Primal Infon Logic: Derivability in Polynomial Time

Anguraj Baskar[1], Prasad Naldurg[2], K. R. Raghavendra[3], and S. P. Suresh[4]

1 **Institute of Mathematical Sciences, Chennai, India**
abaskar@imsc.res.in
2 **IBM Research India, Bangalore, India**
pnaldurg@in.ibm.com
3 **International Institute of Information Technology, Bangalore, India**
rkr@iiitb.ac.in
4 **Chennai Mathematical Institute, Chennai, India**
spsuresh@cmi.ac.in

Abstract

Primal infon logic (PIL), introduced by Gurevich and Neeman in 2009, is a logic for authorization in distributed systems. It is a variant of the $(\to, \wedge)$-fragment of intuitionistic modal logic. It presents many interesting technical challenges – one of them is to determine the complexity of the derivability problem. Previously, some restrictions of propositional PIL were proved to have a linear time algorithm, and some extensions have been proved to be PSPACE-complete. In this paper, we provide an $O(N^3)$ algorithm for derivability in propositional PIL. The solution involves an interesting interplay between the sequent calculus formulation (to prove the subformula property) and the natural deduction formulation of the logic (based on which we provide an algorithm for the derivability problem).

1998 ACM Subject Classification D.4.6 Access control, F.4.1 Proof theory

Keywords and phrases Authorization logics, Intuitionistic modal logic, Proof theory, Cut elimination, Subformula property

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.163

1 Introduction

Infon logic [10, 11, 12] is a version of modal intuitionistic logic specially designed to reason about trust and delegation in authorization systems. Its application is in the domain of access control design for distributed or federated systems, where principals who request access to resources need to present a set of assertions (or certificates) that encodes their right to access. This right may be conditioned upon attribute values, or encoded as a chain of delegations. A reasoning engine examines this query and uses the presented assertions, along with any local assertions, to derive whether the access is allowed or not according to the rules of inference in an underlying logic.

The work on infon logic is situated in the larger context of authorization languages, including SecPAL [5] and DKAL [10, 11]. These languages provide constructs to specify communication of assertions between principals, and to import or derive knowledge arising from these communications. In infon logic, the basic unit of an assertion is an *infon*, which is any information that can be communicated between two principals [12]. Infons range over relation terms, e.g., *CanRead*(*Alice*, *ncfile*), which represents a right for Alice to read *ncfile*, and form the basis of access control design.

In addition to basic terms, infon logic also allows one to express authorization (and delegation) using the modal operators said and implied. To illustrate, consider an administrator who has the right to decide access to a network configuration file *ncfile*, and can authorize a user Alice the right to read the file by giving her the following assertion: *Admin* said *CanRead*(*Alice*, *ncfile*). The statement *CanRead*(*Alice*, *ncfile*) is true if Admin is trusted i.e., $(Admin\ \mathsf{said}\ x) \to x$. The said operator is similar in function to the says operator in the SpeaksFor calculus [2, 14]. Assertions can be conditional, e.g., the administrator can decide that Alice can be granted this right if she owns the file: *Admin* said [*CanRead*(*Alice*, *ncfile*) **if** *Owns*(*Alice*, *ncfile*)]. A reasoning engine needs to check the validity of *Owns*(*Alice*, *ncfile*) to derive an answer to the query *CanRead*(*Alice*, *ncfile*). Delegation is captured by formulas like $Alice\ \mathsf{said}\ x \to Bob\ \mathsf{said}\ x$ (this would be expressed as *Alice* speaksfor *Bob* in [2]). Note that the enforcement of the authorization is decoupled from the mechanism of granting access, enabling flexible design and control.

There are many other systems for authorization whose logical cores have similarities with infon logic. For instance, the SpeaksFor calculus [14], which pioneered the logical formulation of authorization decisions, uses the says modality which is similar to our said modality. The semantics and properties of the SpeaksFor calculus were further explored by Abadi and others [1, 2]. Among other authorization logics, Binder [8], SD3 [13], Delegation Logic [15], and SecPAL [4, 5] use Datalog as basis for both syntax and semantics. DKAL is an authorization logic that extends SecPAL with constructs for specifying and reasoning about localized knowledge and targeted communication of authorization statements.

In [12], Gurevich and Neeman studied the propositional core of infon logic, explored some aspects of its proof theory and semantics, and also the complexity of the deciding validity. They also introduced a *primal* version of the logic, as an alternate system which promises to be computationally more efficient. They also provided efficient algorithms for some restrictions of propositional PIL. In this paper, we show that validity in PIL can be decided in PTIME.

The rest of the paper is structured as follows. In Section 2, we formally present primal infon logic, and explore its interesting proof-theoretical properties. In Section 3, we prove the *subformula property* for PIL, which is used in the algorithm for checking derivability in Section 4. After proving the correctness of the algorithm, we devote Section 5 to the nontrivial analysis of its running time. We end with concluding remarks in Section 6.

2 Primal infon logic

We present **primal infon logic** (PIL) formally in this section. Assume a set of atomic propositions $\mathcal{P}$. The set of formulas of primal infon logic is given by:

$$\Phi ::= p \mid x \wedge y \mid x \to y \mid \Box_a x \mid \boxplus_a x$$

where $a \in \mathcal{A}$, $p \in \mathcal{P}$, and $x, y \in \Phi$. $\Box_a x$ and $\boxplus_a x$ model a said x and a implied x from [12], respectively.

The set of **subformulas** of a formula x, denoted $\mathsf{sf}(x)$, is defined to be the smallest set S such that: $x \in S$; whenever $x \wedge y \in S$ or $x \to y \in S$, $\{x, y\} \subseteq S$; and whenever $\Box_a x \in S$ or $\boxplus_a x \in S$, $x \in S$. For a set X of formulas, $\mathsf{sf}(X) = \bigcup_{x \in X} \mathsf{sf}(x)$.

