Deciding safety properties in infinite-state pi-calculus via behavioural types[☆]

Lucia Acciai[*], Michele Boreale

Dipartimento di Sistemi e Informatica, Università di Firenze

Abstract

In the pi-calculus, we consider decidability of safety properties expressed in a simple spatial logic. We first introduce a behavioural type system that, given a (in general infinite-control) process P, tries to extract a spatial-behavioural type T, in the form of a CCS term that is logically equivalent to the given process. Using techniques based on well-structured transition systems (WSTS), we prove that satisfiability ($T \models \phi$) is decidable for types over an interesting fragment of the logic. The WSTS techniques we use require first endowing the considered transition system with a well-quasi order, then defining a finite basis for the denotation of each formula. This is achieved by viewing types as forests, with a well-quasi order that essentially corresponds to forest embedding. Finally, as a consequence of the logical equivalence between types and processes, we obtain the decidability of the considered fragment of the logic for well-typed pi-processes.

Keywords: pi-calculus, behavioural types, spatial logic, decidability, safety

1. Introduction

In the setting of process calculi, *spatial logics* [10, 9] and *behavioural type systems* [19, 17, 11, 3] have recently gained attention as useful tools for the analysis of concurrent systems. Spatial logics are well suited to express properties related to distribution, thanks to a combination of spatial and dynamic connectives. An example is

[☆]Research partly supported by the EU within the FET-GC2 initiative, project ASCENS. This paper is an extended and revised version of [2].

[*]Corresponding Author: Lucia Acciai, Dipartimento di Sistemi e Informatica, Viale Morgagni, 65 I-50134 Firenze, Italy. Tel: +39 055 4237441 Fax: +39 055 4237436, Email: lucia.acciai@unifi.it.

Email addresses: `lucia.acciai@unifi.it` (Lucia Acciai), `boreale@dsi.unifi.it` (Michele Boreale) the property expressing race-freedom on some channel a: "it is never the case that somewhere in the system there are two concurrent outputs ready at channel a". As discussed in [9], these logics are very rich and the equivalences they induce come very close to structural congruence. On the other hand, behavioural type systems are used to obtain abstract representation of message-passing systems and to simplify their analysis. The paper that initiated this approach is Igarashi and Kobayashi's work on generic type systems [17], where pi-calculus processes are abstracted by means of CCS types. The main property of Igarashi and Kobayashi's system is type soundness: any safety property (expressed in a simple logic) satisfied by a type is also satisfied by the processes that inhabit that type. A further elaboration on this theme can be found in [11].

In [3], we have combined ideas from spatial logics and behavioural type system into a single framework. Like in [17], the language of processes we consider is the pi-calculus and types are CCS terms. Differently from [17], though, types of [3] account for both the behaviour and the spatial structure of processes. This fact allows one to establish a strong correspondence between processes and their types. This makes it possible to prove type soundness for a fairly general class of properties, not just safety invariants. This enhancement comes at some price in terms of flexibility of the type system, though. A prominent feature of [3] is that *structural congruence* is used as a subtyping relation. This is consistent with the principle that processes and their types share, at least at a "shallow" level, the same spatial structure. This principle would be violated if adopting other forms of semantic subtyping, such as simulation preorders.

A driving motivation in all the mentioned works is being able to combine type checking and model checking. The idea is that, rather than model checking a given property against a process, with a behavioural type system at hand, one checks the property against a simpler model, that is a type. Moving from processes to types certainly implies a gain in simplicity in terms of reasoning [17, 3]. But surely something more precise can be said about the effectiveness of this approach: this is the goal of the present paper.

In [18], undecidability of behavioural type systems using the simulation preorder as a sub-typing relation has been proven. This result is a consequence of undecidability of simulation in BPP's. The result does not directly apply to the system of [17], which is "generic", but certainly suggests that any "reasonable" instance of this system based on simulation preorders might turn out to be undecidable. We may hope the situation is better for our system in [3]. As mentioned, this system adopts structural congruence as a subtyping relation, which is easily seen to be decidable for the considered language.

In the present paper, our goal is to show decidability of an interesting fragment of Spatial Logic over a class of (in general, infinite-control) pi-calculus processes. The fragment in question is expressive enough to capture interesting safety invariants. We achieve our goal in two steps. First, we devise a behavioural type system whose purpose is basically to extract a behavioural CCS type T out of given processes P. The type extracted this way is logically equivalent to the original process, in the sense that, for the considered logic, a formula is satisfied by P if and only if it is satisfied by T. This part of the work is based on behavioural types techniques similar to those discussed in [3].

Second, we show that it is actually decidable whether a CCS type satisfies a formula in the considered fragment. This part, which is largely independent from the first one, heavily relies on the notion of *well-structured transition systems* (WSTS) introduced by Finkel and Schnoebelen [15]. WSTS's are a general technique for proving decidability of reachability-related problems. Applying the technique requires two conditions to be satisfied. First, one has to endow the transition system at hand, in our case CCS's, with a *well-quasi order* (WQO) that be compatible with the transition system, i.e. be a simulation relation. Second, one has to show that certain sets of states, in our work those corresponding to denotations of logical formulae, have a *finite basis*.

In our case, the first condition is met by viewing type terms as labelled *forests* and endowing them with a preorder, which we name rooted tree embedding. Essentially, a forest $\mathcal{F}$ is greater than another forest $\mathcal{G}$ if $\mathcal{F}$ contains an isomorphic copy of $\mathcal{G}$. This preorder is different from the standard Kruskal's preorder often found in Graph Theory, which would not result in a compatible WQO for our model.

The second condition is met by showing that, for each formula, a finite set of terms (aka *finite basis*) is computable, whose upward closure wrt the quasi-order coincides with the formula's denotation. The definition of a finite basis is made nontrivial by the presence of the boolean connective $\wedge$, which requires some ingenuity.

In the end, we obtain decidability of the mentioned fragment of Spatial Logic over types. In this respect, our result generalizes a previous result by Busi et al. [5], who had proven decidability of *weak barbs* in CCS with replication, a very simple example of structural property expressible in our logic. As a corollary of the logical correspondence given by the type system, decidability of the considered logic carries over to well-typed pi-processes. As far as we know, our work is the first attempt at combining tree-theoretic reasoning and WSTS techniques in a process calculus setting.

It is worth to stress that, in the economy of the proof, being able to go from the pi-calculus to CCS via the behavioural type system is essential. In particular, the WSTS techniques do not apply directly to pi-calculus. The technical reason is that there is no upper bound on the run-time nesting depth of restrictions in pi-terms, a fact that prevents defining our syntax-based WQO directly on the pi-calculus. On the contrary, there is such a bound in CCS, and this allows one to view CCS terms as forests. A minimalistic view of the behavioural type system would be that it selects a set of pi-processes to whom the WSTS technique can be applied. In practice, though, the type system does more than that, since it computes CCS abstractions logically equivalent to the original processes, but much easier to manipulate.

Structure of the paper. In Section 2 we introduce the language of processes, a standard polyadic pi-calculus with guarded summations. Section 3 describes the type system and its main properties. In Section 4 we introduce *Shallow Logic*, the fragment of Spatial Logic we are interested in, and two classes of formulae: monotone and anti-monotone, the latter corresponding to safety properties. Logical equivalence of processes and types is also discussed. Section 5 presents some definitions and results on forests and WQO's, which will be useful in the rest of the paper. Section 6 discusses how to endow behavioural types with a WSTS structure. Decidability of (anti-)monotone formulae is proven in Section 7. Further and related work is discussed in Section 8. Some technical material has been confined to three separate appendices.

