formula $X\psi$. Let $\tilde{\varphi}$ be the translation of φ obtained by replacing every occurrence of a formula $X\psi$ by the corresponding q_ψ, in conjunction with formulas $q_\psi \Leftrightarrow Kq_\psi$ for all $X\psi$ appearing in φ. This translation is essentially compositional: $\widetilde{\varphi_1 \wedge \varphi_2}$ is logically equivalent to $\tilde{\varphi}_1 \wedge \tilde{\varphi}_2$, $\widetilde{\neg\varphi}$ is logically equivalent to $\neg\tilde{\varphi}$, and $\widetilde{K\varphi}$ is logically equivalent to $K\tilde{\varphi}$. Note that $|\tilde{\varphi}|$ is polynomial in $|\varphi|$.

Lemma B.3. *If $\varphi \in \mathcal{L}^{\mathrm{KX}}(\Phi_0)$, then φ is satisfiable in $\mathcal{M}^{\mathrm{alg}}$ if and only if $\tilde{\varphi}$ is satisfiable in an epistemic structure.*

Proof. Assume φ is satisfiable in $\mathcal{M}^{\mathrm{alg}}$, that is, there is an algorithmic knowledge structure $M = (W, \mathcal{V}, \pi, \mathtt{A})$ such that $(M, w) \models \varphi$ for some $w \in W$. Construct an epistemic structure $M' = (W, \mathcal{K}, \pi')$ by taking $\pi'(w)(p) = \pi(w)(p)$, if $p \in \Phi_0$, and $\pi'(w)(q_\psi) =$ **true** if and only if $(M, w) \models X\psi$, if $q_\psi \in \Phi'_0$, and by taking $\mathcal{K} = \sim$ on W. It is easy to check by induction on the structure of φ that if $(M, w) \models \varphi$, then $(M', w) \models_K \tilde{\varphi}$.

Conversely, assume $\tilde{\varphi}$ is satisfied in some epistemic structure. By Lemma B.1, we know that there exists an epistemic structure $M = (W, \mathcal{K}, \pi)$ where $|W| \leq |\tilde{\varphi}|$

and $(M, w) \models_K \tilde{\varphi}$ for some $w \in W$. Let $[w_1]_{\mathcal{K}}, \ldots, [w_k]_{\mathcal{K}}$ be an enumeration of the equivalence classes of $\mathcal{K}$, of which there are at most $|\tilde{\varphi}|$, a polynomial in $|\varphi|$. Construct the algorithmic knowledge structure $M' = (W, \mathcal{V}, \pi', \mathtt{A})$, where $\mathcal{V}(w) = i$ (such that $w \in [w_i]_{\mathcal{K}}$), where π' is the restriction of π to the primitive propositions in Φ_0, and where $\mathtt{A}(\psi, i)$ is a lookup algorithm that returns "Yes" if and only if $X\psi$ is a subformula of φ, and $\pi(w_i)(q_\psi) = \textbf{true}$. It is easy to check by induction on the structure of φ that if $(M, w) \models_K \tilde{\varphi}$, then $(M', w) \models \varphi$. □

Theorem 2.11. *The problem of deciding whether a formula φ of $\mathcal{L}_n^{\mathrm{KX}}$ is satisfiable in an algorithmic knowledge structure for n agents is NP-complete if $n = 1$ and PSPACE-complete if $n > 1$.*

Proof. Consider first the case $n = 1$. For the lower bound, we show how to reduce from the decision problem of $\mathcal{L}^{\mathrm{K}}$. Let f be a formula of $\mathcal{L}^{\mathrm{K}}$. By Lemma B.2, f is satisfiable in an epistemic structure if and only if $\hat{f}$ is satisfiable in $\mathcal{M}^{\mathrm{alg}}$. Thus, the complexity of the decision problem for $\mathcal{L}^{\mathrm{K}}$ is a lower bound for our decidability problem, that is, NP. For the upper bound, we need to exhibit a nondeterministic polynomial time algorithm that decides if $\varphi \in \mathcal{L}^{\mathrm{KX}}$ is satisfiable. We will use the decision problem for $\mathcal{L}^{\mathrm{K}}$ itself as an algorithm. By Lemma B.3, φ is satisfiable if and only if $\tilde{\varphi}$ is satisfiable, so we can simply invoke the NP algorithm for $\mathcal{L}^{\mathrm{K}}$ satisfiability on $\tilde{\varphi}$.

The proof for the case $n > 1$ is entirely analogous, except that we use the modal logic $\mathcal{L}_n^{\mathrm{K}}$ rather than $\mathcal{L}^{\mathrm{K}}$. Let f be a formula of $\mathcal{L}_n^{\mathrm{K}}$. We can prove the analogue of Lemma B.2, that f is satisfiable in Kripke structures for n agents if and only if f is satisfiable in $\mathcal{M}_n^{\mathrm{alg}}$, with a proof similar to that of Proposition B.2. This gives us an immediate lower bound, as follows. Let f be an $\mathcal{L}_n^{\mathrm{K}}$ formula. We know f is satisfiable if and only if f is satisfiable in $\mathcal{M}_n^{\mathrm{alg}}$ structures. Since the decision problem for $\mathcal{L}_n^{\mathrm{K}}$ ($n \geq 2$) is PSPACE-complete, the lower bound of PSPACE follows.

Let φ be a formula of $\mathcal{L}_n^{\mathrm{KX}}(\Phi_0)$. For every i, the set of formulas $\{X_i\psi \mid \psi \in \mathcal{L}_n^{\mathrm{KX}}\}$ is countable, so let $\Phi_0^i = \{q_\psi^i \mid \psi \in \mathcal{L}_n^{\mathrm{KX}}\}$ be a countable set of primitive propositions, disjoint from Φ_0, where q_ψ^i corresponds to the formula $X_i\psi$. Let $\tilde{\varphi}$ be the translation of φ obtained by replacing every occurrence of a formula $X_i\psi$ by the corresponding q_ψ^i, in conjunction with formulas $q_\psi^i \Leftrightarrow K_i q_\psi^i$ for all formulas $X_i\psi$ appearing in φ. Note that $|\tilde{\varphi}|$ is polynomial in $|\varphi|$. We can prove an analogue of Lemma B.3, that φ is satisfiable in $\mathcal{M}_n^{\mathrm{alg}}$ if and only if $\tilde{\varphi}$ is satisfiable in an epistemic structure for n agents, using a proof similar to that of Lemma B.3. This gives us an immediate upper bound for our decision problem: φ is satisfiable if and only if $\tilde{\varphi}$ is satisfiable, so we can simply invoke the PSPACE algorithm for $\mathcal{L}_n^{\mathrm{K}}$ satisfiability on $\tilde{\varphi}$. □

B.2 Proofs for Chapter 3

Theorem 3.5. *The axiomatization* $\mathrm{AX}^{\mathrm{ded}}$ *is sound and complete for* $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ *with respect to* $\mathcal{M}^{\mathrm{ded}}(\Sigma)$.

Proof. Proving soundness is straightforward. For completeness, we prove the equivalent statement that if φ is consistent (i.e., if $\neg\varphi$ is not provable from the axioms in $\mathrm{AX}^{\mathrm{ded}}$) then φ is satisfiable in some structure in $\mathcal{M}^{\mathrm{ded}}(\Sigma)$. The proof is similar to that of Theorem 2.2.

First, given a set V of formulas, let $V/K = \{\varphi \mid K\varphi \in V\}$. Let $\mathcal{C}$ be the set of all maximal consistent sets of formulas of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. For $V \in \mathcal{C}$, let $Obs(V) = \{ob \in Obs \mid ob \in V\}$. Define the relation $\approx$ over $\mathcal{C}$ by taking $V \approx U$ if and only if $V/K \subseteq U$. We first check that this is an equivalence relation, assuming the axioms K1–K5. For reflexivity, we show $V/K \subseteq V$. Assume $\varphi \in V/K$. Then $K\varphi \in V$, by definition of V/K. By axiom K3, $\varphi \in V$, as desired. For symmetry, we show that $V/K \subseteq U$ implies $U/K \subseteq V$. Let $\varphi \in U/K$. By definition, $K\varphi \in U$. Assume, by way of contradiction, that $\varphi \notin V$. Then by maximality of V, $\neg\varphi \in V$. By K3, we have $\neg K\varphi \in V$. By K5, $K\neg K\varphi \in V$, so that $\neg K\varphi \in V/K \subseteq U$, so that $\neg K\varphi \in U$, but this contradicts $K\varphi \in U$ and U consistent. Thus, $\varphi \in V$, so that $U/K \subseteq V$, as desired. Finally, for transitivity, we show that $T/K \subseteq U$, assuming $T/K \subseteq V$ and $V/K \subseteq U$. Let $\varphi \in T/K$. By definition, $K\varphi \in T$. By K4, $KK\varphi \in T$. Thus, $K\varphi \in T/K \subseteq V$. Therefore, $\varphi \in V/K \subseteq U$, and $T/K \subseteq U$, as desired. So $\approx$ is an equivalence relation.

The following properties of $\approx$ will turn out to be important. First, if $V \approx U$, then $V \sim U$ (i.e., $Obs(V) = Obs(U)$). Let $ob \in Obs(V) \subseteq V$. Since V is maximally consistent, all instances of X3 are in V, and thus $ob \Rightarrow Kob$ is in V, so by MP, $Kob \in V$, and thus $ob \in V/K \subseteq U$. Therefore, $ob \in U$, and $Obs(V) \subseteq Obs(U)$. Since $\approx$ is an equivalence relation, $V \approx U$ implies that $U \approx V$, and by the above result, we get $Obs(U) \subseteq Obs(V)$. Thus, $V \sim U$, as desired.

The second property of interest is that for all ψ in $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$, if $V \approx U$, then $X\psi \in V$ if and only if $X\psi \in U$. This follows easily from X1. Assume $X\psi \in V$. Then $KX\psi \in V$ by X1, and thus $X\psi \in V/K$, and since $V \approx U$, $X\psi \in U$. The converse direction follows from the fact that $\approx$ is symmetric.

Let φ be a consistent formula of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$, and let $Sub(\varphi)$ be the set of subformulas of φ (including φ itself). Since φ is consistent, there is a set $V^\varphi \in \mathcal{C}$ with $\varphi \in V^\varphi$ with $|Obs(V^\varphi)| < \infty$: construct the set S starting with φ, adding ob for every observations ob appearing in φ if $ob \wedge \varphi$ is consistent, and adding $\neg ob$ for every observation ob either not appearing in φ or inconsistent with φ; it easy to establish that S is consistent, so S is extensible to a maximally consistent set V^φ with $|Obs(V^\varphi)| < \infty$. Let $obs^\varphi = Obs(V^\varphi)$. Let $[V^\varphi]_\approx$ be the $\approx$-equivalence

class that contains V^φ. We will use $[V^\varphi]_\approx$ to define the states of our canonical structure. More specifically, define the canonical deductive algorithmic knowledge structure $M^\varphi = (W^\varphi, \pi^\varphi, D^\varphi)$ by taking:

$$W^\varphi = \{(w_V, obs^\varphi) \mid V \in [V^\varphi]_\approx\}$$

$$\pi^\varphi((w_V, obs^\varphi))(p) = \begin{cases} \textbf{true} & \text{if } p \in V \\ \textbf{false} & \text{if } p \notin V \end{cases}$$

$$D^\varphi = \{(\varnothing, \psi^T) \mid X\psi \in Sub(\varphi), X\psi \in V^\varphi, \psi^T \notin Obs\}.$$

To simplify the discussion, and because obs^φ is fixed in M^φ, we refer to the state (w_V, obs^φ) as simply w_V; for instance, we freely write $w_V \in W^\varphi$. We can check that D^φ defines a KD deductive system, since ψ^T cannot be an observation, nor a variable that can be substituted with an observation. Decidability of D^φ holds trivially, since D^φ contains finitely many deduction rules, as $Sub(\varphi)$ is finite. We can also check that π^φ respects the observation made at a state, since $\pi^\varphi(w_V, obs^\varphi)(ob) = \textbf{true}$ if and only if $ob \in V$ if and only $ob \in Obs(V) = obs^\varphi$. Thus, M^φ is a deductive algorithmic knowledge structure.

We now show that for all $w_V \in W^\varphi$ and all subformulas $\psi \in Sub(\varphi)$, we have $(M^\varphi, w_V) \models \psi$ if and only if $\psi \in V$, by induction on the structure of formulas.

For $true$ and $false$, the result is immediate, since $true$ is in every maximally consistent set, and $false$ is in none. For a primitive proposition p, which is recall a term in T^g_Σ, the result follows immediately. $(M^\varphi, s) \models p$ if and only $\pi^\varphi(s)(p) = \textbf{true}$ (by definition) if and only if $p \in V$ (by definition of π^φ). For a conjunction $\psi_1 \wedge \psi_2$, we have $(M^\varphi, s) \models \psi_1 \wedge \psi_2$ if and only if $(M^\varphi, s) \models \psi_1$ and $(M^\varphi, s) \models \psi_2$ if and only (by the induction hypothesis) $\psi_1 \in V$ and $\psi_2 \in V$ if and only $\psi_1 \wedge \psi_2 \in V$ (by maximal consistency of V). For a negation $\neg\psi$, we have $(M^\varphi, s) \models \neg\psi$ if and only if $(M^\varphi, s) \not\models \psi$ if and only if $\psi \notin V$ (by the induction hypothesis) if and only if $\neg\psi \in V$ (by maximal consistency of V).

Now, consider a deductive algorithmic knowledge formula $X\psi$. First, assume that we have $(M^\varphi, w_V) \models X\psi$. By definition, $obs^\varphi \vdash_{D^\varphi} \psi^T$. If ψ^T is an observation, then by construction of D^φ, it is easy to see that $\psi^T = ob$ for some $ob \in obs^\varphi$, and thus $\psi \in obs^\varphi \subseteq V$. By axiom X2, we have $X\psi \in V$. If ψ^T is not an observation, then again by construction of D^φ, there must exist a rule $\triangleright\psi^T$ in D^φ. In other words, $X\psi \in V^\varphi$. Since $V \approx V^\varphi$ by choice of W^φ, we get $X\psi \in V$, following the result we established above. Conversely, assume that $X\psi \in V$. If ψ^T is an observation, then $\psi \in V$ by axiom X2, meaning that $\psi \in obs^\varphi$, meaning that $(M^\varphi, w_V) \models X\psi$ immediately. If ψ^T is not an observation, then by definition of D^φ, $(\varnothing, \psi^T) \in D^\varphi$, and thus $obs^\varphi \vdash_{D^\varphi} \psi^T$, meaning that $(M^\varphi, w_V) \models X\psi$.

For a knowledge formula $K\psi$, we have the result following from essentially the

same proof as that of Halpern and Moses [1992]. First, assume $(M^\varphi, w_V) \models K\psi$. It follows that $(V/K) \cup \{\neg\psi\}$ is not consistent. (Otherwise, it would be contained in some maximal consistent set U in $\mathcal{C}$, and by construction, we would have $V/K \subseteq U$, and thus $V \approx U$, and hence $V \sim U$; but since we have $\neg\psi \in U$, we have $\psi \notin U$, and by the induction hypothesis, $(M^\varphi, w_U) \not\models \psi$, contradicting $(M^\varphi, w_V) \models K\psi$.) Since $(V/K) \cup \{\neg\psi\}$ is not consistent, there must be some finite subset $\{\varphi_1, \ldots, \varphi_k, \neg\psi\}$ which is not consistent. By propositional reasoning, we can derive that $\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots))$ is provable, and thus $K(\varphi_1 \Rightarrow (\varphi_2 \Rightarrow (\ldots \Rightarrow (\varphi_k \Rightarrow \psi)\ldots)))$ is provable by K2. It is straightforward to derive from this by induction, propositional reasoning, and K1, that $K\varphi_1 \Rightarrow (K\varphi_2 \Rightarrow (\ldots \Rightarrow (K\varphi_k \Rightarrow K\psi)\ldots))$ is provable. Thus, $K\varphi_1 \Rightarrow (K\varphi_2 \Rightarrow (\ldots \Rightarrow (K\varphi_k \Rightarrow K\psi)\ldots)) \in V$. Because $\varphi_1, \ldots, \varphi_k \in V$, we have $K\varphi_1, \ldots, K\varphi_k \in V$, and by MP, we have $K\psi \in V$, as desired. Conversely, if we assume $K\psi \in V$, then $\psi \in V/K$. Let w_U be an arbitrary state of W^φ. By construction of M^φ, $V \approx U$ and thus $V/K \subseteq U$. Therefore, we have $\psi \in u$, and by the induction hypothesis, $(M^\varphi, w_U) \models \psi$. Since w_U was arbitrary, and since $U \sim V$ (since $U \approx V$ by choice of W^φ), this means that $(M^\varphi, w_V) \models K\psi$.

Completeness of $\mathrm{AX}^{\mathrm{ded}}$ now follows immediately. Since $\varphi \in V^\varphi$ and $\varphi \in Sub(\varphi)$, we have $(M^\varphi, w_{V^\varphi}) \models \varphi$, and thus φ is satisfiable. □

Theorem 3.6. *The axiomatization* $\mathrm{AX}^{\mathrm{ded}} + \{Ax^D\}$ *is sound and complete for* $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ *with respect to* $\mathcal{M}^{\mathrm{ded}}_{D\subseteq}(\Sigma)$.

Proof. Soundness is again straightforward. For completeness, we prove the equivalent statement that if φ is consistent (i.e., if $\neg\varphi$ is not provable from the axioms in $\mathrm{AX}^{\mathrm{ded}} \cup Ax^D$) then φ is satisfiable in some structure in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$. The procedure is exactly the one that is used to prove Theorem 3.5, except with a different deductive system D^φ.

We simply indicate where the proof differs from that of Theorem 3.5, and let the reader fill in the details. We construct, for a given φ, a deductive algorithmic knowledge structure $M^\varphi = (W^\varphi, \pi^\varphi, D^\varphi)$, where W^φ and π^φ are constructed as in Theorem 3.5, and D^φ is given by

$$D \cup \{(\varnothing, \psi^T) \mid X\psi \in Sub(\varphi), X\psi \in V^\varphi, \psi^T \notin Obs\}.$$

We can check that M^φ is a deductive algorithmic knowledge structure in $\mathcal{M}^{\mathrm{ded}}_{D\subseteq}(\Sigma)$. The deductive system D^φ has the following interesting property: if $obs^\varphi \vdash_D \psi^T$, then there is a rule $\triangleright\psi^T$ in D^φ. In other words, every term ψ^T derivable from the rules in D is derivable directly with a single rule in D^φ. Here is the proof of this property. Assume $obs^\varphi \vdash_D \psi^T$. Clearly, it is sufficient to show that $X\psi \in V^\varphi$. If

ψ^T is an observation, then because D is a KD deductive system, we have $\psi^T = ob$ for some $ob \in obs^\varphi$, and thus $\psi \in obs^\varphi \subseteq V$. By axiom X2, we have $X\psi \in V^\varphi$. If ψ^T is not an observation, then there must exist a deduction $t_1, \ldots, t_m$ in D such that $t_m = \psi^T$ is a conclusion of the deduction. We show by induction on the length of the deduction that for every i, $X(t_i)^R \in V^\varphi$. If $i = 1$, then either t_i is an observation in V^φ, and thus we have $(t_1)^R = t_1$, and $Xt_1 \in V^\varphi$ follows from X2, or t_i follows from the application of a deduction rule in D, with no antecedents. By construction, there is an instance of this rule in V^φ, of the form $true \Rightarrow X(t_1)$, and thus $X(t_1) \in V^\varphi$. For $i > 1$, again, either t_i is an observation, and the result follows as above, or there is a rule $t'_1, \ldots, t'_k \triangleright t'$ in D such that for some ground substitution ρ such that $\rho(t') = t_i$ and $\rho(t'_j)$ appears in the deduction before term t_i. By construction, there is an instance of $X(t'_1)^R \wedge \cdots \wedge X(t'_k)^R \Rightarrow X(t')^R$ in V^φ, and by induction hypothesis, we have $X(t'_{i_j}) \in V^\varphi$ for each $i_j < i$. Thus, by MP, we have $X(t_i)^R \in V^\varphi$. Since $\varphi^T = t_m$, the last element of the deduction, we get that $X(\varphi^T)^R = X\varphi$ is in V^φ, as desired.

The rest of the proof follows as before. □

For the decision procedures, we can use the same ideas as in the proof of Theorem 2.11, but adapted to deductive systems. As before, we can easily reduce the decision problem for $\mathcal{L}^{\mathrm{K}}$ to our logic, by simply ignoring the $X\varphi$ formulas. One difficulty is that we need to account for the fact that primitive propositions are terms in a term algebra in $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. Consider the following construction. Let f be a formula of $\mathcal{L}^{\mathrm{K}}$. Let $p_1, \ldots, p_k$ be the primitive proposition appearing in f. We first come up with an encoding of these primitive propositions into the language of Σ. For example, we can take p_1 to be true, p_2 to be $\mathsf{not}(\mathsf{true})$, p_3 to be $\mathsf{not}(\mathsf{not}(\mathsf{true}))$, and so forth. Let t_p be the term encoding the primitive proposition p. The one restriction we make on this encoding is that no t_p is an observation in Obs. Let $\hat{f}$ be the formula obtained by replacing every instance of a primitive proposition p in f by t_p. Note that $|\hat{f}|$ is polynomial in $|f|$, and that $\hat{f}$ contains no instance of the X operator.

Proposition B.4. *Given $f \in \mathcal{L}^{\mathrm{K}}$, and given D an arbitrary KD deductive system over Σ, the following are equivalent:*

(1) *f is satisfiable in an epistemic structure,*
(2) *$\hat{f}$ is satisfiable in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$,*
(3) *$\hat{f}$ is satisfiable in $\mathcal{M}^{\mathrm{ded}}(\Sigma)$.*

Proof. $(1) \Rightarrow (2)$: Assume f is satisfiable in an epistemic structure. By Lemma B.1, we know that there exists an epistemic structure $M = (W, \mathcal{K}, \pi)$ where $|W| \leq |f|$,

$\mathcal{K}$ is an equivalence relation on W and $(M, w) \models_K f$ for some $w \in W$.[1] Let $\{[w]_{\mathcal{K}} \mid w \in W\}$ be the set of equivalence classes of $\mathcal{K}$, of which there are at most $|f|$. We encode these equivalence classes using an encoding similar to that for primitive propositions, except we take the encoding terms to be observations. Let $\mathsf{ob}(\mathsf{false}), \mathsf{ob}(\mathsf{not}(\mathsf{false})), \mathsf{ob}(\mathsf{not}(\mathsf{not}(\mathsf{false}))), \ldots$ be an encoding of these equivalence classes, where we denote by t_w the encoding of $[w]_{\mathcal{K}}$. Thus, $(w, w') \in \mathcal{K}$ if and only if $t_w = t_{w'}$. Construct the deductive algorithmic knowledge structure $M' = (W', \pi', D)$, where $W' = \{(w, \{t_w\}) \mid w \in W\}$, and π' is given as follows. For a term t_p, $\pi'((w, \{t_w\}))(t_p) = \pi(w)(p)$. For a term t_w, then $\pi'((w', \{t_{w'}\}))(t_w) =$ **true** if and only if $t_w = t_{w'}$. For all other terms t, we take $\pi'((w, \{t_{w'}\}))(t) =$ **false**. It is easy to see that π' respects observations. It is also easy to check by induction on the structure of f that if $(M, w) \models_K f$, then $(M', (w, \{t_w\})) \models \hat{f}$. Here are the interesting cases of the induction. If f is p, then by assumption, $(M, w) \models_K p$, so $\pi(w)(p) =$ **true**; thus, $\pi'((w, \{t_w\}))(t_p) =$ **true**, and $(M', (w, \{t_w\})) \models t_p$. If f is Kg, then by assumption, $(M, w) \models_K Kg$, so that for all $w' \in \mathcal{K}(w)$, $(M, w') \models_K g$. By the induction hypothesis, we have for all $w' \in \mathcal{K}(w)$, $(M', (w', \{t_{w'}\})) \models \hat{g}$, which is equivalent to saying that for all $(w', \{t_{w'}\}) \sim (w, \{t_w\})$, $(M', (w', \{t_{w'}\})) \models \hat{g}$, and thus $(M', (w, \{t_w\})) \models K\hat{g}$, as required.

