

Deductive Algorithmic Knowledge*

Riccardo Pucella
Department of Computer Science
Cornell University
Ithaca, NY 14853
riccardo@cs.cornell.edu

Abstract

The framework of algorithmic knowledge assumes that agents use algorithms to compute the facts they explicitly know. In many cases of interest, a logical theory, rather than a particular algorithm, can be used to capture the formal reasoning used by the agents to compute what they explicitly know. We introduce a logic for reasoning about both implicit and explicit knowledge, where the latter is given with respect to a deductive system formalizing a logical theory for agents. The highly structured nature of such logical theories leads to a very natural axiomatization of the resulting logic over a deductive system. In the case when the deductive system is in fact tractable, we show that the decision problem for the logic is NP-complete, no harder than propositional logic.

1 Introduction

It is well known that the standard model of knowledge based on possible worlds is useful when reasoning about knowledge ascribed to agents by an external observer, but is not appropriate when agents need to explicitly compute what they know in order to make decisions. Part of the problem is that such a form of knowledge is subject to the problem of logical omniscience, that is, the agents know all the logical consequences of their knowledge [Fagin, Halpern, Moses, and Vardi 1995, Chapter 9].

Levesque [1984], among others, distinguishes implicit knowledge as described above from *explicit knowledge*, which represents the knowledge that is available to the agent in order to make decisions. A general approach to model explicit knowledge is that of *algorithmic knowledge* [Fagin, Halpern, Moses, and Vardi 1995, Chapter 10]. In the algorithmic knowledge framework, the explicit knowledge of the agents is given by an algorithm that the agents use to establish whether they know a particular fact. The algorithmic knowledge framework has the advantage of being general; this same generality also means that there are no specific properties of algorithmic knowledge unless we focus on specific algorithms.

In this paper, 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

*Work supported in part by NSF under grant CTC-0208535, by ONR under grant 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.

their reasoning about the facts they know. Many useful forms of explicit knowledge can be formalized in such a logical theory for agents. This can be viewed as a form of algorithmic knowledge, where the 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 the properties of explicit knowledge in this context. Intuitively, the deduction rules of the logical theory directly translate into properties of explicit knowledge.

To motivate the use of logical theories to capture explicit knowledge, consider the following example. In previous work [Halpern and Pucella 2002], we showed how the framework of algorithmic knowledge could be used to reason about agents communicating via cryptographic protocols. Algorithmic knowledge is useful to model an adversary that has a certain number of capabilities to decode the messages it intercepts. There are of course restrictions on the capabilities of a reasonable adversary. For instance, the adversary may not explicitly know that it has a given message if that message is encrypted using a key that the adversary does not know. To capture these restrictions, Dolev and Yao [1983] gave a now-standard description of capabilities of adversaries. Roughly speaking, a Dolev-Yao adversary can decompose messages, or decipher them if it 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 it has intercepted, and what messages the adversary can construct.

To reason about such examples, we introduce a modal logic that lets us reason about both the implicit knowledge of agents, which is useful for specifications, and the explicit knowledge of agents, formalized as a logical theory. In contrast with existing work on developing models for agents reasoning via logical theories (for instance, Konolige [1986] and Giunchiglia *et al* [1993]), which reason completely within the logical theories, assuming perhaps a global logical theory for the world, we base our logic on a standard possible-worlds semantics. This 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. Our approach moreover easily extends to the probabilistic setting, such as when there is a probability measure over the possible worlds [Fagin and Halpern 1994]. We focus in this paper on the technical properties of the resulting logic.

The rest of this paper is structured as follows. In the next section, we give the background on deductive systems, that we use to define the logical theories of the agents. In Section 3, we introduce the formal logic to reason about knowledge given by deductive systems. In Section 4, we show how we derive sound and complete axiomatizations for the logic for reasoning about particular deductive systems. In Section 5, we address the complexity of the decision procedure for the logic. We conclude with a discussion of natural extensions of our framework. For reasons of space, we leave these extensions as well as the proof of our technical results for the full paper.

