Università di Pisa
Dipartimento di Informatica

Technical Report: TR-03-11

Decidability of Freshness, Undecidability of Revelation

Giorgio Ghelli Giovanni Conforti

July 24, 2003

Address: via F. Buonarroti 2, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

Decidability of Freshness, Undecidability of Revelation

Giorgio Ghelli Giovanni Conforti

July 24, 2003

Abstract

We study decidability of a logic for describing processes with restricted names. We choose a minimal fragment of the Ambient Logic, but the techniques we present should apply to every logic which uses Cardelli and Gordon revelation and hiding operators, and Gabbay and Pitts freshness quantifier.

We start from the static fragment of ambient logic that Calcagno Cardelli and Gordon proved to be decidable. We prove that the addition of a hiding quantifier makes the logic undecidable. Hiding can be decomposed as freshness plus revelation. Quite surprisingly, freshness alone is decidable, but revelation alone is not.

1 Introduction

The term *Spatial Logics* has been recently used to refer to logics equipped with the composition-separation operator $A \,|\, B$. Spatial logics are emerging as an interesting tool to describe properties of several structures. Models for spatial logics include computational structures such as heaps [19, 17], trees [6], trees with hidden names [8], graphs [7], concurrent objects [4], as well as process calculi such as the π-calculus [2, 3] and the Ambient Calculus [10, 12].

In all these structures, a notion of *name restriction* arises. The restriction $(\nu n)\, P$ (in π-calculus notation) of a name n in a structure P, is a powerful abstraction mechanism that can be used to model many kinds of information that are protected by the computational model, such as hidden encryption keys [1], the actual variable names in λ-calculus, object identifiers in object calculi, locations in a garbage-collected heap. Here "protected" means that no public name can ever clash with one that is protected, and that any observable behaviour may depend on the equality between two names, but not on the actual value of a protected name.

Reasoning about protected names is difficult, since you have no name for them. Cardelli and Gordon suggest an elegant solution to this problem [11]. They adopt Gabbay and Pitts fresh name quantification, original used for binder manipulation and for Nominal Logics [15, 18], and combine it with a new operator, *revelation*, which can use a public name to denote a private one. The combination of freshness quantification and revelation gives raise to a new quantifier, *hidden name quantification*, which can be used to describe properties of restricted names in a natural way.

In [5] decidability of validity and model-checking of a spatial logic describing trees without local names is studied. This logic is the quantifier free static fragment of the Ambient Logic. Extensions of this logic can be used to describe [6], query [9], and reason about [14] tree-shaped semistructured data.

In this paper we study decidability of validity and model-checking for spatial logics describing trees (or static ambients) with local names. In particular we study how the introduction of freshness, revelation, and hiding influences the decidability problem. While we started this work with the aim of proving decidability of hiding, we found out quite a different situation. Our main results are:

- freshness without revelation gives a rich decidable logic (Theorem 4.8)
- even a minimal logic (conjunction, negation, and binary relations) becomes undecidable if it is enriched with revelation (Corollary 5.28) or with hiding (Corollary 5.33)

Another contribution is the study of quantifier extrusion in spatial logic. We introduce an extrusion algorithm for freshness (Lemma 4.6), and we prove that no extrusion algorithm exists for first order quantifiers, revelation, and hiding (Corollary 4.12).

We will discuss decidability of validity, satisfiability, and model-checking for various logics. Throughout the paper, “decidability of a logic” is used for “decidability of validity and satisfiability for that logic”.

2 The Tree Model

Definition 2.1 *The set $\mathcal{T}_{\mathcal{N}}$ of the abstract trees generated by an infinite name set $\mathcal{N}$ is defined by the following grammar, with $n \in \mathcal{N}$.*

trees	$T, U ::=$	$\mathbf{0}$	*empty tree*
		$n[T]$	*tree branch*
		$T \mid U$	*composition of trees*
		$(\nu n)\, T$	*local name*

Free names $fn(T)$ and bound names are defined as usual. On these trees we define the usual congruence rules, with extrusion of local names:

Table 2.1. *Congruence rules*

$T \equiv T$	(Struct Refl)
$T \equiv U \Rightarrow U \equiv T$	(Struct Symm)

$T \equiv U, U \equiv V \Rightarrow T \equiv V$	(Struct Trans)
$T \equiv U \Rightarrow T \mid V \equiv U \mid V$	(Struct Par)
$T \equiv U \Rightarrow n[T] \equiv n[U]$	(Struct Amb)
$T \equiv U \Rightarrow (\nu n)\, T \equiv (\nu n)\, U$	(Struct Res)
$T \mid U \equiv U \mid T$	(Struct Par Comm)
$(T \mid U) \mid V \equiv T \mid (U \mid V)$	(Struct Par Assoc)
$T \mid \mathbf{0} \equiv T$	(Struct Zero Par)
$m \notin fn(T) \Rightarrow (\nu n)\, T \equiv (\nu m)\, T\{n \leftarrow m\}$	(Struct Renaming)
$n \notin fn(T) \Rightarrow T \mid (\nu n)\, U \equiv (\nu n)\, (T \mid U)$	(Struct Extr Par)
$(\nu n)\, \mathbf{0} \equiv \mathbf{0}$	(Struct Extr Zero)
$n_1 \neq n_2 \Rightarrow n_1[(\nu n_2)\, T] \equiv (\nu n_2)\, n_1[T]$	(Struct Extr Amb)
$(\nu n_1)\, (\nu n_2)\, T \equiv (\nu n_2)\, (\nu n_1)\, T$	(Struct Extr Comm)

Lemma 2.2 (Free Names) *If* $T \equiv U$ *then* $fn(T) = fn(U)$

Lemma 2.3 (Inversion (see [11]))

1. *If* $(\nu n)\, T \equiv \mathbf{0}$ *then* $T \equiv \mathbf{0}$
2. *If* $(\nu n)\, T \equiv m[U]$ *then* $\exists U' \in \mathcal{T}_{\mathcal{N}}.\ T \equiv m[U'],\ U \equiv (\nu n)\, U'$
3. *If* $(\nu n)\, T \equiv U \mid U'$ *then* $\exists \bar{U}, \bar{U}' \in \mathcal{T}_{\mathcal{N}}.\ T \equiv \bar{U} \mid \bar{U}',\ \bar{U} \equiv (\nu n)\, U,\ \bar{U}' \equiv (\nu n)\, U'$

Definition 2.4 *The set of trees in* extruded normal form *(ENF) is the minimum set such that:*

- *a tree with no local restriction is in ENF;*
- *if* T *is in ENF and* $n \in fn(T)$ *then* $(\nu n)\, T$ *is in ENF.*

Hence, a tree is in ENF iff it is composed by a prefix of restrictions followed by an restriction-free *matrix*, all the restrictions define names that actually appear in the tree, and all the restricted names are mutually different.

Lemma 2.5 *For every tree* T *there exists* U *such that* $T \equiv U$ *and* U *is in ENF.*

The next lemma says that the ENF of a congruence class of trees is unique modulo renaming of bound names, reordering of the prefix, and congruence of the renamed matrix.

Lemma 2.6 *If* $(\nu n_1)\ \ldots (\nu n_j)\, U \equiv (\nu n'_1)\ \ldots (\nu n'_k)\, U'$*, and the two trees are in ENF, and* U *and* U' *are the matrixes, then* $j = k$ *and there exists a bijection* τ *between* $\{n_1, \ldots, n_j\}$ *and* $\{n'_1, \ldots, n'_k\}$ *such that*

$$\exists U''.\ U \equiv U'' = U'\{n'_i \leftarrow \tau(n_i)\}^{i \in 1..j}$$

Notation 2.7 *Let n be injective from $I = \{i_1, \ldots, i_j\}$ to $\mathcal{N}$. Then:*

$$(\vec{\nu}_{i\in I} n_i)\, T \triangleq (\nu n_{i_1})\ \ldots (\nu n_{i_j})\, T$$

Notation 2.8 *We will use ENF to denote the set of all terms in ENF, and $ENF(T)$ to denote the set $\{U : U \in ENF,\ U \equiv T\}$.*

3 The Logic

3.1 Definition

We will study a sublogic of the Ambient Logic without recursion and where no temporal operator appears. We will call it *Spatial Logic* (SL). It is defined as follows.

Definition 3.1 *The set $\mathcal{A}$ of the formulas of the full logic is defined by the following grammar; we will consider some sub-logics later on. η stands for either a name $n \in \mathcal{N}$ or a name variable $x \in \mathcal{X}$.*

$A, B ::=$	$\mathbf{0}$	*empty tree*
	$\eta[A]$	*location*
	$A \mid B$	*composition of trees*
	$\text{И}x.\, A$	*fresh quantification*
	$\exists x.\, A$	*existential quantification*
	$A \wedge B$	*conjunction*
	$\neg A$	*negation*
	$\eta \circledR A$	*revelation*
	$A \triangleright B$	*composition adjunct*
	$A @ \eta$	*location adjunct*
	$A \oslash \eta$	*revelation adjunct*

We will also study the property of the following derived operators:

$\mathsf{H}x.\, A$	$\triangleq$	$\text{И}x.\, x \circledR A$	hiding
$\copyright \eta$	$\triangleq$	$\neg \eta \circledR \mathsf{T}$	η appears free
$\eta = \eta'$	$\triangleq$	$(\eta[\mathsf{T}]) @ \eta'$	equality

We will always assume that $\exists x$, $\text{И}x$ and $\eta\circledR$ bind as far to the right as possible, so that, for example, $\exists x.\, A \wedge \exists y.\, B$ is the same as $\exists x.\, (A \wedge \exists y.\, A)$.

We assume the usual definitions for: (i) the derived operators $A \vee B$, T, F, $\forall x.A$, $\eta \neq \eta'$, $A \Rightarrow B$, $A \Leftrightarrow B$; (ii) free variables $fv(A)$. It is worth emphasizing that revelation is not a binder, i.e. $fv(\eta \circledR A) = fv(\eta) \cup fv(A)$. $fv(\eta)$ is defined as $\{\eta\}$ when η is a variable x, and as $\emptyset$ when η is a name n. Closed formulas are formulas without free variables.

We will use the $nm(A)$ to denote the set of all names that appear in a formula. We define $fn(T, A) \triangleq fn(T) \cup nm(A)$; notice that every name that appears in a formula is free, since $\exists$ and И only bind variables.

Definition 3.2 *We define the satisfaction relation $T \models A$ between trees in $\mathcal{T}_\mathcal{N}$ and closed formulas as follows:*

$$
\begin{array}{lcl}
T \models \mathbf{0} & \triangleq & T \equiv \mathbf{0} \\
T \models n[A] & \triangleq & \exists U \in \mathcal{T}_\mathcal{N}.\, T \equiv n[U] \textit{ and } U \models A \\
T \models A \mid B & \triangleq & \exists T_1, T_2 \in \mathcal{T}_\mathcal{N}.\, T \equiv T_1 \mid T_2 \textit{ and } T_1 \models A \textit{ and } T_2 \models B \\
T \models \text{И}x.\, A & \triangleq & \exists n \notin fn(T) \cup nm(A).\, T \models A\{x \leftarrow n\} \\
T \models \exists x.\, A & \triangleq & \exists n \in \mathcal{N}.\, T \models A\{x \leftarrow n\} \\
T \models A \wedge B & \triangleq & T \models A \textit{ and } T \models B \\
T \models \neg A & \triangleq & T \not\models A \\
T \models n \circledR A & \triangleq & \exists U \in \mathcal{T}_\mathcal{N}.\, T \equiv (\nu n)\, U \textit{ and } U \models A \\
T \models A \triangleright B & \triangleq & \forall U \in \mathcal{T}_\mathcal{N}.\, U \models A \Rightarrow T \mid U \models B \\
T \models A @ n & \triangleq & n[T] \models A \\
T \models A \oslash n & \triangleq & (\nu n)\, T \models A
\end{array}
$$

We will also use the notation $T, \rho \models A$, where ρ is a ground substitution mapping all the free variables of A into names in $\mathcal{N}$, with the following meaning (where $\rho\!\downarrow$ is the domain of ρ):

$$T, \rho \models A \quad \Leftrightarrow_{def} \quad \rho\!\downarrow \supseteq fv(A) \;\wedge\; T \models A\rho$$

Now we define the standard notions of formula validity, satisfiability, of formula implication, and of formula equivalence for SL.

$$
\begin{array}{lcll}
\mathbf{vld}(A) & \triangleq & \forall T \in \mathcal{T}_\mathcal{N}, \rho : fv(A) \to \mathcal{N}.\ T, \rho \models A & \textit{(validity)} \\
\mathbf{sat}(A) & \triangleq & \exists T \in \mathcal{T}_\mathcal{N}, \rho : fv(A) \to \mathcal{N}.\ T, \rho \models A & \textit{(satisfiability)} \\
A \vdash B & \triangleq & \forall T \in \mathcal{T}_\mathcal{N}, \rho : (fv(A) \cup fv(B)) \to \mathcal{N}. & \\
 & & \qquad\qquad T, \rho \models A \;\Rightarrow\; T, \rho \models B & \textit{(implication)} \\
A \dashv\vdash B & \triangleq & A \vdash B \text{ and } B \vdash A & \textit{(equivalence)}
\end{array}
$$

The following lemmas come from [11], or are easily derivable from there.

Lemma 3.3 (Implication)

$$
\begin{array}{lcl}
A \vdash B & \Leftrightarrow & \mathbf{vld}(A \Rightarrow B) \\
A \dashv\vdash B & \Leftrightarrow & \mathbf{vld}(A \Leftrightarrow B)
\end{array}
$$

Let $\vec{\forall} A$ denote $\forall x_1 \ldots \forall x_n.\ A$, where $\{x_1 \ldots x_n\} = fv(A)$, and similarly for $\vec{\exists} A$.

Lemma 3.4 (Closure)

$$\mathbf{vld}(A) \Leftrightarrow \mathbf{vld}(\vec{\forall} A)$$
$$\mathbf{sat}(A) \Leftrightarrow \mathbf{sat}(\vec{\exists} A)$$

The following lemma expresses the fact that validity can be reduced to model-checking using $\triangleright$ and quantification, or just $\triangleright$ alone, when the formula is closed.

Lemma 3.5 (Validity by Model-Checking)

$$\mathbf{vld}(A) \;\Leftrightarrow\; \mathbf{0} \models \mathsf{T} \triangleright \vec{\forall} A \;\Leftrightarrow\; \mathbf{0} \models \vec{\forall}(\mathsf{T} \triangleright A)$$

Definition 3.6 (Formula with Holes (see [11])) *We use $B\{-\}$ to indicate a formula with a set of formula holes, indicated by $-$, and $B\{A\}$ to denote the formula obtained by filling these holes with A, after renaming the variables in B to avoid capturing variables of A.*

Lemma 3.7 (Substitution (see [11]))

$$\begin{array}{lll} & \mathbf{vld}(A \Leftrightarrow A') & \Rightarrow \quad \mathbf{vld}(B\{A\} \Leftrightarrow B\{A'\}) \\ i.e. & A \dashv\vdash A' & \Rightarrow \quad B\{A\} \dashv\vdash B\{A'\} \end{array}$$

Notation 3.8 (Disjointness)

$$K \# K' \quad \Leftrightarrow_{def} \quad K \cap K' = \emptyset$$

Notation 3.9 *We write $M : \mathbf{M} \overset{in}{\rightharpoonup} \mathbf{N}$ to specify that M is partial and injective from $\mathbf{M}$ to $\mathbf{N}$, and $M : \mathbf{M} \overset{in}{\rightarrow} \mathbf{N}$ to specify that M is total and injective from $\mathbf{M}$ to $\mathbf{N}$.*

For any partial function $N : \mathbf{M} \rightharpoonup \mathbf{N}$, we will use $N\!\downarrow$ to denote its actual domain and $M\!\uparrow$ to denote its actual range, i.e.:

$$N\!\downarrow = \{m : \exists n \in \mathbf{N}.\ N(m) = n\} \quad N\!\uparrow = \{n : \exists m \in \mathbf{M}.\ N(m) = n\}$$

The following properties hold for the Ambient Logic with revelation and fresh quantification [11].

Lemma 3.10 (Satisfaction is up to $\equiv$) *If $T \models A$ and $T \equiv U$ then $U \models A$*

Lemma 3.11 *If $m \notin fn(T, A)$ and $n \notin fn(T, A)$ then:*

$$T \models A\{x \leftarrow m\} \Leftrightarrow T \models A\{x \leftarrow n\}$$

Corollary 3.12 *If $\rho, \rho' : \mathcal{X} \overset{in}{\rightharpoonup} \mathcal{N}$, $\rho\!\downarrow = \rho'\!\downarrow$, $\rho\!\uparrow \# fn(T, A)$, and $\rho'\!\uparrow \# fn(T, A)$, then:*

$$T, \rho \models A \;\Leftrightarrow\; T, \rho' \models A$$

The next Corollary will be used throughout the paper: we will use each of (1)-(4) as if it were the definition of (5). From now on, every quantification on sets on names will always be implictly (or explictly) qualified to range over finite sets of names only.

