Primal infon logic: derivability in polynomial time

A 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 $(\rightarrow, \wedge)$-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).

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

1 Introduction

Infon logic [9, 10, 11] 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 [4] and DKAL [9, 10]. 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 [11]. 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, 13]. Assertions can be conditional, e.g., the administrator can decide that Alice can be granted this right if she owns the file. This is modelled as $Admin\ \mathsf{said}\ [CanRead(Alice, ncfile)\ \mathsf{if}\ Owns(Alice, ncfile)]$. 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 is expressed as $Alice\ \mathsf{speaksfor}\ Bob$ 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 [13], 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 [7], SD3 [12], Delegation Logic [14], and SecPAL [3, 4] 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 [11], 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 $\mathscr{P}$. 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 $a \in \mathscr{A}$, $p \in \mathscr{P}$, and $x, y \in \Phi$. $\Box_a x$ and $\boxplus_a x$ model a **said** x and a **implied** x from [11], 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 $X - x$ 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

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

Figure 1 The system $\textsf{PIL}_{nd}$

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

Figure 2 The system $\textsf{PIL}_{sc}$

$\textsf{PIL}_{nd}$. The **derivation problem** asks, given X and x, whether $X \vdash_{nd} x$. We also introduce the sequent calculus formulation of PIL, $\textsf{PIL}_{sc}$, in Figure 2. We use $X \vdash_{sc} x$ to denote that there is a derivation of the sequent $X \vdash x$ in $\textsf{PIL}_{sc}$.

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

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

The implication in FIL involves *discharging* assumptions, while the implication in PIL is just a weakening of the consequent from y to $x \rightarrow y$, 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 $\{\rightarrow\}$-fragment of full infon logic is PSPACE-hard (see [15]), 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 $X, Y - x \vdash y$.

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

But with the weaker primal $\to i$ rule and modalities, it can be shown that *cut* is not admissible in cut-free $\mathbf{PIL}_{nd}$, 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 $\mathbf{PIL}_{nd}$ (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 $\mathscr{S}$ has the **subformula property** if the following holds:

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

It can be easily shown that cut-free $\mathbf{PIL}_{nd}$ has the subformula property. (A detailed proof is given in Appendix A.) Now suppose there is a cut-free $\mathbf{PIL}_{nd}$ proof of $\Box_a p \wedge \Box_a q \vdash \Box_a(p \wedge q)$. Then there is a proof π with the same conclusion such that only formulas from $\mathbf{sf}(\{\Box_a p \wedge \Box_a q, \Box_a(p \wedge q)\})$ 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 [11]). Thus there is no cut-free $\mathbf{PIL}_{nd}$ proof of $\Box_a p \wedge \Box_a q \vdash \Box_a(p \wedge q)$, even though it is provable in $\mathbf{PIL}_{nd}$. This means that the *cut* rule is not admissible in cut-free $\mathbf{PIL}_{nd}$, and therefore that that cut cannot be eliminated in $\mathbf{PIL}_{nd}$. This makes it difficult to prove the subformula property for $\mathbf{PIL}_{nd}$.

But the subformula property is essential for our algorithms on $\mathbf{PIL}_{nd}$. How then are we to prove it? Our solution is simple. We move to the system $\mathbf{PIL}_{sc}$ of Figure 2 (this system was already considered in [5]). 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 $\mathbf{PIL}_{sc}$ have the subformula property. We show that one can always translate between derivations in $\mathbf{PIL}_{nd}$ and $\mathbf{PIL}_{sc}$ without introducing new formulas in the process. This yields the subformula property for $\mathbf{PIL}_{nd}$. The formal details are presented in the next section.[1]

[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 $\mathbf{PIL}_{nd}$ non-eliminable.

A natural question arises now – 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 $\square_a$ but has the same flavour. It is conjunctive: $\boxplus_a p \wedge \boxplus_a q \vdash \boxplus_a(p \wedge q)$. 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 [11], after introducing PIL_{nd}, the authors consider a Hilbert-style proof system that helps develop efficient (linear time) algorithms for some special cases. In [6], 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 [8] that there are sequents for which all derivations in the Hilbert-style system of [11] are exponential in size. This has the consequence that the linear-time algorithm developed in [6] does not extend to unrestricted PIL. In [5], 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 for PIL_{nd}

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

[2] In fact, this fragment, called *basic primal infon logic*, is now used in DKAL [10] instead of unrestricted PIL. But the $\boxplus_a$ is justified in its own right (see [11]) 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.

systems; and the $\to i$ and $\wedge i$ rules have the same form as the $\to r$ 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.

$$\dfrac{\begin{array}{c}\pi_1\\ \vdots\\ X\vdash x\to y\end{array}\quad\begin{array}{c}\pi_2\\ \vdots\\ X\vdash x\end{array}}{X\vdash y}\to e$$

   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.