2 Deductive Systems

We start with defining the framework in which we capture the logical theories of the agents, their deductive or inferential powers. Reasoning occurs over the terms of some term algebra. More

precisely, assume a fixed finite signature $\Sigma = (f_1, \dots, f_n)$, where each f_i is an operation symbol, with arity r_i . (Operation symbols of arity 0 are called constants.) Assume a 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 $t_1, \dots, t_n \in T_\Sigma$, then $f(t_1, \dots, t_n) \in T_\Sigma$. Intuitively, T_Σ contains all the terms that can be built from the variables, constants, and operations in Σ . We say 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 substitution ρ is a mapping from variables in $Vars$ to ground terms. The application of a 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 substitution to a term yields a ground term.

A *deductive system* D is a subset of $\wp(T_\Sigma) \times T_\Sigma$. A *deduction rule* $(\{t_1, \dots, t_n\}, t)$ of D is typically written $t_1, \dots, t_n \triangleright t$, and means that t can be immediately deduced from t_1, \dots, t_n . A deduction of t from a set Γ of terms is a sequence of ground terms t_1, \dots, t_n , such that $t_n = t$, and every t_i is either (1) of the form $\rho(t')$, for some substitution ρ and some term $t' \in \Gamma$, or (2) of the form $\rho(t')$, for some substitution ρ and some term t' for which there is a deduction rule $t'_{i_1}, \dots, t'_{i_k} \triangleright t'$ in D , where $\rho(t'_{i_j}) = t_{i_j}$ for all j , and $i_1, \dots, i_k < i$. 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 .

Example 2.1: The following deductive system DY over the signature $\Sigma = (\text{recv}, \text{has}, \text{encr}, \text{conc}, \text{inv})$ captures the Dolev-Yao adversary described in the introduction. Here, $\text{recv}(m)$ represents the fact that the adversary has received the term m , $\text{has}(m)$ represents the fact that the adversary understands the term m , $\text{encr}(m, k)$ represents the encryption of term m with key k , $\text{conc}(m_1, m_2)$ represents the concatenation of terms m_1 and m_2 , and $\text{inv}(k)$ represents the inverse of the key k .

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

Assume further that Σ contains constants such as m, k_1, k_2 . We can therefore derive, for instance, that $\text{recv}(\text{encr}(m, k_1)), \text{recv}(\text{encr}(\text{inv}(k_1), k_2)), \text{recv}(\text{inv}(k_2)) \vdash_{\text{DY}} \text{has}(m)$. In other words, it is possible for a Dolev-Yao adversary to derive the message m if it has received m encrypted under a key k_1 , which inverse it has received encrypted under a key k_2 , which inverse it has received.

To account for constructing new messages, consider the signature Σ' which extends Σ with a unary constructor constr , where $\text{constr}(m)$ 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.

$$\begin{aligned} \text{has}(m) &\triangleright \text{constr}(m) \\ \text{constr}(k), \text{constr}(m) &\triangleright \text{constr}(\text{encr}(m, k)) \\ \text{constr}(m_1), \text{constr}(m_2) &\triangleright \text{constr}(\text{conc}(m_1, m_2)) \end{aligned}$$

For instance, we have $\text{recv}(\text{encr}(m, k_1)), \text{recv}(\text{inv}(k_1)), \text{recv}(k_2) \vdash_{\text{DY}} \text{constr}(\text{encr}(m, k_2))$. ■

It should be clear from the definitions that the deductive systems we consider in this paper are monotonic. In other words, if $\Gamma \vdash_D t$, then $\Gamma' \vdash_D t$ when $\Gamma \subseteq \Gamma'$. While the framework we introduce in Section 3 would remain unchanged were we to consider nonmonotonic deductive systems, the axiomatizations of Section 4 would not be possible.

