Logic of Infons: the Propositional Case

YURI GUREVICH
Microsoft Research
and
ITAY NEEMAN
UCLA

Infons are statements viewed as containers of information (rather then representations of truth values). The logic of infons turns out to be a conservative extension of logic known as constructive or intuitionistic. Distributed Knowledge Authorization Language uses additional unary connectives "p said" and "p implied" where p ranges over principals. Here we investigate infon logic and a narrow but useful *primal* fragment of it. In both cases, we develop model theory and analyze the derivability problem: Does the given query follow from the given hypotheses? Our more involved technical results are on primal infon logic. We construct an algorithm for the multiple derivability problem: Which of the given queries follow from the given hypotheses? Given a bound on the quotation depth of the hypotheses, the algorithm runs in linear time. We quickly discuss the significance of this result for access control.

Categories and Subject Descriptors: F.2.2. [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Complexity of proof procedures*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Computational logic, Modal logic, Model theory, Proof theory*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Logic, complexity, access control, derivation problem, infon, infon logic, intuitionistic logic, linear time

ACM Reference Format:

Gurevich, Y. and Neeman, I. 2011. Logic of infons: the propositional case. ACM Trans. Comput. Logic 12, 2, Article 9 (January 2011), 28 pages.
DOI = 10.1145/1877714.1877715 http://doi.acm.org/10.1145/1877714.1877715

1. INTRODUCTION

The notion of infon is basic in DKAL, Distributed Knowledge Authorization Language, that we have been developing [Gurevich and Neeman 2008, 2009; Gurevich and Roy 2009]. Infons are statements, that is declarative statements,

Authors' addresses: Y. Gurevich, Microsoft Research, Redmond, WA 98052; email: gurevich@microsoft.com; I. Neeman, Department of Mathematics, UCLA, Los Angeles, CA 90095.

DOI 10.1145/1877714.1877715 http://doi.acm.org/10.1145/1877714.1877715

viewed as containers of information, for example, John has the right to read File 13 of the given computer system.

> Quisani[1]: How is this different from the usual treatment of statements in mathematical logic?
>
> Authors[2]: We don't ask whether an infon is true or false. Instead we ask whether it is known to relevant principals. For example, does the owner of File 13 know that John has the right to read the file? Does the administrator of the system know? Does John know? Besides, the notion of infon is purely semantical.
>
> Q: Did you invent the term?
>
> A: No, though for a while we thought that we did. The term has been used in situation theory where the name is "intended to emphasize the fact that 'infons' are semantic objects, not syntactic representations" [Devlin 1991, 22]. But infons are assigned truth values in situation theory, so our usage of the term is quite different.

One may study the algebra of infons. Order $a \leq b$ if the information in a is a part of that in b. At the bottom of that partial order are uninformative infons carrying no information whatsoever. There is a natural union operation $a + b$ on infons. You know $a + b$ if you know a and you know b. Infon $a + b$ is the least upper bound of a and b. But one has to be careful with the information order because of the omniscience problem well known in epistemic logic. The partial order is intractable. Indeed, valid logic statements are uninformative. In Gurevich and Neeman [2008] we used a rule-based feasible approximation of the true information order.

The right tool to deal with infons is logic. The addition operation $a + b$ gives rise to conjunction $a \wedge b$. And the implication connective is natural. If you know a and you know $a \to b$ then you know b. In terms of the information order, $a \to b$ is the least solution x of the inequality $a + x \geq b$.

> Q: Give me an example of implication.
>
> A: You have the right to fish in this river if you have an appropriate licence from the state.

The use of implication allowed us to simplify DKAL [Gurevich and Neeman 2009]. For example "principal p is trusted on saying infon x," used to be a primitive notion in DKAL, subject to a stipulation "$x \leq (p$ said $x) + (p$ trusted on saying $x)$." Now we define "p trusted on saying x" as "$(p$ said $x) \to x$." The stipulation follows. It turns out that infon logic is a conservative extension of the appropriate fragment of intuitionistic logic. In addition to conjunction and implication, infon logic has two unary connectives "p said" and "p implied" for any principal p. The number of principals is unbounded.

[1]Quisani is an inquisitive friend of ours.
[2]Speaking one at a time.

Q: How is infon logic related to authorization?

A: In DKAL, a principal uses infon logic to derive consequences from, (1) his own knowledge assertions, and (2) the communications from other principals.

Q: How are "said" and "implied" different?

A: A principal p may just say x to q. Then (if the communication reaches q and q accepts it but let us ignore all that) q learns infon p `said` x. However p may condition his saying x on q knowing y. Then q learns infon $y \to (p$ `implied` $x)$.

Q: Give me an example.

A: Suppose that Alice and Bob work in the same company, and Charlie is an outside visitor. If Alice tells Bob that Charlie can read the latest company letter then Bob learns this: Alice `said` Charlie can read the letter. But if Alice tells Bob that Charlie can read the letter provided that Charlie signed the nondisclosure agreement (NDA) then Bob learns this: if Charlie signed the NDA then Alice `implied` that Charlie can read the letter.

Q: What does this buy you? Suppose that Bob knows the proviso that Charlie signed the NDA. Then he learns that Alice implied that Charlie can read the letter. Shouldn't he learn that Alice said that Charlie can read the letter?

A: If one is not careful, a proviso can be used for undesired delegation and even a probing attack [Gurevich and Neeman 2009, §7]. To get from knowing p `said` x to knowing x, q needs to trust p on saying x. To get from knowing p `implied` x to knowing x, q needs to trust p on implying x which is a stronger condition.

The derivability problem (whether formulas Γ entail a formula x) for intuitionistic logic may seem to be intractable. Even the termination of the proof search does not seem to be guaranteed. But there are intuitionistic calculi with the subformula property: if Γ entails x then there is a derivation of x from Γ that uses only subformulas of formulas $\Gamma \cup \{x\}$. This helps to establish that the problem is solvable in polynomial space. The problem is in fact polynomial space complete in the worst case [Statman 1979].

Starting with a known natural deduction calculus for intuitionistic logic, which is sound and complete with respect to the standard Kripke semantics and which has the subformula property, we extend the calculus and semantics to infon logic and prove that the extended calculus is sound and complete and has the subformula property. The derivability problem remains polynomial space complete in the worst case. This does not make infon logic useless. It works in a great many practical cases. Typical cases are far from the worst ones.

Further, we identify a narrow *primal* fragment of infon logic that is surprisingly expressive. For the sake of contrast, the original infon logic may be called full. We modify the natural deduction calculus for full infon logic to fit primal infon logic. The new calculus has a version of subformula property; instead of subformulas we speak of relatives of theirs, which we call local (to the given hypotheses and query) formulas. The new calculus is sound and complete with respect to Kripke semantics adjusted in a simple and natural way. The definition of when an implication $x \to y$ holds in a world w becomes nondeterministic, and that's it.

Q: What about primal intuitionistic logic? I mean the intersection of primal infon logic with intuitionistic logic. It's got to be known. Intuitionistic logic has been researched for such a long time.

A: Well, the only references of relevance that we know are proof-theoretic papers [Avron 2010] and [Avron and Lahav 2009]. Their "semi-implication" is our primal implication. But even the two papers have not been published yet; we learned of them when we presented infon logic at Tel Aviv University. Primal intuitionistic logic is weak indeed. That may explain the lack of attention.

The more involved technical results of this article are related to the derivability problem for primal infon logic.

Definition 1.1. The *multiple derivability problem* MD(L) for a logic L is to compute, given formulas $x_1, \dots, x_m$ (the hypotheses) and $y_1, \dots, y_n$ (the queries), which of the queries are derivable from the hypotheses.

We construct an algorithm that solves the multiple derivability problem for primal infon logic. We stratify primal logic according to the quotation depth of infons. For example, both infons "Alice said that Bob implied that John has the right to read File 13" and "Bob said that John has the right to read File 13" belong to stratum 2 but only the second belongs to stratum 1. At each stratum, our algorithm works in linear time. (Our computation model is the usual random access machine, as in Cormen et al. [1990].)

In applications, quotation depth tends to be small. Most authorization languages do not even allow nesting the pronouncements of principals. This applies to all Datalog-based authorization languages that we know.

Q: The lowest stratum where pronouncements are nested is stratum 2. Is stratum 2 of primal logic of any use for authorization purposes?

A: Very much so. In our experience, stratum 2 suffices for most purposes.

Q: Is there any evidence beyond your own experience?

A: An authorization language SecPAL [Becker et al. 2007] expresses many important access control scenarios. A rather natural translation of SecPAL to DKAL uses only stratum 2 of primal infon logic.

This article is self-contained. We do not presume that the reader is familiar with DKAL or intuitionistic logic.

2. NATURAL DEDUCTION CALCULI FOR FULL AND PRIMAL INFON LOGIC

In DKAL, infons are expressed by means of terms, and formulas have the form "principal knows infon". Primitive terms have the form $t_0\ A(t_1, \dots, t_k)$, where A is an attribute name, each t_i is a term in the sense of first-order logic, and t_0 is of type Principal. Compound terms are built from primitive ones by means of functions $\wedge$ and $\to$ of type Infon $\times$ Infon $\longrightarrow$ Infon and functions `said` and `implied` of type Principal $\times$ Infon $\longrightarrow$ Infon.

This article is devoted to infon logic. (Readers interested primarily in DKAL and not so much in infon logic per se may want to go directly to [Gurevich and Neeman 2009].) Here we treat infon terms as propositional formulas. We use symbol $\top$ to represent an uninformative infon. We presume an infinite vocabulary of infon variables and another infinite vocabulary of (the names of) principals. Formulas are built from infon variables and the infon constant $\top$ by the following means.

—*Conjunction*. If x, y are formulas then so is $x \wedge y$.
—*Implication*. If x, y are formulas then so is $x \to y$.
—*Two unary connectives* p `said` *and* p `implied` *for every principal* p. If x is a formula then so are p `said` x and p `implied` x.

Formulas that do not involve the unary connectives will be called quotation-free.

Some of our algorithms take formulas or sequences of formulas as inputs. In this connection, we presume that the syntax of formulas is such that an occurrence of a subformula y in a formula x is uniquely defined by the position of the first symbol of y in x.