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

   - 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.

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

   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.

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

   - 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 [5]))**.** *If $X\vdash_{sc} x$, then there is a derivation π of $X\vdash x$ in* ***PIL**$_{sc}$* *such that the cut rule does not occur in π.*

A detailed proof is presented in Appendix B, for easy reference.

▶ **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 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 [11], where a linear time algorithm for *primal constructive logic* is provided. The proof is also presented in Appendix C, for ease of reference.

The algorithm for computing *closure* is presented in Algorithm 1 as two mutually recursive functions $f : \wp(Y_0) \to \wp(Y_0)$ and $g : \wp(Y_0) \to \wp(Y_0)$. The function g simulates *one* application of the $\Box_a$ and $\boxplus_a$ rules for each $a \in \mathscr{A}$, composed with an application of *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 $\Sigma = \{\Box_a, \boxplus_a \mid a \in \mathscr{A}\}$. The set of **modal contexts** of a formula x, denoted $\mathscr{C}(x)$, is a subset of Σ^*, defined by induction as follows:

- $\mathscr{C}(p) = \{\epsilon\}$

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

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

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

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

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

Algorithm 1 Algorithm to compute *closure*

```
function $f(\sigma, X)$
    if $\sigma \notin \mathscr{C}$ or $X = \varnothing$ then
        return $\varnothing$;
    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 $a \in \mathscr{A} : Y_a \leftarrow \Box_a^{-1}(X)$;
    for all $a \in \mathscr{A} : Z_a \leftarrow \Box_a^{-1}(X) \cup \boxplus_a^{-1}(X)$;
    return $closure'(X \cup \bigcup_{a \in \mathscr{A}} \Box_a f(\sigma \Box_a, Y_a) \cup \bigcup_{a \in \mathscr{A}} \boxplus_a f(\sigma \boxplus_a, Z_a))$;
end function
```

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

1. $X \subseteq closure'(X) \subseteq closure(X)$.
2. $closure'(closure(X)) = closure(closure(X)) = closure(X)$.
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 $m \leq N$.*

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

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

Proof. We shall assume that

$$g_\tau^n(Y) \subseteq closure(Y) \text{ for all } Y \subseteq Y_0, \tau \in \mathscr{C}, \text{ and } n \geq 0 \text{ such that } (depth(Y), n) <_{\text{lex}} (depth(X), m)$$

and prove that

$$g_\sigma^m(X) \subseteq closure(X) \text{ for all } \sigma \in \mathscr{C}.$$

For $a \in \mathscr{A}$, 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' be $g_\sigma^{m-1}(X) \cup \bigcup_{a\in\mathscr{A}} \Box_a f_{\sigma\Box_a}(Y_a) \cup \bigcup_{a\in\mathscr{A}} \boxplus_a f_{\sigma\boxplus_a}(Z_a)$. Then $g_\sigma^m(X) = \mathit{closure}'(X')$. It can be seen easily 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 $(\mathit{depth}(X), m-1) <_{\text{lex}} (\mathit{depth}(X), m)$, $g_\sigma^{m-1}(X) \subseteq \mathit{closure}(X)$ by induction hypothesis, and hence $x \in \mathit{closure}(X)$.
- Suppose $x = \Box_a y$ for some $a \in \mathscr{A}$ and some $y \in f_{\sigma\Box_a}(Y_a)$. But $\exists n \leq N : f_{\sigma\Box_a}(Y_a) = g_{\sigma\Box_a}^n(Y_a)$. 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 $a \in \mathscr{A}$ and some $y \in f_{\sigma\boxplus_a}(Z_a)$. But $\exists n \leq N : f_{\sigma\boxplus_a}(Z_a) = g_{\sigma\boxplus_a}^n(Z_a)$. 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 we have that $g_\sigma^m(X) = \mathit{closure}'(X')$, $g_\sigma^m(X) \subseteq \mathit{closure}(X)$. ◀

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 is nontrivial. We illustrate the subtleties with an example. Let X_0 be $\{\Box_a\boxplus_b p, \boxplus_a\Box_b q\}$. 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 $x_0 \in f_\epsilon(X_0)$. 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 $\mathscr{C}(X_0 \cup \{x_0\})$. 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 $\mathscr{C}(X_0 \cup \{x_0\})$. 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 $f_\epsilon(X_0)$. A weaker condition holds, though – for every context in X, we can prepend either $\Box_a$ or $\boxplus_a$ 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 $a \in \mathscr{A}$. We say that $\sigma \in \mathscr{C}$ is **safe** for $X \subseteq Y_0$ if for every $\tau \in \mathscr{C}(X)$, there is some $\sigma' \geq \sigma$ such that $\sigma' \cdot \tau \in \mathscr{C}$.

