Deciding Validity in a Spatial Logic for Trees

Cristiano Calcagno

Imperial College London

Luca Cardelli, Andrew D. Gordon

Microsoft Research

Abstract

We consider a propositional spatial logic for finite trees. The logic includes $\mathcal{A} \mid \mathcal{B}$ (tree composition), $\mathcal{A} \triangleright \mathcal{B}$ (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. We envisage applications in the area of logic-based type systems for semistructured data. We describe a small programming language based on this idea.

---

1 Introduction

Due to the growing popularity of semistructured data (Buneman, 1997), and particularly XML (Bray *et al.*, 1998), there is a renewed interest in typed programming languages that can manipulate tree-like data structures. Unfortunately, semistructured data cannot be checked by conventional type systems with sufficient flexibility. More advanced type systems are being proposed that better match the data schemas used with semistructured data (Hosoya & Pierce, 2000).

In general, we are going to have some tree-like data t, and some description language T that can flexibly describe the shape of the data. We are interested in description languages so flexible that they are akin to logics rather than to type systems. The question is: what is needed to use a description language T as a type system in some programming language that manipulates t data?

First of all, the programming language needs to analyze the data, so it needs to check at run-time whether a tree value matches a description. In type system terms this is a run-time typing problem: does tree t have type A. In logical terms this is a satisfaction problem: does tree t satisfy formula A.

Second, the programming language needs (most likely) to check at compile time whether a description A is less general than a description B. In terms of type system this is a subtyping test: is type A a subtype of type B. In logic terms this is a validity test: does every tree t satisfying formula A also satisfy formula B.

Given both a satisfaction and a validity algorithm, it is then fairly routine to

build a type system around the description language, along with an operational semantics obeying standard typing soundness properties. The key problem, though, is to find rich description languages that admit satisfaction and (more crucially) validity algorithms. In the case of XDuce (Hosoya & Pierce, 2001), for example, these algorithms are found in tree automata theory.

We propose here a logic that can be used as a rich description language for tree-like data. It emerges as an application of the novel area of "spatial" logics used for describing data and network structures. The logic of this paper is so expressive that, in fact, satisfaction and validity are equivalent problems (validity can be defined internally). For a restricted version of the spatial logics studied so far, we are able to obtain a validity algorithm, and this is sufficient for language applications. We end this paper by describing a simple language based on these ideas.

In a spatial logic, the truth of a formula depends on its location. Models for spatial logics include computational structures such as concurrent objects (Caires & Monteiro, 1998), heaps (Reynolds, 2002; Ishtiaq & O'Hearn, 2001; O'Hearn *et al.*, 2001), trees (Cardelli & Ghelli, 2001), graphs (Cardelli *et al.*, 2002), and also process calculi such as the π-calculus (Caires & Cardelli, 2003; Caires & Cardelli, 2002) and the ambient calculus (Cardelli & Gordon, 2000; Cardelli & Gordon, 2001). Previous applications of spatial logics include specifying and verifying imperative and concurrent programs, and querying semistructured data.

The spatial logic of this paper describes 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 optimizations than the algorithm based directly on model checking.

Finally, we show an application of our decidability results: a λ-calculus to manipulate tree data, where logical formulas are used as base types. We present a typechecking algorithm parameterized on a validity algorithm for the tree logic, allowing the use of optimized validity algorithms when they become available.

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 describes our λ-calculus for manipulating trees, to illustrate the idea of using spatial logic formulas as programming language types. Section 6 concludes.

An abridged version of this work appears in a conference paper (Calcagno *et al.*, 2003). A technical report (Calcagno *et al.*, 2002) includes all omitted proofs.

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 certain characteristic formulas, we note that satisfaction and validity are interderivable.

In a study of a richer spatial logic than the one considered here, Hirschkoff, Lozes, and Sangiorgi (2002) also define characteristic formulas for ambient processes, and note equivalences between the satisfaction and validity problems.

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 $\mathcal{A}$. $\quad\square$

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. $\quad\square$

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 (Cardelli & Gordon, 2000), 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:

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

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: $\mathbf{vld}(\mathcal{A}) \Leftrightarrow (\forall P.P \models \mathcal{A}) \Leftrightarrow (\forall P.P \models \mathbf{T} \Rightarrow P \mid \mathbf{0} \models \mathcal{A}) \Leftrightarrow \mathbf{0} \models \mathbf{T} \triangleright \mathcal{A}$. $\quad\square$

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: $\mathbf{vld}(\underline{P} \Rightarrow \mathcal{A}) \Leftrightarrow (\forall Q.Q \models \underline{P} \Rightarrow Q \models \mathcal{A}) \Leftrightarrow (\forall Q.Q \equiv P \Rightarrow Q \models \mathcal{A}) \Leftrightarrow P \models \mathcal{A}$. $\quad\square$

Hence, the validity and satisfaction problems are equivalent. The goal of the paper is to show 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 $Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}$ 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 $Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}$ 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 $Q \models \mathcal{A} \Rightarrow P \mid Q \models \mathcal{B}$ 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

We need the following facts relating substitution with the operators for adding an edge to a tree and for composing trees.

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

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

Proof

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

Lemma 7

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

Proof
Immediate since substitution preserves the structure of trees. $\square$

Given these facts we can show that satisfaction of a formula is independent of any name not occurring in the formula.

Lemma 8

If $m, m' \notin fn(\mathcal{A})$, $P \models \mathcal{A} \Leftrightarrow P\{m \leftarrow m'\} \models \mathcal{A}$.

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:

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

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

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

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:

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

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

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

This lemma is not true 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:

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

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 $Q \models \mathcal{A} \Leftrightarrow Q' \models \mathcal{A}$. 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:

$$\begin{aligned}
&(P \mid Q')\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\} \\
&= \quad P' \mid Q' \\
&= \quad (P \mid Q)\{n_1 \leftarrow m\} \cdots \{n_k \leftarrow m\}
\end{aligned}$$

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

3.2 Bounding the Sizes to Consider

We introduce measures of the height and width (h, w) of both trees and formulas, and define a binary relation on trees parameterized on size. We show that formulas of size (h, w) cannot distinguish between (h, w)-related trees, and that for any tree there is an (h, w)-related tree of size at most (h, w).