2. Processes

The language we consider is a synchronous polyadic pi-calculus [24] with guarded summations and replications. We presuppose a countable set of *names* $\mathcal{N}$ and let $a, b, \ldots, x, y, \ldots$ range over names. Processes $P, Q, R, \ldots$ are defined by the grammar below

$$\mathsf{t} ::= (\tilde{x} : \tilde{\mathsf{t}})T$$

$$\alpha ::= a(\tilde{b}) \;\big|\; \overline{a}\langle\tilde{b}\rangle \;\big|\; \tau \qquad P ::= \textstyle\sum_{i\in I} \alpha_i.P_i \;\big|\; P|P \;\big|\; (\nu b : \mathsf{t})P \;\big|\; !a(\tilde{b}).P$$

where $\tilde{b}$ is a tuple of distinct names and $\mathsf{t} = (\tilde{x} : \tilde{\mathsf{t}}')T$ is a *channel type*, with T a *process type* to be defined in Section 3 and $\tilde{x}$ a tuple of distinct names. In a channel type, $(\tilde{x} : \tilde{\mathsf{t}})$ is a binder with scope T (with $\tilde{x} \subseteq \mathrm{fn}(T)$); $\tilde{x}$ and $\tilde{\mathsf{t}}$ (with $\tilde{x} \cap \mathrm{fn}(\tilde{\mathsf{t}}) = \emptyset$) represent, respectively, the formal parameters and types of objects that can be passed along the channel; type T is a process type prescribing a usage of those parameters. The calculus is equipped with standard notions of free names $\mathrm{fn}(\cdot)$ and bound names $\mathrm{bn}(\cdot)$. We let $\mathrm{fn}((\nu b : \mathsf{t})P) = (\mathrm{fn}(P) \cup \mathrm{fn}(\mathsf{t})) \setminus \{b\}$ and that we identify terms up to alpha-equivalence, defined as usual. To prevent arity mismatch in synchronizations, from now on we will only consider well-sorted terms in some fixed sorting system (see e.g. [24]), and call $\mathcal{P}$ the resulting set of *processes*.

In the rest of the paper, we write $\mathbf{0}$ for the empty summation, often omit trailing $\mathbf{0}$'s and sometimes abbreviate $(\nu b_1 : \mathsf{t}_1) \cdots (\nu b_n : \mathsf{t}_n)P$ as $(\nu \tilde{b}_i : \tilde{\mathsf{t}}_i)_{i\in 1..n}P$, or simply $(\nu \tilde{b} : \tilde{\mathsf{t}})P$ or $(\nu \tilde{b})P$.

Over $\mathcal{P}$, we define a *reduction semantics*, based as usual on a notion of structural congruence and on a reduction relation. These relations are defined as the least congruence $\equiv$ and as the least relation $\rightarrow$ generated by the axioms in Table 1 and Table 2, respectively.

$$(\nu y:\mathtt{t})\mathbf{0}\equiv\mathbf{0} \qquad (P|Q)|R\equiv P|(Q|R) \qquad P|Q\equiv Q|P$$

$$P|\mathbf{0}\equiv P \qquad (\nu x:\mathtt{t})P|Q\equiv(\nu x:\mathtt{t})(P|Q) \text{ if } \tilde{x}\notin \mathrm{fn}(Q)$$

Table 1: Laws for structural congruence $\equiv$ on processes

$$(\textsc{com})\,\frac{\alpha_l=a(\tilde{x}) \quad \alpha'_n=\overline{a}\langle\tilde{b}\rangle \quad l\in I \quad n\in J}{\sum_{i\in I}\alpha_i.P_i|\sum_{j\in J}\alpha'_j.Q_j\to P_l[\tilde{b}/\tilde{x}]|Q_n} \qquad (\textsc{tau})\,\frac{j\in I \quad \alpha_j=\tau}{\sum_{i\in I}\alpha_i.P_i\to P_j}$$

$$(\textsc{rep})\,\frac{\alpha_n=\overline{a}\langle\tilde{b}\rangle \quad n\in J}{!a(\tilde{x}).P|\sum_{j\in J}\alpha_j.Q_j\to !a(\tilde{x}).P|P[\tilde{b}/\tilde{x}]|Q_n} \qquad (\textsc{par})\,\frac{P\to P'}{P|Q\to P'|Q}$$

$$(\textsc{struct})\,\frac{P\equiv Q \quad Q\to Q' \quad Q'\equiv P'}{P\to P'} \qquad (\textsc{res})\,\frac{P\to P'}{(\nu x:\mathtt{t})P\to(\nu x:\mathtt{t})P'}$$

Table 2: Rules for the reduction relation $\to$ on processes.

Concerning Table 1, note that we have dropped the law $(\nu x:\mathtt{t})(\nu y:\mathtt{t}')P\equiv(\nu y:\mathtt{t}')(\nu x:\mathtt{t})P$, which would allow one to swap restrictions: swapping $\mathtt{t}$ with $\mathtt{t}'$ would imply the capture of y in case y is used as a free name in $\mathtt{t}$. The rules in Table 2 are standard and do not deserve explanations.

In the sequel, we say that a process P has a *barb* a, written $P\searrow_a$, if $P\equiv(\nu\tilde{b})(\sum_i\alpha_i.P_i+a(\tilde{x}).Q|R)$ or $P\equiv(\nu\tilde{b})(!a(\tilde{x}).Q|R)$, with $a\notin\tilde{b}$. $P\searrow_{\overline{a}}$ is defined similarly. By $P\xrightarrow{\langle a\rangle}Q$ we denote a reduction $P\to Q$ arising from a synchronization on the channel name (subject) $a\in\mathrm{fn}(P)$.

3. Type System

This section introduces the type system and some basic results about it. As to the class of properties captured by the type system (Type Soundness), we postpone the discussion to the next section, where we introduce Shallow Logic. Most of the material in this section is adapted from [3]; however Proposition 2, which states the decidability of the type system, is new.

3.1. Types

Types are essentially CCS terms, carrying some extra annotation on input prefixes and restrictions. Let $\mathfrak{a}$, $\mathfrak{b}$, ... range over finite set of names. The set $\mathcal{T}$ of types is generated by the following grammar:

$$\mu ::= a^{\mathfrak{a}} \;\big|\; \overline{a} \;\big|\; \tau \qquad\qquad T,S,U ::= \textstyle\sum_{i\in I}\mu_i.T_i \;\big|\; !a^{\mathfrak{a}}.T \;\big|\; T|T \;\big|\; (\nu a^{\mathfrak{a}})T\,.$$

Notions of free and bound names (resp. $\mathrm{fn}(\cdot)$ and $\mathrm{bn}(\cdot)$), alpha-equivalence, structural congruence and reduction for types parallel those for processes and are not repeated here. Note that, in the case of channel types, we let $\mathrm{fn}((\tilde{x}:\tilde{\mathtt{t}})T) = (\mathrm{fn}(\tilde{\mathtt{t}}) \cup \mathrm{fn}(T)) \setminus \tilde{x}$ while in $a^{\mathfrak{a}}.T$ and $(\nu a^{\mathfrak{a}})T$, the annotations $\mathfrak{a}$ contribute to the set of free names of a type, indeed $\mathrm{fn}(a^{\mathfrak{a}}.S) = \{a\} \cup \mathfrak{a} \cup \mathrm{fn}(S)$. In the type system, annotations will be employed so as to ensure that each free name in a process is also free in the corresponding type, so that scope extrusion, hence structural congruence, works properly in P and T (See [3, Remark 3] for additional details; the annotations used there take a slightly different form but serve the same purpose.)

In the sequel, we shall often omit the channel type $()\mathbf{0}$, writing e.g. $(x)\overline{x}$ instead of $(x:()\mathbf{0})\overline{x}$ and annotations on input prefixes and restrictions when not relevant for the discussion. We will let $G,F,\ldots$ range over guarded summation and replications.

$$G,F \;::=\; \textstyle\sum_{i\in I}\,\mu_i.T_i \;\big|\; !a^{\mathfrak{a}}.T$$

Consider any $a \in \mathcal{N}$ and any term T. In the following, we abbreviate $a \notin \mathrm{fn}(T)$ as $a\#T$. This notation is extended to tuples of names $\tilde{a}$ as expected. Similarly, given two sets of names A and B, we write $A\#B$ to mean $A \cap B = \emptyset$.

3.2. Typing rules

The judgements of type system are of the form $\Gamma \vdash P : T$, where: $P \in \mathcal{P}$, $T \in \mathcal{T}$ and Γ is a *context*, that is, a finite partial map from names to channel types. We write $\Gamma \vdash a : \mathtt{t}$ if $a \in \mathrm{dom}(\Gamma)$ and $\Gamma(a) = \mathtt{t}$. We say that a context is *well-formed* if whenever $\Gamma \vdash a : (\tilde{x}:\tilde{\mathtt{t}})T$ then $\mathrm{fn}(T,\tilde{\mathtt{t}}) \subseteq \tilde{x} \cup \mathrm{dom}(\Gamma)$. In what follows *we shall only consider well-formed contexts*.