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

Proof. Suppose $x \in \mathit{closure}(X)$. 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', it holds that $y \in \mathsf{sf}(X' \cup \{x'\})$. 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 $x' \in g_\sigma^m(X)$ and $x'' \in g_\sigma^n(X)$. Without loss of generality, let $m \geq n$. Then $x', x'' \in g_\sigma^m(X)$, and $x \in \mathit{closure}'(\{x', x''\}) \subseteq g_\sigma^m(X)$.
- 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 $x \in g_\sigma^m(X') \subseteq g_\sigma^m(X)$.

- Suppose π has the following form (and $Y = \Box_a^{-1}(X)$ and $x = \Box_a y$):

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

  Now for every $\tau \in \mathscr{C}(Y \cup \{y\})$, $\Box_a \tau \in \mathscr{C}(X \cup \{x\})$. 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 $x = \boxplus_a y$):

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

  For every $\tau \in \mathscr{C}(Y)$, $\Box_a \tau \in \mathscr{C}(X \cup \{x\})$, and for every $\tau \in \mathscr{C}(Z \cup \{y\})$, $\boxplus_a \tau \in \mathscr{C}(X \cup \{x\})$. But σ is safe for $X \cup \{x\}$. So for every $\tau \in \mathscr{C}(Y \cup Z \cup \{y\})$, there is a strengthening σ' of σ such that either $\sigma'\Box_a \tau \in \mathscr{C}$ or $\sigma'\boxplus_a \tau \in \mathscr{C}$. In other words, for every $\tau \in \mathscr{C}(Y \cup Z \cup \{y\})$, there is a strengthening $\widehat{\sigma}$ of $\sigma\boxplus_a$ such that $\widehat{\sigma}\tau \in \mathscr{C}$. 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 $X = Y' \cup (Y'' \setminus \{y\})$):

$$\begin{array}{c}\begin{array}{cc}\pi' & \pi'' \\ \vdots & \vdots\end{array} \\ \dfrac{Y' \vdash y \quad Y'' \vdash x}{X \vdash x}\ \mathit{cut}\end{array}$$

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

◀

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

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

5 Complexity

Fix a set of formulas X_0 and a formula x_0 as before, and let Y_0 to be $\mathbf{sf}(X_0 \cup \{x_0\})$. Let $N = |Y_0|$. We focus on a call of $f(\epsilon, X_0)$ 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 $f(\epsilon, X_0)$. We use the notation $(\sigma, X) \rightarrow_f (\tau, Y)$ to denote that $f(\sigma, X)$ is an earlier recursive call and $f(\tau, Y)$ is a later recursive call in the computation of $f(\epsilon, X_0)$. The notation $(\sigma, X) \rightarrow_g (\tau, Y)$ has a similar interpretation.

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

▶ **Lemma 10.** *Suppose $\sigma \in \mathscr{C}$, and $X, Y \subseteq Y_0$.*
1. *If $(\sigma, X) \rightarrow_f (\sigma, Y)$ then $f_\sigma(X) \subseteq Y$.*
2. *If $(\sigma, X) \rightarrow_g (\sigma, Y)$ 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, $\sigma = \epsilon$.

1. There is only one call of f with first argument ϵ. So the statement is vacuously true.
2. Suppose $(\epsilon, X) \rightarrow_g (\epsilon, Y)$. This means that $X = g^i_\epsilon(X_0)$ and $Y = g^j_\epsilon(X_0)$ for some i, j with $i < j$. Thus $g_\epsilon(X) = g^{i+1}_\epsilon(X_0) \subseteq g^j_\epsilon(X_0) = Y$.

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 $(\sigma, X) \rightarrow_f (\sigma, Y)$. This means that there exist X', Y' such that $X = \Box_a^{-1}(X')$, $Y = \Box_a^{-1}(Y')$, and parent calls $g(\tau, X'), g(\tau, Y')$ with $(\tau, X') \rightarrow_g (\tau, Y')$. 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 $(\sigma, X) \rightarrow_f (\sigma, Y)$. This means that there exist 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 $(\tau, X') \rightarrow_g (\tau, Y')$. Thus $g_\tau(X') \subseteq Y'$. But then, by definition of g,

   $$\boxplus_a f_\sigma(X) = \boxplus_a f_{\tau\boxplus_a}(\Box_a^{-1}(X') \cup \boxplus_a^{-1}(X')) \subseteq g_\tau(X') \subseteq Y.$$

   Hence $f_\sigma(X) \subseteq \boxplus_a^{-1}(Y') \subseteq Y$.

