Deciding Validity in a Spatial Logic for Trees

Cristiano Calcagno[1], Luca Cardelli[2], and Andrew D. Gordon[2]

[1] Queen Mary, University of London
[2] Microsoft Research

Version of May 10, 2002

Abstract. We consider a propositional spatial logic for finite trees. The logic includes $\mathcal{A} \mid \mathcal{B}$ (spatial composition), and $\mathcal{A} \triangleright \mathcal{B}$ (the implication induced by composition), and $\mathbf{0}$ (the unit of composition). We show that the satisfaction and validity problems are equivalent, and decidable. The crux of the argument is devising a finite enumeration of trees to consider when deciding whether a spatial implication is satisfied. We introduce a sequent calculus for the logic, and show it to be sound and complete with respect to an interpretation in terms of satisfaction. Finally, we describe a complete proof procedure for the sequent calculus.

1 Introduction

In a spatial logic, the truth of a formula depends on its location. Models for spatial logics include computational structures such as heaps [Rey00,IO01,ORY01], trees [CG01a], graphs [CGG02], concurrent objects [CM98], as well as process calculi such as the π-calculus [CC01,CC02] and the ambient calculus [CG00,CG01b]. Applications of spatial logics include specifying and verifying imperative and concurrent programs, and querying semistructured data.

This paper concerns a spatial logic describing properties of finite edge-labelled trees. In our textual notation, $n_1[P_1] \mid \cdots \mid n_k[P_k]$ is a tree consisting of k edges, labelled $n_1, \ldots, n_k$, leading to k subtrees $P_1, \ldots, P_k$, respectively. Our logic starts with propositional primitives: conjunction $\mathcal{A} \wedge \mathcal{B}$, implication $\mathcal{A} \Rightarrow \mathcal{B}$, and falsity $\mathbf{F}$. To this basis, we add spatial primitives: *composition* $\mathcal{A} \mid \mathcal{B}$ (satisfied by composite trees $P \mid Q$ where P and Q satisfy $\mathcal{A}$ and $\mathcal{B}$, respectively), *guarantee* $\mathcal{A} \triangleright \mathcal{B}$ (the spatial implication corresponding to composition, satisfied by trees that, whenever composed with any tree that satisfies $\mathcal{A}$, result in trees that satisfy $\mathcal{B}$) and *void* $\mathbf{0}$ (the unit of composition, satisfied by the empty tree). We complete the logic with primitives for labelled edges: *location* $n[\mathcal{A}]$ (satisfied by a tree $n[P]$ if P satisfies $\mathcal{A}$) and *placement* $\mathcal{A}@n$ (satisfied by a tree P if the tree $n[P]$ satisfies $\mathcal{A}$).

We consider the satisfaction problem (whether a given tree satisfies a given formula) and the validity problem (whether every tree satisfies a given formula). Since satisfaction of the guarantee operator $\mathcal{A} \triangleright \mathcal{B}$ is defined as an infinite quantification over all trees, neither problem is obviously decidable. Our first significant result, is that both are, in fact, decidable (Theorem 2). In effect, we show how

to decide validity by model checking. The main auxiliary result (Theorem 1) is that we need consider only a finite enumeration of trees when model checking a formula $\mathcal{A} \triangleright \mathcal{B}$.

Subsequently, we introduce a sequent calculus for our spatial logic, and show how to decide validity by deduction in this calculus. The finite enumeration of trees introduced in the first half is built into the right rule for $\mathcal{A} \triangleright \mathcal{B}$. Our sequent calculus has a standard interpretation in terms of the satisfaction predicate. By appeal to Theorem 1, we show the sequent calculus to be sound (Theorem 3) and complete (Theorem 4) with respect to its interpretation. Moreover, we obtain and verify a complete algorithm for finding proofs in the sequent calculus (Theorem 5). The resulting algorithm for validity is better suited to optimisations than the algorithm based directly on model checking.

Section 2 gives formal definitions of our logic and its model. Section 3 develops our first algorithm for validity, based on model checking. Section 4 develops our second algorithm, based on our sequent calculus. Section 5 concludes.

2 Ground Propositional Spatial Logic (Review)

This section introduces our spatial logic and its model. First, we define our notation for edge-labelled finite trees. Second, we introduce the formulas of the logic and their semantics: the satisfaction predicate, $P \models \mathcal{A}$, means that the tree P satisfies the formula $\mathcal{A}$. Third, we define the validity predicate, $\mathbf{vld}(\mathcal{A})$, to mean $P \models \mathcal{A}$ for every tree P. By constructing suitable formulas, we note that satisfaction and validity are interderivable.

2.1 Edge-Labelled Finite Trees

Let m, n range over an infinite set N of names. The model of our logic is the set of edge-labelled trees, finitely branching and of finite depth.

Trees:

$P, Q ::=$	tree
$\mathbf{0}$	empty tree
$P \mid Q$	composition
$m[P]$	edge labelled by m, atop tree P

Let $fn(P)$ be the set of names free in P. For any $X \subseteq \mathsf{N}$, let $\mathsf{Tree}_X \triangleq \{P \mid fn(P) \subseteq X\}$.

Structural Equivalence: $P \equiv Q$

$P \equiv P$	(Struct Refl)
$Q \equiv P \Rightarrow P \equiv Q$	(Struct Symm)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Struct Trans)

$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \Rightarrow M[P] \equiv M[Q]$	(Struct Amb)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)

Lemma 1. *If $P \in \mathsf{Tree}_X$ and $P \equiv Q$ then $Q \in \mathsf{Tree}_X$.*

2.2 Logical Formulas and Satisfaction

Logical Formulas:

$\mathcal{A}, \mathcal{B} ::=$	formula
$\mathbf{F}$	false
$\mathcal{A} \wedge \mathcal{B}$	conjunction
$\mathcal{A} \Rightarrow \mathcal{B}$	implication
$\mathbf{0}$	void
$\mathcal{A} \mid \mathcal{B}$	composition
$\mathcal{A} \triangleright \mathcal{B}$	guarantee
$n[\mathcal{A}]$	location
$\mathcal{A}@n$	placement

The derived propositional connectives $\mathbf{T}$, $\neg\mathcal{A}$, $\mathcal{A} \vee \mathcal{B}$, are defined in the usual way. Name equality can be defined by $m = n \triangleq m[\mathbf{T}]@n$; this formula holds if and only if $m = n$. We write $\mathcal{A}\{m \leftarrow m'\}$ for the outcome of substituting each occurrence of the name m in the formula $\mathcal{A}$ with the name m'.

We define the satisfaction predicate, $P \models \mathcal{A}$, as follows.

Satisfaction: $P \models \mathcal{A}$

$P \models \mathbf{F}$	never
$P \models \mathcal{A} \wedge \mathcal{B}$	$\triangleq P \models \mathcal{A} \wedge P \models \mathcal{B}$
$P \models \mathcal{A} \Rightarrow \mathcal{B}$	$\triangleq P \models \mathcal{A} \Rightarrow P \models \mathcal{B}$
$P \models \mathbf{0}$	$\triangleq P \equiv \mathbf{0}$
$P \models \mathcal{A} \mid \mathcal{B}$	$\triangleq \exists P', P''.P \equiv P' \mid P'' \wedge P' \models \mathcal{A} \wedge P'' \models \mathcal{B}$
$P \models \mathcal{A} \triangleright \mathcal{B}$	$\triangleq \forall P'.P' \models \mathcal{A} \Rightarrow P \mid P' \models \mathcal{B}$
$P \models n[\mathcal{A}]$	$\triangleq \exists P'.P \equiv n[P'] \wedge P' \models \mathcal{A}$
$P \models \mathcal{A}@n$	$\triangleq n[P] \models \mathcal{A}$

A basic property is that structural congruence preserves satisfaction:

Lemma 2. *If $P \models \mathcal{A}$ and $P \equiv P'$ then $P' \models \mathcal{A}$.*

Proof. An easy induction on the structure of $\mathcal{A}$. □

It is useful to know that every tree P has a characteristic formula $\underline{P}$. Let $\underline{\mathbf{0}} \triangleq \mathbf{0}$, $\underline{P \mid Q} \triangleq \underline{P} \mid \underline{Q}$, and $\underline{m[P]} \triangleq m[\underline{P}]$. The formula $\underline{P}$ identifies P up to structural equivalence:

Lemma 3. *For all P and Q, $Q \models \underline{P}$ if and only if $Q \equiv P$.*

Proof. An easy induction on the structure of P. □

Now, to turn the definition of satisfaction into an algorithm, that is, to build a model checker for the logic, we must show that the three quantifications in the clauses for $\mathcal{A} \mid \mathcal{B}$, $\mathcal{A} \triangleright \mathcal{B}$, and $n[\mathcal{A}]$ can be reduced to finite problems. It is not hard to reduce the clauses for $\mathcal{A} \mid \mathcal{B}$ and $n[\mathcal{A}]$ to finite quantifications [CG00], but it seems far from obvious how to reduce satisfaction of $\mathcal{A} \triangleright \mathcal{B}$ to a finite problem. The principal result of the paper, Theorem 1, is that for any $\mathcal{A}'$, $\mathcal{A}''$ there is a finite set $T(\mathcal{A}' \triangleright \mathcal{A}'')$ such that:

$$P \models \mathcal{A}' \triangleright \mathcal{A}'' \iff \forall P' \in T(\mathcal{A}' \triangleright \mathcal{A}'').P' \models \mathcal{A}' \Rightarrow P' \mid P \models \mathcal{A}''$$

2.3 Validity of a Formula

The validity predicate, $\mathbf{vld}(\mathcal{A})$, means every tree satisfies the formula $\mathcal{A}$.

Validity: $\mathbf{vld}(\mathcal{A})$

$$\mathbf{vld}(\mathcal{A}) \triangleq \forall P.P \models \mathcal{A}$$

The next two lemmas exhibit formulas to encode validity in terms of satisfaction, and the converse.

Lemma 4 (Validity from Satisfaction). $\mathbf{vld}(\mathcal{A})$ *if and only if* $\mathbf{0} \models \mathbf{T} \triangleright \mathcal{A}$

Proof. With appeal to Lemma 2, we get: $\mathbf{vld}(\mathcal{A}) \Leftrightarrow (\forall P.P \models \mathcal{A}) \Leftrightarrow (\forall P.P \models \mathbf{T} \Rightarrow P \mid \mathbf{0} \models \mathcal{A}) \Leftrightarrow \mathbf{0} \models \mathbf{T} \triangleright \mathcal{A}$. □

Lemma 5 (Satisfaction from Validity). $P \models \mathcal{A}$ *if and only if* $\mathbf{vld}(\underline{P} \Rightarrow \mathcal{A})$.

Proof. With appeal to Lemmas 2 and 3, we get: $\mathbf{vld}(\underline{P} \Rightarrow \mathcal{A}) \Leftrightarrow (\forall Q.Q \models \underline{P} \Rightarrow Q \models \mathcal{A}) \Leftrightarrow (\forall Q.Q \equiv P \Rightarrow Q \models \mathcal{A}) \Leftrightarrow P \models \mathcal{A}$. □

Hence, the validity and satisfaction problems are equivalent. The goal of the paper is to show that both are decidable.