(2) $\Rightarrow$ (3): This is immediate, since $\mathcal{M}_D^{\mathrm{ded}}(\Sigma) \subseteq \mathcal{M}^{\mathrm{ded}}(\Sigma)$.

(3) $\Rightarrow$ (1): Assume $\hat{f}$ is satisfiable in a deductive algorithmic knowledge structure $M = (W, \pi, D')$, that is, $(M, w) \models \hat{f}$ for some $w \in W$. Construct the epistemic structure $M' = (W, \mathcal{K}, \pi')$ by taking $\pi'(w)(p) =$ **true** if and only if $\pi(w)(t_p) =$ **true**, and $\mathcal{K} = \sim$. It is easy to check by induction on the structure of f that if $(M, w) \models \hat{f}$, then $(M', w) \models_K f$. Here are the interesting cases of the induction. If f is p, then by assumption, $(M, w) \models t_p$, so that $\pi(w)(t_p) =$ **true**. Thus means $\pi'(w)(p) =$ **true**, and thus $(M', w) \models_K p$. If f is Kg, then by assumption, $(M, w) \models K\hat{g}$, that is, for all $w' \sim w$, $(M, w') \models \hat{g}$. By the induction hypothesis, this yields for all $w' \sim w$, $(M', w') \models_K g$, which is equivalent to the fact that for all $w' \in \mathcal{K}(w)$, $(M', w') \models_K g$, that is, $(M', w) \models_K Kg$, as required. □

There is a similar relationship between satisfiability of a formula φ in our logic, and satisfiability in $\mathcal{L}^{\mathrm{K}}$. More precisely, given $\varphi \in \mathcal{L}^{\mathrm{KX}}(T_\Sigma^g)$, let $\tilde{\varphi}$ be defined as follows. The set T_Σ^g is countable, so let $\{p_t \mid t \in T_\Sigma^g\}$ be a countable set of primitive propositions corresponding to the ground terms of T_Σ. Similarly, the set of formulas $\{X\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}(T_\Sigma^g), \psi^T \notin Obs\}$ is countable, so let $\{q_\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}(T_\Sigma^g), \psi^T \notin Obs\}$ be a countable set of primitive propositions where

[1] While Proposition B.1 says that the equivalence relation $\mathcal{K}$ can be taken to be universal, we will not take advantage of this in this proof or the proof of Proposition B.5. This is in order to simplify the generalization of these proofs to the multiple agents case (Theorem 3.13).

q_ψ corresponds to the formula $X\psi$. Let $\tilde{\varphi}$ be the translation of φ obtained by replacing every occurrence of a term t in T^g_Σ by p_t, every occurrence of a formula $X\psi$ where $\psi^T \in Obs$ by p_{ψ^T}, and every occurrence of a formula $X\psi$ where $\psi^T \notin Obs$ by the corresponding q_ψ, in conjunction with formulas $p_{ob} \Leftrightarrow Kp_{ob}$ for all observations ob appearing in φ. This translation is essentially compositional: $\widetilde{\varphi_1 \wedge \varphi_2}$ is logically equivalent to $\tilde{\varphi}_1 \wedge \tilde{\varphi}_2$, $\widetilde{\neg\varphi}$ is logically equivalent to $\neg\tilde{\varphi}$, and $\widetilde{K\varphi}$ is logically equivalent to $K\tilde{\varphi}$. Note that $|\tilde{\varphi}|$ is polynomial in $|\varphi|$.

Proposition B.5. *If $\varphi \in \mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$, then φ is satisfiable in $\mathfrak{M}^{\mathrm{ded}}(\Sigma)$ if and only if $\tilde{\varphi}$ is satisfiable in an epistemic structure.*

Proof. Assume φ is satisfiable in $\mathfrak{M}^{\mathrm{ded}}(\Sigma)$, that is, there is a deductive algorithmic knowledge structure $M = (W, \pi, D)$ such that $(M, w) \models \varphi$ for some $w \in W$. Construct an epistemic structure $M' = (W, \mathcal{K}, \pi')$ by taking $\pi'(w)(p_t) = \pi(w)(p_t)$ and $\pi'(w)(q_\psi) =$ **true** if and only if $(M, w) \models X\psi$, and $\mathcal{K} = \sim$. It is easy to check by induction on the structure of φ that if $(M, w) \models \varphi$, then $(M', w) \models_K \tilde{\varphi}$. Here are the interesting cases of the induction. If φ is t, then by assumption, $(M, w) \models t$, and $\pi(w)(t) =$ **true**. This yields $\pi'(w)(p_t) =$ **true**, and $(M', w) \models_K p_t$. If φ is $X\psi$, then by assumption, $(M, w) \models X\psi$, and therefore $\pi'(w)(q_\psi) =$ **true**, so that $(M', w) \models_K q_\psi$. If φ is $K\psi$, then by assumption, $(M, w) \models K\psi$, that is, for all $w' \sim w$, $(M, w') \models \psi$. By the induction hypothesis, and the definition of $\mathcal{K}$, we have for all $w' \in \mathcal{K}(w)$, $(M', w') \models_K \tilde{\psi}$, that is, $(M', w) \models_K K\tilde{\psi}$, as required.

Conversely, assume $\tilde{\varphi}$ is satisfied in some epistemic structure. By Lemma B.1, we know that there exists an epistemic structure $M = (W, \mathcal{K}, \pi)$ where $|W| \leq |\tilde{\varphi}|$ and $(M, w) \models_K \tilde{\varphi}$ for some $w \in W$. Let $\{[w]_\mathcal{K} \mid w \in W\}$ be the set of equivalence classes of $\mathcal{K}$, of which there are at most $|\tilde{\varphi}|$, which is polynomial in $|\varphi|$. Let $\mathsf{ob}(t_1), \mathsf{ob}(t_2), \ldots$ be an encoding of these equivalence classes using terms $t_i \in T^g_\Sigma$ such that none of $\mathsf{ob}(t_i)$ appears in φ. We denote by t_w the term encoding the class $[w]_\mathcal{K}$. Thus, $(w, w') \in \mathcal{K}$ if and only if $t_w = t_{w'}$. For every world $w \in W$, let $obs(w) = \{ob \mid ob \in Obs, \pi(w)(p_{ob}) =$ **true**, ob appears in $\varphi\}$, that is, the observations made at w. By the construction of $\tilde{\varphi}$, we have that if $(w, w') \in \mathcal{K}$, then $obs(w) = obs(w')$. Construct the deductive algorithmic knowledge structure $M' = (W', \pi', D)$, where $S' = \{(w, \{t_w\} \cup obs(w)) \mid w \in W\}$, and π' is given as follows. For a term $\mathsf{ob}(t_i)$, $\pi'((w, obs))(\mathsf{ob}(t_i)) =$ **true** if and only if $\mathsf{ob}(t_i) \in obs$. For any other term t, $\pi'((w, obs))(t) =$ **true** if and only if $\pi(w)(p_t) =$ **true**. It is easy to see, from the definition of $obs(w)$, that π' respects observations. Finally, take $D = \{(\{t_w\}, \psi^T) \mid \pi(w)(q_\psi) =$ **true**$, \psi^T \notin Obs)\}$. It is easy to check by induction on the structure of φ that if $(M, w) \models_K \tilde{\varphi}$, then $(M', (w, \{t_w\} \cup obs(w))) \models \varphi$. Here are the interesting cases of the induction.

If φ is a term t (not any of $\mathsf{ob}(t_i)$, by the choice of encoding), then by assumption, $(M, w) \models_K p_t$, so that $\pi(w)(p_t) = \textbf{true}$. Thus, we have $\pi'((w, \{t_w\} \cup obs(w)))(t) = \textbf{true}$, and $(M', (w, \{t_w\} \cup obs(w))) \models t$. If φ is $X\psi$ where $\psi^T \in Obs$, then by assumption, $(M, w) \models_K p_{\psi^T}$, so $\pi(w)(p_{\psi^T}) = \textbf{true}$, and thus $\pi'((w, \{t_w\} \cup obs(w)))(\psi^T) = \textbf{true}$, yielding $(M', (w, \{t_w\} \cup obs(w))) \models \psi^T$, and thus $(M', (w, \{t_w\} \cup obs(w))) \models X\psi^T$. If φ is $X\psi$ where $\psi^T \notin Obs$, then we have by assumption $(M, w) \models_K q_\psi$, and so $\pi(w)(q_\psi) = \textbf{true}$, meaning that $(\{t_w\}, \psi^T)$ is a deduction rule in D, and thus $(M', (w, \{t_w\} \cup obs(w))) \models X\psi$. If φ is $K\psi$, consider an arbitrary w' such that $(w'', \{t_{w'}\} \cup obs(w')) \sim (w, \{t_w\} \cup obs(w))$. This certainly implies, by the assumptions on the encoding, that $t_w = t_{w'}$, and thus $w' \in \mathcal{K}(w)$. By the fact that $(M, w) \models K\psi$, we have $(M, w') \models \psi$, and by the induction hypothesis, $(M, \{t_{w'}\} \cup obs(w')) \models \tilde{\psi}$. Since w' was arbitrary, we get $(M, \{t_w\} \cup obs(w)) \models K\tilde{\psi}$, as required. □

Theorem 3.7. *The problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}(\Sigma)$ is NP-complete.*

Proof. For the lower bound, we show how to reduce from the decision problem of $\mathcal{L}^{\mathrm{K}}$. Let f be a formula of $\mathcal{L}^{\mathrm{K}}$. By Lemma B.4, f is satisfiable if and only if $\hat{f}$ is satisfiable in $\mathcal{M}^{\mathrm{ded}}(\Sigma)$. Thus, the complexity of the decision problem for $\mathcal{L}^{\mathrm{K}}$ is a lower bound for our decidability problem, that is, NP. For the upper bound, we need to exhibit a nondeterministic polynomial time algorithm that decides if $\varphi \in \mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable. We will use the decision problem for $\mathcal{L}^{\mathrm{K}}$ itself as an algorithm. By Lemma B.5, φ is satisfiable if and only if $\tilde{\varphi}$ is satisfiable, so we can simply invoke the NP algorithm for $\mathcal{L}^{\mathrm{K}}$ satisfiability on $\tilde{\varphi}$. □

Theorem 3.8. *For any given propositional deductive system D that is decidable in polynomial time, the problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ is NP-complete.*

Proof. The lower bound follows from Lemma B.4: let f be an $\mathcal{L}^{\mathrm{K}}$ formula, and f is satisfiable if and only if $\hat{f}$ is satisfiable over $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ structures. Since the decision problem for $\mathcal{L}^{\mathrm{K}}$ is NP-complete, the lower bound follows.

For the upper bound, we can do something similar to what we did in Theorem 3.7, except we need to keep track of the size of the objects we manipulate. Let φ be a formula of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$. We exhibit an algorithm that nondeterministically decides if φ is satisfiable. First, we prove a small model theorem for $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ over $\mathcal{M}^{\mathrm{ded}}_D(\Sigma)$: if φ is satisfiable in $M \in \mathcal{M}^{\mathrm{ded}}_D(\Sigma)$, then φ is satisfiable in a structure $M' \in \mathcal{M}^{\mathrm{ded}}_D(\Sigma)$ with the set of worlds in M' polynomial in $|\varphi|$. Assume φ is satisfiable in some structure M. Let $M_1 = (W_1, \mathcal{K}_1, \pi_1)$ be the epistemic

structure obtained by the construction in Lemma B.5, with $(M_1, w_1) \models_K \tilde{\varphi}$, for some $w_1 = (e, obs)$ in W_1. By Lemma B.1, we know that $\tilde{\varphi}$ is satisfied in an epistemic structure $M_2 = (W_2, \mathcal{K}_2, \pi_2)$ where $|W_2| \leq |\tilde{\varphi}|$, $\mathcal{K}_2$ is a universal relation on W_2 (that is, $\mathcal{K}_2 = W_2 \times W_2$), and $(M_2, w_2) \models_K \tilde{\varphi}$ for some $w_2 \in W_2$. We reconstruct a satisfying deductive algorithmic knowledge structure from M_2. Specifically, define $M' = (W', \pi', D)$ by taking $W' = \{(w, obs) \mid w \in W_2\}$ (where obs is the set of observations from w_1), and $\pi'((w, obs))(t) = \pi_2(w)(p_t)$ when t is not in Obs, and $\pi'(w)(ob) =$ **true** if and only if $ob \in obs$. Clearly, π' respects observations. A straightforward induction on the structure of φ shows that if $(M, w) \models \varphi$ (or equivalently, by Lemma B.5, $(M_1, w_1) \models_K \tilde{\varphi}$ for some w_1), then $(M', (w_2, obs)) \models \varphi$, for some w_2. Here are the interesting cases of the induction. If φ is $t \in Obs$, then $(M_1, w_1) \models p_t$, with $t \in obs$, where $w_1 = (e, obs)$, which immediately yields that $\pi'((w_2, obs))(t) =$ **true**, and thus $(M', (w_2, obs)) \models t$. If φ is $t \notin Obs$, then $(M_1, w_1) \models p_t$, and $\pi_1(w_1)(p_t) =$ **true**; this means that $\pi_2(w_2)(p_t) =$ **true** (by construction of M_2), so that $\pi'((w_2, obs))(t) =$ **true**, and $(M', (w_2, obs)) \models t$. If φ is $X\psi$, then by the fact that $(M, s) \models X\psi$, and that $s = (e, obs)$ where $obs = \{ob_1, \ldots, ob_n\}$, we have $ob_1, \ldots, ob_n \vdash_D \psi^T$, and thus, $(M', (w_2, obs)) \models X\psi$, since the same observations are used. Finally, if φ is $K\psi$, then consider an arbitrary w' such that $(w', obs) \sim (w_2, obs)$; since all states have the same observations, w' can be arbitrary in W_2. Since $\mathcal{K}_2$ was the universal relation on W_2, we have $w' \in \mathcal{K}_2(w_2)$. By assumption, we know $(M_1, w_1) \models_K K\tilde{\psi}$, and thus $(M_2, w_2) \models_K K\tilde{\psi}$, so that $(M_2, w') \models_K \tilde{\psi}$. By the induction hypothesis, $(M', (w', obs)) \models \psi$, and since w' was arbitrary, $(M', (w_2, obs)) \models K\psi$, as required.

The upper bound follows directly from this result. It suffices to nondeterministically guess a satisfying structure M with a set of worlds polynomial in $|\varphi|$, which is guaranteed to exist if and only if φ is satisfiable. We can verify that φ is satisfied in M in time polynomial in $|\varphi|$, by adapting the polynomial time algorithm of Theorem 2.9. Roughly speaking, the algorithm consists of enumerating all the subformulas of φ, and for each subformula ψ (in order of length), marking the every state of M with either ψ or $\neg\psi$ depending on whether ψ or $\neg\psi$ holds at the state: primitive propositions are handled by invoking the interpretation, formulas of the form $X\psi'$ are handled by invoking the polynomial time decision procedure for the deductive system D, conjunctions and negations are handled in the obvious way, and formulas $K\psi'$ are handled by looking up whether every reachable state from the current state is marked with ψ'. □

Corollary 3.9. *For any local KD deductive system D, the problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}(T^g_\Sigma)$ is satisfiable in a structure in $\mathfrak{M}^{\mathrm{ded}}_D(\Sigma)$ is NP-complete.*

Proof. Immediate from the property of local deductive system, and from Theorem 3.8. □

Theorem 3.11. *The axiomatization* $\mathrm{AX}_n^{\mathrm{ded}}$ *is sound and complete for* $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$ *with respect to* $\mathcal{M}_n^{\mathrm{ded}}(\Sigma)$.

Proof. This is a straightforward generalization of the proof of Theorem 3.5. Soundness is easy to check. For completeness, we again show that if φ is consistent, then φ is satisfiable. We give the definitions here, leaving the details of the proof to the reader. Given a set V of formulas, let $V/K_i = \{\varphi \mid K_i\varphi \in V\}$. Let $\mathcal{C}$ be the set of all maximal consistent sets of formulas of $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$. For $V \in \mathcal{C}$, let $Obs_i(V) = \{ob \in Obs_i \mid ob \in V\}$. We define $\approx_i$ over $\mathcal{C}$, for every i, by taking $V \approx_i W$ if and only if $V/K_i \subseteq W$. We can check that $\approx_i$ is an equivalence relation for every i, assuming the axioms K1–K5, just like in the proof of Theorem 3.5. We can also check that if $V \approx_i W$, then $V \sim_i W$, and that for all ψ, if $V \approx_i W$, then $X_i\psi \in V$ if and only if $X_i\psi \in W$.

Let φ be a consistent formula of $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$, and let $Sub(\varphi)$ be the set of subformulas of φ (including φ itself). Since φ is consistent, there is a set $V^\varphi \in \mathcal{C}$ with $\varphi \in V^\varphi$ and $|Obs_i(V^\varphi)| < \infty$ for every i. For every i, let $obs_i^\varphi = Obs_i(V^\varphi)$, and let $[V^\varphi]_{\approx_i}$ be the $\approx_i$-equivalence class that contains V^φ. We will use $[V^\varphi]_{\approx_1} \cap \cdots \cap [V^\varphi]_{\approx_n}$ to define the states of our canonical structure. More specifically, define the canonical deductive algorithmic knowledge structure $M^\varphi = (W^\varphi, \pi^\varphi, D_1^\varphi, \ldots, D_n^\varphi)$ by taking

$$W^\varphi = \{(w_V, obs_1^\varphi, \ldots, obs_n^\varphi) \mid V \in [V^\varphi]_{\approx_1} \cap \cdots \cap [V^\varphi]_{\approx_n}\}$$

$$\pi^\varphi((w_V, obs_1^\varphi, \ldots, obs_n^\varphi))(p) = \begin{cases} \textbf{true} & \text{if } p \in V \\ \textbf{false} & \text{if } p \notin V \end{cases}$$

$$D_i^\varphi = \{(\varnothing, \psi^T) \mid X_i\psi \in Sub(\varphi), X_i\psi \in V^\varphi, \psi^T \notin Obs_i\}.$$

We can check that M^φ is a deductive algorithmic knowledge structure with n agents.

We can prove, adapting the proof of Theorem 3.5, that for all $w_V \in W^\varphi$ and all subformulas $\psi \in Sub(\varphi)$, $(M^\varphi, w_V) \models \psi$ if and only if $\psi \in V$. Completeness follows from the fact that $\varphi \in V^\varphi$ and $\varphi \in Sub(\varphi)$, so that $(M^\varphi, w_V) \models \varphi$, and thus φ is satisfiable. □

Theorem 3.12. *The axiomatization* $\mathrm{AX}_n^{\mathrm{ded}} + \{Ax_n^{D_1}, \ldots, Ax_n^{D_n}\}$ *is sound and complete for* $\mathcal{L}_n^{\mathrm{KX}}(T_\Sigma^g)$ *with respect to* $\mathcal{M}_{D_1,\ldots,D_n\subseteq}^{\mathrm{ded}}(\Sigma)$.

Proof. Soundness is again straightforward. For completeness, we prove the equivalent statement that if φ is consistent then φ is satisfiable in some structure in $\mathcal{M}^{\mathrm{ded}}_{D_1,\ldots,D_n}(\Sigma)$. The procedure is exactly the one that is used to prove Theorem 3.11, except that we construct the deductive systems $D^\varphi_1,\ldots,D^\varphi_n$ differently.

We simply indicate where the proof differs from that of Theorem 3.11, and let the reader fill in the details. We construct, for a given φ, a deductive algorithmic knowledge structure with n agents $M^\varphi = (W^\varphi, \pi^\varphi, D^\varphi_1,\ldots,D^\varphi_n)$, where W^φ and π^φ are constructed as in Theorem 3.11, and $D^\varphi_1,\ldots,D^\varphi_n$ are obtained by taking D^φ_i to be

$$D_i \cup \{(\varnothing, \psi^T) \mid X_i\psi \in \mathit{Sub}(\varphi), X_i\psi \in V^\varphi, \psi^T \notin \mathit{Obs}_i\}.$$

M^φ is a deductive algorithmic knowledge structure in $\mathcal{M}^{\mathrm{ded}}_{D_1,\ldots,D_n\subseteq}(\Sigma)$. As in the proof of Theorem 3.6, we can show that if $obs^\varphi \vdash_{D_i} \psi^T$, then there is a rule $\triangleright\psi^T$ in D^φ_i. The rest of the proof follows that of Theorem 3.11. □

Theorem 3.13. *If $n \geq 2$, the problem of deciding whether a formula φ of $\mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma)$ is satisfiable in a structure in $\mathcal{M}^{\mathrm{ded}}_n(\Sigma)$ is PSPACE-complete.*

Proof. The proof is entirely analogous to that of Theorem 3.7, except that we use the modal logic $\mathcal{L}^{\mathrm{K}}_n$ rather than $\mathcal{L}^{\mathrm{K}}$. We can define translations between $\mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma)$ and $\mathcal{L}^{\mathrm{K}}_n$, and we can prove analogues of Lemmas B.4 and B.5. We simply give the translations here, leaving the reader to fill in the details.

Let f be a formula of $\mathcal{L}^{\mathrm{K}}_n$. Let $p_1,\ldots,p_k$ be the primitive propositions appearing in f. We first come up with an encoding of these primitive propositions into the language of Σ. For example, we can take p_1 to be true, p_2 to be $\mathsf{not}(\mathsf{true})$, p_3 to be $\mathsf{not}(\mathsf{not}(\mathsf{true}))$, and so forth. Let t_p be the term encoding the primitive proposition p. We again make the restriction on this encoding that no t_p is an observation in $\mathit{Obs}_1,\ldots,\mathit{Obs}_n$. Let $\hat{f}$ be the formula obtained by replacing every instance of a primitive proposition p in f by t_p. Note that $|\hat{f}|$ is polynomial in $|f|$, and that $\hat{f}$ contains no instance of the X operator. We can show that f is satisfiable in epistemic structures for n agents if and only if $\hat{f}$ is satisfiable in $\mathcal{M}^{\mathrm{ded}}_n(\Sigma)$, with a proof similar to that of Lemma B.4. This gives us an immediate lower bound, as follows. Let f be an $\mathcal{L}^{\mathrm{K}}_n$ formula. We know f is satisfiable if and only if $\hat{f}$ is satisfiable over $\mathcal{M}^{\mathrm{ded}}_n(\Sigma)$ structures. Since the decision problem for $\mathcal{L}^{\mathrm{K}}_n$ ($n \geq 2$) is PSPACE-complete, the lower bound of PSPACE follows.