2.1 Natural Deduction Calculus for Full Infon Logic

Our natural deduction calculus, NDF, for full infon logic is essentially an extension of the disjunction-free version of the intuitionistic propositional system, NJp [Mints 2000, Section 2.2]. It works with sequents of the form $\Gamma \vdash x$, where x is a formula and Γ is a set of formulas written as a sequence.[3] Here are the axioms and rules of inference.

Axioms.

(⊤) $\vdash \top.$
(x2x) $x \vdash x.$

Inference rules.

(Premise Inflation) $$\frac{\Gamma \vdash y}{\Gamma, x \vdash y};$$

[3]Even though NDF works with sequents, we refrain from calling it a sequent calculus. The term sequent calculus has a different meaning in modern proof theory.

$$(\wedge\mathtt{E})\qquad \frac{\Gamma \vdash x \wedge y}{\Gamma \vdash x} \qquad \frac{\Gamma \vdash x \wedge y}{\Gamma \vdash y};$$

$$(\wedge\mathtt{I})\qquad \frac{\Gamma \vdash x \quad \Gamma \vdash y}{\Gamma \vdash x \wedge y};$$

$$(\to\mathtt{E})\qquad \frac{\Gamma \vdash x \quad \Gamma \vdash x \to y}{\Gamma \vdash y};$$

$$(\to\mathtt{I})\qquad \frac{\Gamma, x \vdash y}{\Gamma \vdash x \to y};$$

$$(\mathtt{Said})\qquad \frac{\Delta \vdash y}{q \ \mathtt{said}\ \Delta \ \vdash \ q \ \mathtt{said}\ y};$$

$$(\mathtt{Implied})\qquad \frac{(\Delta_1 \cup \Delta_2) \vdash y}{(q \ \mathtt{said}\ \Delta_1) \cup (q \ \mathtt{implied}\ \Delta_2) \ \vdash \ q \ \mathtt{implied}\ y}.$$

Here E and I allude to "elimination" and "introduction" respectively. If $\Delta = \{x_1, \ldots, x_n\}$ then "q said Δ" is the set $\{(q\ \mathtt{said}\ x_i) : i = 1, \ldots, n\}$ and "q implied Δ" is the set $\{(q\ \mathtt{implied}\ x_i) : i = 1, \ldots, n\}$.

Corollary 2.1. $q\ \mathtt{said}\ x \ \vdash \ q\ \mathtt{implied}\ x$.

Proof. Apply rule (Implied) to $x \vdash x$. □

Corollary 2.2 (Transitivity). *If* $\Gamma \vdash x$ *and* $\Gamma, x \vdash y$ *then* $\Gamma \vdash y$.

Proof. By (→I), we have $\Gamma \vdash (x \to y)$. It remains to apply (→E). □

Q: I guess $\top$ is the propositional constant usually called "true."

A: This is a reasonable point of view as far as this article is concerned.

Q: Is there any semantical difference between $\top$ and "true"?

A: Yes, $\top$ is an infon known to all principals. It is natural to assume that it is true but we don't have to lump all true infons together. Some true infons may be informative.

Q: Rule (Implied) is too complicated. Turn Corollary 2.1 into an axiom and then replace (Implied) with a simpler rule $\dfrac{\Delta \ \vdash y}{q\ \mathtt{implied}\ \Delta \ \vdash q\ \mathtt{implied}\ y}$.

A: Calculus NDF has the subformula property. If a sequent $\Gamma \vdash x$ is provable then it has a proof that uses only subformulas of the sequent formulas; see Theorem 3.2. The modified system loses the subformula property. For example any proof of sequent

$$q\ \mathtt{said}\ r\ \mathtt{said}\ x \ \vdash \ q\ \mathtt{implied}\ r\ \mathtt{implied}\ x.$$

involves additional formulas.

2.2 Natural Deduction Calculus for Primal Infon Logic

We introduce primal infon logic by means of a natural deduction calculus, NDP, obtained from the calculus NDF for full infon logic by replacing the implication-introduction rule (→I) with two rules:

$$(\to\mathtt{IW}) \qquad \frac{\Gamma \vdash y}{\Gamma \vdash (x \to y)}$$

$$(\mathtt{Trans}) \qquad \frac{\Gamma \vdash x, \quad \Gamma, x \vdash y}{\Gamma \vdash y}$$

Rule (→IW) is a weaker (hence the W) implication-introduction rule. Rule (Trans) reflects Corollary 2.2.

LEMMA 2.3. *In either calculus,*

if $\Gamma \cup \{x_1, \ldots, x_n\} \vdash y$ and $\Gamma \vdash x_1, \ldots, \Gamma \vdash x_n$
then $\Gamma \vdash y$.

PROOF. Induction on n. If $n = 0$, the lemma is trivial. Suppose that $n > 0$ and the lemma has been proved for $n - 1$. Let $\Gamma' = \Gamma \cup \{x_1, \ldots, x_{n-1}\}$. Use (Premise Inflation) to derive $\Gamma' \vdash x_n$ from the last premise. The first premise is $\Gamma', x_n \vdash y$. Now use (Trans) and then the induction hypothesis. □

Q: Rule (→IW) is ridiculous. Why do you need the implication if you already have the conclusion?

A: A principal may know the conclusion y but may be willing to share only an implication $x \to y$ with another principal. Besides, the implication $x \to y$ may be needed for derivation as it may occur as the premise of another implication.

3. SEMANTICS, SOUNDNESS AND COMPLETENESS SUBFORMULA PROPERTY

3.1 Semantics for Full Infon Logic

Recall that a quasi-order $\leq$ is a binary relation that is reflexive and transitive. We say that subset U of a quasi-ordered set $(W, \leq)$ is a *cone* if $u \in U$ and $u \leq v$ imply $v \in U$ for all u, v in W. A Kripke structure for propositional intuitionistic logic [Kripke 1965] can be defined as a triple $(W, \leq, C)$, where the following hold.

K1. W is a nonempty set whose elements are traditionally called worlds.
K2. $\leq$ is a quasi-order of W.
K3. C assigns a cone to every infon variable.

By induction, every quotation-free formula z is assigned a cone $C(z)$ of worlds as follows.

K4. $C(\top) = W$.
K5. $C(x \wedge y) = C(x) \cap C(y)$.
K6. $C(x \to y) = \{u : C(x) \cap \{v : v \geq u\} \subseteq C(y)\}$.

Here $\top$ plays the traditional role of propositional constant "true." It is easy to check, by induction on formula z, that every $C(z)$ is indeed a cone.

We extend this definition to accommodate full infon logic. A Kripke structure for full infon logic is a quintuple $(W, \leq, C, S, I)$ where $W, \leq, C$ are as defined earlier and where S and I assign binary relations S_q and I_q over W respectively to every principal q in such a way that the following two requirements are satisfied.

K7. $I_q \subseteq S_q$.
K8. If $u \leq w$ and $w\, S_q\, v$ then $u\, S_q\, v$, and the same for I_q.

The cone map C is extended to all formulas by means of clauses K4–K6 and the following two clauses.

K9. $C(q \texttt{ said } x) = \{u : \{v : u\, S_q\, v\} \subseteq C(x)\}$.
K10. $C(q \texttt{ implied } x) = \{u : \{v : u\, I_q\, v\} \subseteq C(x)\}$.

If $u \in C(z)$, we say that z holds in u and that u models z, and we write $u \models z$. Again, it is easy to check, by induction on formula z, that every $C(z)$ is indeed a cone. We consider here only the case when $z = (q \texttt{ said } x)$. Suppose that $u \in C(z)$ and $u \leq w$. By K9, $\{v : u\, S_q\, v\} \subseteq C(z)$, and we need to show that $\{v : w\, S_q\, v\} \subseteq C(z)$. This follows from K8.

The cone map C extends naturally to sets of formulas and to sequents:

— $C(\Gamma) = \bigcap_{x \in \Gamma} C(x)$,
— $C(\Gamma \vdash y) = \{u : C(\Gamma) \cap \{v : v \geq u\} \subseteq C(y)\}$.

The Kripke structure itself *models* a sequent $s = [\Gamma \vdash y]$ if $C(s) = W$ which is equivalent to $C(\Gamma) \subseteq C(y)$. A sequent s is *valid* if every Kripke structure models s.

Corollary 3.1. $C(q \texttt{ said } x) \subseteq C(q \texttt{ implied } x)$

Proof. By K9 and K10, it suffices to show that, for every u, $\{v : u\, I_q\, v\} \subseteq \{v : u\, S_q\, v\}$. This is exactly K7. □

Unary Connectives (q `said`) and (q `implied`) can be viewed as modalities; see for example, Section 4.1 of Bella et al. 2009.

3.2 Semantics for Primal Infon Logic

The definition of Kripke structures for primal infon logic is similar to that for full infon logic except that the deterministic requirement K6 is replaced with the following nondeterministic requirement.

K6W. $C(x \to y)$ is an arbitrary cone subject to the following two constraints:

$$C(y) \subseteq C(x \to y) \subseteq \{u : C(x) \cap \{v : v \geq u\} \subseteq C(y)\}.$$

3.3 Soundness and Completeness

THEOREM 3.2. *In the case of either infon logic, full or primal, the following claims are equivalent for any sequent s.*

(1) *s is provable.*

(2) *s is valid.*

(3) *Every finite Kripke structure models s.*

(4) *There is a proof of s that uses only subformulas of s.*

PROOF. We deal with both logics at once. We prove that (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3) $\Longrightarrow$ (4) $\Longrightarrow$ (1). Implications (4) $\Longrightarrow$ (1), and (2) $\Longrightarrow$ (3) are obvious.

(1) $\Longrightarrow$ (2). Let $K = (W, \leq, C, S, I)$ be an arbitrary Kripke structure. By induction on the given derivation of sequent s we show that K models s. If s is the ($\top$) axiom, use K4. If s is an (x2x) axiom, the desired $C(x) \subseteq C(x)$ is obvious.

Suppose that s is not an axiom. Let u be a world in the cone of the premise of s. We need to show that u models the conclusion of s. Consider the last step in the given derivation of s. Several cases arise. The case of (Premise Inflation) is obvious. The cases ($\wedge$E) and ($\wedge$I) are obvious as well; just use K5.

($\rightarrow$E). By the induction hypothesis, u models x as well as $x \to y$. By K6W (use the second inclusion of the constraint), u models y.

($\rightarrow$I). This case is relevant for the full, but not primal, infon logic. By K6, it suffices to check that $v \models x$ implies $v \models y$ for all $v \geq u$. Suppose $v \models x$. By the induction hypothesis, $v \models y$.