The logic is defined by the derivation system in Figure 1. In the rules, X and X' stand for sets of formulas, and we use $\Box_a X$ and $\boxplus_a X$ to denote $\{\Box_a x \mid x \in X\}$ and $\{\boxplus_a x \mid x \in X\}$, respectively. We also use $\Box_a^{-1}(X)$ and $\boxplus_a^{-1}(X)$ to denote $\{x \mid \Box_a x \in X\}$ and $\{x \mid \boxplus_a x \in X\}$, respectively. We use X, X' to denote $X \cup X'$ and X, x to denote $X \cup \{x\}$. We also use $X - x$

$\dfrac{}{X, x \vdash x}\,\mathit{ax}$	$\dfrac{X \vdash x}{X, X' \vdash x}\,\mathit{weaken}$
$\dfrac{X \vdash x \quad X \vdash y}{X \vdash x \wedge y}\,\wedge i$	$\dfrac{X \vdash x_0 \wedge x_1}{X \vdash x_i}\,\wedge e_i$
$\dfrac{X \vdash y}{X \vdash x \to y}\,\to i$	$\dfrac{X \vdash x \to y \quad X \vdash x}{X \vdash y}\,\to e$
$\dfrac{X \vdash x}{\square_a X \vdash \square_a x}\,\square_a$	$\dfrac{X, Y \vdash x}{\square_a X, \boxplus_a Y \vdash \boxplus_a x}\,\boxplus_a$
$\dfrac{X \vdash x \quad Y \vdash y}{X, Y - x \vdash y}\,\mathit{cut}$	

Figure 1 The system PIL_{nd}.

$\dfrac{}{X, x \vdash x}\,\mathit{ax}$	$\dfrac{X \vdash x}{X, X' \vdash x}\,\mathit{weaken}$
$\dfrac{X \vdash x \quad X \vdash y}{X \vdash x \wedge y}\,\wedge r$	$\dfrac{X, x_i \vdash y}{X, x_0 \wedge x_1 \vdash y}\,\wedge \ell_i$
$\dfrac{X \vdash y}{X \vdash x \to y}\,\to r$	$\dfrac{X \vdash x \quad X, y \vdash z}{X, x \to y \vdash z}\,\to \ell$
$\dfrac{X \vdash x}{\square_a X \vdash \square_a x}\,\square_a$	$\dfrac{X, Y \vdash x}{\square_a X, \boxplus_a Y \vdash \boxplus_a x}\,\boxplus_a$
$\dfrac{X \vdash x \quad Y \vdash y}{X, Y - x \vdash y}\,\mathit{cut}$	

Figure 2 The system PIL_{sc}.

to denote $X \setminus \{x\}$. In a **sequent** $X \vdash x$, X is the **antecedent** and x is the **consequent**. In a rule, the sequent occurring below the line is the **conclusion** and the sequents occurring above the line are the **premises**. The formula x occurring in the *cut* rule is called the **cut formula**. We use $X \vdash_{nd} x$ to denote that there is a derivation of the sequent $X \vdash x$ in PIL_{nd}. The **derivation problem** asks, given X and x, whether $X \vdash_{nd} x$. We also introduce the sequent calculus formulation of PIL, PIL_{sc}, in Figure 2. We use $X \vdash_{sc} x$ to denote that there is a derivation of the sequent $X \vdash x$ in PIL_{sc}.

Three features of PIL_{nd} are significant: the $\to i$ rule, the presence of the *cut* rule, and the $\boxplus_a$ rule. The $\to i$ is what distinguishes PIL from **full infon logic (FIL)**, which has the following (more standard) version of the $\to i$ rule.

$$\dfrac{X, x \vdash y}{X \vdash x \to y}\,\to i$$

The implication in FIL involves *discharging* assumptions, while the implication in PIL is just a weakening of the consequent from y to $x \to y$, without discharging any assumptions. Thus, the implication in PIL is a new kind of operator. It is worth noting that the derivability problem for just the $\{\to\}$-fragment of full infon logic is PSPACE-hard (see [16]), while even with modalities, the corresponding problem for PIL is in PTIME (Theorem 11 in this paper).

The other noteworthy feature is the presence of the *cut* rule, which is usually a feature of sequent calculus formulations. In most reasonable proof systems, though, this rule is **admissible**, i.e. whenever there are cut-free proofs of $X \vdash x$ and $Y \vdash y$, there is a cut-free proof of $X, Y - x \vdash y$. The *cut* rule is easily seen to be admissible in FIL. If π_1 and π_2 are cut-free proofs of $X \vdash x$ and $Y \vdash y$, the following is a cut-free proof of $X, Y - x \vdash y$.

$$\dfrac{\dfrac{\begin{matrix}\pi_2\\ \vdots\\ Y \vdash y\end{matrix}}{Y - x \vdash x \to y}\,\to i \qquad \begin{matrix}\pi_1\\ \vdots\\ X \vdash x\end{matrix}}{X, Y - x \vdash y}\,\to e$$

But with the weaker primal $\to i$ rule and modalities, it can be shown that *cut* is not admissible in cut-free PIL_{nd}, and hence needs to be added as an explicit rule.

Consider the sequent $\Box_a x \wedge \Box_a y \vdash \Box_a(x \wedge y)$, for instance. Here is one possible derivation in PIL_{nd} (which crucially uses the *cut* rule).

$$\dfrac{\dfrac{\dfrac{}{\Box_a x \wedge \Box_a y \vdash \Box_a x \wedge \Box_a y}\,ax}{\Box_a x \wedge \Box_a y \vdash \Box_a y}\,\wedge e_1 \qquad \dfrac{\dfrac{\dfrac{}{\Box_a x \wedge \Box_a y \vdash \Box_a x \wedge \Box_a y}\,ax}{\Box_a x \wedge \Box_a y \vdash \Box_a x}\,\wedge e_0 \qquad \dfrac{\dfrac{\dfrac{}{x, y \vdash x}\,ax \quad \dfrac{}{x, y \vdash y}\,ax}{x, y \vdash x \wedge y}\,\wedge i}{\Box_a x, \Box_a y \vdash \Box_a(x \wedge y)}\,\Box_a}{\Box_a x \wedge \Box_a y, \Box_a y \vdash \Box_a(x \wedge y)}\,cut}{\Box_a x \wedge \Box_a y \vdash \Box_a(x \wedge y)}\,cut$$

We say that a proof system $\mathcal{S}$ has the **subformula property** if the following holds:

- Whenever $X \vdash_{\mathcal{S}} x$, there is a $\mathcal{S}$-derivation π of $X \vdash x$ such that every formula y occurring in π belongs to $\mathsf{sf}(X \cup \{x\})$.