Definition 1 (Notation)
Write $a \cdot P$ for $a \geq 0$ 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, $\ldots$, a_k, n_k, P_k, for some k, such that the following properties hold:

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

When $|P|^{\mathsf{hw}} = (h, w)$, we write $|P|^{\mathsf{h}}$ for h and $|P|^{\mathsf{w}}$ for w. We 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 non-empty trees under the same edge. For example:

- $|\mathbf{0}|^{\mathsf{hw}} = (0, 0)$

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

Next, we define height and width measures for logical formulas.

Size of Logical Formulas

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

Here are the sizes for the derived propositional connectives:

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

We define a relation $\sim_{h,w}$ between trees, parameterized by the size (h, w). Intuitively, a formula of height h can only distinguish trees on the basis of structure no deeper than h, and a formula of width w can only distinguish trees on the basis of up to w duplicate occurrences of equivalent subtrees. 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 $P \sim_{h,w} Q$)

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

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 antimonotone in (h, w).

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

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

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 $P \mid Q \sim_{h+1,w} P' \mid Q'$. $\square$

Lemma 12 (Inversion)
If $P' \mid P'' \sim_{h,w_1+w_2} Q$ then there exist Q', Q'' such that $Q \equiv Q'|Q''$ 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 $P \sim_{h+1,w} n[P_1] \mid n[P_1] \mid P_3$. This shows that for any (h, w) and any P it is possible to find a $P^\dagger$ such that $P^\dagger$ is in (h, w)-normal form and $P \sim_{h+1,w} P^\dagger$.

Let w be $w_1 + w_2$. We can assume without loss of generality that P' and P'' and Q are in (h, w)-normal form, since P' and P'' can be freely replaced with $(h + 1, w)$-related trees and, by Lemma 10, Q can be replaced with any tree that can equally be split in $(h+1, w)$-related subtrees. Hence, there exist k, P_j, a_j', a_j'', b_j for $j \in 1..k$ such that

$$P' \equiv a_1' \cdot n_1[P_1] \mid \cdots \mid a_k' \cdot n_k[P_k]$$
$$P'' \equiv a_1'' \cdot n_1[P_1] \mid \cdots \mid a_k'' \cdot n_k[P_k]$$
$$Q \equiv b_1 \cdot n_1[P_1] \mid \cdots \mid b_k \cdot n_k[P_k]$$

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:

$$b_i = b_i' + b_i'' \tag{1}$$
$$a_i' \cdot n_i[P_i] \sim_{h+1,w_1} b_i' \cdot n_i[P_i] \tag{2}$$
$$a_i'' \cdot n_i[P_i] \sim_{h+1,w_2} b_i'' \cdot n_i[P_i] \tag{3}$$

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 to (2) and (3) we get $P' \sim_{h+1,w_1} Q'$ and $P'' \sim_{h+1,w_2} Q''$ respectively. $\square$

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:

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

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 $P \sim_{h+1,w} P'$. $\square$

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

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

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 we have:

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

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:

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

Proof
We have:

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

Proposition 1 justifies the first step, Proposition 3 the second. $\square$

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 bounded by $max(|\mathcal{A}|^{\mathsf{hw}}, |\mathcal{A}|^{\mathsf{hw}})$.

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 metavariable c ranges over sets of trees modulo structural congruence:

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

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

Definition 5
If $X = \{n_1, \ldots, n_k\}$, define $EQ_{h,w}^X$ as follows:

$$
\begin{aligned}
EQ_{0,w}^X &\triangleq \{\langle \mathbf{0} \rangle_{\equiv}\} \\
EQ_{h+1,w}^X &\triangleq (n_1[EQ_{h,w}^X] + \cdots + n_k[EQ_{h,w}^X])^{\leq w}
\end{aligned}
$$

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

Lemma 14
$\langle P \rangle_{\equiv} \in EQ_{h,w}^X \Leftrightarrow P \in \mathsf{Tree}_X \wedge |P|^{\mathsf{hw}} \leq (h,w)$.

Proof
By construction, if $\langle P \rangle_{\equiv} \in EQ_{h,w}^X$ then $P \in \mathsf{Tree}_X$ and $|P|^{\mathsf{hw}} \leq (h,w)$. The converse follows from a simple induction on h. $\square$

We show that congruence and (h,w)-equivalence coincide on trees of size at most (h,w).

Lemma 15
If $|P|^{\mathsf{hw}} \leq (h,w)$ and $|P'|^{\mathsf{hw}} \leq (h,w)$, then

$$
P \equiv P' \iff P \sim_{h,w} P'
$$

Proof
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 $P \equiv P'$. $\square$

The following proposition shows that $EQ_{h,w}^X$ is an enumeration of the representatives 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 $f' : EQ_{h,w}^X \to \mathsf{Tree}_X / \sim_{h,w}$.

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 $\langle P' \rangle_\equiv \in EQ_{h,w}^X$ by Lemma 14.

To show that f' is injective, consider any $P, Q \in \mathsf{Tree}_X$ with $\langle P \rangle_\equiv, \langle Q \rangle_\equiv \in EQ_{h,w}^X$ 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 14, hence $P \equiv Q$ by Lemma 15. This proves $\langle P \rangle_\equiv = \langle Q \rangle_\equiv$. $\square$

Theorem 1 (Finite Bound)
Consider any formulas $\mathcal{A}$ and $\mathcal{B}$. Let $EQ_{h,w}^X = \{\langle Q_1 \rangle_\equiv, \ldots, \langle Q_n \rangle_\equiv\}$, 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:

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

Proof
Using Proposition 4, Lemma 14, and Lemma 2:

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

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 $EQ_{h,w}^{X}$, 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 (singleton or empty) set $\{Q \mid P \equiv n[Q]\}/\equiv$.

Lemma 16

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

Proof
Part (2) follows at once from Theorem 1. The other parts follow easily, as in earlier work (Cardelli & Gordon, 2000). $\square$

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. $\square$

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 $EQ_{h,w}^{X} = \{\langle P_1 \rangle_{\equiv}, \ldots, \langle P_n \rangle_{\equiv}\}$, where $(h, w) = |\mathcal{A}|^{\mathsf{hw}}$ and $X = fn(\mathcal{A}) \cup \{m\}$ for some $m \notin fn(\mathcal{A})$. Then

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

It is straightforward to implement the algorithms for satisfaction and validity suggested above. However, they are of limited practical interest, since the size of $EQ_{h,w}^{X}$ is not elementary (not bounded by any tower of exponentials) in the worst case. The only lower bound we know is PSPACE. Still, the algorithm terminates in a reasonable time on small formulas. Here is a selection of formulas found to be valid by our implementation.