($\rightarrow$IW). This case is relevant for primal infon logic. By the induction hypothesis, u models y. By K6W (use the first inclusion of the constraint), u models $x \to y$.

(Trans). This case is relevant for primal infon logic. By the choice of u, we have $u \in C(\Gamma)$. By the induction hypothesis applied to the first premise, $u \in C(x)$. Now we apply the induction hypothesis to the second premise: $u \in C(y)$.

(S) and (I). These two cases are similar. We consider only (I). By the choice of u, we have $u \in C(q \texttt{ said } \Delta_1) \cap C(q \texttt{ implied } \Delta_2)$. By Corollary 3.1, $u \in C(q \texttt{ implied } (\Delta_1 \cup \Delta_2))$, which by K10, is equivalent to this: $v \in C(\Delta_1 \cup \Delta_2)$ for all v with $u\, I_q\, v$. For any such v, by the induction hypothesis, $v \in C(y)$. By K10, $u \in C(q \texttt{ implied } y)$.

(3) $\Longrightarrow$ (4). Assuming that (4) fails, we construct a finite model $K = (W, \leq, C, S, I)$ for sequent s. Call a formula *native* if it is a subformula of s. A *native theory* is any set u of native formulas closed under native deduction in the following sense: u contains every native formula x such that sequent $u \vdash x$ is provable using only native formulas.

The quasi-order $(W, \leq)$ of K is the set of native theories ordered by inclusion. Set $u\, I_q\, v$ true if v contains every formula x such that $(q \texttt{ implied } x)$ belongs to u or $(q \texttt{ said } x)$ belongs to u. Set $u\, S_q\, v$ true if v contains every formula x such that $q \texttt{ said } x$ belongs to u. Requirements K1–K2 and K7–K8 are obviously satisfied. It remains to define the cone map C. In the case of full infon logic, we could have defined C on the variables and then used clauses K4–K6 and K9–K10 to extend C to compound formulas. For the uniformity of the proof, we choose a different route.

If z is a native formula or $\top$ define $C(z) = \{u : z \in u\}$, so that $u \models z$ if and only if $z \in u$. Clearly requirements K3 and K4 are satisfied. Now use clauses K5–K6 and K9–K10 to extend C to the remaining formulas composed from native formulas and $\top$ by means of connectives $\wedge, \to, q$ `said`$, q$ `implied`. To complete the definition of K, we check that requirements K5, K6 (resp. K6W) and K9–K10 on $C(z)$ are satisfied for any native formula z. This is done by induction on z. The base of induction when z is a variable, is trivial. The induction step splits into several cases.

Case K5. Using the definition of C on native formulas and the fact that every world is closed under native deduction, we have:

$$u \in C(x \wedge y) \iff (x \wedge y) \in u \iff$$
$$x \in u \ \wedge \ y \in u \iff u \in C(x) \cap C(y).$$

Case K6. First we prove that $C(x \to y) \subseteq \{u : C(x) \cap \{v : v \geq u\} \subseteq C(y)\}$. This part is relevant to both infon logics. Suppose that u contains $x \to y$. If a native theory $v \geq u$ also contains x then it contains y.

Second we prove that $\{u : C(x) \cap \{v : v \geq u\} \subseteq C(y)\} \subseteq C(x \to y)$. This part is relevant only to full infon logic. Pick an arbitrary world u such that $C(x) \cap \{v : v \geq u\} \subseteq C(y)$. We claim that sequent $u, x \vdash y$ is provable. Otherwise, there is a native theory v that includes u, contains x but does not contain y, which contradicts the choice of u. By ($\to$I) with $\Gamma = u$, we have that u contains $x \to y$.

Case K6W. This case is relevant for primal infon logic. The right inclusion has already been proven. To prove the left inclusion, suppose that $u \in C(y)$, so that $u \vdash y$ is provable. By ($\to$IW), $u \vdash (x \to y)$ is provable, and so $u \in C(x \to y)$.

Cases K9 and K10. The two cases are similar; we consider only case K9. Consider an arbitrary world u. First, suppose that $u \in C(q$ `said` $x)$, that is u contains $(q$ `said` $x)$. We need to prove that any v with $u\, S_q\, v$ contains x. This follows from the definition of S_q. Second, suppose that $\{v : u\, S_q\, v\} \subseteq C(x)$, that is every native theory v with $u\, S_q\, v$ contains x. Let Δ be the set of formulas y such that u contains $(q$ `said` $y)$, and let Δ^* be the least native theory that includes Δ. By the definition of S_q , we have $u\, S_q\, \Delta^*$. Then Δ^* contains x, so that the sequent $\Delta \vdash x$ is provable. By rule (Said), sequent $u \vdash (q$ `said` $x)$ is provable, and so u contains q `said` x.

Thus K is a legitimate Kripke structure. Finally let s be $\Gamma \vdash x$, and let Γ^* be the least native theory. By the assumption that (4) fails, Γ^* does not contain x. It follows that s fails in K. $\square$

4. FULL INFON LOGIC: COMPLEXITY

THEOREM 4.1. *The validity problem (whether a given formula is valid) for full infon logic is polynomial-space complete.*

The rest of this section is devoted to proving the theorem. The quotation-free fragment of our natural deduction calculus for full infon logic is a calculus for a fragment of propositional intuitionistic logic whose validity problem is pspace (that is polynomial space) hard [Statman 1979]. Hence our problem is pspace hard. It remains to show that the validity problem for full infon logic is pspace.

We use the following idea that goes back to [Ladner 1977], who proved that the validity problem for some modal logics is pspace. Instead of checking validity, check whether the given formula can be refuted "in a tree-like model structure," which may be exponentially large but where the branches have only polynomial length and thus "can be constructed one branch at a time." The idea was developed in a variety of publications, in particular in Halpern and Moses [1992] and in [Schröder and Pattinson 2008]. The latter paper is quite recent, has a survey of related literature, and will delight the fans of category theory.

Instead of constructing a tree-like model structure and examining it branch by branch, we use games; this helps to avoid bookkeeping. Recall that pspace equals alternating polynomial time [Chandra et al. 1981]. We show that the unprovability problem for the natural deduction calculus NDF for full infon logic, whether a given sequent is unprovable, is solvable in alternating polynomial time.

We start with a few auxiliary definitions. A sequent $s = [\Gamma \vdash \varphi]$ *self-refuting* if Γ is closed under s-native deduction but does not contain φ. A formula is *potent* if it has the form $x \to y$ or p `said` x or p `implied` x. Define quotation depth in the obvious way.

$$\begin{aligned}
&\mathrm{QD}(z) = 0 \text{ if } z \text{ is a variable or } \top,\\
&\mathrm{QD}(x \wedge y) = \mathrm{QD}(x \to y) = \max\{\mathrm{QD}(x), \mathrm{QD}(y)\},\\
&\mathrm{QD}(p \texttt{ said } x) = \mathrm{QD}(p \texttt{ implied } x) = 1 + \mathrm{QD}(x),\\
&\mathrm{QD}(\Gamma \vdash z) = \max\{\mathrm{QD}(x) : x \in \Gamma \vee x = z\}.
\end{aligned}$$

THE GAME. Given a sequent $s_0 = [\Gamma_0 \vdash \varphi_0]$, we define a game $G(s_0)$ between two players, *Refuter*, who intends to refute s, and Challenger. Let n be the length of s_0 and d the quotation depth $\mathrm{QD}(s_0)$. Notice that the number of s_0-native formulas is $\leq n$.

Phase 0. Refuter starts the game by guessing a sequent $s_1 = [\Gamma_1 \vdash \varphi_1]$ such that $\Gamma_1 \supseteq \Gamma_0$ and $\varphi_1 = \varphi_0$; Refuter claims that s_1 is self-refuting.

Winning condition. For any $k > 0$, the state of the game after phase $k - 1$ (unless the game terminated earlier) is given by a sequent $s_k = [\Gamma_k \vdash \varphi_k]$. The game continues further provided the following conditions are satisfied.

R1 $\top \in \Gamma_k$, and $\varphi \notin \Gamma_k$, and every formula in Γ_k is s_{k-1}-native.

R2 If $x \wedge y$ is in Γ_k then both x and y are in Γ_k.

R3 If x, y are in Γ_k and $x \wedge y$ is s_k-native then $x \wedge y$ is in Γ_k.

R4 If x is in Γ_k and $x \to y$ is in Γ_k then y is in Γ_k.

C1 There is an s_k native potent formula outside of Γ_k.

Otherwise the game terminates. If at least one of conditions R1–R4 fails, then Challenger wins. If conditions R1–R4 hold but condition C1 fails, then Refuter wins.

Phase number $k > 0$. In state given by sequent $s_k = [\Gamma_k, \varphi_k]$, Challenger chooses a potent s_k-native formula z outside of Γ_k.

Case $z = (x \to y)$. Refuter guesses a sequent $s_{k+1} = [\Gamma_{k+1} \vdash y]$ with $\Gamma_{k+1} \supseteq \Gamma_k \cup \{x\}$.

Case $z = (p\ \texttt{said}\ y)$. Let Δ be the set of formulas x such that formula $p\ \texttt{said}\ x$ belongs to Γ_k. Refuter guesses a sequent $s_{k+1} = [\Gamma_{k+1} \vdash y]$, where Γ_{k+1} consists of formulas native to $\Delta \cup \{y\}$ and $\Gamma_{k+1} \supseteq \Delta$.

Case $z = (p\ \texttt{implied}\ y)$. Let Δ be the set of formulas x such that either $(p\ \texttt{said}\ x)$ or $(p\ \texttt{implied}\ x)$ belongs to Γ_k. Refuter guesses a sequent $s_{k+1} = [\Gamma_{k+1} \vdash y]$, where Γ_{k+1} consists of formulas native to $\Delta \cup \{y\}$ and $\Gamma_{k+1} \supseteq \Delta$.

That completes the description of the game.

Termination. The game terminates in at most $1 + (d+1)n$ phases. Indeed, at every phase k, where Challenger chooses a said-formula or implied-formula, we have $\mathrm{QD}(s_{k+1}) < \mathrm{QD}(s_k)$. Thus there can be at most d such cases. Every stretch of phases where Challenger chooses implications can contain at most n phases as at each such phase the set of hypotheses, grows but there can only be $\leq n$ hypotheses, as there are only $\leq n$ s_0-native formulas. The only exception is the very first stretch, because the set of hypotheses may not grow during the very first phase.