It can be easily shown that cut-free PIL_{nd} has the subformula property. (A detailed proof is given in [3].) Now suppose there is a cut-free PIL_{nd} proof of $\Box_a p \wedge \Box_a q \vdash \Box_a(p \wedge q)$, for $p, q \in \mathcal{P}$. Then there is a proof π with the same conclusion such that only formulas from $\mathsf{sf}(\{\Box_a p \wedge \Box_a q, \Box_a(p \wedge q)\})$ can occur in π. It is easy to see that the last rule of π cannot be an elimination rule. The only possibility is that the last rule is *weaken*, whose premise is $\vdash \Box_a(p \wedge q)$. This can only be got by using the $\Box_a$ rule from the premise $\vdash p \wedge q$. But this is not provable, since it is not a validity (according to the semantics given in [12]). Thus there is no cut-free PIL_{nd} proof of $\Box_a p \wedge \Box_a q \vdash \Box_a(p \wedge q)$, even though it is provable in PIL_{nd}. This means that the *cut* rule is not admissible in cut-free PIL_{nd}, and therefore that that cut cannot be eliminated in PIL_{nd}. This makes it difficult to prove the subformula property for PIL_{nd}.

But the subformula property is essential for our algorithms on PIL_{nd}. How then are we to prove it? Our solution is simple. We move to the system PIL_{sc} of Figure 2 (this system was already considered in [6]). It is reasonably straightforward to prove that cut is eliminable for this system (cut elimination also holds for the sequent calculus formulation of FIL and many extensions). It is also straightforward to show that cut-free derivations in PIL_{sc} have the subformula property. We show that one can always translate between derivations in PIL_{nd} and PIL_{sc} without introducing new formulas in the process. This yields the subformula property for PIL_{nd}. The formal details are presented in the next section.[1]

A natural question now arises – why not work with PIL_{sc} throughout, since it behaves better proof-theoretically? The answer is that it is not algorithmically well-behaved, since the left-hand sides of the sequents in a proof shrink and grow in an uncontrolled manner. On the other hand, we shall exploit precisely the controlled nature of the change in the left-hand side of the sequents in a PIL_{nd} derivation to extract an algorithm for the derivation problem. In particular, Lemma 5, which solves the non-modal fragment of PIL in linear time, uses an algorithm that closely mimics the rules in PIL_{nd}.

The third feature of interest is the $\boxplus_a$ modality and the $\boxplus_a$ rule. The intention is that $\boxplus_a$ is a weaker modality than $\Box_a$ but has the same flavour. In particular, it is conjunctive:

[1] An interesting aspect of these results is that the standard translation of a cut-free sequent-calculus proof to a normal derivation does not work in the presence of modalities. The left-rules of sequent calculus usually translate to elimination rules at the level of the hypotheses in an equivalent natural deduction derivation, but the rules for modalities are non-local – they modify *both* the hypotheses and conclusion. This is the source of the proof-theoretic complexity of these systems, and makes the cut-rule in PIL_{nd} non-eliminable.

$\boxplus_a p \wedge \boxplus_a q \vdash \boxplus_a (p \wedge q)$. This is to be contrasted with the $\diamondsuit_a$ modality from modal logic which is not conjunctive, and which has the following rule (which looks similar to the $\boxplus_a$ rule, but is very different in spirit):

$$\frac{X, y \vdash x}{\square_a X, \diamondsuit_a y \vdash \diamondsuit_a x}$$

Note that this rule insists that there be *exactly one* $\diamondsuit_a$ formula in the antecedent. Because of this difference, the algorithm in our paper does not extend to $\diamondsuit$-like modalities, but it is interesting to seek restrictions to which our techniques can apply.

It should be noted that PIL_{nd} and PIL_{sc} are not the only formulations of infon logic possible. In [12], after introducing PIL_{nd}, the authors consider a Hilbert-style proof system that helps develop efficient (linear time) algorithms for some special cases. In [7], the fragment of PIL without the $\boxplus_a$ modalities has been shown to have a linear time algorithm for the derivation problem.[2] The algorithm is based on the Hilbert-style formulation of PIL. But for unrestricted PIL (with the $\square_a$ and $\boxplus_a$ modalities), it has been shown by Gurevich and Savateev in [9] that there are sequents for which all derivations in the Hilbert-style system of [12] are exponential in size. This has the consequence that the linear-time algorithm developed in [7] does not extend to unrestricted PIL. In [6], the authors study PIL with the $\vee$ and $\perp$ operators and prove that its derivability problem is PSPACE-complete. In contrast, our paper provides an $O(N^3)$ algorithm for PIL, thus settling an important question in the study of this logic.

3 The subformula property

In this section, we formally prove the equivalence between PIL_{nd} and PIL_{sc} (preserving the set of formulas occurring in the respective proofs). We then state a cut elimination theorem for PIL_{sc}, and as corollaries, derive the subformula property for both PIL_{sc} and PIL_{nd}.

▶ **Proposition 1.** **1.** *Suppose π is a proof of $X \vdash x$ in PIL_{nd}. Then there is a proof π' of $X \vdash x$ in PIL_{sc} such that all formulas occurring in π' occur in π.*

2. *Suppose π' is a proof of $X \vdash x$ in PIL_{sc}. Then there is a proof π of $X \vdash x$ in PIL_{nd} such that all formulas occurring in π occur in π'.*

Proof. The proof is by induction on the structure of derivations, and an analysis of the last rule of π. Most of the cases are straightforward – the *ax*, *weaken*, *cut*, $\square_a$, and $\boxplus_a$ rules are present in both systems; and the $\rightarrow i$ and $\wedge i$ rules have the same form as the $\rightarrow r$ and $\wedge r$ rules, respectively. We only need to look at the other rules.

1. There are two cases to consider.

- Suppose π has the following form.

$$\dfrac{\begin{matrix}\pi_1\\ \vdots\\ X \vdash x \rightarrow y\end{matrix} \quad \begin{matrix}\pi_2\\ \vdots\\ X \vdash x\end{matrix}}{X \vdash y}\rightarrow e$$

[2] In fact, this fragment, called *basic primal infon logic*, is now used in DKAL [11] instead of unrestricted PIL. But the $\boxplus_a$ is justified in its own right (see [12]) and makes the language richer. It is also of potential interest to other authorization logics, and its derivability problem is a technical challenge. Hence the interest in unrestricted PIL.