The type system can be thought of as a procedure that, given P, builds a CCS approximation T of P, with help from a context Γ prescribing channel usage. Like in [17, 3], rules for input and output are asymmetric, in the sense that, when typing a receiver $a(\tilde{x}).P$, the type information on P that depends on the input parameters $\tilde{x}$ is moved to the sender process (rule (T-OUT)). The intuitive reason is that the transmitted names $\tilde{b}$ are statically known only to the sender.

Accordingly, the type of the input continuation P is required to decompose, modulo $\equiv$, as $T|T'$, where T is the type prescribed by the context Γ for a, and T', which does not mention the input parameters $\tilde{x}$, is anything else. As a consequence, on the receiver's side (rule (T-INP)), one only keeps track of the part of the continuation type that does not depend on the input parameters. In essence, in well typed processes, all receivers on a

$$\text{(T-INP)}\quad \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathfrak{t}})T \quad \mathrm{fn}((\tilde{x} : \tilde{\mathfrak{t}})T) = \mathfrak{a} \qquad \Gamma, \tilde{x} : \tilde{\mathfrak{t}} \vdash P : T|T' \quad \tilde{x}\#T'}{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T'} \qquad \text{(T-REP)}\quad \frac{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T}{\Gamma \vdash !a(\tilde{x}).P : !a^{\mathfrak{a}}.T}$$

$$\text{(T-OUT)}\quad \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathfrak{t}})T \quad \Gamma \vdash \tilde{b} : \tilde{\mathfrak{t}} \quad \Gamma \vdash P : S}{\Gamma \vdash \overline{a}\langle \tilde{b} \rangle.P : \overline{a}.(T[\tilde{b}/\tilde{x}] \,|\, S)} \qquad \text{(T-TAU)}\quad \frac{\Gamma \vdash P : T}{\Gamma \vdash \tau.P : \tau.T}$$

$$\text{(T-SUM)}\quad \frac{|I| \neq 1 \quad \forall i \in I :\ \Gamma \vdash \alpha_i.P_i : \mu_i.T_i}{\Gamma \vdash \sum_{i \in I} \alpha_i.P_i : \sum_{i \in I} \mu_i.T_i} \qquad \text{(T-EQ)}\quad \frac{\Gamma \vdash P : T \quad T \equiv S}{\Gamma \vdash P : S}$$

$$\text{(T-RES)}\quad \frac{\Gamma, a : \mathfrak{t} \vdash P : T \quad \mathfrak{a} = \mathrm{fn}(\mathfrak{t})}{\Gamma \vdash (\nu a : \mathfrak{t})P : (\nu a^{\mathfrak{a}})T} \qquad \text{(T-PAR)}\quad \frac{\Gamma \vdash P : T \quad \Gamma \vdash Q : S}{\Gamma \vdash P|Q : T|S}$$

Table 3: Rules of the behavioural type system.

must share a common part that deals with the received names $\tilde{x}$ as prescribed by the type T. Finally, note that (T-EQ) is related to sub-typing. In order to guarantee preservation of the (shallow) spatial structure it is necessary to abandon preorders in favor of an equivalence relation that respects the structure of terms, i.e. structural congruence. In the following we say that a process P *is* Γ*-well-typed* if $\Gamma \vdash P : T$ for some $T \in \mathcal{T}$.

3.3. Results

This section presents the basic properties of the type system. Theorem 1 and 2 guarantee the reduction-based correspondence between processes and the types, while Proposition 1 guarantees the structural one. Note that the structural correspondence is shallow, in the sense that in general it breaks down underneath prefixes. Proofs are omitted and can be found in [3, Subsection 4.3 and Appendix D].

Theorem 1 (subject reduction). *$\Gamma \vdash P : T$ and $P \to P'$ implies that there exists a T' such that $T \to T'$ and $\Gamma \vdash P' : T'$.*

Theorem 2 (type subject reduction). *$\Gamma \vdash P : T$ and $T \to T'$ implies that there exists a P' such that $P \to P'$ and $\Gamma \vdash P' : T'$.*

Proposition 1 (structural correspondence). *Suppose $\Gamma \vdash P : T$.*

1. *$P \searrow_{\alpha}$, with $\alpha ::= a \;\big|\; \overline{a}$, implies $T \searrow_{\alpha}$; and vice-versa for T and P.*
2. *$P \equiv (\nu \tilde{a} : \tilde{\mathfrak{t}})R$ implies $T \equiv (\nu \tilde{a}^{\tilde{\mathfrak{a}}})S$, with $\tilde{\mathfrak{a}} = \mathrm{fn}(\tilde{\mathfrak{t}})$ and $\Gamma, \tilde{a} : \tilde{\mathfrak{t}} \vdash R : S$; and vice-versa for T and P.*
3. *$P \equiv P_1|P_2$ implies $T \equiv T_1|T_2$, with $\Gamma \vdash P_i : T_i$, for $i = 1, 2$; and vice-versa for T and P.*

We come now to discuss decidability of the type system. The actual proof is based on a type inference algorithm that, given Γ and P, computes a sort of symbolic most general type for P under Γ. A few preliminary definitions are in order.

Consider a countable set of type variables $\mathcal{V}$, ranged over by $X, \ldots$. We let E range over open type expressions, $\mathtt{c}$ range over constraints and $\mathcal{C}$ range over tuples of constraints, as defined below.

$$\begin{array}{rcl} E & ::= & \sum_{i\in I}\mu_i.E_i \;\big|\; !a^{\mathfrak{a}}.E \;\big|\; E|E \;\big|\; (\nu a^{\mathfrak{a}})E \;\big|\; X \\ \mathtt{c} & ::= & \langle E \equiv X|T, \tilde{y}\#X\rangle \\ \mathcal{C} & ::= & \mathtt{c}\cdot\mathcal{C} \;\big|\; (\mathcal{C}_i)_{i\in I} \end{array}$$

We will denote by $\emptyset$ the empty forest $(\mathcal{C}_i)_{i\in\emptyset}$.

Below, we give a syntax-driven definition of the inference algorithm $\mathrm{inf}(\cdot,\cdot)$. In the definition, we assume that all the bound names in Γ and P are distinct from one another and from the free names. In the clauses for parallel composition and summation, renaming of type variables is implicitly applied in order to ensure that each type variable occurs at most once in a constraint.

$$\begin{array}{rll} \mathrm{inf}(a(\tilde{y}).P,\Gamma) = & \big(a^{\mathfrak{a}}.X,\, \langle E\equiv X|T,\, \tilde{y}\#X\rangle\cdot\mathcal{C}\big), & (E,\mathcal{C}) = \mathrm{inf}(P,(\Gamma,\tilde{y}:\tilde{\mathtt{t}})),\ \mathfrak{a} = \mathrm{fn}(\tilde{\mathtt{t}},T)\setminus\tilde{y} \\ & & \quad\text{if } \Gamma\vdash a:(\tilde{y}:\tilde{\mathtt{t}})T \text{ and } X\#\mathcal{C} \\ \mathrm{inf}(\overline{a}\langle\tilde{b}\rangle.P,\Gamma) = & \big(\overline{a}.(E\,|\,T[\tilde{b}/\tilde{x}]),\mathcal{C}\big), & (E,\mathcal{C}) = \mathrm{inf}(P,\Gamma),\ \Gamma\vdash\tilde{b}:\tilde{\mathtt{t}} \\ & & \quad\text{if } \Gamma\vdash a:(\tilde{x}:\tilde{\mathtt{t}})T \\ \mathrm{inf}(\tau.P,\Gamma) = & \big(\tau.E,\mathcal{C}\big), & (E,\mathcal{C}) = \mathrm{inf}(P,\Gamma) \\ \mathrm{inf}(\sum_{i\in I}\alpha_i.P_i,\Gamma) = & \big(\sum_{i\in I}\mu_i.E_i,(\mathcal{C}_i)_{i\in I}\big), & (\mu_i.E_i,\mathcal{C}_i) = \mathrm{inf}(\alpha_i.P_i,\Gamma),\ \text{for each } i\in I,\ |I|\neq 1 \\ \mathrm{inf}(P_1|P_2,\Gamma) = & \big(E_1|E_2,(\mathcal{C}_1,\mathcal{C}_2)\big), & (E_i,\mathcal{C}_i) = \mathrm{inf}(P_i,\Gamma),\ \text{for } i=1,2 \\ \mathrm{inf}(!a(\tilde{x}).P,\Gamma) = & \big(!a^{\mathfrak{a}}.E,\mathcal{C}\big), & (a^{\mathfrak{a}}.E,\mathcal{C}) = \mathrm{inf}(a(\tilde{x}).P,\Gamma) \\ \mathrm{inf}((\nu a:\mathtt{t})P,\Gamma) = & \big((\nu a^{\mathfrak{a}})E,\mathcal{C}\big), & (E,\mathcal{C}) = \mathrm{inf}(P,(\Gamma,a:\mathtt{t})),\ \mathfrak{a} = \mathrm{fn}(\mathtt{t})\setminus a \end{array}$$