2. Suppose $(\sigma, X) \rightarrow_g (\sigma, Y)$. 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 $X = g^i_\sigma(Z)$ and $Y = g^j_\sigma(Z)$. Thus $g_\sigma(X) = g^{i+1}_\sigma(Z) \subseteq g^j_\sigma(Z) = Y$.
   - 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 $X = g^i_\sigma(X')$ and $Y = g^j_\sigma(Y')$. Therefore

   $$g_\sigma(X) = g^{i+1}_\sigma(X') \subseteq g^N_\sigma(X') = f_\sigma(X') \subseteq Y' \subseteq g^j_\sigma(Y') = Y.$$

◀

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

Proof. For each $\sigma \in \mathscr{C}$, if $g_\sigma(X)$ is a recursive call and if $g_\sigma(Y)$ is a later recursive call, we know that $X \subseteq g_\sigma(X) \subseteq Y$. Thus the arguments to g_σ (in temporal order of the calls) constitutes a non-decreasing 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 $\sigma \in \mathscr{C}$, 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.

Algorithm 2 Improved algorithm to compute *closure*

```
Initialization: for all $\sigma \in \mathscr{C}$ : $G_\sigma \leftarrow \varnothing$;

function $f(\sigma, X)$
    if $\sigma \notin \mathscr{C}$ or $X = \varnothing$ then
        return $\varnothing$;
    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
```

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 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.

4 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.

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

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

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

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

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

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

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

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

13 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.

14 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.

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

A Subformula property for cut-free PIL_{nd}

The idea is to transform all proofs to *normal proofs* and show that normal proofs have the subformula property. To begin with, we can restrict our attention to proofs where the premise of a *weaken* rule is not a conclusion of another *weaken* rule, since we can collapse the two rules together. A proof is said to be **normal** if none of its subproofs is of the form of either of the patterns on the left in Fig 3. Starting from a derivation, we replace any subproof that fits either of the patterns on the left in Figure 3 by the corresponding proof on the right. Since the transformations reduce the size of the proof, repeatedly applying these transformations terminates in a normal proof.

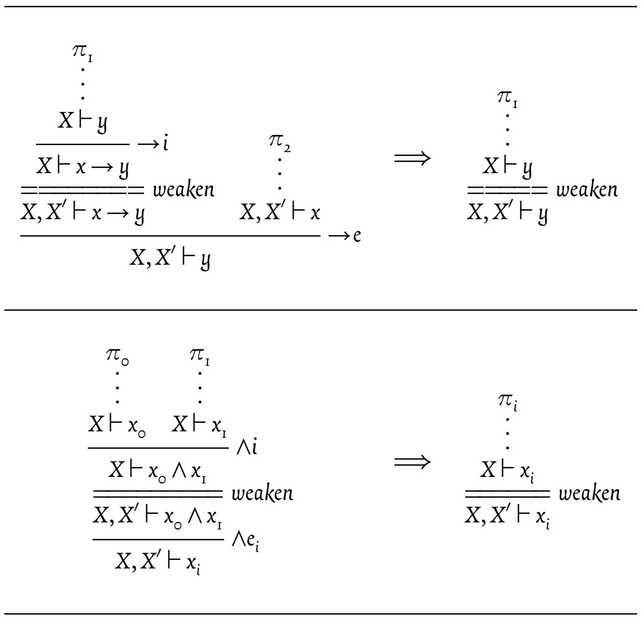

Figure 3 Transformation rules for cut-free PIL_{nd}. The double lines represent zero or one application of the *weaken* rule.

The last logical rule of a proof π is either the last rule of π (if it is not *weaken*) or the last rule of its immediate subproof (otherwise).

▶ **Theorem 12.** *Let π be a normal proof of $X \vdash x$ in cut-free PIL_{nd}. Then $y \in \mathbf{sf}(X \cup \{x\})$ for all formulas y occurring in π. Moreover, if the last logical rule of π is either ax or $\wedge e_i$ or $\to e$, $y \in \mathbf{sf}(X)$.*

Proof. We prove this by induction on the structure of proofs. We will use the fact that subproofs of normal proofs are also normal. So the induction hypothesis is always available to us. We present only some illustrative cases.

- Suppose the last rule of π is *ax*. Then π has the following form (and $x \in X$):

$$\frac{}{X \vdash x}\ ax$$

Since $x \in X$, $x \in \mathbf{sf}(X)$.

- Suppose the last rule of π is *weaken*. Then π has the following form (and $X' \subseteq X$):

$$\dfrac{\begin{array}{c}\pi' \\ \vdots \\ X' \vdash x\end{array}}{X \vdash x}\,\mathit{weaken}$$

  Any y occurring in π either occurs in π' or belongs to $X \cup \{x\}$. But if it occurs in π', by induction hypothesis, $y \in \mathsf{sf}(X' \cup \{x\}) \subseteq \mathsf{sf}(X \cup \{x\})$.
  Further, if the last rule of π' is *ax* or $\wedge e_i$ or $\to e$, by induction hypothesis we know that $y \in \mathsf{sf}(X') \subseteq \mathsf{sf}(X)$ for any y occurring in π'. Since x also occurs in π', $x \in \mathsf{sf}(X') \subseteq \mathsf{sf}(X)$ as well. Hence if the last logical rule of π is *ax* or $\wedge e_i$ or $\to e$, for any y occurring in π, $y \in \mathsf{sf}(X)$.
- Suppose the last rule of π is $\wedge i$. Then π has the following form (and $x = x_0 \wedge x_1$):

$$\dfrac{\begin{array}{c}\pi_0 \\ \vdots \\ X \vdash x_0\end{array} \quad \begin{array}{c}\pi_1 \\ \vdots \\ X \vdash x_1\end{array}}{X \vdash x}\,\wedge i$$