LEMMA 4.2.
1. If s_0 is refutable then Refuter has a winning strategy in $G(s_0)$.
2. If s_0 is valid then Challenger has a winning strategy in $G(s_0)$.

PROOF. (1) Assuming that $s_0 = [\Gamma_0 \vdash \varphi_0]$ is refutable, we construct a winning strategy for Refuter. At phase 0, Refuter chooses Γ_1 to be the least s_0-native theory that includes Γ_0, and he chooses $\varphi_1 = \varphi_0$ so that sequent $s_1 = [\Gamma_1 \vdash \varphi_1]$ is self-refuting.

Suppose that $k > 0$, phase $k-1$ has been executed, the current sequent s_k is self-refuting, the game continues, and Challenger chose a formula z. We show that Refuter can reply with a self-refuting sequent $s_{k+1} = \Gamma_{k+1} \vdash \varphi_{k+1}$ whose form depends on the form of z. But first notice that s_k is not valid because it is self-refuting and condition R1 holds. Let K be the counter-model constructed in the proof of Theorem 3.2 with s_k playing the role of s. Since s_k is self-refuting, Γ_k is an s_k-native theory and thus a world in K.

Case $z = (x \to y)$. Since $(x \to y)$ does not belong to s_k-native theory Γ_k, there is a world $\Gamma_{k+1} \supseteq \Gamma_k$ in K that contains x but not y. The desired $s_k = [\Gamma_{k+1} \vdash y]$.

Case $z = (p\ \texttt{said}\ y)$. Let Δ be as in the definition of phase $k > 0$. Taking into account the rule (Said) of calculus NDF for full infon logic, we see that sequent $t = \Delta \vdash y$ is unprovable. The desired Γ_{k+1} is the least t-native theory that includes Δ and the desired $\varphi_{k+1} = y$.

Case $z = (p\ \texttt{implied}\ y)$ is similar to the previous one.

(2) Suppose that s_0 is valid and thus provable in calculus NDF for full infon logic. We construct a winning strategy for Challenger. Suppose that $k > 0$, phase $k-1$ has been executed and the current sequent s_k is provable. If at least one of the clauses R1–R4 in the winning definition fails, then Challenger wins. Suppose that all four clauses hold. We show that Challenger can choose a potent s_k-native formula in such a way that every legal response s_{k+1} of Refuter is provable.

Since s_k is provable and condition R1 holds, Γ_k is not closed under s_k-native deduction. Hence there exist s_k-native formulas ψ outside of Γ_k such that sequent $[\Gamma_k \vdash \psi]$ is provable in NDF. Challenger should choose a formula z among such formulas ψ, subject to an additional constraint. The proof P of sequent $t = [\Gamma_k \vdash z]$ should be the minimal possible. Clearly t is not an axiom of NDF. Let R be the inference rule used to obtain t in P. Taking into account that conditions R2–R4 hold, R cannot be (Premise Inflation), ($\wedge$E), ($\wedge$I), or ($\to$E). We consider the remaining cases. In all those remaining cases formula z, chosen by Challenger, is potent.

Case $R = (\to\text{I})$, so that z has the form $x \to y$. Refuter guesses a sequent $s_{k+1} = [\Gamma_{k+1} \vdash y]$ with $\Gamma_{k+1} \supseteq \Gamma_k \cup \{x\}$. But the premise of (our application of) R is sequent $\Gamma_k, x \vdash y$, so s_{k+1} is provable.

Case $R =$ (Said), so that z has the form p `said` x. Let Δ be as in the definition of phase k. Refuter guesses a sequent $s_{k+1} = [\Gamma_{k+1} \vdash y]$, where Γ_{k+1} consists of formulas native to $\Delta \cup \{y\}$ and $\Gamma_{k+1} \supseteq \Delta$. But the premise of R is sequent $\Delta \vdash y$; s_{k+1} is provable.

Case $R =$ (Implied) is similar to the previous one. □

Thus the question of whether a given sequent s_0 is valid or refutable can be decided in alternating polynomial time. That concludes the proof of Theorem 4.1.

5. HILBERT-TYPE CALCULUS FOR PRIMAL INFON LOGIC

The rest of this article is devoted to primal infon logic. Let `told`, with or without a subscript, range over {`implied`, `said`}. A string π of the form q_1 $\texttt{told}_1$ q_2 $\texttt{told}_2$ $\ldots q_k$ $\texttt{told}_k$ is a *quotation prefix*; the *length* k of π may be zero. Let `pref` with or without a subscript range over quotation prefixes. If x is a formula:

$$q_1\ \texttt{told}_1\ q_2\ \texttt{told}_2\ \ldots q_m\ \texttt{told}_m\ y,$$

where y is a variable, conjunction, or implication then every quotation prefix (q_1 $\texttt{told}_1$ q_2 $\texttt{told}_2 \ldots q_k$ $\texttt{told}_k$) with $k \le m$ is a *quotation prefix* of x so that (q_1 $\texttt{told}_1$ q_2 $\texttt{told}_2 \ldots q_m$ $\texttt{told}_m$) is the maximal quotation prefix of x.

We say that $\texttt{pref}_1$ is *dominated* by $\texttt{pref}_2$ and write $\texttt{pref}_1 \le \texttt{pref}_2$ if $\texttt{pref}_1$ is obtained from $\texttt{pref}_2$ by replacing some (possibly none) occurrences of `said` with `implied`.

CALCULUS $\mathcal{H}$

Axioms. $\texttt{pref}\ \top$;

Inference rules.

(Pref Deflation) $\dfrac{\texttt{pref}_2\, x}{\texttt{pref}_1\, x}$; where $\texttt{pref}_1 \le \texttt{pref}_2$;

(Pref $\wedge$E) $\dfrac{\texttt{pref}\,(x \wedge y)}{\texttt{pref}\, x}$ $\dfrac{\texttt{pref}\,(x \wedge y)}{\texttt{pref}\, y}$;

(Pref $\wedge$I) $$\frac{\texttt{pref}\,x \qquad \texttt{pref}\,y}{\texttt{pref}\,(x \wedge y)}\,;$$

(Pref $\to$E) $$\frac{\texttt{pref}\,x \qquad \texttt{pref}\,(x \to y)}{\texttt{pref}\,y}\,;$$

(Pref $\to$I) $$\frac{\texttt{pref}\,y}{\texttt{pref}\,(x \to y)}\,.$$

5.1 Soundness and Completeness

Calculus $\mathcal{H}$ is sound and complete with respect to the Kripke semantics for primal logic. Recall the natural deduction calculus NDP of Section 2.2 for primal infon logic.

THEOREM 5.1 (SOUNDNESS AND COMPLETENESS).
For every sequent $s = [\Gamma \vdash \varphi]$, the following claims are equivalent.

(1) *s is derivable in NDP.*

(2) *s is valid.*

(3) *φ is derivable from Γ in $\mathcal{H}$.*

PROOF. (1) and (2) are equivalent by Theorem 3.2. It suffices to prove that (4) implies (1) and (1) implies (4).

(4) *implies (1)*. By induction on a given derivation of φ from Γ in $\mathcal{H}$, we prove $\Gamma \vdash \varphi$ in NDP. If φ is an axiom $\texttt{pref}\ \top$ of $\mathcal{H}$, start with the axiom $\top$ of NDP; repeatedly apply rules (Said) or (Implied) to obtain sequent $\emptyset \vdash (\texttt{pref}\ \top)$; then repeatedly apply (Premise Inflation) to obtain sequent $\Gamma \vdash \texttt{pref}\ \top$. If φ is a hypothesis, use axiom (x2x) of NDP and then repeatedly apply (Premise Inflation). Otherwise several cases arise according to the last step in the given derivation of φ. We consider only the case when rule (Pref$\to$E) was applied at the last step; other cases are similar. By the induction hypothesis, $\Gamma \vdash \texttt{pref}\ x$ and $\Gamma \vdash \texttt{pref}(x \to y)$ are provable. By Lemma 2.3, it suffices to prove the sequent:

$$\Gamma,\ \texttt{pref}\,x,\ \texttt{pref}\,(x \to y)\ \vdash\ \texttt{pref}\,y.$$

Start with an obviously provable sequent $x, (x \to y) \vdash y$ and apply rules (Said), (Implied) to obtain sequent:

$$\texttt{pref}\,x,\ \texttt{pref}\,(x \to y)\ \vdash\ \texttt{pref}\,y;$$

and then repeatedly apply (Premise Inflation).

(1) *implies (4)*. By induction on a given proof P of sequent $s = [\Gamma \vdash \varphi]$ in NDP, we construct a derivation of φ from Γ in $\mathcal{H}$. The base case, where s is an axiom, is obvious. Otherwise several cases arise according to the last step in P. We consider here only two cases.

($\to$E). Suppose that rule in the last step of P is implication elimination. By the induction hypothesis, there exist derivations of x and $x \to y$ from Γ in $\mathcal{H}$. It remains to apply rule (Pref$\to$E) with the empty prefix.

(Implied). Suppose that rule (Implied) was applied in the last step of P, so that Γ has the form $(q\ \texttt{said}\ \Delta_1) \cup (q\ \texttt{implied}\ \Delta_2)$, and φ has the form $q\ \texttt{implied}\ \varphi_0$. By the induction hypothesis, there is a derivation D of φ_0 from $\Delta_1 \cup \Delta_2$ in $\mathcal{H}$. Without loss of generality, D has the form:

$$\Delta_1, \Delta_2, \text{Tail},$$

so that first Δ_1 formulas are listed, then Δ_2 formulas are listed, and then the remaining *tail formulas* z are listed. Let D' be:

$$q\ \texttt{implied}\ \Delta_1,\ q\ \texttt{implied}\ \Delta_2,\ q\ \texttt{implied}\ \text{Tail}.$$

D' is a derivation of φ from $(q\ \texttt{implied}\ \Delta_1) \cup (q\ \texttt{implied}\ \Delta_2)$ in $\mathcal{H}$. Indeed, if a tail formula z of D is an axiom or is obtained from earlier members of D by means of a rule R then $(q\ \texttt{implied}\ z)$ is an axiom or is obtained from the corresponding members of D' by rule R. But formulas $q\ \texttt{implied}\ \Delta_1$ follow from $q\ \texttt{said}\ \Delta_1$. □

5.2 Local Derivations

Calculus $\mathcal{H}$ does not have the subformula property.