3 Deciding Validity by Model Checking

The crux of our problem is the infinite quantification in the definition of satisfaction for $\mathcal{A} \triangleright \mathcal{B}$. We bound this infinite quantification in three steps, which lead to an alternative definition in terms of a finite quantification. This leads to a model checking procedure, and hence to an algorithm for validity.

– In Section 3.1, we bound the alphabet of distinct names that may occur in trees that need to be considered. Let $fn(\mathcal{A})$ be the set of names occurring free in any formula $\mathcal{A}$. Let m be some other name. Proposition 1 asserts that $P \models \mathcal{A} \triangleright \mathcal{B}$ if and only if $Q \models \mathcal{A} \Longrightarrow P \mid Q \models \mathcal{B}$ for all trees Q with edge-labels drawn from the set $fn(\mathcal{A}) \cup \{m\}$.
– In Section 3.2, we introduce a measure of the size of a tree, and bound both the alphabet and size of trees that need to be considered. Proposition 4 asserts that $P \models \mathcal{A} \triangleright \mathcal{B}$ if and only if $Q \models \mathcal{A} \Longrightarrow P \mid Q \models \mathcal{B}$ for all the trees Q smaller than a size determined by $\mathcal{A}$ and with edge-labels drawn from a particular finite alphabet.
– In Section 3.3, we give a procedure to enumerate a finite set of structural equivalence classes of trees determined by a formula. Theorem 1 asserts that $P \models \mathcal{A} \triangleright \mathcal{B}$ if and only if $Q \models \mathcal{A} \Longrightarrow P \mid Q \models \mathcal{B}$ for all the representatives Q of these equivalence classes. Hence, we prove in Theorem 2 that satisfaction, and hence validity, is decidable.

3.1 Bounding the Names to Consider

Lemma 6. *If $n \notin \{m, m'\}$ then:*

$$P\{m{\leftarrow}m'\} \equiv n[Q] \iff \exists P'.P \equiv n[P'] \wedge P'\{m{\leftarrow}m'\} \equiv Q$$

Proof.

$$\begin{aligned}
&P\{m{\leftarrow}m'\} \models n[Q] \\
&\iff \exists m'', P'.P \equiv m''[P'] \wedge m''\{m{\leftarrow}m'\} = n \wedge P'\{m{\leftarrow}m'\} \equiv Q \\
&\iff \exists P'.P \equiv n[P'] \wedge P'\{m{\leftarrow}m'\} \equiv Q
\end{aligned}$$

□

Lemma 7.

$$\begin{aligned}
&P\{m{\leftarrow}m'\} \equiv Q' \mid Q'' \iff \\
&\exists P', P''.P \equiv P' \mid P'' \wedge P'\{m{\leftarrow}m'\} \equiv Q' \wedge P''\{m{\leftarrow}m'\} \equiv Q''
\end{aligned}$$

Proof. Immediate since substitution preserves the structure of trees. □

Lemma 8. *If $m, m' \notin fn(\mathcal{A})$ then $P \models \mathcal{A} \iff P\{m{\leftarrow}m'\} \models \mathcal{A}$.*

Proof. By induction on the structure of $\mathcal{A}$. We only consider the interesting cases.

Case $\mathcal{A} \mid \mathcal{B}$. We have $m, m' \notin fn(\mathcal{A}) \cup fn(\mathcal{B})$. With appeal to Lemma 2 and Lemma 7, and the induction hypothesis, we calculate:

$$\begin{aligned}
P \models \mathcal{A} \mid \mathcal{B} &\iff \exists P', P''.P \equiv P' \mid P'' \wedge P' \models \mathcal{A} \wedge P'' \models \mathcal{B} \\
&\iff \exists P', P''.P \equiv P' \mid P'' \wedge P'\{m{\leftarrow}m'\} \models \mathcal{A} \wedge P''\{m{\leftarrow}m'\} \models \mathcal{B} \\
&\iff \exists P', P'', Q', Q''.P \equiv P' \mid P'' \wedge Q' \equiv P'\{m{\leftarrow}m'\} \wedge \\
&\qquad Q'' \equiv P''\{m{\leftarrow}m'\} \wedge Q' \models \mathcal{A} \wedge Q'' \models \mathcal{B} \\
&\iff \exists Q', Q''.P\{m{\leftarrow}m'\} \equiv Q' \mid Q'' \wedge Q' \models \mathcal{A} \wedge Q'' \models \mathcal{B} \\
&\iff P\{m{\leftarrow}m'\} \models \mathcal{A} \mid \mathcal{B}
\end{aligned}$$

Case $\mathcal{A} \triangleright \mathcal{B}$**.** We have $m, m' \notin fn(\mathcal{A}) \cup fn(\mathcal{B})$. With appeal to the induction hypothesis, we calculate:

$$\begin{aligned} P \models \mathcal{A} \triangleright \mathcal{B} &\iff \forall Q.Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B} \\ &\iff \forall Q.Q \models \mathcal{A} \Rightarrow (P \mid Q)\{m \leftarrow m'\} \models \mathcal{B} \\ &\iff \forall Q.Q \models \mathcal{A} \Rightarrow (P\{m \leftarrow m'\} \mid Q)\{m \leftarrow m'\}) \models \mathcal{B} \\ &\iff \forall Q.Q \models \mathcal{A} \Rightarrow P\{m \leftarrow m'\} \mid Q \models \mathcal{B} \\ &\iff P\{m \leftarrow m'\} \models \mathcal{A} \triangleright \mathcal{B} \end{aligned}$$

Case $n[\mathcal{A}]$**.** We have $m, m' \notin \{n\} \cup fn(\mathcal{A})$. With appeal to Lemma 2 and Lemma 6, and the induction hypothesis, we calculate:

$$\begin{aligned} P \models n[\mathcal{A}] &\iff \exists P'.P \equiv n[P'] \wedge P' \models \mathcal{A} \\ &\iff \exists P'.P \equiv n[P'] \wedge P'\{m \leftarrow m'\} \models \mathcal{A} \\ &\iff \exists P', P''.P \equiv n[P'] \wedge P'\{m \leftarrow m'\} \equiv P'' \wedge P'' \models \mathcal{A} \\ &\iff \exists P''.P\{m \leftarrow m'\} \equiv n[P''] \wedge P'' \models \mathcal{A} \\ &\iff P\{m \leftarrow m'\} \models n[\mathcal{A}] \end{aligned}$$

Case $\mathcal{A}@n$**.** We have $m, m' \notin \{n\} \cup fn(\mathcal{A})$. With appeal to the induction hypothesis, we calculate:

$$\begin{aligned} P \models \mathcal{A}@n &\iff n[P] \models \mathcal{A} \\ &\iff (n[P])\{m \leftarrow m'\} \models \mathcal{A} \\ &\iff n[P\{m \leftarrow m'\}] \models \mathcal{A} \\ &\iff P\{m \leftarrow m'\} \models \mathcal{A}@n \end{aligned}$$

□

This lemma does not hold for the logic extended with quantifiers: we have $m[] \mid n[] \models \exists x, y.(x[] \mid y[]) \wedge x \neq y$ but $m[] \mid m[] \not\models \exists x, y.(x[] \mid y[]) \wedge x \neq y$.

Proposition 1 (Bounding Names). *Suppose* $m \notin fn(\mathcal{A} \triangleright \mathcal{B})$*. Then:*

$$P \models \mathcal{A} \triangleright \mathcal{B} \iff (\forall Q \in \mathsf{Tree}_{fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}}.\ Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B})$$

Proof. The forwards direction is immediate. For the backwards direction, assume that $(\forall Q \in \mathsf{Tree}_{fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}}.\ Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B})$ and consider any tree Q such that $Q \models \mathcal{A}$. Suppose that $fn(P \mid Q) \subseteq fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m, n_1, \ldots, n_k\}$ where $\{n_1, \ldots, n_k\} \cap (fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}) = \emptyset$. Let $P' = P\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\}$ and $Q' = Q\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\}$. By repeated application of Lemma 8, we get that $Q \models \mathcal{A} \iff Q' \models \mathcal{A}$. Since $Q' \in \mathsf{Tree}_{fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}}$ and $Q' \models \mathcal{A}$, we obtain $P \mid Q' \models \mathcal{B}$ by assumption. Now, we have:

$$(P \mid Q')\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\} = P' \mid Q' = (P \mid Q)\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\}$$

Hence, by repeated application of Lemma 8, we get that $P \mid Q' \models \mathcal{B} \iff P' \mid Q' \models \mathcal{B} \iff P \mid Q \models \mathcal{B}$. Hence $P \mid Q \models \mathcal{B}$ follows. □

3.2 Bounding the Sizes to Consider

Definition 1 (Notation). *Write $a \cdot P$ for a copies of P in parallel: $P \mid \ldots \mid P$.*

Definition 2 (Size of Trees).
$|P|^{\mathsf{hw}} \triangleq (h, w)$ iff there are a_1, n_1, P_1, …, a_k, n_k, P_k, for some k, such that:

- $P \equiv a_1 \cdot n_1[P_1] \mid \ldots \mid a_k \cdot n_k[P_k]$
- $\forall i, j \in 1..k.\, n_i[P_i] \equiv n_j[P_j] \Rightarrow i = j$
- $(h_i, w_i) = |P_i|^{\mathsf{hw}}$ *for each* $i \in 1..k$
- *if* $k = 0$, $h = 0$; *otherwise* $h = 1 + max(h_1, \ldots, h_k)$
- *if* $k = 0$, $w = 0$; *otherwise* $h = max(a_1, \ldots, a_k, w_1, \ldots, w_k)$

When $|P|^{\mathsf{hw}} = (h, w)$, write $|P|^{\mathsf{h}}$ for h and $|P|^{\mathsf{w}}$ for w. Write $(h_1, w_1) \leq (h_2, w_2)$ for $(h_1 \leq h_2) \wedge (w_1 \leq w_2)$.

