Decidability of Freshness, Undecidability of Revelation
(Extended Abstract)

Giovanni Conforti and Giorgio Ghelli

Università di Pisa, Italy

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* (SL) has been recently used to refer to logics equipped with the composition-separation operator $A \mid B$. Spatial logics are emerging as an interesting tool to describe properties of several structures. Models for spatial logics include computational structures such as heaps [21, 19], trees [7], trees with hidden names [9], graphs [8], concurrent objects [5], as well as process calculi such as the π-calculus [3, 4] and the Ambient Calculus [11, 13].

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 information that is protected by the computational model, such as hidden encryption keys [1], the actual variable names in λ-calculus, object identifiers in object calculi, and locations in a heap. Here "protected" means that no public name can ever clash with one that is protected, and that any observable behavior may depend on the equality between two names, but not on the actual value of a protected name.

Reasoning about protected names is difficult because they are "anonymous". Cardelli and Gordon suggest an elegant solution to this problem [12]. They adopt Gabbay and Pitts fresh name quantification, originally used for binder manipulation and Nominal Logics [20, 16], and combine it with a new operator, *revelation*, which allows a public name to be used to denote a protected one. The combination of freshness quantification and revelation gives rise to a new quantifier, *hidden name quantification*, which can be used to describe properties of restricted names in a natural way.

In [6] decidability of validity and model-checking of a spatial logic describing trees *without* restricted names is studied. This logic is the quantifier-free static fragment of the Ambient Logic. Extensions of this logic can be used to describe [7], query [10], and reason about [15] tree-shaped semistructured data.

In this paper we study decidability of validity, satisfiability, and model-checking for spatial logics describing trees (or static ambients) with restricted names (throughout the paper, "decidability of a logic" is used for "decidability of validity and satisfiability for closed formulas of that logic").

In particular we study how the introduction of freshness, revelation, and hiding influences decidability. While we started this work with the aim of proving decidability of hiding, we found out quite a different situation:

- freshness without revelation gives a rich decidable logic (Corollary 4.7)
- even a minimal logic (conjunction, negation, and binary relations) becomes undecidable if it is enriched with revelation (Corollary 5.13) or with hiding (Corollary 5.14).

Another contribution is the study of quantifier extrusion in SL. We introduce an extrusion algorithm for freshness (Lemma 4.4), and we prove that no extrusion algorithm exists for first order quantifiers, revelation, and hiding (Corollary 4.8).

2 The Tree Model

We study logics that describe trees labeled with public and restricted names.

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 $n \in \mathcal{N}$.*

$$\begin{array}{rlll} T, U ::= & \mathbf{0} & \textit{empty tree} & \mid n[T] \quad \textit{tree branch} \mid \\ & T \mid U & \textit{composition of trees} & \mid (\nu n)\, T \; \textit{restricted name} \end{array}$$

Free names $fn(T)$ and bound names are defined as usual. On these trees we define the usual congruence rules, with extrusion of restricted names. (Renaming) is the crucial rule, expressing the computational irrelevance of restricted names.

Table 2.1. *Congruence rules*

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

Definition 2.2 *The set of trees in* extruded normal form *(ENF) is the least set such that: (i) a tree with no restriction is in ENF, and (ii) 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 a restriction-free *matrix*, all the restricted names actually appear in the tree, and all the restricted names are mutually different. We will use *ENF* to denote the set of all terms in ENF, and $ENF(T)$ to denote the set $\{U : U \in ENF,\ U \equiv T\}$. In the full paper [17] we show that every term admits an equivalent one in *ENF*.

3 The Logic

We will study sublogics of the Ambient Logic without recursion and where no temporal operator appears. The logic is very rich, but we give here only a brief description for lack of space. For more details see [3, 12, 13].

Definition 3.1 *The set $\mathcal{A}$ of the formulas of the full logic is defined by the grammar shown in Table 3.1 (we will consider some sub-logics later on). η stands for either a name $n \in \mathcal{N}$ or a name variable $x \in \mathcal{X}$. In Table 3.1 we also define the satisfaction of a closed formula A by a model T ($T \models A$). We use $nm(A)$ to denote the set of all names n that appear in a formula.*

Table 3.1. Spatial Logic formulas and satisfaction

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

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

We will also use $T, \rho \models A$, where ρ is a ground substitution mapping $fv(A)$ into $\mathcal{N}$, as an alternative notation for $T \models A\rho$, where $A\rho$ is the closed formula obtained by applying ρ to all of its free variables.