3 Deductive Algorithmic Knowledge

We now introduce a simple modal logic for reasoning about the implicit and explicit knowledge of an agent, where the explicit knowledge is formalized as a logical theory. For simplicity, we focus on the single agent case, and moreover on the purely propositional case. We start by introducing the simple setting, where the deductive system is used to reason about primitive propositions. We then extend it to full propositional formulas.

Primitive Deductive Systems. We first define the logic $\mathcal{L}^{\text{prim}}(\Sigma)$, over a signature Σ . We take the primitive propositions to be T_Σ^g , the ground terms over Σ . The language of the logic is obtained by starting with the primitive propositions in T_Σ^g , and closing off under negation, conjunction, the K operator, and the X operator applied to primitive propositions only. Intuitively, $K\varphi$ is read as “the agent implicitly knows φ ”, while Xp is read as “the agent knows p according to his logical theory”. (We extend the framework so that X applies to more general formulas later in this section.) We define the usual abbreviations, $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, and $\varphi \Rightarrow \psi$ for $\neg\varphi \vee \psi$. We define *true* as an abbreviation for an arbitrary but fixed propositional tautology, and *false* as an abbreviation for $\neg\text{true}$.

We use a special form of deductive system, that essentially takes into account observations that the agent can make. A *primitive deductive system* D over Σ is a deductive system with a distinguished class $Obs \subseteq T_\Sigma^g$ such that, intuitively, no term in Obs arises as the conclusion of a deductive rule in D . Formally, for all $ob \in Obs$ and for all rules $t_1, \dots, t_n \triangleright t$ of D , there does not exist a substitution ρ such that $\rho(t) = ob$.

The semantics of the logic follows the standard possible-worlds presentation for modal logics of knowledge [Hintikka 1962]. A *deductive knowledge structure* is a tuple $M = (S, \pi, D)$, where S is a set of states, π is an interpretation of the primitive propositions, and D is a primitive deductive system over Σ , with observation set Obs . Every state s in S 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 finite set of observations, taken from Obs , representing the observations that the agent has made at that state. Hence, $S \subseteq E \times \wp(Obs)$.¹ The interpretation π associates with every state the set of primitive propositions that are true at that state, so that for every primitive proposition $p \in T_\Sigma^g$, we have $\pi(s)(p) \in \{\mathbf{true}, \mathbf{false}\}$. The only assumption we make is that the interpretation respects the observations made at a state, that is, $\pi(e, obs)(ob) = \mathbf{true}$ if and only if $ob \in obs$.

Let $\mathcal{M}(\Sigma)$ be the set of all deductive knowledge structures with signature Σ , while for a fixed deductive system D over Σ , let $\mathcal{M}_D(\Sigma)$ be the set of all deductive knowledge structures over the deductive system D .

We define a relation between states that captures the states that the agent cannot distinguish based on the observations. Define $s \sim s'$ if $s = (e, obs)$ and $s' = (e', obs)$ for some e, e' , and set of observations obs . Clearly, \sim is an equivalence relation over the states.

We define what it means for a formula φ to be true at a state s of M , written $(M, s) \models \varphi$, inductively as follows.

$$(M, s) \models p \text{ if } \pi(s)(p) = \mathbf{true}$$

¹For simplicity, we assume that the observations form a set. This implies that repetition of observations and their order is unimportant. We can easily model the case where the observations form a sequence, at the cost of complicating the presentation.

$$\begin{aligned}
(M, s) &\models \neg\varphi \text{ if } (M, s) \not\models \varphi \\
(M, s) &\models \varphi \wedge \psi \text{ if } (M, s) \models \varphi \text{ and } (M, s) \models \psi \\
(M, s) &\models K\varphi \text{ if } (M, s') \models \varphi \text{ for all } s' \sim s \\
(M, s) &\models Xp \text{ if } s = (e, \{ob_1, \dots, ob_n\}) \text{ and } ob_1, \dots, ob_n \vdash_D p.
\end{aligned}$$