By induction hypothesis there are PIL_{sc} derivations π_1' of $X \vdash x \to y$ and π_2' of $X \vdash x$ such that every formula occurring in π_1' occurs in π_1, and every formula occurring in π_2' occurs in π_2. π' can be taken to be the following PIL_{sc} derivation.

$$\dfrac{\begin{matrix}\pi_1'\\ \vdots\\ X \vdash x \to y\end{matrix} \qquad \dfrac{\begin{matrix}\pi_2'\\ \vdots\\ X \vdash x\end{matrix} \quad \dfrac{}{X, y \vdash y}\,ax}{X, x \to y \vdash y}\,{\to}\ell}{X \vdash y}\,cut$$

- The case when the last rule of π is $\wedge e_i$ is similarly handled, using the $\wedge \ell_i$ and *cut* rules.

2. There are two cases to consider.
- Suppose π' has the following form.

$$\dfrac{\begin{matrix}\pi_1'\\ \vdots\\ X \vdash x\end{matrix} \quad \begin{matrix}\pi_2'\\ \vdots\\ X, y \vdash z\end{matrix}}{X, x \to y \vdash z}\,{\to}\ell$$

By induction hypothesis there are PIL_{nd} derivations π_1 of $X \vdash x$ and π_2 of $X, y \vdash z$ such that every formula occurring in π_1 occurs in π_1', and every formula occurring in π_2 occurs in π_2'. π can be taken to be the following PIL_{nd} derivation.

$$\dfrac{\dfrac{\dfrac{}{X, x \to y \vdash x \to y}\,ax \quad \begin{matrix}\pi_1\\ \vdots\\ X \vdash x\end{matrix}}{X, x \to y \vdash y}\,{\to}e \qquad \begin{matrix}\pi_2\\ \vdots\\ X, y \vdash z\end{matrix}}{X, x \to y \vdash z}\,cut$$

- The case when the last rule of π is $\wedge \ell_i$ is similarly handled, using the $\wedge e_i$ and *cut* rules.

Clearly the translated proofs do not contain formulas not occurring in the original proof, in all these cases. ◀

The main reason to consider PIL_{sc} is the following important property.

▶ **Theorem 2** (Cut elimination for PIL_{sc} (Theorem 5.1 in [6])). *If $X \vdash_{sc} x$, then there is a proof π of $X \vdash x$ in PIL_{sc} such that the cut rule does not occur in π.*

▶ **Proposition 3** (Subformula property for PIL_{sc}). *Let π be a cut-free proof of $X \vdash x$ in PIL_{sc} and y be any formula that belongs to a sequent (either in the antecedent or in the consequent) occurring in π. Then $y \in \mathsf{sf}(X \cup \{x\})$.*

Proof. Observe that in every rule of PIL_{sc} other than *cut*, all formulas occurring in the premises are subformulas of the ones occurring in the conclusion. Thus any formula occurring in a cut-free PIL_{sc} derivation of $X \vdash x$ is in $\mathsf{sf}(X \cup \{x\})$. ◀

▶ **Theorem 4** (Subformula property for PIL_{nd}). *Suppose $X \vdash_{nd} x$. Then there is a proof π of $X \vdash x$ in PIL_{nd} such that any formula y occurring in π is in $\mathsf{sf}(X \cup \{x\})$.*

Proof. Since $X \vdash_{nd} x$, it follows (from Proposition 1) that $X \vdash_{sc} x$. Therefore there is a cut-free PIL_{sc} proof π' of $X \vdash x$, by Theorem 2. By the subformula property for PIL_{sc} (Proposition 3), every formula occurring in π' is from $\mathsf{sf}(X \cup \{x\})$. We use Proposition 1 again, to translate π' back to a proof π in PIL_{nd}, such that every formula occurring in π also occurs in π', and hence is in $\mathsf{sf}(X \cup \{x\})$. ◀

4 Algorithm for derivability

We present the algorithm for the derivation problem of PIL_{nd} in this section and prove its correctness. Fix a set of formulas X_0 and a formula x_0, and let Y_0 to be $\mathsf{sf}(X_0 \cup \{x_0\})$. Let $N = |Y_0|$. For any $X \subseteq Y_0$:

- $\mathit{closure}(X) = \{x \in Y_0 \mid X \vdash_{nd} x\}$.
- $\mathit{closure}'(X) = \{x \in Y_0 \mid$ there is a proof of $X \vdash x$ that does not use the $\Box$ and $\boxplus$ rules$\}$.

▶ **Lemma 5.** *For $X \subseteq Y_0$, $\mathit{closure}'(X)$ can be computed in $O(N)$ time.*

The above result is an immediate adaptation of Theorem 6.1 in [12], where a linear time algorithm for *primal constructive logic* is provided.

The algorithm for computing *closure* is presented as two mutually recursive functions $f : 2^{Y_0} \to 2^{Y_0}$ and $g : 2^{Y_0} \to 2^{Y_0}$, defined in Algorithm 1. The function g simulates *one* application of the $\Box_a$ and $\boxplus_a$ rules for each $a \in \mathcal{A}$, composed with an application of $\mathit{closure}'$. This might yield modal formulas that can be used in further $\Box_a$ and $\boxplus_a$ rules, so f makes repeated calls to g till a fixpoint is reached. f can thus be thought of as repeatedly simulating the *cut* rule after each call to g.

An application of a modal rule involves stripping the modalities from the set of formulas currently derivable, computing *closure* of the stripped set, and applying the modalities again to this set. Towards this, g makes a recursive call to f with the appropriate arguments. To make the complexity analysis easier, we keep track of the sequence of modalities stripped along each path in the recursive call tree. We call these sequences **modal contexts**, and provide them as further arguments to the functions f and g. For ease of notation, for any modal context σ, we refer to $f(\sigma, \cdot)$ and $g(\sigma, \cdot)$ as f_σ and g_σ, respectively.

Let $\Sigma = \{\Box_a, \boxplus_a \mid a \in \mathcal{A}\}$. The set of **modal contexts** of a formula x, denoted $\mathcal{C}(x)$, is a subset of Σ^*, defined by induction as follows:

- $\mathcal{C}(p) = \{\varepsilon\}$
- $\mathcal{C}(x \wedge y) = \mathcal{C}(x \to y) = \mathcal{C}(x) \cup \mathcal{C}(y)$
- $\mathcal{C}(\Box_a x) = \{\varepsilon\} \cup \{\Box_a \cdot \sigma \mid \sigma \in \mathcal{C}(x)\}$
- $\mathcal{C}(\boxplus_a x) = \{\varepsilon\} \cup \{\boxplus_a \cdot \sigma \mid \sigma \in \mathcal{C}(x)\}$.