Corollary 3.13 (Gabbay-Pitts Property) *For any* $\mathbf{N} \subset \mathcal{N}$ *finite, all the following are equivalent:*

1. $\forall m \notin fn(T, A).\ T \models A\{x \leftarrow m\}$
2. $\forall m \notin (fn(T, A) \cup \mathbf{N}).\ T \models A\{x \leftarrow m\}$
3. $\exists m \notin (fn(T, A) \cup \mathbf{N}).\ T \models A\{x \leftarrow m\}$
4. $\exists m \notin fn(T, A).\ T \models A\{x \leftarrow m\}$
5. $T \models \text{И} x.\, A$

Proof $(1) \Rightarrow (2)$: $m \notin (fn(T, A) \cup \mathbf{N}) \Rightarrow m \notin fn(T, A)$.
$(2) \Rightarrow (3)$: $\mathcal{N} \setminus (fn(T, A) \cup \mathbf{N})$ is not empty, since $fn(T, A) \cup \mathbf{N}$ is finite.
$(3) \Rightarrow (4)$: $m \notin (fn(T, A) \cup \mathbf{N}) \Rightarrow m \notin fn(T, A)$.
$(4) \Rightarrow (1)$: by Lemma 3.11
$(4) \Leftrightarrow (5)$: by Definition 3.2 □

We introduce a couple of lemmas and corollaries which will be useful in Section 5.

Lemma 3.14 *Let* $(\vec{\nu}_{i\in I} n_i)\, T$ *be in ENF, let* T *be the matrix, and let* m *be a name not in* $\{n_i\}^{i\in I}$*; then:*

$$\begin{aligned}(\vec{\nu}_{i\in I} n_i)\, T \models m \circledR A \quad \Leftrightarrow \quad & m \notin fn(T) \wedge ((\exists h \in I.\ (\vec{\nu}_{i\in (I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A) \\ & \vee\ (\vec{\nu}_{i\in I} n_i)\, T \models A)\end{aligned}$$

Corollary 3.15 *For* $(\vec{\nu}_{i\in I} n_i)\, T$ *in ENF, where* T *is the matrix, if*

$$\forall n \in fn(T).\ \exists T'.\ T \equiv n[\mathbf{0}] \mid T'$$

then:

$$\begin{aligned}& (\vec{\nu}_{i\in I} n_i)\, T \models m \circledR (A \wedge (m[\mathbf{0}] \mid \mathsf{T})) \\ \Leftrightarrow\ & m \notin fn(T) \wedge \exists h \in I.\ (\vec{\nu}_{i\in (I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A\end{aligned}$$

Lemma 3.16 *For* $(\vec{\nu}_{i\in I} n_i)\, T$ *in ENF, where* T *is the matrix:*

$$\begin{aligned}& (\vec{\nu}_{i\in I} n_i)\, T \models \mathsf{H} x.\, A \\ & \quad \Leftrightarrow \quad (\forall m \notin (nm(A) \cup fn(T)). \\ & \qquad\qquad \exists h \in I.\ (\vec{\nu}_{i\in (I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A\{x \leftarrow m\}) \\ & \qquad\quad \vee\ (\vec{\nu}_{i\in I} n_i)\, T \models A \\ & \quad \Leftrightarrow \quad (\exists m \notin (nm(A) \cup fn(T)). \\ & \qquad\qquad \exists h \in I.\ (\vec{\nu}_{i\in (I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A\{x \leftarrow m\}) \\ & \qquad\quad \vee\ (\vec{\nu}_{i\in I} n_i)\, T \models A\end{aligned}$$

Corollary 3.17 *For $(\vec{\nu}_{i\in I} n_i)\, T$ in ENF, where T is the matrix, if*

$$\forall n \in fn(T).\ \exists T'.\ T \equiv n[\mathbf{0}] \mid T'$$

then:

$$\begin{aligned} & (\vec{\nu}_{i\in I} n_i)\, T \models \mathsf{H}x.\, A \wedge (x[\mathbf{0}] \mid \mathsf{T}) \\ \Leftrightarrow\quad & \forall m \notin (nm(A) \cup fn(T)). \\ & \quad \exists h \in I.\ (\vec{\nu}_{i\in(I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A\{x \leftarrow m\} \\ \Leftrightarrow\quad & \exists m \notin (nm(A) \cup fn(T)). \\ & \quad \exists h \in I.\ (\vec{\nu}_{i\in(I\setminus\{h\})} n_i)\, T\{n_h \leftarrow m\} \models A\{x \leftarrow m\} \end{aligned}$$

4 Decidable Sublogics

4.1 Quantifier-free Decidable Sublogics

We start from the following result presented in [5].

Theorem 4.1 (Calcagno-Cardelli-Gordon) *The model-checking problem restricted to closed formulas with no quantification and revelation is decidable over trees with no local names.*

Model-checking and closed-validity problems are equivalent in SL fragments with composition adjunct (Lemma 3.5).

Corollary 4.2 (Calcagno-Cardelli-Gordon) *The validity and satisfiability problems restricted to closed formulas with no quantification and revelation are decidable over trees with no local names.*

We now extend this result by adding restricted names to the models and the revelation adjunct ($A \oslash n$) to the logic.

Theorem 4.3 (Adding Restricted Names and Revelation Adjunct) *The model-checking problem restricted to closed formulas generated by the following grammar ($\exists, \mathsf{И}, \mathsf{H}$, Ⓡ: no, $\oslash$: yes):*

$$A ::= \mathbf{0} \mid n[A] \mid A \,|\, A \mid A \wedge A \mid \neg A \mid A \triangleright A \mid A@n \mid A \oslash n$$

is decidable over all trees (i.e., including trees with restricted names).

Proof (Sketch) We follow the schema of [5]. We define the following equivalence relation $\sim_{h,w,N}$:

$$\begin{aligned} & T \sim_{0,w,N} U \quad \Leftrightarrow fn_N(T) = fn_N(U) \qquad \text{where } fn_N(T) = fn(T) \cap N \\ & T \sim_{h+1,w,N} U \Leftrightarrow \\ & \quad \forall M \subseteq N,\ i \in 1..w,\ n \in N,\ T_j \text{ such that for } j \in 1..i.\ fn_N(T_j) = M \\ & \qquad \text{if } T \equiv n[T_1] \mid \ldots \mid n[T_i] \mid T' \\ & \qquad \text{then } U \equiv n[U_1] \mid \ldots \mid n[U_i] \mid U' \end{aligned}$$

such that $T_j \sim_{h,w,N} U_j$ for $j \in 1..i$
and vice versa
and $\forall i \in 1..w,\ T_j$
if $T \equiv T_1 \mid \ldots \mid T_i$
then $U \equiv U_1 \mid \ldots \mid U_i$
with $fn_N(T_j) = fn_N(U_j)$ for $j \in 1..i$
and vice versa

We first prove that $\sim_{h,w,N}$ is a congruence:

$$\begin{array}{lll} T \sim_{h,w,N} U & \Rightarrow & n[T] \sim_{h+1,w,N} n[U] \\ T \sim_{h,w,N} U & \Rightarrow & (\nu n)\, T \sim_{h,w,N} (\nu n)\, U \\ T \sim_{h,w,N} U,\ T' \sim_{h,w,N} U' & \Rightarrow & T \mid T' \sim_{h,w,N} U \mid U' \qquad (1) \end{array}$$

Then we show how to enumerate a witness for each equivalence class of $\sim_{h,w,N}$. We say that a tree T is "sealed" if it is only congruent to restricted terms, i.e.

$$T \text{ is sealed} \quad \Leftrightarrow \quad \forall U.\ U \equiv T \Rightarrow \exists n, U'.\ U = (\nu n)\, U'.$$

(For example, $(\nu n)\,(n[\mathbf{0}] \mid m[n])$ is sealed while $(\nu n)\,(\nu m)\,(n[\mathbf{0}] \mid m[\mathbf{0}])$ is not.) We say that a term is in $NF(h,w,N)$ iff it can be written as

$$\Sigma_{n \in N, T \in NF(h,w,N)}\ b_{n,T} \cdot n[T] \ \mid\ \Sigma_{M \subseteq N}\ b_M \cdot (\nu m)\, m[M]$$

Where $b_{n,T}$ and b_M satisfy $0 \leq b_i \leq w$, and $i \cdot T$ stands for i copies of T separated by $|$. We show that for each T and for each (h,w,N) there exists $U \sim_{h,w,N} T$ such that $U \in NF(h,w,N)$. By induction on (h,w,N) we show that, for each (h,w,N), the corresponding set of vectors $b_{n,T}$ and b_M is finite and can be effectively enumerated.

Now we define a size $|A|$ for each formula A, and we prove that

$$|A| = (h,w,N),\ T \models A,\ T \sim_{h,w,N} U \ \Rightarrow\ U \models A \qquad (2).$$

If $|B| = (h,w,N)$ and $U_i^{(h,w,N)}$ is an enumeration of a witness for each equivalence class of $\sim_{h,w,N}$, (1) + (2) imply that checking $T \models A \triangleright B$ can be reduced to checking that, for each $U_i^{(h,w,N)}$, $U_i^{(h,w,N)} \models A \Rightarrow U_i^{(h,w,N)} \mid T \models B$. Model-checking the other operators is easy. $\square$

We show now that presence ©n can be encoded in terms of revelation adjunct $A \oslash n$.

Lemma 4.4 (Presence from Revelation Adjunct) *Given $n \neq m$:*

$$©n \dashv\vdash (n[\mathbf{0}] \triangleright ((\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \oslash n))@m$$

Proof

$$
\begin{array}{rrl}
 & & T \models (n[\mathbf{0}] \triangleright ((\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \oslash n))@m \\
 & \Leftrightarrow & m[T] \models n[\mathbf{0}] \triangleright ((\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \oslash n) \\
 & \Leftrightarrow & m[T] \mid n[\mathbf{0}] \models (\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \oslash n \\
 & \Leftrightarrow & (\nu n)\,(m[T] \mid n[\mathbf{0}]) \models \neg(\neg\mathbf{0} \mid \neg\mathbf{0}) \\
(1) & \Leftrightarrow & \forall T_1, T_2.\, T_1 \mid T_2 \equiv\ (\nu n)\,(m[T] \mid n[\mathbf{0}]) \Rightarrow T_1 \equiv \mathbf{0} \vee T_2 \equiv \mathbf{0}
\end{array}
$$

Now if $n \in fn(T)$ then the tree $U = (\nu n)\,(m[T] \mid n[\mathbf{0}])$ is sealed, thus it can be splitted into $U' \mid T_{\mathbf{0}}$ and $T_{\mathbf{0}} \mid U'$ only (where $U' \equiv U$ and $T_{\mathbf{0}} \equiv \mathbf{0}$) and (1) is true. If $n \notin fn(T)$ then $(\nu n)\,(m[T] \mid n[\mathbf{0}]) \equiv m[T] \mid (\nu n)\, n[\mathbf{0}]$ and (1) is false. Hence (1) $\Leftrightarrow n \in fn(T) \Leftrightarrow T \models Ⓒ n$ □

Note that the previous encoding $Ⓒ^m \eta = (\eta[\mathbf{0}] \triangleright ((\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \oslash \eta))@m$ is not applicable when η is a variable, since the encoding relies on m never clashing with η. For example:

$$\mathbf{0} \not\models \exists x.\, Ⓒ x \quad \text{but} \quad \mathbf{0} \models \exists x.\, Ⓒ^m x \quad \text{since} \quad \mathbf{0} \models Ⓒ^m m$$

However we can use Lemma 4.4 to encode the general case: given two names m, m' such that $m \neq m'$, $Ⓒ\eta \dashv\vdash Ⓒ^m \eta \wedge Ⓒ^{m'} \eta$

$$
\begin{array}{lll}
 & & T, \rho \models Ⓒ^m \eta \wedge Ⓒ^{m'} \eta \Leftrightarrow \text{ (by cases)} \\
\eta\rho = m & \Rightarrow & T \models \mathsf{T} \wedge Ⓒ^{m'} m \Leftrightarrow T, \rho \models Ⓒ\eta \\
\eta\rho = m' & \Rightarrow & T \models Ⓒ^m m' \wedge \mathsf{T} \Leftrightarrow T, \rho \models Ⓒ\eta \\
\eta\rho \neq m, \eta\rho \neq m' & \Rightarrow & T \models Ⓒ^m \eta\rho \wedge T \models Ⓒ^{m'} \eta\rho \Leftrightarrow T, \rho \models Ⓒ\eta
\end{array}
$$

Of course, $\text{И} y.\, Ⓒ^y \eta$, where y is a fresh variable, would work as well, but we are trying to encode $Ⓒ\eta$ without quantifiers.

The encoding gives us the following corollary.

Corollary 4.5 (Adding Presence Revelation)
The model-checking problem restricted to closed formulas generated by the following grammar ($\exists$, И, H, Ⓡ: no, Ⓒ, $\oslash$: yes):

$$A ::= \mathbf{0} \mid n[A] \mid A \,|\, A \mid A \wedge A \mid \neg A \mid Ⓒ n \mid A \triangleright A \mid A@n \mid A \oslash n$$

is decidable over all trees (i.e., including trees with restricted names).

4.2 Quantifier Extrusion

An extrusion algorithm for a set of logical operators O is an algorithm that transforms a formula into an equivalent formula in O-prenex form, i.e. into a formula formed by a prefix of operators from O followed by a matrix where they do not appear.

First order logic admits a simple extrusion algorithm for the pair $\exists, \forall$. We will show that:

- in a spatial logic with the $\triangleright$ operator, extrusion implies decidability (Corollaries 4.9, 4.10, and 4.11);
- the freshness quantifier admits extrusion (Lemma 4.6), hence is decidable;
- undecidability of the revelation operator, existential quantifier, and hiding quantifier, implies that no extrusion algorithm can exist for them (Corollary 4.12).

We start our discussion of extrusion on a familiar ground, by listing, in Table 4.1, some logical equivalences that can be used to extrude universal and existential quantifiers from some of the other operators. The first four are the usual First Order Logic (FOL) rules.

Table 4.1. *Extrusion of existential quantifier*

$x \notin fv(B)$	$(\forall x.\, A) \wedge B$	$\dashv\vdash$	$\forall x.\, (A \wedge B)$	$(\forall\text{-}\wedge)$
$x \notin fv(B)$	$(\exists x.\, A) \wedge B$	$\dashv\vdash$	$\exists x.\, (A \wedge B)$	$(\exists\text{-}\wedge)$
	$\neg(\forall x.\, A)$	$\dashv\vdash$	$\exists x.\, (\neg A)$	$(\forall\text{-}\neg)$
	$\neg(\exists x.\, A)$	$\dashv\vdash$	$\forall x.\, (\neg A)$	$(\exists\text{-}\neg)$
$y \neq \eta$	$\eta[\forall y.\, A]$	$\dashv\vdash$	$\forall y.\, (\eta[A])$	$(\forall\text{-}[])$
$y \neq \eta$	$\eta[\exists y.\, A]$	$\dashv\vdash$	$\exists y.\, (\eta[A])$	$(\exists\text{-}[])$
$x \notin fv(B)$	$(\forall x.\, A) \mid B$	$\vdash$	$\forall x.\, (A \mid B)$	$(\forall\text{-}\mid\ \vdash)$
$x \notin fv(B)$	$(\exists x.\, A) \mid B$	$\dashv\vdash$	$\exists x.\, (A \mid B)$	$(\exists\text{-}\mid)$
$y \neq x$	$\text{И}x.\, \forall y.\, A$	$\vdash$	$\forall y.\, (\text{И}x.\, A)$	$(\forall\text{-И}\ \vdash)$
$y \neq x$	$\text{И}x.\, \exists y.\, A$	$\dashv$	$\exists y.\, (\text{И}x.\, A)$	$(\exists\text{-И}\ \dashv)$
	$m \circledR \forall y.\, A$	$\vdash$	$\forall y.\, (m \circledR A)$	$(\forall\text{-}\circledR\ \vdash)$
	$m \circledR \exists y.\, A$	$\dashv\vdash$	$\exists y.\, (m \circledR A)$	$(\exists\text{-}\circledR)$
$x \notin fv(B)$	$(\forall x.\, A) \triangleright B$	$\dashv$	$\exists x.\, (A \triangleright B)$	$(\forall\text{-}\triangleright l\ \dashv)$
$x \notin fv(B)$	$(\exists x.\, A) \triangleright B$	$\dashv\vdash$	$\forall x.\, (A \triangleright B)$	$(\exists\text{-}\triangleright l)$
$x \notin fv(A)$	$A \triangleright (\forall x.\, B)$	$\dashv\vdash$	$\forall x.\, (A \triangleright B)$	$(\forall\text{-}\triangleright r)$
$x \notin fv(A)$	$A \triangleright (\exists x.\, B)$	$\dashv$	$\exists x.\, (A \triangleright B)$	$(\exists\text{-}\triangleright r\ \dashv)$
$y \neq \eta$	$(\forall y.\, A)@\eta$	$\dashv\vdash$	$\forall y.\, (A@\eta)$	$(\forall\text{-}@)$
$y \neq \eta$	$(\exists y.\, A)@\eta$	$\dashv\vdash$	$\exists y.\, (A@\eta)$	$(\exists\text{-}@)$
$y \neq \eta$	$(\forall y.\, A) \oslash \eta$	$\dashv\vdash$	$\forall y.\, (A \oslash \eta)$	$(\forall\text{-}\oslash)$
$y \neq \eta$	$(\exists y.\, A) \oslash \eta$	$\dashv\vdash$	$\exists y.\, (A \oslash \eta)$	$(\exists\text{-}\oslash)$