Intuitively $|P|^{\mathsf{h}}$ is the height of P, and $|P|^{\mathsf{w}}$ is the width, defined as the maximum *multiplicity* of the subtrees of P. The multiplicity is the number of structurally equivalent trees under the same edge. For example:

- $|\mathbf{0}|^{\mathsf{hw}} = (0, 0)$
- $|n[\mathbf{0}]|^{\mathsf{hw}} = (1, 1)$
- $|n[\mathbf{0}] \mid m[\mathbf{0}]|^{\mathsf{hw}} = (1, 1)$
- $|n[\mathbf{0}] \mid n[\mathbf{0}]|^{\mathsf{hw}} = (1, 2)$
- $|n[m[\mathbf{0}]]|^{\mathsf{hw}} = (2, 1)$
- $|n[n[\mathbf{0}]]|^{\mathsf{hw}} = (2, 1)$

Size of Logical Formulas

$\lvert\mathbf{F}\rvert^{\mathsf{h}}$	$\triangleq 0$	$\lvert\mathbf{F}\rvert^{\mathsf{w}}$	$\triangleq 0$
$\lvert\mathcal{A} \wedge \mathcal{B}\rvert^{\mathsf{h}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{h}}, \lvert\mathcal{B}\rvert^{\mathsf{h}})$	$\lvert\mathcal{A} \wedge \mathcal{B}\rvert^{\mathsf{w}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{w}}, \lvert\mathcal{B}\rvert^{\mathsf{w}})$
$\lvert\mathcal{A} \Rightarrow \mathcal{B}\rvert^{\mathsf{h}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{h}}, \lvert\mathcal{B}\rvert^{\mathsf{h}})$	$\lvert\mathcal{A} \Rightarrow \mathcal{B}\rvert^{\mathsf{w}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{w}}, \lvert\mathcal{B}\rvert^{\mathsf{w}})$
$\lvert\mathbf{0}\rvert^{\mathsf{h}}$	$\triangleq 1$	$\lvert\mathbf{0}\rvert^{\mathsf{w}}$	$\triangleq 1$
$\lvert\mathcal{A} \mid \mathcal{B}\rvert^{\mathsf{h}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{h}}, \lvert\mathcal{B}\rvert^{\mathsf{h}})$	$\lvert\mathcal{A} \mid \mathcal{B}\rvert^{\mathsf{w}}$	$\triangleq \lvert\mathcal{A}\rvert^{\mathsf{w}} + \lvert\mathcal{B}\rvert^{\mathsf{w}}$
$\lvert\mathcal{A} \triangleright \mathcal{B}\rvert^{\mathsf{h}}$	$\triangleq \lvert\mathcal{B}\rvert^{\mathsf{h}}$	$\lvert\mathcal{A} \triangleright \mathcal{B}\rvert^{\mathsf{w}}$	$\triangleq \lvert\mathcal{B}\rvert^{\mathsf{w}}$
$\lvert n[\mathcal{A}]\rvert^{\mathsf{h}}$	$\triangleq 1 + \lvert\mathcal{A}\rvert^{\mathsf{h}}$	$\lvert n[\mathcal{A}]\rvert^{\mathsf{w}}$	$\triangleq max(2, \lvert\mathcal{A}\rvert^{\mathsf{w}})$
$\lvert\mathcal{A}@n\rvert^{\mathsf{h}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{h}} - 1, 0)$	$\lvert\mathcal{A}@n\rvert^{\mathsf{w}}$	$\triangleq \lvert\mathcal{A}\rvert^{\mathsf{w}}$

Here are the sizes for the derived propositional connectives:

$\lvert\mathbf{T}\rvert^{\mathsf{h}}$	$\triangleq 0$	$\lvert\mathbf{T}\rvert^{\mathsf{w}}$	$\triangleq 0$
$\lvert\neg\mathcal{A}\rvert^{\mathsf{h}}$	$\triangleq \lvert\mathcal{A}\rvert^{\mathsf{h}}$	$\lvert\neg\mathcal{A}\rvert^{\mathsf{w}}$	$\triangleq \lvert\mathcal{A}\rvert^{\mathsf{w}}$
$\lvert\mathcal{A} \vee \mathcal{B}\rvert^{\mathsf{h}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{h}}, \lvert\mathcal{B}\rvert^{\mathsf{h}})$	$\lvert\mathcal{A} \vee \mathcal{B}\rvert^{\mathsf{w}}$	$\triangleq max(\lvert\mathcal{A}\rvert^{\mathsf{w}}, \lvert\mathcal{B}\rvert^{\mathsf{w}})$

We define a relation $\sim_{h,w}$ between trees, parameterized by the size (h, w). The main property of the relation is that if $P \sim_{h,w} Q$ then no formula with size (h, w) can distinguish between P and Q (Proposition 2).

Definition 3 (Relation $P \sim_{h,w} Q$).

$$\begin{aligned} P \sim_{0,w} Q \quad & \textit{always} \\ P \sim_{h+1,w} Q \iff & \forall i \in 1..w.\, \forall n, P_j \textit{ with } j \in 1..i. \\ & \textit{if } P \equiv n[P_1] \mid \cdots \mid n[P_i] \mid P_{i+1} \\ & \textit{then } Q \equiv n[Q_1] \mid \cdots \mid n[Q_i] \mid Q_{i+1} \\ & \textit{such that } P_j \sim_{h,w} Q_j \textit{ for } j \in 1..i \\ & \textit{and vice versa} \end{aligned}$$

Note that $\sim_{h,w}$ is an equivalence relation: reflexivity, symmetry, and transitivity are immediate consequences of the definition. Moreover, it is preserved by structural congruence:

Lemma 9. *If $P \sim_{h,w} Q$ and $Q \equiv R$ then $P \sim_{h,w} R$.*

Proof. By an easy induction on h. □

The following lemma shows that the relation $\sim_{h,w}$ is monotone in (h, w).

Lemma 10 (Monotonicity). *If $P \sim_{h,w} Q$ and $(h', w') \leq (h, w)$ then $P \sim_{h',w'} Q$.*

Proof. By induction on h. The case $h = 0$ is immediate.

For $h + 1$, suppose $P \sim_{h+1,w} Q$ and $(h', w') \leq (h + 1, w)$. If $h' = 0$ then clearly $P \sim_{h',w'} Q$. If $h' = h'' + 1$ for some h'', then consider any $i \in 1..w'$, n, P_j for $j \in 1..i$ such that

$$P \equiv n[P_1] \mid \cdots \mid n[P_i] \mid P_{i+1}$$

Since $w' \leq w$, then $i \in 1..w$, and from $P \sim_{h+1,w} Q$ we have

$$Q \equiv n[Q_1] \mid \cdots \mid n[Q_i] \mid Q_{i+1} \text{ such that } P_j \sim_{h,w} Q_j \text{ for } j \in 1..i$$

Since $(h'', w') \leq (h, w)$, by induction hypothesis we have $P_j \sim_{h'',w'} Q_j$ for $j \in 1..i$. This proves $P \sim_{h''+1,w'} Q$, that is, $P \sim_{h',w'} Q$. □

The following lemma shows that the relation $\sim_{h,w}$ is a congruence.

Lemma 11 (Congruence). *The following hold:*

(1) *If $P \sim_{h,w} Q$ then $n[P] \sim_{h+1,w} n[Q]$.*
(2) *If $P \sim_{h,w} P'$ and $Q \sim_{h,w} Q'$ then $P \mid Q \sim_{h,w} P' \mid Q'$.*

Proof. We prove both parts directly.

(1) Suppose $P \sim_{h,w} Q$. If $w = 0$ then the conclusion is immediate. Otherwise, consider any $i \in 1..w$, n, P_j for $j \in 1..i$ such that

$$n[P] \equiv n[P_1] \mid \cdots \mid n[P_i] \mid P_{i+1}$$

Then $i = 1$ and $P_1 \equiv P$ and $P_{i+1} \equiv \mathbf{0}$. We have $n[Q] \equiv n[Q] \mid \mathbf{0}$, and $P_1 \sim_{h,w} Q$ by Lemma 9. This proves $n[P] \sim_{h+1,w} n[Q]$.

(2) There are two cases. If $h = 0$ then the conclusion is immediate.
For $h+1$, suppose $P \sim_{h+1,w} P'$ and $Q \sim_{h+1,w} Q'$; then consider any $i \in 1..w$, n, R_j for $j \in 1..i$ such that

$$P \mid Q \equiv n[R_1] \mid \cdots \mid n[R_i] \mid R_{i+1}$$

Suppose without loss of generality that the R_j are ordered in a way that there exist $k \in 1..i, P_\dagger, Q_\dagger$ such that

$$P \equiv n[R_1] \mid \cdots \mid n[R_k] \mid P_\dagger \quad Q \equiv n[R_{k+1}] \mid \cdots \mid n[R_i] \mid Q_\dagger \quad R_{i+1} \equiv P_\dagger \mid Q_\dagger$$

Since $k \in 1..w$, from $P \sim_{h+1,w} P'$ we have

$$P' \equiv n[P'_1] \mid \cdots \mid n[P'_k] \mid P'_\dagger \text{ such that } R_j \sim_{h,w} P'_j \text{ for } j \in 1..k$$

Similarly, from $Q \sim_{h+1,w} Q'$ we have

$$Q' \equiv n[Q'_{k+1}] \mid \cdots \mid n[Q'_i] \mid Q'_\dagger \text{ such that } R_j \sim_{h,w} Q'_j \text{ for } j \in (k+1)..i$$

Hence, we have

$$P' \mid Q' \equiv n[P'_1] \mid \cdots \mid n[P'_k] \mid n[Q'_{k+1}] \mid \cdots \mid n[Q'_i] \mid P'_\dagger \mid Q'_\dagger$$

Since $R_j \sim_{h,w} P'_j$ for $j \in 1..k$ and $R_j \sim_{h,w} Q'_j$ for $j \in (k+1)..i$, this proves $P \mid Q \sim_{h+1,w} P' \mid Q'$. □

Lemma 12 (Inversion). *If $P' \mid P'' \sim_{h,w_1+w_2} Q$ then there exist Q', Q'' such that $Q \equiv Q' \mid Q''$ and $P' \sim_{h,w_1} Q'$ and $P'' \sim_{h,w_2} Q''$.*

Proof. There are two cases. If $h = 0$ then the conclusion is immediate.
For $h+1$, suppose $P' \mid P'' \sim_{h+1,w_1+w_2} Q$. Consider the following definition:

A tree P is in (h,w)-normal form if whenever $P \equiv n[P_1] \mid n[P_2] \mid P_3$ for some P_1, P_2, P_3, if $P_1 \sim_{h,w} P_2$ then $P_1 \equiv P_2$. Note that $P \sim_{h+1,w} n[P_1] \mid n[P_1] \mid P_3$, hence it is always possible to find a $P^\dagger$ such that $P^\dagger$ is in (h,w)-normal form and $P \sim_{h+1,w} P^\dagger$.

We can assume without loss of generality that P' and P'' are in (h,w)-normal form, and by Lemma 10 that Q is in (h,w)-normal form. Hence, there exist k, P_j, a'_j, a''_j, b_j for $j \in 1..k$ such that

$$\begin{gathered} P' \equiv a'_1 \cdot n_1[P_1] \mid \cdots \mid a'_k \cdot n_k[P_k] \\ P'' \equiv a''_1 \cdot n_1[P_1] \mid \cdots \mid a''_k \cdot n_k[P_k] \\ Q \equiv b_1 \cdot n_1[P_1] \mid \cdots \mid b_k \cdot n_k[P_k] \end{gathered}$$

where if $P_i \sim_{h,w} P_j$ and $n_i = n_j$ then $i = j$.