Notation 3.2 *$SL_{\{\}}$ will denote the logic fragment without quantifiers, revelation and revelation adjunct. SL_X will denote the extension of $SL_{\{\}}$ with the logical operators in X. Hence the full logic of Definition 3.1 is $SL_{\{\mathsf{И},\circledR,\exists,\circledcirc\}}$.*

We assume that $\exists x$, $\mathsf{И}x$ 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 $\exists x.\, (A \wedge \exists y.\, B)$. 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.

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

operator	*definition*	*fundamental property (may be used as a definition)*
$\mathsf{H}x.\, A$	$\triangleq \mathsf{И}x.\, x\circledR A$	$T \models \mathsf{H}x.\, A \Leftrightarrow \exists n \notin nm(A).\, \exists U \in \mathcal{T}_{\mathcal{N}}.\, T \equiv (\nu n)\, U,\, T \models A\{x \leftarrow n\}$
$\circledcirc n$	$\triangleq \neg n\circledR\mathsf{T}$	$T \models \circledcirc n \quad \Leftrightarrow n \in fn(T)$
$n{=}m$	$\triangleq (n[\mathsf{T}])@m$	$T \models n{=}m \Leftrightarrow n{=}m$

In a nutshell, the structural operators $\mathbf{0}$, $\eta[A]$, $A \,|\, B$, allow one to explore the structure of the model, so that $T \models n[(m[\mathsf{T}] \vee p[\mathbf{0}])]$ specifies that T matches either $n[m[U]]$ or $n[p[\mathbf{0}]]$. The adjunct operators $@, \triangleright, \circledcirc$ describe how the model behaves when it is inserted into a context $n[\_], U \,|\, \_$, or $(\nu n)\, \_ .\, \triangleright$ is very expressive, since it can be used to reduce validity to model-checking (Table 3.2, line 3). Consider now a tree $T \equiv (\nu p)\, m[p[\mathbf{0}]]$ with a restricted name. This can be described by the formula $n\circledR m[n[\mathsf{T}]]$, which uses n to talk about the "anonymous" p:

$$(\nu p)\, m[p[\mathbf{0}]] \models n\circledR m[n[\mathsf{T}]] \;\Leftrightarrow\; (\nu p)\, m[p[\mathbf{0}]] \equiv (\nu n)\, m[n[\mathbf{0}]],\; m[n[\mathbf{0}]] \models m[n[\mathsf{T}]]$$

However, the satisfaction of this formula depends upon the specific name n: $T \models n\circledR n[\mathsf{T}]$, literally means that $T \equiv (\nu n)\, n[U]$ for some U, which is satisfied by any $(\nu p)\, p[U]$, unless n happens to be free in $(\nu p)\, p[U]$ (in this case, $(\nu p)\, p[U] \not\equiv (\nu n)\, n[U]$). In many situations, we really want to say things like 'T has a shape $(\nu x)\, x[U]$' where no name should be prevented from matching x by the irrelevant fact that it appears free in T. To this aim, we must use a name that is guaranteed to be fresh, which can be obtained through Gabbay-Pitts fresh name quantification: $\mathsf{И}x.\, x\circledR x[\mathsf{T}]$. The $\mathsf{И}$-$\circledR$ jargon is encoded by hiding quantification: $\mathsf{И}x.\, x\circledR x[\mathsf{T}] \triangleq \mathsf{H}x.\, x[\mathsf{T}]$.

H may be taken as primitive instead of $\mathsf{И}$ and $\circledR$, but one would lose (in a logic without adjuncts) the ability to express the property $\circledcirc\eta$. Hence, one would consider the pair H-$\circledcirc$ as an alternative to $\mathsf{И}$-$\circledR$. This motivated us to study the decidability properties of all these operators. The result is symmetric: each pair contains one operator ($\mathsf{H}/\circledR$) which is undecidable even when confined to a tiny sublogic, and an operator which we prove to be decidable ($\circledcirc/\mathsf{И}$); $\circledcirc$ and $\mathsf{И}$ are even decidable together. (We prefer the canonical choice of $\mathsf{И}$-$\circledR$ because we find their definitions more elegant, and since the encoding of the other two operators is very direct; the reverse encoding is much harder.)

$\mathsf{H}x.\, A$ is quite similar to an existential quantification over the names that are restricted in the model, but there are some subtleties. For example, two different

hiding-quantified variables cannot be bound to the same restricted name, i.e., while $n[n[\mathbf{0}]] \models \exists x.\ \exists y.\ x[y[\mathbf{0}]]$, $(\nu n)\, n[n[\mathbf{0}]] \not\models \mathsf{H}x.\, \mathsf{H}y.\, x[y[\mathbf{0}]]$: after x is bound to n, n is not restricted any more, hence y cannot be bound to n.

Hiding, freshness, appearance ($\circledcirc$), and revelation can be used to express essential properties in any specialization of this logic to specific computational structures. We present here some examples in a very informal way, just to give the flavour of the applications of the hiding operator.