If all the rules were double implications ($\dashv\vdash$), we may use them to extrude the existential quantifier in any formula, thanks to Lemma 3.7. However, the presence of some single implications prevents their direct use for this aim. Each simple implication we write is actually strict, i.e. whenever we write $A \vdash B$ in the table above we also mean that $B \vdash A$ is not valid in general. We prove this fact by exhibiting, for any such schematic implication, an instance $A' \vdash B'$ and a tree T such that $T \not\models A'$ and $T \models B'$. This proves that $B \vdash A$ cannot be valid. Below, we will often use a name m as an abbreviation for $m[\mathbf{0}]$. We write

$T \models A$ when any tree satisfies A, and $T \not\models A$ when no tree satisfies A. The notation $\eta \in \{\eta_1, \ldots, \eta_i\}$ stands for the formula $\eta = \eta_1 \vee \ldots \vee \eta = \eta_i$.

$$
\begin{array}{ll}
(\forall\text{-}\mid\ \not\dashv) & n[\mathbf{0}] \mid m[\mathbf{0}] \not\models (\forall x.\, x \in \{n, m\} \Rightarrow x[\mathbf{0}]) \mid \mathsf{T} \\
 & n[\mathbf{0}] \mid m[\mathbf{0}] \models \forall x.\, (x \in \{n, m\} \Rightarrow x[\mathbf{0}] \mid \mathsf{T}) \\
(\forall\text{-}\text{И}\ \not\dashv) & T \not\models \text{И}x.\, \forall y.\, x \neq y \\
 & T \models \forall y.\, \text{И}x.\, x \neq y \\
(\exists\text{-}\text{И}\ \not\vdash) & T \not\models \exists y.\, \text{И}x.\, x = y \\
 & T \models \text{И}x.\, \exists y.\, x = y \\
(\forall\text{-}\circledR\ \not\dashv) & (\nu n)\, (\nu n')\, n[m] \mid n'[m'] \not\models p \circledR \forall x.\, (p[\neg \copyright x] \mid \mathsf{T}) \\
 & (\nu n)\, (\nu n')\, n[m] \mid n'[m'] \models \forall x.\, p \circledR (p[\neg \copyright x] \mid \mathsf{T}) \\
(\forall\text{-}\triangleright l\ \not\vdash) & T \not\models \exists x.\, (x[\mathbf{0}] \triangleright \mathsf{F}) \\
 & T \models (\forall x.\, x[\mathbf{0}]) \triangleright \mathsf{F} \\
(\exists\text{-}\triangleright r\ \not\vdash) & T \not\models \exists x.\, (\mathsf{T} \triangleright \neg \copyright x) \\
 & T \models \mathsf{T} \triangleright (\exists x.\, \neg \copyright x)
\end{array}
$$

The table above shows that $\forall$-$\exists$ extrusion is not trivial in SL, but it does not prove it to be impossible (for example, simple double-implication rules for $\exists$-И and $\forall$-И do exist); the actual impossibility proof will come later.

Similar rules, riddled with single implications, govern the extrusion of hiding quantifiers and of $\circledR$. In this case as well, we will show later that they cannot be adjusted.

The situation looks very similar for the freshness quantifier (Table 4.2), apart from the fact that, thanks to its self-duality, we only need half of the rules.

Table 4.2. *Extrusion of freshness quantifier*

$x \notin fv(B)$	$(\text{И}x.\, A) \wedge B$	$\dashv\vdash$	$\text{И}x.\, (A \wedge B)$	$(\text{И}\text{-}\wedge)$
	$\neg(\text{И}x.\, A)$	$\dashv\vdash$	$\text{И}x.\, (\neg A)$	$(\text{И}\text{-}\neg)$
$y \neq \eta$	$\eta[\text{И}y.\, A]$	$\dashv\vdash$	$\text{И}y.\, (\eta[A])$	$(\text{И}\text{-}[])$
$x \notin fv(B)$	$(\text{И}x.\, A) \mid B$	$\dashv\vdash$	$\text{И}x.\, (A \mid B)$	$(\text{И}\text{-}\mid)$
$y \neq x$	$\exists x.\, \text{И}y.\, A$	$\vdash$	$\text{И}y.\, (\exists x.\, A)$	$(\text{И}\text{-}\exists\ \vdash)$
$y \neq \eta$	$\eta \circledR \text{И}y.\, A$	$\dashv\vdash$	$\text{И}y.\, (\eta \circledR A)$	$(\text{И}\text{-}\circledR)$
$x \notin fv(B)$	$(\text{И}x.\, A) \triangleright B$	$\dashv$	$\text{И}x.\, (A \triangleright B)$	$(\text{И}\text{-}\triangleright l\ \dashv)$
$x \notin fv(A)$	$A \triangleright (\text{И}x.\, B)$	$\dashv$	$\text{И}x.\, (A \triangleright B)$	$(\text{И}\text{-}\triangleright r\ \dashv)$
$y \neq \eta$	$(\text{И}y.\, A) @ \eta$	$\dashv\vdash$	$\text{И}y.\, (A @ \eta)$	$(\text{И}\text{-}@)$
$y \neq \eta$	$(\text{И}y.\, A) \oslash \eta$	$\dashv\vdash$	$\text{И}y.\, (A \oslash \eta)$	$(\text{И}\text{-}\oslash)$

And, once more, all the single implications are strict.

(И-∃ $\not\dashv$) $\quad (\nu n)\, n[\mathbf{0}] \not\models \exists x.\, \text{И} y.\, y \circledR \copyright x$

$(\nu n)\, n[\mathbf{0}] \models \text{И} y.\, \exists x.\, y \circledR \copyright x$

(И-$\triangleright l$ $\not\vdash$) $\quad T \not\models \text{И} x.\, (\copyright x \triangleright \mathsf{F})$

$\Leftrightarrow \neg(\forall n.\ n \notin fn(T) \Rightarrow T \models \copyright n \triangleright \mathsf{F})$

$\Leftrightarrow \neg(\forall n.\ n \notin fn(T) \Rightarrow \forall U.\ U \models \copyright n \Rightarrow T \,|\, U \models \mathsf{F})$

$\Leftrightarrow \neg(\forall n.\ n \notin fn(T) \Rightarrow \forall U.\ U \models \neg \copyright n)$

$\Leftrightarrow \exists n.\ n \notin fn(T) \wedge \exists U.\ U \models \copyright n$

consider $U = n[\mathbf{0}]$

$T \models (\text{И} x.\, \copyright x) \triangleright \mathsf{F}$

$\Leftrightarrow \forall U.\ U \models \text{И} x.\, \copyright x \Rightarrow T \,|\, U \models \mathsf{F}$

$\Leftrightarrow \forall U.\ U \not\models \text{И} x.\, \copyright x$

$\Leftrightarrow \forall U, n.\ n \notin fn(U) \Rightarrow U \not\models \copyright n$

$\Leftrightarrow \forall U, n.\ n \notin fn(U) \Rightarrow U \models \neg \copyright n$

(И-$\triangleright r$ $\not\vdash$) $\quad T \not\models \text{И} x.\, (\mathsf{T} \triangleright \neg \copyright x)$

$\Leftrightarrow \neg(\forall n.\ n \notin fn(T) \Rightarrow T \models \mathsf{T} \triangleright \neg \copyright n)$

$\Leftrightarrow \neg(\forall n, U.\ n \notin fn(T) \Rightarrow T \,|\, U \models \neg \copyright n)$

$\Leftrightarrow \exists n, U.\ n \notin fn(T) \wedge T \,|\, U \models \copyright n$

consider $U = n[\mathbf{0}]$

$T \models \mathsf{T} \triangleright (\text{И} x.\, \neg \copyright x)$

$\Leftrightarrow \forall U.\ T \,|\, U \models \text{И} x.\, \neg \copyright x$

$\Leftrightarrow \forall U, n.\ n \notin fn(T, U) \Rightarrow T \,|\, U \models \neg \copyright n$

However, the three single-implication rules admit a double-implication version, as shown in the following table.

Table 4.3. *Extrusion of freshness quantifier - part two*

$x \neq y$	$\exists x.\, \text{И} y.\, A$	$\dashv\vdash$	$\text{И} y.\, (\exists x.\, A \wedge x \neq y)$	(И-∃)
$y \notin fv(B)$	$(\text{И} y.\, A) \triangleright B$	$\dashv\vdash$	$\text{И} y.\, ((\neg \copyright y \wedge A) \triangleright B)$	(И-$\triangleright l$)
$y \notin fv(A)$	$A \triangleright (\text{И} y.\, B)$	$\dashv\vdash$	$\text{И} y.\, ((\neg \copyright y \wedge A) \triangleright B)$	(И-$\triangleright r$)

The last two rules are bizarre: regardless of which side (of $\triangleright$) И is extruded from, y must always be excluded from the left hand side. The proof of the next Lemma shows that this is indeed the case.

Lemma 4.6 (Extrusion of freshness) *There is an algorithm to transform any formula in the full logic in an equivalent formula in* И*-prenex form.*

Proof The algorithm exhaustively applies the double-implication rules of Tables 4.2 and 4.3, left to right, until possible. The result is equivalent to the original formula thanks to Lemma 3.7. Termination is easy.

To prove the correctness of the rules, we must prove that any ground instance of the left hand side is equivalent to the corresponding instance of the right hand side. We assume that ρ is an arbitrary ground substitution defined on all the free variables of the involved formulas (in all the rules both sides have the same free variables). We will also assume that all bound variables in A (and in B) are different, and that ρ is not defined on those variables (hence, in all cases below we assume that ρ is not defined on either x or y).

In the proof we will make extensive use of Corollary 3.13, which expresses the fundamental semantic property of the Gabbay-Pitts freshness quantifier. (И-$\wedge$), (И-$\neg$), (И-$|$): see [11].

(И-[]): assume $y \neq \eta$, hence $y \neq (\eta\rho)$.

$T \models (\eta[\text{И}y.\, A])\rho \Leftrightarrow$
$T \models \eta\rho[\text{И}y.\, A\rho] \Leftrightarrow$
$\exists T'.\, T \equiv \eta\rho[T'] \;\wedge\; T' \models \text{И}y.\, A\rho \Leftrightarrow$
$\exists T'.\, T \equiv \eta\rho[T'] \;\wedge\; \exists m \notin fn(T', A\rho).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
By Corollary 3.13
$\exists T'.\, T \equiv \eta\rho[T'] \;\wedge\; \exists m \notin (fn(T', A\rho) \cup \{\eta\rho\}).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
By Lemma 2.2, we have that $fn(T', A\rho) \cup \{\eta\rho\} = fn(T, \eta\rho, A\rho)$
$\exists T'.\, T \equiv \eta\rho[T'] \;\wedge\; \exists m \notin fn(T, \eta\rho, A\rho).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\, \exists T'.\, T \equiv \eta\rho[T'] \;\wedge\; T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\, T \models \eta\rho[A\rho\{y \leftarrow m\}] \Leftrightarrow$ (By $y \neq \eta\rho$)
$\exists m \notin fn(T, \eta\rho, A\rho).\, T \models (\eta\rho[A\rho])\{y \leftarrow m\} \Leftrightarrow$
$T \models \text{И}y.\, \eta\rho[A\rho] \Leftrightarrow$
$T \models (\text{И}y.\, \eta[A])\rho$

(И-®) with $y \neq \eta\rho$
$T \models (\eta ® \text{И}y.\, A)\rho \Leftrightarrow$
$T \models \eta\rho ® \text{И}y.\, A\rho \Leftrightarrow$
$\exists T'.\, T \equiv (\nu\eta\rho)\, T' \;\wedge\; T' \models \text{И}y.\, A\rho \Leftrightarrow$
$\exists T'.\, T \equiv (\nu\eta\rho)\, T' \;\wedge\; \exists m \notin fn(T', A\rho).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
By Corollary 3.13
$\exists T'.\, T \equiv (\nu\eta\rho)\, T' \;\wedge\; \exists m \notin (fn(T', A\rho) \cup \{\eta\rho\}).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
By Lemma 2.2, we have that $fn(T') \cup \{\eta\rho\} = fn(T, \eta\rho)$
$\exists T'.\, T \equiv (\nu\eta\rho)\, T' \;\wedge\; \exists m \notin fn(T, \eta\rho, A\rho).\, T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\, \exists T'.\, T \equiv (\nu\eta\rho)\, T' \;\wedge\; T' \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\, T \models \eta\rho ® A\rho\{y \leftarrow m\} \Leftrightarrow$ (By $y \neq \eta\rho$)
$\exists m \notin fn(T, \eta\rho, A\rho).\, T \models (\eta\rho ® A\rho)\{y \leftarrow m\} \Leftrightarrow$
$T \models \text{И}y.\, \eta\rho ® A\rho \Leftrightarrow$
$T \models (\text{И}y.\, \eta ® A)\rho$

(И-@) with $y \neq \eta\rho$
$T \models (\text{И}y.\, A@\eta)\rho \Leftrightarrow$
$T \models \text{И}y.\, A\rho@\eta\rho \Leftrightarrow$
$\eta\rho[T] \models \text{И}y.\, A\rho \Leftrightarrow$

$\exists m \notin fn(\eta\rho[T], A\rho).\ \eta\rho[T] \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\ T \models A\rho\{y \leftarrow m\}@\eta\rho \Leftrightarrow$ (By $y \neq \eta\rho$)
$\exists m \notin fn(T, \eta\rho, A\rho).\ T \models (A\rho@\eta\rho)\{y \leftarrow m\} \Leftrightarrow$
$T \models \text{И}y.\ A\rho@\eta\rho \Leftrightarrow$
$T \models (\text{И}y.\ A@\eta)\rho$

(И-$\oslash$) with $y \neq \eta\rho$
$T \models ((\text{И}y.\ A)\oslash\eta)\rho \Leftrightarrow$
$T \models (\text{И}y.\ A\rho)\oslash\eta\rho \Leftrightarrow$
$(\nu\eta\rho)\, T \models (\text{И}y.\ A\rho) \Leftrightarrow$
$\exists m \notin fn((\nu\eta\rho)\, T, A\rho).\ (\nu\eta\rho)\, T \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin (fn((\nu\eta\rho)\, T, A\rho) \cup \{\eta\rho\}).\ (\nu\eta\rho)\, T \models A\rho\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \eta\rho, A\rho).\ T \models A\rho\{y \leftarrow m\}\oslash\eta\rho \Leftrightarrow$ (By $y \neq \eta\rho$)
$\exists m \notin fn(T, \eta\rho, A\rho).\ T \models (A\rho\oslash\eta\rho)\{y \leftarrow m\} \Leftrightarrow$
$T \models \text{И}y.\ A\rho\oslash\eta\rho \Leftrightarrow$
$T \models (\text{И}y.\ A\oslash\eta)\rho$

(И-$\exists$) Assume $x \neq y$. In the following proof, we use A^y_x, A^m_x, A^y_n, and A^m_n, to abbreviate $A\rho$, $A\rho\{y \leftarrow m\}$, $A\rho\{x \leftarrow n\}$, and $A\rho\{x \leftarrow n\}\{y \leftarrow m\}$, respectively. $A\rho\{x \leftarrow n\}\{y \leftarrow m\}$ and $A\rho\{y \leftarrow m\}\{x \leftarrow n\}$ are equal by $x \neq y$, hence are both abbreviated as A^m_n.

$T \models (\exists x.\ \text{И}y.\ A)\rho \Leftrightarrow$
$T \models \exists x.\ \text{И}y.\ A^y_x \Leftrightarrow$
$\exists n.\ T \models \text{И}y.\ A^y_n \Leftrightarrow$
$\exists n.\ \exists m \notin fn(T, A^y_n).\ T \models A^m_n \Leftrightarrow$ (Corollary 3.13)
$\exists n.\ \exists m \notin (fn(T, A^y_x) \cup \{n\}).\ T \models A^m_n \Leftrightarrow$
$\exists m \notin fn(T, A^y_x).\ \exists n.\ m \neq n \wedge T \models A^m_n \Leftrightarrow$
$\exists m \notin fn(T, A^y_x).\ \exists n.\ T \models (A^m_n \wedge m \neq n) \Leftrightarrow$
$\exists m \notin fn(T, A^y_x).\ T \models \exists x.\ (A^m_x \wedge m \neq x) \Leftrightarrow$
$\exists m \notin fn(T, A^y_x).\ T \models (\exists x.\ (A^y_x \wedge y \neq x))\{y \leftarrow m\} \Leftrightarrow$
$\exists m \notin fn(T, \exists x.\ (A^y_x \wedge y \neq x)).\ T \models (\exists x.\ (A^y_x \wedge y \neq x))\{y \leftarrow m\} \Leftrightarrow$
$T \models \text{И}y.\ (\exists x.\ A\rho \wedge x \neq y) \Leftrightarrow$
$T \models (\text{И}y.\ (\exists x.\ A \wedge x \neq y))\rho$

(И-$\triangleright l$) Assume $y \notin fv(B)$.