Let φ be a formula of $\mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma)$. The set T^g_Σ is countable, so let $\{p_t \mid t \in T^g_\Sigma\}$ be a countable set of primitive propositions corresponding to the ground terms of T_Σ. Similarly, for every i, the set of formulas $\{X_i\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma), \psi^T \notin \mathit{Obs}_i\}$ is countable, so let $\{q^i_\psi \mid \psi \in \mathcal{L}^{\mathrm{KX}}_n(T^g_\Sigma), \psi^T \notin \mathit{Obs}_i\}$ be a countable set of primitive propositions where q^i_ψ corresponds to the formula $X_i\psi$. Let $\tilde{\varphi}$ be the translation of

φ obtained by replacing every occurrence of a term t in T^g_Σ by p_t, every occurrence of a formula $X_i\psi$ where $\psi^T \in Obs_i$ by p_{ψ^T}, and every occurrence of a formula $X_i\psi$ where $\psi^T \notin Obs_i$ by the corresponding q^i_ψ, in conjunction with formulas $p_{ob} \Leftrightarrow K_i p_{ob}$ for all observations $ob \in Obs_i$ appearing in φ. Note that $|\tilde{\varphi}|$ is polynomial in $|\varphi|$. We can show that φ is satisfiable in $\mathcal{M}^{\mathrm{ded}}_n(\Sigma)$ if and only if $\tilde{\varphi}$ is satisfiable in an epistemic structure for n agents, using a proof similar to that of Lemma B.5. This gives us an immediate upper bound for our decision problem: φ is satisfiable if and only if $\tilde{\varphi}$ is satisfiable, so we can simply invoke the PSPACE algorithm for $\mathcal{L}^{\mathrm{K}}_n$ satisfiability on $\tilde{\varphi}$. □

B.3 Proofs for Chapter 4

Theorem 4.1. *Let $N = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}^d_1, \ldots, \mathtt{A}^d_n, \nu)$ be a probabilistic algorithmic knowledge structure, where $\mathtt{A}_1, \ldots, \mathtt{A}_n$ are deterministic. Let $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$. If there are no occurrences of* Pr *in φ, then for all $w \in W$ and all $v \in V$, $(N, w, v) \models \varphi$ if and only if $(M, w) \models \varphi$.*

Proof. The key observation here is that if a knowledge algorithm $\mathtt{A}$ is deterministic, then for all $v \in V$, $\mathtt{A}^d(\varphi, \ell, v_i) = \mathtt{A}(\varphi, \ell)$. The result then follows easily by induction on the structure of φ. If φ is p, then $(N, w, v) \models p$ if and only if $\pi(w)(p) =$ **true** if and only if $(M, w) \models p$. If φ is $\psi_1 \wedge \psi_2$, then $(N, w, v) \models \psi_1 \wedge \psi_2$ if and only if $(N, w, v) \models \psi_1$ and $(N, w, v) \models \psi_2$ if and only if $(M, w) \models \psi_1$ and $(M, w) \models \psi_2$ (by the induction hypothesis) if and only if $(M, w) \models \psi_1 \wedge \psi_2$. If φ is $\neg\psi$, then $(N, w, v) \models \neg\psi$ if and only if $(N, w, v) \not\models \psi$ if and only $(M, w) \not\models \psi$ (by the induction hypothesis) if and only if $(M, w) \models \neg\psi$. If φ is $K_i\psi$, consider first $(N, w, v) \models K_i\psi$, that is, for all $v' \in V$ and all $w' \sim_i w$, $(N, w', v') \models \psi$; by the induction hypothesis, this means that for all $w' \sim_i w$, $(M, w') \models \psi$, that is, $(M, w) \models K_i\psi$. Conversely, assume $(M, w) \models K_i\psi$, so that for all $w' \sim_i w$, $(M, w') \models \psi$; by the induction hypothesis, for every $w' \sim_i w$, we have $(N, w', v') \models \psi$ for all $v' \in V$, and thus $(N, w, v) \models K_i\psi$. If φ is $X_i\psi$, then $(N, w, v) \models X_i\psi$ if and only if $\mathtt{A}^d_i(\psi, \mathcal{V}_i(w), v_i^{\psi}) =$ "Yes" if and only if $\mathtt{A}_i(\psi, \mathcal{V}_i(w)) =$ "Yes" (since $\mathtt{A}_i$ is deterministic) if and only if $(M, w) \models X_i\psi$. □

Theorem 4.2. *Let $N = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}^d_1, \ldots, \mathtt{A}^d_n, \nu)$ be a probabilistic algorithmic knowledge structure, and let $M = (W, \mathcal{V}_1, \ldots, \mathcal{V}_n, \pi, \mathtt{A}'_1, \ldots, \mathtt{A}'_n)$ be an algorithmic knowledge structure where $\mathtt{A}'_1, \ldots, \mathtt{A}'_n$ are arbitrary deterministic knowledge algorithms. If there are no occurrences of X_i and* Pr *in φ, then for all $w \in W$ and all $v \in V$, $(N, w, v) \models \varphi$ if and only if $(M, w) \models \varphi$.*

Proof. This result in fact follows from the proof of Theorem 4.1, since the only use of the assumption that knowledge algorithms are deterministic is in the inductive step for subformulas of the form $X_i\psi$. □

Theorem 4.3. *For all ob, we have $w_{\mathcal{E}}(ob, h_i) \geq w_{\mathcal{E}}(ob, h_{2-i})$ if and only if $l(ob, h_i) \geq l(ob, h_{2-i})$, for $i = 1, 2$, and for all h, ob, and ob', we have $w_{\mathcal{E}}(ob, h) \geq w_{\mathcal{E}}(ob', h)$ if and only if $l(ob, h) \geq l(ob', h)$.*

Proof. Let ob be an arbitrary observation. The result follows from the following observation:

$$
\begin{aligned}
& w_{\mathcal{E}}(ob, h_i) \geq w_{\mathcal{E}}(ob, h_{2-i}) \\
& \quad \text{iff } \mu_{h_i}(ob)/(\mu_{h_i}(ob) + \mu_{h_{2-i}}(ob)) \geq \mu_{h_{2-i}}(ob)/(\mu_{h_i}(ob) + \mu_{h_{2-i}}(ob)) \\
& \quad \text{iff } \mu_{h_i}(ob)\mu_{h_i}(ob) \geq \mu_{h_{2-i}}(ob)\mu_{h_{2-i}}(ob) \\
& \quad \text{iff } \mu_{h_i}(ob)/\mu_{h_{2-i}}(ob) \geq \mu_{h_{2-i}}(ob)/\mu_{h_i}(ob) \\
& \quad \text{iff } l(ob, h_i) \geq l(ob, h_{2-i}).
\end{aligned}
$$

A similar argument establishes the result for hypotheses. □

Theorem 4.5. *For all probabilistic algorithmic knowledge structures N, worlds w of N, and derandomizers $v \in V$, $w_{\mathcal{E}_{\mathtt{A}_i,\varphi,\ell}}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner, \varphi)$ is defined.*

Proof. Given N, w a world of N, and a derandomizer v, $(N, w, v) \models \mathrm{Ev}_i(\varphi)$ is defined if and only if $\mu_{\varphi,\mathcal{V}_i(w)}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner) + \mu_{\neg\varphi,\mathcal{V}_i(w)}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner) > 0$. This condition holds if and only if at least one of $\mu_{\varphi,\mathcal{V}_i(w)}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner) > 0$ or $\mu_{\neg\varphi,\mathcal{V}_i(w)}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner) > 0$ holds.

Suppose that $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "Yes", so that $\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner = \{$"Yes"$\}$. Clearly, either $W_{\varphi,\mathcal{V}_i(w)} \neq \varnothing$, or $W_{\neg\varphi,\mathcal{V}_i(w)} \neq \varnothing$. If $W_{\varphi,\mathcal{V}_i(w)} \neq \varnothing$, then by our assumption about derandomizers, $\mu_{\varphi,\mathcal{V}_i(w)} = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') =$ "Yes"$\}) > 0$, since v is such a derandomizer. Similarly, if $W_{\neg\varphi,\mathcal{V}_i(w)} \neq \varnothing$, then by our assumption about derandomizers, $\mu_{\varphi,\mathcal{V}_i(w)} = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') =$ "Yes"$\}) > 0$. In either case, $\mu_{\varphi,\ell}($"Yes"$) > 0$, as desired. A similar argument applies if $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "No" or $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "?". □

Theorem 4.6. *For all probabilistic algorithmic knowledge structures N, we have*

$$N \models \mathrm{Ev}_i(\varphi) = 1 \Rightarrow K_i\varphi.$$

Proof. Assume $(N, w, v) \models \mathrm{Ev}_i(\varphi) = 1$, so $w_{\mathcal{E}_{\mathtt{A}_i,\varphi,\mathcal{V}_i(w)}}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner, \varphi) = 1$. By definition of $w_{\mathcal{E}_{\mathtt{A}_i,\varphi,\mathcal{V}_i(w)}}$, this means that $\mu_{\neg\varphi,\mathcal{V}_i(w)}(\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i)\urcorner) = 0$.

First, we establish that if $(N, w, v) \models \mathrm{Ev}_i(\varphi) = 1$, it must be the case that

$\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "Yes". If not, then $\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner = \{$"No", "?"$\}$. By assumption, $\mu_{\neg\varphi,\mathcal{V}_i(w)}(\{$"No", "?"$\}) = 0$, so that $\mu_{\neg\varphi,\mathcal{V}_i(w)}($"Yes"$) = 1$, and thus $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') =$ "Yes"$\}) = 1$. Equivalently, $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') \neq$ "Yes"$\}) = 0$, so that $\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') \neq$ "Yes"$\} = \varnothing$, by our assumption on ν. But this contradicts the fact that v is just such a derandomizer.

Therefore, we must have $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "Yes", so that $\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner =$ {"Yes"}. Since $\mu_{\neg\varphi,\mathcal{V}_i(w)}($"Yes"$) = 0$, we must have $\{(w', v') \mid \mathcal{V}_i(w') = \mathcal{V}_i(w), (N, w', v') \models \neg\varphi\} = \varnothing$. (Otherwise, $\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') =$ "Yes"$\}) = 0$, which means that $\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') =$ "Yes"$\} = \varnothing$, contradicting the fact that v is just such a derandomizer.) Let (w', v') be such that $\mathcal{V}_i(w') = \mathcal{V}_i(w)$. By the above, we cannot have $(N, w', v') \models \neg\varphi$, and thus $(N, w', v') \models \varphi$. This simply means that $(N, w, v) \models K_i\varphi$, as desired. □

To prove Theorems 4.7, 4.9, and 4.10, the following lemma, alluded to in the text, is useful. It describes the relationship between the reliability of a knowledge algorithm and the properties of the probability measures $\mu_{\varphi,\ell}$ and $\mu_{\neg\varphi,\ell}$ appearing in the evidence space $\mathcal{E}_{\mathtt{A}_i,\varphi,\ell}$, as defined in Section 4.3.

Lemma B.6. *If $\mathtt{A}_i$ is (α, β)-reliable for φ in N, then we have $\mu_{\varphi,\ell}($"Yes"$) \geq \alpha$, $\mu_{\varphi,\ell}(\{$"No", "?"$\}) \leq 1-\alpha$, $\mu_{\neg\varphi,\ell}($"Yes"$) \leq \beta$, and $\mu_{\neg\varphi,\ell}(\{$"No", "?"$\}) \geq 1-\beta$ for all local states ℓ of agent i in N.*

Proof. Let ℓ be a local state of agent i in N, and let $W_\ell = \{(w, v) \mid \mathcal{V}_i(w) = \ell\}$, which is nonempty by assumption. Note that $W_\ell = W_{\varphi,\ell} \cup W_{\neg\varphi,\ell}$.

Consider $\mu_{\varphi,\ell}($"Yes"$)$. If $W_{\varphi,\ell} = \varnothing$, $\mu_{\varphi,\ell}($"Yes"$) = 1 \geq \alpha$. If $W_{\varphi,\ell} \neq \varnothing$, let $(w, v) \in W_{\varphi,\ell}$, so that $(N, w, v) \models \varphi$, and since $\mathtt{A}_i$ is (α, β)-reliable, $\mu_{\varphi,\ell}($"Yes"$) = \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \ell, v_i') =$ "Yes"$\}) \geq \alpha$. Thus, $\mu_{\varphi,\ell}(\{$"No", "?"$\}) = 1 - \mu_{\varphi,\ell}($"Yes"$) \leq 1 - \alpha$.

A completely symmetric argument applies for $\mu_{\neg\varphi,\ell}$. We leave the details to the reader. □

The following lemma captures the algebraic relationships that are useful in assessing the evidence in Theorems 4.7, 4.9, and 4.10.

Lemma B.7. *Suppose x, y, a, b are real numbers in $[0, 1]$, such that $(x, y) \neq (0, 0)$, and $(a, b) \neq (0, 0)$. If $x \geq a$ and $y \leq b$, then $x/(x + y) \geq a/(a + b)$ and $y/(x + y) \leq b/(a + b)$.*

Proof. Note that $x(a + b) = xa + xb \geq xa + ay = a(x + y)$, so that $x/(x + y) \geq a/(a + b)$. Similarly, $y(a + b) = ya + yb \leq xb + yb = b(x + y)$, so that $y/(x + y) \leq b(a + b)$. □

Theorem 4.7. *If* $\mathtt{A}_i$ *is* (α, β)*-reliable for* φ *in* N *then*

(a) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta}$ *if* $(\alpha, \beta) \neq (0, 0)$*;*
(b) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0, 0)$*;*
(c) $N \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $(\alpha, \beta) \neq (1, 1)$*;*
(d) $N \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 0$ *if* $(\alpha, \beta) = (1, 1)$.

Proof. For part (a), suppose that $(\alpha, \beta) \neq (0, 0)$. Let w be a world of N and let v be a derandomizer. If $(N, w, v) \models X_i\varphi$, then $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "Yes", so $\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner = \{\text{“Yes”}\}$. By Lemma B.6, we have $\mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“Yes”}\}) \geq \alpha$ and $\mu_{\neg\varphi, \mathcal{V}_i(w)}(\{\text{“Yes”}\}) \leq \beta$. Therefore, by Lemma B.7, we have

$$\begin{aligned} &w_{\mathcal{E}_{\mathtt{A}_i, \varphi, \mathcal{V}_i(w)}}(\{\text{“Yes”}\}, \varphi) \\ &\quad = \mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“Yes”}\})/(\mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“Yes”}\}) + \mu_{\neg\varphi, \mathcal{V}_i(w)}(\{\text{“Yes”}\})) \\ &\quad \geq \alpha/(\alpha + \beta). \end{aligned}$$

Thus, we have $(N, w, v) \models \mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha + \beta)$. Since w and v were arbitrary, we have $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha + \beta)$.

For part (c), suppose that $(\alpha, \beta) \neq (1, 1)$, so that $(1 - \alpha, 1 - \beta) \neq (0, 0)$. If $(N, w, v) \models \neg X_i\varphi$, then either $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "No" or $\mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) =$ "?". Thus, we have $\ulcorner \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i) \urcorner = \{\text{“No”}, \text{“?”}\}$. By Lemma B.6, we have $\mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“No”}, \text{“?”}\}) \leq 1 - \alpha$ and $\mu_{\neg\varphi, \mathcal{V}_i(w)}(\{\text{“No”}, \text{“?”}\}) \geq 1 - \beta$. Therefore, by Lemma B.7, we have

$$\begin{aligned} &w_{\mathcal{E}_{\mathtt{A}_i, \varphi, \mathcal{V}_i(w)}}(\{\text{“No”}, \text{“?”}\}, \varphi) \\ &\quad = \mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“No”}, \text{“?”}\})/(\mu_{\varphi, \mathcal{V}_i(w)}(\{\text{“No”}, \text{“?”}\}) + \mu_{\neg\varphi, \mathcal{V}_i(w)}(\{\text{“No”}, \text{“?”}\})) \\ &\quad \leq 1 - \alpha/(2 - (\alpha + \beta)). \end{aligned}$$

Thus, we have $(N, w, v) \models \mathrm{Ev}_i(\varphi) \leq 1 - \alpha/(2 - (\alpha + \beta))$. Since w and v were arbitrary, we have

$$N \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq 1 - \alpha/(2 - (\alpha + \beta)).$$

The proofs of parts (b) and (d) are similar and left to the reader. □

Theorem 4.8. *If* $\mathtt{A}_i$ *weakly respects negation and* $\mathtt{A}_i$ *is* (α, β)*-reliable for* φ *in* N*, then* $\mathtt{A}_i$ *is* $(0, 1 - \alpha)$*-reliable for* $\neg\varphi$ *in* N*. If* $\mathtt{A}_i$ *strongly respects negation, then* $\mathtt{A}_i$ *is* (α, β)*-reliable for* φ *in* N *if and only if* $\mathtt{A}_i$ *is* $(1 - \beta, 1 - \alpha)$*-reliable for* $\neg\varphi$ *in* N.

Proof. Suppose that $\mathtt{A}_i$ weakly respects negation and is (α, β)-reliable for φ in N. Consider the reliability of $\mathtt{A}_i$ with respect to $\neg\varphi$. If $(N, w, v) \models \varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“No”}\}) - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“?”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“?”}\}) \\
&\quad \leq 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad \leq 1 - \alpha.
\end{aligned}$$

Thus, $\mathtt{A}_i$ is $(0, 1-\alpha)$-reliable for $\neg\varphi$ in N.

Now suppose that $\mathtt{A}_i$ strongly respects negation and is (α, β)-reliable for φ in N. Consider the reliability of $\mathtt{A}_i$ with respect to $\neg\varphi$. If $(N, w, v) \models \neg\varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad \geq 1 - \beta.
\end{aligned}$$

Similarly, if $(N, w, v) \models \varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}) \\
&\quad \leq 1 - \alpha.
\end{aligned}$$

Thus, $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N.

Conversely, suppose that $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N, and consider the reliability of $\mathtt{A}_i$ with respect to φ. If $(N, w, v) \models \varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') \neq \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad \geq 1 - (1 - \alpha) \\
&\quad = \alpha.
\end{aligned}$$

Similarly, if $(N, w, v) \models \neg\varphi$, then

$$\begin{aligned}
&\nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\quad = 1 - \nu(\{v' \mid \mathtt{A}_i^d(\varphi, \mathcal{V}_i(w), v_i') \neq \text{“Yes”}\})
\end{aligned}$$

$$
\begin{aligned}
&= 1 - \nu(\{v' \mid \mathtt{A}_i^d(\neg\varphi, \mathcal{V}_i(w), v_i') = \text{“Yes”}\}) \\
&\leq 1 - (1-\beta) \\
&= \beta.
\end{aligned}
$$

Thus, $\mathtt{A}_i$ is (α, β)-reliable for φ in N. □

To prove Theorems 4.9 and 4.10, we need a result similar to Theorem 2.6.

Lemma B.8. *If N is a probabilistic algorithmic knowledge structure where agent i uses a knowledge algorithm $\mathtt{A}_i$ that weakly (resp., strongly) respects negation, then the formula $X_i\varphi \Rightarrow \neg X_i\neg\varphi$ (resp., $X_i\varphi \Leftrightarrow \neg X_i\neg\varphi$) is valid in N.*

Proof. A straightforward adaptation of the proof of Theorem 2.6. □

Theorem 4.9. *If $\mathtt{A}_i$ is (α, β)-reliable for φ in N and $\mathtt{A}_i$ weakly respects negation, then*

(a) $N \models X_i\varphi \Rightarrow \left(\mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \mathrm{Ev}_i(\neg\varphi) \leq \frac{1}{1+\alpha}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*
(b) $N \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) = 1$ *if* $(\alpha, \beta) = (0, 0)$*;*
(c) $N \models X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}$ *if* $\alpha \neq 1$*;*
(d) $N \models X_i\neg\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) = 1 \wedge \mathrm{Ev}_i(\varphi) = 0)$ *if* $\alpha = 1$.

Proof. Suppose that $\mathtt{A}_i$ is (α, β)-reliable for φ in N. For part (a), suppose that $(\alpha, \beta) \neq 0$. Since $\mathtt{A}_i$ weakly respects negation, by Theorem 4.8, $\mathtt{A}_i$ is $(0, 1-\alpha)$-reliable for $\neg\varphi$ in N. Let w be a world of N and let v be a derandomizer. By Theorem 4.7 applied to φ, $(N, w, v) \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta)$. By Lemma B.8, $(N, w, v) \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. By Theorem 4.7 applied to $\neg\varphi$, $(N, w, v) \models \neg X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq (1-0)/(1-(1-\alpha)+1-0)$, that is, $(N, w, v) \models \neg X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq 1/(1+\alpha)$. Putting this together, we get

$$(N, w, v) \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq 1/(1+\alpha)).$$

Since w and v are arbitrary,

$$N \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq 1/(1+\alpha)).$$

For part (c), suppose that $\alpha \neq 1$. By Lemma B.8, $(N, w, v) \models X_i\neg\varphi \Rightarrow \neg X_i\varphi$. By Theorem 4.7 applied to φ, $(N, w, v) \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq (1-\alpha)/(2-(\alpha+\beta))$. Thus, we get

$$(N, w, v) \models X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq (1 - \alpha/(2-(\alpha+\beta))).$$

Since w and v are arbitrary, it follows that

$$N \models X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq (1 - \alpha/(2-(\alpha+\beta))).$$

The proof of parts (b) and (d) are similar and left to the reader. □

Theorem 4.10. *If* $\mathtt{A}_i$ *is* (α, β)*-reliable for* φ *in* N *and* $\mathtt{A}_i$ *strongly respects negation, then*

(a) $N \models X_i\varphi \Rightarrow \left(\mathrm{Ev}_i(\varphi) \geq \frac{\alpha}{\alpha+\beta} \wedge \mathrm{Ev}_i(\neg\varphi) \leq \frac{\beta}{\alpha+\beta}\right)$ *if* $(\alpha, \beta) \neq (0, 0)$*;*

(b) $N \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) = 1 \wedge \mathrm{Ev}_i(\neg\varphi) = 0)$ *if* $(\alpha, \beta) = (0, 0)$*;*

(c) $N \models X_i\neg\varphi \Rightarrow \left(\mathrm{Ev}_i(\neg\varphi) \geq \frac{1-\beta}{2-(\alpha+\beta)} \wedge \mathrm{Ev}_i(\varphi) \leq \frac{1-\alpha}{2-(\alpha+\beta)}\right)$ *if* $(\alpha, \beta) \neq (1, 1)$*;*

(d) $N \models X_i\neg\varphi \Rightarrow \left(\mathrm{Ev}_i(\neg\varphi) \geq \frac{1}{2} \wedge \mathrm{Ev}_i(\varphi) \leq \frac{1}{2}\right)$ *if* $(\alpha, \beta) = (1, 1)$.

Proof. Suppose that $\mathtt{A}_i$ is (α, β)-reliable for φ in N. For part (a), suppose that $(\alpha, \beta) \neq (0, 0)$. Since $\mathtt{A}_i$ strongly respects negation, by Theorem 4.8, $\mathtt{A}_i$ is $(1-\beta, 1-\alpha)$-reliable for $\neg\varphi$ in N. Let w be a world of N and let v be a derandomizer. By Theorem 4.7 applied to φ, $(N, w, v) \models X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta)$. By Lemma B.8, $(N, w, v) \models X_i\varphi \Rightarrow \neg X_i\neg\varphi$. By Theorem 4.7 applied to $\neg\varphi$, $(N, w, v) \models \neg X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq (1-(1-\beta))/(1-(1-\alpha)+1-(1-\beta))$, that is, $(N, w, v) \models \neg X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta)$. Putting this together, we get

$$(N, w, v) \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta)).$$

Since w and v are arbitrary,

$$N \models X_i\varphi \Rightarrow (\mathrm{Ev}_i(\varphi) \geq \alpha/(\alpha+\beta) \wedge \mathrm{Ev}_i(\neg\varphi) \leq \beta/(\alpha+\beta)).$$

For part (c), suppose that $(\alpha, \beta) \neq (1, 1)$. By Theorem 4.7 applied to $\neg\varphi$, $(N, w, v) \models X_i\neg\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \geq (1-\beta)/(2-(\alpha+\beta))$. By Lemma B.8, $(N, w, v) \models X_i\neg\varphi \Rightarrow \neg X_i\varphi$. By Theorem 4.7 applied to φ, $(N, w, v) \models \neg X_i\varphi \Rightarrow \mathrm{Ev}_i(\varphi) \leq (1-\alpha)/(2-(\alpha+\beta))$. Putting this together, we get

$$(N, w, v) \models X_i\neg\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) \geq (1-\beta)/(2-(\alpha+\beta)) \wedge \mathrm{Ev}_i(\varphi) \leq (1-\alpha/(2-(\alpha+\beta)))).$$

Since w and v are arbitrary,

$$N \models X_i\neg\varphi \Rightarrow (\mathrm{Ev}_i(\neg\varphi) \geq (1-\beta)/(2-(\alpha+\beta)) \wedge \mathrm{Ev}_i(\varphi) \leq (1-\alpha/(2-(\alpha+\beta)))).$$

Again, we leave the proof of parts (b) and (d) to the reader. □

B.4 Proofs for Chapter 5

Theorem 5.3. *Let $\mathcal{H} = \{h_1, \ldots, h_m\}$ and $\mathcal{O} = \{ob_1, \ldots, ob_n\}$, and let f be a real-valued function with domain $\mathcal{O} \times \mathcal{H}$ such that $f(ob, h) \in [0, 1]$. Then there exists an evidence space $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \mu_{h_1}, \ldots, \mu_{h_m})$ such that $f = w_{\mathcal{E}}$ if and only if f satisfies the following properties:*

WF1. *For every $ob \in \mathcal{O}$, $f(ob, \cdot)$ is a probability measure on $\mathcal{H}$.*

WF2. *There exists $x_1, \ldots, x_n > 0$ such that, for all $h \in \mathcal{H}$, $\sum_{i=1}^{n} f(ob_i, h)x_i = 1$.*

Proof. ($\Rightarrow$) Assume that $f = w_{\mathcal{E}}$ for an evidence space $\mathcal{E} = (\mathcal{H}, \mathcal{O}, \mu_{h_1}, \ldots, \mu_{h_m})$. It is routine to verify WF1, that for a fixed $ob \in \mathcal{O}$, $w_{\mathcal{E}}(ob, \cdot)$ is a probability measure on $\mathcal{H}$. To verify WF2, note that we can simply take $x_i = \sum_{h' \in \mathcal{H}} \mu_{h'}(ob_i)$.