  Any y occurring in π either occurs in π_0 or occurs in π_1 or is the same as x. Therefore (by induction hypothesis), $y \in \mathsf{sf}(X \cup \{x_0\})$ or $y \in \mathsf{sf}(X \cup \{x_1\})$ or $y = x$. But x_0 and x_1 are in $\mathsf{sf}(x)$. Therefore, for any y occurring in π, $y \in \mathsf{sf}(X \cup \{x\})$.
- Suppose the last rule of π is $\to e$. Then π has the following form:

$$\dfrac{\begin{array}{c}\pi' \\ \vdots \\ X \vdash z \to x\end{array} \quad \begin{array}{c}\pi'' \\ \vdots \\ X \vdash z\end{array}}{X \vdash x}\,\to e$$

  Any y occurring in π either occurs in π' or occurs in π'' or is the same as x. Note that because π is a normal proof, it cannot be the case that $\to i$ is the last logical rule of π'. It cannot be one of the other introduction rules either, because of the form of the conclusion. Thus by induction hypothesis, if y occurs in π', $y \in \mathsf{sf}(X)$. In particular, $z \to x \in \mathsf{sf}(X)$. By induction hypothesis, if y occurs in π'', $y \in \mathsf{sf}(X \cup \{z\}) \subseteq \mathsf{sf}(X \cup \{z \to x\}) \subseteq \mathsf{sf}(X)$. Further $x \in \mathsf{sf}(z \to x)$ and $\mathsf{sf}(z \to x) \subseteq \mathsf{sf}(X)$. Thus for any y occurring in π, $y \in \mathsf{sf}(X)$.

◀

B Cut elimination for PIL_{sc}

We prove cut elimination for PIL_{sc} in this section. This is already proved as Theorem 5.1 in [5] in a more general setting. We prove it here for ease of reference.

Throughout this section, we use $|x|$ to denote the **size** of a formula x, i.e. the number of symbols in x, and $||\pi||$ to denote the size of a proof π, i.e. the number of nodes in π viewed as a tree.

The **rank** of a proof π is defined as follows:

- If the last rule of π is *ax*, then $\mathit{rank}(\pi) = 0$.
- If the last rule of π is *cut*, with immediate subproofs π' and π'', and if x is the cut formula, then $\mathit{rank}(\pi) = \max(|x|, \mathit{rank}(\pi'), \mathit{rank}(\pi''))$.
- If the last rule of π is neither *ax* nor *cut* and the immediate subproofs of π are $\pi_1, \ldots, \pi_i$, $\mathit{rank}(\pi) = \max(\mathit{rank}(\pi_1), \ldots, \mathit{rank}(\pi_i))$.

Clearly a cut-free proof is a proof of rank 0.

The following proposition is the heart of the weak normalization theorem.

▶ **Proposition 13.** *Let $d > 0$. Suppose that π is a proof of rank d, and all proper subproofs of π are of rank strictly smaller than d. Then there exists a proof ϖ of rank strictly smaller than d, whose conclusion is the same as that of π.*

Proof. We argue by induction on $||\pi||$. Since all proper subproofs of π are of strictly smaller rank, it means that the last rule of π is *cut*. Let the cut formula be x, with $|x| = d$. Let π have the following form:

$$\dfrac{\dfrac{\begin{matrix}\pi' \\ \vdots\end{matrix}}{X \vdash x}\,\mathbf{r}' \qquad \dfrac{\begin{matrix}\pi'' \\ \vdots\end{matrix}}{Y \vdash y}\,\mathbf{r}''}{X, Y - x \vdash y}\,cut$$

We have the following cases to consider:

1. $x \notin Y$. In this case we get the desired proof ϖ by using the *weaken* rule on π''. Clearly $rank(\varpi) < d$. In all the succeeding cases, we will assume that $x \in Y$.
2. $\mathbf{r}'$ introduces x on the right, $\mathbf{r}''$ introduces x on the left, and x does not occur on the left in the premises of $\mathbf{r}''$. The following combinations can arise.
   - $\mathbf{r}'$ is $\wedge r$ and $\mathbf{r}''$ is $\wedge\ell_i$. In this case π has the following form ($x = x_0 \wedge x_1$ and $Y' = Y - x$):

   $$\dfrac{\dfrac{\begin{matrix}\pi'_0 \\ \vdots \\ X \vdash x_0\end{matrix} \quad \begin{matrix}\pi'_1 \\ \vdots \\ X \vdash x_1\end{matrix}}{X \vdash x_0 \wedge x_1}\,\wedge r \qquad \dfrac{\begin{matrix}\pi''_i \\ \vdots \\ Y', x_i \vdash y\end{matrix}}{Y', x_0 \wedge x_1 \vdash y}\,\wedge\ell_i}{X, Y' \vdash y}\,cut$$