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

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

To see why, for example, that the formula $(\mathbf{0} \vee p[\mathbf{0}]) \mid \neg(p[\mathbf{0}])$ is valid, consider any process P. Either $P \models p[\mathbf{0}]$ or not. If so, we have $P \equiv P \mid \mathbf{0}$, and $P \models \mathbf{0} \vee p[\mathbf{0}]$ and $\mathbf{0} \models \neg(p[\mathbf{0}])$. If not, we have $P \equiv \mathbf{0} \mid P$, and $\mathbf{0} \models \mathbf{0} \vee p[\mathbf{0}]$ and $P \models \neg(p[\mathbf{0}])$. So, in either case, the process satisfies $(\mathbf{0} \vee p[\mathbf{0}]) \mid \neg(p[\mathbf{0}])$.

4 Deciding Validity by Deduction

We present a sequent calculus for our spatial logic, following the pattern of Caires and Cardelli (2002). 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. The algorithm of the previous section decides validity; in addition, the algorithm suggested by the sequent calculus of this section yields a deductive proof of validity, which is amenable to the optimizations typically used in theorem proving.

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 $T(P)$, $T(\mathcal{A} \rhd \mathcal{B})$, 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 $(\mid \text{L})$, $(\rhd \text{R})$, and $(n[] \text{L})$.

Rules of the Sequent Calculus:

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

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

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

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

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

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

$$\text{(}|\text{ L)}$$
$$\dfrac{\forall\langle Q,R\rangle\in T(P).\ \Gamma,Q:\mathcal{A},R:\mathcal{B}\vdash\Delta}{\Gamma,P:\mathcal{A}\,|\,\mathcal{B}\vdash\Delta}$$

$$\text{(}|\text{ R)}$$
$$\dfrac{\Gamma\vdash Q:\mathcal{A},\Delta\quad\Gamma\vdash R:\mathcal{B},\Delta\quad P\equiv Q\,|\,R}{\Gamma\vdash P:\mathcal{A}\,|\,\mathcal{B},\Delta}$$

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

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

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

The variables Q, R in $(|\ \text{L})$ and the variable Q in $(\triangleright\ \text{R})$ cannot occur free (in a formalistic reading) in Γ, P, Δ. Compare the side conditions on these rules in Caires and Cardelli (2002). Here, these are meta-level variables ranging over terms, so there is no need for such side conditions. Note that $(n[]\ \text{L})$ applies also when $T(n,P)$ is empty (something that never happens for $(|\ \text{L})$), 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 clause $(n[]\ R)$ of Lemma 18 below.

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). $\square$

4.2 Soundness and Completeness

We make a conventional interpretation of sequents:

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

To prove soundness and completeness of the sequent calculus, we need the following two lemmas.

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 $[\![\Gamma \vdash P : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma \vdash P : \mathcal{A}'', \Delta]\!]$
($\Rightarrow$ L) $[\![\Gamma, P : \mathcal{A}' \Rightarrow \mathcal{A}'' \vdash \Delta]\!]$ iff $[\![\Gamma \vdash P : \mathcal{A}', \Delta]\!] \wedge [\![\Gamma, P : \mathcal{A}'' \vdash \Delta]\!]$
($\Rightarrow$ R) $[\![\Gamma \vdash P : \mathcal{A}' \Rightarrow \mathcal{A}'', \Delta]\!]$ 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. $\square$

Lemma 19 (Finite Test Sets)

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

Proof
By expanding definitions, and appeal to Lemma 16. $\square$

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

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

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, $(\rhd$ 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 induction hypothesis, $\forall \langle P', P'' \rangle \in T(P).\Gamma, P' : \mathcal{A}', P'' : \mathcal{A}'' \vdash \Delta$ 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 induction hypothesis, $\forall P' \in T(n, P).\Gamma, P' : \mathcal{A}' \vdash \Delta$ has a derivation. Hence by ($n[\,]$ L) we can construct a (finite) derivation for $\Gamma, P : n[\mathcal{A}'] \vdash \Delta$.

Subcase $[\![\Gamma \vdash P : \mathcal{A}' \rhd \mathcal{A}'', \Delta]\!]$**.** By Lemma 18($\rhd$ 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 $T(\mathcal{A}' \rhd \mathcal{A}'')$ such that $\forall P' \in T(\mathcal{A}' \rhd \mathcal{A}'').[\![\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta]\!]$. By induction hypothesis, $\forall P' \in T(\mathcal{A}' \rhd \mathcal{A}'').\Gamma, P' : \mathcal{A}' \vdash P' \mid P : \mathcal{A}'', \Delta$ has a derivation. Hence by ($\rhd$ R) we can construct a (finite) derivation for $\Gamma \vdash P : \mathcal{A}' \rhd \mathcal{A}'', \Delta$.

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). $\square$

Proposition 6 (Id, Cut, and Contraction Elimination)
If $\Gamma \vdash \Delta$ has a derivation, then there is a derivation 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). $\square$

By combining Theorems 2, 3, and 4 we obtain:

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 $Q_j \models \mathcal{B}_j$. $\square$

Moreover, as the following theorem asserts, there is a procedure that yields an actual derivation, in the case that $\Gamma \vdash \Delta$ is derivable. The proof relies on Lemma 18 and appears elsewhere (Calcagno *et al.*, 2002).

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 $\Gamma \vdash \Delta$.

