THE REVIEW OF SYMBOLIC LOGIC
Volume 6, Number 2, June 2013

TRANSITIVE PRIMAL INFON LOGIC

CARLOS COTRINI
Swiss Federal Institute of Technology
and
YURI GUREVICH
Microsoft Research

Abstract. Primal infon logic was introduced in 2009 in connection with access control. In addition to traditional logic constructs, it contains unary connectives p `said` indispensable in the intended access control applications. Propositional primal infon logic is decidable in linear time, yet suffices for many common access control scenarios. The most obvious limitation on its expressivity is the failure of the transitivity law for implication: $x \to y$ and $y \to z$ do not necessarily yield $x \to z$. Here we introduce and investigate equiexpressive "transitive" extensions TPIL and TPIL* of propositional primal infon logic as well as their quote-free fragments TPIL_0 and $\text{TPIL}_0{}^*$ respectively. We prove the subformula property for $\text{TPIL}_0{}^*$ and a similar property for TPIL*; we define Kripke models for the four logics and prove the corresponding soundness-and-completeness theorems; we show that, in all these logics, satisfiable formulas have small models; but our main result is a quadratic-time derivation algorithm for TPIL*.

§1. Introduction. In a brick-and-mortar setting, some access control policies may be vague and even unwritten. The clerks ordinarily know them. When in doubt, they know whom to ask. In the cloud there are no clerks, and policies have to be managed automatically. The most challenging are federated scenarios where the policies of different principals interact in the absence of central authority. Distributed Knowledge Authorization Language (DKAL) was created to deal with such problems (Gurevich & Neeman, 2008; Blass *et al*., 2011). The DKAL project led to the introduction of infon logic and its primal fragment (Gurevich & Neeman, 2009, 2011; Blass & Gurevich, 2010); here infons are pieces (or items) of information.

It turns out that full infon logic is a natural conservative extension of intuitionistic logic with the quotation construct p `said` (Beklemishev & Gurevich, 2012). If x is a formula then p `said` x is a formula. The derivability problem (decide whether a given formula, the query, follows from given hypotheses) is PSPACE complete. Gurevich & Neeman (2009) introduced primal infon logic. Similar ideas were independently discovered from purely proof-theoretic considerations (Avron & Lahav, 2009). A general analysis of infon logic shows that primal infon logic is a natural fragment of full infon logic Beklemishev et al., in preparation.

PROVISO 1.1. *In the rest of the paper, logics are by default propositional.*

Received: January 29, 2012.

doi:10.1017/S1755020312000366

Gurevich & Neeman (2011) investigated basic primal infon logic; the presentation was improved and simplified[1] in Cotrini & Gurevich (2013). We view the version BPIL of basic primal infon logic in Cotrini & Gurevich (2013) as the canonic basic propositional infon logic. The quote-free fragment of BPIL will be denoted BPIL_0. The multiderivation problem (decide which of given queries follows from given hypotheses) for BPIL is solvable in linear time, yet BPIL suffices for many common access control scenarios. The most obvious limitation on the expressivity of BPIL is the failure of the transitivity law for implication: $x \to y$ and $y \to z$ do not necessarily yield $x \to z$. Here we investigate transitive extensions of BPIL and BPIL_0.

The extension TPIL_0 of BPIL_0 with axiom $x \to x$ and rule

$$(\mathrm{trans}_0)\ \frac{x \to y \qquad y \to z}{x \to z}$$

leads to a polynomial-time derivation problem (Savateev, 2009). More exactly, the multiderivation problem for TPIL_0 is solvable in cubic time. Replacing axiom $x \to x$ and rule (trans_0) with a rule

$$(\mathrm{trans}_0{}^*)\ \frac{x_1 \to x_2 \qquad x_2 \to x_3 \quad \ldots \qquad x_{k-1} \to x_k}{x_1 \to x_k}$$

where k is any positive integer, gives an equiexpressive logic $\mathrm{TPIL}_0{}^*$. The replacement seems innocuous. Surprisingly the multiderivation problem for $\mathrm{TPIL}_0{}^*$ is solvable in quadratic time.

One of our reviewers wrote that "[t]he use of quotation prefixes makes the presentation extremely more complicated ... If ... quotation prefixes do play some important role, then this should be clarified." Quotations are indispensable in our applications. Here is a simple example. Imagine that a principal p tells you an infon x. It is reasonable that you learn only p `said` x rather than x itself, isn't it? Accordingly, in the center of our attention, are transitive primal infon logics TPIL and TPIL*. TPIL extends BPIL with axiom $x \to x$ and rule

$$(\mathrm{trans})\ \frac{\mathtt{pref}\,(x \to y) \qquad \mathtt{pref}\,(y \to z)}{\mathtt{pref}\,(x \to z)}$$

where `pref` ranges over quotation prefixes of the form

$$q_1\ \mathtt{said}\ q_2\ \mathtt{said} \ldots\ q_\ell\ \mathtt{said}$$

TPIL* extends BPIL with rule

$$(\mathrm{trans}^*)\ \frac{\mathtt{pref}(x_1 \to x_2) \qquad \mathtt{pref}(x_2 \to x_3) \quad \ldots \qquad \mathtt{pref}(x_{k-1} \to x_k)}{\mathtt{pref}(x_1 \to x_k)}$$

In §2 we introduce Hilbertian calculi for TPIL, TPIL* and their quote-free fragments TPIL_0 and $\mathrm{TPIL}_0{}^*$. In §3, we prove that $\mathrm{TPIL}_0{}^*$ has the subformula property: if φ follows from Γ then there is a derivation of φ from Γ composed from subformulas of (formulas in) $\Gamma \cup \{\varphi\}$. The main result of §3 is that TPIL* has a similar property. In §4, we define semantics and prove the soundness-and-completeness theorems for all four of our logics. Every

[1] The logic itself was simplified. Originally there were two quotation constructs, p `said` and p `implied`, but the subsequent evolution of the DKAL project demonstrated that latter construct was superfluous, and it was removed (Gurevich, 2011).

satisfiable TPIL_0^* formula has a one-element model. Every satisfiable TPIL* formula has a small model.

In §5, we give our main result: a quadratic-time algorithm for the multiderivability problem for TPIL*. The algorithm is rather involved and uses numerous algorithmic tools. Note that linear and quadratic time complexities are much more fragile than the more familiar and robust polynomial time complexity. The algorithm has been implemented (DKAL at CodePlex).

The final §6 is devoted to related work.

§2. Hilbertian calculi. Formulas are built from propositional variables and the propositional constants $\top, \bot$ by means of the binary connectives $\wedge$, $\vee$ and $\rightarrow$ and unary connectives q said, where q ranges over principal constants. Thus every formula x has the form

$$q_1 \texttt{ said } q_2 \texttt{ said } \ldots q_\ell \texttt{ said } y$$

where y, the *body* of x, is an atomic formula or a binary combination (conjunction, disjunction, or implication) of two formulas. If $\ell > 0$ then x is a *quote formula* or simply a *quote*. Every string $\pi = q_1 \texttt{ said } \ldots q_j \texttt{ said}$ with $j \leq \ell$ is a *quotation prefix* of x. If $j = 0$ then π is empty, and if $j = \ell$ then π is the maximal quotation prefix of x.

If a formula x is not a quote, we say that x is a *nonquote* formula. A nonquote formula may have quote subformulas. We say that x is *quote free* if it has no quote subformulas.

2.1. Hilbertian calculus for TPIL. Let x, y range over formulas and pref range over quotation prefixes.

Axioms

$$(\top)\ \texttt{pref}\ \top \qquad\qquad (\mathrm{x2x})\texttt{pref}\ (x \rightarrow x)$$

Inference rules

$$(\wedge\mathrm{i})\ \frac{\texttt{pref}\ x \qquad \texttt{pref}\ y}{\texttt{pref}\ (x \wedge y)} \qquad (\wedge\mathrm{e})\ \frac{\texttt{pref}\ (x \wedge y)}{\texttt{pref}\ x} \qquad \frac{\texttt{pref}\ (x \wedge y)}{\texttt{pref}\ y}$$

$$(\rightarrow\mathrm{i})\ \frac{\texttt{pref}\ y}{\texttt{pref}\ (x \rightarrow y)} \qquad (\rightarrow\mathrm{e})\ \frac{\texttt{pref}\ x \qquad \texttt{pref}\ (x \rightarrow y)}{\texttt{pref}\ y}$$

$$(\vee\mathrm{i})\ \frac{\texttt{pref}\ x}{\texttt{pref}(x \vee y)} \qquad\qquad \frac{\texttt{pref}\ y}{\texttt{pref}(x \vee y)}$$

$$(\mathrm{trans})\ \frac{\texttt{pref}\ (x \rightarrow y) \qquad \texttt{pref}\ (y \rightarrow z)}{\texttt{pref}\ (x \rightarrow z)}$$

Thus we have three introduction rules ($\wedge$i), ($\rightarrow$i), ($\vee$i), two elimination rules ($\wedge$e), ($\rightarrow$e), and the rule (trans).