In order to solve the constraints $\mathcal{C}$, we make use of an auxiliary function $\mathrm{split}(\cdot)$, which takes a constraint $\mathtt{c}$ of the form $\langle T\equiv X|S,\tilde{y}\#X\rangle$ and returns an arbitrarily chosen type U such that $U|S\equiv T$ and $\tilde{y}\#U$, if it exists:

$$\mathrm{split}(\langle T\equiv X|S,\tilde{y}\#X\rangle) = \begin{cases} [U/X] & \text{with } U \text{ s.t. } U|S\equiv T \text{ and } \tilde{y}\#U \\ \textit{undefined} & \text{if such a } U \text{ does not exists.} \end{cases}$$

Lemma 1. $\mathrm{split}(\langle T\equiv X|S,\tilde{x}\#X\rangle)$ *is computable.*

PROOF: (Outline) First, we observe that $\equiv$, as defined in Table 1, is decidable: this can be proved by an argument similar to that used in the proof of decidability of $\equiv^{!\mathrm{fr}}$ in [12], where $\equiv^{!\mathrm{fr}}$ corresponds to $\equiv$ plus the rule $(\nu x)(\nu y)P\equiv(\nu y)(\nu x)P$.

Consider now the constraint $\langle T \equiv X|S, \tilde{x}\#X\rangle$ and assume that in T and S the bound names are pairwise distinct and distinct from the free names and names in $\tilde{x}$. The algorithm proceeds by distinguishing the following cases.

1. If $T \equiv S$ then $\text{split}(\langle T \equiv X|S, \tilde{x}\#X\rangle) = [^{\mathbf{0}}/_X]$.
2. If $S \equiv \mathbf{0}$ and $\tilde{x}\#T$ then $\text{split}(\langle T \equiv X|S, \tilde{x}\#X\rangle) = [^{T}/_X]$.
3. If (1) and (2) do not apply then let T' and S' be the types obtained by deleting from T and S any $\mathbf{0}$ component, at any nesting level (i.e. by repeatedly applying rule $P|\mathbf{0} \equiv P$ from left to right, until this is possible). Hence both $T' \equiv T$ and $S' \equiv S$. The algorithm proceeds by trying to find a substitution for X satisfying $T' \equiv X|S'$. Let
$$\Pi \stackrel{\triangle}{=} \{\langle T_1, T_2\rangle : T' \equiv^- T_1|T_2\},$$
where $\equiv^-$ denotes the congruence induced by the axioms of $\equiv$ except alpha-conversion and $P|\mathbf{0} \equiv P$. Note that Π is finite and can be built by applying $\equiv^-$ and reducing T' into a *web-form* (see [12]), a normal form identifying the maximal number of parallel components which a term can be decomposed into. If there exists a pair $\langle T_1, T_2\rangle \in \Pi$ such that $(S \equiv)S' \equiv T_1$ and $\tilde{x}\#T_2$ then $\text{split}(\langle T \equiv X|S, \tilde{x}\#X\rangle) = [^{T_2}/_X]$.
4. In case no one of the above cases applies, $\text{split}(\langle T \equiv X|S, \tilde{x}\#X\rangle)$ is undefined.

$\square$

The function $\text{solve}(\mathcal{C})$ we define below yields either a substitution σ from type variables to types satisfying $\mathcal{C}$ (written $\sigma \models \mathcal{C}$), if it exists, or is undefined otherwise. The constraints are solved by proceeding bottom-up. Formally, the function $\text{solve}(\cdot)$ is defined as follows (where $\emptyset$ is the empty substitution):

$$\text{solve}(\emptyset) = \emptyset$$

$$\text{solve}(\mathtt{c} \cdot \mathcal{C}) = \begin{cases} [^{U}/_X] \cup \sigma & \text{if } \sigma = \text{solve}(\mathcal{C}) \text{ and } \text{split}(\mathtt{c}\sigma) = [^{U}/_X] \\ \textit{undefined,} & \text{otherwise} \end{cases}$$

$$\text{solve}((\mathcal{C}_i)_{i\in I}) = \textstyle\bigcup_{i\in I} \sigma_i \qquad \text{if } \sigma_i = \text{solve}(\mathcal{C}_i).$$

As a consequence of Lemma 1 we get the following result.

Lemma 2. *Both* $\text{inf}(\cdot,\cdot)$ *and* $\text{solve}(\cdot)$ *are computable.*

Proposition 2. *Let* Γ *be a context. It is decidable whether* P *is* Γ*-well-typed. Moreover, if this is the case, it is possible to compute a type* T *such that* $\Gamma \vdash P : T$.

PROOF: The proof rests on the type inference algorithm defined above. It is sufficient to prove correctness and completeness of the algorithm, that is

1. $\mathrm{inf}(\Gamma, P) = (E, \mathcal{C})$ and $\mathrm{solve}(\mathcal{C}) = \sigma$ implies $\Gamma \vdash P : E\sigma$;
2. if $\Gamma \vdash P : T$ then there are $E, \mathcal{C}$ and σ s.t. $\mathrm{inf}(\Gamma, P) = (E, \mathcal{C})$, $\mathrm{solve}(\mathcal{C}) = \sigma$ and $E\sigma \equiv T$.

Statements (1) and (2) above, can be proven by relying on the existence of *normal derivations*. A *normal derivation*, written $\Gamma \vdash_{\mathrm{N}} P : T$, is a typing derivation where rule (T-EQ) can be applied only immediately before (T-INP) (see also Appendix D.1 of [3]). It is an easy matter to prove that for any process P and derivation $\Gamma \vdash P : S$ there exists a normal derivation $\Gamma \vdash_{\mathrm{N}} P : T$, for some $T \equiv S$. The proof of (1) is straightforward by induction on the last rule applied for deducing $\mathrm{inf}(P, \Gamma) = (E, \mathcal{C})$. Statement (2) can be proven by induction on the normal derivation $\Gamma \vdash_{\mathrm{N}} P : T$, where $S \equiv T$.

Lemma 2 shows that $\mathrm{inf}(\cdot, \cdot)$ is computable and that a solution for the inferred constraints can be computed. Therefore, by (1) it is possible to compute a type T such that $\Gamma \vdash P : T$. $\square$

4. Shallow Logic and Type Soundness

The logic for the pi-calculus presented below can be regarded as a fragment of Caires and Cardelli's Spatial Logic [9]. In [3] we have christened this fragment *Shallow Logic*, to emphasize the fact that it allows one to speak about the dynamic as well as the "shallow" spatial structure of processes and types. In particular, the logic does not provide for modalities that allow one to "look underneath" prefixes. Another important feature of this fragment is that the basic modalities focus on channel *subjects*, ignoring the object part at all. The selected mix of operators is sufficient to express a variety of interesting process properties (no race condition, unique receptiveness [26], deadlock freedom, to mention a few). In what follows we present the logic. Then discuss an undecidability result that motivates the introduction of a restricted fragment of the logic. We then introduce the logical correspondence between types and processes.