$T \models ((\text{И}y.\ A) \triangleright B)\rho \Leftrightarrow$
$T \models (\text{И}y.\ A\rho) \triangleright B\rho \Leftrightarrow$
$\forall T'.\ T' \models \text{И}y.\ A\rho \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$
$\forall T'.\ (\exists n \notin fn(T', A\rho).\ T' \models A\rho\{y \leftarrow n\}) \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$
$\forall T'.\ (\exists n \notin fn(T, T', A\rho, B\rho).\ T' \models A\rho\{y \leftarrow n\}) \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$
$\forall T'.\ \forall n \notin fn(T, T', A\rho, B\rho).\ (T' \models A\rho\{y \leftarrow n\} \ \Rightarrow\ T \mid T' \models B\rho) \Leftrightarrow$
$\forall n \notin fn(T, A\rho, B\rho).$

$$\forall T'.\ n \notin fn(T') \Rightarrow (T' \models A\rho\{y \leftarrow n\}) \ \Rightarrow\ T \mid T' \models B\rho) \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).$$
$$\forall T'.\ (n \notin fn(T')\ \wedge\ T' \models A\rho\{y \leftarrow n\}) \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ \forall T'.\ (T' \models \neg \copyright n \wedge A\rho\{y \leftarrow n\}) \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ \forall T'.\ T' \models (\neg \copyright y \wedge A\rho)\{y \leftarrow n\} \ \Rightarrow\ T \mid T' \models B\rho \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ T \models (\neg \copyright y \wedge A\rho)\{y \leftarrow n\} \triangleright B\rho \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ T \models ((\neg \copyright y \wedge A\rho) \triangleright B\rho)\{y \leftarrow n\} \Leftrightarrow$$
$$T \models \text{И} y.\, ((\neg \copyright y \wedge A\rho) \triangleright B\rho)$$
$$T \models (\text{И} y.\, (\neg \copyright y \wedge A) \triangleright B)\rho$$

(И-$\triangleright r$) Assume $y \notin fv(A)$.

$$T \models (A \triangleright \text{И} y.\, B)\rho \Leftrightarrow$$
$$T \models A\rho \triangleright \text{И} y.\, B\rho \Leftrightarrow$$
$$\forall T'.\ T' \models A\rho \ \Rightarrow\ T \mid T' \models \text{И} y.\, B\rho \Leftrightarrow$$
$$\forall T'.\ T' \models A\rho \ \Rightarrow\ (\forall n \notin fn(T, T', B\rho).\ T \mid T' \models B\rho\{y \leftarrow n\}) \Leftrightarrow \text{(Cor. 3.13)}$$
$$\forall T'.\ T' \models A\rho \ \Rightarrow\ (\forall n \notin fn(T, T', A\rho, B\rho).\ T \mid T' \models B\rho\{y \leftarrow n\}) \Leftrightarrow$$
$$\forall T'.\ \forall n \notin fn(T, T', A\rho, B\rho).\ (T' \models A\rho \ \Rightarrow\ T \mid T' \models B\rho\{y \leftarrow n\}) \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ \forall T'.\ n \notin fn(T') \Rightarrow (T' \models A\rho \ \Rightarrow\ T \mid T' \models B\rho\{y \leftarrow n\}) \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ \forall T'.\ (n \notin fn(T') \wedge T' \models A\rho) \ \Rightarrow\ T \mid T' \models B\rho\{y \leftarrow n\} \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ \forall T'.\ (T' \models \neg \copyright n \wedge A\rho) \ \Rightarrow\ T \mid T' \models B\rho\{y \leftarrow n\} \Leftrightarrow$$
$$\forall n \notin fn(T, A\rho, B\rho).\ T \models (\neg \copyright n \wedge A\rho) \triangleright (B\rho\{y \leftarrow n\}) \Leftrightarrow \text{(By } y \notin fv(A))$$
$$\forall n \notin fn(T, A\rho, B\rho).\ T \models (\neg \copyright y \wedge A\rho \triangleright B\rho)\{y \leftarrow n\} \Leftrightarrow$$
$$T \models \text{И} y.\, ((\neg \copyright y \wedge A\rho) \triangleright B\rho)$$
$$T \models (\text{И} y.\, (\neg \copyright y \wedge A) \triangleright B)\rho$$

□

We now use this result to prove decidability of the freshness quantifier.

4.3 Decidable Sublogics With Quantifiers

We first observe that *model-checking* is decidable for prenex logics; of course, this is not true, in general, for validity.

Theorem 4.7 (Decidability of Prenex Model-Checking) *Model-checking is decidable for the closed formulas F generated by the following grammar ($\exists$, Н, Ⓡ, И : outermost only, ©, ⊘: unlimited):*

$$F ::= \exists x.\, F \mid x \circledR F \mid \text{Н} x.\, F \mid \text{И} x.\, F \mid \neg F \mid A$$
$$A ::= \mathbf{0} \mid \eta[A] \mid A \mid A \mid A \wedge A \mid \neg A \mid \copyright \eta \mid A \triangleright A \mid A @ \eta \mid A \oslash \eta$$

is decidable over all trees.

Proof By induction on the size of F and by cases.
Case $\neg F$ is trivial induction.

Case A is Corollary 4.5.

To model-check $T \models \exists x. F$, consider a finite set $\mathbf{N}$ of names containing $fn(T, F)$ plus one more fresh name m and model-check $T \models F\{x \leftarrow n\}$ for $n \in \mathbf{N}$. No other name needs to be considered, by Lemma 3.11.

To model-check $T \models n®F$, transform T in ENF and apply Lemma 3.14.

To model-check $T \models \mathsf{H}x. F$, transform T in ENF and apply Lemma 3.16.

To model-check $T \models \text{И}x. F$, choose a name $n \notin fn(T, F)$ and model-check $T \models F\{x \leftarrow n\}$. The result does not depend on the name by Corollary 3.13. $\square$

Now, we can prove that the addition of freshness preserves the decidability of the logic of Corollary 4.5.

Theorem 4.8 (Decidability of Fresh Quantifiers) *Model-checking and validity are decidable for the closed formulas generated by the following grammar ($\exists, \mathsf{H}, ®$: no, ©, $\oslash$, И : yes):*

$$A ::= \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid \text{И}x. A \mid A \wedge A \mid \neg A \mid ©\eta \mid A \triangleright A \mid A@\eta \mid A\oslash\eta$$

are decidable over all trees.

Proof Model-checking: we apply the algorithm of Lemma 4.6 to transform the formula in И-prenex form. This can be model-checked by Theorem 4.7.

Validity: given a formula A, we extrude the freshness quantifier from $\mathsf{T} \triangleright A$, obtaining B. We then model-check $\mathbf{0} \models \vec{\forall} B$, which is decidable by Theorem 4.7. By Lemma 3.5, $\mathbf{0} \models \vec{\forall} B$ iff A is valid. $\square$

To sum up, fresh quantification alone is not enough to lose decidability, even if combined with a limited form of revelation ($©n$).

The proof is based on the possibility of extruding freshness quantifiers through all operators, including the parallel adjunct operator that internalizes validity in the logic and, more essentially, negation. This reveals a deep algebraic difference between fresh and existential quantification, where such extrusion is not possible. We now formalize this fact.

4.4 Impossibility of Extrusion

Theorem 4.7 has the following Corollaries, whose proof is identical to the second part of the proof of Theorem 4.8.

Corollary 4.9 (Extrusion of $\exists$ Implies Decidability) *Consider the following logic:*

$$A ::= \exists x. A \mid \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid \text{И}x. A \mid A \wedge A \mid \neg A \mid ©\eta \mid A \triangleright A \mid A@\eta \mid A\oslash\eta$$

If there exists an extrusion algorithm for this logic, i.e. an algorithm that transform every formula into an equivalent formula generated by the grammar of Theorem 4.7, then the logic is decidable.

Corollary 4.10 (Extrusion of Revelation Implies Decidability) *Consider the following logic:*

$$A ::= x \circledR A \mid \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid \text{И}x.\, A \mid A \wedge A \mid \neg A \mid \circledc \eta \mid A \triangleright A \mid A@\eta \mid A \oslash \eta$$

If there exists an extrusion algorithm for the revelation operator, i.e. an algorithm that transform every formula into an equivalent formula generated by the grammar of Theorem 4.7, then the logic is decidable.

Corollary 4.11 (Extrusion of Hiding Implies Decidability) *Consider the following logic:*

$$A ::= \mathsf{H}x.\, A \mid \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid \text{И}x.\, A \mid A \wedge A \mid \neg A \mid \circledc \eta \mid A \triangleright A \mid A@\eta \mid A \oslash \eta$$

If there exists an extrusion algorithm for the hiding quantifier, i.e. an algorithm that transform every formula into an equivalent formula generated by the grammar of Theorem 4.7, then the logic is decidable.

By Fact 5.1, Corollary 5.28 and Corollary 5.33, the three logics of Corollaries 4.9, 4.10, 4.11 are all undecidable. Hence, we have the following Corollary.

Corollary 4.12 (No Extrusion) *No extrusion algorithm (as defined in Corollaries 4.9, 4.10, 4.11) exists for the existential quantifier, the revelation operator, and the hiding quantifier.*

5 Undecidable Logics

5.1 Known Results

Fact 5.1 (Undecidability of Existential Quantification) *Validity of closed formulas built from $\exists x.\, A$, $A \wedge A$, $\neg A$, $x[A] \,|\, \mathsf{T}$ is not decidable.*

Proof Proved in [13], by encoding any first-order formula whose vocabulary is just a binary relation in the fragment above. The undecidability over finite trees follows by Trakhtenbrot Theorem. The undecidability over infinite trees follows by Church-Turing Theorem. □

5.2 Standard Model

In this section we focus on a tiny sublogic of SL that contains the revelation operator and show that for each formula A of that sublogic, when a tree T satisfies A, there exists a cut-down version of T that satisfies the same formula. This will be a key technical tool in order to prove that the decidability of this tiny logic is already as hard as decidability of first order logic.

Notation 5.2 (Path-Formulas) *A path-formula p is a formula that denotes the existence of a path of edges, starting from the root and leading to a leaf, as* *follows (we only define path formulas of length one and two, since we need no more).*

$$\begin{array}{llll} \textit{length one:} & .\eta & =_{def} & \eta[\mathbf{0}] \mid \mathsf{T} \\ \textit{length two:} & .\eta'.\eta & =_{def} & \eta'[\eta[\mathbf{0}] \mid \mathsf{T}] \mid \mathsf{T} \end{array}$$

When a tree satisfies $.m.n$ we say that it "contains a path $m.n$"; observe that the path must end with a leaf. The minimal tree containing such path, $m[n[\mathbf{0}]]$, is called a "line for the path $m.n$", and similarly $m[\mathbf{0}]$ is a line for m.

We now introduce a notion of path cutting. Intuitively, the tree $Cut_N(T)$ contains one line for each of those paths $m.n$ of T such that m and n are either bound or in N. In this way, for any formula A with shape $.n.m$, $n® m®.n.m$, $n®.n.m$, $Cut_{nm(A)}(T)$ is A-equivalent to T, i.e. $Cut_{nm(A)}(T) \models A$ iff $T \models A$. Moreover, $Cut_N(T)$ contains a list $n_1[\mathbf{0}] \mid \ldots \mid n_j[\mathbf{0}]$, where $\{n_i\}^{i \in I} = fn(T) \cap N$, so that the validity of formulas $n®\mathsf{T}$, for $n \in N$, is preserved as well.

We will prove that this cut-down structure is logically equivalent to the original tree, with respect to those formulas that only contain path-formulas of length 2 and names that are in N (Corollary 5.17).

Before giving the formal definition, we give some examples. In the examples below, n, m, p abbreviate $n[\mathbf{0}], m[\mathbf{0}], p[\mathbf{0}]$. Cutting is only defined up-to-congruence.

flattening	$Cut_{\{n,m\}}(n[m \mid n])$	$\equiv$	$n[m] \vert n[n] \vert n \vert m$
cutting long paths	$Cut_{\{n,m\}}(n[m[n]])$	$\equiv$	$n \vert m$
cutting w.r.t. more names	$Cut_{\{n,m,p\}}(n[m \mid n])$	$\equiv$	$n[m] \vert n[n] \vert n \vert m$
deleting free names	$Cut_{\{n\}}(n[m \mid n])$	$\equiv$	$n[n] \vert n$
preserving bound names	$Cut_{\{n\}}((\nu m)\, n[m \mid n])$	$\equiv$	$(\nu m)\, n[m] \vert n[n] \vert n \vert m$
name clashes don't matter	$Cut_{\{n,m\}}((\nu m)\, n[m \mid n])$	$\equiv$	$(\nu m)\, n[m] \vert n[n] \vert n \vert m$
preserving the name m	$Cut_{\{n,m\}}(n[n] \mid m[p])$	$\equiv$	$n[n] \vert n \vert m$

We first define an auxiliary partial function $enfCut_N(T)$, that is only defined on trees in ENF, and is deterministic (while $Cut_N(T)$ is total but is defined only up to congruence). $enfCut_N(T)$ behaves as $Cut_N(T)$ in all the examples above. Then we define $Cut_N(T)$ by closing the partial function $enfCut_N(T)$ with respect to tree equivalence.

Definition 5.3 (Path cutting for ENF) *For each tree $(\nu m_1) \ldots (\nu m_j)\, U$ in ENF, where U is the matrix, for each set of names $N = \{n_i : i \in I\}$ we define the operation $enfCut_N()$ as follows. $Par\{T : cond\}$ combines (using $\mid$) all instances $T\sigma$ of T such that the corresponding instance of "cond" is satisfied.*

$$\begin{array}{l} enfCut_N((\nu m)\, T) \\ \quad =_{def} \quad (\nu m)\, enfCut_{N \cup \{m\}}(U) \\ enfCut_N(U) \quad \textit{(where } U \textit{ contains no } (\nu n)\, A' \textit{ subterm)} \\ \quad =_{def} \quad Par\{n_1[n_2[\mathbf{0}]] : U \models .n_1.n_2,\ \{n_1, n_2\} \subseteq N\} \\ \qquad\qquad \mid Par\{n[\mathbf{0}] : n \in (fn(U) \cap N)\} \end{array}$$

Remark 5.4 (Cutting, Reordering, and Renaming) *Some remarks about cutting:*

1. *In definition 5.3 we do not require that* $\{m_j\}$ *and* N *are disjoint. This is exploited in Corollary 5.10.*

2. $enfCut_N((\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\, U)$*, i.e.* $(\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\, enfCut_{N\cup\{n_1,\ldots,n_j\}}(U)$*, is always in ENF: all the names in* $\{n_i\}^{i\in\{1,\ldots,j\}}$ *are free in* U*, hence they appear in an edge* $n_i[\mathbf{0}]$ *of the result.*

3. $fn(enfCut_N(T)) = fn(T) \cap N$

4. *If* T' *has the same matrix of* T *and a reordered prefix, then* $enfCut_N(T')$ *has the same matrix as* $enfCut_N(T)$ *(modulo the edge order, which is not fixed) but has the prefix of* T'*. In other terms, when the prefix of the input of* $enfCut_N()$ *is reordered, the prefix of the output is reordered in the same way.*

Lemma 5.5 (Congruence of $enfCut_N()$**)** *if* T *and* T' *are in ENF then*

$$T \equiv T' \;\Rightarrow\; enfCut_N(T) \equiv enfCut_N(T')$$

Proof We consider the case when $N \subseteq fn(T)$, hence, by congruence, $N \subseteq fn(T')$. The general case follows immediately by observing that $enfCut_N(T) = enfCut_{N\cap fn(T)}(T)$.

By Lemma 2.6, $T \equiv T'$ implies that exist U, T'', U'', U', $\{n_i\}^{i\in\{1..j\}}$, $\{n'_i\}^{i\in\{1..j\}}$, and a bijection τ, such that all the U's are restriction-free and

$$\begin{array}{rcl} T & = & (\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\, U \\ T'' & = & (\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\, U'' \\ T' & = & (\vec{\nu}_{i\in\{1,\ldots,j\}} n'_i)\, U' \end{array}$$

with

- $N \# \{n_i\}^{i\in\{1..j\}}$ and $N \# \{n'_i\}^{i\in\{1..j\}}$, by $N \subseteq fn(T) = fn(T')$;
- $U \equiv U''$;
- $U'' = U'\{n'_l \leftarrow \tau(n_l)\}^{l\in\{1..j\}}$.