   The desired ϖ is the following proof:

   $$\dfrac{\begin{matrix}\pi'_i \\ \vdots \\ X \vdash x_i\end{matrix} \quad \begin{matrix}\pi''_i \\ \vdots \\ Y', x_i \vdash y\end{matrix}}{X, Y' \vdash y}\,cut$$

   Clearly $rank(\varpi) < d$, since the size of the new cut formula x_i is smaller than d.
   - $\mathbf{r}'$ is $\rightarrow r$ and $\mathbf{r}''$ is $\rightarrow \ell$. This is handled similar to the above subcase.
   - $\mathbf{r}'$ is $\Box_a$ and $\mathbf{r}''$ is $\Box_a$. This is handled similar to the next subcase.
   - $\mathbf{r}'$ is $\Box_a$ and $\mathbf{r}''$ is $\boxplus_a$. In this case π has the following form, with $x = \Box_a x'$, $X = \Box_a X'$, $Y = \Box_a Y'_0, \boxplus_a Y'_1$, and $y = \boxplus_a y'$):

   $$\dfrac{\dfrac{\begin{matrix}\pi'_1 \\ \vdots \\ X' \vdash x'\end{matrix}}{X \vdash x}\,\Box_a \qquad \dfrac{\begin{matrix}\pi''_1 \\ \vdots \\ Y'_0, Y'_1 \vdash y'\end{matrix}}{Y \vdash y}\,\boxplus_a}{X, Y - x \vdash y}\,cut$$

The desired ϖ is the following proof:

$$\dfrac{\dfrac{\begin{array}{c}\pi'_1\\ \vdots\\ X'\vdash x'\end{array}\quad\begin{array}{c}\pi''_1\\ \vdots\\ Y'_0,Y'_1\vdash y'\end{array}}{X',Y'_0-x',Y'_1\vdash y'}\,cut}{X,Y-x\vdash y}\,\boxplus_a$$

Clearly $rank(\varpi)<d$, since the size of the new cut formula x' is smaller than d.

- r′ is $\boxplus_a$ and r″ is $\boxplus_a$. This is handled similar to the above subcase.

3. r′ introduces x on the right, r″ introduces x on the left, and x occurs on the left in the premises of r″. There are only two subcases:
   - r′ is $\wedge r$ and r″ is $\wedge\ell_i$. In this case π has the following form (and $x=x_0\wedge x_1$):

$$\dfrac{\dfrac{\begin{array}{c}\pi'_0\\ \vdots\\ X\vdash x_0\end{array}\quad\begin{array}{c}\pi'_1\\ \vdots\\ X\vdash x_1\end{array}}{X\vdash x_0\wedge x_1}\,\wedge r\qquad\dfrac{\begin{array}{c}\pi''_i\\ \vdots\\ Y,x_i\vdash y\end{array}}{Y\vdash y}\,\wedge\ell_i}{X,Y-x\vdash y}\,cut$$

   Consider the following proof $\widehat{\pi}$:

$$\dfrac{\dfrac{\begin{array}{c}\pi'_0\\ \vdots\\ X\vdash x_0\end{array}\quad\begin{array}{c}\pi'_1\\ \vdots\\ X\vdash x_1\end{array}}{X\vdash x_0\wedge x_1}\,\wedge r\qquad\begin{array}{c}\pi''_i\\ \vdots\\ Y,x_i\vdash y\end{array}}{X,Y-x,x_i\vdash y}\,cut$$

   Now $||\widehat{\pi}||=||\pi'||+||\pi''_i||+1<||\pi'||+||\pi''||+1=||\pi||$. Further, $rank(\widehat{\pi})=d$ and all proper subproofs have rank strictly smaller than d. Thus by induction hypothesis, there is a proof $\widehat{\varpi}$ of rank smaller than d with the same conclusion as $\widehat{\pi}$. Now the desired ϖ is the following proof:

$$\dfrac{\begin{array}{c}\pi'_i\\ \vdots\\ X\vdash x_i\end{array}\quad\begin{array}{c}\widehat{\varpi}\\ \vdots\\ X,Y-x,x_i\vdash y\end{array}}{X,Y-x\vdash y}\,cut$$

   Clearly $rank(\varpi)<d$, since the size of the new cut formula x_i is smaller than d.
   - r′ is $\rightarrow r$ and r″ is $\rightarrow\ell$. This is handled similar to the above subcase.