When restricted names are used to represent pointers, the presence of a dangling pointer can be formalized as follows [9]; here $.n[A]$ abbreviates $n[A] \mid \mathsf{T}$, hence means: there is a branch $n[U]$ that satisfies $n[A]$.

$$\mathsf{H}x.\, (.\mathit{paper}[.\mathit{citing}[x]] \wedge \neg.\mathit{paper}[.\mathit{paperId}[x]])$$

If restricted names represent passwords in a concurrent system (e.g. in [3]), we can specify properties like 'inside k we find a password which will not be communicated', with the following sentence, where '$\Diamond A$' means 'in some process deriving from the current process A holds', and '$\mathit{send}(m, n)$' means 'm is ready for transmission on a channel n'.

$$\mathsf{H}x.\, .k[x] \wedge \neg\exists n.\, \Diamond \mathit{send}(x, n)$$

If restricted names represent α-renamable variable names, the following sentence describes any tree that represents a lambda term; $\mu X.A$ is a recursive definition, where each occurrence of X can be expanded with the body A. It says: a lambda term is either a free variable, or an application, or a lambda binder that pairs an α-renamable name with a body, where that name may appear free. The interplay between μ and H ensures that no variable appears twice in the same scope.

$$\mu LT.\ (\exists x.\, \mathit{var}[x]) \vee (\mathit{function}[LT] \mid \mathit{argument}[LT]) \vee (\mathsf{H}x.\, \mathit{lambda}[x] \mid \mathit{body}[LT])$$

We now define the standard notions of formula validity, satisfiability, of formula implication, and of formula equivalence for spatial logics.

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

Let $\forall A$ denote $\forall x_1 \ldots \forall x_n.\ A$, where $\{x_1 \ldots x_n\} = \mathit{fv}(A)$, and similarly for $\exists A$. The following properties come from [12, 6], or are easily derivable from there.

Table 3.2. *Properties of SL*

(Implication)	$A \vdash B \ \Leftrightarrow\ \mathbf{vld}(A \Rightarrow B)$	$A \dashv\vdash B \ \Leftrightarrow\ \mathbf{vld}(A \Leftrightarrow B)$
(Closure)	$\mathbf{vld}(A) \ \Leftrightarrow\ \mathbf{vld}(\forall A)$	$\mathbf{sat}(A) \ \Leftrightarrow\ \mathbf{sat}(\exists A)$
($\mathbf{vld}$ by $\models$)	$\mathbf{vld}(A) \ \Leftrightarrow\ \mathbf{0} \models \mathsf{T} \triangleright \forall A \ \Leftrightarrow\ \mathbf{0} \models \forall(\mathsf{T} \triangleright A)$	

The last property shows how validity can be reduced to model-checking using $\triangleright$ and quantification, or just $\triangleright$ alone, when the formula is closed [6].

4 Decidable Sublogics

In this section we prove decidability of $SL_{\{\diamond,\circledcirc\}}$ and we extend the result to $SL_{\{\diamond,\circledcirc,\mathsf{N}\}}$ using an extrusion algorithm for freshness quantification.

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. In the following we will show that: (i) in a spatial logic with the $\triangleright$ operator, extrusion implies decidability (Corollary 4.6); (ii) the freshness quantifier admits extrusion (Lemma 4.4), hence is decidable; (iii) undecidability of the revelation operator, existential quantifier, and hiding quantifier, implies that no extrusion algorithm can exist for them (Corollary 4.8).

4.1 Quantifier-free Decidable Sublogics

We start from the following result presented in [6].

Theorem 4.1 (Calcagno-Cardelli-Gordon). *The model-checking, validity, and satisfiability problems for closed formulas in $SL_{\{\}}$ are decidable over trees with no restricted names.*

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

Theorem 4.2 (Model-checking with Restricted Names in the Model and Revelation Adjunct in the Logic). *The model-checking problem restricted to closed formulas in $SL_{\{\diamond\}}$ is decidable over all trees (i.e., including trees with restricted names).*

Proof. (Sketch, see [17]) We follow the schema of [6], and define an equivalence relation $\sim_{h,w,N}$ (N is a set of names), an algorithm to enumerate a witness $U_i^{(h,w,N)}$ for each equivalence class of $\sim_{h,w,N}$, and a size $|A|$ for each formula A. If $|B| = (h, w, N)$, we show that model-checking $T \models A \triangleright B$ can be reduced to checking that, for each $U \in U_i^{(h,w,N)}$, $U \models A \Rightarrow U \mid T \models B$.