5 A Language for Manipulating Trees

We describe a typed λ-calculus that manipulates tree data. The type system of this calculus has, at its basis, tree types. Function types are built on top of the tree types in standard higher-order style. The tree types, however, are unusual: they are the formulas of our logic. Therefore, we can write types such as:

$$\mathbf{T} \to \neg\mathbf{0}$$
$$((\mathcal{A} \wedge \neg\mathbf{0}) \mid n[\mathcal{B}]) \to (n[\mathcal{A}] \mid \mathcal{B})$$

Logical operators can be applied only to tree types, not to higher-order types. A subtyping relation is defined between types. On tree types, subtyping is defined as validity of logical implication; that is, $\mathcal{A} <: \mathcal{B}$ means $\mathbf{vld}(\mathcal{A} \Rightarrow B)$. Subtyping is then extended to function types by the usual contravariant rule. This implies that a logical validity check is used during static typechecking, whenever we need to check type inclusion. Tree data is manipulated via pattern matching constructs that perform "run-time type checks". Since tree types are formulas, we have the full power of the logic to express the pattern matching conditions. Those run-time type checks are executed as run-time satisfaction checks. For example, one of our

matching constructs is a test to see whether the value denoted by expression t has type $\mathcal{A}$:

$$t?(x{:}\mathcal{A}).u, v$$

This construct first computes the tree P denoted by the expression t, and then performs a test $P \models \mathcal{A}$. If the test is successful, it binds P to x and executes u; otherwise it binds P to x and executes v. The variable x can be used both inside u and v, but in u it has type $\mathcal{A}$, while in v it has type $\neg\mathcal{A}$.

To summarize, our formulas are used as a very expressive type system for tree data, within a typed λ-calculus. A satisfaction algorithm is used to analyze data at run-time, and a validity algorithm is needed during static typechecking. We of course have such algorithms available, as described in previous sections, at least for ground terms and types. In the absence of polymorphism or dependent types, types are in fact ground. And, at run-time, all values are ground too. As usual, the type system checks whether an open term has a (ground) type: it can do so without additional difficulties, even though the basic satisfaction test we have is for closed terms (that is, trees).

Again, we do not claim that the validity algorithms of this paper yield efficient typecheckers; however, any improved algorithm can be slotted in without change.

5.1 Syntax

The λ-calculus is stratified in terms of low types and high types. The low types are, in this case, just tree types, but could in general include other basic data types such as integers and names. The high types are function types over the low types. The tree types are the formulas of our logic.

The same stratification holds on terms: there are terms of low types (the trees) and terms of high types (the functions). This stratification is not reflected in the syntax, essentially because variables can hold high or low values, but it is reflected in the operational semantics.

Syntax:

$\mathcal{F}, \mathcal{G}, \mathcal{H} ::=$	High Types
$\quad \mathcal{A}$	tree types (formulas of the logic)
$\quad \mathcal{F} \to \mathcal{G}$	function types
$t, u, v ::=$	terms
$\quad \mathbf{0}$	void
$\quad n[t]$	location
$\quad t \mid u$	composition
$\quad t?n[x{:}\mathcal{A}].u$	location match
$\quad t?(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u$	composition match
$\quad t?(x{:}\mathcal{A}).u, v$	tree type match
$\quad x$	variable
$\quad \lambda x{:}\mathcal{F}.t$	function
$\quad t(u)$	application

The syntax of terms provides: a standard λ-calculus fragment, the three basic tree constructors, and three matching operators for analyzing tree data. The tree type match construct performs a run-time check to see whether a tree matches a given formula. Then one needs other constructs to decompose the trees: a composition match splits a tree in two components, and a location match strips an edge from a tree. A zero match is redundant because of the tree type match construct.

These multiple matching constructs are designed to simplify the operational semantics and the type rules. In practice, one would use a single case statement over the structure of trees; this can be easily translated to the given matching constructs.

Example: Case Statement

case t of	analyze t
$\mathbf{0}.u_1,$	if $t \equiv \mathbf{0}$, run u_1, else
$n[x{:}\mathcal{A}].u_2,$	if $t \equiv n[P]$ and $P \models A$,
	bind P to x and run u_2, else
$(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u_3,$	if $t \equiv P \mid Q$ and $P \models \mathcal{A}$, $Q \models \mathcal{B}$,
	bind P to x, Q to y and run u_3,
else u_4	else run u_4
$\triangleq$	can be translated as:

$t?(z_1{:}\mathbf{0}).u_1,$
$t?(z_2{:}n[\mathcal{A}]).z_2?n[x{:}\mathcal{A}].u_2,$
$t?(z_3{:}\mathcal{A} \mid \mathcal{B}).z_3?(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u_3,$
u_4

Further, one may want to allow complex nested patterns, that can be translated to nested uses of the given matching constructs.

5.2 Values

Programs in the syntax of the previous section produce values; either tree values or function values (that is, closures). Over the tree values we define the usual structural congruence $\equiv$; the matching constructs of the language are not able to distinguish between structurally congruent trees. The function values are triples of a term t with respect to an input variable x (that is, essentially $\lambda x.t$) and a stack for free variables ρ. A stack ρ is a list of bindings x, F of variables to values, with possible repetitions of the variables.

Values:

$F, G, H ::=$	High Values
$\quad P$	tree values
$\quad \langle \rho, x, t \rangle$	function values
ρ is a list of x, F pairs	Stacks
$\rho[x{\leftarrow}F]$ is ρ plus an x, F pair at the end	
$\rho(x)$ is the last F associated with x (if any)	

5.3 Operational Semantics

The operational semantics is given by a relation $t \Downarrow_\rho F$ between terms t, stacks ρ, and values F, meaning that t can evaluate to F on stack ρ. The semantics makes use of the satisfaction relation $P \models \mathcal{A}$ from Section 2. We use, for example, $t \Downarrow_\rho P$ to indicate that t evaluates to a tree value P. We use $t \Downarrow_\rho \equiv P$ as an abbreviation for $t \Downarrow_\rho Q$ and $Q \equiv P$, for some Q.

Operational Semantics

(Red **0**)

$$\mathbf{0} \Downarrow_\rho \mathbf{0}$$

(Red |)

$$\frac{t \Downarrow_\rho P \quad u \Downarrow_\rho Q}{t \mid u \Downarrow_\rho P \mid Q}$$

(Red $n[]$)

$$\frac{t \Downarrow_\rho P}{n[t] \Downarrow_\rho n[P]}$$

(Red ?$n[]$)

$$\frac{t \Downarrow_\rho \equiv n[P] \quad P \models \mathcal{A} \quad u \Downarrow_{\rho[x \leftarrow P]} F}{t?n[x{:}A].u \Downarrow_\rho F}$$

(Red ?|)

$$\frac{t \Downarrow_\rho \equiv P' \mid P'' \quad P' \models \mathcal{A} \quad P'' \models \mathcal{B} \quad u \Downarrow_{\rho[x \leftarrow P'][y \leftarrow P'']} F}{t?(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u \Downarrow_\rho F}$$