To split Q into two parts, we now specify how to split each b_i, for $i \in 1..k$, into b'_i and b''_i, such that:

$$\begin{gathered} b_i = b'_i + b''_i \\ a'_i \cdot n_i[P_i] \sim_{h+1,w_1} b'_i \cdot n_i[P_i] \\ a''_i \cdot n_i[P_i] \sim_{h+1,w_2} b''_i \cdot n_i[P_i] \end{gathered}$$

For each $i \in 1..k$, we choose b'_i and b''_i according to the following cases:

– Case $a'_i + a''_i < w_1 + w_2$. Then $P' \mid P'' \sim_{h+1,w} Q$ implies $b_i = a'_i + a''_i$, so we can choose $b'_i = a'_i$ and $b''_i = a''_i$.
– Case $a'_i + a''_i \geq w_1 + w_2$. Then $P' \mid P'' \sim_{h+1,w} Q$ implies $b_i \geq w_1 + w_2$. There are three subcases:
  • Subcase $a'_i \geq w_1$ and $a''_i \geq w_2$. Then we choose $b'_i = w_1$ and $b''_i = b_i - w_1$ (note that b''_i is saturated, that is, $b''_i \geq w_2$, since $b_i \geq w_1 + w_2$).
  • Subcase $a'_i < w_1$. We must have $a''_i \geq w_2$. Then we choose $b'_i = a'_i$ and $b''_i = b_i - a'_i$. So $b''_i \geq w_2$ since $b_i \geq w_1 + w_2$ and $b'_i < w_1$.
  • Subcase $a''_i < w_2$. This is symmetric to the previous case. We must have $a'_i \geq w_1$. We choose $b''_i = a''_i$ and $b'_i = b_i - a''_i$. So $b'_i \geq w_1$ since $b_i \geq w_1 + w_2$ and $b''_i < w_2$.

Now we define Q' and Q'' as follows:

$$Q' \equiv b'_1 \cdot n_1[P_1] \mid \cdots \mid b'_k \cdot n_k[P_k] \qquad Q'' \equiv b''_1 \cdot n_1[P_1] \mid \cdots \mid b''_k \cdot n_k[P_k]$$

We have $Q \equiv Q' \mid Q''$, and by repeated application of Lemma 11 we get $P' \sim_{h+1,w_1} Q'$ and $P'' \sim_{h+1,w_2} Q''$. □

Proposition 2. *If* $|\mathcal{A}|^{\mathsf{hw}} = (h, w)$ *and* $P \models \mathcal{A}$ *and* $P \sim_{h,w} Q$ *then* $Q \models \mathcal{A}$.

Proof. By induction on the structure of $\mathcal{A}$. We consider only some interesting cases.

Case 0. Suppose $P \models \mathbf{0}$ and $P \sim_{1,1} Q$. Then $P \equiv \mathbf{0}$. Since $P \sim_{1,1} Q$, if $Q \equiv n[Q_1] \mid Q_2$ for some n, Q_1, Q_2 then $P \equiv n[P_1] \mid P_2$ for some P_1, P_2. Hence $Q \equiv \mathbf{0}$; thus $Q \models \mathbf{0}$.

Case $\mathcal{A}_1 \mid \mathcal{A}_2$**.** Suppose $|\mathcal{A}_i|^{\mathsf{hw}} = (h_i, w_i)$ for $i = 1, 2$ and $P \models \mathcal{A}_1 \mid \mathcal{A}_2$. We have $|(\mathcal{A}_1 \mid \mathcal{A}_2)|^{\mathsf{hw}} = (max(h_1, h_2), w_1 + w_2)$ and there exist P_1, P_2 such that $P \equiv P_1 \mid P_2$ and $P_i \models \mathcal{A}_i$ for $i = 1, 2$. Then by Lemma 12 there exist Q_1, Q_2 such that $Q \equiv Q_1 \mid Q_2$ and $P_i \sim_{max(h_1,h_2),w_i} Q_i$ for $i = 1, 2$. Then $P_i \sim_{h_i,w_i} Q_i$ for $i = 1, 2$ by Lemma 10, hence $Q_i \models \mathcal{A}_i$ for $i = 1, 2$ by induction hypothesis. This proves $Q \models \mathcal{A}_1 \mid \mathcal{A}_2$.

Case $\mathcal{A} \triangleright \mathcal{B}$**.** Suppose $|\mathcal{B}|^{\mathsf{hw}} = (h, w)$ and $P \models \mathcal{A} \triangleright \mathcal{B}$. We have $|\mathcal{A} \triangleright \mathcal{B}|^{\mathsf{hw}} = (h, w)$ and $P \sim_{h,w} Q$. Consider any P_1 such that $P_1 \models \mathcal{A}$; then $P \mid P_1 \models \mathcal{B}$. Since $P \sim_{h,w} Q$ and $P_1 \sim_{h,w} P_1$ we have $P \mid P_1 \sim_{h,w} Q \mid P_1$ by Lemma 11. Hence $Q \mid P_1 \models \mathcal{B}$ by induction hypothesis. This proves $Q \models \mathcal{A} \triangleright \mathcal{B}$.

Case $n[\mathcal{A}]$**.** Suppose $|\mathcal{A}|^{\mathsf{hw}} = (h, w)$. We have $|n[\mathcal{A}]|^{\mathsf{hw}} = (h+1, max(w, 2))$ and $P \sim_{h+1,max(w,2)} Q$ and $P \models n[A]$. Then there exists P' such that $P \equiv n[P']$ and $P' \models \mathcal{A}$. ¿From $P \sim_{h+1,max(w,2)} Q$ we deduce that there exists Q' such that $Q \equiv n[Q']$ and $P' \sim_{h,max(w,2)} Q'$. Lemma 10 implies $P' \sim_{h,w} Q'$, and by induction hypothesis we have $Q' \models \mathcal{A}$. This proves $Q \models n[\mathcal{A}]$.

Case $\mathcal{A}@n$**.** Suppose $|\mathcal{A}|^{\mathsf{hw}} = (h, w)$. We have $|\mathcal{A}@n|^{\mathsf{hw}} = (max(h-1, 0), w)$ and $P \sim_{max(h-1,0),w} Q$. If $h > 0$ then we have $n[P] \sim_{h,w} n[Q]$ by Lemma 11. If $h = 0$ then $n[P] \sim_{h,w} n[Q]$ is immediate. With appeal to the induction hypothesis, we calculate:

$$\begin{aligned} P \models \mathcal{A}@n &\iff n[P] \models \mathcal{A} \\ &\iff n[Q] \models \mathcal{A} \\ &\iff Q \models \mathcal{A}@n \end{aligned}$$

□

The following lemma shows that each equivalence class determined by $\sim_{h,w}$ contains a tree of size bounded by (h, w).

Lemma 13 (Pruning). *For all $P \in \mathsf{Tree}_X, h, w$ there exists $P' \in \mathsf{Tree}_X$ such that $P \sim_{h,w} P'$ and $|P'|^{\mathsf{hw}} \leq (h, w)$.*

Proof. We describe how to construct P' by induction on h. For $h = 0$ define $P' \triangleq \mathbf{0}$.

For $h + 1$, suppose $P \equiv n_1[P_1] \mid \cdots \mid n_k[P_k]$, for some k and n_j, P_j with $j \in 1..k$. Let P'_j, for $j \in 1..k$, be the tree obtained by pruning P_j to size h, w. Define $Q \triangleq n_1[P'_1] \mid \cdots \mid n_k[P'_k]$. We can write Q in a canonical form with respect to $\equiv$, that is, there exist i and a_j, m_j, Q_j for $j \in 1..i$ such that $Q \equiv a_1 \cdot m_1[Q_1] \mid \cdots \mid a_i \cdot m_i[Q_i]$ and, for all $j, j' \in 1..i$, if $m_j[Q_j] \equiv m_{j'}[Q_{j'}]$ then $j = j'$. For each $j \in 1..i$, define $b_i \triangleq min(a_i, w)$. Then we can define $P' \triangleq b_1 \cdot m_1[Q_1] \mid \cdots \mid b_i \cdot m_i[Q_i]$. It is easy to see that $|P'|^{\mathsf{hw}} \leq (h + 1, w)$ and $P \sim_{h+1,w} P'$. □

Proposition 3 (Bounding Size). *For any tree P, set of names X and formulas $\mathcal{A}$ and $\mathcal{B}$, if $h = max(|\mathcal{A}|^{\mathsf{h}}, |\mathcal{B}|^{\mathsf{h}})$ and $w = max(|\mathcal{A}|^{\mathsf{w}}, |\mathcal{B}|^{\mathsf{w}})$ then*

$$(\forall Q \in \mathsf{Tree}_X.\, Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \iff (\forall Q \in \mathsf{Tree}_X.\, |Q|^{\mathsf{hw}} \leq (h, w) \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B})$$

Proof. The forwards direction is immediate. For the backwards direction, assume that the right hand side holds. Take any $Q \in \mathsf{Tree}_X$ such that $Q \models \mathcal{A}$. Then

$\exists Q'.\, Q \sim_{h,w} Q' \wedge \lvert Q'\rvert^{\mathsf{hw}} \leq (h, w)$	by Lemma 13
$Q \sim_{\lvert\mathcal{A}\rvert^{\mathsf{h}}, \lvert\mathcal{A}\rvert^{\mathsf{w}}} Q'$	by Lemma 10 since $\lvert\mathcal{A}\rvert^{\mathsf{hw}} \leq (h, w)$
$Q' \models \mathcal{A}$	by Proposition 2
$P \mid Q' \models \mathcal{B}$	by assumption
$P \mid Q \sim_{h,w} P \mid Q'$	by Lemma 11
$P \mid Q \sim_{\lvert\mathcal{B}\rvert^{\mathsf{h}}, \lvert\mathcal{B}\rvert^{\mathsf{w}}} P \mid Q'$	by Lemma 10 since $\lvert\mathcal{B}\rvert^{\mathsf{hw}} \leq (h, w)$
$P \mid Q \models \mathcal{B}$	by Proposition 2

□

Proposition 4 (Bounding Size and Names). *For any tree P and formulas $\mathcal{A}$ and $\mathcal{B}$, if $m \notin fn(\mathcal{A} \triangleright \mathcal{B})$ and $X = fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}$ and $h = max(|\mathcal{A}|^{\mathsf{h}}, |\mathcal{B}|^{\mathsf{h}})$ and $w = max(|\mathcal{A}|^{\mathsf{w}}, |\mathcal{B}|^{\mathsf{w}})$, then:*

$$P \models \mathcal{A} \triangleright \mathcal{B} \iff (\forall Q \in \mathsf{Tree}_X.\, |Q|^{\mathsf{hw}} \leq (h, w) \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B})$$

Proof. We have:

$$\begin{aligned} P \models \mathcal{A} \triangleright \mathcal{B} &\iff (\forall Q \in \mathsf{Tree}_X.\, Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \\ &\iff (\forall Q \in \mathsf{Tree}_X.\, |Q|^{\mathsf{hw}} \leq (h, w) \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \end{aligned}$$

Proposition 1 justifies the first step, Proposition 3 the second. □

So, to check satisfaction of $\mathcal{A} \triangleright \mathcal{B}$, we need only consider trees whose free names are drawn from $fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}$, and whose size is no more than $max(|\mathcal{A}|^{\mathsf{hw}}, |\mathcal{A}|^{\mathsf{hw}})$. We show in the next section, that the number of such trees, modulo structural equivalence, is finite. Hence, we obtain an algorithm for satisfaction of $\mathcal{A} \triangleright \mathcal{B}$.