In the full paper [17] we show that $\textcircled{c}\eta$ can be encoded in $SL_{\{\diamond\}}$ making use of $\textcircled{c}^m\eta \triangleq (\eta[\mathbf{0}] \triangleright ((\neg(\neg\mathbf{0} \mid \neg\mathbf{0})) \diamond \eta))@m$. Hence we have the following corollary.

Corollary 4.3 (Adding $\textcircled{c}$). *The model-checking problem for closed formulas in $SL_{\{\diamond,\circledcirc\}}$ is decidable over all trees (i.e., including trees with restricted names).*

4.2 Quantifier Extrusion

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*

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

If all the rules were double implications ($\dashv\vdash$), we could use them to extrude the existential quantifier in any formula. 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 $B \vdash A$ has a counterexample (see the full paper [17]).

The table above shows that $\forall$-$\exists$ extrusion is not trivial, 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 ®. 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*

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

Once more, all the single implications are strict (see the full paper [17]).

However, the three single-implication rules admit a double-implication version, as shown in the Table 4.3.

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-$\exists$)
$y \notin fv(B)$	$(\text{И} y.\, A) \triangleright B$	$\dashv\vdash$	$\text{И} y.\, ((\neg ⓒ y \wedge A) \triangleright B)$	(И-$\triangleright l$)
$y \notin fv(A)$	$A \triangleright (\text{И} y.\, B)$	$\dashv\vdash$	$\text{И} y.\, ((\neg ⓒ y \wedge A) \triangleright B)$	(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. In the full paper we prove the correctness of all the extrusion rules.

Lemma 4.4 (Extrusion of freshness). *There is an algorithm to transform any formula in the full logic into 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. Termination is easy.

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

4.3 Decidable Sublogics With Quantifiers and Impossibility of Extrusion

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

Theorem 4.5 (Decidability of Prenex Model-Checking). *Model-checking over all trees is decidable for the closed formulas F generated by the following grammar ($\exists$, H, Ⓡ, И : outermost only; ⓒ, ⊘: unlimited):*

$$F ::= \exists x.\, F \mid x Ⓡ F \mid \mathsf{H} x.\, F \mid \text{И} x.\, F \mid \neg F \mid A$$
$$A ::= \mathbf{0} \mid \eta[A] \mid A \,|\, A \mid A \wedge A \mid \neg A \mid ⓒ\eta \mid A \triangleright A \mid A @ \eta \mid A ⊘ \eta$$

Proof. (Sketch, see [17]) By induction on the size of F and by cases. Case $\neg F$ is trivial. Case A is Corollary 4.3. To model-check $T \models \exists x.\, F$, check $T \models F\{x \leftarrow n\}$ for $n \in (fn(T) \cup nm(F) \cup \{m\})$, where m is fresh. To model-check $T \models n Ⓡ F$, transform T in ENF $(\nu n_1) \ldots (\nu n_k)\, U$ and check that $n \notin fn(T)$ and that either $T \models F$ or $\exists i.\, (\nu n_1) \ldots (\nu n_{i-1})\, (\nu n_{i+1}) \ldots (\nu n_k)\, U\{n_i \leftarrow n\} \models F$. $T \models \mathsf{H} x.\, F$ is similar. To model-check $T \models \text{И} x.\, F$, choose a name $n \notin fn(T) \cup nm(F)$ and model-check $T \models F\{x \leftarrow n\}$.

Theorem 4.5 has the following Corollary.

Corollary 4.6 (Extrusion implies Decidability). *The existence of an extrusion algorithm, i.e. an algorithm that transforms every formula into an equivalent formula generated by the grammar of Theorem 4.5, for any sublogic L of $SL_{\{\exists,\, \text{И},\, ⊘,\, \mathsf{H},\, Ⓡ\}}$ containing $\triangleright$ implies the decidability of L.*

Proof. To decide $\mathbf{vld}(A)$ for a closed formula A, reduce it to $\mathbf{0} \models \mathsf{T} \triangleright A$, apply the extrusion algorithm, and use the algorithm of Theorem 4.5.

As a consequence, the addition of freshness preserves the decidability of the logic of Corollary 4.3.

Corollary 4.7 (Decidability of Fresh Quantifiers). *Model-checking and validity for the closed formulas in $SL_{\{\mathsf{N},\textcircled{\scriptsize N},\textcircled{\scriptsize @}\}}$ are decidable over all trees.*

To sum up, fresh quantification alone is not enough to lose decidability, even if combined with a limited form of revelation ($\textcircled{c}\eta$).

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

By undecidability of $SL_{\{\exists\}}$ (follows from [14]), of $SL_{\{\textcircled{R}\}}$ (follows from Corollary 5.13), and of $SL_{\{\mathsf{H}\}}$ (follows from Corollary 5.14), the three logics of Corollary 4.6 are all undecidable. Hence, we have the following Corollary.