4.1. Definitions

The set $\mathcal{L}$ of *Shallow Logic* formulae $\phi, \psi, \ldots$ is given by the following syntax, where $a \in \mathcal{N}$:

$$\phi ::= \mathbf{T} \mid a \mid \overline{a} \mid \phi|\phi \mid \neg\phi \mid \mathrm{H}^*\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid \Diamond^* \phi \ .$$

The set of logical operators includes spatial $(a, \overline{a}, |, \mathrm{H}^*)$ as well as dynamic $(\langle a \rangle, \Diamond^*)$ connectives[1], beside the usual boolean connectives, including a constant $\mathbf{T}$ for "true". We

[1]The version of the Shallow Logic considered in [3] features slightly more general versions of the eventuality modality. Here we have left these operators out to simplify the presentation, however they can be accommodated with only cosmetic changes.

$$\begin{array}{rlrl}
[\![\mathbf{T}]\!] = \mathcal{U} & & [\![\mathrm{H}^*\phi]\!] = & \left\{A \,\middle|\, \exists\tilde{a}, B : A \equiv (\nu\tilde{a})B,\ \tilde{a}\#\mathrm{n}(\phi),\ B \in [\![\phi]\!]\right\} \\
[\![\phi_1 \vee \phi_2]\!] = [\![\phi_1]\!] \cup [\![\phi_2]\!] & & [\![\phi_1 \wedge \phi_2]\!] = & [\![\phi_1]\!] \cap [\![\phi_2]\!] \\
[\![\neg\phi]\!] = \mathcal{U} \setminus [\![\phi]\!] & & [\![\langle a\rangle\phi]\!] = & \left\{A \,\middle|\, \exists B : A \xrightarrow{\langle a\rangle} B,\ B \in [\![\phi]\!]\right\} \\
[\![a]\!] = \left\{A \,\middle|\, A \searrow_a \right\} & & [\![\phi_1|\phi_2]\!] = & \left\{A \,\middle|\, \exists A_1, A_2 : A \equiv A_1|A_2,\ A_1 \in [\![\phi_1]\!],\ A_2 \in [\![\phi_2]\!]\right\} \\
[\![\overline{a}]\!] = \left\{A \,\middle|\, A \searrow_{\overline{a}} \right\} & & [\![\Diamond^*\phi]\!] = & \left\{A \,\middle|\, \exists B : A \to^* B,\ \text{and } B \in [\![\phi]\!]\right\}
\end{array}$$

Table 4: Interpretation of formulae.

have included both disjunction and conjunction to present more smoothly "monotone" properties, that is, properties that are preserved when "adding structure" to terms. The set of names of a formula ϕ, written $\mathrm{n}(\phi)$, is defined as expected. The interpretation of $\mathcal{L}$ over processes and types is given in Table 4. We let $\mathcal{U}$ be the set including all processes and all types. We write $A \models \phi$ if $A \in [\![\phi]\!]$, where $A \in \mathcal{U}$. Similarly, given any sequence $\tilde{A} \in \mathcal{U}^*$ we write $\tilde{A} \models \phi$ if $\tilde{A} = (A_1, \ldots, A_n)$ and $A_1|\cdots|A_n \models \phi$. Connectives are interpreted in the standard manner. In particular, concerning spatial modalities, the barb atom a (resp. $\overline{a}$) requires that A has an input (resp. output) barb on a; $\phi|\psi$ requires that A can be split into two parallel components satisfying ϕ and ψ; $\mathrm{H}^*\phi$ requires that A satisfies ϕ, up to some top level restrictions. Concerning the dynamic part, formula $\langle a\rangle\phi$ checks if an interaction with subject a may lead A to a state where ϕ is satisfied; $\Diamond^*\phi$ checks if any number, including zero, of reductions may lead A to a state where ϕ is satisfied.

In this paper, we shall mainly focus on safety properties, that is, properties of the form "nothing bad will ever happen". The following definition is useful to syntactically identify classes of formulae that correspond to safety properties.

Definition 1 (monotone and anti-monotone formulae). *We say a formula* ϕ *is* monotone *if it does not contain occurrences of* $\neg$ *and* anti-monotone *if it is of the form* $\neg\psi$*, with* ψ *monotone.*

Safety invariants can often be written as anti-monotone formulae $\neg\Diamond^*\psi$ with ψ a monotone formula representing the bad event one does not want to occur.

Example 1. The following formulae define properties depending on generic names, a and l.

$$\begin{array}{rrcl}
\text{No race condition:} & \mathit{NoRace}(a) & \stackrel{\triangle}{=} & \neg\Diamond^*\mathrm{H}^*(\overline{a}|\overline{a}) \\
\text{Linearity:} & \mathit{Linear}(a) & \stackrel{\triangle}{=} & \neg\Diamond^*\langle a\rangle\Diamond^*\langle a\rangle \\
\text{Lock:} & \mathit{Lock}(a,l) & \stackrel{\triangle}{=} & \neg\Diamond^*\mathrm{H}^*(l|\langle a\rangle)
\end{array}$$

$\mathit{NoRace}(a)$ says that it will never be the case that there are two concurrent outputs competing for synchronization on a. $\mathit{Linear}(a)$ says that it is not the case that a is used twice in a computation. In $Lock(a,l)$, a represents a shared resource and l a lock: the formula says that it is never the case that the resource is acquired in the presence of the lock, that is, without prior acquisition of the lock.

The driving motivation leading us to identify the classes of monotone and anti-monotone formulae and to focus on safety properties is that satisfiability of formulae like $\Diamond^*(a \wedge \neg b)$ – that are neither monotone nor anti-monotone – is not decidable even in the case of CCS.

Theorem 3. *Let a and b be any two distinct names. The problem of deciding whether* $T \models \Diamond^*(a \wedge \neg b)$ *is undecidable.*

PROOF: (Outline) Undecidability of the logic can be proven by reduction from the (undecidable) termination problem for Random Access Machines (RAMs) [27]. More precisely, one defines a nondeterministic encoding of RAMs into CCS. The encoding is not entirely faithful, in that translation of a decrement operation can perform the jump even in case the register is not empty, as follows.

$$
\begin{aligned}
[\![i : Succ(r_j)]\!] &: !p_i.(\overline{inc_j} \mid ack.\overline{p_{i+1}}) \\
[\![i : DecJump(r_j,s)]\!] &: !p_i.(\overline{dec_j} \mid (ack.\overline{p_{i+1}} \; + \; jmp.ack.\overline{p_s})) \\
[\![r_j]\!] &: !nr_j.(\nu\, m,u) \\
&\quad \big(\overline{m} \mid !m.(\overline{ack} \mid inc_j.(\overline{m} \mid \overline{u} + wrong) + dec_j.(u.\overline{m} + \overline{jmp}.\overline{nr_j}))\big)
\end{aligned}
$$

The execution of these wrong jumps can be detected at the end of the computation by checking the presence of the barb *wrong*. Thus, if a computation reaches a state with invocation to instruction i_{n+1} (indicating the terminating instruction) but without the barb *wrong*, we can conclude that the modeled RAM terminates. In this way we have reduced RAM termination to the problem of checking whether the process modeling the RAM satisfies the formula $\Diamond^*(i_{n+1} \wedge \neg wrong)$. From this undecidability of the logic follows (see [1, Appendix A] for more details). □

4.2. Type Soundness

The following theorem is crucial: it basically asserts that, under a condition of well-typing, model checking on processes can be reduced to model checking on types. The proof is based on the structural and operational correspondences seen in Section 3 and can be found in [3, Section 5].

Theorem 4 (type-process correspondence). *Suppose* $\Gamma \vdash P : T$. *Let* ϕ *be any formula. Then* $P \models \phi$ *if and only if* $T \models \phi$.

$$(\overline{a}.T)\downarrow_{\tilde{x}} = \begin{cases} \tau.(T\downarrow_{\tilde{x}}) & \text{if } a \notin \tilde{x} \\ \overline{a}.(T\downarrow_{\tilde{x}}) & \text{otherwise} \end{cases} \qquad (a^{\mathfrak{a}}.T)\downarrow_{\tilde{x}} = \begin{cases} \tau^{\mathfrak{a}\cap\tilde{x}}.(T\downarrow_{\tilde{x}}) & \text{if } a \notin \tilde{x} \\ a^{\mathfrak{a}\cap\tilde{x}}.(T\downarrow_{\tilde{x}}) & \text{otherwise} \end{cases}$$

$$(T_1|T_2)\downarrow_{\tilde{x}} = (T_1\downarrow_{\tilde{x}})|(T_2\downarrow_{\tilde{x}}) \qquad (\tau.T)\downarrow_{\tilde{x}} = \tau.(T\downarrow_{\tilde{x}}) \qquad ((\nu\tilde{b}^{\mathfrak{b}})T)\downarrow_{\tilde{x}} = (\nu\tilde{b}^{\mathfrak{b}\cap\{\tilde{x},\tilde{b}\}})(T\downarrow_{\tilde{x},\tilde{b}})$$

$$(\textstyle\sum_i \mu_i.T_i)\downarrow_{\tilde{x}} = \sum_i \big((\mu_i.T_i)\downarrow_{\tilde{x}}\big) \qquad (!a^{\mathfrak{a}}.T)\downarrow_{\tilde{x}} = !\big((a^{\mathfrak{a}}.T)\downarrow_{\tilde{x}}\big) \qquad ((\tilde{y}:\tilde{\mathfrak{t}})T)\downarrow_{\tilde{x}} = (\tilde{y}:\tilde{\mathfrak{t}}\downarrow_{\tilde{x},\tilde{y}})T\downarrow_{\tilde{x},\tilde{y}}$$

Table 5: $T\downarrow_{\tilde{x}}$.