($\Leftarrow$) Let f be a function from $\mathcal{O} \times \mathcal{H}$ to $[0, 1]$ that satisfies WF1 and WF2. Let $x_1^*, \ldots, x_{n_{\mathsf{h}}}^*$ be the positive reals guaranteed by WF2. It is straightforward to verify that taking $\mu_h(ob_i) = f(ob_i, h)/x_i^*$ for each $h \in \mathcal{H}$ yields an evidence space $\mathcal{E}$ such that $f = w_{\mathcal{E}}$. □

The following lemmas are useful to prove the completeness of the axiomatizations in this section. These results depend on the soundness of the axiomatization $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$.

Lemma B.9. $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ *is a sound axiomatization for $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ with respect to evidential structures.*

Proof. It is easy to see that each axiom is valid in evidential structures. □

Lemma B.10. *For all hypothesis formulas ρ, if $[\![\rho]\!] = \{h_1, \ldots, h_k\}$, then $\rho \Leftrightarrow h_1 \vee \cdots \vee h_k$ is provable in $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$.*

Proof. Using Taut, we can show that ρ is provably equivalent to a formula ρ' in disjunctive normal form. Moreover, by axiom H2, we can assume without loss of generality that each of the disjuncts in ρ' consists of a single hypothesis. Thus, ρ is $h_1 \vee \cdots \vee h_k$. An easy induction on structure shows that for a hypothesis formula ρ, evidential structure M, and world w, we have that $(M, w) \models \rho$ iff $(M, w) \models h$ for some $h \in [\![\rho]\!]$. Moreover, it follows immediately from the soundness of the axiom system (Lemma B.9) that $\rho \Leftrightarrow h_1 \vee \ldots \vee h_k$ is provable iff for all evidential strucures M and worlds w, $(M, w) \models \rho$ iff $(M, w) \models h_i$ for some $i \in \{1, \ldots, k\}$. Thus, $\rho \Leftrightarrow h_1 \vee \ldots \vee h_k$ is provable iff $[\![\rho]\!] = \{h_1, \ldots, h_k\}$. □

An easy consequence of Lemma B.10 is that ρ_1 is provably equivalent to ρ_2 if and only if $[\![\rho_1]\!] = [\![\rho_2]\!]$.

Lemma B.11. *Let ρ be an hypothesis formula. The formulas*

$$\Pr(\rho) = \sum_{h \in [\![\rho]\!]} \Pr(h),$$
$$\Pr^0(\rho) = \sum_{h \in [\![\rho]\!]} \Pr^0(h), \textit{ and}$$
$$\mathrm{w}(ob, \rho) = \sum_{h \in [\![\rho]\!]} \mathrm{w}(ob, h), \textit{ for all } ob,$$

are provable in $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$.

Proof. Let $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_{n_{\mathsf{h}}}\}$ and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_{n_{\mathsf{o}}}\}$. We prove the result for $\Pr$. We proceed by induction on the size of $[\![\rho]\!]$. For the base case, assume that $|[\![\rho]\!]| = 0$. By Lemma B.10, this implies that ρ is provably equivalent to false. By Po4, $\Pr(\rho) = \Pr(\mathit{false})$, and it is easy to check that $\Pr(\mathit{false}) = 0$ is provable using Po1, Po3, and Po4, thus $\Pr(\rho) = 0$, as required. If $|[\![\rho]\!]| = n + 1 > 0$, then $[\![\rho]\!] = \{h_{i_1}, \ldots, h_{i_{n+1}}\}$, and by Lemma B.10, ρ is provably equivalent to $h_{i_1} \vee \cdots \vee h_{i_{n+1}}$. By Po4, $\Pr(\rho) = \Pr(\rho \wedge h_{i_{n+1}}) + \Pr(\rho \wedge \neg h_{i_{n+1}})$. It is easy to check that $\rho \wedge h_{i_{n+1}}$ is provably equivalent to $h_{i_{n+1}}$ (using H2), and similarly $\rho \wedge \neg h_{i_{n+1}}$ is provably equivalent to $h_{i_1} \vee \cdots \vee h_{i_n}$. Thus, $\Pr(\rho) = \Pr(h_{i_{n+1}}) + \Pr(h_{i_1} \vee \cdots \vee h_{i_n})$ is provable. Since $|[\![h_{i_1} \vee \cdots \vee h_{i_n}]\!]| = n$, by the induction hypothesis, $\Pr(h_{i_1} \vee \cdots \vee h_{i_n}) = \sum_{h \in \{h_{i_1}, \ldots, h_{i_n}\}} \Pr(h) = \sum_{h \in [\![\rho]\!] - \{h_{i_{n+1}}\}} \Pr(h)$. Thus, $\Pr(\rho) = \Pr(h_{i_{n+1}}) + \sum_{h \in [\![\rho]\!] - \{h_{i_{n+1}}\}} \Pr(h)$, that is, $\Pr(\rho) = \sum_{h \in [\![\rho]\!]} \Pr(h)$, as required.

The same argument applies *mutatis mutandis* for $\Pr^0$ and $\mathrm{w}(ob, \cdot)$, using axioms Pr1–4 and axioms E1–4 (respectively), instead of Po1–4. □

Theorem 5.5. $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ *is a sound and complete axiomatization for the logic* $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ *with respect to evidential structures.*

Proof. Soundness was established in Lemma B.9. To prove completeness, recall the following definitions. A formula φ is *consistent* with the axiom system $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ if $\neg\varphi$ is not provable from $\mathrm{AX}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. To prove completeness, it is sufficient to show that if φ is consistent, then it is satisfiable, that is, there exists an evidential structure M and valuation v such that $(M, s, \mu, v) \models \varphi$.

As in the body of the paper, let $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_{n_{\mathsf{h}}}\}$ and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_{n_{\mathsf{o}}}\}$. Let φ be a consistent formula. By way of contradiction, assume that φ is unsatisfiable. We reduce the formula φ to an equivalent formula in the language

of real closed fields. Let $u_1, \ldots, u_{n_\mathsf{h}}$, $v_1, \ldots, v_{n_\mathsf{o}}$, $x_1, \ldots, x_{n_\mathsf{h}}$, $y_1, \ldots, y_{n_\mathsf{o}}$, and $z^1_1, \ldots, z^1_{n_\mathsf{h}}, \ldots, z^{n_\mathsf{o}}_1, \ldots, z^{n_\mathsf{o}}_{n_\mathsf{h}}$ be new variables, where, intuitively,

- u_i gets value 1 if hypothesis h_i holds, 0 otherwise;
- v_i gets value 1 if observation ob_i holds, 0 otherwise;
- x_i represents $\mathrm{Pr}^0(h_i)$;
- y_i represents $\mathrm{Pr}(h_i)$;
- $z_{i,j}$ represents $\mathrm{w}(ob_i, h_j)$.

Let **v** represent that list of new variables. Consider the following formulas. Let φ_h be the formula saying that exactly one hypothesis holds:

$$(u_1 = 0 \vee u_1 = 1) \wedge \cdots \wedge (u_{n_\mathsf{h}} = 0 \vee u_{n_\mathsf{h}} = 1) \wedge u_1 + \cdots + u_{n_\mathsf{h}} = 1.$$

Similarly, let φ_o be the formula saying that exactly one observation holds:

$$(v_1 = 0 \vee v_1 = 1) \wedge \cdots \wedge (v_{n_\mathsf{o}} = 0 \vee v_{n_\mathsf{h}} = 1) \wedge v_1 + \cdots + v_{n_\mathsf{h}} = 1.$$

Let φ_{pr} be the formula that expresses that Pr^0 is a probability measure:

$$\varphi_{pr} = x_1 \geq 0 \wedge \cdots \wedge x_{n_\mathsf{h}} \geq 0 \wedge x_1 + \cdots + x_{n_\mathsf{h}} = 1.$$

Similarly, let φ_{po} be the formula that expresses that Pr is a probability measure:

$$\varphi_{po} = y_1 \geq 0 \wedge \cdots \wedge y_{n_\mathsf{h}} \geq 0 \wedge y_1 + \cdots + y_{n_\mathsf{h}} = 1.$$

Finally, we need formulas saying that w is a weight of evidence function. The formula $\varphi_{w,p}$ simply says that w satisfies WF1, that is, it acts as a probability measure for a fixed observation:

$$\begin{aligned} &z_{1,1} \geq 0 \wedge \cdots \wedge z_{1,n_\mathsf{h}} \geq 0 \wedge z_{n_\mathsf{o},1} \geq 0 \wedge \cdots \wedge z_{n_\mathsf{o},n_\mathsf{h}} \geq 0 \wedge \\ &z_{1,1} + \cdots + z_{1,n_\mathsf{h}} = 1 \wedge \cdots \wedge z_{n_\mathsf{o},1} + \cdots + z_{n_\mathsf{o},n_\mathsf{h}} = 1. \end{aligned}$$

The formula $\varphi_{w,f}$ says that w satisfies WF2:

$$\begin{aligned} \exists w_1, \ldots, w_{n_\mathsf{o}} (&w_1 > 0 \wedge \cdots \wedge w_{n_\mathsf{o}} > 0 \wedge z_{1,1} w_1 + \cdots + z_{n_\mathsf{o},1} w_{n_\mathsf{o}} = 1 \wedge \\ &\cdots \wedge z_{1,n_\mathsf{h}} w_1 + \cdots + z_{n_\mathsf{o},n_\mathsf{h}} w_{n_\mathsf{o}} = 1) \end{aligned}$$

where $w_1, \ldots, w_{n_\mathsf{o}}$ are new variables.

Finally, the formula $\varphi_{w,up}$ captures the fact that weights of evidence can be viewed as updating a prior probability into a posterior probability, via Dempster's Rule of Combination:

$$\begin{aligned} &(v_1 = 1 \Rightarrow (x_1 z_{1,1} = y_1 x_1 z_{1,1} + \cdots + y_1 x_{n_\mathsf{h}} z_{1,n_\mathsf{h}} \wedge \\ &\qquad \cdots \wedge x_{n_\mathsf{h}} z_{1,n_\mathsf{h}} = y_{n_\mathsf{h}} x_1 z_{1,1} + \cdots + y_{n_\mathsf{h}} x_{n_\mathsf{h}} z_{1,n_\mathsf{h}})) \wedge \\ &\cdots \wedge \end{aligned}$$

$$(v_{n_\mathsf{o}} = 1 \Rightarrow (x_1 z_{n_\mathsf{o},1} = y_1 x_1 z_{n_\mathsf{o},1} + \cdots + y_1 x_{n_\mathsf{h}} z_{n_\mathsf{o},n_\mathsf{h}} \wedge \\ \cdots \wedge x_{n_\mathsf{h}} z_{n_\mathsf{o},n_\mathsf{h}} = y_{n_\mathsf{h}} x_1 z_{n_\mathsf{o},1} + \ldots y_{n_\mathsf{h}} x_{n_\mathsf{h}} z_{n_\mathsf{o},nh})).$$

Let $\hat{\varphi}$ be the formula in the language of real closed fields obtained from φ by replacing each occurrence of the primitive proposition h_i by $u_i = 1$, each occurrence of ob_i by $v_i = 1$, each occurrence of $\Pr^0(\rho)$ by $\sum_{h_i \in [\![\rho]\!]} x_i$, each occurrence of $\Pr(\rho)$ by $\sum_{h_i \in [\![\rho]\!]} y_i$, each occurrence of $\mathrm{w}(ob_i, \rho)$ by $\sum_{h_j \in [\![\rho]\!]} z_{i,j}$, and each occurrence of an integer coefficient k by $1 + \cdots + 1$ (k times). Finally, let φ' be the formula $\exists \mathbf{v}(\varphi_h \wedge \varphi_o \wedge \varphi_{pr} \wedge \varphi_{po} \wedge \varphi_{w,p} \wedge \varphi_{w,f} \wedge \varphi_{w,up} \wedge \hat{\varphi})$.

It is easy to see that if φ is unsatisfiable over evidential structures, then φ' is false when interpreted over the real numbers. Therefore, $\neg\varphi'$ must be a formula valid in real closed fields, and hence an instance of RCF. Thus, $\neg\varphi'$ is provable. It is straightforward to show, using Lemma B.11, that $\neg\varphi$ itself is provable, contradicting the fact that φ is consistent. Thus, φ must be satisfiable, establishing completeness. □

Lemma B.12. *Let φ be a formula of $\mathcal{L}^{fo\text{-}ev}(\Phi_\mathsf{h}, \Phi_\mathsf{o})$ that is satisfied in some evidential structure $M = (S \times \mathcal{P}, \mathcal{E})$. Then φ is satisfied in a structure $M = ((\Phi_\mathsf{h} \times \Phi_\mathsf{o}) \times \{\mu\}, \mathcal{E})$, where $\mu \in \mathcal{P}$.*

Proof. Let φ be a formula of $\mathcal{L}^{fo\text{-}ev}(\Phi_\mathsf{h}, \Phi_\mathsf{o})$ that is satisfied in some evidential structure $M = (S \times \mathcal{P}, \mathcal{E})$. Since φ is satisfied in M, there exists a world $w = (h, ob, \mu)$ and a valuation v (in case φ is not closed) such that $(M, w, v) \models \varphi$. Consider the model $M' = ((\Phi_\mathsf{h} \times \Phi_\mathsf{o}) \times \{\mu\}, \mathcal{E})$. Clearly, w is a world of M'. A straightforward induction on the structure of φ shows that $(M', w, v) \models \varphi$. □

As we saw at the beginning of Section 5.4, $\mathcal{L}^{fo\text{-}ev}$ is not monotone with respect to validity: axiom H1 depends on the set of hypotheses and observations, and will in general no longer be valid if the set is changed. The same is true for O1, E5, and E6. We do, however, have a form of monotonicity with respect to satisfiability, as the following lemma shows.

Lemma B.13. *Given Φ_h and Φ_o, let φ be a formula of $\mathcal{L}^{fo\text{-}ev}(\Phi_\mathsf{h}, \Phi_\mathsf{o})$, and let $\mathcal{H} \subseteq \Phi_\mathsf{h}$ and $\mathcal{O} \subseteq \Phi_\mathsf{o}$ be the hypotheses and observations that occur in φ. If φ is satisfiable in an evidential structure over Φ_h and Φ_o, then φ is satisfiable in an evidential structure over Φ'_h and Φ'_o, where $|\Phi'_\mathsf{h}| = |\mathcal{H}| + 1$ and $|\Phi'_\mathsf{o}| = |\mathcal{O}| + 1$.*

Proof. We do this in two steps, to clarify the presentation. First, we show that we can add a single hypothesis and observation to Φ_h and Φ_o and preserve satisfiability of φ. This means that the second step below can assume that $\Phi_\mathsf{h} \neq \mathcal{H}$ and $\Phi_\mathsf{o} \neq \mathcal{O}$. Assume that φ is satisfied in an evidential structure over Φ_h and Φ_o. By

Lemma B.12, φ is satisfied in an evidential structure $M = ((\Phi_{\mathsf{h}} \times \Phi_{\mathsf{o}}) \times \{\mu\}, \mathcal{E})$, that is, there exists h, ob such that $(M, (h, ob, \mu), v) \models \varphi$. Let $\Phi'_{\mathsf{h}} = \Phi_{\mathsf{h}} \cup \{h^*\}$, where h^* is a new hypothesis not in Φ_{h}, and let $\Phi'_{\mathsf{o}} = \Phi_{\mathsf{o}} \cup \{ob^*\}$, where ob^* is a new observation not in Φ_{o}. Define the evidential structure $M = ((\Phi'_{\mathsf{h}} \times \Phi'_{\mathsf{o}}) \times \{\mu'\}, cE')$ over Φ'_{h} and Φ'_{o}. Define the probability measure μ' by taking

$$\mu'(h) = \begin{cases} \mu(h) & \text{if } h \in \Phi_{\mathsf{h}} \\ 0 & \text{if } h = h^*. \end{cases}$$

Similarly, define the evidence space $\mathcal{E}' = (\Phi'_{\mathsf{h}}, \Phi'_{\mathsf{o}}, \{\mu'_h \mid h \in \Phi'_{\mathsf{h}}\})$ derived from $\mathcal{E} = (\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}}, \{\mu_h \mid h \in \Phi_{\mathsf{h}}\})$ by taking:

$$\mu'_h(ob) = \begin{cases} \mu_h(ob) & \text{if } h \in \Phi_{\mathsf{h}} \text{ and } ob \in \Phi_{\mathsf{o}} \\ 0 & \text{if } h \in \Phi_{\mathsf{h}} \text{ and } ob = ob^* \\ 0 & \text{if } h = h^* \text{ and } ob \in \Phi_{\mathsf{o}} \\ 1 & \text{if } h = h^* \text{ and } ob \in ob^*. \end{cases}$$

Thus, μ'_h extends the existing μ_h by assigning a probability of 0 to the new observation ob^*; in contrast, the new probability μ'_{h^*} assigns probability 1 to the new observation ob^*. We can check that $(M', (h, ob, \mu'), v) \models \varphi$.

The second step is to "collapse" all the hypotheses and observations that do not appear in φ into one of the hypotheses that do not appear in $\mathcal{H}$ and $\mathcal{O}$, which by the previous step are guaranteed to exist. By the previous step, we can assume that $\Phi_{\mathsf{h}} \neq \mathcal{H}$ and $\Phi_{\mathsf{o}} \neq \mathcal{O}$. Assume φ is satisfiable in an evidential structure over Φ_{h} and Φ_{o}. Thus, by Lemma B.12, there exist a structure $M = ((\Phi_{\mathsf{h}} \times \Phi_{\mathsf{o}}) \times \{\mu\}, \mathcal{E})$ and h, ob such that $(M, (h, ob, \mu), v) \models \varphi$. Pick an hypothesis and an observation from Φ_{h} and Φ_{o} as follows, depending on the world (h, ob, μ) where φ is satisfied. Let $h^\dagger$ be h if $h \notin \mathcal{H}$, otherwise, let $h^\dagger$ be an arbitrary element of $\Phi_{\mathsf{h}} - \mathcal{H}$; let $\Phi'_{\mathsf{h}} = \mathcal{H} \cup \{h^\dagger\}$. Similarly, let $ob^\dagger$ be ob if $ob \notin \mathcal{O}$, otherwise, let $ob^\dagger$ be an arbitrary element of $\Phi_{\mathsf{o}} - \mathcal{O}$; let $\Phi'_{\mathsf{o}} = \mathcal{O} \cup \{ob^\dagger\}$. Let $M' = ((\Phi'_{\mathsf{h}} \times \Phi'_{\mathsf{o}}) \times \{\mu'\}, \mathcal{E}')$ be an evidential structure over Φ'_{h} and Φ'_{o} obtained from M as follows. Define the probability measure μ' by taking

$$\mu'(h) = \begin{cases} \mu(h) & \text{if } h \in \mathcal{H} \\ \sum_{h' \in \Phi_{\mathsf{h}} - \mathcal{H}} \mu(h') & \text{if } h = h^\dagger. \end{cases}$$

Define $\mathcal{E}' = (\Phi'_{\mathsf{h}}, \Phi'_{\mathsf{o}}, \{\mu'_h \mid h \in \Phi'_{\mathsf{h}}\})$ from $\mathcal{E} = (\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}}, \{\mu_h \mid h \in \Phi_{\mathsf{h}}\})$ by

taking

$$\mu'_h(ob) = \begin{cases} \mu_h(ob) & \text{if } h \in \mathcal{H} \text{ and } ob \in \mathcal{O} \\ \sum_{ob' \in \Phi_{\mathsf{o}} - \mathcal{O}} \mu_h(ob') & \text{if } h \in \mathcal{H} \text{ and } ob = ob^\dagger \\ \sum_{h' \in \Phi_{\mathsf{h}} - \mathcal{H}} \mu_{h'}(ob) & \text{if } h = h^\dagger \text{ and } ob \in \mathcal{O} \\ \sum_{h' \in \Phi_{\mathsf{h}} - \mathcal{H}} \sum_{ob' \in \Phi_{\mathsf{o}} - \mathcal{O}} \mu_{h'}(ob') & \text{if } h = h^\dagger \text{ and } ob = ob^\dagger. \end{cases}$$

We can check by induction that $(M', (h, ob, \mu'), v) \models \varphi$, where h, ob are taken from the satisfiaction of φ in M. □

Theorem 5.6. *There is a procedure that runs in space exponential in $|\varphi| \cdot \|\varphi\|$ for deciding, given Φ_{h} and Φ_{o}, whether a formula φ of $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is satisfiable in an evidential structure.*

Proof. Let φ be a formula of $\mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. By Lemmas B.12 and B.13, φ is satisfiable if we can construct a probability measure μ on $\Phi'_{\mathsf{h}} = \mathcal{H} \cup \{h^*\}$ (where $\mathcal{H}$ is the set of hypotheses appearing in φ, and $h^* \notin \mathcal{H}$) and probability measures $\mu_{h_1}, \ldots, \mu_{h_m}$ on $\Phi'_{\mathsf{o}} = \mathcal{O} \cup \{ob^*\}$ (where $\mathcal{O}$ is the set of observations appearing in φ and $ob^* \notin \mathcal{O}$) such that $\mathcal{E} = (\Phi'_{\mathsf{h}}, \Phi'_{\mathsf{o}}, \{\mu_h \mid h \in \Phi'_{\mathsf{h}}\})$, $M = ((\Phi'_{\mathsf{h}} \times \Phi'_{\mathsf{o}}) \times \{\mu\}, \mathcal{E})$, and $(M, w, v) \models \varphi$ for some state w of M.