For a set X of formulas, $\mathcal{C}(X) = \bigcup_{x \in X} \mathcal{C}(x)$. To simplify notation, we let $\mathcal{C}$ denote $\mathcal{C}(Y_0)$. Note that for any $X \subseteq Y_0$, $|\mathcal{C}(X)| \leq |\mathcal{C}| \leq |Y_0| \leq N$.

The **modal depth** of a formula x, denoted $\mathit{depth}(x)$, is defined by induction as follows:

- $\mathit{depth}(p) = 0$ for $p \in \mathcal{P}$.
- $\mathit{depth}(x \wedge y) = \mathit{depth}(x \to y) = \max(\mathit{depth}(x), \mathit{depth}(y))$.
- $\mathit{depth}(\Box_a x) = \mathit{depth}(\boxplus_a x) = \mathit{depth}(x) + 1$.

For a set X of formulas, $\mathit{depth}(X) = \max\{\mathit{depth}(x) \mid x \in X\}$.

▶ **Lemma 6.** *Suppose $X, Y \subseteq Y_0$ and $\sigma \in \mathcal{C}$. Then:*

1. *$X \subseteq \mathit{closure}'(X) \subseteq \mathit{closure}(X)$.*
2. *$\mathit{closure}'(\mathit{closure}(X)) = \mathit{closure}(\mathit{closure}(X)) = \mathit{closure}(X)$.*
3. *If $X \subseteq Y$ then $g_\sigma(X) \subseteq g_\sigma(Y)$ and $f_\sigma(X) \subseteq f_\sigma(Y)$.*
4. *$X \subseteq g_\sigma(X) \subseteq g_\sigma^2(X) \subseteq \cdots Y_0$.*
5. *$f_\sigma(X) = g_\sigma^m(X)$ for some $m \leq N$.*

Algorithm 1 Algorithm to compute *closure*

```
function $f(\sigma, X)$
    if $\sigma \notin \mathcal{C}$ or $X = \emptyset$ then
        return $\emptyset$;
    end if
    $Y \leftarrow X$;
    while $Y \neq g(\sigma, Y)$ do
        $Y \leftarrow g(\sigma, Y)$;
    end while
    return $Y$;
end function

function $g(\sigma, X)$
    for all $a \in \mathcal{A} : Y_a \leftarrow \Box_a^{-1}(X)$;
    for all $a \in \mathcal{A} : Z_a \leftarrow \Box_a^{-1}(X) \cup \boxplus_a^{-1}(X)$;
    return $\mathit{closure}'(X \cup \bigcup_{a\in\mathcal{A}} \Box_a f(\sigma\Box_a, Y_a) \cup \bigcup_{a\in\mathcal{A}} \boxplus_a f(\sigma \boxplus_a, Z_a))$;
end function
```

The last fact is true because $|Y_0| = N$ and the g_σ^i's form a nondecreasing sequence.

▶ **Proposition 7** (Soundness). *For $X \subseteq Y_0$, $\sigma \in \mathcal{C}$, and $m \geq 0$, $g_\sigma^m(X) \subseteq \mathit{closure}(X)$.*

Proof. We shall assume that $g_\tau^n(Y) \subseteq \mathit{closure}(Y)$ for all $Y \subseteq Y_0$, $\tau \in \mathcal{C}$, and all $n \geq 0$ such that $(\mathit{depth}(Y), n) <_{\text{lex}} (\mathit{depth}(X), m)$, and prove that $g_\sigma^m(X) \subseteq \mathit{closure}(X)$ for all $\sigma \in \mathcal{C}$.

For $a \in \mathcal{A}$, let $Y_a = \Box_a^{-1}(g_\sigma^{m-1}(X))$ and $Z_a = \Box_a^{-1}(g_\sigma^{m-1}(X)) \cup \boxplus_a^{-1}(g_\sigma^{m-1}(X))$. Further, let $X' = g_\sigma^{m-1}(X) \cup \bigcup_{a\in\mathcal{A}} \Box_a f_{\sigma\Box_a}(Y_a) \cup \bigcup_{a\in\mathcal{A}} \boxplus_a f_{\sigma\boxplus_a}(Z_a)$. Then $g_\sigma^m(X) = \mathit{closure}'(X')$. Note that $\mathit{depth}(Y_a) < \mathit{depth}(X)$ and $\mathit{depth}(Z_a) < \mathit{depth}(X)$. Now if $x \in X'$ we can distinguish the following three cases that can arise:

- Suppose $x \in g_\sigma^{m-1}(X)$. Since $(\mathit{depth}(X), m-1) <_{\text{lex}} (\mathit{depth}(X), m)$, by induction hypothesis, $g_\sigma^{m-1}(X) \subseteq \mathit{closure}(X)$, and hence $x \in \mathit{closure}(X)$.
- Suppose $x = \Box_a y$ for some $a \in \mathcal{A}$ and some $y \in f_{\sigma\Box_a}(Y_a)$. But $f_{\sigma\Box_a}(Y_a) = g_{\sigma\Box_a}^n(Y_a)$ for some $n \leq N$. Since $\mathit{depth}(Y_a) < \mathit{depth}(X)$, $(\mathit{depth}(Y_a), n) <_{\text{lex}} (\mathit{depth}(X), m)$, and hence by induction hypothesis, $g_{\sigma\Box_a}^n(Y_a) \subseteq \mathit{closure}(Y_a)$. Therefore $y \in \mathit{closure}(Y_a)$. Now one can use the $\Box_a$ rule and *weaken* rule to show that $\Box_a y \in \mathit{closure}(g_\sigma^{m-1}(X))$.
- Suppose $x = \boxplus_a y$ for some $a \in \mathcal{A}$ and some $y \in f_{\sigma\boxplus_a}(Z_a)$. But $f_{\sigma\boxplus_a}(Z_a) = g_{\sigma\boxplus_a}^n(Z_a)$ for some $n \leq N$. Since $\mathit{depth}(Z_a) < \mathit{depth}(X)$, $(\mathit{depth}(Z_a), n) <_{\text{lex}} (\mathit{depth}(X), m)$, and hence by induction hypothesis, $g_{\sigma\boxplus_a}^n(Z_a) \subseteq \mathit{closure}(Z_a)$. Therefore $y \in \mathit{closure}(Z_a)$. Now one can use the $\boxplus_a$ rule and *weaken* rule to show that $\boxplus_a y \in \mathit{closure}(g_\sigma^{m-1}(X))$.