REMARK 2.1. *The TPIL treatment of quotation prefixes may seem simplistic. In fact, one of our reviewers wrote that "[t]he calculus ... does not include any rule for changing the*

quotation prefix of a formula. It follows that ... one can simply split the [multi-derivation] problem to separate disjoint problems according to the quotation prefixes." Things are more complicated, however. For example, the minor premise $\mathtt{pref}\, x$ *and the conclusion* $\mathtt{pref}\, y$ *of* $(\to e)$ *may have longer maximal prefixes than the maximal prefix* $\mathtt{pref}$ *of the major premise* $\mathtt{pref}(x \to y)$*. By the way, the prefix preservation property does not hold in the sequent calculus for primal infon logic (Gurevich & Neeman, 2011).*

2.2. Hilbertian calculus for TPIL*. The calculus for TPIL* is obtained from that for TPIL by removing the axiom $x \to x$ and replacing the derivation rule (trans) with

$$(\text{trans}^*)\ \frac{\mathtt{pref}(x_1 \to x_2) \qquad \mathtt{pref}(x_2 \to x_3) \quad \dots \quad \mathtt{pref}(x_{k-1} \to x_k)}{\mathtt{pref}(x_1 \to x_k)}$$

where k is any positive integer. Think of the sequence of k premises as a chain from x_1 to x_k. In the case $k = 1$, the rule has no premises, and the conclusion is $x_1 \to x_1$, so that the rule is the inference-rule form of the removed axiom.

COROLLARY 2.2. *The same formulas are derivable in TPIL and TPIL*.*

The rational for introducing TPIL* will be given in §3.

2.3. Hilbertian calculi TPIL$_0$ and TPIL$_0$*. The calculus for TPIL$_0$ (resp. TPIL$_0$*) is the quote-free fragment of the calculus for TPIL (resp. TPIL*) obtained by removing $\mathtt{pref}$ and restricting the ranges of x, y, z to quote-free formulas.

COROLLARY 2.3. *The same formulas are derivable in TPIL$_0$ and TPIL$_0$*.*

2.4. Derivations. As usual, a *derivation* D of a formula x from a set Γ of hypotheses is a finite tree (or, more generally, a finite dag—directed acyclic graph—with a single source node, the root) where each node u is labeled with a formula $D(u)$. The root is labeled with x. If $v_1, v_2, \dots, v_n$ are the children of node u, then

$$\frac{D(v_1) \qquad \dots \qquad D(v_n)}{D(u)}$$

is an instance of an inference rule. Of course in our case, n is 1 or 2. The *length* of a derivation is the number of its nodes.

§3. Locality.

DEFINITION 3.1 (Local formulas). *Formulas* local *to a formula z are defined by induction. First, z is local to z. Second, for any binary connective $*$, if $\mathtt{pref}(x * y)$ is local to z then $\mathtt{pref}\, x$ and $\mathtt{pref}\, y$ are also local to z. A formula is* local *to a set Γ of formulas if it is local to some formula in Γ.*

If we remove the axiom $x \to x$ and the rule (trans) from TPIL, we obtain the Hilbert calculus for basic primal infon logic BPIL (Cotrini & Gurevich, 2013). The BPIL calculus has a locality property: Every formula y derivable from Γ can be derived from Γ using only formulas local to $\Gamma \cup \{y\}$. The locality property plays a key role in Cotrini & Gurevich (2013). Unfortunately TPIL does not have the locality property. For example, any derivation of $x \to w$ from $\{x \to y, y \to z, z \to w\}$ requires either $x \to z$ or $y \to w$. In this section, we prove that TPIL* has the locality property.

DEFINITION 3.2. *A derivation D_1 is* lighter *than D_2 if the number of instances of (trans*) if D_1 is less than that in D_2, or else the number of instances of (trans*) is the same in both derivations but the length of D_1 is less than that of D_2.*

LEMMA 3.3. *In a lightest derivation D of a formula φ from hypotheses Γ, the premises and conclusions of elimination rules are local to Γ.*

Proof. It suffices to prove this claim: For any instance of ($\wedge$e) or ($\rightarrow$e) in D, the premises are local to Γ. The claim is proved by induction on the length of the subderivation of the major premise. Note that, in the case of implication elimination, it suffices to prove that the major (the longer) premise is local to Γ.

The basis of induction is obvious: due to its form, the major premise isn't an axiom, and so is a hypothesis. In the induction step, the major premise is the conclusion of an instance of some inference rule R. If R is an elimination rule, use the induction hypotheses. The remaining rules are the three introduction rules ($\wedge$i), ($\vee$i), ($\rightarrow$i) and (trans*). We show that R cannot be any of these four rules.

Consider an instance

$$\dfrac{\dfrac{\vdots}{\mathtt{pref}\, x_1 \wedge x_2}}{\mathtt{pref}\, x_i}$$

of ($\wedge$e). Due to the form of the major premise, R cannot be ($\rightarrow$i), ($\vee$i) or (trans*). It cannot be ($\wedge$i) because

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathtt{pref}\, x_1} \quad \dfrac{\vdots}{\mathtt{pref}\, x_2}}{\mathtt{pref}\,(x_1 \wedge x_2)}}{\mathtt{pref}\, x_i}\ \text{can be shortened to}\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\, x_i}}{\vdots}$$

Next consider an instance

$$\dfrac{\dfrac{\vdots}{\mathtt{pref}\, x} \quad \dfrac{\vdots}{\mathtt{pref}\,(x \rightarrow y)}}{\mathtt{pref}\, y}$$

of ($\rightarrow$e). Due to the form of the major premise, R cannot be ($\wedge$i) or ($\vee$i). If R were ($\rightarrow$i) then D could be shorten in the obvious way. If R were the premiseless (trans*) then $x = y$ and

$$\dfrac{\dfrac{\vdots}{\mathtt{pref}\, x} \quad \dfrac{}{\mathtt{pref}\,(x \rightarrow x)}}{\mathtt{pref}\, x}\ \text{could be shorten to}\ \dfrac{\vdots}{\mathtt{pref}\, x}$$

Finally if R were a version of (trans*) with premises then

$$(\rightarrow\text{e})\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\, x_1} \quad (\text{trans}^*)\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\,(x_1 \rightarrow x_2)} \quad \cdots \quad \dfrac{\vdots}{\mathtt{pref}\,(x_{k-1} \rightarrow x_k)}}{\mathtt{pref}\,(x_1 \rightarrow x_k)}}{\mathtt{pref}\, x_k}$$

could be replaced with

$$(\to\text{e})\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\,x_1} \quad \dfrac{\vdots}{\mathtt{pref}\,(x_1 \to x_2)}}{(\to\text{e})\ \dfrac{\mathtt{pref}\,x_2 \quad \dfrac{\vdots}{\mathtt{pref}\,(x_2 \to x_3)}}{\mathtt{pref}\,x_3}}$$

$$(\to\text{e})\ \dfrac{\vdots \quad \vdots \quad \dfrac{\vdots}{\mathtt{pref}\,(x_{k-1} \to x_k)}}{(\to\text{e})\ \dfrac{\mathtt{pref}\,x_{k-1}}{\mathtt{pref}\,x_k}}$$

thus eliminating one instance of (trans*). □

LEMMA 3.4. *In a lightest derivation D of a formula φ from hypotheses Γ, for any instance*

$$\frac{\mathtt{pref}(x_1 \to x_2) \quad \mathtt{pref}(x_2 \to x_3) \quad \ldots \quad \mathtt{pref}(x_{k-1} \to x_k)}{\mathtt{pref}(x_1 \to x_k)}$$

of (trans), every premise $\mathtt{pref}(x_i \to x_{i+1})$ is local to Γ.*

Proof. Induction on the length of the subderivation of $\mathtt{pref}(x_i \to x_{i+1})$. The basis of induction is obvious: the premise is not an axiom so it should be a hypothesis.

Induction step. The premise $\mathtt{pref}(x_i \to x_{i+1})$ is the conclusion of an instance of some inference rule R. If R is an elimination rule, use Lemma 3.3. Due to the form of the premise, R is neither ($\wedge$i) nor ($\vee$i). If R were the premiseless (trans*) then $x_i = x_{i+1}$ and D could be shortened by removing $\mathtt{pref}(x_i \to x_{i+1})$. If R were a (trans*) with premises then

$$\frac{\mathtt{pref}(x_1 \to x_2) \quad \ldots \quad \dfrac{\mathtt{pref}(x_i \to y) \quad \ldots \quad \mathtt{pref}(z \to x_{i+1})}{\mathtt{pref}(x_i \to x_{i+1})} \quad \ldots \quad \mathtt{pref}(x_{k-1} \to x_k)}{\mathtt{pref}(x_1 \to x_k)}$$

could be shortened to

$$\frac{\mathtt{pref}(x_1 \to x_2) \quad \ldots \quad \mathtt{pref}(x_i \to y) \quad \ldots \quad \mathtt{pref}(z \to x_{i+1}) \quad \ldots \quad \mathtt{pref}(x_{k-1} \to x_k)}{\mathtt{pref}(x_1 \to x_k)}$$

Finally, if R were ($\to$i) then

$$(\text{trans*})\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\,(x_1 \to x_2)} \quad \ldots \quad (\to\text{i})\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\,x_i}}{\mathtt{pref}\,(x_{i-1} \to x_i)} \quad \ldots \quad \dfrac{\vdots}{\mathtt{pref}\,(x_{k-1} \to x_k)}}{\mathtt{pref}\,(x_1 \to x_k)}$$

could be replaced with

$$(\to\text{e})\ \dfrac{\dfrac{\vdots}{\mathtt{pref}\,x_i} \quad \dfrac{\vdots}{\mathtt{pref}\,(x_i \to x_{i+1})}}{\mathtt{pref}\,x_{i+1}}$$

$$(\to\text{i})\ \dfrac{(\to\text{e})\ \dfrac{\vdots \quad \vdots \quad \dfrac{\vdots}{\mathtt{pref}\,(x_{k-1} \to x_k)}}{(\to\text{e})\ \dfrac{\mathtt{pref}\,x_{k-1}}{\mathtt{pref}\,x_k}}}{\mathtt{pref}\,(x_1 \to x_k)}$$

thus eliminating one instance of (trans*). □

THEOREM 3.5 (Local-formula property for TPIL*). *In a lightest derivation D of φ from Γ, every node formula $D(u)$ is local to $\Gamma \cup \{\varphi\}$.*

Proof. Clearly φ is local to $\Gamma \cup \{\varphi\}$. Now suppose that we have proved x to be local to $\Gamma \cup \{\varphi\}$ and let x' be a premise of x in D, according to an inference rule R. If R is an introduction rule then obviously x' is local to $\Gamma \cup \{\varphi\}$. If R is an elimination rule, use Lemma 3.3. If R is (trans*), use Lemma 3.4. □

COROLLARY 3.6 (Subformula property for TPIL$_0$*). *In a lightest derivation D of φ from Γ in TPIL$_0$*, every node formula $D(u)$ is a subformula of a hypothesis.*

Proof. The formulas local to a quote-free formula z are subformulas of z. □

§4. Semantics.