$U \equiv U''$ implies that $enfCut_N(U) \equiv enfCut_N(U'')$, since $fn(U) = fn(U'')$ and $U \models .n_1.n_2$ iff $U'' \models .n_1.n_2$, hence $enfCut_N(T) \equiv enfCut_N(T'')$.

Similarly, $U'' = U'\{n'_l \leftarrow \tau(n_l)\}^{l\in\{1..j\}}$. implies that

$$\begin{array}{l} enfCut_{N\cup\{n_i\}^{i\in\{1..j\}}}(U'') \\ = enfCut_{N\cup\{n_i\}^{i\in\{1..j\}}}(U'\{n'_l \leftarrow \tau(n_l)\}^{l\in\{1..j\}}) \\ = (enfCut_{N\cup\{n'_i\}^{i\in\{1..j\}}}(U'))\{n'_l \leftarrow \tau(n_l)\}^{l\in\{1..j\}} \end{array}$$

hence

$$
\begin{aligned}
&\mathit{enfCut}_N(T'') = (\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\ \mathit{enfCut}_{N\cup\{n_i\}^{i\in\{1..j\}}}(U'')\\
&= (\vec{\nu}_{i\in\{1,\ldots,j\}} n_i)\ (\mathit{enfCut}_{N\cup\{n'_i\}^{i\in\{1..j\}}}(U')\{n'_l \leftarrow \tau(n_l)\}^{l\in\{1..j\}})\\
&\equiv (\vec{\nu}_{i\in\{1,\ldots,j\}} n'_i)\ \mathit{enfCut}_{N\cup\{n'_i\}^{i\in\{1..j\}}}(U')\\
&= \mathit{enfCut}_N((\vec{\nu}_{i\in\{1,\ldots,j\}} n'_i)\, U')\\
&= \mathit{enfCut}_N(T')
\end{aligned}
$$

□

We now extend cutting from ENF to general terms.

Definition 5.6

$$\mathit{Cut}_N(T) \ \triangleq\ \{\mathit{enfCut}_N(U) : U \in \mathit{ENF}(T)\}$$

Corollary 5.7 (Congruence of $\mathit{Cut}_N()$)

$$T \equiv T' \ \wedge\ U \in \mathit{Cut}_N(T) \ \wedge\ U' \in \mathit{Cut}_N(T') \ \Rightarrow\ U \equiv U'$$

Proof By definition,

$$
\begin{array}{ll}
U \in \mathit{Cut}_N(T) \ \wedge\ U' \in \mathit{Cut}_N(T') \ \Rightarrow & \\
\exists \bar{T}, \bar{T}'. & \bar{T} \in \mathit{ENF}(T),\ \mathit{enfCut}_N(\bar{T}) = U\\
& \bar{T}' \in \mathit{ENF}(T'),\ \mathit{enfCut}_N(\bar{T}') = U'
\end{array}
$$

By transitivity, $\bar{T} \equiv \bar{T}'$. By Lemma 5.5, $U \equiv U'$.

□

Because of the property above, hereafter we will always write, with a slight notational abuse, $T' = \mathit{Cut}_N(T)$ or $\mathit{Cut}_N(T) = T'$, instead of $T' \in \mathit{Cut}_N(T)$, i.e. we will have $\mathit{Cut}_N(T)$ standing for an arbitrary element of the set, whenever we are only interested in its value modulo congruence.

Lemma 5.8

$$\mathit{fn}(\mathit{Cut}_N(T)) = \mathit{fn}(T) \cap N$$

Lemma 5.9

$$\mathit{Cut}_N((\nu n)\, T) \equiv (\nu n)\ \mathit{Cut}_{N\cup\{n\}}(T)$$

Proof If $n \notin \mathit{fn}(T)$, then it neither appears free in $\mathit{Cut}_{N\cup\{n\}}(T)$, hence the property holds trivially:

$$\mathit{Cut}_N((\nu n)\, T) \equiv \mathit{Cut}_N(T) \equiv \mathit{Cut}_{N\cup\{n\}}(T) \equiv (\nu n)\ \mathit{Cut}_{N\cup\{n\}}(T).$$

Otherwise, a ENF of $(\nu n)\, T$ is $(\nu n)\, T'$, where T' is a ENF of T, and

$$
\begin{aligned}
&\mathit{Cut}_N((\nu n)\, T) \equiv \mathit{enfCut}_N((\nu n)\, T') \equiv (\nu n)\ \mathit{enfCut}_{N\cup\{n\}}(T')\\
&\equiv (\nu n)\ \mathit{Cut}_{N\cup\{n\}}(T).
\end{aligned}
$$

□

Corollary 5.10

$$n \in N \;\Rightarrow\; Cut_N((\nu n)\, T) \equiv (\nu n)\; Cut_N(T)$$

Lemma 5.11 (Inversion) *For any $n \notin fn(T)$:*

$$Cut_N(T) \equiv (\nu n)\, U' \Rightarrow \exists T'.\; T \equiv (\nu n)\, T' \wedge U' \equiv Cut_{N\cup\{n\}}(T')$$

Proof If $n \notin fn(U')$ the thesis follows immediately with $T' = T$:

$$T \equiv (\nu n)\, T \;\wedge\; U' \equiv (\nu n)\, U' \equiv Cut_N(T) = Cut_{N\cup\{n\}}(T)$$

We consider now the case $n \in fn(U')$.
$Cut_N(T) \equiv (\nu n)\, U'$ means that

$$\exists \bar{T}, \bar{U}.\; \bar{T} \in ENF,\; \bar{U} \in ENF,\; T \equiv \bar{T},\; enfCut_N(\bar{T}) = \bar{U},\; \bar{U} \equiv (\nu n)\, U'$$

Choose a $\bar{\bar{U}} \in ENF$ with $\bar{\bar{U}} \equiv U'$. $\bar{U}$ and $(\nu n)\, \bar{\bar{U}}$ are congruent and ENF, hence they only differ for prefix reordering and renaming, and have equivalent matrixes. By $\bar{U} = enfCut_N(\bar{T})$, if we apply the same reordering and renaming that transform $\bar{U}$ into $(\nu n)\, \bar{\bar{U}}$ to $\bar{T}$, we get $\bar{\bar{T}} \equiv \bar{T}$ such that $enfCut_N(\bar{\bar{T}}) = (\nu n)\, \bar{\bar{U}}$. Hence, $\exists T'.\; \bar{\bar{T}} = (\nu n)\, T'$ and $enfCut_N(T') = \bar{\bar{U}}$. This is the thesis, since $\bar{\bar{U}} \equiv U'$. $\square$

Corollary 5.12 (Inversion with $n \in N$) *If $n \in N$, then:*

$$Cut_N(T) \equiv (\nu n)\, U' \Rightarrow \exists T'.\; T \equiv (\nu n)\, T' \wedge U' \equiv Cut_N(T')$$

Proof $(\nu n)\, U' \in Cut_N(T)$ implies that n is not free in $Cut_N(T)$, hence, by Lemma 5.8 and $n \in N$, $n \notin fn(T)$. Then we apply the lemma above. $\square$

Before proving our key lemma, we introduce the De Morgan dual of revelation. This is useful so that we can distribute negation down to the leaves of any formula in our logic.

Notation 5.13 (Corevelation)

$$\eta \circledU A \quad =_{def} \quad \neg(\eta \circledR \neg A)$$

Lemma 5.14

$$T \models n \circledU A \quad \Leftrightarrow \quad \forall T'.\; T \equiv (\nu n)\, T' \Rightarrow T' \models A$$

Lemma 5.15 (Standard Model) *Let A be a closed formula generated by the following grammar:*

$$A ::= .\eta_1.\eta_2 \mid \neg.\eta_1.\eta_2 \mid A \wedge A \mid A \vee A \mid \eta \circledR A \mid \eta \circledU A \mid \text{И}x.\, A$$

then:

$$T \models A \Rightarrow Cut_{nm(A)}(T) \models A.$$

Proof We prove the following stronger property, that goes better through induction, by induction on the size of A, and by cases:

$$\forall \mathbf{N} \text{ finite. } T \models A \Rightarrow Cut_{nm(A)\cup\mathbf{N}}(T) \models A.$$

If A is a path-formula $.n_1.n_2$ then $n_1[n_2[\mathbf{0}]]$ is in $Cut_{\{n_1,n_2\}\cup\mathbf{N}}(T)$ iff $T \models .n_1.n_2$, by construction. This proves the lemmas for both cases $.n_1.n_2$ and $\neg.n_1.n_2$. Observe that, by the closure hypothesis, we do not consider cases $.x.n$, $.n.x$, $.x.y$.

If $A = A' \wedge A''$, or $A = A' \vee A''$, the thesis follows by induction.

If $A = n\circledR A'$, then:

$$\begin{array}{ll}
T \models n\circledR A' & \Leftrightarrow \text{def of } \circledR \\
\exists T'.\, T \equiv (\nu n)\,T',\ T' \models A' & \Rightarrow \text{by ind.} \\
\exists T'.\, T \equiv (\nu n)\,T',\ \forall \mathbf{M}.\ Cut_{nm(A')\cup\mathbf{M}}(T') \models A' & \Rightarrow \mathbf{M} \leftarrow \mathbf{N} \cup \{n\} \\
\exists T'.\, T \equiv (\nu n)\,T',\ \forall \mathbf{N}.\ Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T') \models A' & \Rightarrow \text{def of } \circledR \\
\exists T'.\, T \equiv (\nu n)\,T',\ \forall \mathbf{N}.\ (\nu n)\ Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T') \models n\circledR A' & \Leftrightarrow \text{by Cor. 5.10} \\
\exists T'.\, T \equiv (\nu n)\,T',\ \forall \mathbf{N}.\ Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}((\nu n)\,T') \models n\circledR A' & \Rightarrow \\
\forall \mathbf{N}.\ \exists T'.\, T \equiv (\nu n)\,T',\ Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}((\nu n)\,T') \models n\circledR A' & \Leftrightarrow \text{by } T \equiv (\nu n)\,T' \\
\forall \mathbf{N}.\ Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T) \models n\circledR A' & \Leftrightarrow \\
\forall \mathbf{N}.\ Cut_{nm(n\circledR A')\cup\mathbf{N}}(T) \models n\circledR A' &
\end{array}$$

Now, assume $A = nⓊA'$ and $T \models A$. We want to prove that.:

$$\begin{array}{ll}
\forall \mathbf{N}.\ Cut_{nm(nⓊA')\cup\mathbf{N}}(T) \models nⓊA' & i.e. \\
\forall \mathbf{N}, T'.\ (\nu n)\,T' \equiv Cut_{nm(nⓊA')\cup\mathbf{N}}(T) \Rightarrow T' \models A' & (1)
\end{array}$$

We assumed that $T \models nⓊA'$ i.e. $\forall T''.\ T \equiv (\nu n)\,T'' \Rightarrow T'' \models A'$; by induction:

$$\forall T'', \mathbf{M}.\ T \equiv (\nu n)\,T'' \Rightarrow Cut_{nm(A')\cup\mathbf{M}}(T'') \models A' \quad (2)$$

To prove (1), we assume $(\nu n)\,T' \equiv Cut_{nm(nⓊA')\cup\mathbf{N}}(T)$ (a).

Since $n \in nm(nⓊA') \cup \mathbf{N}$, we can apply Corollary 5.12 to (a), obtaining that:

$$\begin{array}{ll}
\exists T'''. & T \equiv (\nu n)\,T''' \\
& \wedge\ T' \equiv Cut_{nm(nⓊA')\cup\mathbf{N}}(T''') = Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T''') \quad (3)
\end{array}$$

Hence we can apply (2), with $T'' \leftarrow T'''$ and $\mathbf{M} \leftarrow \mathbf{N} \cup \{n\}$, obtaining

$$Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T''') \models A',$$

which implies the thesis $T' \models A'$, since, by (3)

$$T' \equiv Cut_{nm(A')\cup\mathbf{N}\cup\{n\}}(T''').$$

If $A = \text{И}x.\, A'$, then:

$$
\begin{array}{ll}
T \models \text{И}x.\, A' & \Leftrightarrow \text{def of } \text{И} \\
\forall \mathbf{N}.\ \exists n.\ n \notin (fn(T, A') \cup \mathbf{N}) & \\
\qquad \wedge\ T \models A'\{x \leftarrow n\} & \Rightarrow \text{by ind.} \\
\forall \mathbf{N}.\ \exists n.\ n \notin (fn(T, A') \cup \mathbf{N}) & \\
\qquad \wedge\ \forall \mathbf{M}.\ Cut_{nm(A'\{x \leftarrow n\}) \cup \mathbf{M}}(T) \models A'\{x \leftarrow n\} & \Rightarrow \\
\forall \mathbf{N}.\ \exists n.\ n \notin (fn(T, A') \cup \mathbf{N}) & \\
\qquad \wedge\ Cut_{nm(A'\{x \leftarrow n\}) \cup \mathbf{N}}(T) \models A'\{x \leftarrow n\} & \Leftrightarrow \text{by } n \notin fn(T) \\
\forall \mathbf{N}.\ \exists n.\ n \notin (fn(T, A') \cup \mathbf{N}) & \\
\qquad \wedge\ Cut_{nm(A'\{x \leftarrow n\}) \setminus \{n\} \cup \mathbf{N}}(T) \models A'\{x \leftarrow n\} & \Leftrightarrow \text{by } n \notin nm(A') \\
\forall \mathbf{N}.\ \exists n.\ n \notin (fn(T, A') \cup \mathbf{N}) & \\
\qquad \wedge\ Cut_{nm(\text{И}x.\, A') \cup \mathbf{N}}(T) \models A'\{x \leftarrow n\} & \Leftrightarrow \text{by Cor. 3.13 } (3 \Leftrightarrow 5) \\
\forall \mathbf{N}.\ Cut_{nm(\text{И}x.\, A') \cup \mathbf{N}}(T) \models \text{И}x.\, A' &
\end{array}
$$

□

Corollary 5.16 (Standard Model) *Let A be a closed formula generated by the following grammar:*

$$A ::= .\eta_1.\eta_2 \mid A \wedge A \mid \eta \circledR A \mid \text{И}x.\, A \mid \neg A$$

then:

$$T \models A \Rightarrow Cut_{nm(A)}(T) \models A.$$

Corollary 5.17 *Let A be a closed formula generated by the following grammar:*

$$A ::= .\eta_1.\eta_2 \mid A \wedge A \mid \eta \circledR A \mid \text{И}x.\, A \mid \neg A$$

then:

$$T \models A \Leftrightarrow Cut_{nm(A)}(T) \models A.$$

Proof $T \models A \Rightarrow Cut_{nm(A)}(T) \models A$ by Corollary 5.16.

For the other direction, assume $T \not\models A$; then:

$$
\begin{array}{ll}
T \not\models A & \Leftrightarrow \text{def of } \neg A \\
T \models \neg A & \Rightarrow \text{Corollary 5.16} \\
Cut_{nm(A)}(T) \models \neg A & \Leftrightarrow \text{def of } \neg A \\
Cut_{nm(A)}(T) \not\models A &
\end{array}
$$

□

5.3 Undecidability Results

Since we are studying undecidability, we focus here on weak versions of the logic. We prove undecidability for a logic with just $\wedge$, $\neg$, $\circledR$, and path formulas, hence we show that undecidability comes from revelation, and not from the other spatial operators. The undecidability of any richer logic follows immediately.

We are going to define a translation of FOL formulas into SL formulas, and FOL structures into SL trees, in order to reduce SL satisfiability to FOL satisfiability over a finite domain, which is known to be undecidable.

We first define our specific flavour of FOL. We consider formulas over a vocabulary which only consists of a binary relation R, i.e. formulas generated by the following grammar:

$$\phi \ ::= \ \exists x.\, \phi \mid \phi \wedge \psi \mid \neg\phi \mid R(x, x')$$

We define satisfaction of a formula, over an interpretation consisting of a domain $\mathcal{D}$ and a binary relation $\mathcal{R}$ over $\mathcal{D}$, with respect to a variable assignment σ with $\sigma{\downarrow} \supseteq fv(\phi)$, as follows.

$$\begin{array}{lcl}
\mathcal{D}, \mathcal{R}, \sigma \models \exists x.\, \phi & \Leftrightarrow & \exists c \in \mathcal{D}.\ \mathcal{D}, \mathcal{R}, \sigma[x \mapsto c] \models \phi \\
\mathcal{D}, \mathcal{R}, \sigma \models \phi \wedge \psi & \Leftrightarrow & \mathcal{D}, \mathcal{R}, \sigma \models \phi \ \wedge \ \mathcal{D}, \mathcal{R}, \sigma \models \psi \\
\mathcal{D}, \mathcal{R}, \sigma \models \neg\phi & \Leftrightarrow & \neg(\mathcal{D}, \mathcal{R}, \sigma \models \phi) \\
\mathcal{D}, \mathcal{R}, \sigma \models R(x, x') & \Leftrightarrow & (\sigma(x), \sigma(x')) \in \mathcal{R}
\end{array}$$

Essentially, we will translate a model $\mathcal{D}, \mathcal{R}$ into an ENF term $(\vec{\nu} n_i)\, [\![\mathcal{D}]\!] \mid [\![\mathcal{R}]\!]$, with one name n_i for each element of $\mathcal{D}$, with $\mathcal{R}$ encoded as set of lines of length two, and $\mathcal{D}$ encoded as a set of lines of length one, obtaining structures that have the same shape as the cut-down trees introduced in Section 5.2.