Thus $X' \subseteq \mathit{closure}(g_\sigma^{m-1}(X))$. Also, $g_\sigma^{m-1}(X) \subseteq \mathit{closure}(X)$. Therefore $X' \subseteq \mathit{closure}(X)$. And since $g_\sigma^m(X) = \mathit{closure}'(X')$, $g_\sigma^m(X) \subseteq \mathit{closure}(X)$. ◀

We next prove that whenever $X \vdash x$, $x \in g_\sigma^n(X)$ for an appropriate σ and $n \leq N$. Because of our use of contexts, this direction is nontrivial. We illustrate the subtleties with an example. Let $X_0 = \{\Box_a\boxplus_b p, \boxplus_a\Box_b q\}$. One can easily see that $x_0 = \boxplus_a\boxplus_b(p \wedge q)$ is derivable from X_0. It is also easy to see that $x_0 \in f_\varepsilon(X_0)$. But $x_0 \notin f_{\boxplus_a}(X_0)$. This is because all the recursive subcalls to f return $\emptyset$, either because $\emptyset$ is passed as argument or because the context passed does not belong to $\mathcal{C}(X_0 \cup \{x_0\})$. Thus we need to ensure that the contexts supplied to recursive calls are proper. One way to ensure this is that the given context σ concatenated with any context in the argument set X belongs to $\mathcal{C}(X_0 \cup \{x_0\})$. But that condition is too strong and does not apply to the recursive call $f_{\boxplus_a}(X)$ (where $X = \{\boxplus_b p, \Box_b q\}$) even though this call will be made by $f_\varepsilon(X_0)$. A weaker condition holds, though – for every context in X, we can prepend at least one of $\Box_a$ and $\boxplus_a$ to it to get a context from X_0. We formalize this intuition below.

For two contexts $\sigma = \mathsf{M}_1 \cdots \mathsf{M}_n$ and $\sigma' = \mathsf{M}'_1 \cdots \mathsf{M}'_n$, we say that σ' is a strengthening of σ (in symbols: $\sigma' \geq \sigma$) if for all $i \leq N$, either $\mathsf{M}_i = \mathsf{M}'_i$, or $\mathsf{M}_i = \boxplus_a$ and $\mathsf{M}'_i = \Box_a$ for some $a \in \mathcal{A}$. We say that $\sigma \in \mathcal{C}$ is **safe** for $X \subseteq Y_0$ if for every $\tau \in \mathcal{C}(X)$, there is some $\sigma' \geq \sigma$ such that $\sigma'\tau \in \mathcal{C}$.

▶ **Proposition 8** (Completeness). *Suppose $X \subseteq Y_0$, $x \in closure(X)$, and $\sigma \in \mathcal{C}$. If σ is safe for $X \cup \{x\}$, then there is $m \geq 0$ such that $x \in g^m_\sigma(X)$.*

Proof. Suppose $x \in closure(X)$. Then there is a proof π of $X \vdash x$. By Theorem 4, we can assume that for every subproof π' of π with conclusion $X' \vdash x'$, and all formulas y occurring in π', $y \in \mathsf{sf}(X' \cup \{x'\})$. We now prove the desired claim by induction on the structure of π.

- Suppose the last rule of π is ax, $\wedge i$, $\rightarrow i$, $\wedge e$, or $\rightarrow e$. Without loss of generality, let the last rule have two premises and let x' and x'' be the consequents in the two premises. Suppose σ is safe for $X \cup \{x\}$. It is also safe for $X \cup \{x'\}$ and $X \cup \{x''\}$. By induction hypothesis, there exist $m, n \geq 0$ such that $x' \in g^m_\sigma(X)$ and $x'' \in g^n_\sigma(X)$. Without loss of generality, let $m \geq n$. Then $x', x'' \in g^m_\sigma(X)$, and $x \in closure'(\{x', x''\}) \subseteq g^m_\sigma(X)$.
- Suppose the last rule of π is *weaken*. Suppose the last rule has premise $X' \vdash x$, for some $X' \subseteq X$. Since σ is safe for X, it is also safe for X'. Hence by induction hypothesis, there is some m such that $x \in g^m_\sigma(X') \subseteq g^m_\sigma(X)$.
- Suppose π has the following form (and $Y = \Box_a^{-1}(X)$ and $x = \Box_a y$):

$$\dfrac{\begin{array}{c}\pi' \\ \vdots \\ Y \vdash y\end{array}}{X \vdash x}\,\Box_a$$

  Now for every $\tau \in \mathcal{C}(Y \cup \{y\})$, $\Box_a\tau \in \mathcal{C}(X \cup \{x\})$. Since σ is safe for $X \cup \{x\}$, it follows that $\sigma\Box_a$ is safe for $Y \cup \{y\}$. Thus by induction hypothesis, $y \in g^n_{\sigma\Box_a}(Y) \subseteq f_{\sigma\Box_a}(Y)$, for some $n \geq 0$. Now it is immediately seen that $x \in g_\sigma(X)$, by definition of g_σ.
- Suppose π has the following form (and $Y = \Box_a^{-1}(X)$, $Z = \boxplus_a^{-1}(X)$ and $x = \boxplus_a y$):

$$\dfrac{\begin{array}{c}\pi' \\ \vdots \\ Y, Z \vdash y\end{array}}{X \vdash x}\,\boxplus_a$$