DEFINITION 4.1. *A formula x is an* offshoot *of a set Δ of formulas if x is a subformula of Δ or else $x = (x_1 \to x_2)$ where x_1, x_2 are subformulas of Δ.*

4.1. The quote-free case. In this subsection, we restrict attention to quote-free formulas. Semantics is simpler in that case. We adopt (and adapt to our purposes) the notion of valuation from Beklemishev *et al.* (in preparation).

DEFINITION 4.2 (Valuations). *A* valuation *is a Boolean-valued function υ on formulas such that $\upsilon(x \to z) = 1$ when $\upsilon(x \to y) = \upsilon(y \to z) = 1$.* □

DEFINITION 4.3 (Holds under). *A relation $\models_\upsilon x$ (formula x holds under valuation υ) is defined by induction on x.*

- $\models_\upsilon \top$.
- *If x is atomic then $\models_\upsilon x$ if and only if*[2] $\upsilon(x) = 1$.
- $\models_\upsilon x_1 \wedge x_2$ *if and only if* $\models_\upsilon x_1$ *and* $\models_\upsilon x_2$.
- $\models_\upsilon x_1 \vee x_2$ *if and only if* $\models_\upsilon x_1$ *or* $\models_\upsilon x_2$ *or* $\upsilon(x_1 \vee x_2) = 1$.
- $\models_\upsilon x \to x$.
- *If $x \neq y$, three cases arise:*
  - — *If* $\models_\upsilon y$ *then* $\models_\upsilon x \to y$.
  - — *If* $\models_\upsilon x$ *but* $\not\models_\upsilon y$ *then* $\not\models_\upsilon x \to y$.
  - — *If* $\not\models_\upsilon x$ *and* $\not\models_\upsilon y$ *then* $\models_\upsilon x \to y$ *if and only if* $\upsilon(x \to y)$.

Further, let Γ be a set of formulas. Define $\models_\upsilon \Gamma$ (Γ holds under valuation υ) if and only if every formula in Γ holds under υ. □

THEOREM 4.4 (Quotation-free soundness and completeness). *Let y be a quote-free formula and Γ a set of quote-free formulas. The following claims are equivalent.*

(1) $\Gamma \vdash y$ *in TPIL$_0$.*

(2) $\Gamma \vdash y$ *in TPIL$_0$*.*

(3) For any valuation υ, if $\models_\upsilon \Gamma$ then $\models_\upsilon y$.

---

[2] Often relations are defined by specifying explicitly the cases where the relation holds; it is implicit that in all other cases the relation fails. Here and in Definition 4.7 below we need to specify explicitly both, positive and negative cases; hence the use of "if and only if" rather than just "if."

Proof. By Corollary 2.3, claims 1 and 2 are equivalent. We prove that claims 1 and 3 are equivalent.

$1 \Rightarrow 3$. Let D be a derivation of y from Γ, and assume that Γ holds under a valuation v. We prove that $\models_v y$ by induction on the length of D. The base case, where y is the axiom or a hypothesis, is obvious. Suppose then that y is obtained in D by inference rule R. We consider only the case where R is

$$\text{(trans)}\ \frac{x_1 \to x_2 \qquad x_2 \to x_3}{x_1 \to x_3}$$

The other cases are simpler. By the induction hypothesis, $\models_v x_1 \to x_2$ and $\models_v x_2 \to x_3$.

First suppose that $\not\models_v x_i$ for all $i \leq 3$. Then $v(x_1 \to x_2) = v(x_2 \to x_3) = 1$ which entails $v(x_1 \to x_3) = 1$. But $v(x_1 \to x_3) = 1$ and $\not\models_v x_1$ entail $\models_v x_1 \to x_3$.

Second suppose $\models_v x_i$ for some $i \leq 3$. $\models_v x_3$ entails $\models_v x_1 \to x_3$ immediately. Taking the induction hypothesis into account, $\models_v x_2$ entails $\models_v x_3$ which entails $\models_v x_1 \to x_3$. Similarly $\models_v x_1$ entails $\models_v x_2$ which entails $\models_v x_3$ which entails $\models_v x_1 \to x_3$.

$3 \Rightarrow 1$. Assume that y is not derivable from Γ and let v be the valuation such that $v(x) = 1$ if and only if x is an offshoot of $\Gamma \cup \{y\}$ and $\Gamma \vdash x$ in TPIL_0. It is easy to see that v is indeed a valuation.

LEMMA 4.5. *For any offshoot x of $\Gamma \cup \{y\}$, $\models_v x$ if and only if $\Gamma \vdash x$.*

Proof of the lemma. Induction on x. The cases where x is atomic or an axiom or a conjunction are obvious.

If $x = x_1 \vee x_2$, consider two cases. If $\models_v x_i$ for some i, then (i) by the definition of $\models$, we have $\models_v x$ and (ii) by the induction hypothesis, we have $\Gamma \vdash x_i$ and therefore $\Gamma \vdash x$. If $\not\models_v x_i$ for $i = 1, 2$, then $\models_v x \Leftrightarrow v(x) = 1$ by the definition of $\models$, and $v(x) = 1 \Leftrightarrow \Gamma \vdash x$ by the definition of v.

If $x = x_1 \to x_2$ and $x_1 \neq x_2$, consider the three cases in the definition of $\models_v x$. If $\models_v x_2$, then we have $\models_v x$ and $\Gamma \vdash x$. If $\models_v x_1$ but $\not\models_v x_2$ then (i) by the definition of $\models$, we have $\not\models_v x$, and (ii) by the induction hypothesis, we have $\Gamma \vdash x_1$ and $\Gamma \nvdash x_2$ which gives rise to $\Gamma \nvdash x$. If $\not\models_v x_1$ and $\not\models_v x_2$, then $\models_v x \Leftrightarrow v(x) = 1$ by the definition of $\models$, and $v(x) = 1 \Leftrightarrow \Gamma \vdash x$ by the definition of v. □

Since $\Gamma \nvdash y$, it follows that $\models_v \Gamma$ but $\not\models_v y$. □

Valuations can be seen as single-world Kripke models.

4.2. The general case.

DEFINITION 4.6 (Kripke models). *A* Kripke model *for transitive primal infon logic is a structure M such that*

- *the vocabulary of M consists of (i) binary relations S_q where q ranges over principal constants and (ii) unary relations V_x where x ranges over formulas, and*
- *M satisfies the constraint $V_{x\to y} \cap V_{y\to z} \subseteq V_{x\to z}$.* □

DEFINITION 4.7 (Holds in). *Let w range over the worlds (that is elements) of a given Kripke model. By induction on formula x, we define relation $w \models x$ (x holds in w).*

(1) $\top$ *holds in w.*

(2) *A variable x holds in w if and only if $w \in V_x$.*

(3) *$y \wedge z$ holds in w if and only if $w \models y$ and $w \models z$.*

(4) *$y \vee z$ holds in w if and only if $w \vDash y$ or $w \vDash z$ or $w \in V_{y \vee z}$.*

(5) *$y \to y$ holds in w.*

(6) *Suppose that y, z are distinct formulas. To define when $y \to z$ holds in w, we consider three cases.*

  (a) If $w \vDash y$ then $w \vDash x \to y$.

  (b) If $w \vDash x$ but $w \nvDash y$ then $w \nvDash x \to y$.

  (c) $w \nvDash x$ and $w \nvDash y$ then $w \vDash x \to y$ if and only if $w \in V_{x \to y}$.

(7) *q `said` y holds in w if and only if $w \vDash y$ for every w' with $w\ S_q\ w'$.*

A set of formulas Γ holds in w if every formula in Γ holds there.

DEFINITION 4.8 (Local prefixes). *A quotation prefix* `pref` *is* local *to a set Γ of formulas if some formula* `pref` x *is local to Γ.* □

THEOREM 4.9 (Soundness and completeness). *Let y be a formula and Γ a set of formulas. The following claims are equivalent.*

(1) $\Gamma \vdash y$ in TPIL.

(2) $\Gamma \vdash y$ in TPIL.*

(3) For every Kripke model M, y holds in every world of M where Γ holds.

Proof. By Corollary 2.2, claims 1 and 2 are equivalent.

$1 \Rightarrow 3$. Assume that Γ holds in a world w of a given Kripke model. We prove $w \vDash y$ by induction on the length of a given derivation of y from Γ. The case $y \in \Gamma$ is obvious. To prove $w \vDash \texttt{pref}\ \top$ (resp. $w \vDash \texttt{pref}\,(x \to x)$), induct on `pref`. Suppose then that y is obtained via an inference rule R. Several cases arise, all straightforward. We consider only the case where R is the rule

$$(\to\text{e})\ \frac{\texttt{pref}\,x \qquad \texttt{pref}(x \to z)}{\texttt{pref}\,z}$$

By the induction hypothesis, $w \vDash \texttt{pref}\,x$ and $w \vDash \texttt{pref}(x \to z)$. Without loss of generality x and z are distinct formulas. We prove that $w \vDash \texttt{pref}\,z$ by auxiliary induction on the length of `pref`. If `pref` is empty then we have case (6a) of Definition 4.7 and so $w \vDash z$. Otherwise $\texttt{pref} = q\ \texttt{said}\ \texttt{pref}'$. Consider any world w' with $w\ S_q\ w'$. Since $w \vDash \texttt{pref}\,x$, we have $w' \vDash \texttt{pref}'\,x$. Similarly $w' \vDash \texttt{pref}'\,(x \to z)$. By the auxiliary induction hypothesis $w' \vDash \texttt{pref}'\,z$. By Definition 4.7, $w \vDash \texttt{pref}\,z$.