Example. Hypotheses

$$p\ \texttt{said}\ x, \quad p\ \texttt{said}\ (x \to y), \quad p\ \texttt{said}\ (y \to z)$$

entail $p\ \texttt{said}\ z$. Indeed, the first two hypotheses entail formula $p\ \texttt{said}\ y$; that formula and the third hypotheses entail $p\ \texttt{said}\ z$. That derivation and indeed all derivations of $p\ \texttt{said}\ z$ from the hypotheses contain formula $p\ \texttt{said}\ y$, which is not a subformula of the hypotheses or the conclusion.

To remedy the absence of the subformula property we will introduce the notion of formulas local to a set of formulas and we will prove the following. If Γ yields φ, then there is a derivation of φ from Γ that uses only formulas local to $\Gamma \cup \{\varphi\}$.

It will be convenient to view derivations as follows. A derivation D of a formula φ from hypotheses Γ as a finite quasi ordered set of items where each item i is a pair: a formula $F(i)$ and a witness $W(i)$ that justifies $F(i)$. The witness $W(i)$ may indicate that $F(i)$ is an axiom or hypothesis. Alternatively, $W(i)$ may specify one or two preceding items (the premise items for i) and an inference rule $R(i)$ that produces $F(i)$ from the formulas of the premise items. The rule $R(i)$ could be a prefix deflation rule, an introduction rule, or an elimination rule; the item i is called a deflation item, an introduction item, or an elimination item, respectively. And there is an item i in D such that $F(i)$ is φ.

Definition 5.2 (*Formula domination*). A formula x *dominates* a formula y if there is a formula z such that $x = \texttt{pref}_1\, z$ and $y = \texttt{pref}_2\, z$ for some $\texttt{pref}_1 \geq \texttt{pref}_2$. A set Γ of formulas dominates a formula y if some formula x in Γ dominates y.

Definition 5.3 (*Components*). The *components* of a formula z are defined by induction:

—z is a component of z,

—if $\texttt{pref}\,(x \wedge y)$ is a component of z then $\texttt{pref}\, x$ and $\texttt{pref}\, y$ are components of z, and

—if $\mathtt{pref}\,(x \to y)$ is a component of z then $\mathtt{pref}\,x$ and $\mathtt{pref}\,y$ are components of z.

The components of a set Δ of formulas are the components of the formulas in Δ.

COROLLARY 5.4 (COMPONENTS). *If x is a component of z then p $\mathtt{told}$ x is a component of p $\mathtt{told}$ z.*

Definition 5.5 (*Local*). A formula x is *local* to a formula z if it is dominated by a component of z. Formula x is local to a set Δ of formulas if it is local to a formula in Δ.

LEMMA 5.6. *Let Γ contain every Γ-local formula derivable from Γ. If Γ yields a formula φ then there is a derivation of φ from Γ that does not use elimination rules.*

PROOF. We prove the theorem by induction on the number ε of elimination items in the given derivation D of φ from Γ. The base case $\varepsilon = 0$ is obvious. Suppose that $\varepsilon > 0$ and pick an earliest elimination item ℓ.

Case 1. $R(\ell)$ is an implication elimination rule. Then $R(\ell)$ has the form:

$$\frac{\mathtt{pref}\,x \qquad \mathtt{pref}\,(x \to y)}{\mathtt{pref}\,y}.$$

Let k be the premise item with $F(k) = \mathtt{pref}\,(x \to y)$. Item k may or may not be a deflation item. In either case, there is a chain $j_0, \ldots, j_n$ of items such that j_0 is not a deflation item, each j_{m+1} is a deflation item with premise j_m, and $j_n = k$. If k is a deflation item, then $n > 0$; otherwise $n = 0$. In either case $F(j_0) = \mathtt{pref}'\,(x \to y)$ for some $\mathtt{pref}' \geq \mathtt{pref}$.

$F(j_0)$ is not an axiom. If $F(j_0) \in \Gamma$ then $F(k) \in \Gamma$ because Γ contains all Γ-local formulas derivable from Γ. But $F(\ell)$ is a component of $F(k)$ and so is local to Γ; therefore $F(\ell) \in \Gamma$, and the claim is obvious. So we may assume that $F(j_0) \notin \Gamma$. By the choice of ℓ, j_0 is not an elimination item. By the construction, j_0 is not a deflation item. Thus j_0 is an introduction item. Taking into account that $F(j_0) = \mathtt{pref}'\,(x \to y)$, we have:

$$R(j_0) = \frac{\mathtt{pref}'\,y}{\mathtt{pref}'\,(x \to y)}.$$

Modify $R(\ell)$ to $\frac{\mathtt{pref}'\,y}{\mathtt{pref}\,y}$. This decreases the number of elimination items and leaves the proof valid.

Case 2. $R(\ell)$ is a conjunction elimination rule. Then $R(\ell)$ has the form $\frac{\mathtt{pref}\,(x \wedge y)}{\mathtt{pref}\,x}$ or the form $\frac{\mathtt{pref}\,(x \wedge y)}{\mathtt{pref}\,y}$. We consider only the second subcase; the first subcase is similar. Let k be the premise item with $F(k) = \mathtt{pref}\,(x \wedge y)$, and let a chain $j_0, \ldots, j_n$ be as in Case 1, so that $F(j_0) = \mathtt{pref}'\,(x \wedge y)$ for some $\mathtt{pref}' \geq \mathtt{pref}$.

As in the previous case, j_0 has to be an introduction item. Taking into account that $F(k) = \texttt{pref}\,(x \wedge y)$, we have:

$$R(j_0) = \frac{\texttt{pref}'\,x \qquad \texttt{pref}'\,y}{\texttt{pref}'\,(x \wedge y)}.$$

Modify $R(\ell)$ to $\frac{\texttt{pref}'\,y}{\texttt{pref}\,y}$. This decreases the number of elimination items and leaves the proof valid. □

LEMMA 5.7. *Let Γ contain every Γ-local formula derivable from Γ. If Γ yields a formula φ, then there is a derivation of φ from Γ and possibly some axioms by means of introduction rules only.*

PROOF. For brevity, derivations from Γ and possibly some axioms, will be called derivations from Γ^+. For every derivation D, the *depth* of an item k is the length d of the longest chain $j_0, \dots, j_n$, where every predecessor is a premise item for the successor. In addition, the depth of D itself is the maximal depth of its items.

By Lemma 5.6, we may assume that the given derivation D of φ from Γ^+ uses no elimination rules. Without loss of generality, D has a unique deepest item ℓ and $F(\ell) = \varphi$. We prove the lemma by induction on the depth d of D. The base case $d = 1$ is obvious. Suppose that $d > 1$. Two cases arise.

Case 1. ℓ is an introduction item. By the induction hypothesis, for every premise item k for ℓ there is an introduction-only derivation of $F(k)$ from Γ^+. It follows that there is an introduction-only derivation of φ from Γ^+.

Case 2. ℓ is a deflation item. Let k be the premise item for ℓ so that $F(k)$ dominates φ. If $F(k)$ is an axiom, then so is φ, and the claim is obvious. Suppose that $F(k)$ is a hypothesis. Then φ is local to Γ. Recall that Γ contains all Γ-local formulas derivable from Γ. Hence $\varphi \in \Gamma$, and the claim is obvious. So we may assume that $F(k)$ is neither axiom nor hypothesis. Since D has no elimination rules, $R(k)$ is a deflation or introduction rule.

If $R(k)$ is a deflation rule and j is the premise item for k, then $F(j)$ dominates $F(k)$, which dominates φ. Modify $W(\ell)$ by using j, rather than k, as the premise item; and then remove item k. This decreases the derivation depth of φ. By the induction hypothesis, there is an introduction-only derivation of φ from Γ^+.

Thus we may assume that $R(k)$ is an introduction rule. If $R(k)$ is implication introduction, then in terms of formulas only, the final segment of D has the form:

$$\frac{\texttt{pref}'\,y}{\texttt{pref}'\,(x \to y)} \qquad \frac{\texttt{pref}'\,(x \to y)}{\texttt{pref}\,(x \to y)}.$$

Modify it to

$$\frac{\texttt{pref}'\,y}{\texttt{pref}\,y} \qquad \frac{\texttt{pref}\,y}{\texttt{pref}\,(x \to y)}.$$

This reduces Case 2 to Case 1.

Next suppose that $R(k)$ is conjunction introduction. In terms of formulas only, the final segment of D has the form:

$$\frac{\texttt{pref}'\,x \qquad \texttt{pref}'\,y}{\texttt{pref}'\,(x \wedge y)} \qquad \frac{\texttt{pref}'\,(x \wedge y)}{\texttt{pref}\,(x \wedge y)}.$$

Modify it to:

$$\frac{\texttt{pref}'\,x}{\texttt{pref}\,x} \qquad \frac{\texttt{pref}'\,y}{\texttt{pref}\,y} \qquad \frac{\texttt{pref}\,x \qquad \texttt{pref}\,y}{\texttt{pref}\,(x \to y)},$$

so that introduction item k is replaced with two deflation items of the same depth $d-1$, and ℓ becomes an introduction item of depth d. This reduces Case 2 to Case 1. □

Note that the axioms and hypotheses used in the introduction-only proof of φ are components of φ.

THEOREM 5.8 (NORMAL FORM). *If Γ yields φ and Γ^* is the set of Γ-local formulas derivable from Γ, then there is a derivation of φ from Γ^* and possibly some axioms by means of introduction rules only.*

Q: I noticed something. There is a stronger normal form. Narrow the definition of components by modifying the last clause to this: if $\texttt{pref}\,(x \to y)$ is a component of z then $\texttt{pref}\,y$ is a component of z. Do not require that $\texttt{pref}\,x$ is also a component of z. Lemmas 5.6 and 5.7 as well as Theorem 5.8 remain true. The proofs require no changes whatsoever.

Let me reformulate that. Define *precomponents* of a formula z exactly as you defined the components of z except that the last clause is this: if $\texttt{pref}\,(x \to y)$ is a precomponent of z then $\texttt{pref}\,y$ is a precomponent of z. Say that a formula x is *prelocal* to a formula z if it is dominated by a precomponent of z. Then you have the following stronger normal form theorem.