The aim now is to derive a formula φ' in the language of real closed fields that asserts the existence of these probability measures. More precisely, we can adapt the construction of the formula φ' from φ in the proof of Theorem 5.5. The one change we need to make is ensure that φ' is polynomial in the size of φ, which the construction in the proof of Theorem 5.5 does not guarantee. The culprit is the fact that we encode integer constants k as $1 + \cdots + 1$. It is straightforward to modify the construction so that we use a more efficient representation of integer constants, namely, a binary representation. For example, we can write 42 as $2(1 + 2^2(1 + 2^2))$, which can be expressed in the language of real closed fields as $(1 + 1)(1 + (1 + 1)(1 + 1)(1 + (1 + 1)(1 + 1)))$. We can check that if k is a coefficient of length k (when written in binary), it can be written as a term of length $O(k)$ in the language of real closed fields. Thus, we modify the construction of φ' in the proof of Theorem 5.5 so that integer constants k are represented using the above binary encoding. It is easy to see that $|\varphi'|$ is polynomial in $|\varphi| \cdot \|\varphi\|$ (since $|\Phi'_{\mathsf{h}}|$ and $|\Phi'_{\mathsf{o}}|$ are both polynomial in $|\varphi|$). We can now use the exponential space algorithm of [Ben-Or, Kozen, and Reif 1986] on φ': if φ' is satisfiable, then we can construct the required probability measures, and φ is satisfiable; otherwise, no such probability measures exist, and φ is unsatisfiable. □

Theorem 5.7. *There is a procedure that runs in space exponential in $|\varphi| \cdot \|\varphi\|$ for deciding whether there exist sets of primitive propositions Φ_{h} and Φ_{o} such that $\varphi \in \mathcal{L}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ and φ is satisfiable in an evidential structure.*

Proof. Let $h_1, \ldots, h_m$ be the hypotheses appearing in φ, and let $ob_1, \ldots, ob_n$ be the observations appearing in φ. Define the sets $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_m, h^*\}$ and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_n, ob^*\}$, where h^* and ob^* are an hypothesis and observation not appearing in φ. Clearly, $|\Phi_{\mathsf{h}}|$ and $|\Phi_{\mathsf{o}}|$ are polynomial in $|\varphi|$. By Lemma B.13, if φ is satisfiable in an evidential structure, it is satisfiable in an evidential structure over Φ_{h} and Φ_{o}. By Theorem 5.6, we have an exponential space algorithm to determine if φ is satisfied in an evidential structure over Φ_{h} and Φ_{o}. □

Theorem 5.8. *The problem of deciding, given Φ_{h} and Φ_{o}, whether a formula φ of $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ is satisfiable in an evidential structure is NP-complete.*

Proof. To establish the lower bound, observe that we can reduce propositional satisfiability to satisfiability in $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. More precisely, let f be a propositional formula, where $p_1, \ldots, p_n$ are the primitive propositions appearing in f. Let $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_n, h^*\}$ be a set of hypotheses, where hypothesis h_i corresponds to the primitive proposition p_i, and h^* is another (distinct) hypothesis; let Φ_{o} be an arbitrary set of observations. Consider the formula $\hat{f}$ obtained by replacing every occurrence of p_i in f by $\Pr^0(h_i) > 0$. It is straightforward to verify that f is satisfiable if and only if $\hat{f}$ is satisfiable in $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. (We need the extra primitive proposition h^* to take care of the case f is satisfiable in a a model where each of $p_1, \ldots, p_n$ is false. In that case, $\Pr^0(h_1) = \cdots \Pr^0(h_n) = 0$, but we can take $\Pr^0(h^*) = 1$.) This establishes the lower bound,

The upper bound is straightforward. By Lemma B.12, if φ is satisfiable, it is satisfiable in a structure with a single probability measure μ. The number of states in that structure is $|\Phi_{\mathsf{h}}| \cdot |\Phi_{\mathsf{o}}|$. We can therefore guess such a structure in time polynomial in $|\Phi_{\mathsf{h}}| + |\Phi_{\mathsf{o}}|$. We can verify that this structure satisfies φ in time polynomial in $|\varphi| + |\Phi_{\mathsf{h}}| + |\Phi_{\mathsf{h}}|$. □

Theorem 5.9. *The problem of deciding, for a formula φ, whether there exists sets of primitive propositions Φ_{h} and Φ_{o} such that $\varphi \in \mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ and φ is satisfiable in an evidential structure is NP-complete.*

Proof. For the lower bound, we reduce from the decision problem of $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ over fixed Φ_{h} and Φ_{o}. Let $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_m\}$ and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_n\}$, and let φ be a formula in $\mathcal{L}^{ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$. We can check that φ is satisfiable in evidential structure over Φ_{h} and Φ_{o} if and only if $\varphi \wedge (h_1 \vee \cdots \vee h_m) \wedge (ob_1 \vee \cdots \vee h_n)$ is sat-

isfiable in an evidential structure over arbitrary Φ'_{h} and Φ'_{o}. Thus, by Theorem 5.8, we get our lower bound.

For the upper bound, by Lemmas B.12 and B.13, if φ is satisfiable, it is satisfiable in a structure with a single probability measure μ, where the states are taken from $\Phi_{\mathsf{h}} \times \Phi_{\mathsf{o}}$, $\Phi_{\mathsf{h}} = \mathcal{H} \cup \{h^*\}$, $\mathcal{H}$ consists of the hypotheses appearing in φ, $\Phi_{\mathsf{o}} = \mathcal{O} \cup \{ob^*\}$, $\mathcal{O}$ consists of the observations appearing in φ, and h^* and ob^* are new hypotheses and observations. Thus, $|\Phi_{\mathsf{h}}| \leq |\varphi| + 1$, and $|\Phi_{\mathsf{o}}| \leq |\varphi| + 1$. The number of states in that structure is $|\Phi_{\mathsf{h}}| \cdot |\Phi_{\mathsf{o}}| \leq (|\varphi| + 1)^2$. We can therefore guess such a structure in time polynomial in $|\varphi|$. We can verify that this structure satisfies φ in time polynomial in $|\varphi|$, establishing that the problem is in NP. □

Theorem 5.11. $\mathrm{AX}_{dyn}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ *is a sound and complete axiomatization for* $\mathcal{L}_{dyn}^{fo\text{-}ev}(\Phi_{\mathsf{h}}, \Phi_{\mathsf{o}})$ *with respect to evidential systems.*

Proof. It is easy to see that each axiom is valid in evidential systems. To prove completeness, we follow the same procedure as in the proof of Theorem 5.5, showing that if φ is consistent, then it is satisfiable, that is, there exists an evidential system I and valuation v such that $(I, r, m, \mu, v) \models \varphi$ for some point (r, m, μ) of I.

As in the body of the paper, let $\Phi_{\mathsf{h}} = \{h_1, \ldots, h_{n_{\mathsf{h}}}\}$ and $\Phi_{\mathsf{o}} = \{ob_1, \ldots, ob_{n_{\mathsf{o}}}\}$. Let φ be a consistent formula. The first step of the process is to reduce the formula φ to a canonical form with respect to the $\bigcirc$ operator. Intuitively, we push down every occurrence of a $\bigcirc$ to the polynomial inequality formulas present in the formula. It is easy to see that axioms T1–T6 can be used to establish that φ is provably equivalent to a formula φ' where every occurrence of $\bigcirc$ is in the form of subformulas $\bigcirc^n(ob)$ and $\bigcirc^n(p \geq c)$, where p is a polynomial term that contains at least one occurrence of the $\Pr$ operator. (As usual, we use the notation $\bigcirc^n\varphi$ for $\bigcirc \ldots \bigcirc\varphi$, the n-fold application of $\bigcirc$ to φ.) We write $\bigcirc^0\varphi$ for φ. Let N be the maximum coefficient of $\bigcirc$ in φ', when iterated application of $\bigcirc$ are written using the n-fold notation.

By way of contradiction, assume that φ' (and hence φ) is unsatisfiable. As in the proof of Theorem 5.5, we reduce the formula φ' to an equivalent formula in the language of real closed fields. Let $u_1, \ldots, u_{n_{\mathsf{h}}}$, $v_1^0, \ldots, v_{n_{\mathsf{o}}}^0, \ldots, v_1^N, \ldots, v_{n_{\mathsf{o}}}^N$, $y_1^0, \ldots, y_{n_{\mathsf{o}}}^0, \ldots, y_1^N, \ldots, y_{n_{\mathsf{o}}}^N$, and $z_{\langle i_1,\ldots,i_k\rangle,1}, \ldots, z_{\langle i_1,\ldots,i_k\rangle,n_{\mathsf{h}}}$ (for every sequence $\langle i_1, \ldots, i_k\rangle$) be new variables, where, intuitively,

- u_i gets value 1 if hypothesis h_i holds, 0 otherwise;
- v_i^n gets value 1 if observation ob_i holds at time n, 0 otherwise;
- y_i^n represents $\Pr(h_i)$ at time n;
- $z_{\langle i_1,\ldots,i_k\rangle,j}$ represents $\mathrm{w}(\langle ob_{i_1}, \ldots, ob_{i_k}\rangle, h_j)$.

The main difference with the construction in the proof of Theorem 5.5 is that we have variables v_i^n representing the observations at every time step n, rather than variables representing observations at the only time step, variables y_i^n representing each hypothesis probability at every time step, rather than variables representing prior and posterior probabilities, and variables $z_{\langle i_1,\ldots,i_k\rangle,j}$ representing the weight of evidence of sequences of observations, rather than variables representing the weight of evidence of single observations. Let $\mathbf{v}$ represent that list of new variables. We consider the same formulas as in the proof of Theorem 5.5, modified to account for the new variables, and the fact that we are reasoning over multiple time steps. More specifically, the formula φ_h is unchanged. Instead of φ_o, we consider formulas $\varphi_o^1, \ldots, \varphi_o^N$ saying that exactly one observation holds at each time time step, where φ_o^n is given by:

$$(v_1^n = 0 \vee v_1^n = 1) \wedge \cdots \wedge (v_{n_\mathsf{o}}^n = 0 \vee v_{n_\mathsf{h}}^n = 1) \wedge v_1^n + \cdots + v_{n_\mathsf{h}}^n = 1.$$

Let $\varphi_o' = \varphi_o^1 \wedge \cdots \wedge \varphi_o^N$.

Similarly, instead of φ_{pr} and φ_{po}, we consider formulas $\varphi_p^1, \ldots, \varphi_p^N$ expressing that $\Pr$ is a probability measure at each time step, where φ_p^n is given by:

$$y_1^n \geq 0 \wedge \cdots \wedge y_{n_\mathsf{h}}^n \geq 0 \wedge y_1^n + \cdots + y_{n_\mathsf{h}}^n = 1.$$

Let $\varphi_p = \varphi_p^1 \wedge \cdots \wedge \varphi_p^N$.

Similarly, we consider $\varphi_{w,p}$ and $\varphi_{w,f}$, except where we replace variables $z_{i,j}$ by $z_{\langle i\rangle,j}$, to reflect the fact that we now consider sequences of observations. The formula $\varphi_{w,up}$, capturing the update of a prior probability into a posterior probability, is replaced by the formulas $\varphi_{w,up}^1, \ldots, \varphi_{w,up}^N$ representing the update of the probability at each time step, where $\varphi_{w,up}^n$ is given by the obvious generalization of $\varphi_{w,up}$:

$$\begin{aligned}
&(v_1^n = 1 \Rightarrow (y_1^{n-1} z_{1,1} = y_1^n y_1^{n-1} z_{1,1} + \cdots + y_1^n y_{n_\mathsf{h}}^{n-1} z_{1,n_\mathsf{h}} \wedge \\
&\qquad\qquad \cdots \wedge y_{n_\mathsf{h}}^{n-1} z_{1,n_\mathsf{h}} = y_{n_\mathsf{h}}^n y_1^{n-1} z_{1,1} + \cdots + y_{n_\mathsf{h}}^n y_{n_\mathsf{h}}^{n-1} z_{1,n_\mathsf{h}}))\wedge \\
&\cdots \wedge \\
&(v_{n_\mathsf{o}}^n = 1 \Rightarrow (y_1^{n-1} z_{n_\mathsf{o},1} = y_1^n y_1^{n-1} z_{n_\mathsf{o},1} + \cdots + y_1^n y_{n_\mathsf{h}}^{n-1} z_{n_\mathsf{o},n_\mathsf{h}} \wedge \\
&\qquad\qquad \cdots \wedge y_{n_\mathsf{h}}^{n-1} z_{n_\mathsf{o},n_\mathsf{h}} = y_{n_\mathsf{h}}^n y_1^{n-1} z_{n_\mathsf{o},1} + \ldots y_{n_\mathsf{h}}^n y_{n_\mathsf{h}}^{n-1} z_{n_\mathsf{o},n_\mathsf{h}})).
\end{aligned}$$

Let $\varphi_{w,up}' = \varphi_{w,up}^1 \wedge \cdots \wedge \varphi_{w,up}^N$.

Finally, we need a new formula $\varphi_{w,c}$ capturing the relationship between the weight of evidence of a sequence of observations, and the weight of evidence of the individual observations, to capture axiom E8:

$$\bigwedge_{\substack{1\leq k\leq N\\ 1\leq i_1,\ldots,i_k\leq n_\mathsf{o}}} z_{\langle i_1\rangle,h_1} \cdots z_{\langle i_k\rangle,h_1} = z_{\langle i_1,\ldots,i_k\rangle,h_1} z_{\langle i_1\rangle,h_1} \cdots z_{\langle i_k\rangle,h_1} \\ + \cdots + z_{\langle i_1,\ldots,i_k\rangle,h_1} z_{\langle i_1\rangle,h_{n_\mathsf{h}}} \cdots z_{\langle i_k\rangle,h_{n_\mathsf{h}}} \wedge$$

$$\cdots \wedge \bigwedge_{\substack{1\le k\le N\\ 1\le i_1,\ldots,i_k\le n_{\mathsf{o}}}} z_{\langle i_1\rangle,h_{n_{\mathsf{h}}}} \cdots z_{\langle i_k\rangle,h_{n_{\mathsf{h}}}} = z_{\langle i_1,\ldots,i_k\rangle,h_{n_{\mathsf{h}}}} z_{\langle i_1\rangle,h_1} \cdots z_{\langle i_k\rangle,h_1} + \cdots + z_{\langle i_1,\ldots,i_k\rangle,h_{n_{\mathsf{h}}}} z_{\langle i_1\rangle,h_{n_{\mathsf{h}}}} \cdots z_{\langle i_k\rangle,h_{n_{\mathsf{h}}}}.$$

Let $\hat{\varphi}$ be the formula in the language of real closed fields obtained from φ by replacing each occurrence of the primitive proposition h_i by $u_i = 1$, each occurrence of $\bigcirc^n ob_i$ by $v_i^n = 1$, and within each polynomial inequality formula $\bigcirc^n(p \geq c)$, replacing each occurrence of $\Pr(\rho)$ by $\sum_{h_i \in [\![\rho]\!]} y_i^n$, each occurrence of $\mathrm{w}(\langle ob_{i_1}, \ldots, ob_{i_k}\rangle, \rho)$ by $\sum_{h_j \in [\![\rho]\!]} z_{\langle i_1,\ldots,i_k\rangle,j}$, and each occurrence of an integer coefficient k by $1 + \cdots + 1$ (k times). Finally, let φ' be the formula $\exists \mathbf{v}(\varphi_h \wedge \varphi'_o \wedge \varphi_p \wedge \varphi_{w,p} \wedge \varphi_{w,f} \wedge \varphi'_{w,up} \wedge \varphi_{w,c} \wedge \hat{\varphi})$.

It is easy to see that if φ is unsatisfiable over evidential systems, then φ' is false about the real numbers. Therefore, $\neg\varphi'$ must be a formula valid in real closed fields, and hence an instance of RCF. Thus, $\neg\varphi'$ is provable. It is straightforward to show, via the obvious variant of Lemma B.11 (which establishes, for instance, that $\mathrm{w}(\underline{ob}, \rho) = \sum_{h \in [\![\rho]\!]} \mathrm{w}(\underline{ob}, h)$ is provable for all $\underline{ob}$) that $\neg\varphi$ itself is provable, contradicting the fact that φ is consistent. Thus, φ must be satisfiable, establishing completeness. □

B.5 Proofs for Chapter 7

Theorem 7.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system.*

Proof. Let V_a consist of all the histories $r_a(t)$ for $r \in \mathcal{R}(\Sigma, \mathcal{A}, A)$. Let $\mathcal{R}'$ be the strand system generated by the sequence $\langle V_a \mid a \in \mathcal{A}\rangle$. To show that $\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system, it clearly suffices to show that $\mathcal{R}(\Sigma, \mathcal{A}, A) = \mathcal{R}'$. It is easy to check from the construction that every run in $\mathcal{R}(\Sigma, \mathcal{A}, A)$ satisfies MP1–3, and thus is in $\mathcal{R}'$. This shows that $\mathcal{R}(\Sigma, \mathcal{A}, A) \subseteq \mathcal{R}'$.

To show that $\mathcal{R}' \subseteq \mathcal{R}(\Sigma, \mathcal{A}, A)$, let r be a run in $\mathcal{R}'$. We know that r satisfies MP1–3, and that $r_a(t) \in V_a$ for all $t \geq 0$. We need to construct a chain C such that $r_a(t) = hist_a^t(C)$ for all $a \in \mathcal{A}$. Unfortunately, we cannot simply construct the chain inductively, bundle by bundle. While this would work if different strands were associated with different agents, in general, making the correct choice of strands at each step (correct in the sense that the construction will not get stuck at a later point) turns out to require arbitrary lookahead into the run. Roughly speaking, this is because it is not clear which combination of strands for agent a to choose to make up a's local state in a particular bundle.

Instead, we proceed as follows. Intuitively, we want to determine for each agent which strand prefix to extend at every step of the chain. Once we have found for

each agent an appropriate way of extending strand prefixes at every step, it is not hard to construct the bundles in the chain.

We start with some definitions. Given a node $\langle s, k\rangle$ in Σ, let $\mathrm{tr}(s, k)$ be the prefix of $\mathrm{tr}(s)$ of length k. Given a bundle B and an agent a, let

$$\mathrm{Tr}_a(B) = \{\{\mathrm{tr}(s, k) \mid \langle s, k\rangle \in \mathcal{N}_B,\ \langle s, k+1\rangle \notin \mathcal{N}_B,\ k \geq 1,\ A(s) = a\}\},$$

where we use the $\{\{\}\}$ notation to denote multisets. Thus, $\mathrm{Tr}_a(B)$ is the multiset consisting of all the maximal prefixes of strands associated with a having at least one node in B. Note that $\mathrm{Tr}_a(B)$ is a multiset, not a set. It is quite possible that there are distinct nodes $\langle s, k\rangle$ and $\langle s', k\rangle$ in $\mathcal{N}_B$ such that $\mathrm{tr}(s, k) = \mathrm{tr}(s', k)$ and $\langle s, k+1\rangle, \langle s', k+1\rangle \notin B$. In this case, $\mathrm{tr}(s, k)$ is listed at least twice in the multiset. Given a multiset M of sequences, let $\mathcal{B}_a(M) = \{B \mid \mathrm{Tr}_a(B) = M\}$. That is, $\mathcal{B}_a(M)$ consists of all bundles where the actions performed are precisely those specified by the sequences in M.

For each agent a, we inductively construct the following tree, whose vertices are labeled by multisets of sequences. The root is labeled by the empty multiset. Suppose a vertex u at level m (that is, at distance m from the root) is labeled with the multiset M. If $r_a(m+1) = r_a(m)$, then u has a unique successor, also labeled with M. If, on the other hand, $r_a(m+1) = r_a(m) \cdot e$ for some event e, then let t be the term corresponding to e (i.e., if e is $\mathit{send}(u)$ then t is $+u$, and if e is $\mathit{recv}(u)$ then t is $-u$). For each sequence S in M, let M_S be the multiset that results from replacing S in M by $S \cdot t$. We construct a successor of M labeled M_S if $\mathcal{B}_a(M_S) \neq \varnothing$. (If $\mathcal{B}_a(M_S) \neq \varnothing$ and there are several occurrences of S in M, then we construct one successor for each occurrence.) In addition, if $\mathcal{B}_a(M \cup \{\{\langle t\rangle\}\}) \neq \varnothing$, we construct a successor of u labeled $M \cup \{\{\langle t\rangle\}\}$. Note that, for all multisets labeling a level-m vertex, the set of events specified by the sequences in M are precisely those performed in $r_a(m)$.

Our goal is to find an infinite path in this tree. That such a path exists is immediate from König's Lemma, once we show that the tree has an infinitely many vertices, each with finite outdegree.

An easy induction shows that a multiset at level m has at most m elements (counted with multiplicity). Moreover, it is immediate from the construction that the outdegree of a vertex on the tree is at most one more than the cardinality of the multiset labeling it. Thus, it follows that the outdegree of each vertex is finite.

Showing that the tree has an infinite number of vertices is also relatively straightforward. We show by induction on m that for all times t, if $r_a(t) = \mathit{hist}_a^k(C)$ and $C = B_0 \mapsto B_1 \mapsto \ldots$, then there is a vertex at level t in the tree labeled by the multiset $\mathrm{Tr}_a(B_k)$. The base case is immediate, since $\mathrm{Tr}_a(\varnothing) = \{\{\}\}$ is the label of the root of the tree. Now suppose that the result holds for t; we prove it for $t+1$. Suppose that $r_a(t+1) = \mathit{hist}_a^k(C)$. Then there must be some

$k' \leq k$ such that $r_a(t) = hist_a^{k'}(C)$. Moreover, either $hist_a^{k'}(C) = hist_a^k(C)$, in which case $r_a(t) = r_a(t+1)$, or $hist_a^k(C)$ is the result of appending one event, say e, to $hist_a^{k'}(C)$ and $r_a(t+1)$ is the result of appending e to $r_a(t)$. If $C = B_0 \mapsto B_1 \mapsto \ldots$ then, by the induction hypothesis, there is a vertex u of the tree at level t labeled by $M = \mathrm{Tr}_a(B_{k'})$. If $r_a(t) = r_a(t+1)$, then $M = \mathrm{Tr}_a(B_k)$ is also the label of a successor of u. Otherwise, if $M' = \mathrm{Tr}_a(B_k)$, it is clear that M' is the result of extending one of the strands in M by one node (corresponding to event e). Thus, M' is the label of some successor of u. This completes the inductive step. Since r is in $\mathcal{R}'$, it follows that, for all t, there exists some chain C and k such that $r_a(t) = hist_a^k(C)$. Thus, there are infinitely many vertices in the tree.

It now follows from König's Lemma that there is an infinite path in the tree. Thus, it follows that, for every agent a, there exists an infinite sequence $M_0^a, M_1^a, \ldots$ of multisets, such that $\mathcal{B}_a(M_k^a) \neq \varnothing$ for all k. We now construct a chain $C = B_0 \mapsto B_1 \mapsto \ldots$, by building the bundle B_k from the traces in $\{M_k^a \mid a \in \mathcal{A}\}$. For each a and k, there is a bundle B_k^a such that $\mathrm{Tr}_a(B_k^a) = M_k^a$. Let B_k consists of the nodes in $\cup_{a \in \mathcal{A}} B_k^a$ (so that the strands associated with a in B_k are precisely those associated with a in B_k^a), adding $\rightarrow$ edges between corresponding nodes according to MP2 in the run r. That B_k is a bundle follows from the fact that every node appearing in a multiset M_k^a corresponds to an event in $r_a(k)$, by construction. It should be clear that for all k, $B_k \mapsto B_{k+1}$, since for each agent, the traces are extended by a single node, and we can pick the bijection f to map strands from B_k to B_{k+1} so that the corresponding sequences in M_k^a and M_{k+1}^a match.

A straightforward induction argument shows that the chain $C = B_0 \mapsto B_1 \mapsto \ldots$ is such that $r_a(t) = hist_a^t(C)$ for all $t \geq 0$. □

In order to prove Theorem 7.2, we first prove two lemmas about chains.