The correspondence given by the previous theorem can be enhanced by the next result (the technical development in the rest of the paper does not depend on it, though). This result says that, under certain conditions, model checking can be safely carried out against a more abstract version of the type T, with a further potential gain in efficiency. This more abstract version is obtained by "masking" the free names of the type that are not found in the formula. Moreover, if this masking produces a top-level sub-term with no free names, this term can be safely discarded. More precisely, for any type T and set of names $\tilde{x}$, we let $T\downarrow_{\tilde{x}}$ denote the type obtained by replacing each annotation $(\cdot)^{\mathfrak{a}}$ with the annotation $(\cdot)^{\mathfrak{a}\cap\tilde{x}}$ and each *free* occurrence of a prefix $a.$ or $\overline{a}.$, for $a \notin \tilde{x}$, with the prefix $\tau.$. The formal definition can be found in Table 5. In this definition we assume as usual that all bound names in T are distinct from each other and disjoint from the set of free names and from $\tilde{x}$.

For instance, we have that

$$(\nu a^{\mathfrak{a}})(a^{\mathfrak{a}}.b^{\mathfrak{b}}|a^{\mathfrak{a}}.c^{\mathfrak{c}}|\overline{c}|\overline{a})\downarrow_b = (\nu a^{\mathfrak{a}\cap\{a,b\}})(a^{\mathfrak{a}\cap\{a,b\}}.b^{\mathfrak{b}\cap\{a,b\}}|a^{\mathfrak{a}\cap\{a,b\}}.\tau^{\mathfrak{c}\cap\{a,b\}}|\tau|\overline{a})\,.$$

Notice that terms produced by the hiding operator are in general not in $\mathcal{T}$. Consider e.g. the $\tau^{\mathfrak{c}\cap\{b\}}$ prefix above or $(!a^{\mathfrak{a}}.\overline{c})\downarrow_c = !\tau^{\mathfrak{a}\cap\{c\}}.\overline{c}$. More precisely, $T\downarrow_{\tilde{x}}$ belongs to the set of terms defined by the syntax of types extended as follows:

$$T \ ::= \ \cdots \ \big| \ !\tau^{\mathfrak{a}}.T \ \big| \ \tau^{\mathfrak{a}}.T\ .$$

Again, the proof of the proposition below can be found in [3, Section 5 and Appendix A].

Proposition 3. *(a) Suppose* $\Gamma \vdash P : T$ *and let* ϕ *be an anti-monotone formula with* $\mathsf{n}(\phi) \subseteq \tilde{x}$. *Then* $T\downarrow_{\tilde{x}} \models \phi$ *implies that* $T \models \phi$. *(b) Suppose* $\mathrm{fn}(U) = \emptyset$. *Then, for any* T *and* ϕ, $T|U \models \phi$ *if and only if* $T \models \phi$.

As shown in the following example, in some cases Proposition 3 *(b)* allows one to safely discard subterms of T possibly producing an infinite behaviour. This is clearly useful in the model checking phase.

Example 2. Consider the formula $NoRace(a)$ introduced in Example 1 and the process

$$P = \overline{b}\langle a\rangle + \overline{a} \,|\, b(x).(\nu c)(\overline{c} \,|\, !c.\overline{x}.\overline{c}) \,|\, !a.\overline{f} \,|\, !f.\overline{n}\,.$$

Here, at runtime, the number of occurrences of $\overline{n}$ "counts" the number of interactions performed on a. For a suitable Γ, one finds $\Gamma \vdash P : T$, where (ignoring annotations)

$$T = \overline{b}.(\nu c)(\overline{c} \,|\, !c.\overline{a}.\overline{c}) + \overline{a} \,|\, b \,|\, !a.\overline{f} \,|\, !f.\overline{n}\,.$$

It can be easily seen that

$$T \downarrow_a = (\overline{b}.(\nu c)(\overline{c} \,|\, !c.\overline{a}.\overline{c}) + \overline{a} \,|\, b \,|\, !a.\overline{f} \,|\, !f.\overline{n}) \downarrow_a = \tau.(\nu c)(\overline{c} \,|\, !c.\overline{a}.\overline{c}) + \overline{a} \,|\, \tau \,|\, !a.\tau \,|\, !\tau.\tau\,.$$

Now, $\big(\tau.(\nu c)(\overline{c} \,|\, !c.\overline{a}.\overline{c}) + \overline{a}\big) \,|\, !a.\tau \models NoRace(a)$, and, by Proposition 3 and Theorem 4, $P \models NoRace(a)$.

5. Ordered forests

In this section we will introduce some definitions and prove results concerning trees and forests, which will be useful in the rest of the paper.

5.1. Definitions

Let L be a non-empty set ranged over by $\ell, \ell', \ldots$. We define *ordered forests* $\mathcal{F}, \mathcal{G}, \ldots$ with labels in L (from now on, simply *forests*) to be the set of objects inductively defined as follows:

(i) the empty sequence ε is a forest;

(ii) if $\mathcal{F}_1, \ldots, \mathcal{F}_k$ are forests ($k \geq 0$) then the sequence $(\ell_1, \mathcal{F}_1) \cdots (\ell_k, \mathcal{F}_k)$, with $\ell_i \in L$, is a forest with roots $\ell_1, \ldots, \ell_k$.

A forest of the form $(\ell, \mathcal{F})$ is called an *(ordered) tree* with root ℓ. A tree of the form (ℓ, ε) is called a *leaf*.

It is useful to introduce a syntax to denote forests and forest contexts. We presuppose a countable set of forest variables ranged over by $X_1, X_2, \ldots$. Let $\mathfrak{L}$ be the set of terms generated by the following grammar:

$$C, C' ::= X \;\big|\; \mathbb{0} \;\big|\; \ell(C) \;\big|\; C|C'$$

where $\ell \in L$.

Any term $C \in \mathfrak{L}$ describes a forest context on the set of labels L where one interprets composition $|$ as concatenation of forests. Formally, each C is mapped to a forest context $\mathcal{F}_C$ as follows:

$$\mathcal{F}_{\mathbb{0}} = \varepsilon \qquad \mathcal{F}_X = (X, \varepsilon) \qquad \mathcal{F}_{\ell(C)} = (\ell, \mathcal{F}_C) \qquad \mathcal{F}_{C_1 |\, C_2} = \mathcal{F}_{C_1} \cdot \mathcal{F}_{C_2} \, .$$

Note that open terms represent *contexts*, with variables acting as the "holes". In what follows, we will let C range over possibly open terms, reserving the letters $t, s, \dots$ for the subset of closed terms. We will often use the familiar pictorial representation of trees and forests, as illustrated in the following example.

Example 3 (pictorial representation of forests). *Let $A = \{a, b\}$ and consider the term $C = a(b|a(a(X)))|b(a(b))$, where b abbreviates $b(\mathbb{0})$. The corresponding forest context $\mathcal{F}_C$ is depicted below.*

$$\begin{tikzcd}
& a \arrow[dl] \arrow[dr] & & b \arrow[d] \\
b & & a \arrow[d] & a \arrow[d] \\
& & a \arrow[d] & b \\
& & X &
\end{tikzcd}$$

Via the correspondence $C - \mathcal{F}_C$, we can identify the term C with the forest context $\mathcal{F}_C$; in what follows we shall not notationally distinguish between the two.