3.3 Enumerating Equivalence Classes

In this section we present an explicit characterization of the equivalence classes on trees, modulo structural equivalence, determined by $\sim_{h,w}$.

Definition 4 (Notation). *Consider the following notation, where c ranges over sets of trees modulo structural congruence:*

$$\begin{array}{ll}
\langle P \rangle_{\equiv} & \triangleq \{P' \mid P \equiv P'\} \\
\langle P \rangle_{h,w} & \triangleq \{P' \mid P \sim_{h,w} P'\} \\
c_1 + c_2 & \triangleq c_1 \cup c_2 \\
n[c] & \triangleq \{\langle n[P] \rangle_{\equiv} \mid \langle P \rangle_{\equiv} \in c\} \\
c^{\leq n} & \triangleq \{\langle a_1 \cdot P_1 \mid \cdots \mid a_k \cdot P_k \rangle_{\equiv} \mid 0 \leq a_i \leq n \text{ for } i \in 1..k\} \text{ when } c = \{\langle P_1 \rangle_{\equiv}, \ldots, \langle P_k \rangle_{\equiv}\}
\end{array}$$

We can now give a direct definition of the set of equivalence classes $EQ^X_{h,w}$ determined by $\sim_{h,w}$, given a set of names X.

Definition 5. *For $X = \{n_1, \ldots, n_k\}$, define $EQ^X_{h,w}$ as follows:*

$$\begin{array}{ll}
EQ^X_{0,w} & \triangleq \{\langle 0 \rangle_{\equiv}\} \\
EQ^X_{h+1,w} & \triangleq (n_1[EQ^X_{h,w}] + \cdots + n_k[EQ^X_{h,w}])^{\leq w}
\end{array}$$

Lemma 14. *If $|P|^{\mathsf{hw}} \leq (h, w)$ and $|P'|^{\mathsf{hw}} \leq (h, w)$, then*

(1) $P \in \mathsf{Tree}_X$ *implies* $\langle P \rangle_{\equiv} \in EQ^X_{h,w}$.
(2) $P \equiv P' \iff P \sim_{h,w} P'$.

Proof. Part (1) is a simple induction on h.

For Part (2), the interesting direction is $\Leftarrow$. We proceed by induction on h. If $h = 0$ then $|P|^{\mathsf{h}} = |Q|^{\mathsf{h}} = 0$, hence $P \equiv Q \equiv \mathbf{0}$.

For the case $h + 1$, suppose $|P|^{\mathsf{hw}} \leq (h + 1, w)$ and $|P'|^{\mathsf{hw}} \leq (h + 1, w)$ and $P \sim_{h+1,w} P'$. Write P and P' in canonical form with respect to $\equiv$, that is, there exist k and a_j, a'_j, n_j, P_j for $j \in 1..k$ such that

$$P \equiv a_1 \cdot n_1[P_1] \mid \cdots \mid a_k \cdot n_k[P_k] \qquad P' \equiv a'_1 \cdot n_1[P_1] \mid \cdots \mid a'_k \cdot n_k[P_k]$$

where, for all $i, j \in 1..k$, if $n_i[P_i] \equiv n_j[P_j]$ then $i = j$. Since $|P|^{\mathsf{hw}} \leq (h + 1, w)$ and $|P'|^{\mathsf{hw}} \leq (h + 1, w)$ we have $a_j \leq w$ and $a'_j \leq w$ for each $j \in 1..k$. For each $i \in 1..k$ we show $a_i \leq a'_i$:

There exists $P_\dagger$ such that $P \equiv a_i \cdot n_i[P_i] \mid P^\dagger$. Then by definition of $P \sim_{h+1,w} P'$ there exist $P'_1, \ldots, P'_{a_i}, P'_\dagger$ such that $P' \equiv n_i[P'_1] \mid \cdots \mid n_i[P'_{a_i}] \mid P'_\dagger$ and $P_i \sim_{h,w} P'_j$ for $j \in 1..a_i$. By induction hypothesis we have $P_i \equiv P'_j$ for each $j \in 1..a_i$, hence $P' \equiv a_i \cdot n_i[P_i] \mid P'_\dagger$. This proves $a_i \leq a'_i$.

With a symmetric argument we can show $a'_i \leq a_i$ for each $i \in 1..k$. This proves $P \equiv P'$. □

The following lemma shows that $EQ^X_{h,w}$ contains exactly the trees (modulo $\equiv$) of size at most (h, w) with free names drawn from X.

Lemma 15. $\langle P \rangle_\equiv \in EQ^X_{h,w} \iff P \in \mathsf{Tree}_X \wedge |P|^{\mathsf{hw}} \leq (h, w)$.

Proof. By construction, if $\langle P \rangle_\equiv \in EQ^X_{h,w}$ then $P \in \mathsf{Tree}_X$ and $|P|^{\mathsf{hw}} \leq (h, w)$. The converse follows from Lemma 14. □

The following proposition shows that $EQ^X_{h,w}$ is an enumeration of the representants of the equivalence classes in $\mathsf{Tree}_X / \sim_{h,w}$.

Proposition 5. *The function* $f : \mathsf{Tree}_X \to \mathsf{Tree}_X / \sim_{h,w}$ *sending* P *to* $\langle P \rangle_{h,w}$ *extends to a bijection* $f' : EQ^X_{h,w} \to \mathsf{Tree}_X / \sim_{h,w}$.

Proof. Let f' be the function sending $\langle P \rangle_\equiv$ to $\langle P \rangle_{h,w}$. Clearly f' is well defined since $P \equiv P'$ implies $P \sim_{h,w} P'$.

To show that f' is surjective, take any $\langle P \rangle_{h,w} \in \mathsf{Tree}_X / \sim_{h,w}$. By lemma 13 there exists $P' \in \mathsf{Tree}_X$ such that $P \sim_{h,w} P'$ and $|P'|^{\mathsf{hw}} \leq (h, w)$. So $\langle P' \rangle_{h,w} = \langle P \rangle_{h,w}$ and $\langle P' \rangle_\equiv \in EQ^X_{h,w}$ by lemma 14.

To show that f' is injective, consider any $P, Q \in \mathsf{Tree}_X$ such that $\langle P \rangle_\equiv, \langle Q \rangle_\equiv \in EQ^X_{h,w}$ and $\langle P \rangle_{h,w} = \langle Q \rangle_{h,w}$. Then $|P|^{\mathsf{hw}} \leq (h, w)$ and $|Q|^{\mathsf{hw}} \leq (h, w)$ by Lemma 15, hence $P \equiv Q$ by Lemma 14. This proves $\langle P \rangle_\equiv = \langle Q \rangle_\equiv$. □

Theorem 1 (Finite Bound). *Consider any formulas* $\mathcal{A}$ *and* $\mathcal{B}$*. Let* $EQ^X_{h,w} = \{\langle Q_1 \rangle_\equiv, \ldots, \langle Q_n \rangle_\equiv\}$*, where* $h = max(|\mathcal{A}|^{\mathsf{h}}, |\mathcal{B}|^{\mathsf{h}})$ *and* $w = max(|\mathcal{A}|^{\mathsf{w}}, |\mathcal{B}|^{\mathsf{w}})$ *and* $X = fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}$ *for some* $m \notin fn(\mathcal{A} \triangleright \mathcal{B})$*.*

Then, for any tree P*:*

$$P \models \mathcal{A} \triangleright \mathcal{B} \iff (\forall i \in 1..n.\, Q_i \models \mathcal{A} \Rightarrow P \mid Q_i \models \mathcal{B})$$

Proof. Using Proposition 4, Lemma 15, and Lemma 2:

$$\begin{aligned}
P \models \mathcal{A} \triangleright \mathcal{B} &\iff (\forall Q \in \mathsf{Tree}_X.\, |Q|^{\mathsf{hw}} \leq (h, w) \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \\
&\iff (\forall Q.\, \langle Q \rangle_\equiv \in EQ^X_{h,w} \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \\
&\iff (\forall Q.\, (\exists i \in 1..n.\, Q \equiv Q_i) \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \\
&\iff (\forall i \in 1..n.\, \forall Q.\, Q \equiv Q_i \wedge Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}) \\
&\iff (\forall i \in 1..n.\, Q_i \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B})
\end{aligned}$$

□

Given this result, we can now show that each of the three quantifications in the definition of satisfaction can be reduced to a finite problem.

Finite Test Sets: $T(P)$, $T(\mathcal{A} \triangleright \mathcal{B})$, and $T(n, P)$

$T(P)$ is the finite non-empty set $\{\langle Q, R\rangle \mid P \equiv Q \mid R\}/(\equiv \times \equiv)$.
$T(\mathcal{A} \triangleright \mathcal{B})$ is the finite non-empty set $EQ^X_{h,w}$, where $h = max(|\mathcal{A}|^{\mathsf{h}}, |\mathcal{B}|^{\mathsf{h}})$ and $w = max(|\mathcal{A}|^{\mathsf{w}}, |\mathcal{B}|^{\mathsf{w}})$ and $X = fn(\mathcal{A} \triangleright \mathcal{B}) \cup \{m\}$ for some $m \notin fn(\mathcal{A} \triangleright \mathcal{B})$.
$T(n, P)$ is the finite, possibly empty, set $\{Q \mid P \equiv n[Q]\}/\equiv$.

Lemma 16.

(1) *For any P, $P \models \mathcal{A}' \mid \mathcal{A}'' \iff \exists \langle P', P''\rangle \in T(P).P' \models \mathcal{A}' \wedge P'' \models \mathcal{A}''$.*
(2) *For any $\mathcal{A}$, $\mathcal{B}$, $P \models \mathcal{A} \triangleright \mathcal{B} \iff \forall Q \in T(\mathcal{A} \triangleright \mathcal{B}).Q \models \mathcal{A} \Rightarrow Q \mid P \models \mathcal{B}$.*
(3) *For any P, $P \models n[\mathcal{A}'] \iff \exists P' \in T(n, P).P' \models \mathcal{A}'$.*

Proof. Part (2) follows at once from Theorem 1. The other parts follow easily, as in earlier work [CG00]. □

Theorem 2. *Satisfaction and validity are interderivable and decidable.*

Proof. As noted in Section 2, Lemmas 4 and 5 establish the equivalence of satisfaction and validity. An algorithm for satisfaction follows from the rules of its definition in Section 2, together with the facts in Lemma 16. □

Validity is defined in terms of an infinite quantification over trees. We end with a corollary of Lemma 4 and Theorem 1, which reduces validity to a finite quantification over a computable sequence of trees. Hence, we obtain an explicit algorithm for validity.

Corollary 1. *Consider any formula $\mathcal{A}$. Suppose $EQ^X_{h,w} = \{\langle P_1\rangle_{\equiv}, \ldots, \langle P_n\rangle_{\equiv}\}$, where $(h, w) = |\mathcal{A}|^{\mathsf{hw}}$ and $X = fn(\mathcal{A}) \cup \{m\}$ for some $m \notin fn(\mathcal{A})$. Then*

$$\mathbf{vld}(\mathcal{A}) \iff (\forall i \in 1..n.\ P_i \models \mathcal{A})$$

It is straightforward to implement the algorithms for satisfaction and validity suggested above. A precise complexity analysis is currently missing, but certainly there can be a state explosion on quite small formulas. Still, the algorithm terminates in a reasonable time on many formulas. Here is a selection of formulas found to be valid by our implementation.