(Red ?1)

$$\frac{t \Downarrow_\rho P \quad P \models \mathcal{A} \quad u \Downarrow_{\rho[x \leftarrow P]} F}{t?(x{:}\mathcal{A}).u, v \Downarrow_\rho F}$$

(Red ?2)

$$\frac{t \Downarrow_\rho P \quad P \models \neg\mathcal{A} \quad v \Downarrow_{\rho[x \leftarrow P]} F}{t?(x{:}\mathcal{A}).u, v \Downarrow_\rho F}$$

(Red Var)

$$\frac{x \in dom(\rho)}{x \Downarrow_\rho \rho(x)}$$

(Red Lam)

$$\lambda x{:}\mathcal{F}.t \Downarrow_\rho \langle \rho, x, t \rangle$$

(Red App)

$$\frac{t \Downarrow_\rho \langle \rho', x, t' \rangle \quad u \Downarrow_\rho G \quad t' \Downarrow_{\rho'[x \leftarrow G]} H}{t(u) \Downarrow_\rho H}$$

5.4 Type System

The type system uses environments E, which are lists of associations $x{:}\mathcal{F}$ of unique variables and their types. We indicate by $dom(E)$ the set of variables defined in E, by $E, x{:}\mathcal{F}$ the extension of E with a new association $x{:}\mathcal{F}$ (provided that $x \notin dom(E)$), and by $E(x)$ the type associated with x in E (provided that $x \in dom(E)$).

The judgments are:

Judgments:

$\mathcal{F} <: \mathcal{G}$	$\mathcal{F}$ is a subtype of $\mathcal{G}$
$E \vdash \diamond$	E is well-formed
$E \vdash t : \mathcal{F}$	t has type $\mathcal{F}$ in E

A validity test is used in the (Sub Tree) rule.

Type Rules:

(Env $\emptyset$) (Env x)

$$\dfrac{E \vdash \diamond \quad x \notin dom(E)}{}$$

$$\emptyset \vdash \diamond \qquad\qquad E, x{:}\mathcal{F} \vdash \diamond$$

(Term **0**) (Term $|$) (Term $n[]$)

$$\dfrac{E \vdash \diamond}{E \vdash \mathbf{0} : \mathbf{0}} \qquad \dfrac{E \vdash t : \mathcal{A} \quad E \vdash u : \mathcal{B}}{E \vdash t \,|\, u : \mathcal{A} \,|\, \mathcal{B}} \qquad \dfrac{E \vdash t : \mathcal{A}}{E \vdash n[t] : n[\mathcal{A}]}$$

(Term ?$|$) (Term ?$n[]$)

$$\dfrac{E \vdash t : \mathcal{A} \,|\, \mathcal{B} \quad E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{F}}{E \vdash t?(x{:}\mathcal{A} \,|\, y{:}\mathcal{B}).u : \mathcal{F}} \qquad \dfrac{E \vdash t : n[\mathcal{A}] \quad E, x{:}\mathcal{A} \vdash u : \mathcal{F}}{E \vdash t?n[x{:}\mathcal{A}].u : \mathcal{F}}$$

(Term ?)

$$\dfrac{E \vdash t : \mathcal{B} \quad E, x{:}\mathcal{A} \vdash u : \mathcal{F} \quad E, x{:}\neg\mathcal{A} \vdash v : \mathcal{F}}{E \vdash t?(x{:}\mathcal{A}).u, v : \mathcal{F}}$$

(Term Var) (Term Lam) (Term App)

$$\dfrac{E \vdash \diamond}{E \vdash x : E(x)} \qquad \dfrac{E, x{:}\mathcal{F} \vdash t : \mathcal{G}}{E \vdash \lambda x{:}\mathcal{F}.t : \mathcal{F} \to \mathcal{G}} \qquad \dfrac{E \vdash t : \mathcal{F} \to \mathcal{G} \quad E \vdash u : \mathcal{F}}{E \vdash t(u) : \mathcal{G}}$$

(Subsumption) (Sub Tree) (Sub $\to$)

$$\dfrac{E \vdash t : \mathcal{F} \quad \mathcal{F} <: \mathcal{G}}{E \vdash t : \mathcal{G}} \qquad \dfrac{\mathbf{vld}(\mathcal{A} \Rightarrow \mathcal{B})}{A <: B} \qquad \dfrac{\mathcal{F}' <: \mathcal{F} \quad \mathcal{G} <: \mathcal{G}'}{\mathcal{F} \to \mathcal{G} <: \mathcal{F}' \to \mathcal{G}'}$$

Since types are ground, we do not need reflexivity and transitivity rules for subtyping. Reflexivity for the base case derives from $\mathbf{vld}(\mathcal{A} \Rightarrow \mathcal{A})$.

In order to derive some basic results, we need to define a satisfaction relation between values and types. Over tree types, this is just the satisfaction relation of Section 2, $P \models \mathcal{A}$. This is then generalized to closures by saying that $\langle \rho, x, t \rangle \models \mathcal{F} \to \mathcal{G}$ if for every $F \models \mathcal{F}$, the result G of evaluating t with F bound to x on stack ρ, is such that $G \models \mathcal{G}$. Moreover we say that a stack satisfies an environment, $\rho \models E$, if $\rho(x) \models E(x)$ for all the variables defined in E.

Satisfaction:

$P \models \mathcal{A}$ as in Section 2

$H \models \mathcal{F} \to \mathcal{G}$ iff $H = \langle \rho, x, t \rangle$ and $\forall F, G.(F \models \mathcal{F} \land t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \models \mathcal{G}$

$\rho \models E$ iff $\forall x \in dom(E).\rho(x) \models E(x)$

Proposition 8 (Subsumption)
If $\mathcal{F} <: \mathcal{G}$ and $H \models \mathcal{F}$ then $H \models \mathcal{G}$.

Proof
Induction on the derivation of $\mathcal{F} <: \mathcal{G}$.

(Sub Tree) We have $\mathbf{vld}(\mathcal{A} \Rightarrow \mathcal{B})$ and $H \models \mathcal{A}$; hence H is a tree value, and $H \models \mathcal{B}$ by definition of $\mathbf{vld}$.