THEOREM. If Γ yields φ and Γ^* is the set of formulas derivable from Γ and prelocal to the formulas of Γ, then there is a derivation of φ from Γ^* and possibly some axioms by means of introduction rules only.

A: You are right. However, the notion of prelocal formulas has a drawback. It is possible that Γ yields φ but every derivation of φ from Γ contains a formula that is not prelocal to $\Gamma \cup \{\varphi\}$. For example, let x, y, z be distinct infon variables and Γ consist of x, y, and $(x \wedge y) \to z$. Then Γ yields z but every derivation of z from Γ uses formula $x \wedge y$ that is not pre-local to Γ.

Definition 5.9 (*Local Proofs*). A derivation D of φ from Γ is *local* if all item formulas in D are local to set $\Gamma \cup \{\varphi\}$.

LEMMA 5.10. *If Γ yields a formula φ that is local to Γ, then there is a local derivation of φ from Γ.*

PROOF. We prove the lemma by induction on the number of nonlocal (to Γ) items i in the given derivation of φ from Γ. If all items are local, we are done. Suppose that there are nonlocal items. Define the *weight* of an item i to be the number of the occurrences of propositional connectives in the formula $F(i)$. Let w be the weight of the heaviest nonlocal item. Pick a latest nonlocal item k of weight w.

First consider the case where $F(k)$ is an axiom. Then $w = 0$, and all items of positive weight are local. If no items use k as a premise item then remove k, and we are done. Otherwise let ℓ range over items that use k as a premise item. If ℓ is an elimination item, $R(\ell)$ is implication elimination, and $F(k)$ is the minor premise of $F(\ell)$. The corresponding major premise is of positive weight and thus local to Γ. But then k is local to Γ, which is impossible. If ℓ is an introduction item, then ℓ is of positive weight and thus local. But then k is local, which is impossible. Thus every ℓ is a deflation item so that every $F(\ell)$ is an axiom. Modify every $W(\ell)$ so that the justification for $F(\ell)$ is that it is an axiom. After that we can remove k, reducing the number of nonlocal items.

Assume that $F(k)$ is not an axiom. Since k is nonlocal, $F(k)$ is not a hypothesis. Hence there is a rule $R(k)$. It may or may not be a deflation rule. In either case, there is a chain $j_0, \ldots, j_n$ of items such that j_0 is not a deflation item, each j_{m+1} is a deflation item with premise j_m, and $j_n = k$. Here n can be zero. Formula $F(j_0)$ cannot be an axiom; otherwise $F(k)$ would be an axiom. $F(k)$ cannot be a hypothesis; otherwise $F(k)$ would be local, which is not the case. Thus there is a rule $R(j_0)$. That rule cannot be an elimination rule; otherwise one of its premises would be a nonlocal formula heavier than $F(k)$. Thus $R(j_0)$ is an introduction rule.

We consider only the case where $R(j_0)$ is an implication introduction so that it has the form $\frac{\texttt{pref}_1\, y}{\texttt{pref}_1\, (x \to y)}$. Accordingly $R(k)$ is $\texttt{pref}_2\, (x \to y)$ for some $\texttt{pref}_2 \le \texttt{pref}_1$. Let ℓ range over the items that use k as a premise item for ℓ. In particular $R(\ell)$ exists. Since k is a heaviest nonlocal item, ℓ cannot be an introduction item. Since k is a latest item of weight w, $R(\ell)$ cannot be a deflation rule. Thus $R(\ell)$ is an elimination rule. Taking into account the form of $F(k)$, the rule $R(\ell)$ is implication elimination. Taking into account that k is a heaviest item, $F(k)$ is the longer premise in $R(\ell)$. Thus $R(\ell)$ has the form:

$$\frac{\texttt{pref}_2\, x \qquad \texttt{pref}_2\, (x \to y)}{\texttt{pref}_2\, y}.$$

Modify $R(\ell)$ to $\frac{\texttt{pref}_1\, y}{\texttt{pref}_2\, y}$. In other words, ignore k and use j_0 as the premise item. When this is done for all ℓ, item k can be removed. $\square$

Theorem 5.8 and Lemma 5.10 give the following theorem.

THEOREM 5.11 (LOCAL PROOFS). *If Γ yields φ then there is local derivation of φ from Γ.*

Q: Do you really need local formulas in addition to the components of $\Gamma \cup \{\varphi\}$? Maybe there is a derivation of φ from Γ that uses only the components whenever φ is derivable from Γ.

A: Here is a counter example to your conjecture. Let x, y, z, be distinct infon variables, and let Γ consist of formulas:

$$p\ \texttt{implied}\ x, \quad p\ \texttt{said}\ (x \to y), \quad p\ \texttt{said}\ (y \to z).$$

Formula $\varphi = (p\ \texttt{implied}\ z)$ is derivable from Γ but any derivation of it from Γ involves local formula $(p\ \texttt{implied}\ y)$ that is not a component of $\Gamma \cup \{\varphi\}$. Furthermore, if y is not a variable and $y = (q\ \texttt{said}\ y')$, then any derivation of φ from Γ involves a quotation prefix $(p\ \texttt{implied}\ q\ \texttt{said})$ that is not a prefix of any component.

Andreas Blass noticed the following corollary.

THEOREM 5.12 (INTERPOLATION). *If Γ yields φ, then there is a set Δ of formulas that are simultaneously local to Γ and local to φ, and such that, (1) there is a derivation of every Δ formula from Γ using only formulas local to Gamma, and (2) there is a derivation of φ from Δ using only formulas local to φ.*

PROOF. Suppose that Γ yields φ, and let Γ^* be the set of Γ-local formulas derivable from Γ. By Theorem 5.8, there is an introduction-only derivation D of φ from Γ^* and possibly some axioms. The desired Δ consists of item formulas $F(i)$ in D, where i is a hypothesis in D according to the justification $W(i)$.

Every $x \in \Delta$ is local to Γ and thus, by Theorem 5.11, there is a derivation of x from Γ that uses only formulas local to Γ. Since D is introduction-only, it uses only formulas local to φ. $\square$

6. PRIMAL INTUITIONISTIC LOGIC

To logicians, primal intuitionistic (or constructive) logic may be of interest in its own right. In this connection, in Section 6.1, we specialize the relevant results of the previous section to the case of primal intuitionistic logic. In Section 6.2 we construct a linear time algorithm for the multiple derivation problem (defined in Section 6.1) for primal intuitionistic logic. The algorithm will be generalized in the next section. For brevity, primal intuitionistic logic will be called PCL, which is an allusion to "Primal Constructive Logic."

6.1 Syntax and Semantics

PCL formulas are built from variables and constant $\top$ by means of conjunction and implication. A PCL natural deduction calculus is obtained from the natural deduction calculus for primal infon logic by removing inference rules (Said) and (Implied):

Axioms.

$(\top)$	$\vdash \top$
(x2x)	$x \vdash x$

Inference rules.

(Premise Inflation) $$\frac{\Gamma \vdash y}{\Gamma, x \vdash y}$$

(∧E) $$\frac{\Gamma \vdash x \wedge y}{\Gamma \vdash x} \qquad \frac{\Gamma \vdash x \wedge y}{\Gamma \vdash y}$$

(∧I) $$\frac{\Gamma \vdash x \quad \Gamma \vdash y}{\Gamma \vdash x \wedge y}$$

(→E) $$\frac{\Gamma \vdash x \quad \Gamma \vdash x \to y}{\Gamma \vdash y}$$

(→IW) $$\frac{\Gamma \vdash y}{\Gamma \vdash x \to y}$$

(Trans) $$\frac{\Gamma \vdash x, \quad \Gamma, x \vdash y}{\Gamma \vdash y}$$

Kripke structures for PCL are triples $(W, \leq, C)$ subject to conditions K1–K5 in Section 2.1 and condition K6W in Section 2.2. Theorem 3.2 holds for PCL: for every PCL sequent s the clauses 1–4 are equivalent.

A Hilbert type calculus $\mathcal{H}_{\text{PCL}}$ for PCL is a simplification of calculus $\mathcal{H}$ of the previous section:

Axiom. $\top$

Inference rules.

(∧e) $$\frac{x \wedge y}{x} \qquad \frac{x \wedge y}{y}$$

(∧i) $$\frac{x \quad y}{x \wedge y}$$

(→e) $$\frac{x \quad x \to y}{y}$$

(→i) $$\frac{y}{x \to y}$$

The Soundness and Completeness Theorem 5.1, the Normal Form Theorem 5.8 and the Local Proofs Theorem 5.11 remain valid. In the case of primal constructive logic, the components of a formula z are exactly the subformulas of z.

6.2 Linear Time Theorem for Primal Constructive Logic

THEOREM 6.1. *There is a linear time algorithm for the multiple derivability problem for primal constructive logic. Given hypotheses Γ and queries Q, the algorithm determines which of the queries in Q follow from the hypotheses Γ.*

PROOF. Formulas local to $\Gamma \cup Q$ will be simply called local. By Theorem 5.11, we may restrict attention to derivations where all formulas are local. The idea is to compute all local (to $\Gamma \cup Q$) consequences of Γ. This is obviously sufficient.

Without loss of generality we may assume that $\top$ does not occur in $\Gamma \cup Q$. It follows that $\top$ does not occur in any local formula. Let n be the length of the input sequence Γ, Q. The *key* $K(y)$ of a local formula y is the position of the first symbol of the first occurrence of y in the input sequence.

Parse tree. Run a parser on the input string producing a parse tree. The subtrees of the hypotheses and the queries hang directly under the root. Each node of the parse tree has a label that is, or represents (according to the lexical analyzer), a variable or connective. The label length is $O(\log(n))$ due to the lexical analysis phase of parsing. Extra tags mark hypotheses and queries; such a tag is not a part of the official label.

By induction, define the *locutions* $L(u)$ of nodes u. If u is a node with children $u_1, \ldots, u_k$, then:

$$L(u) = \text{Label}(u)(L(u_1), \ldots, L(u_k)).$$

The locutions are being introduced for use in our analysis of the algorithm; the algorithm will not have the time to produce all the locutions.

Each node u is associated with a particular occurrence of $L(u)$ in the input string. The initial position of $L(u)$ in the input string is the *key* $K(u)$ of u. Nodes u and v are *homonyms* if $L(u) = L(v)$. A node with the least key in its homonymy class is a *homonymy original*. A homonymy original u represents the locution $L(u)$; its key is the key of the locution. We presume that the nodes u come with *homonymy pointers* $H(u)$ initially set to nil.