Example 3.1: Consider the deductive system DY from Example 2.1. This deductive system can be viewed as a primitive deductive system by taking

$$Obs = \{\text{recv}(t) : t \in T_{DY}^g \text{ and has does not appear in } t\}.$$

Intuitively, an observation represents a message intercepted by the adversary. Define the subterm relation \sqsubseteq on T_{DY}^g as the smallest relation subject to:

$$\begin{aligned}
t &\sqsubseteq t, \\
\text{if } t &\sqsubseteq t_1 \text{ then } t \sqsubseteq \text{conc}(t_1, t_2), \\
\text{if } t &\sqsubseteq t_2 \text{ then } t \sqsubseteq \text{conc}(t_1, t_2), \\
\text{if } t &\sqsubseteq t_1 \text{ then } t \sqsubseteq \text{encr}(t_1, t_2).
\end{aligned}$$

Consider a structure $M = (S, \pi, \text{DY})$, where we record at every state all messages intercepted by the adversary at that state. Let π be an interpretation that respects the observations made at a state, and such that $\pi(e, obs)(\text{has}(t)) = \mathbf{true}$ if and only if there exists $t' \in T_{DY}^g$ such that $\text{recv}(t') \in obs$ and $t \sqsubseteq t'$. In other words, $\text{has}(t)$ holds at a state if t is a subterm of a message intercepted by the adversary. Let s_1 be a state with observations $\{\text{recv}(\text{encr}(m, k_1)), \text{recv}(\text{encr}(\text{inv}(k_1), k_2)),$ and s_2 a state with observations $\{\text{recv}(\text{encr}(m, k_1)), \text{recv}(\text{encr}(\text{inv}(k_1), k_2)), \text{recv}(\text{inv}(k_2))\}$. By definition of π , we have $(M, s_1) \models K(\text{has}(m))$ and $(M, s_2) \models K(\text{has}(m))$, so that at both states, the adversary implicitly has the message m . However, from the results of Example 2.1, we see that $(M, s_2) \models X(\text{has}(m))$, while $(M, s_1) \models \neg X(\text{has}(m))$. In other words, the adversary explicitly knows it has m at state s_2 (where it has intercepted the appropriate terms), but not at state s_1 . ■

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

Propositional Deductive Systems. The logic we introduced above is restricted to taking X over primitive propositions only. This is sufficient for many situations of interest, but it is often too limiting. Among other things, it cannot capture the fact that the agent may not have the deductive capabilities to do full propositional reasoning. For instance, an agent could explicitly know p and explicitly know q , but perhaps not that $p \wedge q$. This is what we address in this section. A *propositional signature* Σ is a signature with propositional constructors, that is, $\{\text{true}, \text{false}, \text{not}, \text{and}\} \subseteq \Sigma$, where $\text{true}, \text{false}$ have arity 0, not has arity 1, and and has arity 2. We define a logic $\mathcal{L}^{\text{PROP}}(\Sigma)$ over a propositional signature Σ where we can reason about explicit knowledge of propositional formulas. The primitive propositions are again taken to be T_{Σ}^g . The syntax is that of $\mathcal{L}^{\text{PRIM}}(\Sigma)$, except that we allow X to apply to propositional formulas. We also take *true* and *false* as primitives, rather than defining them as abbreviations.

As before, the semantics of this logic is given in terms of deductive algorithmic structures $M = (S, \pi, D)$, but here we take D to be a *propositional deductive system*, that is, a primitive deductive system defined over a propositional signature Σ . The satisfaction relation $(M, s) \models \varphi$ is defined as before (along with obvious rules for *true* and *false*), except that we modify the semantics of the X operator to take into account the fact that we can query for explicit knowledge of propositional formulas.