(Sub $\rightarrow$) We have $\mathcal{F}' <: \mathcal{F}$ and $\mathcal{G} <: \mathcal{G}'$, and $H \models \mathcal{F} \rightarrow \mathcal{G}$. By definition, $H = \langle \rho, x, t \rangle$ and $\forall F, G.(F \models \mathcal{F} \wedge t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \models \mathcal{G}$. Take any $F \models \mathcal{F}'$; by Ind Hyp $F \models \mathcal{F}$. Assume $t \Downarrow_{\rho[x \leftarrow F]} G$, then $G \models \mathcal{G}$, and by Ind Hyp $G \models \mathcal{G}'$. We have shown that $\forall F, G.(F \models \mathcal{F}' \wedge t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \models \mathcal{G}'$. That is, we have shown that $\langle \rho, x, t \rangle \models \mathcal{F}' \rightarrow \mathcal{G}'$. $\square$

Proposition 9 (Subject Reduction)
If $E \vdash t : \mathcal{F}$ and $\rho \models E$ and $t \Downarrow_\rho F$, then $F \models \mathcal{F}$.

Proof
Induction on the derivation of $E \vdash t : \mathcal{F}$.

(Term 0) We have $E \vdash \mathbf{0} : \mathbf{0}$ and $\rho \models E$ and $\mathbf{0} \Downarrow_\rho \mathbf{0}$. By definition, $\mathbf{0} \models \mathbf{0}$.

(Term $n[]$) We have $E \vdash n[t] : n[\mathcal{A}]$ and $\rho \models E$ and $n[t] \Downarrow_\rho F$. We must have from (Term $n[]$) that $E \vdash t : \mathcal{A}$. We must have from (Red $n[]$) that $F = n[P]$ and $t \Downarrow_\rho P$. By Ind Hyp, $P \models \mathcal{A}$, hence by definition $n[P] \models n[\mathcal{A}]$.

(Term $|$) We have $E \vdash t \mid u : \mathcal{A} \mid \mathcal{B}$ and $\rho \models E$ and $t \mid u \Downarrow_\rho F$. We must have from (Term $|$) that $E \vdash t : \mathcal{A}$ and $E \vdash u : \mathcal{B}$. We must have from (Red $|$) that $F = P \mid Q$ and $t \Downarrow_\rho P$ and $u \Downarrow_\rho Q$. By Ind Hyp, $P \models \mathcal{A}$ and $Q \models \mathcal{B}$, hence by definition $t \mid u \models \mathcal{A} \mid \mathcal{B}$.

(Term $?n[]$) We have $E \vdash t?n[x{:}\mathcal{A}].u : \mathcal{F}$ and $\rho \models E$ and $t?n[x{:}\mathcal{A}].u \Downarrow_\rho F$. We must have from (Term $?n[]$) that $E \vdash t : n[\mathcal{A}]$ and $E, x{:}\mathcal{A} \vdash u : \mathcal{F}$. We must have from (Red $?n[]$) that $t \Downarrow_\rho \equiv n[P]$ and $P \models \mathcal{A}$ and $u \Downarrow_{\rho[x \leftarrow P]} F$. We have that $\rho[x \leftarrow P] \models E, x{:}\mathcal{A}$. By Ind Hyp $E, x{:}\mathcal{A} \vdash u : \mathcal{F}$ and $\rho[x \leftarrow P] \models E, x{:}\mathcal{A}$ and $u \Downarrow_{\rho[x \leftarrow P]} F$ implies $F \models \mathcal{F}$.

(Term $?|$) We have $t?(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u : \mathcal{F}$ and $\rho \models E$ and $t?(x{:}A \mid y{:}B).u \Downarrow_\rho F$. We must have from (Term $?|$) that $E \vdash t : \mathcal{A} \mid \mathcal{B}$ and $E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{F}$. We must have from (Red $?|$) that $t \Downarrow_\rho \equiv P' \mid P''$ and $P' \models \mathcal{A}$ and $P'' \models \mathcal{B}$ and $u \Downarrow_{\rho[x \leftarrow P'][y \leftarrow P'']} F$. We have that $\rho[x \leftarrow P'][y \leftarrow P''] \models E, x{:}\mathcal{A}, y{:}\mathcal{B}$. By Ind Hyp $E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{F}$ and $\rho[x \leftarrow P'][y \leftarrow P''] \models E, x{:}\mathcal{A}, y{:}\mathcal{B}$ and $u \Downarrow_{\rho[x \leftarrow P'][y \leftarrow P'']} F$ implies $F \models \mathcal{F}$.

(Term $?$) We have $E \vdash t?(x{:}\mathcal{A}).u, v : \mathcal{F}$ and $\rho \models E$ and $t?(x{:}\mathcal{A}).u, v \Downarrow_\rho F$. We must have from (Term $?$) that $E \vdash t : \mathcal{B}$ and $E, x{:}\mathcal{A} \vdash u : \mathcal{F}$ and $E, x{:}\neg\mathcal{A} \vdash v : \mathcal{F}$. The reduction may come from (Red $?1$); then $t \Downarrow_\rho P$ and $P \models \mathcal{A}$ and $u \Downarrow_{\rho[x \leftarrow P]} F$. We have that $\rho[x \leftarrow P] \models E, x{:}\mathcal{A}$. By Ind Hyp $E, x{:}\mathcal{A} \vdash u : \mathcal{F}$ and $\rho[x \leftarrow P] \models E, x{:}\mathcal{A}$ and $u \Downarrow_{\rho[x \leftarrow P]} F$ implies $F \models \mathcal{F}$. Else the reduction must come from (Red $?2$); then $t \Downarrow_\rho P$ and $P \models \neg\mathcal{A}$ and $v \Downarrow_{\rho[x \leftarrow P]} F$. We have that $\rho[x \leftarrow P] \models E, x{:}\neg\mathcal{A}$. By Ind Hyp $E, x{:}\neg\mathcal{A} \vdash v : \mathcal{F}$ and $\rho[x \leftarrow P] \models E, x{:}\neg\mathcal{A}$ and $v \Downarrow_{\rho[x \leftarrow P]} F$ implies $F \models \mathcal{F}$.

(Term Var) We have $E \vdash x : E(x)$ and $\rho \models E$ and $x \Downarrow_\rho F$. We must have from (Red Var) that $F = \rho(x)$. Since $\rho \models E$, we have that $\rho(x) \models E(x)$, that is, $F \models E(x)$.