Corollary 4.8 (No Extrusion). *No extrusion algorithm (as defined in Corollary 4.6) exists for SL_X if X includes $\{\exists\}$,$\{\textcircled{R}\}$, or $\{\mathsf{H}\}$.*

5 Undecidability Results

5.1 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 is a key technical tool in order to prove (later) that the decidability of this tiny logic is already as hard as decidability of first order logic.

Notation 5.1 (Path-Formulas) *A path-formula p is a formula denoting 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).*

$$.\eta \triangleq \eta[\mathbf{0}] \mid \mathsf{T} \qquad .\eta'.\eta \triangleq \eta'[\eta[\mathbf{0}] \mid \mathsf{T}] \mid \mathsf{T}$$

When a tree satisfies $.m.n$ we say that it "contains a path $m.n$"; the path ends with a leaf. The minimal tree containing such path, $m[n[\mathbf{0}]]$ (which we also write $m[n]$), is called a "line for the path $m.n$", and similarly $m[\mathbf{0}]$ (abbreviated as m) 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 (longer paths, and paths with free names not in N, are cut away). By this construction, for any formula A with shape $.n_1.n_2$, $n_1\textcircled{R}.n_2.n_3$,

$n_1 ® n_2 ® .n_3.n_4$ (where n_i may be equal to n_j), $Cut_{nm(A)}(T)$ 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 $\{n_i\}^{i \in \{1..j\}} = fn(T) \cap N$, so that the validity of formulas $n®\mathsf{T}$, for $n \in N$, is preserved as well. In other words, we cut away long paths and paths with free names not in N, and we rewrite trees like "$n[m \mid p]$" as lines "$n[m] \mid n[p] \;\mid\; n \mid m \mid p$".

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 (Theorem 5.4).

Before giving the formal definition, we give some examples. Cutting is only defined up-to-congruence.

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

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

Definition 5.2 (Path cutting for ENF). *For each tree in ENF, for each set of names N, we define the operation $enfCut_N()$ as follows. $Par\{T : cond\}$ combines (using $\mid$) all instances $(T)\sigma$ of T such that $(cond)\sigma$ is satisfied.*

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

Definition 5.3. $Cut_N(T) \triangleq \{enfCut_N(U) : U \in ENF(T)\}$

In the full paper [17] we prove that $Cut_N()$ preserves congruence, i.e. that $T \equiv T' \wedge U \in Cut_N(T) \wedge U' \in Cut_N(T') \Rightarrow U \equiv U'$. Hence, $Cut_N(T)$ only contains one tree modulo equivalence, and we will abuse notation by using $Cut_N(T)$ to denote that tree.

Theorem 5.4 (Standard Model). *Let A be a closed formula generated by the following grammar:*

$$A ::= .\eta_1.\eta_2 \mid A \wedge A \mid \eta ® A \mid \mathsf{M}x.\, A \mid \neg A$$

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

Proof. For the ($\Rightarrow$) direction we prove, by induction on the size of A, the following stronger property: $\forall \mathbf{N}$ finite. $T \models A \Rightarrow Cut_{nm(A)\cup\mathbf{N}}(T) \models A$, for an equivalent logic without negation, but with De Morgan duals for each operator (see [17]). The other direction is easily derived by contradiction and definition of negation.

5.2 Undecidability of Revelation

Since we are studying undecidability, we focus here on weak versions of the logic. We will prove undecidability for a logic with just $\wedge$, $\neg$, Ⓡ, and path formulas. 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 (this logic is already undecidable [2]):

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

We define satisfaction of a closed 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 $\sigma{\downarrow} \supseteq fv(\phi)$ (where $f{\downarrow}$ is the domain of a function f) as follows.

$$\begin{array}{lll}
\mathcal{D},\mathcal{R},\sigma \models \exists x.\, \phi & \Leftrightarrow_{\text{def}} & \text{exists } c{\in}\mathcal{D}.\ \mathcal{D},\mathcal{R},\sigma\{x{\leftarrow}c\} \models \phi \\
\mathcal{D},\mathcal{R},\sigma \models \phi \wedge \psi & \Leftrightarrow_{\text{def}} & \mathcal{D},\mathcal{R},\sigma \models \phi \ \text{ and } \ \mathcal{D},\mathcal{R},\sigma \models \psi \\
\mathcal{D},\mathcal{R},\sigma \models \neg\phi & \Leftrightarrow_{\text{def}} & \text{not } (\mathcal{D},\mathcal{R},\sigma \models \phi) \\
\mathcal{D},\mathcal{R},\sigma \models R(x,x') & \Leftrightarrow_{\text{def}} & (\sigma(x),\sigma(x')) \in \mathcal{R}
\end{array}$$

Essentially, we will translate a model $\mathcal{D},\mathcal{R}$ into an ENF term $(\boldsymbol{\nu} n_i)\, [\![\mathcal{D}]\!] \mid [\![\mathcal{R}]\!]$, 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 5.1.