Homonymy originals. Run the Cai-Paige algorithm [Cai and Paige 1995] on the parse tree. The algorithm partitions the (keys of) nodes u into buckets B_ℓ according to the height ℓ of u, that is the height (or depth) of the subtree rooted at u. Furthermore, every bucket is ordered according to the lexicographic order of locutions $L(u)$. Note that two homonyms have the same height and thus belong to the same bucket. Homonyms are ordered according to their keys. The Cai-Paige algorithm sets every homonymy pointer $H(u)$ to the homonymy original of u. This wonderful algorithm runs in linear time.

Preprocessing. Create a table T of records indexed by the homonymy originals u such that $L(u)$ is a formula. A record $T(u)$ has five fields. One of them is the *status field* $S(u)$ with values in the set $\{1, 2, 3\}$. The status field is dynamic; its value may change as our algorithm runs. All other fields are static; once created their values remain immutable.

The status field $S(u)$ determines the current status of the formula $L(u)$. Formulas of status $1, 2, 3$ will be called *raw, pending, processed*, respectively. A raw formula has not been derived. A pending formula has been derived but remains a subject of some processing. A processed formula has been derived and processed. Initially every $S(u) = 1$. Traverse the parse tree setting the status $S(u)$ to 2 if $L(u)$ is tagged as a hypothesis.

The static fields of the record $T(u)$ are $(\wedge, \text{left})$, $(\wedge, \text{right})$, $(\rightarrow, \text{left})$, and $(\rightarrow, \text{right})$. Each entry in these fields is a sequence of (the keys of) nodes. To compute those sequences, traverse the parse tree in the depth-first manner. If the current node v is the left child of a node v' with label "$\wedge$" then append $H(v')$ to the $(\wedge, \text{left})$ sequence of $H(v)$ and move on. The cases where v is the right child of v' or the label of v' is $\to$ or both are treated similarly.

Processing. Walk through the table T and process every pending formula $L(u)$ in turn. When the processing of $L(u)$ is finished do the following.

—Set $S(u) = 3$, indicating that $L(u)$ is processed.

—If u is tagged as a query, then output $K(u)$.

The processing consists of firing, one after another, the inference rules applicable to $L(u)$.

Rule ($\wedge$e) requires that $L(u)$ has the form $x \wedge y$. If any of the formulas x, y, is raw, make it pending. More exactly, let u_1, u_2, be the left and right children of u. If $S(H(u_i)) = 1$, set $S(H(u_i)) = 2$.

Rule ($\wedge$i) may be applied in two ways depending on whether we view $L(u)$ as the left or right premise of the rule. The two cases—call them the left and right cases—are similar; we describe here only the left case, where $L(u)$ is the left conjunct x of a formula $x \wedge y$. For any such y that has been derived but $x \wedge y$ is still raw, make $x \wedge y$ pending. More exactly, for every v in the $(\wedge, \text{left})$ sequence of $T(u)$ do the following. Note that $L(v) = L(u) \wedge L(w)$ where w is the right child of v. If $S(H(w)) > 1$ and $S(v) = 1$, set $S(v) = 2$.

Rule ($\to$e) may be applied in two ways depending on whether we view $L(u)$ as the left or right premise of the rule. This time around the two cases—call them the left and right cases—are quite different. The left case is similar to the left case in the application of rule ($\wedge$i). For every v in the $(\to, \text{left})$ sequence of $T(u)$ do the following. Note that $L(v) = L(u) \to L(w)$ where w is the right child of v. If $S(v) > 1$ and $S(H(w)) = 1$, set $S(H(w)) = 2$. The right case requires that $L(u)$ has the form $x \to y$. If x has been derived and y is raw, make y pending. More exactly, let u_1, u_2 be the left and right children of u. If $S(H(u_1)) > 1$ and $S(H(u_2)) = 1$, set $S(H(u_2)) = 2$.

Rule ($\to$i) has only one premise $y = L(u)$. Any raw formula of the form $x \to y$ becomes pending. More exactly, for every v in the $(\to, \text{right})$ sequence of $T(u)$ do the following. Note that $L(v) = L(w) \to L(u)$, where w is the homonymy original of the left child of v. If $S(v) = 1$, set $S(v) = 2$.

This completes the algorithm.

Proof of correctness. Note that every pending formula becomes processed. Obviously only provable formulas become pending. To prove the correctness of the algorithm, it suffices to prove that every provable formula $L(v)$ becomes pending. We do that by induction on the proof length of $L(v)$. If $L(v)$ is a hypothesis, it becomes pending when the table is formed. Otherwise several cases arise depending on the rule used at the last step of the given proof of $L(v)$. All cases are pretty obvious. We consider just one of those cases.

Case ($\wedge$i) is where $L(v) = x \wedge y$ and x, y have been derived earlier. By symmetry we may assume without loss of generality that x became pending first. Formula y is $L(u)$ for some u. But then v occurs in the $(\wedge, \text{right})$ sequence of

$T(u)$. Accordingly $L(v)$ becomes pending as a result of applying the right case of rule ($\wedge$i) in the processing of $L(u)$.

Time complexity. It remains to check that the algorithm is indeed linear time. Obviously the parse-tree and table stages are linear time. The only question is whether the processing stage is linear time. Instead we claim something else. Note that the processing of a pending formula $L(u)$ is done in a fixed finite number of phases. At each phase, we make some number $k(u)$ of attempts to apply a particular inference rule viewing $L(u)$ as one of the premises. Each attempt takes bounded time. The number of attempts is not bounded. It suffices to prove, however, that $\sum_u k(u) = O(n)$.

The proof is similar for all phases. Here we consider only the phase when we attempt to apply rule ($\wedge$e) with $L(u)$ as the left premise x. In this phase the number $k(u)$ is the length of the ($\wedge$, left) sequence of u. But the ($\wedge$, left) sequences for distinct records are disjoint. And so $\sum_u k(u) \le n$. □

7. LINEAR TIME THEOREM FOR PRIMAL INFON LOGIC

We return to primal infon logic. We will be working with fragments of the Hilbert-type calculus $\mathcal{H}$ for primal infon logic. To recall $\mathcal{H}$, see Section 5.

In Section 4 we gave the obvious definition of the quotation depth of formulas. In the case of primal infon logic, another definition of quotation depth is more pertinent.

Definition 7.1 (*Primal quotation depth*). The primal quotation depth of formulas is defined by induction:

—$\delta(x) = 0$ if x is a variable.
—$\delta(p\ \texttt{told}\ x) = 1 + \delta(x)$.
—$\delta(x \wedge y) = \max\{\delta(x), \delta(y)\}$.
—$\delta(x \to y) = \delta(y)$

Further $\delta(\Gamma) = \max_{x\in\Gamma} \delta(x)$ for any set Γ of formulas.

Recall Definition 1.1 of the multiple derivability problem MD(L) for a logic L.

THEOREM 7.2. *For every natural number d, there is a linear time algorithm for the multiple derivability problem for primal infon logic restricted to formulas z with* $\delta(z) \le d$.

Q: The definition of primal quotation depth is strange. The clause $\delta(x \to y) = \delta(y)$ ignores $\delta(x)$. If `foo` is a quotation-free formula then $\delta((p\ \texttt{said}\ q\ \texttt{said foo}) \to \texttt{foo}) = 0$. The formula involves quotation but its primal quotation depth is zero.

A: Well, the strange definition makes the theorem stronger.

Q: Does the additional strength matter?

A: It does. For example it allows us to interpret authorization logic SecPAL [Becker et al. 2007] in the stratum of depth ≤ 2 of primal infon logics [Gurevich and Neeman 2009].

7.1 Reduction to Main Lemma

Definition 7.3 (*Regular*). A formula x is regular if it has no occurrences of $\top$. A derivation is regular if all its formulas are regular. $\mathcal{R}$ is the fragment of $\mathcal{H}$ obtained by removing $\top$ from the language and removing the axioms from the calculus.

Two formulas are *equivalent* if each of them entails the other in primal infon logic.

LEMMA 7.4. *(1) For any formula z there is an equivalent formula z' that is either an axiom or regular.*
(2) There is a linear time algorithm that, given a formula z, computes the equivalent formula z'.
(3) Any local derivation of a regular formula from regular hypotheses is regular.
(4) There is a linear-time reduction of MD($\mathcal{H}$) *to* MD($\mathcal{R}$).

PROOF. (1) Easy induction on z. We consider only the case $z = x \to a$ in the induction step, where a is an axiom. In that case z' may be a. Indeed $z \vdash a$ because a is an axiom, and $a \vdash z$ by (Pref$\to$I).
(2) Execute the *regularization algorithm* implicit in the proof of 1. One appropriate data structure is parse tree. Even if z is given as a string, in linear time you can construct the parse tree of z and then execute the regularization algorithm.
(3) By the definition of local derivation.
(4) Apply the regularization algorithm to the hypotheses and queries. If a modified hypothesis is an axiom, remove it. If a modified query is an axiom, mark it derivable and remove it. □

LEMMA 7.5. *Theorem 5.1 remains true if sequent s is assumed to be regular and if $\mathcal{H}$ is replaced with $\mathcal{R}$.*

PROOF. The proof of Theorem 5.1 needs only two minor modifications, in fact simplifications. In the proof that (4) implies (1), ignore the case where φ is an axiom of $\mathcal{H}$. Similarly, in the proof that (2) implies (4), in the proof that D' is a derivation, ignore the case where the tail formula z is an axiom of $\mathcal{H}$. □

LEMMA 7.6. *(1) For any inference rule in $\mathcal{R}$, the primal quotation depth of the conclusion is bounded by the primal quotation depth of the premise(s).*
(2) If Γ entails φ in $\mathcal{R}$, then $\delta(\varphi) \leq \delta(\Gamma)$.
(3) The restriction $\mathcal{R}_d$ of $\mathcal{R}$ to formulas z with $\delta(z) \leq d$ is a calculus in its own right; all inference rules of $\mathcal{R}$ produce formulas in $\mathcal{R}_d$ when applied to formulas in $\mathcal{R}_d$.

PROOF. Claim (1) is obvious, but note that the unusual clause $\delta(x \to y) = \delta(y)$ in the definition of primal quotation depth is used in the case of rule (Pref $\to$I). Claims (2) and (3) are obvious as well. □

MAIN LEMMA 7.7. *For every natural number d, there is a linear time algorithm for the multiple derivation problem* MD(R_d) *for R_d.*

Clearly Theorem 7.2 follows from Main Lemma. We prove Main Lemma in the next subsection.