$3 \Rightarrow 1$. We consider only those formulas that are offshoots of $\Gamma \cup \{y\}$. Assume that y is not derivable from Γ and let M be the structure where

- The elements are the quotation prefixes local to $\Gamma \cup \{y\}$,
- $\texttt{pref}\ S_q\ \texttt{pref}'$ if $\texttt{pref}' = q\ \texttt{said}\ \texttt{pref}$,
- $V_x = \{\texttt{pref} : \Gamma \vdash \texttt{pref}\,x\}$.

If $\texttt{pref} \in V_{x_1 \to x_2} \cap V_{x_2 \to x_3}$, then $\Gamma \vdash \texttt{pref}(x_1 \to x_2)$ and $\Gamma \vdash \texttt{pref}(x_2 \to x_3)$, therefore $\Gamma \vdash \texttt{pref}(x_1 \to x_3)$, therefore $\texttt{pref} \in V_{x_1 \to x_3}$; so M is a Kripke model. We will check that Γ holds in ϵ, the empty-prefix world, but y does not.

LEMMA 4.10. *`pref` $\vDash x$ if and only if $\Gamma \vdash$ `pref` x.*

Proof of the lemma. Induction on x. The cases where x is atomic, an axiom or a conjunction are straightforward.

Suppose $x = x_1 \vee x_2$. If $\mathtt{pref} \vDash x_i$ for some i, then $\mathtt{pref} \vDash x$, $\Gamma \vdash x_i$ and $\Gamma \vdash \mathtt{pref}\, x$. If $\mathtt{pref} \nvDash x_i$ for $i = 1, 2$, then $\mathtt{pref} \vDash x$ if and only if $\mathtt{pref} \in V_x$ only if $\Gamma \vdash \mathtt{pref}\, x$.

Suppose $x = x_1 \to x_2$ where x_1, x_2 are distinct. If $\mathtt{pref} \vDash x_2$, then $\mathtt{pref} \vDash x$, $\Gamma \vdash x_2$ and $\Gamma \vdash \mathtt{pref}\, x$. If $\mathtt{pref} \vDash x_1$ but $\mathtt{pref} \nvDash x_2$, then $\mathtt{pref} \nvDash x$ and $\Gamma \nvdash \mathtt{pref}\, x$. If $\mathtt{pref} \nvDash x_1$ and $\mathtt{pref} \nvDash x_2$, then $\mathtt{pref} \vDash x$ if and only if $\mathtt{pref} \in V_x$ if and only if $\Gamma \vdash \mathtt{pref}\, x$.

Finally suppose that $x = q\ \mathtt{said}\, x'$. We have

$$\begin{aligned}
&\mathtt{pref} \vDash q\ \mathtt{said}\, x' \Leftrightarrow \\
&\mathtt{pref}\ q\ said \vDash x' \Leftrightarrow \quad \text{(by the induction hypothesis)} \\
&\Gamma \vdash (\mathtt{pref}\ q\ said) x' \Leftrightarrow \\
&\Gamma \vdash \mathtt{pref}\ (q\ said\ x')
\end{aligned}$$

□

It follows that $\epsilon \vDash \Gamma$, but $\epsilon \nvDash y$. This completes the proof of the theorem. □

DEFINITION 4.11 (Width of a formula). *By induction on a formula x, we define its* width $|x|$. *If x is a variable then $|x| = 1$; if $x = x_1 * x_2$, where $*$ is any binary connective, then $|x| = |x_1| + |x_2| + 1$, and if $x = q\ \mathtt{said}\, x_1$ then $|x| = |x_1| + 2$. For a set Δ of formulas, the* width $|\Delta| = \sum\{|x| : x \in \Delta\}$.

THEOREM 4.12 (Small models). *If $\Gamma \nvdash y$ then there is a counterexample Kripke model of size $\leq |\Gamma \cup \{y\}|/2$.*

Proof. We start with an auxiliary lemma.

LEMMA 4.13. *The number $LP(x)$ of prefixes local to a formula x is less than $|x|/2$.*

Proof of the lemma. An easy induction on x. For the case $x = q\ \mathtt{said}\, x'$, note that the nonempty prefixes local to x are $q\ \mathtt{said}$ plus all the prefixes of the form $q\ \mathtt{said}\ \mathtt{pref}$, where $\mathtt{pref}$ is nonempty and local to x'. Hence,

$$\begin{aligned}
LP(q\ \mathtt{said}\, x') = 1 + LP(x') &< 1 + |x'|/2 = (2 + |x'|)/2 \\
&= |q\ \mathtt{said}\, x'|/2.
\end{aligned}$$

□

Now, consider the Kripke model M built in the proof of Theorem 4.9. Recall that the worlds of M are all prefixes local to $\Gamma \cup \{y\}$. The previous lemma implies that the number of nonempty prefixes local to a set of formulas Δ is less than $|\Delta|/2$; so the number of prefixes local to Δ is less than $1 + |\Delta|/2$. Hence, the size of M is less than $1 + |\Gamma \cup \{y\}|/2$. □

REMARK 4.14 (Possible worlds). *The original definition of Kripke models for primal logic in Gurevich & Neeman (2011) contained a partial order on the worlds. We simplified the definition because the partial order is not needed for the soundness and completeness theorem. But, in the intended applications, a partial order on the worlds may reflect possible developments. In a world w_1, Bob is proposing Alice to be a Facebook friend of his but she isn't his friend in w_1. However, she is a Facebook friend of his in some world $w_2 > w_1$. Incorporating a partial order $\leq$ on the worlds imposes the following constraints:*

- *If $u \leq v$ and $v\ S_q\ w$ then $u\ S_q\ w$.*
- *If $u \leq v$ and $u \in V_x$ then $v \in V_x$.* □

§5. Decision algorithm.

THEOREM 5.1 (Decision algorithm). *There is a quadratic time algorithm that, given two finite sequences of formulas, H (hypotheses) and Q (queries), decides which formulas in Q are derivable from H.*

This section is devoted to proving the theorem. We construct the desired decision algorithm.

Computation model. We use the standard computation model of the analysis of algorithms; see Cormen *et al*. (2001) for example. It is the random access machine such that (i) the registers are of size $O(\log n)$ where n is the size of the input and (ii) the basic register operations are constant-time. The main justification for that computation model is that the traditional uniprocessor computer can be viewed as a unit-cost random access machine. "In algorithms you use the unit-cost RAM model where basic register operations over $O(\log n)$ bit registers count as a single computation step. There are some good arguments for this: As technology improves for us to handle larger input sizes, the size of the registers tend to increase as well. For example, registers have grown from 8 to 64 bits on microprocessors over the past few decades" (Fortnow, 2009).

Syntax assumptions. We assume that the formal syntax of our formulas satisfies the following rather usual requirements.

- Formulas are strings in a fixed finite alphabet.
- Any occurrence of any subformula of a formula x is a contiguous segment of the string x.
- No two subformula occurrences in a formula x start at the same position of the string x. We will use the starting position of a subformula occurrence o as a key to identify o.
- There is a deterministic pushdown automaton that detects the initial position $\mathrm{Key}(o)$ of every subformula occurrence o and computes the length of the subformula in question.

The standard syntax of formulas with all binary operators in prefix position satisfies the requirements; no parentheses are required. The infix position for the binary operators is no problem; just put parentheses around every nonatomic subformula including the whole formula. We have been allowing ourselves to skip the outermost parentheses because they are not needed for human comprehension. But, formally, they are required.

Input. The input is a sequence H of the given hypotheses followed by some separator and then by a sequence Q of the queries; n is the length of the input. We presume that the input went through a lexical analyzer and so the names of the variables and constants are of length $O(\log n)$.

A formula or quotation prefix is *local* if it is local to H, Q. A local formula is *locally derivable* if it can be derived from H using only local formulas.

If J is a contiguous segment of the input then the initial position p of J is its *key*, symbolically $\mathrm{Key}(J) = p$. In particular, every formula occurrence in the input is uniquely identified by its key.

The stages of the decision algorithm. Our decision algorithm works in five stages.

(1) Parse the input and bind the nodes of the resulting parse tree to appropriate input positions.

(2) Construct a convenient data structure of the local quotation prefixes.

(3) Bind local formulas to nodes of the input parse tree.

(4) Construct additional data structures needed for fast derivation of local formulas.
(5) Derive the locally derivable formulas and output the derivable queries.

Stages 1–3 are also stages 1–3 the BPIL decision algorithm (Cotrini & Gurevich, 2013). To make this exposition self-contained, we summarize these three stages in §5.1. Stages 4 and 5 are described in §5.2 and 5.4 respectively. In §5.4 we prove that the algorithm is indeed quadratic-time, establish its correctness and remark on computing—in quadratic time—not only the derivable queries but also their derivations.

5.1. Parsing and auxiliary algorithms 1–3. Contrary to our derivation trees, which grow up in accordance with logic tradition, our parse trees grow down in accordance with computer science tradition. In particular, the root of a parse tree is at the top of the tree.