In the formula, we will translate $\exists$ into Ⓡ and $R(x, y)$ into $.m.n$. To translate $\exists$ into Ⓡ, we have to overcome some differences between the two operators. The most obvious difference is the fact that $\exists$ is a binder while Ⓡ is not. In FOL semantics, we associate each variable x that is bound in a formula $\exists x.\phi$ with a value c that is "free" in the domain. In the SL translation this becomes an association between a name m that is free in a formula mⓇA and a name n_i that is *bound* in the model $(\vec{\nu} n_i)\, T$. So, while in FOL we are able to bind variables in the formula with values in the domain, in the SL translation we will rename bound names in the model to have them matching the free names in the formula.

A second difference is the fact that the same value can be bound to two different FOL variables, while the same hidden name cannot be revealed twice, hence,

$$\{(c, c)\} \models \exists x_1.\ \exists x_2.\ R(x_1, x_2)$$

but

$$(\nu n)\, n[n[\mathbf{0}]] \not\models n_1 Ⓡ n_2 Ⓡ .n_1.n_2.$$

We solve this problem by translating

$$\exists x_1.\ \exists x_2.\ \phi$$

as if it were

$$\exists x_1.\ ((\exists x_2 \neq x_1.\ \phi) \vee \phi\{x_2 \leftarrow x_1\}),$$

i.e. as:

$$x_1 Ⓡ ((x_2 Ⓡ [\![\phi]\!]) \vee [\![\phi\{x_2 \leftarrow x_1\}]\!]),$$

so that $\exists x.\ \exists y.\ R(x,y)$ becomes $x\circledR((y\circledR.x.y) \vee .x.x)$ (Actually, it is translated as the closed term $(x\circledR((y\circledR.x.y) \vee .x.x))[x \mapsto m][y \mapsto n]$.)

Finally, while x in $\exists x.\ \phi$ can only be associated to an element that is in the domain, n in $n\circledR A$ can also be associated to a name that does not appear in the model at all (see Lemma 3.14). We solve this problem by translating $\exists x.\phi$ as $x\circledR([\![\phi]\!] \wedge .x)$ and by restricting our attention to models where, for every name n in a term, a line $n[\mathbf{0}]$ is present. We use our results on tree-cutting to show that this restriction is without loss of generality.

We can finally define our translation. We map a formula to a formula, an interpretation $\mathcal{D}, \mathcal{R}$ to a tree $[\![\mathcal{D}, \mathcal{R}]\!]^{M,N}$, and a variable assignment to a ground substitution. The translation is parametrized on a couple of functions, M and N, with disjoint domains and ranges, such that MN (see Notation 5.19) injectively maps the whole $\mathcal{D}$ into $\mathcal{N}$. In a nutshell, elements in $M\!\downarrow$ are mapped into names that are free in $[\![\mathcal{D}, \mathcal{R}]\!]^{M,N}$, while $N\!\downarrow$ is mapped over bound names.

Notation 5.18 (Bound Variables) *We use $bv(\phi)$ to denote the set of all the variables bound in ϕ. We will always assume that all bound variables in a formula are distinct.*

Notation 5.19 *When $M, N : \mathbf{M} \rightharpoonup \mathbf{N}$, we use MN to denote function extension, as follows:*

$$MN(x) \ \triangleq \ \textit{if } x \in N\!\downarrow \textit{ then } N(x) \textit{ else } M(x)$$

Hence, $M[c \mapsto n]$ yelds n on c and coincides with M elsewhere.

$M \setminus c$ is undefined on c and coincides with M elsewhere.

When ρ and ρ' are two substitutions, we define $\rho;\rho'$ as the only substitution such that:

$$\begin{array}{rcl} A(\rho;\rho') & = & (A\rho)\rho' \\ (\rho;\rho')(x) & = & \rho'(\rho(x)) \end{array}$$

(e.g., $[x \mapsto y];[y \mapsto c] = [x \mapsto c][y \mapsto c]$.) Hence, $\rho;\rho' = \rho\rho'$ for any pair of ground substitutions.

Definition 5.20 (Formula translation) *We define here a translation of FOL formulas, interpretations, and variable assignments, into SL formulas, interpretations, and variable assignments. Moreover, each FOL formula ϕ is also mapped to a ground substitution, defined on all and only the bound variables in ϕ, which we assume to be mutually distinct. The translation is parametric with respect to a subset $\mathbf{P}$ of $\mathcal{N}$, and to a couple of functions M, N such that $MN : \mathcal{D} \xrightarrow{in} \mathcal{N}$. $\mathbf{P}$ is used to express freshness as "not belonging to $\mathbf{P}$". In the first clause of the "formulas into substitutions" we do not specify how m' is chosen, but we will assume that the choice is deterministic, i.e. that $(\phi)^{\mathbf{P}}$ is* *uniquely determined.*

$$
\begin{array}{lll}
\textit{formulas into formulas} & & \\
\quad [\![\exists x.\,\phi]\!]^{\mathbf{Y}} & =_{def} & x\text{®}([\![\phi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x) \vee \bigvee_{y\in\mathbf{Y}} [\![\phi\{x\leftarrow y\}]\!]^{\mathbf{Y}} \\
\quad [\![\phi\wedge\psi]\!]^{\mathbf{Y}} & =_{def} & [\![\phi]\!]^{\mathbf{Y}} \wedge [\![\psi]\!]^{\mathbf{Y}} \\
\quad [\![\neg\phi]\!]^{\mathbf{Y}} & =_{def} & \neg[\![\phi]\!]^{\mathbf{Y}} \\
\quad [\![R(x,x')]\!]^{\mathbf{Y}} & =_{def} & .x.x' \\
\textit{formulas into substitutions} & & \\
\quad (\exists x.\,\phi)^{\mathbf{P}} & =_{def} & (\phi)^{\mathbf{P}}[x\mapsto m'] \quad \textit{choose } m' \in \mathcal{N} \setminus (\mathbf{P} \cup (\phi)^{\mathbf{P}}\!\uparrow) \\
\quad (\phi\wedge\psi)^{\mathbf{P}} & =_{def} & (\phi)^{\mathbf{P}}(\psi)^{\mathbf{P}\cup(\phi)^{\mathbf{P}}\uparrow} \\
\quad (\neg\phi)^{\mathbf{P}} & =_{def} & (\phi)^{\mathbf{P}} \\
\quad (R(x,x'))^{\mathbf{P}} & =_{def} & \emptyset \\
\textit{interpretations} & & \\
\quad [\![\mathcal{D},\mathcal{R}]\!]^{M,N} & =_{def} & (\vec{\nu}_{c\in N\downarrow} N(c))\,([\![\mathcal{D}]\!]^{MN} \mid [\![\mathcal{R}]\!]^{MN}) \\
\textit{domains} & & \\
\quad [\![\{c\}\cup\mathcal{D}]\!]^{M} & =_{def} & M(c)[\mathbf{0}] \mid [\![\mathcal{D}]\!]^{M} \\
\quad [\![\emptyset]\!]^{M} & =_{def} & \mathbf{0} \\
\textit{relations} & & \\
\quad [\![\{(c,c')\}\cup\mathcal{R}]\!]^{M} & =_{def} & M(c)[M(c')[\mathbf{0}]] \mid [\![\mathcal{R}]\!]^{M} \\
\quad [\![\emptyset]\!]^{M} & =_{def} & \mathbf{0} \\
\textit{assignments} & & \\
\quad [\![\sigma[x\mapsto c]]\!]^{M} & =_{def} & [\![\sigma]\!]^{M}[x\mapsto M(c)] \\
\quad [\![\mathbf{0}]\!]^{M} & =_{def} & \mathbf{0}
\end{array}
$$

Lemma 5.21

$$
\begin{array}{lcll}
 & & fn([\![\mathcal{D},\mathcal{R}]\!]^{M,N}) \subseteq M\!\uparrow & (0) \\
c\in M\!\downarrow \;\wedge\; M : M\!\downarrow \xrightarrow{in} \mathcal{N} & \Rightarrow & [\![\mathcal{D}]\!]^{M[c\mapsto m]} = [\![\mathcal{D}]\!]^{M}\{M(c)\leftarrow m\} & (1) \\
c\in M\!\downarrow \;\wedge\; M : M\!\downarrow \xrightarrow{in} \mathcal{N} & \Rightarrow & [\![\mathcal{R}]\!]^{M[c\mapsto m]} = [\![\mathcal{R}]\!]^{M}\{M(c)\leftarrow m\} & (2) \\
 & & [\![\sigma[x\mapsto c]]\!]^{M} = [\![\sigma]\!]^{M}[x\mapsto M(c)] & (3) \\
c\notin\sigma\!\uparrow & \Rightarrow & [\![\sigma[x\mapsto c]]\!]^{M[c\mapsto m]} = [\![\sigma]\!]^{M}[x\mapsto m] & (4) \\
\{x,y\}\#\, bv(\phi),\; x\notin\mathbf{Y} & \Rightarrow & [\![\phi\{x\leftarrow y\}]\!]^{\mathbf{Y}} = [\![\phi]\!]^{\mathbf{Y}}\{x\leftarrow y\} & (5) \\
x\notin\sigma\!\downarrow,\; y\in\sigma\!\downarrow & \Rightarrow & [x\mapsto y];[\![\sigma]\!]^{M} = [\![\sigma]\!]^{M}[x\mapsto M(\sigma(y))] & (6) \\
x\notin\sigma\!\downarrow & \Rightarrow & [x\mapsto m][\![\sigma]\!]^{M} = [\![\sigma]\!]^{M}[x\mapsto m] & (7)
\end{array}
$$

Proof

(4)

$$
\begin{array}{lcl}
[\![\sigma[x\mapsto c]]\!]^{M[c\mapsto m]} & = & [\![\sigma]\!]^{M[c\mapsto m]}[x\mapsto (M[c\mapsto m])(c)] \\
= \;[\![\sigma]\!]^{M[c\mapsto m]}[x\mapsto m] & = \text{ (by } c\notin\sigma\!\uparrow) & [\![\sigma]\!]^{M}[x\mapsto m]
\end{array}
$$

(5) By induction and by cases. Case $\phi = \exists z.\, \psi$:

$$
\begin{array}{ll}
[\![(\exists z.\, \psi)\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{by } x, y \neq z \\
= [\![\exists z.\, \psi\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{by def.} \\
= z®([\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}\{z\leftarrow w\}]\!]^{\mathbf{Y}} & \text{by } x, y \neq z, \\
 & w \neq x \\
= z®([\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{ind.} \\
 & (x \notin \mathbf{Y}\cup\{z\}) \\
= z®([\![\psi]\!]^{\mathbf{Y}\cup\{z\}}\{x\leftarrow y\} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}]\!]^{\mathbf{Y}}\{x\leftarrow y\} & \text{by } x \neq z \\
= (z®([\![\psi]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}]\!]^{\mathbf{Y}})\{x\leftarrow y\} & \text{by def.} \\
= [\![\exists z.\, \psi]\!]^{\mathbf{Y}}\{x\leftarrow y\} &
\end{array}
$$

□

Theorem 5.22 (Faithfulness of translation) *For any FOL formula ϕ, for any interpretation $(\mathcal{D}, \mathcal{R})$, for any set of variables $\mathbf{Y}$, for any variable assignment σ, for any pair of partial injective functions $M, N : \mathcal{D} \xrightarrow{in} \mathcal{N}$, for any finite set of names $\mathbf{P}$, such that:*

(a)	$\sigma{\downarrow} \supseteq fv(\phi) \cup \mathbf{Y}$	*σ closes the free variables of ϕ and those in $\mathbf{Y}$*
(b)	$\sigma{\uparrow} \subseteq M{\downarrow}$	*σ sends everything into the M-elements*
(c)	$\sigma(\mathbf{Y}) = M{\downarrow}$	*every M-element is reached by σ*
(d)	$M{\downarrow} \cup N{\downarrow} = \mathcal{D}$	*we know how to translate any $c \in \mathcal{D}$*
(e)	$M{\downarrow} \,\#\, N{\downarrow}$	*M-elements and N-elements are distinct*
(f)	$M{\uparrow} \,\#\, N{\uparrow}$	*M-names and N-names are distinct*
(g)	$\mathbf{P} \supseteq (MN){\uparrow}$	*names not in $\mathbf{P}$ are "fresh"*
(h)	$bv(\phi) \# \sigma{\downarrow}$	*$bv(\phi) \# fv(\phi)$ and $bv(\phi) \# \mathbf{Y}$*
(i)	*all the variables bound in ϕ are mutually distinct*	

then we have:

$$(\mathcal{D}, \mathcal{R}), \sigma \models \phi \;\Leftrightarrow\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\phi)^{\mathbf{P}} \models [\![\phi]\!]^{\mathbf{Y}}$$

Note that $[\![\sigma]\!]^{M}{\downarrow} \,\#\, (\phi)^{\mathbf{P}}{\downarrow}$, since $(\phi)^{\mathbf{P}}{\downarrow} = bv(\phi)$ and $bv(\phi) \# \sigma{\downarrow}$, hence $[\![\sigma]\!]^{M}(\phi)^{\mathbf{P}} = (\phi)^{\mathbf{P}}[\![\sigma]\!]^{M}$.

Proof

By induction on ϕ and by cases.

Case $\exists x.\, \psi$

$[\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\exists x.\, \psi)^{\mathbf{P}} \models [\![\exists x.\, \psi]\!]^{\mathbf{Y}}$

choose any $m' \in \mathcal{N} \setminus \mathbf{P}$; let $\mathbf{P}' = \mathbf{P} \cup m'$

$$
\begin{array}{l}
\Leftrightarrow [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'}[x\mapsto m'] \models x®([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x) \\
\qquad\qquad\qquad\qquad\qquad\qquad \vee\ \bigvee_{y\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}} \\
\Leftrightarrow [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'}[x\mapsto m'] \models x®([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x) \\
\qquad \vee\ [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'}[x\mapsto m'] \models \bigvee_{y\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}
\end{array}
$$

By (h) and (i), x is not in the domain of σ or of $(\psi)^{\mathbf{P}'}$,
hence in the first line we can move $[x \mapsto m']$ to the term;
second line: remove $[x \mapsto m']$ since x does not appear in the formula

$$\begin{aligned}
\Leftrightarrow\ & [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \models (x \circledR ([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x))\{x \leftarrow m'\} \\
& \quad \vee\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \models \textstyle\bigvee_{y\in\mathbf{Y}} [\![\psi\{x \leftarrow y\}]\!]^{\mathbf{Y}}
\end{aligned}$$

We apply $\{x \leftarrow m'\}$ and expand $[\![\mathcal{D},\mathcal{R}]\!]^{M,N}$

$$\begin{aligned}
\Leftrightarrow\ & (\vec{\nu}_{c\in N\downarrow} N(c))\,([\![\mathcal{D}]\!]^{MN} \mid [\![\mathcal{R}]\!]^{MN}), \\
& \qquad [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \qquad\qquad \models m' \circledR ([\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x \leftarrow m'\} \wedge .m') \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi\{x \leftarrow y\}]\!]^{\mathbf{Y}}
\end{aligned}$$

Since $m' \notin fn([\![\mathcal{D},\mathcal{R}]\!]^{M,N})$ (by $\mathbf{P} \supseteq M\!\uparrow \supseteq fn([\![\mathcal{D},\mathcal{R}]\!]^{M,N})$), we apply
Corollary 3.15 (first line)

$$\begin{aligned}
\Leftrightarrow\ & \exists c' \in N\!\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,(([\![\mathcal{D}]\!]^{MN} \mid [\![\mathcal{R}]\!]^{MN})\{N(c') \leftarrow m'\}), \\
& \qquad [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x \leftarrow m'\} \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi\{x \leftarrow y\}]\!]^{\mathbf{Y}}
\end{aligned}$$

By Lemma 5.21 (1,2), since $MN[c' \mapsto m']$ is injective by $m' \notin \mathbf{P} \supseteq MN\!\uparrow$
and $N(c') = MN(c')$

$$\begin{aligned}
\Leftrightarrow\ & \exists c' \in N\!\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']}, \\
& \qquad [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x \leftarrow m'\} \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi\{x \leftarrow y\}]\!]^{\mathbf{Y}}
\end{aligned}$$