To do this, we first define the translation of a formula φ in our language into a propositional term φ^T in the term algebra, in the obvious way: p^T is p , $true^T$ is *true*, $false^T$ is *false*, $(\neg\varphi)^T$ is $\text{not}(\varphi^T)$, and $(\varphi \wedge \psi)^T$ is $\text{and}(\varphi^T, \psi^T)$.

There is a subtlety about this translation, and about the logic in general. We have defined \vee and \Rightarrow by abbreviation, which means that any formula containing \vee or \Rightarrow is really a formula containing \wedge and \neg . Thus, in some sense, the agent cannot explicitly distinguish between $\varphi \vee \psi$ and $\neg(\neg\varphi \wedge \neg\psi)$; they are the same formula for him. This makes strong assumptions, again, on the reasoning capabilities of the agent. One way around this problem is to use a syntax that directly uses \vee , \Rightarrow , and perhaps other connectives, rather than introducing them by abbreviations. We would then add similar constructors to the propositional signature, and extend the translation above accordingly. For simplicity, we will not do so in this paper, but it should be clear that it is a straightforward extension.

We modify the semantics of X to deal with all propositional formulas:

$$(M, s) \models X\varphi \text{ if } s = (e, \{ob_1, \dots, ob_n\}) \text{ and } ob_1, \dots, ob_n \vdash_D \varphi^T.$$

Clearly, since p^T is just p , this semantics extends that of the previous section.

Example 3.2: The following deduction rules can be added to any primitive deductive system to obtain a propositional deductive system that captures a subset of the inferences that can be performed in propositional logic.

$$\begin{array}{lll} t \triangleright \text{not}(\text{not}(t)) & \text{not}(\text{and}(t, \text{not}(t'))), t \triangleright t' & \text{and}(t, t') \triangleright t' \\ \text{not}(\text{not}(t)) \triangleright t & \text{not}(\text{and}(t, \text{not}(t'))), t' \triangleright t & t, \text{not}(t) \triangleright \text{false} \\ t \triangleright \text{not}(\text{and}(\text{not}(t), \text{not}(t'))) & t, t' \triangleright \text{and}(t, t') & \text{false} \triangleright t \\ t' \triangleright \text{not}(\text{and}(\text{not}(t), \text{not}(t'))) & \text{and}(t, t') \triangleright t & \end{array}$$

One advantage of these rules, despite the fact that they are incomplete, is that they can be used to perform very efficient (linear-time, in fact) propositional inference [McAllester 1993]. ■

By considering *full deductive systems*, that is, propositional deductive systems with unary constructors *know* and *aknow*, it is clear how to give semantics to $X\varphi$ for an arbitrary formula φ . We study such deductive systems in the full paper.

4 Axiomatizations

In this section, we present a sound and complete axiomatization for reasoning about explicit knowledge given by a deductive system. Clearly, for a particular deductive system, the properties of X depend on that deductive systems. Intuitively, we should be able to read off the properties of X from the deductive rules themselves. The remainder of this section aims at making this statement precise.

First, let us introduce an axiomatization for reasoning about deductive systems in general, independently of the actual deduction rules of the system. First, we need axioms capturing propositional reasoning in the logic.

Taut. All instances of propositional tautologies

MP. From φ and $\varphi \Rightarrow \psi$ infer ψ

Axiom **Taut** can be replaced by an axiomatization of propositional tautologies [Enderton 1972]. The following well-known axioms capture the properties of the knowledge operator [Fagin, Halpern, Moses, and Vardi 1995].

K1. $(K\varphi \wedge K(\varphi \Rightarrow \psi)) \Rightarrow K\psi$

K2. $K\varphi \Rightarrow \varphi$

K3. $K\varphi \Rightarrow KK\varphi$

K4. $\neg K\varphi \Rightarrow K\neg K\varphi$

Gen. From φ infer $K\varphi$

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 two axioms.

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

X2. $\neg X\varphi \Rightarrow K\neg X\varphi$

Finally, observations have a special status in the logic. The interpretation of observation primitives is fixed to depend on the observations made at the state. The following two axioms express the properties of observations.