DEFINITION 5.2 (Formula parse tree (Bjørner *et al.*, 2012)). *By induction on formula x, we define the parse tree $PT(x)$ of x.*

- *If x is atomic, $PT(x)$ consists of one node labeled with x.*
- *If x is a quote $\texttt{pref}\, z$ with body z, then the root r of $PT(x)$ has a unique child r', the r'-rooted subtree is isomorphic to $PT(z)$, and the edge (r, r') is labeled with $\texttt{pref}$.*
- *Let x be a binary combination $x_1 * x_2$. Then the root r of $PT(x)$ is labeled with the binary connective $*$ and has two children, a left child r_1 and right child r_2. Let T_i be the r_i-rooted subtree of $PT(x)$, and let T_i' be the extension of T_i with r and the edge (r, r_i). Three cases arise.*

  (1) *If neither x_i is a quote then each T_i is isomorphic to $PT(x_i)$, and the edges (r, r_i) are unlabeled.*
  (2) *If x_i is a quote but x_j is not then T_j is isomorphic to $PT(x_j)$, the edge (r, r_j) is unlabeled, and T_i' is isomorphic to $PT(x_i)$.*
  (3) *If both x_i are quotes then each T_i' is isomorphic to $PT(x_i)$.*

We present some examples to clarify this definition. See Figures 1–4. Here, x and y are propositional variables.

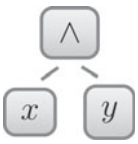

Fig. 1. Parse tree for $x \wedge y$.

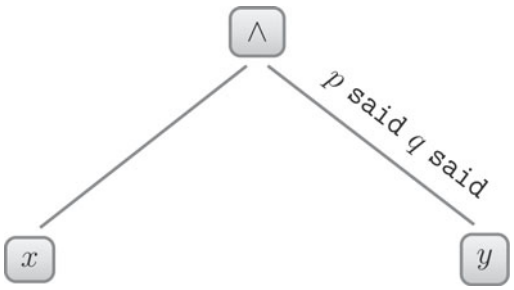

Fig. 2. Parse tree for $x \wedge (p\,\texttt{said}\,q\,\texttt{said}\,y)$.

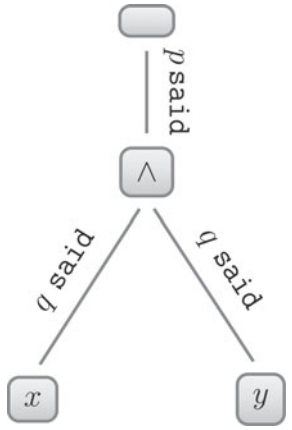

Fig. 3. Parse tree for $p\,\texttt{said}((q\,\texttt{said}\,x) \wedge (q\,\texttt{said}\,y))$.

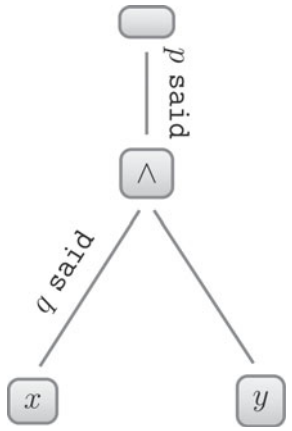

Fig. 4. Parse tree for $p\,\texttt{said}\ ((q\,\texttt{said}\,x) \wedge y)$.

DEFINITION 5.3 (Parse tree for the input). *The root of the input parse tree is labeled with* `input`. *The root has two children labeled* `hypothesis` *and* `query`. *The parse trees of the hypotheses hang under the* `hypothesis` *node in the order they occur in* H. *If the root node of a hypothesis* x *is unlabeled it is merged with the* `hypothesis` *node; otherwise the edge from the* `hypothesis` *node to the root of* x *is unlabeled. In a similar way, the parse of the queries hang under the* `query` *node.* □

Figure 5 shows the parse tree for input

$$H = \{p\,\texttt{said}\,x,\ p\,\texttt{said}\,(q\,\texttt{said}\,y \wedge r\,\texttt{said}\,s\,\texttt{said}\,x)\},$$
$$Q = \{p\,\texttt{said}\,(x \to (q\,\texttt{said}\,x \to x))\}.$$

DEFINITION 5.4 (Regular nodes and their body formulas). *A node* u *of the input parse tree is* regular *if* u *is labeled with an atomic formula or a binary connective; otherwise it is irregular. The* body formula $BF(u)$ *of a regular node* u *is the formula* x *such that the* u*-rooted subtree is isomorphic to the parse tree of* $BF(u)$. □

There are exactly three irregular nodes. These are the nodes labeled with `input`, `hypothesis` and `query`.

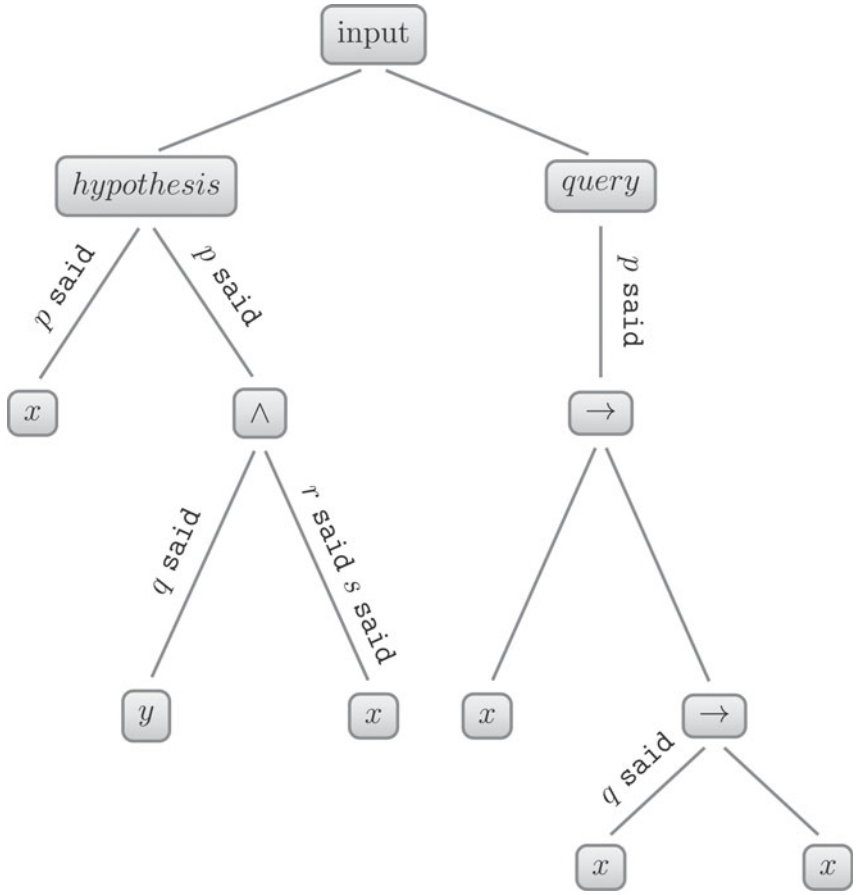

Fig. 5. Parse tree for an instance of the multiderivability problem.

DEFINITION 5.5. *Let x be a nonquote subformula of H, Q, let $o_1, \ldots, o_m$ be the occurrences of x in the input, and let $u_1, \ldots, u_m$ be the m nodes of the input parse tree with x as their body formula. If $o_1, \ldots, o_m$ are listed in order of their keys and $u_1, \ldots, u_m$ are listed in the depth-first order then each u_i* represents *the occurrence o_i of x, and the* key *of u_i is that of o_i.*

COROLLARY 5.6 (Corollary 22 in Cotrini & Gurevich, 2013). *There is a linear time algorithm*—Algorithm 1—*that, given an input in the form of a list of hypotheses and queries, builds the following.*

- *A parse tree for the input where every node u is decorated with the following additional fields.*
  - — *Key(u), the initial position of the occurrence of formula BF(u) represented by u.*
  - — *H(u), a pointer of type node but set to* `nil` *(to be used in stage 3 of the decision algorithm).*
  - — *Length(u), the length of BF(u).*
  - — *Vertex(u) (to be used in stage 2 of the decision algorithm).*

- *An array Node indexed by input positions. If p is the initial position of a subformula occurrence then Node$[p]$ is the node u with Key$(u) = p$. Otherwise Node$[p] =$* `nil`. □

DEFINITION 5.7 (Node prefixes). *For every node u of the input parse tree, Pref(u) is the prefix resulting from the concatenation the prefixes labeling the edges of the route from the root to u. In other words, if u is the root, then Pref$(u) = \epsilon$; otherwise Pref$(u) =$ Pref(v) Label(v, u) where v is the parent of u.*

THEOREM 5.8 (Theorem 25 in Cotrini & Gurevich, 2013). *There is a linear-time algorithm—*Algorithm 2*—that, given the output of* Algorithm 1*, builds an auxiliary data structure of the local prefixes in such a way that questions whether Pref$(u) =$ Pref(w) are decidable in constant time.*

Notation. If u is a node with exactly two children then u_l and u_r are the left and the right child of u respectively.