(Term Lam) We have $E \vdash \lambda x{:}\mathcal{F}.t : \mathcal{F} \to \mathcal{G}$ and $\rho \models E$ and $\lambda x{:}\mathcal{F}.t \Downarrow_\rho F$. We must have from (Red Lam) that $F = \langle \rho, x, t \rangle$. We need to show that $\langle \rho, x, t \rangle \models \mathcal{F} \to \mathcal{G}$, that is, that $\forall F, G.(F \models \mathcal{F} \wedge t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \models \mathcal{G}$. Take any $F \models \mathcal{F}$, then $\rho[x \leftarrow F] \models E, x{:}\mathcal{F}$. Assuming that $t \Downarrow_{\rho[x \leftarrow F]} G$ we need to show that $G \models \mathcal{G}$. We must have from (Term Lam) that $E, x{:}\mathcal{F} \vdash t : \mathcal{G}$. By Ind Hyp if $\rho[x \leftarrow F] \models E, x{:}\mathcal{F}$ and $t \Downarrow_{\rho[x \leftarrow F]} G$, then $G \models \mathcal{G}$.

(Term App) We have $E \vdash t(u) : \mathcal{F}$ and $\rho \models E$ and $t(u) \Downarrow_\rho F$. We must have from (Term App) that $E \vdash t : \mathcal{G} \to \mathcal{F}$ and $E \vdash u : \mathcal{G}$. We must have from (Red App) that $t \Downarrow_\rho \langle \rho', x, t' \rangle$ and $u \Downarrow_\rho \mathcal{G}$ and $t' \Downarrow_{\rho'[x \leftarrow G]} F$. By Ind Hyp if $E \vdash t : \mathcal{G} \to \mathcal{F}$ and $\rho \models E$ and $t \Downarrow_\rho \langle \rho', x, t' \rangle$ then $\langle \rho', x, t' \rangle \models \mathcal{G} \to \mathcal{F}$. That means that $\forall G', F'.(G' \models \mathcal{G} \wedge t' \Downarrow_{\rho'[x \leftarrow G']} F') \Rightarrow F' \models \mathcal{F}$. By Ind Hyp if $E \vdash u : \mathcal{G}$ and $\rho \models E$ and $u \Downarrow_\rho G$ then $G \models \mathcal{G}$. Hence, by taking $G' = G$ and $F' = F$, we conclude $F \models \mathcal{F}$.

(Subsumption) We have $E \vdash t : \mathcal{F}$ and $\rho \models E$ and $t \Downarrow_\rho F$. We must have from (Subsumption) that $E \vdash t : \mathcal{G}$ and $\mathcal{G} <: \mathcal{F}$. By Ind Hyp, $F \models \mathcal{G}$. By Proposition 8, $F \models \mathcal{F}$. $\square$

5.5 Examples

The following program inspects an arbitrary tree (that is, anything of type $\mathbf{T}$). If the tree is $\mathbf{0}$ it returns the tree $a[\mathbf{0}]$, otherwise it returns the input tree. Hence the result is never $\mathbf{0}$, and the result type can be set to $\neg\mathbf{0}$.

$$\lambda x{:}\mathbf{T}.x?(y{:}\mathbf{0}).a[\mathbf{0}], y : \mathbf{T} \to \neg\mathbf{0}$$

Here is a (truncated) typing derivation; note the use of the subsumption rule to determine that $a[\mathbf{0}] <: \neg\mathbf{0}$. Each judgment is derived from the lines above it at the next level of indentation.

$E, x{:}\mathbf{T} \vdash x{:}\mathbf{T}$	(Term Var)
$E, x{:}\mathbf{T}, y{:}\mathbf{0} \vdash \mathbf{0} : \mathbf{0}$	(Term $\mathbf{0}$)
$E, x{:}\mathbf{T}, y{:}\mathbf{0} \vdash a[\mathbf{0}] : a[\mathbf{0}]$	(Term $n[]$)
$a[\mathbf{0}] <: \neg\mathbf{0}$	(Sub Tree)
$E, x{:}\mathbf{T}, y{:}\mathbf{0} \vdash a[\mathbf{0}] : \neg\mathbf{0}$	(Subsumption)
$E, x{:}\mathbf{T}, y{:}\neg\mathbf{0} \vdash y : \neg\mathbf{0}$	(Term Var)
$E, x{:}\mathbf{T} \vdash x?(y{:}\mathbf{0}).a[\mathbf{0}], y : \neg\mathbf{0}$	(Term ?)
$E \vdash \lambda x{:}\mathbf{T}.x?(y{:}\mathbf{0}).a[\mathbf{0}], y : \mathbf{T} \to \neg\mathbf{0}$	(Term Lam)

6 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 $\mathcal{A}@n$. There are two main results. First, satisfaction and validity

are equivalent and decidable. Second, there is a sound and complete proof system for validity.

The spatial logic of this paper is a fragment of the ambient logic introduced by Cardelli and Gordon (2000; 2001). Model checking algorithms for various fragments without guarantee have been proposed (Charatonik *et al.*, 2003). Lugiez and Dal Zilio (2002) show decidability of the satisfiability problem for another fragment of the ambient logic, but without guarantee; their techniques are based on tree automata.

Dal Zilio, Lugiez, and Meyssonnier (2004) recently propose a radically different approach. The key idea is to count the number of parallel occurrences of trees satisfying a given formula: subformulas describe the structure of the possible subtrees and Presburger arithmetic constraints determine the local numbers of subtrees of those shapes that can appear in parallel. Their formalism is naturally suited for extending the logic with a Kleene star operator and vertical recursion. Their approach enjoys a better complexity than ours, bounded by solving Presburger constraints, and hence may yield a practical typechecker for the type system of Section 5.

Validity for some other propositional substructural logics turns out to be undecidable. Urquhart (1984) proves undecidability for propositional relevant logic. Lincoln, Mitchell, Scedrov, and Shankar (1992) prove undecidability for both propositional linear logic and propositional intuitionistic linear logic. See Cardelli and Gordon (2000) for a detailed discussion of the differences between the ambient logic and relevant and linear logics.

Calcagno, Yang, and O'Hearn (2001) show decidability of validity in a propositional substructural logic for reasoning about heaps. The proof in this paper is an adaptation of their proof technique.