4. r′ introduces x on the right, and r″ is an application of *ax*. If $y=x$, we obtain our desired proof by applying *weaken* on the conclusion of π'. If $y\neq x$, we obtain our desired proof by applying *weaken* on the conclusion of π''.

5. r′ introduces x on the right, and r″ is not *ax* and does not introduce x on the left. There are many subcases here but they are similarly handled. We look at just one subcase. Suppose π has the

following form (and $Y = Y_0, Y_1 - z$):

$$\dfrac{\dfrac{\begin{array}{c}\pi' \\ \vdots\end{array}}{X \vdash x}\,\mathbf{r}' \qquad \dfrac{\begin{array}{c}\pi''_0 \\ \vdots \\ Y_0 \vdash z\end{array} \quad \begin{array}{c}\pi''_1 \\ \vdots \\ Y_1 \vdash y\end{array}}{Y_0, Y_1 - z \vdash y}\,\textit{cut}}{X, Y_0 - x, Y_1 - \{z, x\} \vdash y}\,\textit{cut}$$

Clearly $|z| < d$ since it is a cut formula appearing in π'', whose rank is strictly smaller than d. Now consider the following proof π^0:

$$\dfrac{\dfrac{\begin{array}{c}\pi' \\ \vdots\end{array}}{X \vdash x}\,\mathbf{r}' \qquad \begin{array}{c}\pi''_0 \\ \vdots \\ Y_0 \vdash z\end{array}}{X, Y_0 - x \vdash z}\,\textit{cut}$$

and the following proof π^1:

$$\dfrac{\dfrac{\begin{array}{c}\pi' \\ \vdots\end{array}}{X \vdash x}\,\mathbf{r}' \qquad \begin{array}{c}\pi''_1 \\ \vdots \\ Y_1 \vdash y\end{array}}{X, Y_1 - x \vdash x}\,\textit{cut}$$

Clearly since $||\pi^0|| < ||\pi||$ and $||\pi^1|| < ||\pi||$, by induction hypothesis, there are proofs ϖ^0 and ϖ^1 of rank strictly smaller than d, with the same conclusions as π^0 and π^1, respectively. Now the desired proof ϖ is as follows:

$$\dfrac{\begin{array}{c}\varpi^0 \\ \vdots \\ X, Y_0 - x \vdash z\end{array} \quad \begin{array}{c}\varpi^1 \\ \vdots \\ X, Y_1 - x \vdash y\end{array}}{X, Y_0 - x, Y_1 - \{x, z\} \vdash x}\,\textit{cut}$$

Since $\textit{rank}(\varpi^i) < d$ and since $|z| < d$, $\textit{rank}(\varpi) < d$, as desired.

6. $\mathbf{r}'$ is an axiom. This is handled similar to item 4 above.
7. $\mathbf{r}'$ is not an axiom and does not introduce x on the right. This is handled similar to item 5 above.

◀

▶ **Theorem 14** **(Weak normalization).** *For all X, x, if there is a proof of $X \vdash x$, then there is a cut-free proof of $X \vdash x$.*

Proof. For every proof π, define $\mu(\pi)$ to be the pair (d, n) where $d = \textit{rank}(\pi)$, and n is the number of subproofs of π of rank d. If $\textit{rank}(\pi) = 0$, π is already normal. If not, let $\textit{rank}(\pi) = d > 0$ and let ϖ be a subproof of π such that $\textit{rank}(\varpi) = d$ and no proper subproof of ϖ is of rank $\geq d$. By Proposition 13, there is another proof ϖ' with $\textit{rank}(\varpi') < d$ and whose conclusion is the same as that of ϖ. Replace ϖ by ϖ' in π to get the proof π'. Clearly $\mu(\pi') < \mu(\pi)$. Since lexicographic ordering on pairs of natural numbers is a well order, by repeating the above process we eventually reach a proof of rank 0 – a cut-free proof, in other words.

◀

C Linear time algorithm for computing *closure*′

Fix a set of formulas Y_0 such that $\mathsf{sf}(Y_0) \subseteq Y_0$. Let $|Y_0|$ be N. Recall that $closure'(X)$ is the set of all $x \in Y_0$ that can be derived from X without using the $\Box$ rule or the $\boxplus$ rule. We describe below how to compute, for any $X \subseteq Y_0$, the set $closure'(X)$, in $O(N)$ time. The algorithm is basically the one described in Theorem 6.1 of [11].

For a formula x, define the notions $left(x)$, $right(x)$, and $op(x)$ as follows:

- If $x \in \mathscr{P}$, $left(x)$, $right(x)$ and $op(x)$ are all undefined.
- If $x = y \wedge z$, $left(x) = y$, $right(x) = z$, and $op(x) = \wedge$.
- If $x = y \to z$, $left(x) = y$, $right(x) = z$, and $op(x) = \to$.
- If $x = \Box_a y$, $left(x) = y$, $right(x)$ is undefined, and $op(x) = \Box_a$.
- If $x = \boxplus_a y$, $left(x) = y$, $right(x)$ is undefined, and $op(x) = \boxplus_a$.