- $(\mathbf{0} \vee p[\mathbf{0}]) \mid \neg(p[\mathbf{0}])$
- $q[\neg\mathbf{0}] \triangleright \neg(\mathbf{0})$
- $\neg((q[q[\mathbf{0}]] \mid q[\mathbf{0}])@q)$
- $(T \triangleright \neg((q[\mathbf{0}] \vee \mathbf{T}) \triangleright \mathbf{0}))@q$
- $((\mathbf{0} \vee p[\mathbf{0}])@p)@p@p$
- $(\neg(p[\mathbf{T}]) \vee \neg(q[\mathbf{T}]))@q$
- $p[\mathbf{T}] \triangleright (p[\mathbf{T}] \mid \mathbf{T})$
- $\neg(p[\mathbf{T}] \triangleright \mathbf{0})$
- $\neg(\mathbf{T} \mid (\mathbf{T} \triangleright q[\mathbf{0}])@q)$
- $(\mathbf{T} \mid (\neg(\mathbf{0}) \vee \mathbf{0})) \mid \mathbf{T}$
- $(\mathbf{T} \mid q[\mathbf{T}])@q \vee \mathbf{0}$

4 Deciding Validity by Deduction

We present a sequent calculus for our spatial logic, following the pattern of Caires and Cardelli [CC02]. We show the calculus to be sound and complete with respect to an interpretation in terms of the satisfaction relation, and present a complete proof procedure. Hence, we obtain an algorithm for deciding validity by deduction in the sequent calculus.

4.1 A Sequent Calculus

A *context*, Γ or Δ, is a finite multiset of entries of the form $P : \mathcal{A}$ where P is a tree and $\mathcal{A}$ is a formula. A *sequent* is a judgment $\Gamma \vdash \Delta$ where Γ and Δ are contexts. The following table states the rules for deriving sequents. The rules depend on the finite test sets $T(P)$, $T(\mathcal{A} \triangleright \mathcal{B})$, and $T(n, P)$ introduced in Section 3. All that matters for the purpose of this section is that these sets are computable and that they satisfy the properties stated in Lemma 16. Hence, this is a finitary proof system; note the form of the rules ($|$ L), ($\triangleright$ R), and ($n[]$ L).

Rules of the Sequent Calculus:

$$\text{(Id)}\ \frac{P \equiv Q}{\Gamma, P : \mathcal{A} \vdash Q : \mathcal{A}, \Delta} \qquad \text{(Cut)}\ \frac{\Gamma \vdash P : \mathcal{A}, \Delta \quad \Gamma, P : \mathcal{A} \vdash \Delta}{\Gamma \vdash \Delta}$$

$$\text{(C L)}\ \frac{\Gamma, P : \mathcal{A}, P : \mathcal{A} \vdash \Delta}{\Gamma, P : \mathcal{A} \vdash \Delta} \qquad \text{(C R)}\ \frac{\Gamma \vdash P : \mathcal{A}, P : \mathcal{A}, \Delta}{\Gamma \vdash P : \mathcal{A}, \Delta}$$

$$(\mathbf{F}\ \text{L})\ \frac{}{\Gamma, P : \mathbf{F} \vdash \Delta} \qquad (\mathbf{F}\ \text{R})\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash P : \mathbf{F}, \Delta}$$

$$(\wedge\ \text{L})\ \frac{\Gamma, P : \mathcal{A}, P : \mathcal{B} \vdash \Delta}{\Gamma, P : \mathcal{A} \wedge \mathcal{B} \vdash \Delta} \qquad (\wedge\ \text{R})\ \frac{\Gamma \vdash P : \mathcal{A}, \Delta \quad \Gamma \vdash P : \mathcal{B}, \Delta}{\Gamma \vdash P : \mathcal{A} \wedge \mathcal{B}, \Delta}$$

$$(\Rightarrow\ \text{L})\ \frac{\Gamma \vdash P : \mathcal{A}, \Delta \quad \Gamma, P : \mathcal{B} \vdash \Delta}{\Gamma, P : \mathcal{A} \Rightarrow \mathcal{B} \vdash \Delta} \qquad (\Rightarrow\ \text{R})\ \frac{\Gamma, P : \mathcal{A} \vdash P : \mathcal{B}, \Delta}{\Gamma \vdash P : \mathcal{A} \Rightarrow \mathcal{B}, \Delta}$$

$$(\mathbf{0}\ \text{L})\ \frac{P \not\equiv \mathbf{0}}{\Gamma, P : \mathbf{0} \vdash \Delta} \qquad (\mathbf{0}\ \text{R})\ \frac{P \equiv \mathbf{0}}{\Gamma \vdash P : \mathbf{0}, \Delta}$$

$$\frac{\forall \langle Q, R\rangle \in T(P).\ \Gamma, Q : \mathcal{A}, R : \mathcal{B} \vdash \Delta}{\Gamma, P : \mathcal{A} \mid \mathcal{B} \vdash \Delta}\ (\mid \text{L}) \qquad \frac{\Gamma \vdash Q : \mathcal{A}, \Delta \quad \Gamma \vdash R : \mathcal{B}, \Delta \quad P \equiv Q \mid R}{\Gamma \vdash P : \mathcal{A} \mid \mathcal{B}, \Delta}\ (\mid \text{R})$$

$$\frac{\Gamma \vdash Q : \mathcal{A}, \Delta \quad \Gamma, Q \mid P : \mathcal{B} \vdash \Delta}{\Gamma, P : \mathcal{A} \triangleright \mathcal{B} \vdash \Delta}\ (\triangleright \text{L}) \qquad \frac{\forall Q \in T(\mathcal{A} \triangleright \mathcal{B}).\ \Gamma, Q : \mathcal{A} \vdash Q | P : \mathcal{B}, \Delta}{\Gamma \vdash P : \mathcal{A} \triangleright \mathcal{B}, \Delta}\ (\triangleright \text{R})$$

$$\frac{\forall Q \in T(n, P).\ \Gamma, Q : \mathcal{A} \vdash \Delta}{\Gamma, P : n[\mathcal{A}] \vdash \Delta}\ (n[] \text{ L}) \qquad \frac{\Gamma \vdash Q : \mathcal{A}, \Delta \quad P \equiv n[Q]}{\Gamma \vdash P : n[\mathcal{A}], \Delta}\ (n[] \text{ R})$$

$$\frac{\Gamma, n[P] : \mathcal{A} \vdash \Delta}{\Gamma, P : \mathcal{A}@n \vdash \Delta}\ (@n \text{ L}) \qquad \frac{\Gamma \vdash n[P] : \mathcal{A}, \Delta}{\Gamma \vdash P : \mathcal{A}@n, \Delta}\ (@n \text{ R})$$

The variables Q, R in ($\mid$ L) and the variable Q in ($\triangleright$ R) cannot occur free (in a formalistic reading) in Γ, P, Δ. Compare the side conditions on these rules in Caires and Cardelli [CC02]. Here, these are meta-level variables ranging over terms, so there is no need for such side conditions. Note that ($n[]$ L) applies also when $T(n, P)$ is empty (something that never happens for ($\mid$ L)), so we can conclude, for example, $\Gamma, \mathbf{0} : n[\mathcal{A}] \vdash \Delta$. The fact that $T(n, P)$ may be empty explains also the irregular form of Lemma 18($n[]$ R).

Lemma 17 (Weakening). *If $\Gamma \vdash \Delta$ is derivable, then $\Gamma, P : \mathcal{A} \vdash \Delta$ and $\Gamma \vdash P : \mathcal{A}, \Delta$ are derivable. Moreover, if there is a derivation of $\Gamma \vdash \Delta$ free of (Id), (Cut), (C L), (C R), then there are derivations of $\Gamma, P : \mathcal{A} \vdash \Delta$ and $\Gamma \vdash P : \mathcal{A}, \Delta$ free of (Id), (Cut), (C L), (C R).*

Proof. By induction on the derivation of $\Gamma \vdash \Delta$. The second part of the statement comes from inspection of the cases different from (Id), (Cut), (C L), (C R). □

4.2 Soundness and Completeness

We make a conventional interpretation of sequents:

$$^{\wedge}[\![P_1 : \mathcal{A}_1, ..., P_n : \mathcal{A}_n]\!] \triangleq P_1 \models \mathcal{A}_1 \wedge \ldots \wedge P_n \models \mathcal{A}_n$$
$$^{\vee}[\![Q_1 : \mathcal{B}_1, ..., Q_m : \mathcal{B}_m]\!] \triangleq Q_1 \models \mathcal{B}_1 \vee \ldots \vee Q_m \models \mathcal{B}_m$$
$$[\![\Gamma \vdash \Delta]\!] \triangleq {}^{\wedge}[\![\Gamma]\!] \Rightarrow {}^{\vee}[\![\Delta]\!]$$

Lemma 18 (Validity of Antecedents).

(F L) $[\![\Gamma, P : \mathbf{F} \vdash \Delta]\!]$
(F R) $[\![\Gamma \vdash P : \mathbf{F}, \Delta]\!]$ *iff* $[\![\Gamma \vdash \Delta]\!]$
($\wedge$ L) $[\![\Gamma, P : \mathcal{A}' \wedge \mathcal{A}'' \vdash \Delta]\!]$ *iff* $[\![\Gamma, P : \mathcal{A}', P : \mathcal{A}'' \vdash \Delta]\!]$
($\wedge$ R) $[\![\Gamma \vdash P : \mathcal{A}' \wedge \mathcal{A}'', \Delta]\!]$ *iff* $[\![\Gamma \vdash P : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma \vdash P : \mathcal{A}'', \Delta]\!]$
($\vee$ L) $[\![\Gamma, P : \mathcal{A}' \vee \mathcal{A}'' \vdash \Delta]\!]$ *iff* $[\![\Gamma \vdash P : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma, P : \mathcal{A}'' \vdash \Delta]\!]$
($\vee$ R) $[\![\Gamma \vdash P : \mathcal{A}' \vee \mathcal{A}'', \Delta]\!]$ *iff* $[\![\Gamma, P : \mathcal{A}' \vdash P : \mathcal{A}'', \Delta]\!]$
(0 L) $[\![\Gamma, P : \mathbf{0} \vdash \Delta]\!]$ *iff* $P \equiv \mathbf{0} \Rightarrow [\![\Gamma \vdash \Delta]\!]$
(0 R) $[\![\Gamma \vdash P : \mathbf{0}, \Delta]\!]$ *iff* $P \not\equiv \mathbf{0} \Rightarrow [\![\Gamma \vdash \Delta]\!]$
(| L) $[\![\Gamma, P : \mathcal{A}' \mid \mathcal{A}'' \vdash \Delta]\!]$ *iff* $\forall P', P''.P \equiv P' \mid P'' \Rightarrow [\![\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta]\!]$
(| R) $[\![\Gamma \vdash P : \mathcal{A}' \mid \mathcal{A}'', \Delta]\!]$ *iff* $\exists P', P''.P \equiv P' \mid P'' \wedge [\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma \vdash P'' : \mathcal{A}'', \Delta]\!]$
($\triangleright$ L) $[\![\Gamma, P : \mathcal{A}' \triangleright \mathcal{A}'' \vdash \Delta]\!]$ *iff* $\exists P'.[\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma, P' \mid P : \mathcal{A}'' \vdash \Delta]\!]$
($\triangleright$ R) $[\![\Gamma \vdash P : \mathcal{A}' \triangleright \mathcal{A}'', \Delta]\!]$ *iff* $\forall P'.[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$
($n[]$ L) $[\![\Gamma, P : n[\mathcal{A}'] \vdash \Delta]\!]$ *iff* $\forall P'.P \equiv n[P'] \Rightarrow [\![\Gamma, P' : \mathcal{A}' \vdash \Delta]\!]$
($n[]$ R) $[\![\Gamma \vdash P : n[\mathcal{A}'], \Delta]\!]$ *iff* $(\forall P'.P \not\equiv n[P'] \wedge [\![\Gamma \vdash \Delta]\!]) \vee (\exists P'.P \equiv n[P'] \wedge [\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!])$
($@n$ L) $[\![\Gamma, P : \mathcal{A}'@n \vdash \Delta]\!]$ *iff* $[\![\Gamma, n[P] : \mathcal{A}' \vdash \Delta]\!]$
($@n$ R) $[\![\Gamma \vdash P : \mathcal{A}'@n, \Delta]\!]$ *iff* $[\![\Gamma \vdash n[P] : \mathcal{A}', \Delta]\!]$