X3. $ob \Rightarrow Kob$

X4. $ob \Leftrightarrow Xob$

Axiom **X3** captures the fact that indistinguishable states have the same observations. Axiom **X4** follows from the definition of deduction in Section 2: recall that for all terms t of a deductive system D , we have $t \vdash_D t$.

Let **AX** consists of the axioms **Taut**, **MP**, **K1–K4**, **Gen**, and **X1–X4**. The following result is almost immediate.

Theorem 4.1: *The axiomatization **AX** is sound and complete for $\mathcal{L}^{\text{prim}}(\Sigma)$ and $\mathcal{L}^{\text{prop}}(\Sigma)$ with respect to $\mathcal{M}(\Sigma)$.*

If we want to reason about deductive algorithmic structures equipped with a particular deductive system, we can do better. We can capture the reasoning with respect to the deductive system within our logic. The basic idea is to translate deductive rules of the deductive system into formulas of our logic. First, consider a primitive deductive system D . A deductive rule in D of the form $t_1, \dots, t_n \triangleright t$ is translated into an axiom of the form $(Xt_1 \wedge \dots \wedge Xt_n) \Rightarrow Xt$. In fact, such a translation yields an axiom schema, where we view the variables in t_1, \dots, t_n, t as schema metavariables, to be replaced

by appropriate elements of the term algebra.² Let Ax_D^{prim} be the set of axioms derived in this way for the primitive deductive system D .

Theorem 4.2: *The axiomatization $\mathbf{AX} \cup Ax_D^{\text{prim}}$ is sound and complete for $\mathcal{L}^{\text{prim}}(\Sigma)$ with respect to $\mathcal{M}_D(\Sigma)$.*

For propositional deductive systems, we need a slightly more involved translation, since we want to translate the propositional constructors that appear in the terms into logical connectives in the translated formulas. We can do this as follows. A deductive rule of the form $t_1, \dots, t_n \triangleright t$ in D is translated to a formula $(Xt_1^T \wedge \dots \wedge Xt_n^T) \Rightarrow Xt^T$. We define the formula t^T corresponding to the term t by induction on the structure of t : true^T is *true*, false^T is *false*, $(\text{not}(t))^T$ is $\neg(t^T)$, $(\text{and}(t_1, t_2))^T$ is $t_1^T \wedge t_2^T$, and t^T is t for all other terms t . This translation again yields an axiom schema of the kind we defined above. Note that we do not translate propositional constructors that appear under non-propositional constructors within a term. (Intuitively, these constructors will never arise out of the translation of formulas given in Section 3.) Let Ax_D^{prop} be the set of axioms derived in this way for the propositional deductive system D .

Theorem 4.3: *The axiomatization $\mathbf{AX} \cup Ax_D^{\text{prop}}$ is sound and complete for $\mathcal{L}^{\text{prop}}(\Sigma)$ with respect to $\mathcal{M}_D(\Sigma)$.*

5 Complexity

In this section, we study the decision problem for our logics, that is, the problem of determining, for a given formula, whether it is satisfiable. Since our logic extends the logic of knowledge where the knowledge operator is interpreted over an equivalent relation (in our case, \sim), the difficulty of the decision problem is at least as hard. It is known that the decision problem for knowledge over an equivalence relation is NP-complete [Halpern and Moses 1992]. Adding deductive algorithmic knowledge is unproblematic 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 5.1: *The decision problem for $\mathcal{L}^{\text{prim}}(\Sigma)$ and $\mathcal{L}^{\text{prop}}(\Sigma)$ over structures in $\mathcal{M}(\Sigma)$ 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 undecidable for certain deductive systems D , reasoning in our logics can be undecidable 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 the size of Γ and t), then the decision problem for our logics remains relatively easy.