We introduce now some auxiliary notations for forests. The sequence of leaves $\tilde{\ell} = (\ell_1, \dots, \ell_n)$ obtained by visiting $\mathcal{F}$ in depth first order, aka the *frontier* of $\mathcal{F}$, is denoted by $\partial\mathcal{F}$. Similarly, the sequence of roots $(\ell_1, \dots, \ell_n)$ occurring in $\mathcal{F}$, from left to right, is denoted by $\rho\mathcal{F}$. By a slight overload of notation, we will sometimes denote by $\partial\mathcal{F}$ also the set and the multiset of leaves of $\mathcal{F}$: in each case, it should be clear from the context wether $\partial\mathcal{F}$ denotes a sequence, a set or a multiset. The same convention applies to $\rho\mathcal{F}$. We will use both a ground and an open version of the function ∂, denoted respectively ∂^g and ∂^o, yielding the sequence of *ground* (non-variables) and open (variables) leaves. In the example above: $\partial C = (b, X, b)$, $\partial^g C = (b, b)$, $\partial^o C = (X)$ and $\rho C = (a, b)$. We will sometimes write $C[\tilde{X}]$ to indicate that $\partial^o C = \tilde{X} = (X_1, \dots, X_k)$. In this case, taken $\tilde{t} = (t_1, ..., t_k)$, we will denote by $C[\tilde{t}]$ the term obtained by textually replacing each X_i with t_i in $C[\tilde{X}]$.

The *height* of a forest $\mathcal{F}$, written $\mathrm{h}(\mathcal{F})$, is defined as the maximal length of a path from a root to a leaf, as expected; the height of a leaf is 0 while the height of the empty forest is not defined. We will denote as $\mathfrak{F}^{(n)}$ the set of all forests of height at most n.

5.2. A well-quasi order on forests

The next step is to introduce a well-quasi order over forests. We start by recalling some background definitions and results.

Definition 2 (WQO). *Let S be a set. A* quasi-ordering *(QO, aka* preorder*) on S is a reflexive and transitive binary relation over S. A QO $\preceq$ on S is a* well quasi-ordering *(WQO for short) if for any infinite sequence of elements of S, $(s_i)_{i\geq 0}$, there exist i and j, with $i<j$, such that $s_i \preceq s_j$.*

It can be easily seen that the condition on sequence $(s_i)_{i\geq 0}$ in the definition of WQO above can be equivalently reformulated as follows: for any i there exists an infinite sequence of indices $j_0, j_1, \ldots$ such that $i < j_k$ and $s_i \preceq s_{j_k}$, for each $k \geq 0$.

The following definition provides us with a WQO $\preceq$ on forests called *rooted tree embedding*. One can think of this WQO as saying that $\mathcal{F}_1 \preceq \mathcal{F}_2$ if $\mathcal{F}_2$ contains an isomorphic copy of $\mathcal{F}_1$, preserving the roots of $\mathcal{F}_1$. It is worth to note that the embedding considered here is stronger than the one considered by Kruskal in his tree theorem [20] (see Remark 1).

Definition 3 (rooted-tree embedding). *Let $\mathfrak{F}$ be the set of all forests with labels in a nonempty set L. The* rooted-tree embedding *$\preceq$ over $\mathfrak{F}$ is inductively defined as follows:*

$$(\ell_1, \mathcal{F}_1)\cdots(\ell_k, \mathcal{F}_k) \preceq (\ell'_1, \mathcal{G}_1)\cdots(\ell'_h, \mathcal{G}_h)$$

iff there are distinct indices $1 \leq i_1 < \cdots < i_k \leq h$ s.t. for each j, $1 \leq j \leq k$, $\ell_j = \ell'_{i_j}$ and $\mathcal{F}_j \preceq \mathcal{G}_{i_j}$.

Example 4 (embedding). *Let $L = \{a, b\}$. It is easy to see that $a(b(a)|a)|b(a|b) \preceq a(b(a)|a(b))|b(a|a(b)|b)$. Pictorially:*

$$\begin{tikzcd}
 & a \arrow[dl] \arrow[dr] & & & b \arrow[dl] \arrow[dr] & & & & a \arrow[dl] \arrow[dr] & & & b \arrow[dl] \arrow[d] \arrow[dr] & \\
b \arrow[d] & & a & a & & b & \preceq & b \arrow[d] & & a \arrow[d] & a & a \arrow[d] & b & . \\
a & & & & & & & a & & b & & b &
\end{tikzcd}$$

In Theorem 5 below, we prove that the rooted tree embedding is a WQO, by relying on Higman's Lemma. In what follows, given any set S, we will denote by S^* the set of finite sequences of elements of S.

Lemma 3 (Higman's Lemma [16]). *Let S be a finite set and $\preceq$ a WQO over S. Define $\preceq^*$ over S^* as follows: for any $u = u_1 u_2 \cdots u_n$ and $v = v_1 v_2 \cdots v_m$ in S^*, with $1 \leq n \leq m$, $u \preceq^* v$ if and only if there are $j_1, \cdots, j_n$ such that $1 \leq j_1 \leq \cdots \leq j_n \leq m$ and $u_i \preceq v_{j_i}$ for each $1 \leq i \leq n$. Then the relation $\preceq^*$ is a WQO over S^*.*

Theorem 5. *Let $k \geq 0$, L be a finite set of labels and let $\mathfrak{F}^{(k)}$ be the subset of all forests of height at most k with labels in L. Then $(\mathfrak{F}^{(k)}, \preceq)$ is a* WQO.

PROOF: We proceed by induction on k.

The case $k = 0$ follows from Lemma 3.

Assume $k > 0$ and consider any infinite sequence $\mathcal{F}_1, \mathcal{F}_2, \ldots \in \mathfrak{F}^{(k)}$. Fix a generic $i_0 \geq 1$ and let $\mathcal{F}_{i_0} = (\ell_1, \mathcal{G}_1) \cdots (\ell_n, \mathcal{G}_n)$. Therefore, $\rho \mathcal{F}_{i_0} = (\ell_1, \cdots, \ell_n)$. We show that there exists a $j > i_0$ such that $\mathcal{F}_{i_0} \preceq \mathcal{F}_j$.

Consider the infinite sequence of elements in $\mathfrak{F}^{(0)}$ (sequences of leaves):

$$\rho \mathcal{F}_1, \rho \mathcal{F}_2, \ldots .$$

Let $I_0 = \{(j, f) \mid j > i_0 \text{ and } \rho \mathcal{F}_{i_0} \preceq_f \rho \mathcal{F}_j\}$, where $(\ell_1, \cdots, \ell_n) \preceq_f (\ell'_1, \cdots, \ell'_h)$ means that $\ell_i = \ell'_{f(i)}$, with $f : \{1, \ldots, n\} \to \{1, \ldots, h\}$ a monotone and injective total function. Given that $\mathfrak{F}^{(0)}$ is a WQO (Lemma 3, by taking $=$ as the underlying preorder) we have that $\Pi_1(I_0)$, the projection of I_0 on the first coordinate, is an infinite set.

In what follows, for any pair $(j, f) \in I_0$ with $\mathcal{F}_j = (\ell'_1, \mathcal{H}_1) \cdots (\ell'_h, \mathcal{H}_h)$, we let $\mathcal{F}_{j,f(m)}$ stand for $\mathcal{H}_{f(m)}$. Consider the sets

$$\begin{array}{rcl} I_1 & = & \{(j,f) \in I_0 \mid \mathcal{G}_1 \preceq \mathcal{F}_{j,f(1)}\} \\ I_2 & = & \{(j,f) \in I_1 \mid \mathcal{G}_2 \preceq \mathcal{F}_{j,f(2)}\} \\ & \vdots & \\ I_n & = & \{(j,f) \in I_{k-1} \mid \mathcal{G}_n \preceq \mathcal{F}_{j,f(n)}\} . \end{array}$$

Again, we get that $\Pi_1(I_i) \subseteq \mathfrak{F}^{(k-1)}$, for $i = 1, \ldots, n$, above is infinite, by applying the induction hypothesis.