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 important 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 $(\boldsymbol{\nu} n_i)\, T$. So, while in FOL we match variables in the formula with values in the domain, in the SL translation we will match bound names in the model with the free names used to reveal them in the formula.

Technically, we translate a FOL closed formula ϕ into a formula $[\![\phi]\!]$, where all the closed variables of ϕ are left open, and a ground substitution $(\!|\phi|\!)^{\mathbf{P}}$ such that $(\!|\phi|\!)^{\mathbf{P}}{\downarrow} \supseteq fv(\phi)$, so that $[\![\phi]\!](\!|\phi|\!)^{\mathbf{P}}$ is closed. We then reduce satisfiability of ϕ to satisfiability of (a variant of) $[\![\phi]\!](\!|\phi|\!)^{\mathbf{P}}$.

A second difference is the fact that the same value can be bound to two different FOL variables, while the same restricted 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

$$\exists x_1.\ ((\exists x_2 \neq x_1.\ \phi) \vee \phi\{x_2 \leftarrow x_1\}),\ \text{i.e. as: } x_1 ® ((x_2 ® [\![\phi]\!]) \vee [\![\phi\{x_2 \leftarrow x_1\}]\!]),$$

To this aim, in the translation algorithm a parameter $\mathbf{Y}$ keeps track of the quantified variables met during the translation. The first line of Table 5.1 defines how $\mathbf{Y}$ is grown with each quantification, and how it is used to generate a disjunction of $[\![\phi\{x_2 \leftarrow x_1\}]\!]^{\mathbf{Y}}$ clauses.

Finally, while x in $\exists x.\ \phi$ can only be associated to an element that is in the domain, n in $n ® A$ can also be associated to a name that does not appear in the model at all (since, for each $n \notin fv(T)$, $T \equiv (\nu n)\, T$). We solve this problem by translating $\exists x.\ \phi$ as $x ® ([\![\phi]\!] \wedge .x)$ 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.

Notation 5.5 *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 $M : \mathbf{M} \overset{in}{\to} \mathbf{N}$ 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 $N{\uparrow}$ to denote its actual range, i.e.:*

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

When $M, N : \mathbf{M} \rightharpoonup \mathbf{N}$, we use $M \oplus N$ to denote function extension, as follows:
$(M \oplus N)(x) \triangleq$ *if $x \in N{\downarrow}$ then $N(x)$ else $M(x)$*
Hence, $M \oplus \{c \leftarrow n\}$ yields n on c and coincides with M elsewhere.

Notation 5.6 $(\boldsymbol{\nu}_{i \in I} n_i)\, T \triangleq (\nu n_{i_1}) \ldots (\nu n_{i_j})\, T$ *with $I = \{i_1, \ldots, i_j\}$, $n : I \overset{in}{\to} \mathcal{N}$.*

We can finally define our translation. We map an FOL formula to an SL 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 $M \oplus N$ (see Notation 5.5) 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.

Definition 5.7 (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 $M \oplus N : \mathcal{D} \overset{in}{\to} \mathcal{N}$. $\mathbf{P}$ 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.*

Table 5.1. Formula translation

formulas into formulas

$$
\begin{array}{lcl}
[\![\exists x.\,\phi]\!]^{\mathbf{Y}} & \triangleq & x\circledR([\![\phi]\!]^{\mathbf{Y}\cup\{x\}} \wedge .x) \vee \bigvee_{y\in\mathbf{Y}} [\![\phi\{x\leftarrow y\}]\!]^{\mathbf{Y}} \\
[\![\phi \wedge \psi]\!]^{\mathbf{Y}} & \triangleq & [\![\phi]\!]^{\mathbf{Y}} \wedge [\![\psi]\!]^{\mathbf{Y}} \\
[\![\neg\phi]\!]^{\mathbf{Y}} & \triangleq & \neg[\![\phi]\!]^{\mathbf{Y}} \\
[\![R(x,x')]\!]^{\mathbf{Y}} & \triangleq & .x.x'
\end{array}
$$