By Lemma 5.21 (5), since $x \#\, bv(\psi)$ (i), $y \#\, bv(\psi)$ (h), and $x \notin \mathbf{Y}$ (h)

$$\begin{aligned}
\Leftrightarrow\ & \exists c' \in N\!\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']}, \\
& \qquad [\![\sigma]\!]^{M}(\psi)^{\mathbf{P}'} \qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x \leftarrow m'\} \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, ([x \mapsto y]; [\![\sigma]\!]^{M})(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}}
\end{aligned}$$

$y \in \sigma\!\downarrow$ and $x \notin \sigma\!\downarrow$, hence, by Lemma 5.21 (7,6)

$$\begin{aligned}
\Leftrightarrow\ & \exists c' \in N\!\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']}, \\
& \qquad [\![\sigma]\!]^{M}[x \mapsto m'](\psi)^{\mathbf{P}'} \qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}[x \mapsto M(\sigma(y))](\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}}
\end{aligned}$$

By Lemma 5.21 (4), since $c' \notin \sigma\!\uparrow$ by $N\!\downarrow \# M\!\downarrow$ (e) and $M\!\downarrow \supseteq \sigma\!\uparrow$ (b)
and by Lemma 5.21 (3) (second line)

$$\begin{aligned}
\Leftrightarrow\ & \exists c' \in N\!\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']}, \\
& \qquad [\![\sigma[x \mapsto c']]\!]^{M[c'\mapsto m']}(\psi)^{\mathbf{P}'} \quad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \\
& \quad \vee\ \exists y \in \mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma[x \mapsto \sigma(y)]]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}}
\end{aligned}$$

We now prepare the first line for the inductive step. We define $M' = M[c' \mapsto m']$,

$N' = N \setminus c'$, $\mathbf{Y}' = \mathbf{Y} \cup \{x\}$, $\sigma' = \sigma[x \mapsto c']$. We check the induction conditions.

$$
\begin{array}{ll}
(a) & \sigma'{\downarrow} = \sigma{\downarrow} \cup \{x\} \;\supseteq\; fv(\exists x.\,\psi) \cup \mathbf{Y} \cup \{x\} \;\supseteq\; fv(\psi) \cup \mathbf{Y}' \\
(b) & \sigma'{\uparrow} = \sigma{\uparrow} \cup \{c'\} \;\subseteq\; M{\downarrow} \cup \{c'\} \;=\; M'{\downarrow} \\
(c) & \sigma'(\mathbf{Y}') \;=\; \sigma(\mathbf{Y}) \cup \{c'\} \;=\; M{\downarrow} \cup \{c'\} \;=\; M'{\downarrow} \\
(d) & M'{\downarrow} \cup N'{\downarrow} = \; M{\downarrow} \cup \{c'\} \cup (N{\downarrow} \setminus \{c'\}) \;=\; M{\downarrow} \cup N{\downarrow} \;=\; \mathcal{D} \\
(e) & M'{\downarrow} \,\#\, N'{\downarrow} \Leftrightarrow\; (M{\downarrow} \cup \{c'\}) \# (N{\downarrow} \setminus \{c'\}) \;\Leftarrow\; M{\downarrow} \,\#\, N{\downarrow} \\
(f) & M'{\uparrow} \,\#\, N'{\uparrow} \Leftarrow\; (M{\uparrow} \cup \{m'\}) \# N{\uparrow} \Leftrightarrow\; (M{\uparrow} \,\#\, N{\uparrow} \;\wedge m' \notin N{\uparrow}) \\
(g) & \mathbf{P}' \;=\; \mathbf{P} \cup \{m'\} \;\supseteq\; (MN){\uparrow} \cup \{m'\} \;\supseteq\; (M'N'){\uparrow} \\
(h) & bv(\psi) \# \sigma'{\downarrow} \Leftrightarrow\; (bv(\exists x.\,\psi) \setminus \{x\}) \# (\sigma{\downarrow} \cup \{x\}) \;\Leftarrow\; bv(\exists x.\,\psi) \# \sigma{\downarrow}
\end{array}
$$

Second line: $\Rightarrow$: let $c = \sigma(y)$. $\Leftarrow$ follows by $\sigma(\mathbf{Y}) = M{\downarrow}$

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N{\downarrow} \,.\, (\vec{\nu}_{c \in N'{\downarrow}} N'(c))\, [\![\mathcal{D}]\!]^{M'N'} \mid [\![\mathcal{R}]\!]^{M'N'}, [\![\sigma']\!]^{M'}(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}'} \\
\qquad \vee\; \exists c \in M{\downarrow} \,.\, [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma[x \mapsto c]]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}
$$

By def. of $[\![\mathcal{D}, \mathcal{R}]\!]^{M',N'}$

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N{\downarrow} \,.\, [\![\mathcal{D}, \mathcal{R}]\!]^{M',N'} [\![\sigma']\!]^{M'}(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}'} \\
\qquad \vee\; \exists c \in M{\downarrow} \,.\, [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma[x \mapsto c]]\!]^{M}(\psi)^{\mathbf{P}'} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}
$$

We now apply induction to both disjuncts. For the second line, we observe that (g) $\mathbf{P}' \supseteq MN$, (c) $(\sigma[x \mapsto c])(\mathbf{Y}) = \sigma(\mathbf{Y}) = \sigma{\uparrow}$, (b) $(\sigma[x \mapsto c]){\uparrow} = \sigma{\uparrow} \cup \{c\} \subseteq M{\downarrow}$.

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N{\downarrow} \,.\, (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c'] \models \psi \;\vee\; \exists c \in M{\downarrow} \,.\, (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c] \models \psi \\
\Leftrightarrow \exists c \in \mathcal{D}.\; (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c] \models \psi \\
\Leftrightarrow (\mathcal{D}, \mathcal{R}), \sigma \models \exists x.\,\psi
\end{array}
$$

Cases $\neg\psi$, $\psi \vee \psi'$
Simple induction.

Case $R(x, x')$ Observe that, for any $c', c'' \in \mathcal{D}$:

$$(c', c'') \in \mathcal{R} \Leftrightarrow \exists T.\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N} = (\vec{\nu}_{c \in N{\downarrow}} N(c))\, (MN(c')[MN(c'')[\mathbf{0}]] \mid T)$$

By $M{\uparrow} \,\#\, N{\uparrow}$, if $c', c'' \in M{\downarrow}$, the condition above is equivalent to:

$$(c', c'') \in \mathcal{R} \Leftrightarrow \exists T.\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N} \equiv M(c')[M(c'')[\mathbf{0}]] \mid (\vec{\nu}_{c \in N{\downarrow}} N(c))\, T$$

i.e., for $c', c'' \in M{\downarrow}$:

$$
\begin{array}{l}
(c', c'') \in \mathcal{R} \\
\Leftrightarrow \exists U.\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N} \equiv M(c')[M(c'')[\mathbf{0}]] \mid U \\
\Leftrightarrow [\![\mathcal{D}, \mathcal{R}]\!]^{M,N} \models .M(c').M(c'')
\end{array}
$$

Hence:

$$
\begin{array}{rl}
& [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M}(R(x,x'))^{\mathbf{P}} \models [\![R(x,x')]\!]^{\mathbf{Y}} \\
& \Leftrightarrow [\![\mathcal{D},\mathcal{R}]\!]^{M,N}, [\![\sigma]\!]^{M} \models .x.x' \\
& \Leftrightarrow [\![\mathcal{D},\mathcal{R}]\!]^{M,N} \models .M(\sigma(x)).M(\sigma(x')) \\
\text{by } \sigma\!\uparrow\subseteq M\!\downarrow & \Leftrightarrow (\sigma(x),\sigma(x'))\in\mathcal{R} \\
& \Leftrightarrow \mathcal{D},\mathcal{R},\sigma \models R(x,x')
\end{array}
$$

□

Corollary 5.23 *For any closed FOL formula ϕ where all the free and bound variables are disjoint, for any $N : \mathcal{D} \overset{in}{\to} \mathcal{N}$:*

$$\mathcal{D},\mathcal{R} \models \phi \;\Leftrightarrow\; [\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}(\phi)^{\emptyset}$$

Proof By Theorem 5.22, letting M be the empty function, $\mathbf{Y}$ be the empty set, $\mathbf{P} = N\!\uparrow$, and σ be the empty assignment, we have that

$$\mathcal{D},\mathcal{R} \models \phi \;\Leftrightarrow\; [\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}[\![\emptyset]\!]^{\emptyset}(\phi)^{N\uparrow} \;\Leftrightarrow\; [\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}(\phi)^{N\uparrow}$$

$[\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}(\phi)^{N\uparrow}$ is equivalent to $[\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}(\phi)^{\emptyset}$ by Corollary 3.12, since $(\phi)^{\emptyset}$ and $(\phi)^{N\uparrow}$ are injective by construction, $(\phi)^{\emptyset}\!\downarrow = (\phi)^{N\uparrow}\!\downarrow = bv(\phi)$, and $fn([\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N}, [\![\phi]\!]^{\emptyset})$ is empty. □

Corollary 5.24 *For any closed FOL formula ϕ where all the free and bound variables are disjoint $SAT_{FOL}(\phi) \Rightarrow SAT_{SL}([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})$*

Unfortunately, the inverse implication does not hold. Consider $(\exists x.\ \mathsf{T}) \wedge \neg(\exists y.\ \mathsf{T})$. It is clearly unsatisfiable, but it is translated (under $\mathbf{Y} = \emptyset$) as $m \circledR (\mathsf{T} \wedge .m) \wedge \neg n \circledR (\mathsf{T} \wedge .n)$, which is satisfied by the model $(\nu m')\, m'[\mathbf{0}] \,|\, n[\mathbf{0}]$, since the presence of n free prevents the model from satisfying $n \circledR (\mathsf{T} \wedge .n)$, while it satisfies $m \circledR (\mathsf{T} \wedge .m)$.

This fact does not contradict Theorem 5.22, since $(\nu m')\, m'[\mathbf{0}] \,|\, n[\mathbf{0}]$ is not the translation of any FOL model under $\mathbf{Y} = \emptyset$ (because $\mathbf{Y} = \emptyset$ implies that M is the empty function, hence $[\![\mathcal{D},\mathcal{R}]\!]^{M,N}$ must be closed, by Lemma 5.21-(0)). The fact that the model is not closed is actually the core of the problem. We solve this problem by considering a new mapping that rules some of the non-closed models out, and we use the cut operation and Corollary 5.16 to show that each of the remaining models actually corresponds to a FOL model, hence finally reducing SAT_{SL} to SAT_{FOL}.

Lemma 5.25 *Let $T = Cut_{N'}(\mathbf{P})$ for some $N', \mathbf{P}$; then:*

$$fn(T) = \emptyset \;\Rightarrow\; \exists \mathcal{D},\mathcal{R},N.\ T = [\![\mathcal{D},\mathcal{R}]\!]^{\emptyset,N}$$

Proof $T = Cut_{T'}(\mathbf{P})$ implies that T is a set of two-lines $Par\{m_i[m'_i[\mathbf{0}]] : i \in I\}$ plus a set of one-lines $Par\{n_j[\mathbf{0}] : j \in J\}$, with the property that $\forall i \in I.\ m_i \in \{n_j\}^{j\in J}, m'_i \in \{n_j\}^{j \in J}$. This set of one- and two-lines is preceded by a string of restrictions. $fn(T) = \emptyset$ implies that every name in T is actually restricted, i.e. that:

$$T = (\vec{\nu}_{j\in J} n_j)\ Par\{m_i[m'_i[\mathbf{0}]] : i \in I\} \mid Par\{n_j[\mathbf{0}] : j \in J\}$$

Now, the thesis follows by choosing $\mathcal{D} = \{n_j\}^{j\in J}$, $\mathcal{R} = \{(m_i, m'_i)\}^{i\in I}$, and letting N be the identity function over $\mathcal{D}$. □

We can now define a translation that will ensure that the any model of the translated formula is "closed enough", i.e. all its free names are disjoint from the names in the formula. This means that those free names will be erased by the *Cut*() procedure that we exploit in the proof of Lemma 5.27.

Definition 5.26

$$[\![\phi]\!]^{+} =_{def} [\![\phi]\!]^{\emptyset}(\phi)^{\emptyset} \wedge \bigwedge_{m\in nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})} \neg ⓒ m$$

Lemma 5.27 (Equivalence of satisfiability) *For any closed FOL formula* ϕ*,* $SAT_{FOL}(\phi) \Leftrightarrow SAT_{SL}([\![\phi]\!]^{+})$

Proof ($\Rightarrow$) Let $\mathcal{D}, \mathcal{R}$ be such that $(\mathcal{D}, \mathcal{R}), \emptyset \models \phi$. By Corollary 5.23, $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset,N}$ satisfies $[\![\phi]\!]^{\emptyset}(\phi)^{\emptyset}$. Since $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset,N}$ is closed, it also satisfies $\neg ⓒ m$ for any m.

($\Leftarrow$) Assume $SAT_{SL}([\![\phi]\!]^{+})$. Then, there exists T such that:

$$T \models [\![\phi]\!]^{\emptyset}(\phi)^{\emptyset} \qquad (1)$$
$$T \models \bigwedge_{m\in nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})} \neg ⓒ m \qquad (2)$$

Consider now $U = Cut_{nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})}(T)$. By Lemma 5.15:

$$U \models [\![\phi]\!]^{\emptyset}(\phi)^{\emptyset} \qquad (3)$$
$$U \models \bigwedge_{m\in nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})} \neg ⓒ m \qquad (4)$$

By Lemma 5.8

$$fn(U) \subseteq nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})$$

by (4)

$$fn(U) \# nm([\![\phi]\!]^{\emptyset}(\phi)^{\emptyset})$$

hence

$$fn(U) = \emptyset$$

By Lemma 5.25, U is the translation of a FOL interpretation $\mathcal{D}, \mathcal{R}$. By Corollary 5.23, (3) implies $\mathcal{D}, \mathcal{R} \models \phi$; hence $SAT_{FOL}(\phi)$. □

Corollary 5.28 (Undecidability of revelation) *Satisfiability (hence validity) of closed formulas built from* $n ® A$, $A \wedge A$, $\neg A$, $.n$, $.n_1.n_2$, *is not decidable.*

Theorem 4.8 and Corollary 5.28 together constitute our main result: hiding can be expressed as freshness plus revelation, freshness quantification without revelation gives a rich decidable logic (Theorem 4.8) while revelation makes a minimal logic undecidable (Corollary 5.28).

5.4 Undecidability of Hiding Quantification

The proof of undecidability of hiding quantification is very similar to that of revelation. The translation is slightly simpler since we do not need the $(\phi)^{\mathbf{P}}$ substitution any more. Moreover, since the translation of a formula contains no free name, the step from the faithfulness theorem to the undecidability corollary is shorter as well.

Definition 5.29 (Formula translation) *FOL formulas, assignments, interpretations, domains, and relations, are translated as in Definition 5.20, under the same conditions over ϕ, $\mathbf{Y}$, $\mathcal{N}$, M, and N, with the only exception of the existential quantifier, as specified below. This time we need no translation of formulas into substitutions, hence we do not need the set $\mathbf{P}$.*

formulas

$$[\![\exists x.\, \phi]\!]^{\mathbf{Y}} \quad =_{def} \quad \mathsf{H}x.\, ([\![\phi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x) \vee \textstyle\bigvee_{y\in\mathbf{Y}} [\![\phi\{x\leftarrow y\}]\!]^{\mathbf{Y}}$$

$\ldots$

Lemma 5.30 *If $\{x,y\}\#\, bv(\phi)$ and $x \notin \mathbf{Y}$, then:*

$$[\![\phi\{x\leftarrow y\}]\!]^{\mathbf{Y}} = [\![\phi]\!]^{\mathbf{Y}}\{x\leftarrow y\}$$

Proof By induction and by cases. Case $\phi = \exists z.\, \psi$:

$$\begin{array}{ll}
[\![(\exists z.\, \psi)\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{by } x,y \neq z \\
= [\![\exists z.\, \psi\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{by def.} \\
= \mathsf{H}z.\, ([\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}\{z\leftarrow w\}]\!]^{\mathbf{Y}} & \text{by } x,y \neq z \\
 & w \neq x \\
= \mathsf{H}z.\, ([\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}\{x\leftarrow y\}]\!]^{\mathbf{Y}} & \text{ind.} \\
 & (x \notin \mathbf{Y}\cup\{z\}) \\
= \mathsf{H}z.\, ([\![\psi]\!]^{\mathbf{Y}\cup\{z\}}\{x\leftarrow y\} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}]\!]^{\mathbf{Y}}\{x\leftarrow y\} & \text{by } x \neq z \\
= (\mathsf{H}z.\, ([\![\psi]\!]^{\mathbf{Y}\cup\{z\}} \wedge .z) \vee \bigvee_{w\in\mathbf{Y}} [\![\psi\{z\leftarrow w\}]\!]^{\mathbf{Y}})\{x\leftarrow y\} & \text{by def.} \\
= [\![\exists z.\, \psi]\!]^{\mathbf{Y}}\{x\leftarrow y\} &
\end{array}$$

$\square$

Theorem 5.31 (Faithfulness of translation) *For any FOL formula ϕ, for any interpretation $(\mathcal{D},\mathcal{R})$, for any set of variables $\mathbf{Y}$, for any variable assignment σ, for any pair of partial injective functions $M,N : \mathcal{D} \xrightarrow{in} \mathcal{N}$, such that:*

(a)	$\sigma\!\downarrow \supseteq fv(\phi) \cup \mathbf{Y}$	*σ closes the free variables of ϕ and those in $\mathbf{Y}$*
(b)	$\sigma\!\uparrow \subseteq M\!\downarrow$	*σ sends everything into the M-elements*
(c)	$\sigma(\mathbf{Y}) = M\!\downarrow$	*every M-element is reached by σ*
(d)	$M\!\downarrow \cup N\!\downarrow = \mathcal{D}$	*we know how to translate any $c\in\mathcal{D}$*
(e)	$M\!\downarrow \# N\!\downarrow$	*M-elements and N-elements are distinct*
(f)	$M\!\uparrow \# N\!\uparrow$	*M-names and N-names are distinct*
(g)	$bv(\phi) \# \sigma\!\downarrow$	*$bv(\phi)\# fv(\phi)$ and $bv(\phi)\# \mathbf{Y}$*
(h)	*all the variables bound in ϕ are mutually distinct*	

then we have:

$$(\mathcal{D},\mathcal{R}),\sigma \models \phi \;\Leftrightarrow\; [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models [\![\phi]\!]^{\mathbf{Y}}$$

Proof

By induction on ϕ and by cases, along the lines of Theorem 5.22.