DEFINITION 5.9 (Complete node formulas CF(u)). *For every regular node u, CF$(u) =$ Pref(u)BF(u). In other words, if u is a leaf then CF$(u) =$ Pref(u)Label(u), and otherwise*

$$CF(u) = Pref(u)\big[\big(Label(u, u_l)(BF(u_l)\big) * \big(Label(u, u_r)BF(u_r)\big)\big].$$

DEFINITION 5.10 (Homonymy). *Let u, w range over the regular nodes of the input parse tree. If CF$(u) =$ CF(w) then u, w are* homonyms*.*

COROLLARY 5.11 (Corollary 29 in Cotrini & Gurevich, 2013). *Every CF(u) is a local formula, and every local formula is the complete node formula CF(u) for some node u.*

THEOREM 5.12 (Theorem 30 in Cotrini & Gurevich, 2013). *There is a linear-time algorithm—*Algorithm *3—that, given the input sequence of hypotheses and queries and the outputs of* Algorithm 1 *and* Algorithm 2*,*

- *computes a particular node, the* homonymy leader*, in every homonymy class of regular nodes, and*
- *sets the pointer $H(u)$ to the homonymy leader of u, for every regular node u.*

Stages 1–3. *Algorithms 1, 2, and 3 are stages 1, 2, 3 respectively of the decision algorithm.*

5.2. Preprocessing. At its fourth stage the decision algorithm constructs a table T to be used on the final fifth stage. The intended meaning of the table will become clear in the next subsection.

Description of table T. For every homonymy leader u, the entry $T(u)$ is a record with the following fields where $*$ ranges over the binary connectives. Let CF$(u) =$ `pref` x where x may be a quote.

- $(*$, left): A list of all homonymy leaders w such that CF$(w) =$ `pref` $(x * y)$ for some y.
- $(*$, right): A list of all homonymy leaders w such that CF$(w) =$ `pref` $(y * x)$ for some y.
- A numeric field Status(u) takes values 1, 2, and 3. Contrary to the previous fields which stay unchanged during Stage 5, the status of a node may change on that stage.

The intended meaning of the status. Formally Status(u) is the status of a given homonymy leader u but intentionally it is the status of the formula CF(u).

- Status(u) $= 1$ indicates that CF(u) has not been derived yet. In such a case we say that u is *raw*.
- Status(u) $= 2$ indicates that CF(u) has been derived but not processed in the sense explained in the next section. In such a case we say that u is *pending*.
- Status(u) $= 3$ indicates that CF(u) has been derived and processed. In such a case we say that u is *processed*.

In Stage 4, the status of every homonymy leader u is initialized to 1 unless u represents an axiom or hypothesis in which case the status of u is initialized to 2. Recall that u_l and u_r are the left and the right child of a node u provided that u has exactly two children.

Stage 4. *The algorithm traverses the input parse tree in the depth-first order and constructs the table T. On the same occasion, it constructs a queue, called the* pending queue*, and initializes it with the axioms and hypotheses.*

- *If the label of u is a binary connective $*$ and if $H(u) = u$ (so that u is a homonymy leader) then append u to the $(*$, left) field of $T(H(u_l))$ and to the $(*$, right) field of $T(H(u_r))$.*
- *If*
  - *Label(u) $= \top$,*
  - *u is a child of the* `hypothesis` *node or*
  - *u is labeled with $\rightarrow$ and $H(u_l) = H(u_r)$,*

  and if Status($H(u)$) $= 1$, then append u to the pending queue and set Status(u) $= 2$.

The one-node computation is constant-time. Therefore, Stage 4 takes linear time.

5.3. Processing. At stage 5 the decision algorithm derives all the locally derivable formulas. The idea is to repeat the following procedure until there are no pending nodes: Pick the first pending node u and apply all derivation rules to CF(u), which may cause some raw formulas to become pending, and then remove u from the pending queue and set Status(u) $= 3$. The following invariant is maintained: if a homonymy leader u is pending or processed then CF(u) is derivable.

But what does it mean to apply a derivation rule to the formula CF(u)? We explain that. So let u be a homonymy leader. For brevity we say "make a raw homonymy leader w pending" to means this: append w to the pending queue and set Status(w) $= 2$. As before u_l and u_r are the left and the right child of a node u provided that u has exactly two children.

Applying ($\wedge e$) to CF(u).

If u is labeled with $\wedge$ do the following; otherwise do nothing.
If $H(u_l)$ is raw, make it pending. If $H(u_r)$ is raw, make it pending.

Justification If u is labeled with $\wedge$ then CF(u) $=$ `pref` $(x \wedge y)$, CF(u_l) $=$ `pref` x and CF(u_r) $=$ `pref` y. Since u is pending, `pref` $(x \wedge y)$ is derivable. By rule ($\wedge$e), `pref` x and `pref` y are derivable.

Applying ($\wedge i$) to CF(u).

(1) Walk through the nodes w in the ($\wedge$,left) field of $T(u)$. If w is raw and if $H(w_r)$ is pending or processed then make w pending.

Justification. Suppose CF(u) = $\mathtt{pref}\,x$. The ($\wedge$,left) field of $T(u)$ comprises homonymy leaders w such that CF(w) = $\mathtt{pref}\,(x \wedge y)$ for some y. It follows that CF(w_r) = $\mathtt{pref}\,y$. Since u is pending, $\mathtt{pref}\,x$ is derivable. If $H(w_r)$ is pending or processed, then $\mathtt{pref}\,y$ is derivable as well, and then — by the rule ($\wedge$i) — $\mathtt{pref}\,(x \wedge y)$ is derivable.

(2) Similarly, walk through the nodes w in the ($\wedge$,right) field of $T(u)$. If w is raw and if $H(w_l)$ is pending or processed then make w pending.

Applying ($\vee$i) to CF(u).

Walk through the ($\vee$,left) and ($\vee$,right) fields of $T(u)$ and make pending each raw node w there.

Justification. Let CF(u) = $\mathtt{pref}\,x$. The ($\vee$,left) list comprises homonymy leaders w with CF(w) = $\mathtt{pref}\,(x \vee y)$ for some y. Similarly the ($\vee$,right) list comprises homonymy leaders w with CF(w) = $\mathtt{pref}\,(y \vee x)$ for some y. Since u is pending, $\mathtt{pref}\,x$ is derivable. By rule ($\vee$i), $\mathtt{pref}\,(x \vee y)$ and $\mathtt{pref}\,(y \vee x)$ are derivable.

Applying ($\rightarrow$i) to CF(u).

Walk through the ($\rightarrow$, right) field of $T(u)$ and make pending each raw node w there.

Justification. Let CF(u) = $\mathtt{pref}\,x$. The ($\rightarrow$,right) list comprises homonymy leaders w such that CF(w) = $\mathtt{pref}\,(y \rightarrow x)$ for some y. Since $\mathtt{pref}\,x$ is derivable, so is every $\mathtt{pref}\,(y \rightarrow x)$.

Applying ($\rightarrow$e) to CF(u).

CF(u) can be used as the left or the right premise of the rule ($\rightarrow$e). Accordingly, we have two cases.

(1) Walk through the ($\rightarrow$,left) list of $T(u)$. For each node w there, if w is pending or processed but $H(w_r)$ is raw, then make $H(w_r)$ pending.

Justification. Let CF(u) = $\mathtt{pref}\,x$. The ($\rightarrow$,left) field of $T(u)$ comprises homonymy leaders w such that CF(w) = $\mathtt{pref}\,(x \rightarrow y)$ for some y. Then CF(w_r) = $\mathtt{pref}\,y$. Since u is pending, $\mathtt{pref}\,x$ is derivable. If w is pending or processed then $\mathtt{pref}\,(x \rightarrow y)$ is also derivable, and then — by the rule ($\rightarrow$e) — $\mathtt{pref}\,y$ is derivable.

(2) If u is labeled with $\rightarrow$ and if $H(u_l)$ is pending or processed but $H(u_r)$ is raw, then make $H(u_r)$ pending.

Justification. Suppose that CF(u) = $\mathtt{pref}\,(x \rightarrow y)$, so that CF($u_l$) = $\mathtt{pref}\,x$ and CF(u_r) = $\mathtt{pref}\,y$. Since u is pending, $\mathtt{pref}\,(x \rightarrow y)$ is derivable. If $\mathtt{pref}\,x$ is also derivable then, by the rule ($\rightarrow$ e), $\mathtt{pref}\,y$ is derivable.

Toward applying rule (trans) to CF(u).*

The case of (trans*) is more complicated. We need an auxiliary result. Consider a state S of the decision algorithm, and let v_1, v_2 be homonymy leaders with CF(v_1) = $\mathtt{pref}\,x$ and CF(v_2) = $\mathtt{pref}\,y$. We say that v_1 $\mathtt{pred}$ v_2 modulo $\mathtt{pref}$ at S if the formula $\mathtt{pref}\,x \rightarrow \mathtt{pref}\,y$ is local and the homonymy leader w with CF(w) = $\mathtt{pref}\,(x \rightarrow y)$ is pending

or processed at S. Further $\texttt{pred}^*$ is the reflexive, transitive closure of the binary relation pred modulo pref at S on the homonymy leaders.

LEMMA 5.13. *There is a linear time (linear in the input size n) algorithm that, given any* pref, *S and any homonymy leader* v_2 *with* CF(v_2) *of the form* pref y, *marks all homonymy leaders* v_1 *such that* v_1 $\texttt{pred}^*$ v_2 *modulo* pref *at S.*

Proof. We build upon a well-known algorithm that (i) given a directed graph and a vertex v_2, marks every vertex v with a path from v to v_2 and (ii) works in time linear in the number of edges. In our case, the vertices are the homonymy leaders and the edges are the relationships v pred w modulo pref at S. Define the distance $d(v, w)$ to be the number of edges in the shortest path from v to w; if there is no path from v to w then $d(v, w) = \infty$. Now we describe the desired algorithm (skipping some book-keeping details).

Start by marking v_2; this is round 0. Then walk through the ($\rightarrow$, left) field of the record $T(v_2)$. The field comprises all homonymy leaders w such that $\text{CF}(w) = \texttt{pref}\,(x \rightarrow y)$ and therefore $\text{CF}(w_l) = \texttt{pref}\,x$ for some x. For every w with $\text{Status}(w) \geq 2$, mark the homonymy leader $H(w_l)$ unless it is already marked. This completes round 1, and it takes care of the homonymy leaders v with $d(v, v_2) = 1$. For every v with $d(v, v_2) = 1$, walk through the ($\rightarrow$, left) field of the record $T(v)$, and for every w there with $\text{Status}(w) = 2$, mark $H(w_l)$ unless it is already marked. This is round 2, and it takes care of the homonymy leaders v with $d(v, v_2) \leq 2$. And so on. Stop when a round produces no new marking.