Proof. By detailed, but straightforward, calculations. □

Lemma 19 (Finite Test Sets).

(1) *For any P there is a finite set $T(P)$ such that:*
$\forall P', P''.P \equiv P' \mid P'' \Rightarrow [\![\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta]\!]$
iff $\forall \langle P', P'' \rangle \in T(P).[\![\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta]\!]$.
(2) *For any $\mathcal{A}'$, $\mathcal{A}''$, there is a finite set $T(\mathcal{A}' \triangleright \mathcal{A}'')$ such that:*
$\forall P'.[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$
iff $\forall P' \in T(\mathcal{A}' \triangleright \mathcal{A}'').[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$.
(3) *For any P there is a finite set $T(n, P)$ such that:*
$\forall P'.P \equiv n[P'] \Rightarrow [\![\Gamma, P' : \mathcal{A}' \vdash \Delta]\!]$
iff $\forall P' \in T(n, P).[\![\Gamma, P' : \mathcal{A}' \vdash \Delta]\!]$.

Proof. By expanding definitions, and appeal to Lemma 16. □

Theorem 3 (Soundness). *If $\Gamma \vdash \Delta$ is derivable, then $[\![\Gamma \vdash \Delta]\!]$.*

Proof. By induction on the derivation of $\Gamma \vdash \Delta$. □

Theorem 4 (Completeness). *If $[\![\Gamma \vdash \Delta]\!]$, then $\Gamma \vdash \Delta$ has a derivation. Moreover, it has a derivation that does not use (Id), (Cut), (C L), (C R).*

Proof. By induction on the sum of the sizes of all the formulas in $\Gamma \vdash \Delta$. The interesting cases are (| L), ($n[]$ L) and, particularly, ($\triangleright$ R), relying on Lemma 19. These are the only cases we show.

Subcase $[\![\Gamma, P : \mathcal{A}' \mid \mathcal{A}'' \vdash \Delta]\!]$. By Lemma 18(| L) we have $\forall P', P''.P \equiv P' \mid P'' \Rightarrow [\![\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta]\!]$. By Lemma 19(1) there is a finite set $T(P)$ such that $\forall \langle P', P'' \rangle \in T(P).[\![\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta]\!]$. By IndHyp, $\forall \langle P', P'' \rangle \in T(P).\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta$ has a derivation. Hence by (| L) we can construct a (finite) derivation for $\Gamma, P : \mathcal{A}' \mid \mathcal{A}'' \vdash \Delta$.

Subcase $[\![\Gamma, P : n[\mathcal{A}'] \vdash \Delta]\!]$**.** By Lemma 18($n[]$ L) we have $\forall P'.P \equiv n[P'] \Rightarrow [\![\Gamma, P' : \mathcal{A}' \vdash \Delta]\!]$. By Lemma 19(3) there is a finite set $T(n, P)$ such that $\forall P' \in T(n, P).[\![\Gamma, P' : \mathcal{A}' \vdash \Delta]\!]$. By IndHyp, $\forall P' \in T(n, P).\Gamma, P' : \mathcal{A}' \vdash \Delta$ has a derivation. Hence by ($n[]$ L) we can construct a (finite) derivation for $\Gamma, P : n[\mathcal{A}'] \vdash \Delta$.

Subcase $[\![\Gamma \vdash P : \mathcal{A}' \triangleright \mathcal{A}'', \Delta]\!]$**.** By Lemma 18($\triangleright$ R) we have $\forall P'.[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$. By Lemma 19(2) there is a finite set $T(\mathcal{A}' \triangleright \mathcal{A}'')$ such that $\forall P' \in T(\mathcal{A}' \triangleright \mathcal{A}'').[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$. By IndHyp, $\forall P' \in T(\mathcal{A}' \triangleright \mathcal{A}'').\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta$ has a derivation. Hence by ($\triangleright$ R) we can construct a (finite) derivation for $\Gamma \vdash P : \mathcal{A}' \triangleright \mathcal{A}'', \Delta$.

For the second part of the statement, it is sufficient to note that the rules (Id), (Cut), (C L), (C R) are never used in the proof to construct the derivation, and that the cases (**0** L), (**0** R), ($n[]$ R) use Lemma 17 applied to a derivation that, inductively, does not contain (Id), (Cut), (C L), (C R). □

Proposition 6 (Id, Cut and Contraction Elimination). *If there is a derivation of $\Gamma \vdash \Delta$, then there is a derivation of $\Gamma \vdash \Delta$ that does not use (Id), (Cut), (C L), (C R).*

Proof. If $\Gamma \vdash \Delta$ is derivable in the full system, then $[\![\Gamma \vdash \Delta]\!]$ by Theorem 3 (Soundness). Then, by Theorem 4 (Completeness), $\Gamma \vdash \Delta$ has a derivation that does not use (Id), (Cut), (C L), (C R). □

Proposition 7 (Decidability). *It is decidable whether $\Gamma \vdash \Delta$ is derivable.*

Proof. Suppose that $\Gamma = P_1 : \mathcal{A}_1, ..., P_n : \mathcal{A}_n$ and $\Delta = Q_1 : \mathcal{B}_1, ..., Q_m : \mathcal{B}_m$. By Theorems 3 (Soundness) and 4 (Completeness), $P_1 : \mathcal{A}_1, ..., P_n : \mathcal{A}_n \vdash Q_1 : \mathcal{B}_1, ..., Q_m : \mathcal{B}_m$ is derivable if and only if ${}^{\wedge}[\![P_1 : \mathcal{A}_1, \ldots, P_n : \mathcal{A}_n]\!] \Rightarrow {}^{\vee}[\![Q_1 : \mathcal{B}_1, \ldots, Q_m : \mathcal{B}_m]\!]$. By Theorem 2 we know that $P \models \mathcal{A}$ is decidable. Therefore, we just need to test that either there is an i with $P_i \not\models \mathcal{A}_i$, or there is a j with $Q_j \models \mathcal{B}_j$. □

4.3 A Complete Proof Procedure

The following theorem essentially implies Completeness, and uses Lemma 18 in a similar way, but is not quite as clean as Completeness, since it talks about an algorithm. Moreover, the cases for ($\triangleright$ L), (| R) and ($n[]$ R) are harder than in Completeness.

On the other hand, the proposition is interesting because it shows that there is a complete proof procedure that actually builds a derivation, unlike the one in Proposition 7.

Lemma 20 (More on Finite Test Sets).

(1) *For any P there is a finite set $T(P)$ such that:*
$\exists P', P''.P \equiv P' \mid P'' \wedge [\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma \vdash P'' : \mathcal{A}'', \Delta]\!]$
iff $\exists \langle P', P'' \rangle \in T(P).[\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma \vdash P'' : \mathcal{A}'', \Delta]\!]$.

(2) *For any* $\mathcal{A}'$, $\mathcal{A}''$, *there is a finite set* $T(\mathcal{A}' \triangleright \mathcal{A}'')$ *such that:* $\exists P'.[\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma, P' \mid P : \mathcal{A}'' \vdash \Delta]\!]$
iff $\exists P' \in T(\mathcal{A}' \triangleright \mathcal{A}'').[\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma, P' \mid P : \mathcal{A}'' \vdash \Delta]\!]$.
(3) *For any* P *there is a finite set* $T(n, P)$ *such that:*
$\exists P'.P \equiv n[P'] \wedge [\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!]$
iff $\exists P' \in T(n, P).[\![\Gamma \vdash P' : \mathcal{A}', \Delta]\!]$.

Proof. By expanding definitions, and appeal to Lemma 16. □

Theorem 5 (Complete Proof Procedure). *For any* $\Gamma \vdash \Delta$ *there is a procedure such that: if* $\neg[\![\Gamma \vdash \Delta]\!]$, *then the procedure terminates with failure; if* $[\![\Gamma \vdash \Delta]\!]$, *then the procedure terminates with a derivation for* $\Gamma \vdash \Delta$.

Proof. We describe the procedure, but omit the proof of correctness, which, in addition to the properties used in the proof of Theorem 4, uses also Lemma 20. The procedure picks nondeterministically any formula in the sequent to operate on. It terminates because at every recursive call it either reduces the total size, *size*, of the formulas in the sequent, or stops with success or failure.