Case $\exists x.\,\psi$

$$\begin{array}{l}
[\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models [\![\exists x.\,\psi]\!]^{\mathbf{Y}}\\
\Leftrightarrow [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models \mathsf{H}x.\,([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x)\\
\qquad\qquad\qquad\qquad\qquad \vee\ \bigvee_{y\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}\\
\Leftrightarrow [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models \mathsf{H}x.\,([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x)\\
\quad \vee\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models \bigvee_{y\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}\\
\Leftrightarrow (\vec{\nu}_{c\in N\downarrow} N(c))\,[\![\mathcal{D}]\!]^{MN} \mid [\![\mathcal{R}]\!]^{MN},[\![\sigma]\!]^{M} \models \mathsf{H}x.\,([\![\psi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x)\\
\quad \vee\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models \bigvee_{y\in\mathbf{Y}} [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}
\end{array}$$

Choose $m'\notin MN\uparrow$; this makes it fresh enough to apply Corollary 3.17

$$\begin{array}{l}
\Leftrightarrow \exists c'\in N\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,(([\![\mathcal{D}]\!]^{MN} \mid [\![\mathcal{R}]\!]^{MN})\{N(c')\leftarrow m'\}),\\
\qquad\qquad\qquad [\![\sigma]\!]^{M} \qquad\qquad\qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x\leftarrow m'\}\\
\quad \vee\ \exists y\in\mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}
\end{array}$$

By Lemma 5.21 (1,2), since $MN(c') = N(c')$ and $MN : MN \xrightarrow{in} \mathcal{N}$

$$\begin{array}{l}
\Leftrightarrow \exists c'\in N\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']},\\
\qquad\qquad\qquad [\![\sigma]\!]^{M} \qquad\qquad\qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x\leftarrow m'\}\\
\quad \vee\ \exists y\in\mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M} \models [\![\psi\{x\leftarrow y\}]\!]^{\mathbf{Y}}
\end{array}$$

By Lemma 5.30 ($x \notin bv(\psi)$ by (h); $y \notin bv(\psi)$ by $\mathbf{Y}\#\,bv(\exists x.\,\psi)$; $x \notin \mathbf{Y}$ by $\mathbf{Y}\#\,bv(\exists x.\,\psi)$)

$$\begin{array}{l}
\Leftrightarrow \exists c'\in N\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']},\\
\qquad\qquad\qquad [\![\sigma]\!]^{M} \qquad\qquad\qquad\qquad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\{x\leftarrow m'\}\\
\quad \vee\ \exists y\in\mathbf{Y}.\ \ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[x\mapsto y];[\![\sigma]\!]^{M} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}$$

$y\in\sigma\downarrow$ and $x\notin\sigma\downarrow$, hence, by Lemma 5.21 (7,6)

$$\begin{array}{l}
\Leftrightarrow \exists c'\in N\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']},\\
\qquad\qquad\qquad [\![\sigma]\!]^{M}[x\mapsto m'] \qquad\qquad\quad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\\
\quad \vee\ \exists y\in\mathbf{Y}.\ \ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma]\!]^{M}[x\mapsto M(\sigma(y))] \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}$$

By Lemma 5.21 (4), since $c' \notin \sigma\uparrow$ by $N\downarrow \# M\downarrow$ (e) and $M\downarrow \supseteq \sigma\uparrow$ (b) and by Lemma 5.21 (3) (second line)

$$\begin{array}{l}
\Leftrightarrow \exists c'\in N\downarrow.\ (\vec{\nu}_{c\in N\downarrow\setminus\{c'\}} N(c))\,[\![\mathcal{D}]\!]^{MN[c'\mapsto m']} \mid [\![\mathcal{R}]\!]^{MN[c'\mapsto m']},\\
\qquad\qquad\qquad [\![\sigma[x\mapsto c']]\!]^{M[c'\mapsto m']} \qquad\quad \models [\![\psi]\!]^{\mathbf{Y}\cup\{x\}}\\
\quad \vee\ \exists y\in\mathbf{Y}.\ [\![\mathcal{D},\mathcal{R}]\!]^{M,N},[\![\sigma[x\mapsto\sigma(y)]]\!]^{M} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}$$

We now prepare the first line for the inductive step. We define $M' = M[c'\mapsto m']$,

$N' = N \setminus c'$, $\mathbf{Y}' = \mathbf{Y} \cup \{x\}$, $\sigma' = \sigma[x \mapsto c']$. We check the induction conditions.

$$
\begin{array}{ll}
(a) & \sigma'\!\downarrow \; = \; \sigma\!\downarrow \cup \{x\} \;\supseteq\; fv(\exists x.\,\psi) \cup \mathbf{Y} \cup \{x\} \;\supseteq\; fv(\psi) \cup \mathbf{Y}' \\
(b) & \sigma'\!\uparrow \; = \; \sigma\!\uparrow \cup \{c'\} \;\subseteq\; M\!\downarrow \cup \{c'\} \; = \; M'\!\downarrow \\
(c) & \sigma'(\mathbf{Y}') \; = \; \sigma(\mathbf{Y}) \cup \{c'\} \; = \; M\!\downarrow \cup \{c'\} \; = \; M'\!\downarrow \\
(d) & M'\!\downarrow \cup N'\!\downarrow \; = \; M\!\downarrow \cup \{c'\} \cup (N\!\downarrow \setminus \{c'\}) \; = \; M\!\downarrow \cup N\!\downarrow \; = \; \mathcal{D} \\
(e) & M'\!\downarrow \# N'\!\downarrow \;\Leftrightarrow\; (M\!\downarrow \cup \{c'\}) \# (N\!\downarrow \setminus \{c'\}) \;\Leftarrow\; M\!\downarrow \# N\!\downarrow \\
(f) & M'\!\uparrow \# N'\!\uparrow \;\Leftarrow\; (M\!\uparrow \cup \{m'\}) \# N\!\uparrow \;\Leftrightarrow\; (M\!\uparrow \# N\!\uparrow \;\wedge\; m' \notin N\!\uparrow) \\
(g) & bv(\psi) \# \sigma'\!\downarrow \;\Leftrightarrow\; (bv(\exists x.\,\psi) \setminus \{x\}) \# (\sigma\!\downarrow \cup \{x\}) \;\Leftarrow\; bv(\exists x.\,\psi) \# \sigma\!\downarrow
\end{array}
$$

Second line: $\Rightarrow$: let $c = \sigma(y)$. $\Leftarrow$ follows by $\sigma(\mathbf{Y}) = M\!\downarrow$

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N\!\downarrow .\; (\vec{\nu}_{c \in N'\downarrow} N'(c))\, [\![\mathcal{D}]\!]^{M'N'} \mid [\![\mathcal{R}]\!]^{M'N'}, [\![\sigma[x \mapsto c']]\!]^{M'} \models [\![\psi]\!]^{\mathbf{Y}'} \\
\qquad \vee\; \exists c \in M\!\downarrow .\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma[x \mapsto c]]\!]^{M} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}
$$

By def. of $[\![\mathcal{D}, \mathcal{R}]\!]^{M',N'}$

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N\!\downarrow .\; [\![\mathcal{D}, \mathcal{R}]\!]^{M',N'} [\![\sigma[x \mapsto c']]\!]^{M'} \models [\![\psi]\!]^{\mathbf{Y}'} \\
\qquad \vee\; \exists c' \in M\!\downarrow .\; [\![\mathcal{D}, \mathcal{R}]\!]^{M,N}, [\![\sigma[x \mapsto c']]\!]^{M} \models [\![\psi]\!]^{\mathbf{Y}}
\end{array}
$$

We now apply induction to both disjuncts.

$$
\begin{array}{l}
\Leftrightarrow \exists c' \in N\!\downarrow .\; (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c'] \models \psi \;\vee\; \exists c' \in M\!\downarrow .\; (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c'] \models \psi \\
\Leftrightarrow \exists c \in \mathcal{D}.\; (\mathcal{D}, \mathcal{R}), \sigma[x \mapsto c] \models \psi \\
\Leftrightarrow (\mathcal{D}, \mathcal{R}), \sigma \models \exists x.\, \psi
\end{array}
$$

Cases $\neg\psi$, $\psi \vee \psi'$, $R(x, y)$: As in the proof of Theorem 5.22.

□

Lemma 5.32 (Equivalence of satisfiability) *For any closed FOL formula* ϕ, $SAT_{FOL}(\phi) \Leftrightarrow SAT_{SL}([\![\phi]\!]^{\emptyset})$

Proof ($\Rightarrow$) Assume all variables in ϕ are distinct. Let $\mathcal{D}, \mathcal{R}$ be such that $(\mathcal{D}, \mathcal{R}), \emptyset \models \phi$. Consider any $N : \mathcal{D} \xrightarrow{in} \mathcal{N}$. By Theorem 5.31, $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset,N}$ satisfies $[\![\phi]\!]^{\emptyset}$ (as in the proof of Corollary 5.23).

($\Leftarrow$) Assume $SAT_{SL}([\![\phi]\!]^{\emptyset})$. Then, there exists T such that:

$$T \models [\![\phi]\!]^{\emptyset}$$

Consider now $U = Cut_{nm([\![\phi]\!]^{\emptyset})}(T) = Cut_{\emptyset}(T)$. By Lemma 5.15, $U \models [\![\phi]\!]^{\emptyset}$. By Lemma 5.8, $fn(U) = \emptyset$. By Lemma 5.25, $U = [\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset,N}$ for some $\mathcal{D}$, $\mathcal{R}$, N. By Theorem 5.31, $U = [\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset,N} \models [\![\phi]\!]^{\emptyset}$ implies $\mathcal{D}, \mathcal{R} \models \phi$; hence $SAT_{FOL}(\phi)$.
□

Corollary 5.33 (Undecidability of Hiding) *Satisfiability (hence validity) of closed formulas built from* $\mathsf{H}x.\,A$, $A \wedge A$, $\neg A$, $.x_1$, *and* $.x_1.x_2$, *is not decidable.*

6 Related Work

Independently of this study, an extrusion algorithm for the freshness quantifier (Section 4.2) has been developed in [16]. That version is however incorrect, since its rule ($\triangleright R$) is the one that we disprove in counterexample ($\text{И-}\triangleright r \not\vdash$).

That error does not influence the main result of that paper, a surprising adjunct elimination theorem.

Theorem 6.1 (Adjunct elimination (Lozes)) *For any formula A generated by the grammar below, there exists A' such that $A \dashv\vdash A'$ and A' contains no adjunct.*

$$A ::= \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid \text{И}x.\, A \mid A \wedge A \mid \neg A \mid \eta \circledR A \mid A \triangleright A \mid A @ \eta \mid A \oslash \eta$$

The result is surprising in view of the fact that the parallel-adjunct seems to be extremely expressive, being able to quantify over infinite sets of trees, and of internalizing validity into model-checking.

Lozes leaves the open problem of the existence of an effective adjunct-elimination procedure. As a corollary of our undecidability results, we can close that problem.

Corollary 6.2 *No effective adjunct-elimination can exist for the logic of Theorem 6.1.*

A calculus to manipulate trees with hidden names has been presented in [8], whose type system includes SL. As a result, type inclusion in that calculus and validity in SL are mutually reducible. Decidability of subtype-checking was left as an open problem in [8]. Our results imply that it is undecidable.

7 Conclusions

This paper answers these previously open questions:

1. Decidability of Spatial Logics (without revelation and quantifiers) equipped with fresh name quantifier.
2. Decidability of Spatial Logics equipped with revelation or hiding quantifier.
3. Existence of extrusion algorithm for hiding, revelation, existential quantification.
4. Existence of an effective adjunct-elimination procedure in the Spatial Logic with freshness and revelation.

The decidability result (1) is based on the extrusion of freshness into a prenex form. The proof of decidability by extrusion is very attractive because it does not need combinatorial explorations of the model, but is based on the "algebraic" properties of the logic, and is robust with respect to variations on the logic itself.

The undecidable logic (2) is obtained adding revelation to a minimal logic of propositional connectives and simple path formulas, hence we show that undecidability comes from revelation and not from the spatial nature of SL. Undecidability of any richer logic (e.g. the static fragment of the Ambient Logic with revelation and without composition adjunct) follows immediately.

8 Acknowledgments

We would like to thank Luís Caires, Cristiano Calcagno, Luca Cardelli, Dario Colazzo, Anuj Dawar, Philippa Gardner, Andy Gordon, for suggestions and discussions which influenced this work in many ways.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 10 January 1999.

[2] L. Caires and L. Cardelli. A spatial logic for concurrency (Part I). In *Proc. of Theoretical Aspects of Computer Software; 4th International Symposium, TACS 2001*, volume 2215 of *LNCS*, pages 1–37. Springer-Verlag, 2001.

[3] L. Caires and L. Cardelli. A spatial logic for concurrency (Part II). In *Proc. of CONCUR'02*, volume 2421 of *LNCS*, page 209. Springer-Verlag, 2002.

[4] L. Caires and L.Monteiro. Verifiable and executable logic specifications of concurrent objects in L_π. In *Proc. of the 7th European Symposium on Programming (ESOP'98)*, volume 1381 of *LNCS*, pages 42–56. Springer-Verlag, 1998.

[5] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *Proc. of ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI'03)*, 2003.

[6] L. Cardelli. Describing semistructured data. *SIGMOD Record, Database Principles Column*, 30(4), 2001.

[7] L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *Proc. of ICALP*, volume 2380 of *LNCS*, page 597. Springer-Verlag, 2002.

[8] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating trees with hidden labels. In *Proc. of FOSSACS '03*, volume 2620 of *LNCS*, pages 216–232. Springer-Verlag, 2003.

[9] L. Cardelli and G. Ghelli. A query language based on the ambient logic. In *Proc. of European Symposium on Programming (ESOP), Genova, Italy*, volume 2028 of *LNCS*, pages 1–22. Springer-Verlag, 2001.

[10] L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. of POPL*. ACM Press, 2000.

[11] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In *International Conference on Typed Lambda Calculi and Applications (TCLA 2001, Krakow, Poland)*, volume 2044 of *LNCS*, pages 46–60. Springer, 2001.

[12] L. Cardelli and A. D. Gordon. Ambient logic. Submitted for publication, available from the authors, 2002.

[13] W. Charatonik and J.M. Talbot. The decidability of model checking mobile ambients. In *CSL: 15th Workshop on Computer Science Logic*, volume 2142 of *LNCS*, page 339, 2001.

[14] G. Conforti and G. Ghelli. Spatial logics to reason about semistructured data. In *Proc. of SEBD 2003: Eleventh Italian Symposium on Advanced Database Systems*. Rubettino Editore, 2003.

[15] M. Gabbay and A.M. Pitts. A new approach to abstract syntax involving binders. In *Proc. of LICS'99*, pages 214–224. IEEE Computer Society Press, 1999.

[16] É. Lozes. Adjuncts elimination in the static ambient logic. As published in http://www.ens-lyon.fr/~elozes/adjunct.pdf, July 2003.

[17] Peter O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *In Proc. of CSL*, volume 2142 of *LNCS*, pages 1–19. Springer-Verlag, 2001.

[18] A. M. Pitts. Nominal logic: A first order theory of names and binding. In *Proc. of TACS 2001*, volume 2215 of *LNCS*, pages 219–242. Springer-Verlag, 2001.

[19] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc. LICS'02*, pages 55–74. IEEE Computer Society, 2002.