Lemma B.14. *In a chain $C = B_0 \mapsto B_1 \mapsto B_2 \mapsto \ldots$, the height of B_n is at most $2n$.*

Proof. We show this by induction on n. Clearly, the height of B_0 is 0. Assume the result holds for the bundle B_m. Consider the bundle B_{m+1}. Since $B_m \mapsto B_{m+1}$, there is a bijection f such that $B_m \sqsubseteq_f B_{m+1}$. Consider a causal path $n_1 \rightsquigarrow n_2 \rightsquigarrow \ldots$ in B_{m+1}, where $\rightsquigarrow$ is either $\rightarrow$ or $\Rightarrow$. We claim that it contains at most two "new nodes", that is, it contains at most two nodes in B_{m+1} not of the form $\langle f(s), i\rangle$ for some node $\langle s, i\rangle$ in B_m; moreover, the "new" nodes must come at the end of the causal path. To see this, suppose that n is a new node on the path and $n \rightsquigarrow n'$ for some n' on the path. If n' is not a new node, it cannot be the case that $n \rightarrow n'$ (for otherwise, by B2, n would not be a new node), and it cannot be

the case that $n \Rightarrow n'$ (for otherwise, by B3, n would not be a new node). Thus, n' must be a new node. It follows that all the new nodes on the causal path must follow the old nodes on the path. Now suppose that there are three new nodes on the path; then it must be the case that there are three new nodes n, n', n'' such that $n \rightsquigarrow n' \rightsquigarrow n''$. It cannot be the case that $n \Rightarrow n'$, for then n and n' are both on the same strand, contradicting the assumption in the construction that at most one new event is added per agent. Similarly, it cannot be the case that $n' \Rightarrow n''$. Thus, we must have $n \rightarrow n' \rightarrow n''$. But then $\text{term}(n') = -u$ for some message u, and it cannot be the case that $n' \rightarrow n''$. Thus, it follows that the causal path has at most two new nodes. Since, by the induction hypothesis, there are at most $2m + 1$ "old" nodes on the path, the path has at most $2m + 3$ nodes and hence length at most $2m + 2$, as desired. □

Note that Lemma B.14 does not depend on the assumption that each strand is associated with a distinct agent; the following lemma does.

Lemma B.15. *If B is bundle of finite height, then there exists bundles $B_1, \ldots, B_k$ for some k such that $B_0 \mapsto B_1 \mapsto \ldots \mapsto B_k \mapsto B$.*

Proof. First note that if n is the last node on a causal path in a bundle B of maximum length, then either $\text{term}(n) = -u$ for some u or $\text{term}(n) = +u$ for some u and there is no corresponding receive node in B.

We now prove the result by induction on the height of B, that is the length of the longest causal path. Clearly, if the height of B is 0, then $B = B_0$. Otherwise, let B' be the bundle derived from B in the following way: for every strand $s \in \Sigma$, if the last term of the prefix of s in B is $-u$ for some u or if the last term is $+u$ and there is no corresponding $-u$ in B, then let B' contain the prefix of s that consists of every node in s that is in B but the last one; otherwise, let B' contain the same prefix of s as B. Clearly, $B' \mapsto B$. (Here we need the assumption that each strand is associated with a different agent to ensure that in going from B' to B, each agent performs at most one action.) Moreover, by the initial observation, B' does not include the last node of any causal path of maximum length in B. Therefore, the height of B' is one less than the height of B. Applying the induction hypothesis, we get bundles $B_0 \mapsto B_1 \mapsto \ldots \mapsto B_k \mapsto B' \mapsto B$, proving the result. □

Theorem 7.2. *Every global state of $\mathcal{R}(\Sigma, \Sigma, id)$ is message-equivalent to a bundle of Σ of finite height, and every bundle of Σ of finite height is message-equivalent to a global state of $\mathcal{R}(\Sigma, \Sigma, id)$.*

Proof. If $\langle \sigma_s \mid s \in \Sigma \rangle$ is a global state in $\mathcal{R}(\Sigma, \Sigma, id)$, then there must be some chain $C = B_0 \mapsto B_1 \mapsto \ldots$ and time t such that $r^C(t) = \langle \sigma_s \mid s \in \Sigma \rangle$. By

construction, $r_s^C(t) = hist_s^t(C)$, for each strand $s \in \Sigma$. (Recall that $\mathcal{A} = \Sigma$; we are associating each strand with a different agent.) Moreover, $hist_s^t(C)$ is just the sequence of events performed in strand s in B_t (that is, the prefix of $\mathrm{tr}(s)$ in B_t, under the standard correspondence between terms and events). Therefore, $\langle \sigma_s \mid s \in \Sigma \rangle$ is message-equivalent to B_t. Moreover, by Lemma B.14, B_t has finite height.

Conversely, given a bundle B of finite height, by Lemma B.15, there must exist t and bundles $B_0, \ldots, B_t$ such that $B_0 \mapsto \ldots \mapsto B_t \mapsto B$. Thus, $C = B_0 \mapsto \ldots \mapsto B_t \mapsto B \mapsto B \mapsto B \mapsto \ldots$ is a chain. Let r^C be the run in $\mathcal{R}(\Sigma, \Sigma, id)$ corresponding to C. By the same argument as above, $r^C(t+1)$ is message-equivalent to B. □

Theorem 7.3. *There is no agent assignment A and A-history preserving translation T from strand spaces to strand systems such that the strand system $\mathcal{R}_1$ is in the image of T.*

Proof. By way of contradiction, suppose that Σ is a strand space, A is an agent assignment, T is a translation which is A-history preserving, and $T(\Sigma) = \mathcal{R}_1$. Since T is A-history preserving, the presence of r_1 ensures that there is a bundle B_1 in Σ such that associated with agent 2 in B_1 is either a strand with prefix $\langle +u, -v \rangle$ or strands with prefix $\langle +u \rangle$ and $\langle -v \rangle$, and associated with agent 1 in B_1 there is either a strand with prefix $\langle -u, +v \rangle$ or strands with prefix $\langle -u \rangle$ and $\langle +v \rangle$. Similarly, the presence of r_2 in $\mathcal{R}_1$ guarantees that there is a bundle B_2 in Σ such that associated with agent 2 in B_2 is either a strand with prefix $\langle +x, -y \rangle$ or strands with prefix $\langle +x \rangle$ and $\langle -y \rangle$, and associated with agent 3 is either a strand with prefix $\langle -x, +y \rangle$ or strands with prefix $\langle -x \rangle$ and $\langle +y \rangle$. In all those cases, there must be a bundle containing nodes with the terms $+u$, $-u$, $+v$, $-v$, $+x$, $-x$, $+y$, and $-y$. The nodes $+u$, $-v$, $+x$, and $-y$ are all on strands associated with agent 2. Since T is A-history preserving, there must be a run in $\mathcal{R}_1$ that contains four events for agent 2. This is a contradiction. □

B.6 Proofs for Chapter 8

Theorem 8.1. *Let $\mathfrak{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ be an interpreted algorithmic knowledge security system where $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY}}$. Then $(\mathfrak{I}, r, t) \models X_i(has_i(m))$ if and only if $\{m \mid recv(m) \in r_i(t)\} \cup initkeys(r_i(t)) \vdash_{DY} m$. Moreover, if $(\mathfrak{I}, r, t) \models X_i(has_i(m))$ then $(\mathfrak{I}, r, t) \models has_i(m)$.*

Proof. Let $K = keysof(r_i(t))$. First, note that $K \cup \{m' \mid recv(m') \in r_i(t)\} \vdash_{DY} m$ if and only if $K \cup \{m'\} \vdash_{DY} m$ for some m' such that $recv(m') \in r_i(t)$. The re-

sult relies on the invariant that if $m \notin initkeys(r_i(t))$, then $submsg(m, m', K) =$ "Yes" if and only if $K \cup \{m'\} \vdash_{DY} m$. This is established by a straightforward induction on the structure of recursive calls in $submsg$. It is easy to check that if $submsg(m, m', K) =$ "Yes", then $m \sqsubseteq m'$, which immediately yields soundness of $\mathtt{A}_i^{\mathrm{DY}}$ with respect to $has_i(m)$. □

Theorem 8.2. *Let $\mathcal{I} = (\mathcal{R}, \pi, \mathtt{A}_1, \ldots, \mathtt{A}_n)$ be an interpreted algorithmic knowledge security system where $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{L}}$. Then $(\mathcal{I}, r, t) \models X_i(has_i(m))$ if and only if $\{m \mid recv(m) \in r_i(t)\} \cup initkeys(r_i(t)) \vdash_{\mathrm{L}} m$. Moreover, if $(\mathcal{I}, r, t) \models X_i(has_i(m))$ then $(\mathcal{I}, r, t) \models has_i(m)$.*

Proof. Let $K = keysof(r_i(t))$. The result follows readily from Theorem 8.1 and the invariant that if $m \notin initkeys(r_i(t))$ and $K \cup \{m' \mid recv(m') \in r_i(t)\} \nvdash_{DY} m$, then $guess(m, r_i(t)) =$ "Yes" if and only if $K \cup \{m \mid recv(m) \in r_i(t)\} \vdash_{\mathrm{L}} m$. The details of the invariant are similar to those given by Lowe [2002], the algorithm $\mathtt{A}_i^{\mathrm{L}}$ being essentially a translation of the CSP process implementing the Lowe adversary. Again, soundness with respect to $has_i(m)$ is easy to establish. □

Theorem 8.3. *Suppose that $\mathcal{J} = (\mathcal{R}, \pi, \mathtt{A}_1^d, \ldots, \mathtt{A}_n^d, \nu)$ is an interpreted probabilistic algorithmic knowledge security system with an adversary as agent i and that $\mathtt{A}_i = \mathtt{A}_i^{\mathrm{DY+rg}(n)}$. Let K be the number of distinct keys used in the messages in the adversary's local state ℓ (that is, the number of keys used in the messages that the adversary has intercepted at a point (r, t), in local state $r_i(t) = \ell$). Suppose that $K/|\mathcal{K}| < 1/2$ and that ν is the uniform distribution on sequences of coin tosses. If $(\mathcal{J}, r, t, v) \models \neg K_i X_i(has_i(m))$, then $(\mathcal{J}, r, t, v) \models \Pr(X_i(has_i(m))) < 1 - e^{-2nK/|\mathcal{K}|}$. Moreover, if $(\mathcal{J}, r, t, v) \models X_i(has_i(m))$ then $(\mathcal{J}, r, t, v) \models has_i(m)$.*

Proof. It is not hard to show that the n keys that the adversary guesses do no good at all if none of them match a key used in a message intercepted by the adversary. By assumption, K keys are used in messages intercepted by the adversary. The probability that a key chosen at random is one of these K is $K/|\mathcal{K}|$, since there are $|\mathcal{K}|$ keys altogether. Thus, the probability that a key chosen at random is not one of these K is $1 - (K/|\mathcal{K}|)$. The probability that none of the n keys chosen at random is one of these K is therefore $(1 - (K/|\mathcal{K}|))^n$. We now use some standard approximations. Note that $(1 - (K/|\mathcal{K}|))^n = e^{n \ln(1-(K/|\mathcal{K}|)}$, and

$$\begin{aligned} \ln(1 - x) &= -x - x2/2 - x3/3 - \cdots \\ &> -x - x2 - x3 - \cdots \\ &= -x/(1 - x). \end{aligned}$$

Thus, if $0 < x < 1/2$, then $\ln(1-x) > -2x$. It follows that if $K/|\mathcal{K}| < 1/2$, then $e^{n\ln(1-(K/|\mathcal{K}|)} > e^{-2nK/|\mathcal{K}|}$. Since the probability that a key chosen at random does not help to compute algorithmic knowledge is greater than $e^{-2nK/|\mathcal{K}|}$, the probability that it helps is less than $1 - e^{-2nK/|\mathcal{K}|}$.

Soundness of $\mathtt{A}_i$ with respect to $has_i(m)$ follows from Theorem 8.1 (since soundness follows for arbitrary $initkeys(\ell) \subseteq \mathcal{K}$). □

B.7 Proofs for Chapter 9

Lemma B.16. *For all probabilistic interpreted systems $\mathcal{I}$ and points (r,t) in $\mathcal{I}$, if $(\mathcal{I},r,t) \models K_i^\alpha\varphi$ and $(\mathcal{I},r,t) \models K_i^\beta\psi$ then $(\mathcal{I},r,t) \models K_i^{\alpha+\beta}(\varphi \wedge \psi)$.*

Proof. This follows from the standard properties of probability measures. For a probability measure μ, if $\mu(U) \geq 1-\alpha$ and $\mu(V) \geq 1-\beta$, then $\mu(\overline{U}) \leq \alpha$ and $\mu(\overline{V}) \leq \beta$, so that $\mu(\overline{U} \cup \overline{V}) \leq \alpha+\beta$, $\mu(U \cap V) = \mu(\overline{\overline{U} \cup \overline{V}}) \geq 1-(\alpha+\beta)$.

Now, assume $(\mathcal{I},r,t) \models K_i^\alpha\varphi$ and $(\mathcal{I},r,t) \models K_i^\beta\psi$. Therefore, for all $(r',t') \sim_i (r,t)$, $(\mathcal{I},r',t') \models \Pr_i(\varphi) \geq 1-\alpha$ and $(\mathcal{I},r',t') \models \Pr_i(\psi) \geq 1-\beta$, which means that

$$\mu_{r',t',i}(\{(r'',t'') \mid (\mathcal{I},r'',t'') \models \varphi\} \cap \mathcal{K}_i(r',t') \cap \mathcal{C}(r')) \geq 1-\alpha,$$

and

$$\mu_{r',t',i}(\{(r'',t'') \mid (\mathcal{I},r'',t'') \models \psi\} \cap \mathcal{K}_i(r',t') \cap \mathcal{C}(r')) \geq 1-\beta.$$

From the derivation above, we have that

$$\begin{aligned}\mu_{r',t',i}(\{(r'',t'') \mid (\mathcal{I},r'',t'') \models \varphi\} \cap \\ \{(r'',t'') \mid (\mathcal{I},r'',t'') \models \varphi\} \cap \mathcal{K}_i(r',t') \cap \mathcal{C}(r')) = \\ \mu_{r',t',i}(\{(r'',t'') \mid (\mathcal{I},r'',t'') \models \varphi \wedge \psi\} \cap \mathcal{K}_i(r',t') \cap \mathcal{C}(r')) \geq 1-(\alpha+\beta).\end{aligned}$$

Hence, $(\mathcal{I},r',t') \models \Pr_i(\varphi \wedge \psi) \geq 1-(\alpha+\beta)$. Since (r',t') was an arbitrary point such that $(r',t') \sim_i (r,t)$, we have $(\mathcal{I},r,t) \models K_i^{\alpha+\beta}(\varphi \wedge \psi)$. □

Lemma B.17. *Suppose $\mathcal{I}$ is an an interpreted system modeling Dolev-Yao agents with no additional prior information beyond guesses, and agents i and j are non-forging in $\mathcal{I}$. Then if $(\mathcal{I},r,t) \models X_i(has_i(\{m^l\}_k)) \wedge (j \overset{k}{\leftrightarrow} i)^T$ where $l \neq i$ and $(j \overset{k}{\leftrightarrow} i)^T$ is a possible translation of $j \overset{k}{\leftrightarrow} i$, then $(\mathcal{I},r,t) \models send_j(m)$.*

Proof. Since agents have no additional prior information beyong guesses, the message $\{m^l\}_k$ is part of no agent's initial state. Thus, by definition of $can\_compute_i^{\mathrm{NF}}$,

if $(\mathcal{I}, r, t) \models X_i(has_i(\{m^l\}_k))$ with $l \neq i$ then i must have received $\{m^l\}_k$ as a submessage of some message. Let $t' < t$ be the earliest time at which some agent h sends a message m' with $\{m^l\}_k \sqsubseteq m'$. Since all agents in the system are Dolev-Yao, we have $\{m^l\}_k \in can\_compute_h^{\mathrm{NF}}(r_h(t'))$. Since agents have no additional prior information beyond guesses, it follows that $k \in can\_compute_h^{\mathrm{NF}}(r_h(t'))$. Since $(\mathcal{I}, r, t) \models j \overset{k}{\leftrightarrow} i$ and agents have perfect recall, no agent besides i and j can extract k at time t', and it follows that $h = i$ or $h = j$. These agents are both nonforging Dolev-Yao agents, so we have $\{m^l\}_k \in can\_compute_h^{\mathrm{NF}}(r_h(t'))$. Since m' is the first time a message containing $\{m^l\}_k$ is sent, and agents have no additional prior information beyond guesses, $\{m^l\}_k$ must have been constructed at time t' using the condition for formation of messages of this form. By nonforgingness, this implies that $h = j$. □

Theorem 9.1. *Every translation r_{ij}^T of an instance r_{ij} of the BAN inference rule Rn, for $n = 1, 2$, is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses and where agents i and j are nonforging Dolev-Yao agents. Every translation r_{ij}^T of an instance r_{ij} of the BAN inference rule R3 is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses and where agents i and j are γ-honest. Finally, every translation r^T of an instance r of Rn for $n \geq 4$ is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses.*

Proof. We show that for all instances of BAN inference rules "from F_1 and F_2 infer F_3" and all interpreted systems $\mathcal{I}$ satisfying the assumptions of the theorem, we have $\mathcal{I} \models {F_1}^T \wedge {F_2}^T \Rightarrow {F_3}^T$.

Fix an interpreted system $\mathcal{I}$ satisfying the assumptions of the theorem, and a point (r, t) in $\mathcal{I}$. We proceed to show that the translation of an instance of each inference rule holds for that interpreted system at that point, thereby establishing the result. We assume that $\mathcal{I}$ models Dolev-Yao agents that have no additional prior information beyond guesses.

For instances r_{ij} of R1 and R2, we assume that $\mathcal{I}$ models Dolev-Yao agents that have no additional prior information beyond guesses, and agents i and j as nonforging Dolev-Yao agents.

Rule R1. A possible translation of an instance of R1 take the form $(K_i^\alpha(good \Rightarrow (j \overset{k}{\leftrightarrow} i)^T) \wedge X_i(has_i(\{F^T, l\}_k))) \Rightarrow K_i^\alpha(good \Rightarrow send_j(F^T) \wedge \boxdot(\neg send_j(F^T) \wedge \bigcirc send_j(F^T) \Rightarrow X_j(has_j(F^T)))))$, where F^T is a possible translation of F, and $(j \overset{k}{\leftrightarrow} i)^T$ is a possible translation of $j \overset{k}{\leftrightarrow} i$. Assume that

$$(\mathcal{I}, r, t) \models (i \textbf{ believes } j \overset{k}{\leftrightarrow} i)^T \wedge (i \textbf{ sees } \{F^l\}_k)^T,$$

with $l \neq i$. Then $(\mathcal{I}, r, t) \models X_i(has_i(\{F^l\}_k))$. Hence, using the fact that extraction depends only on an agent's local state, for all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models (i \textbf{ believes } j \overset{k}{\leftrightarrow} i)^T \wedge X_i(has_i(\{F^l\}_k))$. By Lemma B.17, we have $(\mathcal{I}, r', t') \models send_j(F)$. This shows that $(\mathcal{I}, r, t) \models K_i send_j(F)$, that is, $(\mathcal{I}, r, t) \models (i \textbf{ believes } j \textbf{ said } F)^T$.

Rule R2. A possible translation of an instance of R2 has the form $(K_i^\alpha(good \Rightarrow (j \overset{k}{\leftrightarrow} i)^T) \wedge X_i(has_i(\{F^T, l\}_{k^{-1}}))) \Rightarrow K_i^\alpha(good \Rightarrow send_j(F^T) \wedge \boxdot(\neg send_j(F^T) \wedge \bigcirc send_j(F^T) \Rightarrow X_j(has_j(F^T))))$, where F^T is a possible translation of F, and $(j \overset{k}{\leftrightarrow} i)^T$ is a possible translation of $j \overset{k}{\leftrightarrow} i$. Assume that

$$(\mathcal{I}, r, t) \models (i \textbf{ believes } j \overset{k}{\leftrightarrow} i)^T \wedge (i \textbf{ sees } \{F^l\}_{k^{-1}})^T,$$

with $l \neq i$. Since k is a symmetric key with $k^{-1} = k$, this rule is the same as R1. Hence, by that argument, $(\mathcal{I}, r, t) \models (i \textbf{ believes } j \textbf{ said } F)^T$.

For an instance r_{ij} of R3, we assume that $\mathcal{I}$ models Dolev-Yao agents that have no additional prior information beyond guesses, and agents i and j are γ-honest.

Rule R3. A possible translation of an instance of R3 has the form $(K_i^\alpha(good \Rightarrow \ominus^l \bigwedge_i(\boxdot \neg send_i(F^T))) \wedge K_i^\beta(good \Rightarrow send_j(F^T) \wedge \boxdot(\neg send_j(F^T) \wedge \bigcirc send_j(F^T) \Rightarrow X_j(has_j(F^T))))) \Rightarrow K_i^{\alpha+\beta}(good \Rightarrow K_j^\gamma(good \Rightarrow F^T))$, where F^T is a possible translation of F. Assume that

$$(\mathcal{I}, r, t) \models (i \textbf{ believes } \textbf{fresh}(F))^T \wedge (i \textbf{ believes } (j \textbf{ said } F))^T,$$

that is, we have $(\mathcal{I}, r, t) \models K_i(\wedge_i \neg \ominus^l X_i(has_i(F^T)))$, $(\mathcal{I}, r, t) \models K_i X_j(has_j(F^T))$, $(\mathcal{I}, r, t) \models K_i(\neg \ominus^l X_j(has_j(F^T)) \Rightarrow K_i F^T)$, and $(\mathcal{I}, r, t) \models K_i send_j(F^T)$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models X_j(has_j(F^T))$ and $(\mathcal{I}, r', t') \models send_j(F^T)$. By honesty of j $(\mathcal{I}, r', t') \models K_j F^T$. Hence $(\mathcal{I}, r, t) \models K_i K_j F^T$, that is, $(\mathcal{I}, r, t) \models (i \textbf{ believes } j \textbf{ believes } F)^T$.

For instances of the remaining rules, we assume that $\mathcal{I}$ models Dolev-Yao agents that have no additional prior information beyond guesses.

Rule R4. A possible translation of an instance of R4 is of the form $(K_i^\alpha(good \Rightarrow (K_j F^T \Leftrightarrow F^T)) \wedge K_i^\beta(good \Rightarrow K_j \delta(good \Rightarrow F^T))) \Rightarrow K_i^{\alpha+\beta}(good \Rightarrow F^T)$, where F^T is a possible translation of F. Assume that

$$(\mathcal{I}, r, t) \models (i \textbf{ believes } j \textbf{ controls } F)^T \wedge (i \textbf{ believes } j \textbf{ believes } F)^T,$$

that is, we have $(\mathcal{I}, r, t) \models K_i(K_j F^T \Rightarrow F^T)$, and $(\mathcal{I}, r, t) \models K_i K_j F^T$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models K_j F^T \Rightarrow F^T$ and $(\mathcal{I}, r', t') \models K_j F^T$. This immediately implies $(\mathcal{I}, r', t') \models F^T$, so that $(\mathcal{I}, r, t) \models K_i F^T$, and $(\mathcal{I}, r, t) \models (i \textbf{ believes } F)^T$.

Rule R5. A possible translation of an instance of R5 is $X_i(has_i((F^T, F'^T))) \Rightarrow$

$X_i(has_i(F^T))$, where F^T and F'^T are possible translations of F and F'. Assume $(\mathcal{I}, r, t) \models (i \textbf{ sees } (F, F'))^T$, that is, $(\mathcal{I}, r, t) \models X_i(has_i((F^T, F'^T)))$. This implies that $(F^T, F'^T) \in can\_compute_i(r_i(t))$, so $F^T \in can\_compute_i(r_i(t))$. Hence, we have $(\mathcal{I}, r, t) \models X_i(has_i(F^T))$, and $(\mathcal{I}, r, t) \models (i \textbf{ sees } F)^T$.