Note that every edge v pred w in our graph is examined at most once, and the examination is constant time. The edges corresponds to local formulas, and different edges correspond to different local formulas. And the number of local formulas is $\leq n$. Hence the algorithm is linear in n. □

Similarly, there is a linear time algorithm that, given any pref, S and any homonymy leader v_1 with CF(v_2) of the form pref x, marks all homonymy leaders v_2 such that v_1 $\texttt{pred}^*$ v_2 modulo pref at S.

Applying (trans) to CF(u)* If $\text{CF}(u) = \texttt{pref}(x \rightarrow y)$ for distinct x, y then do the following.

- Mark "left" all nodes v_1 such that v_1 $\texttt{pred}^*$ u_l.
- Mark "right" all nodes v_2 such that u_r $\texttt{pred}^*$ v_2.
- Traverse the input tree and make pending every raw homonymy leader v such that $\text{CF}(v) = \texttt{pref}(x' \rightarrow y')$ for some x', y' and $H(v_l)$ is marked "left" and $H(v_r)$ is marked "r".
- Traverse the input tree and remove all marks "left" and "right."

Justification. Suppose $\text{CF}(u) = \texttt{pref}(x \rightarrow y)$ and $\text{CF}(v) = \texttt{pref}(x' \rightarrow y')$ and the homonymy leaders $H(v_l)$, $H(v_r)$ are marked "left" and "right" respectively. Then we have

$$H(v_l)\ \texttt{pred}^*\ u_l\ \texttt{pred}\ u_r\ \texttt{pred}^*\ H(v_r)$$

modulo pref at S. Thus there is a chain of homonymy nodes $w_1, w_2, \ldots, w_k$ such that

$$H(v_l) = w_1\ \texttt{pred}\ w_2\ \ldots\ \texttt{pred}\ w_k = H(v_r)$$

modulo pref at S. By the definition of pred modulo pref at S, there are formulas $z_1, z_2, \ldots, z_k$ such that every $\text{CF}(w_i) = \texttt{pref}\,z_i$ and every formula $\texttt{pref}\,(z_i \rightarrow z_{i+1})$ is locally derivable. By (trans*), $\texttt{pref}\,(z_1 \rightarrow z_k)$ is locally derivable. But $\text{CF}(v) = \texttt{pref}\,(z_1 \rightarrow z_k)$.

Stage 5. *While the pending queue is not empty, the algorithm takes the first node u of the queue and applies all the inference rules as explained above. When the pending queue is empty, the algorithm compiles a list of derivable queries by walking through the children of the* `query` *node and observing their status.*

That concludes the construction of the decision algorithm.

5.4. Analysis and correctness.

THEOREM 5.14. *The decision algorithm works in quadratic time.*

Proof. We already mentioned that , according to Cotrini & Gurevich (2013), stages 1–3 take linear time. Clearly, stages 4 takes linear time. For every rule R different from (trans*), the cumulative time of applying R to all formulas CF(u) is linear; this is as in Cotrini & Gurevich (2013). The reason is the same in all cases. Consider, for example the rule ($\wedge$i). To apply the rule to CF(u), we examine every node w in the ($\wedge$, left) and ($\wedge$, right) fields of the record $T(u)$ of a homonymy node u. Every such w has the form $\texttt{pref}(x \wedge y)$. But every particular w of that form appears in exactly one field ($\wedge$, left), namely in the record $T(H(w_l))$, and in exactly one field ($\wedge$, right), namely of the record $T(H(w_r))$; so this w will be examined at most twice.

On the other hand, one application of (trans*) takes time $O(n)$, and there are $O(n)$ such applications. So the cumulative time of applying (trans*) is quadratic. Compiling the list of derivable queries takes linear time, so the whole decision algorithm takes quadratic time. □

THEOREM 5.15. *The decision algorithm is sound (so that every query deemed derivable by the algorithm is indeed derivable) and complete (so that every derivable query is deemed derivable by the algorithm).*

Proof. Soundness follows from the justifications provided in §5.3, so it remains to prove completeness. By Corollary 5.11, every complete node formula CF(u) is local, and every local formula is CF(u) for some node u. Thus it suffices to prove the claim that, for every homonymy leader u, if CF(u) is locally derivable, then u is pending at some state of the decision algorithm, and thus u is pending or processed at the later stages. We prove the claim by induction on a given local derivation of CF(u).

If CF(u) is an axiom or a hypothesis, then u becomes pending at stage 4. So suppose that CF(u) is the conclusion of some inference rule R. Several cases arise.

- R is ($\wedge$i). The derivation of CF(u) looks like this:

$$\dfrac{\dfrac{\begin{matrix}H\\ \vdots\\ \texttt{pref}\,x\end{matrix}} \qquad \dfrac{\begin{matrix}H\\ \vdots\\ \texttt{pref}\,y\end{matrix}}}{\texttt{pref}\,(x \wedge y)}$$

  Thus $\text{CF}(u_l) = \texttt{pref}\,x$ and $\text{CF}(u_r) = \texttt{pref}\,y$. By the induction hypothesis $H(u_l)$ and $H(u_r)$ are pending at some states of the decision algorithm. Without loss of generality we may assume that $H(u_r)$ is processed earlier than $H(u_l)$. When we apply ($\wedge$i) to $H(u_l)$, we walk through the nodes in the ($\wedge$,left) list of u_l and find u there. By that time $H(u_r)$ is processed. If u is raw, we make it pending.
  All other cases are similar to or easier than that of ($\wedge$i) with the exception of the case of (trans*) with multiple premises.

- R is (trans*) with multiple premises. The derivation of CF(u) looks like this:

$$\dfrac{\dfrac{H}{\vdots}\quad \dfrac{H}{\vdots}\quad \dfrac{H}{\vdots}}{\texttt{pref}(x_1 \to x_2)\quad \ldots \qquad \texttt{pref}(x_i \to x_{i+1}) \qquad \ldots\quad \texttt{pref}(x_{k-1} \to x_k)}{\texttt{pref}(x_1 \to x_k)}$$

There are homonymy leaders u_i, v_i such that CF(u_i) = $\texttt{pref}(x_i)$ and CF(v_i) = $\texttt{pref}(x_i \to x_{i+1})$ for $i = 1, \ldots, k-1$. By the induction hypothesis, every CF(v_i) is pending at some state of the decision algorithm. Let CF(v_j) be processed last among the formulas CF(v_i). When CF(v_j) is processed, all other formulas CF(v_i) are processed, so that u_i $\texttt{pred}$ u_{i+1} for all $i = 1, \ldots, k-1$ modulo $\texttt{pref}$. When we apply (trans*) to CF(v_j), we have

$$u_1\, \texttt{pred}^*\, u_j\, \texttt{pred}\, u_{j+1}\, \texttt{pred}^* u_k$$

modulo $\texttt{pref}$. If u is raw at this point, we make it pending. □

COROLLARY 5.16 (Witness extraction). *The decision algorithm can be extended to extract, in quadratic time, a witness that the queries deemed to be locally derivable are indeed derivable from the given hypotheses and that the remaining queries are not derivable from the given hypotheses.*

Proof. The desired derivation dag (directed acyclic graph) D is constructed as follows. As a node u becomes pending, put it on the derivation tree with pointers to the nodes representing the premises, if any, of the derivation rule used to make u pending.

Let H and Q be as above. For every query $y \in Q$ that is claimed to be derived, derivation D includes a derivation of y from H; just consider the subtree of the derivation tree D rooted at a homonymy leader representing y. Derivation D also witnesses that the remaining queries in Q are not derivable. Indeed, consider the collection LD (an allusion to "locally derivable") of the formulas (labeling the nodes) in D. D is closed in the following sense. Consider any application of any derivation rule such that the premises belong to LD and the conclusion is local to H, Q; the conclusion already belongs to LD. □

§6. Related work. We already mentioned, in the introduction, the related work on primal logic. Here we consider other related work.

The unary connectives "p $\texttt{said}$" of primal logic can be viewed as necessity operators. Thus primal logic and transitive primal logic are multimodal extensions of the primal fragment propositional intuitionistic logic. We refer the reader to Wolter & Zakharyaschev (1999) for a presentation of intuitionistic modal logic and to Chapter 1 of Gabbay *et al.* (2003) for a presentation of multimodal logics.

Recall that the derivability problem for a logic is the problem of deciding whether a given formula is derivable from a given set of formulas, and the validity problem is the problem of deciding whether a given formula is valid. Clearly, the second is a particular case of the first, and these two problems are the same when the deduction theorem holds for the logic.

There seems to be very few known natural fragments of intuitionistic logic, let alone its modal extensions, with polynomial-time decidable validity problem. First, we mention some loosely related tractability results on modal and description logics. Halpern proved

that the validity problem for $\mathbf{K}_n$, $\mathbf{T}_n$, $\mathbf{S4}_n$ and $\mathbf{K45}_n$, $\mathbf{KD45}_n$, $\mathbf{S5}_n$ can be decided in linear time if the nesting of modal operators and propositional variables is restricted (Halpern, 1995). See articles (Kurucz *et al*., 2010, 2011) for the analysis of fragments of description logic $\mathcal{EL}$ (and some of its extensions) whose validity problem can be solved in polynomial time. Now, regarding propositional intuitionistic logic, the best known fragment with decidable derivability problem is the Horn fragment, which is solvable in linear time (Dantzin *et al*., 2001; Dowling & Gallier, 1984; Minoux, 1988). Mints (1992) found another fragment whose validity problem is decided in polynomial time.