  For every $\tau \in \mathcal{C}(Y)$, $\Box_a\tau \in \mathcal{C}(X \cup \{x\})$, and for every $\tau \in \mathcal{C}(Z \cup \{y\})$, $\boxplus_a\tau \in \mathcal{C}(X \cup \{x\})$. But σ is safe for $X \cup \{x\}$. So for every $\tau \in \mathcal{C}(Y \cup Z \cup \{y\})$, there is a strengthening σ' of σ such that either $\sigma'\Box_a\tau \in \mathcal{C}$ or $\sigma'\boxplus_a\tau \in \mathcal{C}$. In other words, for every $\tau \in \mathcal{C}(Y \cup Z \cup \{y\})$, there is a strengthening $\widehat{\sigma}$ of $\sigma\boxplus_a$ such that $\widehat{\sigma}\tau \in \mathcal{C}$. Therefore $\sigma\boxplus_a$ is safe for $Y \cup Z \cup \{y\}$. Thus by induction hypothesis, $y \in g^n_{\sigma\boxplus_a}(Y \cup Z) \subseteq f_{\sigma\boxplus_a}(Y \cup Z)$, for some $n \geq 0$. Now it is immediately seen that $x \in g_\sigma(X)$, by definition of g_σ.
- Suppose π has the following form (and $X = Y' \cup (Y'' \setminus \{y\})$):

$$\dfrac{\begin{array}{c}\pi' \\ \vdots \\ Y' \vdash y\end{array} \qquad \begin{array}{c}\pi'' \\ \vdots \\ Y'' \vdash x\end{array}}{X \vdash x}\,cut$$

Since $y \in \mathsf{sf}(X \cup \{x\})$, $\mathcal{C}(Y' \cup \{y\}) \subseteq \mathcal{C}(X \cup \{x\})$ and $\mathcal{C}(Y'' \cup \{x\}) \subseteq \mathcal{C}(X \cup \{x\})$. Thus σ is safe for both $Y' \cup \{y\}$ and $Y'' \cup \{x\}$. By induction hypothesis, there is $m \geq 0$ such that $y \in g_\sigma^m(Y') \subseteq g_\sigma^m(X)$. Therefore $Y'' \subseteq X \cup \{y\} \subseteq g_\sigma^m(X)$. Also by induction hypothesis (since π'' is a smaller proof than π), $x \in g_\sigma^n(Y'')$ for some $n \geq 0$. Therefore $x \in g_\sigma^{m+n}(X)$. ◀

▶ **Theorem 9.** *For all $X \subseteq Y_0$, $f_\varepsilon(X) = \mathit{closure}(X)$.*

Proof. On the one hand, $f_\varepsilon(X) = g_\varepsilon^N \subseteq \mathit{closure}(X)$ by soundness. Conversely, for any $x \in \mathit{closure}(X)$, ε is safe for $X \cup \{x\}$ and hence there is $m \leq N$ such that $x \in g_\varepsilon^m(X) \subseteq f_\varepsilon(X)$, by completeness. ◀

5 Complexity

Fix a set of formulas X_0 and a formula x_0 as before, and let Y_0 to be $\mathsf{sf}(X_0 \cup \{x_0\})$. Let $N = |Y_0|$. We focus on a call of $f(\varepsilon, X_0)$ and all the recursive invocations of f and g in the course of that computation. We use intuitive notions like *parent call*, *later call*, *earlier call*, which formally refer to the call tree of the computation of $f(\varepsilon, X_0)$. We use the notation $(\sigma, X) \to_f (\tau, Y)$ to denote that $f(\sigma, X)$ is an earlier recursive call and $f(\tau, Y)$ is a later recursive call in the computation of $f(\varepsilon, X_0)$. The notation $(\sigma, X) \to_g (\tau, Y)$ has a similar interpretation.

The following lemma is the first step towards analyzing the complexity of the algorithm.

▶ **Lemma 10.** *Suppose $\sigma \in \mathcal{C}$, and $X, Y \subseteq Y_0$.*
1. *If $(\sigma, X) \to_f (\sigma, Y)$ then $f_\sigma(X) \subseteq Y$.*
2. *If $(\sigma, X) \to_g (\sigma, Y)$ then $g_\sigma(X) \subseteq Y$.*

Proof. We prove both the above statements together, by induction on $|\sigma|$. There are two cases to consider.

Case $|\sigma| = 0$: In this case, $\sigma = \varepsilon$.
1. There is only one call of f with first argument ε. So the statement is vacuously true.
2. Suppose $(\varepsilon, X) \to_g (\varepsilon, Y)$. This means that $X = g_\varepsilon^i(X_0)$ and $Y = g_\varepsilon^j(X_0)$ for some i, j with $i < j$. Thus $g_\varepsilon(X) = g_\varepsilon^{i+1}(X_0) \subseteq g_\varepsilon^j(X_0) = Y$.

Case $|\sigma| > 0$: We prove the statement about f assuming the statement about g for a prefix of σ, and then prove the statement about g assuming the statement about f (for σ).
1. There are two subcases to consider.
   Case $\sigma = \tau\Box_a$: Suppose $(\sigma, X) \to_f (\sigma, Y)$. This means that there are sets X', Y' such that $X = \Box_a^{-1}(X')$, $Y = \Box_a^{-1}(Y')$, and parent calls $g(\tau, X')$ and $g(\tau, Y')$ such that $(\tau, X') \to_g (\tau, Y')$. Thus by induction hypothesis $g_\tau(X') \subseteq Y'$. But then, by definition of g, we have that $\Box_a f_\sigma(X) = \Box_a f_{\tau\Box_a}(\Box_a^{-1}(X')) \subseteq g_\tau(X') \subseteq Y'$. Therefore $f_\sigma(X) \subseteq \Box_a^{-1}(Y') = Y$.
   Case $\sigma = \tau \boxplus_a$: Suppose $(\sigma, X) \to_f (\sigma, Y)$. This means that there are sets X', Y' such that $X = \Box_a^{-1}(X') \cup \boxplus_a^{-1}(X')$, $Y = \Box_a^{-1}(Y') \cup \boxplus_a^{-1}(Y')$, and parent calls $g(\tau, X')$ and $g(\tau, Y')$ such that $(\tau, X') \to_g (\tau, Y')$. Thus $g_\tau(X') \subseteq Y'$. But then, by definition of g, $\boxplus_a f_\sigma(X) = \boxplus_a f_{\tau\boxplus_a}(\Box_a^{-1}(X') \cup \boxplus_a^{-1}(X')) \subseteq g_\tau(X') \subseteq Y'$. Therefore $f_\sigma(X) \subseteq \boxplus_a^{-1}(Y') \subseteq Y$.
2. Suppose $(\sigma, X) \to_g (\sigma, Y)$. There are two cases to consider.
   - There is one parent call $f(\sigma, Z)$ of which $g(\sigma, X)$ is a subcall and $g(\sigma, Y)$ is a later subcall. From the definition of f, it follows that there are i, j with $i < j$ such that $X = g_\sigma^i(Z)$ and $Y = g_\sigma^j(Z)$. Thus $g_\sigma(X) = g_\sigma^{i+1}(Z) \subseteq g_\sigma^j(Z) = Y$.