Rule R6. A possible translation of an instance of R6 is of the form $(K_i^\alpha(good \Rightarrow (j \overset{k}{\leftrightarrow} i)^T) \wedge X_i(has_i(\{F^T, j\}_k))) \Rightarrow X_i(has_i(F^T))$, where F^T is a possible translation of F. Assume $(\mathcal{I}, r, t) \models (i \textbf{ believes } j \overset{k}{\leftrightarrow} i)^T \wedge (i \textbf{ sees } \{F\}_k)^T$, that is, we have $(\mathcal{I}, r, t) \models K_i X_i(has_i(k))$, $(\mathcal{I}, r, t) \models K_i X_j(has_j(k))$, $(\mathcal{I}, r, t) \models K_i(\neg X_{i'}(has_{i'}(k)))$ ($i' \neq i, j$), and $(\mathcal{I}, r, t) \models X_i(has_i(\{F^T\}_k))$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models X_i(has_i(k))$ and $(\mathcal{I}, r', t') \models X_i(has_i(\{F^T\}_k))$ (since the interpretation of algorithmic knowledge depends only on the agent's local state). This implies that $k \in can\_compute_i(r_i'(t'))$ and $\{F^T\}_k \in can\_compute_i(r_i'(t'))$, and therefore $F^T \in can\_compute_i(r_i'(t'))$, and $(\mathcal{I}, r', t') \models X_i(has_i(F^T))$. Again, since the interpretation of algorithmic knowledge depends only on the local state, $(\mathcal{I}, r, t) \models X_i(has_i(F^T))$, and $(\mathcal{I}, r, t) \models (i \textbf{ sees } F)^T$.

Rule R7. A possible translation of an instance of R7 is of the form $(K_i^\alpha(good \Rightarrow (\overset{k}{\mapsto} i)^T) \wedge X_i(has_i(\{F^T, j\}_k))) \Rightarrow X_i(has_i(F^T))$, where F^T is a possible translation of F. Assume $(\mathcal{I}, r, t) \models (i \textbf{ believes } \overset{k}{\mapsto} i)^T \wedge (i \textbf{ sees } \{F\}_k)^T$, that is, $(\mathcal{I}, r, t) \models K_i X_i(has_i(k^{-1}))$, $(\mathcal{I}, r, t) \models K_i(\neg X_j(has_j(k^{-1})))$ ($j \neq i$), and thus we have $(\mathcal{I}, r, t) \models X_i(has_i(\{F^T\}_k))$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models X_i(has_i(k^{-1}))$ and $(\mathcal{I}, r', t') \models X_i(has_i(\{F^T\}_k))$ (since the interpretation of algorithmic knowledge depends only on the agent's local state). This implies that $k^{-1} \in can\_compute_i(r_i'(t'))$ and $\{F^T\}_k \in can\_compute_i(r_i'(t'))$, and therefore $F^T \in can\_compute_i(r_i'(t'))$, and $(\mathcal{I}, r', t') \models X_i(has_i(F^T))$. Again, since the interpretation of algorithmic knowledge depends only on the local state, $(\mathcal{I}, r, t) \models X_i(has_i(F^T))$, and $(\mathcal{I}, r, t) \models (i \textbf{ sees } F)^T$.

Rule R8. A possible translation of an instance of R8 is of the form $(K_i^\alpha(good \Rightarrow (\overset{k}{\mapsto} j)^T) \wedge X_i(has_i(\{F^T, j\}_{k^{-1}}))) \Rightarrow X_i(has_i(F^T))$, where F^T is a possible translation of F. Assume $(\mathcal{I}, r, t) \models (i \textbf{ believes } \overset{k}{\mapsto} j)^T \wedge (i \textbf{ sees } \{F\}_{k^{-1}})^T$, that is, $(\mathcal{I}, r, t) \models K_i X_j(has_j(k^{-1}))$, $(\mathcal{I}, r, t) \models K_i(\neg X_{i'}(has_{i'}(k^{-1})))$ ($i' \neq j$), and thus we have $(\mathcal{I}, r, t) \models X_i(has_i(\{F^T\}_{k^{-1}}))$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models X_i(has_i(\{F^T\}_{k^{-1}}))$ (since algorithmic knowledge depends only on the agent's local state). This implies that $\{F^T\}_{k^{-1}} \in can\_compute_i(r_i'(t'))$. Under the assumption of no additional prior information, the agents have in their initial state the public keys of all other agents. Hence, $k \in can\_compute_i(r_i'(t'))$, and therefore $F^T \in can\_compute_i(r_i'(t'))$, and $(\mathcal{I}, r', t') \models X_i(has_i(F^T))$. Again, since the interpretation of algorithmic knowledge depends only on the local state, $(\mathcal{I}, r, t) \models X_i(has_i(F^T))$, and $(\mathcal{I}, r, t) \models (i \textbf{ sees } F)^T$.

Rule R9. A possible translation of an instance of R9 takes the form $K_i^\alpha(good \Rightarrow \ominus^l \bigwedge_i(\boxminus \neg send_i(F^T))) \Rightarrow K_i^\alpha(good \Rightarrow \ominus^l \bigwedge_i(\boxminus \neg send_i((F^T, F'^T))))$, where F^T and F'^T are possible translations of F and F'. Assume that

$$(\mathcal{I}, r, t) \models (i \ \mathbf{believes\ fresh}(F))^T,$$

that is, $(\mathcal{I}, r, t) \models K_i(\wedge_i \neg \ominus^l X_i(has_i(F^T)))$. For all $(r', t') \sim_i (r, t)$, we have $(\mathcal{I}, r', t') \models \neg \ominus^l X_i(has_i(F^T))$, and $(\mathcal{I}, r', t') \not\models \ominus^l X_i(has_i(F^T))$. If $t' < l$, then for any φ, $(\mathcal{I}, r', t') \not\models \ominus^l \varphi$. If $t' \geq l$, then $F^T \notin can\_compute_i(r_i'(t' - l))$. This implies that $(F^T, F'^T) \notin can\_compute_i(r_i'(t' - l))$ for all F', and $(\mathcal{I}, r', t') \not\models \ominus^l X_i(has_i((F, F')^T))$. Therefore, $(\mathcal{I}, r, t) \models K_i(\wedge_i \neg \ominus^l X_i(has_i((F, F')^T)))$ and $(\mathcal{I}, r, t) \models (i \ \mathbf{believes\ fresh}(F, F'))^T$. □

Theorem 9.2. *If r is a run where A's key is k_A, B's key is k_B, A's nonce is n_A, and B's nonce is n_B, then*

$$(\mathcal{I}^{GDY}, r, 0) \models \Box(recv_B(\{n_B\}_{k_B}) \Rightarrow (F(n_A, n_B, k_A, k_B))^{T,0}).$$

Proof. Clearly, it is sufficient to show that for any good run r, for any conjunct C of $F(n_A, n_B, k_A, k_B)$, we have $(\mathcal{I}^{GDY}, r, 0) \models \Box(recv_B(\{n_B\}_{k_B}) \Rightarrow C)$.

The following statements are straightforward to prove by induction on the length of a run: (a) on every good run r, if k_i is i's public key ($i = A, B$), then for all $t \geq 0$, $k_i^{-1} \in can\_compute_i(r_i(t))$ and $k_i^{-1} \notin can\_compute_j(r_j(t))$ for $j \neq i$; (b) on every good run r, if n_i is i's nonce and k_i is i's public key ($i = A, B$), then for all $t \geq 0$, if $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, then $send(B, \{\!|n_B|\!\}_{k_B}) \in r_A(t)$, if $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r_A(t)$, then $send(A, \{\!|n_A, n_B|\!\}_{k_A}) \in r_B(t)$, if $recv(\{\!|n_A, A|\!\}_{k_B}) \in r_B(t)$, then $send(B, \{\!|n_A, A|\!\}_{k_B}) \in r_A(t)$; (c) on every good run r, if n_i is i's nonce and k_i is i's public key ($i = A, B$), then for all $t \geq 0$ and all $i = A, B$, $n_i \notin can\_compute_j(r_j(t))$, for $j \neq A, B$. (The proofs of these facts is essentially the same as those of Paulson [1998].)

Case: A **believes** $\overset{k_B}{\mapsto} B$**.** Let r be a good run where k_B is B's public key, and let $t \geq 0$ be a time where $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$. Let $(r', t') \sim_A (r, t)$ be a point on another good run r'. Because public keys are in the initial state of the agents, it must be the case that on run r', k_B is B's public key. By (a) above, k_B^{-1} is only in $can\_compute_B(r_B'(t'))$, and thus we have $(\mathcal{I}^{GDY}, r', t') \models (\overset{k_B}{\mapsto} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r, t) \models K_A^0(good \Rightarrow (\overset{k_B}{\mapsto} B)^{T,0})$, as desired.

Case: B **believes** $\overset{k_A}{\mapsto} A$**.** Let r be a good run where k_A is A's public key, and let $t \geq 0$ be a time where $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$. Let $(r', t') \sim_B (r, t)$ be a point on another good run r'. Because public keys are in the initial state of the agents, it must be the case that on run r', k_A is A's public key. By (a) above, k_A^{-1} is only

in $can\_compute_A(r'_A(t'))$, and thus we have $(\mathcal{I}^{GDY}, r', t') \models (\overset{k_A}{\mapsto} A)^{T,0}$, so that $(\mathcal{I}^{GDY}, r, t) \models K_B^0(good \Rightarrow (\overset{k_A}{\mapsto} A)^{T,0})$, as desired.

Case: A believes $A \overset{n_A}{\rightleftharpoons} B$. Let r be a good run where k_i is i's public key and n_i is i's nonce ($i = A, B$). If $t \geq 0$ is such that $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, then by (b) above, we have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r_A(t)$. Let $(r', t') \sim_A (r, t)$, with r' a good run. We must have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r'_A(t')$. Thus, r' must be a good run that uses nonce n_A. By (c) above and by the definition of $can\_compute_i$, we have $(\mathcal{I}^{GDY}, r', t') \models (A \overset{n_A}{\rightleftharpoons} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r, t) \models K_A^0(good \Rightarrow (A \overset{n_A}{\rightleftharpoons} B)^{T,0})$, as desired.

Case: B believes A believes $A \overset{n_A}{\rightleftharpoons} B$. The argument is similar to the previous case. Let r be a good run where k_i is i's public key and n_i is i's nonce ($i = A, B$). If $t \geq 0$ is such that $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, then for any good run r' with $(r', t') \sim_B (r, t)$, we must have $recv(\{\!|n_B|\!\}_{k_B}) \in r'_B(t')$. By (b) above, we have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r'_A(t')$. Let r'' be a good run with $(r'', t'') \sim_A (r', t')$; we must have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r''_A(t'')$. Thus, r'' must be a good run that uses nonce n_A. By (c) above and by the definition of $can\_compute_i$, we have $(\mathcal{I}^{GDY}, r'', t'') \models (A \overset{n_A}{\rightleftharpoons} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r', t') \models K_A^0(good \Rightarrow (A \overset{n_A}{\rightleftharpoons} B)^{T,0})$, and $(\mathcal{I}^{GDY}, r, t) \models K_B^0(good \Rightarrow K_A^0(good \Rightarrow (A \overset{n_A}{\rightleftharpoons} B)^{T,0}))$, as desired.

Case: B believes $A \overset{n_B}{\rightleftharpoons} B$. Let r be a good run where k_i is i's public key and n_i is i's nonce ($i = A, B$). If $t \geq 0$ is such that $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, then we for all good runs r' with $(r', t') \sim_B (r, t)$, we must have $recv(\{\!|n_B|\!\}_{k_B}) \in r'_B(t')$. Thus, r' must be a good run that uses nonce n_B. By (c) above and by the definition of $can\_compute_i$, we have $(\mathcal{I}^{GDY}, r', t') \models (A \overset{n_B}{\rightleftharpoons} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r, t) \models K_B^0(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0})$, as desired.

Case: A believes B believes $A \overset{n_B}{\rightleftharpoons} B$. Let r be a good run where k_i is i's public key and n_i is i's nonce ($i = A, B$). If $t \geq 0$ is such that $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, by (b) above, we have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r_A(t)$. For all good runs r' with $(r', t') \sim_A (r, t)$, we must have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r'_A(t')$. By (b) above, we have $send(B, \{\!|n_A, n_B|\!\}_{k_A}) \in r'_B(t')$. Let r'' be a good run with $(r'', t'') \sim_B (r', t')$; we must have $send(B, \{\!|n_A, n_B|\!\}_{k_A}) \in r''_B(t'')$. Thus, r'' is a good run that uses nonce n_B. By (c) above and by the definition of $can\_compute_i$, we have $(\mathcal{I}^{GDY}, r'', t'') \models (A \overset{n_B}{\rightleftharpoons} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r', t') \models K_B^0(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0})$, and $(\mathcal{I}^{GDY}, r, t) \models K_A^0(good \Rightarrow K_B^0(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0}))$, as desired.

Case: B believes A believes B believes $A \overset{n_B}{\rightleftharpoons} B$. Let r be a good run where k_i is i's public key and n_i is i's nonce ($i = A, B$). If $t \geq 0$ is such that $recv(\{\!|n_B|\!\}_{k_B}) \in r_B(t)$, then for all good runs r' with $(r', t') \sim_B (r, t)$, we must have $recv(\{\!|n_B|\!\}_{k_B}) \in r'_B(t')$. By (b) above, we have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r'_A(t')$.

Let r'' be a good run with $(r'', t'') \sim_A (r', t')$; we must have $recv(\{\!|n_A, n_B|\!\}_{k_A}) \in r''_A(t'')$. By (b) above, we have $send(\{\!|n_A, n_B|\!\}_{k_A}) \in r''_B(t'')$. Let r''' be a good run with $(r''', t''') \sim_B (r'', t'')$; we must have $send(\{\!|n_A, n_B|\!\}_{k_A}) \in r'''_B(t''')$. Thus, r''' is a good run that uses nonce n_B. By (c) above and by the definition of $can\_compute_i$, we have $(\mathcal{I}^{GDY}, r''', t''') \models (A \overset{n_B}{\rightleftharpoons} B)^{T,0}$, so that $(\mathcal{I}^{GDY}, r'', t'') \models K^0_B(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0})$, $(\mathcal{I}^{GDY}, r', t') \models K^0_A(good \Rightarrow K^0_B(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0}))$, and $(\mathcal{I}^{GDY}, r, t) \models K^0_B(good \Rightarrow K^0_A(good \Rightarrow K^0_B(good \Rightarrow (A \overset{n_B}{\rightleftharpoons} B)^{T,0})))$, as desired. □

Bibliography

Abadi, M. (1999). Secrecy by typing in security protocols. *Journal of the ACM 46*(5), 749–786.

Abadi, M. (2000). Security protocols and their properties. In *Foundations of Secure Computation*, NATO Science Series, pp. 39–60. IOS Press. Volume for the 20th International Summer School on Foundations of Secure Computation.

Abadi, M. and B. Blanchet (2003a). Computer-assisted verification of a protocol for certified email. In *Proc. 10th International Symposium on Static Analysis (SAS'03)*, Volume 2694 of *Lecture Notes in Computer Science*, pp. 316–335. Springer-Verlag.

Abadi, M. and B. Blanchet (2003b). Secrecy types for asymmetric communication. *Theoretical Computer Science 298*(3), 387–415.

Abadi, M. and C. Fournet (2001). Mobile values, new names, and secure communication. In *Proc. 28th Annual ACM Symposium on Principles of Programming Languages (POPL'01)*, pp. 104–115.

Abadi, M., C. Fournet, and G. Gonthier (2002). Secure implementation of channel abstractions. *Information and Computation 174*(1), 37–83.

Abadi, M. and A. D. Gordon (1998). A bisimulation method for cryptographic protocols. *Nordic Journal of Computing 5*(4), 267–303.

Abadi, M. and A. D. Gordon (1999). A calculus for cryptographic protocols: The Spi calculus. *Information and Computation 148*(1), 1–70.

Abadi, M. and R. Needham (1996). Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering 22*(1), 6–15.

Abadi, M. and P. Rogaway (2002). Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology 15*(2), 103–127.

Abadi, M. and M. R. Tuttle (1991). A semantics for a logic of authentica-

tion. In *Proc. 10th ACM Symposium on Principles of Distributed Computing (PODC'91)*, pp. 201–216.

Accorsi, R., D. Basin, and L. Viganò (2001). Towards an awareness-based semantics for security protocol analysis. In J. Goubault-Larrecq (Ed.), *Proc. Workshop on Logical Aspects of Cryptographic Protocol Verification*, Volume 55.1 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers.

Alchourrón, C. E., P. Gärdenfors, and D. Makinson (1985). On the logic of theory change: partial meet functions for contraction and revision. *Journal of Symbolic Logic 50*, 510–530.

Alpern, B. and F. B. Schneider (1985). Defining liveness. *Information Processing Letters 21*, 181–185.

Anderson, R. and R. Needham (1995). Programming Satan's computer. In J. van Leeuwen (Ed.), *Computer Science Today: Recent Trends and Developments*, Volume 1000 of *Lecture Notes in Computer Science*, pp. 426–440. Springer-Verlag.

Andrews, P. B. (1986). *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof*. Academic Press.

Asokan, N., V. Shoup, and M. Waidner (1998). Asynchronous protocols for optimistic fair exchange. In *Proc. 1998 IEEE Symposium on Security and Privacy*, pp. 86–99. IEEE Computer Society Press.

Aumann, R. J. (1976). Agreeing to disagree. *Annals of Statistics 4*(6), 1236–1239.

Aumann, R. J. (1999). Interactive epistemology I: knowledge. *International Journal of Game Theory 28*(3), 263–301.

Baader, F. and T. Nipkow (1998). *Term Rewriting and All That*. Cambridge University Press.

Basu, S. (1999). New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM 46*(4), 537–555.

Bellare, M. and P. Rogaway (1993). Entity authentication and key distribution. In *Proc. 13th Annual International Cryptology Conference (CRYPTO'93)*, Volume 773 of *Lecture Notes in Computer Science*, pp. 232–249.

Ben-Or, M., O. Goldreich, S. Micali, and R. L. Rivest (1990). A fair protocol for signing contracts. *IEEE Transactions on Information Theory 36*(1), 40–46.

Ben-Or, M., D. Kozen, and J. H. Reif (1986). The complexity of elementary algebra and geometry. *Journal of Computer and Systems Sciences 32*(1), 251–264.

Benthem, J. van (1984). Correspondence theory. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic. Volume II: Extensions of Classical Logic*, pp. 167–247. Reidel.

Berman, P., J. Garay, and K. J. Perry (1989). Towards optimal distributed consensus. In *Proc. 30th IEEE Symposium on the Foundations of Computer Science (FOCS'89)*, pp. 410–415.

Bhargavan, K., C. Fournet, A. D. Gordon, and R. Pucella (2004). TulaFale: A security tool for web services. In *Proc. 2nd International Symposium on Formal Methods for Components and Objects (FMCO'03)*. To appear in LNCS.

Bieber, P. (1990). A logic of communication in hostile environment. In *Proc. 3rd IEEE Computer Security Foundations Workshop (CSFW'90)*, pp. 14–22. IEEE Computer Society Press.

Billingsley, P. (1995). *Probability and Measure*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons.

Blackburn, P., M. de Rijke, and Y. Venema (2001). *Modal Logic*. Cambridge University Press.

Blanchet, B. (2001). An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pp. 82–96. IEEE Computer Society Press.

Blanchet, B. (2002). From secrecy to authenticity in security protocols. In *Proc. 9th International Static Analysis Symposium (SAS'02)*, Volume 2477 of *Lecture Notes in Computer Science*, pp. 342–359. Springer-Verlag.

Boreale, M., R. de Nicola, and R. Pugliese (2001). Proof techniques for cryptographic processes. *SIAM Journal of Computing 31*(3), 947–986.

Boyd, C. (1997). Extensional goals for authentication protocols. In *Proc. DIMACS Workshop on Cryptographic Protocol Design and Verification*. Available at `http://dimacs.rutgers.edu/Workshops/Security/`.

Brandenburger, A. (1989). The role of common knowledge assumptions in game theory. In F. Hahn (Ed.), *The Economics of Information, Games, and Missing Markets*. Oxford University Press.

Burris, S. and H. P. Sankappanavar (1981). *A Course in Universal Algebra*. Springer-Verlag.

Burrows, M., M. Abadi, and R. Needham (1990a). A logic of authentication. *ACM Transactions on Computer Systems 8*(1), 18–36.

Burrows, M., M. Abadi, and R. M. Needham (1990b). Rejoinder to Nessett. *ACM Operating Systems Review 24*(2), 39–40.

Butler, F., I. Cervesato, A. D. Jaggard, and A. Scedrov (2002). A formal analysis of some properties of Kerberos 5 using MSR. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pp. 175–190. IEEE Computer Society Press.

Canny, J. F. (1988). Some algebraic and geometric computations in PSPACE.

In *Proc. 20th Annual ACM Symposium on the Theory of Computing (STOC'88)*, pp. 460–467.

Carnap, R. (1962). *Logical Foundations of Probability* (Second ed.). University of Chicago Press.

Cervesato, I., N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov (1999). A meta-notation for protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pp. 55–69. IEEE Computer Society Press.

Cervesato, I., N. Durgin, J. Mitchell, P. Lincoln, and A. Scedrov (2000). Relating strands and multiset rewriting for security protocol analysis. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pp. 35–51. IEEE Computer Society Press.

Cherniak, C. (1986). *Minimal Rationality*. Bradford Books. MIT Press.

Chomsky, N. (1968). *Language and Mind*. Brace & World.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics 21*, 345–363.

Churchland, P. M. and P. S. Churchland (1983). Stalking the wild epistemic engine. *Nous 17*, 5–18.

Clark, J. and J. Jacob (1997). A survey of authentication protocol literature: Version 1.0. Unpublished manuscript.

Clarke, E., S. Jha, and W. Marrero (1998). Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proc. IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*.

Cohen, E. (2000). TAPS: A first-order verifier for cryptographic protocols. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pp. 144–158. IEEE Computer Society Press.

Cohen, E. (2002). TAPS: The last few slides. In *Proc. Workshop on Formal Aspects of Security (FASec'02)*, Volume 2629 of *Lecture Notes in Computer Science*, pp. 183–190.

Cook, S. A. (1971). The complexity of theorem proving procedures. In *Proc. 3rd Annual ACM Symposium on the Theory of Computing (STOC'71)*, pp. 151–158.

Copeland, B. J. (2002). The Church-Turing thesis. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy (Fall 2002 Edition)*. `http://plato.stanford.edu/archives/fall2002/entries/church-turing/`.

Crazzolara, F. and G. Winskel (2001). Events in security protocols. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pp. 96–105. ACM Press.

Cresswell, M. J. (1972). Intensional logics and logical truth. *Journal of Philosophical Logic 1*, 2–15.

Cresswell, M. J. (1973). *Logics and Languages*. Methuen & Co Ltd.

Davis, R., H. Shrobe, and P. Szolovits (1993). What is a knowledge representation? *AI Magazine 14*(1), 17–33.

de Alfaro, L. (1998). *Formal Verification of Probabilistic Systems*. Ph. D. thesis, Stanford University. Available as Technical Report STAN-CS-TR-98-1601.

Dekker, A. H. (2000). C3PO: A tool for automatic sound cryptographic protocol analysis. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pp. 77–87. IEEE Computer Society Press.

Dienes, Z. and J. Perner (1999). A theory of implicit and explicit knowledge. *Behavioural and Brain Sciences 22*, 735–755.

Dolev, D. and A. C. Yao (1983). On the security of public key protocols. *IEEE Transactions on Information Theory 29*(2), 198–208.

Duc, H. N. (2001). *Resource-Bounded Reasoning about Knowledge*. Ph. D. thesis, Universität Leipzig.

Durgin, N., J. Mitchell, and D. Pavlovic (2001). A compositional logic for protocol correctness. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pp. 241–255. IEEE Computer Society Press.

Dwork, C. and Y. Moses (1990). Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation 88*(2), 156–186.

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.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press.