formulas into substitutions

$$
\begin{array}{lcl}
(\!|\exists x.\,\phi|\!)^{\mathbf{P}} & \triangleq & (\!|\phi|\!)^{\mathbf{P}} \oplus \{x\leftarrow m'\} \quad \textit{choose } m' \in \mathcal{N} \setminus (\mathbf{P} \cup (\!|\phi|\!)^{\mathbf{P}}\uparrow) \\
(\!|\phi \wedge \psi|\!)^{\mathbf{P}} & \triangleq & (\!|\phi|\!)^{\mathbf{P}} \oplus (\!|\psi|\!)^{\mathbf{P}\cup(\!|\phi|\!)^{\mathbf{P}}\uparrow} \\
(\!|\neg\phi|\!)^{\mathbf{P}} & \triangleq & (\!|\phi|\!)^{\mathbf{P}} \\
(\!|R(x,x')|\!)^{\mathbf{P}} & \triangleq & \emptyset
\end{array}
$$

interpretations, domains, and relations into trees

$$
\begin{array}{lcl}
[\![\mathcal{D},\mathcal{R}]\!]^{M,N} & \triangleq & (\boldsymbol{\nu}_{c\in N\downarrow} N(c))\,([\![\mathcal{D}]\!]^{M\oplus N} \mid [\![\mathcal{R}]\!]^{M\oplus N}) \\
[\![\emptyset]\!]^{M} & \triangleq & \mathbf{0} \\
[\![\{c\}\cup\mathcal{D}]\!]^{M} & \triangleq & M(c)[\mathbf{0}] \mid [\![\mathcal{D}]\!]^{M} \\
[\![\{(c,c')\}\cup\mathcal{R}]\!]^{M} & \triangleq & M(c)[M(c')[\mathbf{0}]] \mid [\![\mathcal{R}]\!]^{M}
\end{array}
$$

assignments into assignments

$$
\begin{array}{lcl}
[\![\sigma\oplus\{x\leftarrow c\}]\!]^{M} & \triangleq & [\![\sigma]\!]^{M} \oplus \{x\leftarrow M(c)\} \\
[\![\emptyset]\!]^{M} & \triangleq & \emptyset
\end{array}
$$

Theorem 5.8. *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. In [17] we prove by induction and by cases the more general property $(\mathcal{D},\mathcal{R}),\sigma \models \phi \;\Leftrightarrow\; [\![\mathcal{D},\mathcal{R}]\!]^{M,N} \models [\![\phi]\!]^{\mathbf{Y}}[\![\sigma]\!]^{M}(\!|\phi|\!)^{\mathbf{P}}$ under some hypotheses that essentially constrain σ to be a substitution mapping free variables of ϕ (and those in $\mathbf{Y}$) into M-elements (i.e. elements in $M\downarrow$) without name-clashes with $N\downarrow$ and $\mathcal{N}\setminus\mathbf{P}$. By choosing the empty function for M, the empty set for $\mathbf{Y}$, $\mathbf{P} = N\uparrow$, and the empty assignment for σ, we have that:

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

This is equivalent to the thesis as a consequence of the Gabbay-Pitts property.

Corollary 5.9. *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, because $[\![\phi]\!]^{\emptyset}(\!|\phi|\!)^{\emptyset}$ may be satisfied by SL models which are not the translation of any FOL model. Consider $(\exists x.\ \mathsf{T}) \wedge \neg(\exists y.\ \mathsf{T})$. It is clearly unsatisfiable, but it is translated (under $\mathbf{Y} = \emptyset$, $M = \emptyset$) as $m\circledR(\mathsf{T} \wedge .m) \wedge \neg n\circledR(\mathsf{T} \wedge .n)$, which is satisfied by the model $(\nu m')\, m'[\mathbf{0}] \mid n[\mathbf{0}]$, since the free occourrence of n 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 5.8, since $(\nu m')\, m'[\mathbf{0}] \mid n[\mathbf{0}]$ is not the translation of any FOL model under $M = \emptyset$, because $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset, N}$ has no free names. The fact that the model is not closed is actually the core of the problem. We solve this problem by enriching the mapping with a conjunct that rules some of the non-closed models out.

Definition 5.10. $[\![\phi]\!]^+ \triangleq [\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset} \wedge \bigwedge_{m \in nm([\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset})} \neg ⓒ m$

This new translation will ensure that any SL model of the translated formula is "closed enough", i.e. all its free names are disjoint from the names in the formula. Now we use the cut operation and Theorem 5.4 to show that these "residual" free names are irrelevant, hence that every model of the enriched translation actually corresponds to a FOL model, finally reducing SAT_{SL} to SAT_{FOL}.

Lemma 5.11. *Let $T = Cut_{N'}(U)$ for some N', U; then:*

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

Theorem 5.12 (Reduction of FOL 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 Theorem 5.8, $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset, N}$ satisfies $[\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset}$. Since $[\![\mathcal{D}, \mathcal{R}]\!]^{\emptyset, N}$ is closed, it also satisfies $\neg ⓒ m$ for any m.