The derivability problem for the primal fragment is linear-time decidable as well (Gurevich & Neeman, 2011). Propositional primal infon logic is an extension of the primal fragment of intuitionistic logic with quotation connectives. Originally there were two series of quotation connectives, "p `said`" and "p `implied`" where p ranges over an infinite list of principal constants; the associated derivability problem is linear-time decidable in the case of bounded quotation depth (Gurevich & Neeman, 2011). Later the `implied` series was removed. The derivability problem for the redefined logic is decidable in linear time (with no restriction on the quotation depth) (Gurevich, 2011).

Finally, consider the NNIL fragment of propositional intuitionistic logic (Visser *et al*., 2008). Recall that negation in intuitionistic logic is defined by $\neg\varphi := \varphi \to \bot$ and that the implicational complexity $\rho(\varphi)$ of a formula φ is defined as follows:

- $\rho(\top) = \rho(\bot) = \rho(v) = 0$, where v is a variable.
- $\rho(\varphi_1 \wedge \varphi_2) = \rho(\varphi_1 \vee \varphi_2) = \max\{\rho(\varphi_1), \rho(\varphi_2)\}$.
- $\rho(\varphi_1 \to \varphi_2) = \max\{\rho(\varphi_1) + 1, \rho(\varphi_2)\}$.

NNIL comprises the formulas with implicational complexity ≤ 1. We say that an NNIL formula φ is without *premise disjunctions* if, for every implication subformula $\alpha \to \beta$ of φ, disjunction does not occur in α.

Discussions with Alfred Visser led to the following new result that uses the soundness and completeness of Kripke semantics for propositional intuitionistic logic.

THEOREM 6.1.

(1) The validity problem for NNIL formulas without premise disjunction is decidable in polynomial time Visser 2012.

(2) The validity problem for NNIL is CONP*-complete.*

Proof.

(1) A propositional intuitionistic formula is valid iff it holds in every world of every Kripke model for propositional intuitionistic logic. Accordingly, validity of NNIL formulas without premise disjunctions can be decided inductively as follows:

- Constant $\top$ is valid, constant $\bot$ is not valid and a variable is not valid.
- A conjunction is valid iff both conjuncts are valid.
- A disjunction is valid iff at least one disjunct is valid (Theorem 5.4.2. of van Dalen, 2008).
- For an implication $\varphi_1 \to \varphi_2$, since φ_1 does not have disjunctions, it must have the form $p_1 \wedge p_2 \wedge \ldots \wedge p_k$, where each p_i is a variable. Then, $\varphi_1 \to \varphi_2$ is valid iff $\varphi_2[p_1 := \top, p_2 := \top, \ldots, p_k := \top]$ is valid.

This gives rise to a validity checking algorithm. It is easy to see that the algorithm runs in polynomial time.

(2) A formula $(\varphi_1 \vee \varphi_2) \to \psi$ is valid iff both $\varphi_1 \to \psi$ and $\varphi_2 \to \psi$ are valid. Note that this doubles the amount of work. But if we guess one disjunct φ_i and show that $\varphi_i \to \psi$ is not valid, then we have that $(\varphi_1 \vee \varphi_2) \to \psi$ is not valid. This idea leads to a nondeterministic algorithm for deciding whether a NNIL formula φ is not valid. First, for every disjunction occurring in a premise of some implication, make a guess and replace the disjunction with one of the disjuncts. Then, apply the polynomial-time procedure described in (1). Hence, the validity problem for NNIL is coNP.

Now, we prove the coNP-hardness by a reduction from the non-three-coloring problem. Given a graph $G = (V, E)$, we write a NNIL formula α such that G is three-colorable iff α is not intuitionistically valid. For every $v \in V$, define three propositional variables R_v, G_v and B_v. Define $\alpha = \beta \to (\gamma \vee \delta)$, where

$$\beta = \bigwedge_{v \in V} (R_v \vee G_v \vee B_v)$$

$$\gamma = \bigvee_{v \in V} ((R_v \wedge G_v) \vee (G_v \wedge B_v) \vee (R_v \wedge B_v))$$

$$\delta = \bigvee_{(u,v) \in E} ((R_u \wedge R_v) \vee (G_u \wedge G_v) \vee (B_u \wedge B_v)).$$

Suppose G is 3-colorable, with colors red, green, and blue. Define a classical valuation given by R_v is true iff v is colored red, G_v is true iff v is colored green, and B_v is true iff v is colored blue. In this valuation β is true while γ and δ are false. So α is not valid classicaly and, therefore, it is not valid intuitionistically. For the converse, if α is not valid intuitionistically, then there is a Kripke model for propositional intuitionistic logic with a world in which α does not hold. Hence, in this world, β is true while γ and δ are false; which implies that there is a 3-coloring for G. Since the 3-coloring problem is NP-complete, we conclude that the validity problem for NNIL is coNP-hard, and hence, it is coNP-complete.

§7. Acknowledgments. We thank Artem Melentyev for implementing the decision algorithm and the anonymous referees for their work. □

BIBLIOGRAPHY

Avron, A., & Lahav, O. (2009). Canonical constructive systems. *Proceedings of TABLEAUX 2009*, 62–76.

Beklemishev, L., Blass, A., & Gurevich, Y. What is the logic of information? In preparation.

Beklemishev, L., & Gurevich, Y. (2012). Propositional primal logic with disjunction. *Journal of Logic and Computation*, published online May 29, 2012. doi:10.1093/logcom/exs018.

Bjørner, N., de Caso, G., & Gurevich, Y. (2012). From primal infon logic with individual variables to datalog. In Erdem, E., et al., editors. *Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, Springer Lecture Notes in Computer Science, Springer Verlag, Vol. 7265, pp. 72–86.

Blass, A., & Gurevich, Y. (2010). Hilbertian deductive systems, infon logic, and datalog. Bulletin of the EATCS, **102**, 122–150. A slightly revised version in Microsoft Research Tech, Report MSR-TR-2011-81, June 2011.

Blass, A., Gurevich, Y., Moskał, & Neeman, I. (2011). Evidential authorization. In Nanz, S., editor. *The Future of Software Engineering*. Springer, 77–99.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms* (second edition). MIT Press and McGraw-Hill.

Cotrini, C., & Gurevich, Y. (2013). Basic primal infon logic. *Journal of Logic and Computation*.

Dantzin, E., Eiter, T., Gottlob, G., & Voronkov, A. (2001). Complexity and expressive power of logic programming. *ACM Computing Surveys*, **33**(3), 374–425.

DKAL at CodePlex. `http://dkal.codeplex.com/`, viewed August 10, 2012.

Dowling, W., & Gallier, J. (1984). Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, **1**, 267–284.

Fortnow, L. (2009). Shaving logs with unit cost. http://blog.computationalcomplexity.org/2009/05/shaving-logs-with-unit-cost.html, seen July 22, 2012.

Gabbay, D., Kurucz, A., Wolter, F., & Zakharyaschev, M. (2003). *Many-dimensional Modal Logics: Theory and Applications*. Elsevier.

Gurevich, Y. (2011). *Two Notes on Propositional Primal Logic*. Technical Report MSR-TR-2011-70, Microsoft Research, May 2011.

Gurevich, Y., & Neeman, I. (2008). DKAL: Distributed-knowledge authorization language. In *Proceedings of CSF 2008*. IEEE Computer Society, pp. 149–162.

Gurevich, Y., & Neeman, I. (2009). DKAL 2 — A simplified and improved authorization language. Microsoft Research Tech Report MSR-TR-2009-11, February 2009.

Gurevich, Y., & Neeman, I. (2011). Infon logic: the propositional case. *ACM Transactions on Computation Logic* **12**(2), Article 9, January 2011, and (slight revisied) Microsoft Research Tech. Report MSR-TR-2011-90, July 2011.

Halpern, J. (1995). The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, **75**(2), 361–372.

Kurucz, A., Wolter, F., & Zakharyaschev, M. (2010). Islands of tractability for relational constraints: towards dichotomy results for the description logic EL, In Beklemishev, L., Goranko, V., and Shehtman, V., editors. *Advances in Modal Logic*, College Publications, Vol. 8, pp. 271–291.

Kurucz, A., Wolter, F., & Zakharyaschev, M. (2011). On P/NP dichotomies for EL subsumption under relational constraints. In *Proceedings of DL 2011*.

Minoux, M. (1988). LTUR: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Information Processing Letters*, **29**(1), 1–12.

Mints, G. (1992). Complexity of subclasses of the intuitionistic propositional calculus. *BIT*, **32**, 64–69.

Savateev, Y. (2009). *Investigation of Primal Logic*. Unpublished internship (at Microsoft Research) report.

van Dalen, D. (2008). *Logic and Structure* (fourth edition). Springer.

Visser, A. (2012). Personal communication, January 7, 2012.

Visser, A., van Benthem, J., de Jongh, D., & de Lavalette, G. R. (2008). NNIL, a study in intuitionistic propositional logic. In *Logic Group Preprint Series*, Holland: Dept of Philosophy, Utrecht University, Vol. 111.

Wolter, F., & Zakharyaschev, M. (1999). Intuitionistic modal logic. In Cantini, A., Casari, E., and Minari, P., editors. *Logic and Foundations of Mathematics*. Kluwer Academic Publishers, pp. 227–238.

CARLOS COTRINI
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZÜRICH
SWITZERLAND
E-mail: ccarlos@student.ethz.ch

YURI GUREVICH
MICROSOFT RESEARCH
REDMOND
WA 98052
E-mail: gurevich@microsoft.com