7.2 Proof of Main Lemma

Given a positive integer d, we construct a linear time algorithm for $\mathrm{MD}(L_d)$ by modifying the linear-time algorithm of Section 5. Recall that $\mathtt{told}$ with or without a subscript ranges over $\{\mathtt{said}, \mathtt{implied}\}$, and $\mathtt{pref}$ with or without a subscript ranges over quotation prefixes $q_1\ \mathtt{told}_1\ \ldots\ q_k\ \mathtt{told}_k$, where k is the length of $\mathtt{pref}$. We say that a quotation prefix is relevant if its length is $\leq d$.

Parsing. Parse the input as in Section 6.2 except that now we have additional labels $p\ \mathtt{told}$. There is a problem, however. In Section 6.2, every local formula was the locution $L(u)$ of some node u of the parse tree. Now it is not necessarily the case. To remedy the situation, we graft extra nodes into the parse tree. This is not the optimal remedy but it simplifies the exposition.

By induction on nodes u, define $\mathtt{pref}(u)$. If u is the root, then $\mathtt{pref}(u)$ is empty. Suppose that v is the parent of u. If the label of v is of the form $p\ \mathtt{told}$ then $\mathtt{pref}(u)$ is $\mathtt{pref}(v)$ appended with the label of v; otherwise $\mathtt{pref}(u) = \mathtt{pref}(v)$. Let $C(u)$ be the formula $\mathtt{pref}(u)\,L(u)$. It is easy to check that every $C(u)$ is a component and that every component is $C(u)$ for some u. Call node u relevant if (1) u is not the root of the parse tree so that $L(u)$ is a formula and (2) $\mathtt{pref}(u)$ is relevant.

The *fan* $F(\mathtt{pref})$ of a prefix $\mathtt{pref}$ is a tree of prefixes. It contains all prefixes $\mathtt{pref}' \leq \mathtt{pref}$. Further, it contains a prefix $q_1\ \mathtt{told}_1\ \ldots\ q_i\ \mathtt{told}_i$ of length i if it contains a prefix $q_1\ \mathtt{told}_1\ \ldots\ q_i\ \mathtt{told}_i\ q_{i+1}\ \mathtt{told}_{i+1}$ of length $i+1$, which is a parent of $q_1\ \mathtt{told}_1\ \ldots\ q_i\ \mathtt{told}_i$. Every node in $F(\mathtt{pref})$ has at most two children and at most one parent. If $d = 2$, then $F(\mathtt{pref})$ contains at most 7 nodes. Now turn $F(\mathtt{pref})$ upside down, and let $F'(\mathtt{pref})$ be the result. Normally our trees go downward so that the root is at the top. $F'(\mathtt{pref})$ grows upward.

Traverse the parse tree in the depth-first manner and graft a fresh copy $F(u)$ of $F'(\mathtt{pref}(u))$ at every relevant node u. There is a one-to-one correspondence $\xi : F(u) \longrightarrow F'(\mathtt{pref}(u))$. If $v \in F(u)$ and $\xi(v)$ is a prefix $q_1\ \mathtt{told}_1\ \ldots\ q_i\ \mathtt{told}_i$ of length $i > 0$, then the label of v is $q_i\ \mathtt{told}_i$ and the key of v is the pair:

$$(\mathrm{Key}(u),\ \mathtt{told}_1\ \ldots\ \mathtt{told}_i).$$

The keys are ordered lexicographically. We do not distinguish between $\mathrm{Key}(u)$ and the pair $(\mathrm{Key}(u), s)$, where string s is empty. Every node u of the resulting *parse structure* has at most three parents, and the nodes $\leq u$ form a tree.

Remark 7.8. For every relevant original node u, the formula $\mathtt{pref}(u)\,L(u)$ is a component of $\Gamma \cup Q$. If $\mathtt{pref} \leq \mathtt{pref}(u)$, then $\mathtt{pref}\,L(u)$ is local. Every local formula is obtained this way. Thus the parse structure has a node for every local formula (and for some nonlocal formulas).

Homonymy originals. As in Section 6.2, run the Cai-Paige algorithm on the parse structure and compute homonymy pointers $H(u)$.

Preprocessing. Let T_1 be the table T constructed in Section 6.2. T_1 is a one-dimensional table of records $T_1(u)$ indexed by nodes u such that u is a homonymy original. Now, in addition to T_1, we construct a sparse two-dimensional table T_2. The rows of T_2 are indexed by original nodes u, and the columns are indexed by relevant prefixes π. If there is a node v with $L(v) = \pi\ L(u)$, then $T_2(u, \pi) = \{H(v)\}$; otherwise $T_2(u, \pi) = \emptyset$. A traversal around the parse structure suffices to fill in table T_2.

Remark 7.9. We graft nodes and then put only some of them into table T_2. This is not the most efficient way to do things. One can forgo grafting, construct table T_2 directly, and refer to the table instead of to grafted nodes in the following processing. We thought that grafting would simplify the exposition.

Remark 7.10. It is more efficient to combine preprocessing with parsing.

Processing. As in Section 6.2, we walk through the table T and process every pending formula $L(u)$ in turn. The processing consists of firing, one after another, the inference rules applicable to $L(u)$. The case of rule:

$$\frac{\texttt{pref}_2\, x}{\texttt{pref}_1\, x},$$

is new. Suppose that $L(u) = \texttt{pref}_2\, x$ and let $\texttt{pref}_1 \le \texttt{pref}_2$. The descendant u_0 of u with locution x has an ancestor v with locution $\texttt{pref}_1\, x$. If $H(v)$ is raw, make it pending.

As far as the remaining rules are concerned, the situation is similar to that in Section 6.2. For example, consider the application of the rule:

$$\frac{\texttt{pref}\,(x \wedge y)}{\texttt{pref}\, x},$$

to a formula $L(u) = \texttt{pref}(x \wedge y)$. Find the descendant u_0 of u with $L(u_0) = x \wedge y$. The left child of u_0 has an ancestor v with locution $\texttt{pref}\, x$. If $H(v)$ is raw, make it pending.

For a more interesting example, consider the application of rule:

$$\frac{\texttt{pref}\, x \qquad \texttt{pref}\, y}{\texttt{pref}\,(x \wedge y)},$$

to a formula $L(u) = \texttt{pref}\, x$. Find the descendant u_0 of u with $L(u) = x$. For each v in the $(\wedge, \text{left})$ field of record $T_1(H(u_0))$ do the following.

Check the entry $T_2(v, \texttt{pref})$. If it is empty, do nothing. Otherwise let w be the node in the entry. The locution of v is $x \wedge y$, and the locution of w is $\texttt{pref}\,(x \wedge y)$. Let w_0 be the descendent of w with locution $x \wedge y$. The right child of w_0 has an ancestor w' with locution $L(w') = \texttt{pref}\, y$. If the status of $H(w')$ is > 1, so that $\texttt{pref}\, y$ has been proved, but the status of w is 1, so that $\texttt{pref}\,(x \wedge y)$ has not been proved, then set the status of w to 2; otherwise do nothing.

Correctness and time complexity. The proof of the correctness of the algorithm and the analysis of time complexity of Section 6.2 survive with minor changes.

ACKNOWLEDGMENTS

Conversations with Nikolaj Bjørner, Andreas Blass and Grigori Mints have been most helpful. Last-minute remarks of Sergio Mera were useful as well.

REFERENCES

AVRON, A. 2010. Tonk—A full mathematical solution. In *Hues of Philosophy: Essays in Memory of Ruth Manor,* Ed. A. Biletzki, College Publication, 17–42.

AVRON, A. AND LAHAV, O. 2009. Strict canonical constructive systems. In *Fields of Logic and Computation: Essays Dedicated to Yuri Gurevich on the Occasion of his 70th Birthday,* Eds. A. Blass, N. Dershowitz, and W. Reisig, Lecture Notes in Computer Science, vol. 6300, Springer-Verlag, 75–94.

BECKER, M. Y., FOURNET, C., AND GORDON, A. D. 2007. SecPAL: Design and Semantics of a decentralized authorization language. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium.* 3–15.

BELLA, H., GABBAY, D. M., GENOVESE, V., AND VAN DER TORRE, L. 2009. Fibered security language. *Studia Logica 92*, 3, 395–436.

CAI, J. AND PAIGE, R. 1995. Using multiset discrimination to solve language processing problems without hashing. *Theor. Comput. Sci. 145,* 1–2, 189–228.

CHANDRA, A. K., KOZEN, D. C., AND STOCKMEYER, L. J. 1981. Alternation. *J. ACM 28,* 1, 114–133.

CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. 1990. *Algorithms*. MIT Press.

DEVLIN, K. 1991. *Logic and Information.* Cambridge University Press.

GUREVICH, Y. AND NEEMAN, I. 2008. DKAL: Distributed-knowledge authorization language. In *Proceedings of the 21st Annual IEEE Computer Security Foundations Symposium.* 149–162.

GUREVICH, Y. AND NEEMAN, I. 2009. DKAL 2—A simplified and improved authorization language. Tech. rep. MSR-TR-2009-11, Microsoft Research.

GUREVICH, Y. AND ROY, A. 2009. Operational semantics for DKAL: Application and analysis. In *Proceedings of the 6th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus).* Lecture Notes in Computer Science, vol. 5695, Springer-Verlag, 149–158.

HALPERN, J. Y. AND MOSES, Y. 1992. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell. 54.* 319–379.

KRIPKE, S. 1965. Semantical analysis of intuitionistic logic I. In *Formal Systems and Recursive Functions*, J. W. Addison, L. Henkin, and A. Tarski, Eds., North-Holland.

LADNER, R. E. 1977. The computational complexity of provability in systems of modal propositional logic. *Siam J. Comput.,* 467–480.

MINTS, G. 2000. *A Short Introduction to Intuitionist Logic.* Kluwer.

SCHRÖDER, L. AND PATTINSON, D. 2008. PSPACE bounds for rank-1 modal logics. *ACM Trans, Comput. Logic.*

STATMAN, R. 1979. Intuitionistic propositional logic is polynomial-space complete. *Theor. Comput. Sci.,* 67–72.

Received July 2009; revised March 2010; accepted April 2010