($\Leftarrow$) Assume $SAT_{SL}([\![\phi]\!]^+)$ and let $N = nm([\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset})$. Then, there exists T such that $T \models [\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset}$ and $T \models \bigwedge_{m \in N} \neg ⓒ m$, i.e., $fn(T) \cap N = \emptyset$. Consider now $U = Cut_N(T)$. By Theorem 5.4: $U \models [\![\phi]\!]^{\emptyset} (\!|\phi|\!)^{\emptyset}$, by $fn(T) \cap N = \emptyset$: $fn(U) = \emptyset$, and by Lemma 5.11, U is the translation of a FOL interpretation $\mathcal{D}, \mathcal{R}$. By Theorem 5.8, $\mathcal{D}, \mathcal{R} \models \phi$; hence $SAT_{FOL}(\phi)$.

Corollary 5.13 (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.*

5.3 Undecidability of Hiding Quantification

In the full paper [17] we prove undecidability of hiding quantification in a similar way. The translation is simpler since we do not need the $(\!|\phi|\!)^{\mathbf{P}}$ substitution any more. The key difference is the fact that an existential quantification is directly translated as a closed formula:

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

By reasoning as in Section 5.2, we prove the following Corollary.

Corollary 5.14 (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 Conclusions and Related Work

In SL hiding can be expressed as freshness plus revelation. The main result of this paper is: freshness without revelation gives a rich decidable logic (Corollary 4.7) while revelation makes a minimal logic undecidable (Corollary 5.13). We also proved that hiding is undecidable, and some results about extrusion that we summarize below.

The decidability result 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 is obtained by 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 follows immediately.

We summarize decidability and extrusion results for spatial logics in the following table. Detailed proofs of our results are shown in [17].

Table 6.1. *A summary of decidability/extrusion results*

Logic	Decidable?	Operator	Extrusion algorithm
$SL_{\{\}}$	Yes, proved in [6]	$\mathsf{И}$	Yes, see Table 4.2 and [18]
$SL_{\{\mathsf{И},\circledcirc,\copyright\}}$	Yes, proved in Corollary 4.7	®	No, by Corollary 4.8
$SL_{\{\exists\}}$	No, follows from [14]	H	No, by Corollary 4.8
$SL_{\{®\}}$	No, follows from Corollary 5.13	$\exists$	No, by Corollary 4.8
$SL_{\{\mathsf{H}\}}$	No, follows from Corollary 5.14		

An extrusion algorithm for the freshness quantifier in $SL_{\{®,\circledcirc\}}$ is used in [18] by Lozes to prove a surprising adjunct elimination theorem for $SL_{\{\mathsf{И},®,\circledcirc\}}$.

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 6.1. *No effective adjunct-elimination procedure exists for* $SL_{\{\mathsf{И},®,\circledcirc\}}$.

Proof. An effective adjunct-elimination procedure would reduce model-checking of $SL_{\{\mathsf{И},®,\circledcirc\}}$, which we proved to be undecidable, to model-checking the same logic without adjuncts, which is decidable.

A calculus to manipulate trees with hidden names has been presented in [9], whose type system includes the full SL. Hence, type inclusion in that calculus and validity in SL are mutually reducible. Decidability of subtype-checking was left as an open problem in [9]. Our results imply that it is undecidable.

Acknowledgments We would like to thank Luís Caires, Cristiano Calcagno, Luca Cardelli, Dario Colazzo, and Philippa Gardner, 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. Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997.
3. 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.
4. 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.
5. 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.
6. 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.
7. L. Cardelli. Describing semistructured data. *SIGMOD Record, Database Principles Column*, 30(4), 2001.
8. 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.
9. 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.
10. 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.
11. L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. of POPL*. ACM Press, 2000.
12. L. Cardelli and A. D. Gordon. Logical properties of name restriction. In *Proc. of TCLA'01*, volume 2044 of *LNCS*, pages 46–60. Springer, 2001.
13. L. Cardelli and A. D. Gordon. Ambient logic. Submitted for publication, available from the authors, 2002.
14. 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.
15. G. Conforti and G. Ghelli. Spatial logics to reason about semistructured data. In *Proc. of SEBD'03*. Rubettino Editore, 2003.
16. 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.
17. G. Ghelli and G. Conforti. Decidability of freshness, undecidability of revelation. Technical Report TR-03-11. Dipartimento di Informatica, Università di Pisa, 2003.
18. É. Lozes. Adjuncts elimination in the static ambient logic. In *Proc. of EXPRESS'03*, 2003. To appear.
19. 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.
20. 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.
21. John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc. LICS'02*, pages 55–74. IEEE Computer Society, 2002.