Even, S., O. Goldreich, and A. Shamir (1985). On the security of ping-pong protocols when implemented using the RSA. In *Proc. Conference on Advances in Cryptology (CRYPTO'85)*, Volume 218 of *Lecture Notes in Computer Science*, pp. 58–72. Springer-Verlag.

Fagin, R. and J. Y. Halpern (1988). Belief, awareness, and limited reasoning. *Artificial Intelligence 34*, 39–76.

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, and N. Megiddo (1990). A logic for reasoning about probabilities. *Information and Computation 87*(1/2), 78–128.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning about Knowledge*. MIT Press.

Fagin, R., J. Y. Halpern, and M. Y. Vardi (1990). A nonstandard approach to the logical omniscience problem. In *Proc. 3rd Conference on Theoretical As-*

pects of Reasoning about Knowledge (TARK'90), pp. 41–55. Morgan Kaufmann.

Feller, W. (1957). *An Introduction to Probability Theory and its Applications* (Second ed.), Volume 1. John Wiley & Sons.

Fischer, M. J. and L. D. Zuck (1988). Reasoning about uncertainty in fault-tolerant distributed systems. Technical Report YALEU/DCS/TR–643, Yale University.

Fitelson, B. (1999). The plurality of Bayesian measures of confirmation and the problem of measure sensitivity. *Philosophy of Science 66 (r:supplement)*, S362–378.

Fitting, M. and R. Mendelsohn (1998). *First Order Modal Logic*. Kluwer Academic Publishers.

Focardi, R., R. Gorrieri, and F. Martinelli (2003). A comparison of three authentication properties. *Theoretical Computer Science 291*(3), 285–327.

Fodor, J. A. (1976). *The Language of Thought*. Harvester Press.

Fodor, J. A. (1981). Propositional attitudes. In J. A. Fodor (Ed.), *Representations: Philosophical Essays on the Foundations of Cognitive Science*, pp. 177–203. Harvester Press.

Fournet, C. and G. Gonthier (1996). The reflexive chemical abstract machine and the join calculus. In *Proc. 23rd Annual ACM Symposium on Principles of Programming Languages (POPL'96)*, pp. 372–385. ACM Press.

Frendrup, U., H. Hüttel, and J. N. Jensen (2002). Modal logics for cryptographic processes. In *Proc. 9th International Workshop on Expressiveness in Concurrency (EXPRESS'02)*, Volume 68.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers.

Fudenberg, D. and J. Tirole (1991). *Game Theory*. MIT Press.

Gabbay, D., A. Pnueli, S. Shelah, and J. Stavi (1980). On the temporal analysis of fairness. In *Proc. 7th Annual ACM Symposium on Principles of Programming Languages (POPL'80)*, pp. 163–173.

Gallin, D. (1975). *Intensional and Higher-Order Modal Logic*, Volume 19 of *Mathematics Studies*. North-Holland.

Gamow, G. and M. Stern (1958). *Puzzle Math*. Viking Press.

Garson, J. W. (1984). Quantification in modal logic. In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic. Volume II: Extensions of Classical Logic*, pp. 249–307. Reidel.

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science 50*, 1–102.

Giunchiglia, F., L. Serafini, E. Giunchiglia, and M. Frixione (1993). Non-omniscient belief as context-based reasoning. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pp. 548–554.

Goldblatt, R. (1992). *Logics of Time and Computation*. CSLI Lecture Notes, No. 7. CSLI.

Goldreich, O. (1998). *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, Volume 17 of *Algorithms and Combinatorics*. Springer-Verlag.

Goldreich, O. (2001). *Foundations of Cryptography: Basic Tools*. Cambridge University Press.

Goldwasser, S. and S. Micali (1982). Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc. 14th Annual ACM Symposium on the Theory of Computing (STOC'82)*, pp. 365–377. ACM Press.

Gollmann, D. (1996). What do we mean by entity authentication? In *Proc. 1996 IEEE Symposium on Security and Privacy*, pp. 46–54. IEEE Computer Society Press.

Gollmann, D. (2003). Authentication by correspondence. *IEEE Journal on Selected Areas in Communications 21*(1), 88–95.

Gong, L., R. Needham, and R. Yahalom (1990). Reasoning about belief in cryptographic protocols. In *Proc. 1990 IEEE Symposium on Security and Privacy*, pp. 234–248. IEEE Computer Society Press.

Good, I. J. (1950). *Probability and the Weighing of Evidence*. Charles Griffin & Co. Ltd.

Good, I. J. (1960). Weights of evidence, corroboration, explanatory power, information and the utility of experiments. *Journal of the Royal Statistical Society, Series B 22*, 319–331.

Gordon, A. D. and A. Jeffrey (2001). Authenticity by typing for security protocols. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pp. 145–159. IEEE Computer Society Press.

Gordon, A. D. and A. Jeffrey (2002a). Types and effects for asymmetric cryptography. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pp. 77–91. IEEE Computer Society Press.

Gordon, A. D. and A. Jeffrey (2002b). Typing one-to-one and one-to-many correspondences in security protocols. In *Proceeding of Software Security - Theories and Systems (ISSS 2002)*, Volume 2609 of *Lecture Notes in Computer Science*, pp. 263–282. Springer-Verlag.

Gray, III, J. W. and J. McLean (1995). Using temporal logic to specify and verify cryptographic protocols. In *Proc. 8th IEEE Computer Security Foundations Workshop (CSFW'95)*, pp. 108–117. IEEE Computer Society Press.

Grove, A. J. (1995). Naming and identity in epistemic logic II: a first-order logic for naming. *Artificial Intelligence 74*(2), 311–350.

Grove, A. J. and J. Y. Halpern (1993). Naming and identity in epistemic logics,

Part I: the propositional case. *Journal of Logic and Computation 3*(4), 345–378.

Guttman, J. D. and F. J. Thayer (2002). Authentication tests and the structure of bundles. *Theoretical Computer Science 283*(2), 333–380.

Guttman, J. D., F. J. Thayer, and L. D. Zuck (2001). The faithfulness of abstract protocol analysis: message authentication. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pp. 186–195. ACM Press.

Halpern, J. Y. (1990). An analysis of first-order logics of probability. *Artificial Intelligence 46*, 311–350.

Halpern, J. Y. (2000). A note on knowledge-based programs and specifications. *Distributed Computing 13*, 145–153.

Halpern, J. Y. (2003). *Reasoning About Uncertainty*. MIT Press.

Halpern, J. Y. and R. Fagin (1992). Two views of belief: belief as generalized probability and belief as evidence. *Artificial Intelligence 54*, 275–317.

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 Annual ACM Symposium on the Theory of Computing (STOC'88)*, pp. 132–147.

Halpern, J. Y., Y. Moses, and M. Y. Vardi (1994). Algorithmic knowledge. In *Proc. 5th Conference on Theoretical Aspects of Reasoning about Knowledge (TARK'94)*, pp. 255–266. Morgan Kaufmann.

Halpern, J. Y. and K. O'Neill (2002). Secrecy in multiagent systems. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pp. 32–46. IEEE Computer Society Press.

Halpern, J. Y. and K. O'Neill (2003). Anonymity and information hiding in multiagent systems. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pp. 75–88. IEEE Computer Society Press.

Halpern, J. Y. and R. Pucella (2002). Modeling adversaries in a logic for reasoning about security protocols. In *Proc. Workshop on Formal Aspects of Security (FASec'02)*, Volume 2629 of *Lecture Notes in Computer Science*, pp. 115–132.

Halpern, J. Y. and R. Pucella (2003a). A logic for reasoning about evidence. In *Proc. 19th Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pp. 297–304.

Halpern, J. Y. and R. Pucella (2003b). On the relationship between strand spaces and multi-agent systems. *ACM Transactions on Information and System Security 6*(1), 43–70.

Halpern, J. Y. and R. Pucella (2003c). Probabilistic algorithmic knowledge. In

Proc. 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'03), pp. 118–130.

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 (1991). 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. Academic Press.

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.

Hamlyn, D. W. (1970). *The Theory of Knowledge*. MacMillian Press.

Harel, D., D. Kozen, and J. Tiuryn (2000). *Dynamic Logic*. MIT Press.

Harman, G. (1973). *Thought*. Princeton University Press.

He, J., K. Seidel, and A. McIver (1997). Probabilistic models for the guarded command language. *Science of Computer Programming 28*(2–3), 171–192.

Hennessy, M. and R. Milner (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the ACM 32*, 137–161.

Higgins, P. J. (1963). Algebras with a scheme of operators. *Mathematische Nachrichten 27*, 115–132.

Hintikka, J. (1962). *Knowledge and Belief*. Cornell University Press.

Hintikka, J. (1975). Impossible possible worlds vindicated. *Journal of Philosophical Logic 4*, 475–484.

Hoare, C. (1985). *Communicating Sequential Processes*. Prentice-Hall.

Huang, Z. and K. Kwast (1991). Awareness, negation and logical omniscience. In J. v. Eijck (Ed.), *Proc. European Workshop on Logics in AI (JELIA'90)*, Volume 478 of *Lecture Notes in Computer Science*, pp. 282–300. Springer-Verlag.

Huber, P. J. (1981). *Robust Statistics*. Wiley Interscience.

Hughes, G. and M. Cresswell (1972). *An Introduction to Modal Logic*. Methuen.

Impagliazzo, R. and B. M. Kapron (2003). Logics for reasoning about cryptographic constructions. In *Proc. 44th IEEE Symposium on the Foundations of Computer Science (FOCS'03)*, pp. 372–383.

Jeffrey, R. C. (1992). *Probability and the Art of Judgement*. Cambridge University Press.

Johnson-Laird, P. (1987). How could consciousness arise from the computations of the brain? In C. Blakemore and S. Greenfield (Eds.), *Mindwaves*. Basil Blackwell.

Kaplan, A. N. and L. K. Schubert (2000). A computational model of belief. *Artificial Intelligence 120*, 119–160.

Konolige, K. (1986). *A Deduction Model of Belief*. Morgan Kaufmann.

Kripke, S. (1963). A semantical analysis of modal logic I: normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9*, 67–96.

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. North-Holland.

Kyburg, Jr., H. E. (1974). *The Logical Foundations of Statistical Inference*. Reidel.

Kyburg, Jr., H. E. (1983). Recent work in inductive logic. In T. Machan and K. Lucey (Eds.), *Recent Work in Philosophy*, pp. 87–150. Rowman & Allanheld.

Ladner, R. E. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing 6*(3), 467–480.

Lakshmanan, L. V. S. and F. Sadri (2001). On a theory of probabilistic deductive databases. *Theory and Practice of Logic Programming 1*(1), 5–42.

Lemmon, E. J. (1966a). Algebraic semantics for modal logics I. *Journal of Symbolic Logic 31*(1), 46–65.

Lemmon, E. J. (1966b). Algebraic semantics for modal logics II. *Journal of Symbolic Logic 31*(4), 191–218.

Levesque, H. J. (1984). A logic of implicit and explicit belief. In *Proc. 4th National Conference on Artificial Intelligence (AAAI'84)*, pp. 198–202.

Levi, I. (1980). *The Enterprise of Knowledge*. MIT Press.

Lewis, C. I. and C. H. Langford (1959). *Symbolic Logic* (2nd ed.). Dover.

Lincoln, P., J. C. Mitchell, M. Mitchell, and A. Scedrov (1998). A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security (CCS'98)*, pp. 112–121.

Lipman, B. L. (1999). Decision theory without logical omniscience: Toward an axiomatic framework for bounded rationality. *The Review of Economic Studies 66*(2), 339–361.

Lowe, G. (1995). An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters 56*, 131–133.

Lowe, G. (1997). A hierarchy of authentication specifications. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pp. 31–43. IEEE Computer Society Press.

Lowe, G. (1998). Casper: A compiler for the analysis of security protocols. *Journal of Computer Security 6*, 53–84.

Lowe, G. (2002). Analysing protocols subject to guessing attacks. In *Proc. Workshop on Issues in the Theory of Security (WITS'02)*.

Lukasiewicz, T. (1999). Probabilistic deduction with conditional constraints over basic events. *Journal of Artificial Intelligence Research 10*, 199–241.

Mao, W. (1995). An augmentation of BAN-like logics. In *Proc. 8th IEEE Computer Security Foundations Workshop (CSFW'95)*, pp. 44–56. IEEE Computer Society Press.

McAllester, D. (1993). Automatic recognition of tractability in inference relations. *Journal of the ACM 40*(2), 284–303.

McCarthy, J. (1979). Ascribing mental qualities to machines. Technical Report STAN-CS-79-725, Stanford University.

McKinsey, J. J. C. and A. Tarski (1944). The algebra of topology. *Annals of Mathematics 45*, 141–191.

McLean, J. (1994). Security models. In J. Marciniak (Ed.), *Encyclopedia of Software Engineering*. Wiley Press.

Meadows, C. (1990). Representing partial knowledge in an algebraic security model. In *Proc. 3rd IEEE Computer Security Foundations Workshop (CSFW'90)*, pp. 23–31. IEEE Computer Society Press.

Meadows, C. (1996). The NRL protocol analyzer: An overview. *Journal of Logic Programming 26*(2), 113–131.

Merritt, M. and P. Wolper (1985). States of knowledge in cryptographic protocols. Unpublished manuscript.

Meyden, R. van der (1998). Common knowledge and update in finite environments. *Information and Computation 140*(2), 115–157.

Meyden, R. van der and N. V. Shilov (1999). Model checking knowledge and time in systems with perfect recall (extended abstract). In *Proc. Conference on Foundations of Software Technology and Theoretical Computer Science*, Volume 1738 of *Lecture Notes in Computer Science*, pp. 432–445. Springer-Verlag.

Meyden, R. van der and K. Su (2004). Symbolic model checking the knowledge of the dining cryptographers. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pp. 280–291. IEEE Computer Society Press.

Meyer, J.-J. C. and W. van der Hoek (1995). *Epistemic Logic for AI and Computer Science*, Volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Millen, J. and V. Shmatikov (2003). Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pp. 47–61. IEEE Computer Society Press.

Millen, J. K., S. C. Clark, and S. B. Freedman (1987). The Interrogator: Protocol

security analysis. *IEEE Transactions on Software Engineering 13*(2), 274–288.

Milne, P. (1996). $\log[p(h|eb)/p(h|b)]$ is the one true measure of confirmation. *Philosophy of Science 63*, 21–26.

Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall.

Milner, R. (1999). *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press.

Milner, R., J. Parrow, and D. Walker (1993). Modal logics for mobile processes. *Theoretical Computer Science 114*(1), 149–171.

Mitchell, J., M. Mitchell, and U. Stern (1997). Automated analysis of cryptographic protocols using Murφ. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pp. 141–151. IEEE Computer Society Press.

Moreno, A. (1998). Avoiding logical omniscience and perfect reasoning: A survey. *AI Communications 11*(2), 101–122.

Moses, Y. (1988). Resource-bounded knowledge. In *Proc. 2nd Conference on Theoretical Aspects of Reasoning about Knowledge (TARK'88)*, pp. 261–276. Morgan Kaufmann.

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.

Motwani, R. and P. Raghavan (1995). *Randomized Algorithms*. Cambridge University Press.

Needham, R. M. and M. D. Schroeder (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM 21*(12), 993–999.

Nerode, A. and R. Shore (1994). *Logic for Applications*. Springer-Verlag.

Nessett, D. M. (1990). A critique of the Burrows, Abadi and Needham logic. *ACM Operating Systems Review 24*(2), 35–38.

Neumann, J. von and O. Morgenstern (1947). *Theory of Games and Economic Behavior* (2nd ed.). Princeton University Press.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence 28*(1), 71–88.

Papadimitriou, C. (1994). *Computational Complexity*. Addison Wesley.

Parikh, R. (1987). Knowledge and the problem of logical omniscience. In *Proc. 2nd International Symposium on Methodologies for Intelligent Systems*, pp. 432–439. North-Holland.

Paulson, L. C. (1994). *Isabelle, A Generic Theorem Prover*, Volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag.

Paulson, L. C. (1997). Mechanized proofs for a recursive authentication

protocol. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pp. 84–95.

Paulson, L. C. (1998). The inductive approach to verifying cryptographic protocols. *Journal of Computer Security 6*(1/2), 85–128.

Penrose, R. (1989). *The Emperor's New Mind*. Oxford University Press.

Penrose, R. (1994). *Shadows of the Mind*. Oxford University Press.

Plotkin, G. D. (1981). A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus.

Pnueli, A. (1977). The temporal logic of programs. In *Proc. 18th IEEE Symposium on the Foundations of Computer Science (FOCS'77)*, pp. 46–57.

Pollock, J. L. and J. Cruz (1999). *Contemporary Theories of Knowledge* (Second ed.). Rowman & Littlefield.

Popkorn, S. (1994). *First Steps in Modal Logic*. Cambridge University Press.

Popper, K. R. (1959). *The Logic of Scientific Discovery*. Hutchinson.

Prior, A. N. (1957). *Time and Modality*. Oxford University Press.

Pucella, R. (2004). Deductive algorithmic knowledge. In *Proc. 8th International Symposium on Artificial Intelligence and Mathematics*. AI&M 22-2004.

Rabin, M. O. (1980). Probabilistic algorithm for testing primality. *Journal of Number Theory 12*, 128–138.

Ramanujam, R. (1999). View-based explicit knowledge. *Annals of Pure and Applied Logic 96*(1–3), 343–368.

Rantala, V. (1975). Urn models: a new kind of non-standard model for first-order logic. *Journal of Philosophical Logic 4*, 455–474.

Rantala, V. (1982). Impossible worlds semantics and logical omniscience. *Acta Philosophica Fennica 35*, 18–24.

Renegar, J. (1992). On the computational complexity and geometry of the first order theory of the reals. *Journal of Symbolic Computation 13*(3), 255–352.

Rescher, N. and R. Brandom (1979). *The Logic of Inconsistency*. Rowman and Littlefield.

Roscoe, A. W. (1994). Model-checking CSP. In *A Classical Mind, Essays in Honour of C. A. R. Hoare*, pp. 353–378. Prentice-Hall.

Roscoe, A. W. (1996). Intensional specifications of security protocols. In *Proc. 9th IEEE Computer Security Foundations Workshop (CSFW'96)*, pp. 28–38. IEEE Computer Society Press.

Roscoe, A. W. (1997). *The Theory and Practice of Concurrency*. Prentice-Hall.

Rosenschein, S. J. (1985). Formal theories of knowledge in AI and robotics. *New Generation Computing 3*(4), 345–357.

Rubinstein, A. (1998). *Modeling Bounded Rationality*. MIT Press.

Ryan, P. and S. Schneider (2000). *Modelling and Analysis of Security Protocols*. Addison Wesley.

Ryan, P. Y. A. and S. A. Schneider (1998). An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters 65*(1), 7–10.

Ryle, G. (1949). *The Concept of Mind*. Hutchinson & Company, Ltd.

Sangiorgi, D. and D. Walker (2001). *Pi-calculus: A Theory of Mobile Systems*. Cambridge University Press.

Schneider, S. (1996). Security properties and CSP. In *Proc. 1996 IEEE Symposium on Security and Privacy*, pp. 174–187. IEEE Computer Society Press.

Schneider, S. (1998). Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering 24*(9), 741–758.

Schneier, B. (1996). *Applied Cryptography* (Second ed.). John Wiley & Sons.

Searle, J. (1992). *The Rediscovery of the Mind*. MIT Press.

Selman, B. and H. Kautz (1996). Knowledge compilation and theory approximation. *Journal of the ACM 43*(2), 193–224.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Shafer, G. (1982). Belief functions and parametric models (with commentary). *Journal of the Royal Statistical Society, Series B 44*, 322–352.

Shmatikov, V. and J. C. Mitchell (2000). Analysis of a fair exchange protocol. In *Seventh Annual Symposium on Network and Distributed System Security (NDSS 2000)*, pp. 119–128.

Shoenfield, J. R. (1967). *Mathematical Logic*. Addison Wesley.

Snekkenes, E. (1991). Exploring the BAN approach to protocol analysis. In *Proc. 1991 IEEE Symposium on Security and Privacy*, pp. 171–181. IEEE Computer Society Press.

Snekkenes, E. (1992). Roles in cryptographic protocols. In *Proc. 1992 IEEE Symposium on Security and Privacy*, pp. 105–119. IEEE Computer Society Press.

Song, D. X. (1999). Athena: a new efficient automatic checker for security protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pp. 192–202. IEEE Computer Society Press.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co.

Stalnaker, R. (1984). *Inquiry*. MIT Press.

Stalnaker, R. (1987). Semantics for belief. *Philosophical Topics 15*(1), 177–190.

Stalnaker, R. (1991). The problem of logical omniscience, I. *Synthese 89*, 425–440.

Stalnaker, R. (1996). Impossibilities. *Philosophical Topics 24*, 193–204.

Steiglitz, K. (1988). Two nonstandard paradigms for computation: Analog machines and cellular automata. In J. K. Skwirzynski (Ed.), *Performance Limits*

in Communication Theory and Practice, Number 142 in NATO Advanced Study Institutes Series E, pp. 173–192. Kluwer Academic Publishers.

Stinson, D. R. (1995). *Cryptography: Theory and Practice*. CRC Press.

Stirling, C. (2001). *Modal and Temporal Properties of Processes*. Springer-Verlag.

Stubblebine, S. and R. Wright (1996). An authentication logic supporting synchronization, revocation, and recency. In *Proc. 3rd ACM Conference on Computer and Communications Security (CCS'96)*. ACM Press.

Syverson, P. (1990). A logic for the analysis of cryptographic protocols. NRL Report 9305, Naval Research Laboratory.

Syverson, P. and I. Cervesato (2001). The logic of authentication protocols. In *Proc. 1st International School on Foundations of Security Analysis and Design (FOSAD'00)*, Volume 2171 of *Lecture Notes in Computer Science*, pp. 63–137.

Syverson, P. and C. Meadows (1996). A formal language for cryptographic protocol requirements. *Designs, Codes, and Cryptography 7*(1/2), 27–59.

Syverson, P. F. and P. C. van Oorschot (1994). On unifying some cryptographic protocol logics. In *Proc. 1994 IEEE Symposium on Security and Privacy*, pp. 14–28. IEEE Computer Society Press.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry* (Second ed.). University of California Press.

Thayer, F. J., J. C. Herzog, and J. D. Guttman (1999a). Mixed strand spaces. In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW'99)*. IEEE Computer Society Press.

Thayer, F. J., J. C. Herzog, and J. D. Guttman (1999b). Strand spaces: Proving security protocols correct. *Journal of Computer Security 7*(2/3), 191–230.

Tofte, M. and J.-P. Talpin (1997). Region-based memory management. *Information and Computation 132*(2), 109–176.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2 42*, 230–265.

Vardi, M. Y. (1985). Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symposium on the Foundations of Computer Science (FOCS'85)*, pp. 327–338.

Walley, P. (1987). Belief function representations of statistical evidence. *Annals of Statistics 18*(4), 1439–1465.

Wansing, H. (1990). A general possible worlds framework for reasoning about knowledge and belief. *Studia Logica 49*(4), 523–539.

Wedel, G. and V. Kessler (1996). Formal semantics for authentication logics. In *Proc. 4th European Symposium on Research in Computer Security (ES-*

ORICS'96), Volume 1146 of *Lecture Notes in Computer Science*, pp. 219–241. Springer-Verlag.

Weispfenning, V. (1988). The complexity of linear problems in fields. *Journal of Symbolic Computation 5*(1/2), 3–27.

Winskel, G. (1993). *The Formal Semantics of Programming Languages*. MIT Press.

Woo, T. Y. C. and S. S. Lam (1992). Authentication for distributed systems. *Computer 25*(1), 39–52.

Woo, T. Y. C. and S. S. Lam (1993). A semantic model for authentication protocols. In *Proc. 1993 IEEE Symposium on Security and Privacy*, pp. 178–194.