- There are parent calls $f(\sigma, X')$ and $f(\sigma, Y')$ such that $(\sigma, X') \rightarrow_f (\sigma, Y')$. By induction hypothesis $f_\sigma(X') \subseteq Y'$. But by definition of f it follows that there are $i > 0$ and $j > 0$ such that $X = g_\sigma^i(X')$ and $Y = g_\sigma^j(Y')$. Therefore

$$g_\sigma(X) = g_\sigma^{i+1}(X') \subseteq g_\sigma^N(X') = f_\sigma(X') \subseteq Y' \subseteq g_\sigma^j(Y') = Y.$$

◀

Algorithm 2 Improved algorithm to compute *closure*

```
Initialization: for all $\sigma \in \mathcal{C} : G_\sigma \leftarrow \emptyset$;

function $f(\sigma, X)$
    if $\sigma \notin \mathcal{C}$ or $X = \emptyset$ then
        return $\emptyset$;
    end if
    $Y \leftarrow X$;
    while $Y \neq G_\sigma$ do                      ▷ $G_\sigma = g(\sigma, G_\sigma)$ before the start of the loop.
        $G_\sigma \leftarrow Y$;
        $Y \leftarrow g(\sigma, Y)$;
    end while                                   ▷ $G_\sigma = g(\sigma, G_\sigma)$ at the end of the loop.
    return $G_\sigma$;
end function
```

▶ **Theorem 11.** *It can be checked in $O(N^3)$ time whether $x_0 \in$ closure(X_0).*

Proof. For each $\sigma \in \mathcal{C}$, if $g_\sigma(X)$ is a recursive call and if $g_\sigma(Y)$ is a later recursive call, $X \subseteq g_\sigma(X) \subseteq Y$. Thus the arguments to g_σ (in temporal order of the calls) constitutes a nondecreasing sequence of subsets of Y_0. Such a sequence can have at most N *distinct* elements. But it is possible that there are many invocations of g_σ with the same argument, which constitutes wasteful work. We thus present an improved algorithm using **memoization** in Algorithm 2. (We only redefine the function $f(\sigma, \cdot)$. The function $g(\sigma, \cdot)$ is the same as in Algorithm 1.) In this implementation, for any $\sigma \in \mathcal{C}$, the *total number of calls* to $g(\sigma, \cdot)$ is N. We achieve this by storing the last argument to g_σ in the variable G_σ, preserving this across invocations from different calls to f_σ. The code for f_σ in Algorithm 2 reveals that across different invocations of f_σ, the same argument is never passed to subcalls of g_σ. Thus there are at most N calls of g_σ. Since there is only one call of f_ε and since for every context $\sigma\mathsf{M}$, there is at most one subcall to $f_{\sigma\mathsf{M}}$ from each invocation of g_σ, the *total number of calls* of f_σ is also N, for any fixed σ.

Further, each invocation of g involves computing *closure*$'(\cdot)$, which takes $O(N)$ time, and each invocation of f involves looking up (and updating) each distinct value assumed by G_σ once. Thus overall, there are N lookups and updates of the variable G_σ, which can each be achieved in $O(N)$ time. Across all contexts, there are N^2 computations of *closure*$'(\cdot)$ and N^2 lookups and updates. Thus the overall time taken is $O(N^3)$. ◀

6 Conclusions and Future Work

We have provided an $O(N^3)$ algorithm for the derivability problem for propositional PIL. The interesting aspects of our solution are the proof of the subformula property by going over to PIL_{sc} and back, and exploiting the controlled change in the antecedents of sequents in a

PIL_{nd} proof to derive an efficient algorithm. We believe that these techniques are general, and not specific to authorization logics or PIL. We plan to adapt our techniques to many variants of intuitionistic modal logic. One plan of study is to find (proof-theoretic) variants for other operators like disjunction, with the view of deriving efficient algorithms. Another interesting possibility is to keep the rules standard but restrict the structure of formulas in $X \cup \{x\}$ in such a way that the techniques in our paper can be adapted to the problem of checking if $X \vdash x$. It is also essential to consider extensions of these systems with $\Diamond$-like modalities, as mentioned in Section 2.

References

1 M. Abadi. Logic in access control. In *Proc. 18th Annual IEEE Symposium on Logic in Computer Science*, pages 228–233, 2003.

2 M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

3 A. Baskar, Prasad Naldurg, K.R. Raghavendra, S.P. Suresh. Primal infon logic: derivability in polynomial time. Technical Report. Available at `http://www.cmi.ac.in/~spsuresh/pdffiles/pil-fsttcs2013-tr.pdf`.

4 Moritz Y. Becker. Information flow in credential systems. In *Proc. 23rd IEEE Computer Security Foundations Symposium*, CSF '10, pages 171–185, Washington, DC, USA, 2010. IEEE Computer Society.

5 Moritz Y. Becker, Cedric Fournet, and Andrew D. Gordon. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security*, 18(4):619–665, 2010.

6 Lev Beklemishev and Yuri Gurevich. Propositional primal logic with disjunction. *Journal of Logic and Computation* 22(2012),

7 Carlos Cotrini and Yuri Gurevich. Basic primal infon logic. *Journal of Logic and Computation*, Special issue devoted to Arnon Avron.

8 John DeTreville. Binder, a logic-based security language. In *Proc. 2002 IEEE Symposium on Security and Privacy*, 2002.

9 Yuri Gurevich. Two notes on propositional primal logic. Microsoft Research Technical Report MSR-TR-2011-70, May 2011.

10 Yuri Gurevich and Itay Neeman. DKAL: Distributed-knowledge authorization language. In *CSF*, pages 149–162, 2008.

11 Yuri Gurevich and Itay Neeman. DKAL2 – a simplified and improved authorization language. Microsoft Research Technical report MSR-TR-2009-11, 2009.

12 Yuri Gurevich and Itay Neeman. Logic of infons: The propositional case. *ACM Transactions of Computational Logic*, 12, January 2011.

13 Trevor Jim. Sd3: A trust management system with certified evaluation. In *IEEE Symposium on Security and Privacy*, 2001.

14 B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, 1992.

15 Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information Systems Security*, 6(1):128–171, February 2003.

16 Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9: 67–72, 1979.