Case *size* $= 0$, that is, the empty sequent $- \vdash -$.
The procedure terminates with failure.
Case *size* ≥ 1, left rules.
Subcase $\Gamma, P : \mathbf{F} \vdash \Delta$.
The procedure succeeds with derivation $\Gamma, P : \mathbf{F} \vdash \Delta$.
Subcase $\Gamma, P : \mathcal{A}' \wedge \mathcal{A}'' \vdash \Delta$.
The procedure recurses with $\Gamma, P : \mathcal{A}', P : \mathcal{A}'' \vdash \Delta$; if the recursion fails, it fails; if the recursion succeeds with a derivation for $\Gamma, P : \mathcal{A}', P : \mathcal{A}'' \vdash \Delta$, it produces a derivation for $\Gamma, P : \mathcal{A}' \wedge \mathcal{A}'' \vdash \Delta$ by ($\wedge$ L).
Subcase $\Gamma, P : \mathcal{A}' \Rightarrow \mathcal{A}'' \vdash \Delta$.
The procedure recurses with $\Gamma \vdash P : \mathcal{A}', \Delta$ and $\Gamma, P : \mathcal{A}'' \vdash \Delta$; if either recursion fails, the procedure fails. If the recursions succeed with derivations for $\Gamma \vdash P : \mathcal{A}', \Delta$ and $\Gamma, P : \mathcal{A}'' \vdash \Delta$ the procedure produces a derivation for $\Gamma, P : \mathcal{A}' \Rightarrow \mathcal{A}'' \vdash \Delta$ by ($\Rightarrow$ L).
Subcase $\Gamma, P : \mathbf{0} \vdash \Delta$.
If $P \not\equiv \mathbf{0}$ (a decidable test) the procedure returns with the derivation $\Gamma, P : \mathbf{0} \vdash \Delta$ by ($\mathbf{0}$ L), otherwise it recurses with $\Gamma \vdash \Delta$. If the recursion fails, it fails; if it succeeds with a derivation for $\Gamma \vdash \Delta$, it returns a derivation for $\Gamma, P : \mathbf{0} \vdash \Delta$ by weakening.
Subcase $\Gamma, P : \mathcal{A}' \mid \mathcal{A}'' \vdash \Delta$.
The procedure computes the finite set $T(P)$, and for every $\langle P', P'' \rangle$ belonging to it, it recurses with $\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta$. If all the recursive calls succeed, the procedure builds a derivation for $\Gamma, P : \mathcal{A}' \mid \mathcal{A}'' \vdash \Delta$ by ($\mid$ L), otherwise it fails.
Subcase $\Gamma, P : \mathcal{A}' \triangleright \mathcal{A}'' \vdash \Delta$.
The procedure computes the finite set $T(\mathcal{A}' \triangleright \mathcal{A}'')$, and for every P' belonging to it, it recurses with $\Gamma \vdash P' : \mathcal{A}', \Delta$ and $\Gamma, P' \mid P : \mathcal{A}'' \vdash \Delta$. If one pair of recursive calls succeeds, the procedure builds a derivation for $\Gamma, P : \mathcal{A}' \triangleright \mathcal{A}'' \vdash \Delta$ by ($\triangleright$ L), otherwise it fails.

Subcase $\Gamma, P : n[\mathcal{A}'] \vdash \Delta$.

The procedure computes the finite set $T(n, P)$ (which may be empty). For every P' belonging to it, the procedure recurses with $\Gamma, P' : \mathcal{A}' \vdash \Delta$. If all the recursive calls succeed, the procedure builds a derivation for $\Gamma, P : n[\mathcal{A}'] \vdash \Delta$ by ($n[]$ L), otherwise it fails.

Subcase $\Gamma, P : \mathcal{A}'@n \vdash \Delta$.

The procedure recurses with $\Gamma, n[P] : \mathcal{A}' \vdash \Delta$. If the recursive call succeeds, the procedure builds a derivation for $\Gamma, P : \mathcal{A}'@n \vdash \Delta$ by ($@n$ L), otherwise it fails.

Case *size* ≥ 1, right rules.

Subcase $\Gamma \vdash P : \mathbf{F}, \Delta$.

The procedure recurses with $\Gamma \vdash \Delta$. If the recursion fails, the procedure fails. If the recursion succeeds with a derivation for $\Gamma \vdash \Delta$, the procedure returns a derivation for $\Gamma \vdash P : \mathbf{F}, \Delta$ by ($\mathbf{F}$ R).

Subcase $\Gamma \vdash P : \mathcal{A}' \wedge \mathcal{A}'', \Delta$.

The procedure recurses with $\Gamma \vdash P : \mathcal{A}', \Delta$ and $\Gamma \vdash P : \mathcal{A}'', \Delta$. If both recursive calls succeeds, the procedure builds a derivation for $\Gamma \vdash P : \mathcal{A}' \wedge \mathcal{A}'', \Delta$ by ($\wedge$ R), otherwise it fails.

Subcase $\Gamma \vdash P : \mathcal{A}' \Rightarrow \mathcal{A}'', \Delta$.

The procedure recurses with $\Gamma, P : \mathcal{A}' \vdash P : \mathcal{A}'', \Delta$. If the recursion fails, it fails; if the recursion succeeds with a derivation for $\Gamma, P : \mathcal{A}' \vdash P : \mathcal{A}'', \Delta$, it produces a derivation for $\Gamma, P : \mathcal{A}' \Rightarrow \mathcal{A}'' \vdash \Delta$ by ($\Rightarrow$ R).

Subcase $\Gamma \vdash P : \mathbf{0}, \Delta$.

If $P \equiv \mathbf{0}$ (a decidable test) the procedure returns with the derivation $\Gamma \vdash P : \mathbf{0}, \Delta$ by ($\mathbf{0}$ R), otherwise it recurses with $\Gamma \vdash \Delta$. If the recursion fails, it fails; if it succeeds with a derivation for $\Gamma \vdash \Delta$, it returns a derivation for $\Gamma \vdash P : \mathbf{0}, \Delta$ by weakening.

Subcase $\Gamma \vdash P : \mathcal{A}' \mid \mathcal{A}'', \Delta$.

The procedure computes the finite set $T(P)$, and for every $\langle P', P'' \rangle$ belonging to it, it recurses with $\Gamma \vdash P' : \mathcal{A}', \Delta$ and $\Gamma \vdash P'' : \mathcal{A}'', \Delta$. If one pair of recursive calls succeeds, the procedure builds a derivation for $\Gamma \vdash P : \mathcal{A}' \mid \mathcal{A}'', \Delta$ by ($\mid$ R), otherwise it fails.

Subcase $\Gamma \vdash P : \mathcal{A}' \triangleright \mathcal{A}'', \Delta$.

The procedure computes the finite set $T(\mathcal{A}' \triangleright \mathcal{A}'')$, and for every P' belonging to it, it recurses with $\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta$. If all these recursive calls are successful, the procedure builds a derivation for $\Gamma \vdash P : \mathcal{A}' \triangleright \mathcal{A}'', \Delta$ by ($\triangleright$ R), otherwise it fails.

Subcase $\Gamma \vdash P : n[A'], \Delta$.

The procedure computes the finite set $T(n, P)$. If $T(n, P)$ is empty, then it recurses with $\Gamma \vdash \Delta$; if the recursion fails the procedure fails, and if it succeeds with a derivation for $\Gamma \vdash \Delta$, the procedure returns a derivation for $\Gamma \vdash P : n[\mathcal{A}'], \Delta$ by weakening. If $T(n, P)$ is not empty, then for every P' belonging to it, the procedure recurses with $\Gamma \vdash P' : \mathcal{A}', \Delta$. If one of the recursive calls succeeds, the procedure builds a derivation for $\Gamma \vdash P : n[\mathcal{A}'], \Delta$ by ($n[]$ R), otherwise it fails.

Subcase $\Gamma \vdash P : \mathcal{A}'@n, \Delta$.

The procedure recurses with $\Gamma \vdash n[P] : \mathcal{A}', \Delta$. If the recursive call succeeds, the procedure builds a derivation for $\Gamma \vdash P : \mathcal{A}'@n, \Delta$ by (@n R), otherwise it fails. □

By combining Lemma 4 and Theorems 3 and 4, we can equate the validity problem to a particular proof search problem.

Corollary 2. $\mathbf{vld}(\mathcal{A})$ *if and only if* $\vdash \mathbf{0} : \mathbf{T} \triangleright \mathcal{A}$ *has a derivation.*

Hence, by Theorem 5, we obtain an algorithm for validity based on deduction.

5 Conclusions

This paper concerns a propositional spatial logic for finite edge-labelled trees. The spatial modalities are composition $\mathcal{A} \mid \mathcal{B}$, guarantee $\mathcal{A} \triangleright \mathcal{B}$, void $\mathbf{0}$, location $n[\mathcal{A}]$, and placement $\mathcal{A}@n$. We show two main things. First, satisfaction and validity are equivalent and decidable. Second, there is a sound and complete proof system for validity. We know of no previous algorithms for satisfaction or validity in the presence of the guarantee operator.

The spatial logic of this paper is a fragment of the ambient logic introduced by Cardelli and Gordon [CG00,CG01b]. Model checking algorithms for various fragments without guarantee have been proposed [CDZG$^+$01,CT01].

The proof of decidability for validity presented here is based on a technique introduced by Calcagno, Yang, and O'Hearn [CYO01], in the setting of a spatial logic for mutable data structures.

We briefly consider the prospects of extending our results:

- Charatonik and Talbot [CT01] show that validity becomes undecidable in a spatial logic with name quantification. (Their result depends only on the presence of propositional logic, $\mathbf{0}$, $n[\mathcal{A}]$, $\mathcal{A} \mid \mathcal{B}$, and $\forall x.\mathcal{A}$.)
- Caires and Monteiro [CM98] and Cardelli and Gordon [CG01b] introduce logical modalities to deal with fresh names. A prerequisite of studying these operators would be to enrich our tree model with fresh names.

References

CC01. L. Caires and L. Cardelli. A spatial logic for concurrency (part I). In *Theoretical Aspects of Computer Software (TACS 2001)*, volume 2215 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2001.

CC02. L. Caires and L. Cardelli. A spatial logic for concurrency (part II). Submitted for publication, 2002.

CDZG$^+$01. W. Charatonik, S. Dal Zilio, A. D. Gordon, S. Mukhopadhyay, and J.-M. Talbot. The complexity of model checking mobile ambients. In *Proceedings FoSSaCS'01*, volume 2030 of *LNCS*, pages 152–167. Springer, 2001. An extended version appearas as Technical Report MSR–TR–2001–03, Microsoft Research, 2001.

CG00. L. Cardelli and A.D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *27th ACM Symposium on Principles of Programming Languages (POPL'00)*, pages 365–377, 2000.

CG01a. L. Cardelli and G. Ghelli. A query language based on the ambient logic. In *Proceedings of the 9th European Symposium on Programming (ESOP'01)*, volume 2028 of *LNCS*, pages 1–22. Springer, 2001.

CG01b. L. Cardelli and A.D. Gordon. Logical properties of name restriction. In *Proceedings of the 5th International Conference on Typed Lambda Calculi and Applications (TLCA'01)*, volume 2044 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2001.

CGG02. L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *ICALP'02*, 2002. To appear.

CM98. L. Caires and L. Monteiro. Verifiable and executable logic specifications of concurrent objects in L_π. In *Proceedings of the 7th European Symposium on Programming (ESOP'99)*, volume 1381 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 1998.

CT01. W. Charatonik and J.-M. Talbot. The decidability of model checking mobile ambients. In *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic*, volume 2142 of *LNCS*, pages 339–354. Springer, 2001.

CYO01. C. Calcagno, H. Yang, and P. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *Proceedings of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01)*, volume 2245 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2001.

IO01. S. Ishtiaq and P. W. O'Hearn. BI as an assertion language for mutable data structures. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programmoing Languages (POPL)*, pages 14–26. ACM Press, jan 2001.

ORY01. P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In L. Fribourg, editor, *CSL 2001*, pages 1–19. Springer-Verlag, 2001. LNCS 2142.

Rey00. J. C. Reynolds. Intuitionistic reasoning about shared mutable data structure. In *Millennial Perspectives in Computer Science*. Palgrave, 2000.