We briefly consider the prospects of extending our results:

- Charatonik and Talbot (2001) 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 (1998) and Cardelli and Gordon (2001) introduce logical modalities to deal with fresh names. A prerequisite of studying these operators would be to enrich our tree model with fresh names.
- Conforti and Ghelli (2004) study decidability for fragments of ambient logic with various restrictions on the use of quantifiers. The decidability proofs are based on the techniques described in this paper.

The technique of Lozes (2003) for adjunct elimination uses a measure similar to our notion of size and equivalence classes, where the size is given by the number of connectives in the logical formula. The main difference is that we show how to effectively enumerate equivalence classes, while on the other hand Lozes' approach applies to undecidable logics (where, as a consequence, the enumeration of equivalence classes cannot be effective).

We obtain only preliminary results about the complexity of validity for our logic from the constructions of this paper. It is easy to show that PSPACE is a lower-bound, by reduction from the Quantified Boolean Variables problem. However, there

is still a significant gap between PSPACE and the complexity of our algorithm: it is easy to see that the number of equivalence classes is not elementary (not bounded by a tower of exponentials) in the size parameter. We can obtain a higher complexity lower-bound for an extension of our logic with a Kleene star operator, $\mathcal{A}^*$ (zero or more copies of $\mathcal{A}$ in parallel). The extended logic can encode Presburger arithmetic, whose satisfiability problem is known to be complete for a class between double and triple exponential time. However, our algorithm cannot be trivially extended: there is a formula A^* that would invalidate our results when assigned any finite size.

Finally, building on some of the results of this paper, Cohen (2002) proposes improvements to the algorithms for satisfaction and validity of Section 3. He studies a multiset logic, able to encode our logic, and including Kleene star.

References

Bray, T., Pauli, J., & Sperberg-McQueen, C. M. (1998). *Extensible markup language.* W3C Recommendation 10–February–1998.

Buneman, P. (1997). Semistructured data. *Pages 117–121 of: 16th ACM Symposium on Principles of Database Systems (PODS'97).*

Caires, L., & Cardelli, L. (2002). A spatial logic for concurrency (Part II). *Pages 209–225 of: CONCUR 2002—Concurrency Theory.* Lecture Notes in Computer Science, vol. 2421. Springer.

Caires, L., & Cardelli, L. (2003). A spatial logic for concurrency (Part I). *Information and computation*, **186**(2), 194–235.

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

Calcagno, C., Yang, H., & O'Hearn, P. (2001). Computability and complexity results for a spatial assertion language for data structures. *Pages 108–119 of: Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01).* Lecture Notes in Computer Science, vol. 2245. Springer.

Calcagno, C., Cardelli, L., & Gordon, A. D. (2002). *Deciding validity in a spatial logic for trees.* Tech. rept. MSR–TR–2002–113. Microsoft Research.

Calcagno, C., Cardelli, L., & Gordon, A. D. (2003). Deciding validity in a spatial logic for trees. *Pages 62–73 of: Types in Language Design and Implementation (TLDI).*

Cardelli, L., & Ghelli, G. (2001). A query language based on the ambient logic. *Pages 1–22 of: Proceedings of the 9th European Symposium on Programming (ESOP'01).* LNCS, vol. 2028. Springer.

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

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

Cardelli, L., Gardner, P., & Ghelli, G. (2002). A spatial logic for querying graphs. *Pages*

597–610 of: Automata, Languages and Programming (ICALP'02). Lecture Notes in Computer Science, vol. 2380. Springer.

Charatonik, W., & Talbot, J.-M. (2001). The decidability of model checking mobile ambients. *Pages 339–354 of: Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic.* LNCS, vol. 2142. Springer.

Charatonik, W., Dal Zilio, S., Gordon, A. D., Mukhopadhyay, S., & Talbot, J.-M. (2003). The complexity of model checking mobile ambients. *Theoretical computer science*, **300**, 379–409.

Cohen, E. (2002). *Validity and model checking for logics of finite multisets.* Unpublished note, Microsoft Research.

Conforti, G., & Ghelli, G. (2004). Decidability of freshness, undecidability of revelation. *Pages 105–120 of: Foundations of Software Science and Computation Structures (FOSSACS'04).* Lecture Notes in Computer Science, vol. 2987. Springer.

Dal Zilio, S., Lugiez, D., & Meyssonnier, C. (2004). A logic you can count on. *Pages 135–146 of: 31st Symposium on Principles of Programming Languages (POPL'04).* ACM.

Hirschkoff, D., Lozes, E., & Sangiorgi, D. (2002). Separability, expressiveness, and decidability in the ambient logic. *Pages 423–432 of: Logic in Computer Science (LICS'02).* IEEE.

Hosoya, H., & Pierce, B. C. (2000). XDuce: A typed XML processing language. *Pages 226–244 of: Third International Workshop on the Web and Databases (WebDB2000).* Lecture Notes in Computer Science, vol. 1997. Springer.

Hosoya, H., & Pierce, B. C. (2001). Regular expression pattern matching for XML. *Pages 67–80 of: 28th Symposium on Principles of Programming Languages (POPL'01).* ACM.

Ishtiaq, S., & O'Hearn, P. W. (2001). BI as an assertion language for mutable data structures. *Pages 14–26 of: 28th Symposium on Principles of Programming Languages (POPL'01).* ACM.

Lincoln, P.D., Mitchell, J.C., Scedrov, A., & Shankar, N. (1992). Decision problems for propositional linear logic. *Annals of pure and applied logic*, **56**, 239–311.

Lozes, E. (2003). Adjuncts elimination in the static ambient logic. *10th International Workshop on Expressiveness in Concurrency (EXPRESS'03).*

Lugiez, D., & Dal Zilio, S. (2002). *Multitrees automata, Presburger's constraints and tree logics.* Laboratoire d'Informatique Fondamentale, CNRS and Université de Provence.

O'Hearn, P., Reynolds, J., & Yang, H. (2001). Local reasoning about programs that alter data structures. *Pages 1–19 of: Computer Science Logic (CSL'01).* Lecture Notes in Computer Science, vol. 2142. Springer.

Reynolds, J. C. (2002). Separation logic: a logic for shared mutable data structures. *Pages 55–74 of: Logic in Computer Science (LICS'02).* IEEE.

Urquhart, A. (1984). The undecidability of entailment and relevant implication. *Journal of symbolic logic*, **45**, 1059–1073.