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

Theorem 5.2: *For any given primitive (resp., propositional) deductive system D that is decidable in polynomial time, the decision problem for $\mathcal{L}^{\text{prim}}(\Sigma)$ (resp., $\mathcal{L}^{\text{prop}}(\Sigma)$) over structures in $\mathcal{M}_D(\Sigma)$ is NP-complete.*

In fact, there is a class of deductive systems that can be efficiently decided, and thus by Theorem 5.2 lead to a reasonable complexity for our logics. McAllester [1993] defines a deductive system D to be *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 deductive rule in D . One can show that, for any deductive system D , whether a local deduction of t from Γ exists in polynomial time in the size of Γ 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 2.1 is local, while adding the deduction rules in Example 3.2 to any local primitive deductive system yields a local deductive system.

Corollary 5.3: *For any given local primitive (resp., propositional) deductive system D , the decision problem for $\mathcal{L}^{\text{prim}}(\Sigma)$ (resp., $\mathcal{L}^{\text{prop}}(\Sigma)$) over structures in $\mathcal{M}_D(\Sigma)$ is NP-complete.*

6 Conclusion

We have described in this paper an approach to combining implicit knowledge interpreted over possible worlds with a notion of explicit knowledge based on a logical theory that allows agents to derive what they explicitly know. This additional structure, the agent’s logical theory, can be used to completely characterize the properties of the explicit knowledge operator. More specifically, it lets us derive sound and complete axiomatizations for the logic in a uniform way.

For simplicity, we have focused on the single agent case, but it should be clear that the framework generalizes to multiple agents in a straightforward way. It suffices to provide each agent with a deductive system. This lets one reason about what agents know about what other agents explicitly know. In combination with the full deductive systems mentioned at the end of Section 3, such a framework can capture a form of *simulative inference* [Kaplan and Schubert 2000], where an agent can reconstruct the reasoning of another agent in an attempt to explicitly determine what the latter explicitly knows. We study this extension in the full paper, along with an extension to dynamical systems, where observations are taken over time.

Our framework is also suitable for studying logical theories that are probabilistic. For instance, it is possible to assign a weight to every deduction rule, and to model deduction by randomly choosing which deduction rule to apply based on a distribution proportional to the weight of every applicable rule. This yields a probability distribution on the deductions, about which we can reason following ideas developed in [Halpern and Pucella 2003]. We leave this for future work.

Acknowledgments. Thanks to Hubie Chen and Joe Halpern for comments on previous drafts of this paper.

References

- Dolev, D. and A. C. Yao (1983). On the security of public key protocols. *IEEE Transactions on Information Theory* 29(2), 198–208.
- Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press.
- Fagin, R. and J. Y. Halpern (1994). Reasoning about knowledge and probability. *Journal of the ACM* 41(2), 340–367.
- Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning about Knowledge*. The MIT Press.
- Giunchiglia, F., L. Serafini, E. Giunchiglia, and M. Frixione (1993). Non-omniscient belief as context-based reasoning. In *Proceedings of IJCAI-93*, pp. 548–554.
- 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. and R. Pucella (2002). Modeling adversaries in a logic for reasoning about security protocols. In *Proceedings of FASec'02*. Available as Royal Holloway Department of Computer Science Technical Report CSD-TR-02-13. To appear in LNCS.
- Halpern, J. Y. and R. Pucella (2003). Probabilistic algorithmic knowledge. In *Theoretical Aspects of Rationality and Knowledge: Proc. Ninth Conference (TARK 2003)*, pp. 118–130.
- Hintikka, J. (1962). *Knowledge and Belief*. Ithaca, N.Y.: Cornell University Press.
- 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*. Los Altos, CA: Morgan Kaufmann.
- Levesque, H. J. (1984). A logic of implicit and explicit belief. In *Proceedings, AAAI-84*, pp. 198–202.
- McAllester, D. (1993). Automatic recognition of tractability in inference relations. *Journal of the ACM* 40(2), 284–303.