For every $x \in Y_0$, define the following four sets:

- $\mathsf{A}_\ell(x) = \{y \in Y_0 \mid op(y) = \wedge \text{ and } left(y) = x\}$.
- $\mathsf{A}_r(x) = \{y \in Y_0 \mid op(y) = \wedge \text{ and } right(y) = x\}$.
- $\mathsf{I}_\ell(x) = \{y \in Y_0 \mid op(y) = \to \text{ and } left(y) = x\}$.
- $\mathsf{I}_r(x) = \{y \in Y_0 \mid op(y) = \to \text{ and } right(y) = x\}$.

The algorithm to compute $closure'(X)$ is given in Algorithm 3. For each $x \in Y_0$, we keep track of its status, which we denote $status(x)$. It takes one of the values *raw*, *pending*, or *processed*. We also use a queue Q of formulas, with the corresponding *enqueue* and *dequeue* functions. It is easy to argue that whenever $status(x)$ becomes *pending*, it is the case that $x \in closure'(X)$. It is also the case that all *pending* formulas become *processed* once the algorithm terminates (if it does). Further, one can argue by induction on the size of proofs that for every formula $x \in closure'(X)$, eventually $status(x)$ becomes *pending*.

One can argue that the algorithm terminates in $O(N)$ time as follows. Each element enters Q at most once (when it is *raw*, and it becomes *pending* just before entering Q). We process each element of Q exactly once and mark it *processed* and dequeue it from the queue. For each element u of the queue, we spending constant time setting the status of $left(u)$ and $right(u)$ and perhaps enqueuing them. The sets $\mathsf{A}_\ell(u)$, $\mathsf{A}_r(u)$, $\mathsf{I}_\ell(u)$, and $\mathsf{I}_r(u)$ have no bound on their size, but for distinct u and u', the sets $\mathsf{A}_\ell(u)$ and $\mathsf{A}_\ell(u')$ are disjoint, and similarly for the sets A_r, I_ℓ, and I_r. So across all elements in Y_0, the total time consumed in each of the **for all** blocks inside the **while** loop is $O(N)$. This gives us a linear time algorithm to compute $closure'(X)$ for any given $X \subseteq Y_0$.

Algorithm 3 Linear time algorithm for $closure'(X)$

```
$Q \leftarrow \emptyset$;
for all $x \in X : status(x) \leftarrow pending$; $enqueue(Q, x)$;
for all $x \in Y_0 \setminus X : status(x) \leftarrow raw$;
while $Q \neq \emptyset$ do
    $u \leftarrow dequeue(Q)$;
    if $op(u) = \wedge$ and $status(left(u)) = raw$ then          ▷ $u$ is the premise of the $\wedge e_0$ rule.
        $status(left(u)) \leftarrow pending$;    $enqueue(Q, left(u))$;
    end if
    if $op(u) = \wedge$ and $status(right(u)) = raw$ then         ▷ $u$ is the premise of the $\wedge e_1$ rule.
        $status(right(u)) \leftarrow pending$;    $enqueue(Q, right(u))$;
    end if
    if $op(u) = \to$ and $status(left(u)) \neq raw$ and $status(right(u)) = raw$ then
        $status(right(u)) \leftarrow pending$;                ▷ $u$ is the major premise of the $\to e$ rule.
        $enqueue(Q, right(u))$;
    end if
    for all $v \in \mathsf{A}_\ell(u)$ such that $status(right(v)) \neq raw$ and $status(v) = raw$ do
        $status(v) \leftarrow pending$;                        ▷ $u$ is the left premise of the $\wedge i$ rule.
        $enqueue(Q, v)$;
    end for
    for all $v \in \mathsf{A}_r(u)$ such that $status(left(v)) \neq raw$ and $status(v) = raw$ do
        $status(v) \leftarrow pending$;                       ▷ $u$ is the right premise of the $\wedge i$ rule.
        $enqueue(Q, v)$;
    end for
    for all $v \in \mathsf{I}_\ell(u)$ such that $status(v) \neq raw$ and $status(right(v)) = raw$ do
        $status(right(v)) \leftarrow pending$;                ▷ $u$ is the minor premise of the $\to e$ rule.
        $enqueue(Q, right(v))$;
    end for
    for all $v \in \mathsf{I}_r(u)$ such that $status(v) = raw$ do          ▷ $u$ is the premise of the $\to i$ rule.
        $status(v) \leftarrow pending$;
        $enqueue(Q, v)$;
    end for
    $status(u) \leftarrow processed$;
end while
$closure'(X) = \{u \in Y_0 \mid status(u) = processed\}$;
```