Now, take any $(j, f) \in I_n$ and let $\mathcal{F}_j = (\ell'_1, \mathcal{H}_1) \cdots (\ell'_h, \mathcal{H}_h)$. By construction, it holds that $h \geq n$ and that for any $i = 1, \ldots, n$

$$\ell_i = \ell'_{f(i)} \text{ and } \mathcal{G}_i \preceq \mathcal{H}_{f(i)}$$

hence $\mathcal{F}_{i_0} \preceq \mathcal{F}_j$. □

Remark 1 (on bounded height and Kruskal's preorder). The condition of bounded height in Theorem 5 is necessary to make $\preceq$ a WQO. Indeed, if we drop this condition, it is easy to build an infinite sequence of trees violating the condition of WQO. This is illustrated by the following sequence of trees on the labels $L = \{a, b\}$.

$$\begin{tikzcd}
b & a \arrow[d] & a \arrow[d] & a \arrow[d] \\
 & b & a \arrow[d] & a \arrow[d] \\
 & & b & \vdots \arrow[d] \\
 & & & b
\end{tikzcd}$$

This sort of situations does not arise when considering Kruskal's preorder [20]. In the case of a finite alphabet L, Kruskal's preorder, $\preceq^{\mathrm{kr}}$, can be defined on trees inductively as follows:

1. $a \preceq^{\mathrm{kr}} a(t_1,\ldots,t_n)$, for any $a \in L$;
2. $s \preceq^{\mathrm{kr}} a(\ldots,t,\ldots)$ if $s \preceq^{\mathrm{kr}} t$;
3. $a(s_1,\ldots,s_m) \preceq^{\mathrm{kr}} a(t_1,\ldots,t_n)$ if $m \leq n$ and there exists $j_1,\ldots,j_m$ such that $1 \leq j_1 < j_2 < \cdots < j_m \leq n$ and $s_i \preceq^{\mathrm{kr}} t_{j_i}$, for $i = 1,\ldots,m$.

Thanks to condition 2, the sequence of trees above is strictly increasing wrt $\preceq^{\mathrm{kr}}$. This preorder would not be a good choice for our proposal, though, as discussed later in Remark 4.

5.3. *Other results on forests*

In the remainder of the section we give some properties of the rooted tree embedding that will prove useful in later sections. Some additional notation is in order. A *path* in a forest $\mathcal{F}$ as a sequence of natural numbers that uniquely identifies a node in the forest. As an example, let $L = \{a,b,c\}$ and consider the forest depicted below. The sequence 122 identifies the path characterized by dashed arrows, hence the node labelled c.

$$\begin{tikzcd}
 & a \arrow[dl] \arrow[dr, dashed] & & & & a \arrow[d] & \\
a & & b \arrow[dl] \arrow[dr, dashed] & & & b \arrow[dl] \arrow[dr] & \\
 & b & & c & b & & a
\end{tikzcd}$$

Given any forest $\mathcal{F}$, the set of all its paths, written $\mathrm{path}(\mathcal{F}) \subseteq (\mathbb{N} \setminus \{0\})^*$, is defined as follows

$$\mathrm{path}(\varepsilon) = \emptyset$$

$$\mathrm{path}((\ell_1,\mathcal{F}_1),\ldots,(\ell_n,\mathcal{F}_n)) = \{1,\ldots,n\} \cup \{1\} \cdot \mathrm{path}(\mathcal{F}_1) \cup \cdots \cup \{n\} \cdot \mathrm{path}(\mathcal{F}_n)\,.$$

Note that the set $\mathrm{path}(\cdot)$ is prefix closed. In the example above we have $\mathrm{path}(\mathcal{F}) = \{1,11,12,121,122,2,21,211,212\}$ and, as an example, 122 corresponds to the (only) node labelled c.

In the following we will write $\mathcal{F}_{|I}$, with I a prefix-closedset of paths, for the forest obtained from $\mathcal{F}$ by erasing each path and node not belonging to I. E.g. consider the forest $\mathcal{F}$ depicted above and let $I = \{1, 12, 121, 21, 212\}$, then $\mathcal{F}_{|I}$ is:

$$\begin{tikzcd}
a \arrow[dr] & & a \arrow[d] & \\
& b \arrow[dl] & b \arrow[dr] & \\
b & & & a
\end{tikzcd}$$

The following lemma gives an alternative characterization of rooted tree embedding: it explicitly conveys the idea that $\mathcal{F} \preceq \mathcal{G}$ means that $\mathcal{G}$ contains an isomorphic copy of $\mathcal{F}$.

Lemma 4. *$\mathcal{F} \preceq \mathcal{G}$ if and only if there is an injective $f : \mathrm{path}(\mathcal{F}) \to \mathrm{path}(\mathcal{G})$ such that $\mathcal{G}_{|\mathrm{range}(f)} = \mathcal{F}$.*

PROOF: ($\Rightarrow$). Let $\mathcal{F} = (\ell_1, \mathcal{F}_1) \cdots (\ell_k, \mathcal{F}_k)$ and $\mathcal{G} = (\ell'_1, \mathcal{G}_1) \cdots (\ell'_h, \mathcal{G}_h)$. By Definition 3, $\mathcal{F} \preceq \mathcal{G}$ implies that there are distinct indices $1 \le i_1 < \cdots < i_k \le h$ such that for each j, $1 \le j \le k$, $\ell_j = \ell'_{i_j}$ and $\mathcal{F}_j \preceq \mathcal{G}_{i_j}$.

By applying the induction hypothesis to each $\mathcal{F}_j \preceq \mathcal{G}_{i_j}$, with $1 \le j \le k$, we get that there exists $f_j : \mathrm{dom}(\mathcal{F}_j) \to \mathrm{dom}(\mathcal{G}_{i_j})$ such that $\mathcal{G}_{i_j|\mathrm{range}(f_j)} = \mathcal{F}_j$.

For any $\gamma = j \cdot \gamma' \in \mathrm{path}(\mathcal{F})$, define $f : \mathrm{path}(\mathcal{F}) \to \mathrm{path}(\mathcal{G})$ as follows:

$$f(\gamma) \stackrel{\triangle}{=} i_j \cdot f_j(\gamma') \,.$$

Hence $\mathcal{G}_{|\mathrm{range}(f)} = (\ell'_{i_1}, \mathcal{G}_{i_1|\mathrm{range}(f_1)}) \cdots (\ell'_{i_k}, \mathcal{G}_{i_k|\mathrm{range}(f_k)}) = (\ell_1, \mathcal{F}_1) \cdots (\ell_k, \mathcal{F}_k)$, as expected.
($\Leftarrow$). The vice-versa is obvious. $\square$

In the following, we introduce a couple of results on forest contexts that will be useful when proving the decidability of the logic. The first one guarantees that given two contexts related by $\preceq$, one can always find a new context in between the two, having as leaves all the holes of the bigger one and all the ground leaves of the smaller one.

Lemma 5. *Let $C, C' \in \mathfrak{L}$. If $C' \preceq C$ then there is $C'' \in \mathfrak{L}$ such that $C' \preceq C'' \preceq C$, $\partial^o C'' = \partial^o C$ and $\partial^g C'' = \partial^g C'$.*

PROOF: From $C' \preceq C$ we know that a copy of C' is embedded into C, in the sense of Lemma 4. Let $X_i \in \partial^o C \setminus \partial^o C'$. Consider the path γ joining a root of C to X_i. Let n be the last node from the root in the path γ which is shared between C and C', if any such node exists, otherwise let n be the root in the path. Build a new context C_* by extending C' with the new path from n to X_i: this may possibly require inserting a new tree into C_*

if n does not belong to C'. By construction, the new context C_* thus obtained is such that $C' \preceq C_* \preceq C$, and $\partial^o C_* = \partial^o C' \cup \{X_i\}$. Repeat this construction for another $X_j \in \partial^o C \setminus \partial^o C_*$, and continue until the wanted C'' is obtained. □

The following lemma states some basic properties of context filling and of the preorder $\preceq$. Its proof follows easily from Lemma 4 and is omitted.

Lemma 6. 1. *$C[\tilde{X}] \preceq C'[\tilde{X}]$ implies $C[\tilde{t}] \preceq C'[\tilde{t}]$.*
2. *$t \preceq C[\tilde{s}]$ implies that there are $C', \tilde{t}'$ such that $t = C'[\tilde{t}']$ with $C' \preceq C$ and $\tilde{t}' \preceq \tilde{s}$.*
3. *$C[\tilde{s}] \preceq t$ implies that there is C' such that $t = C'[\tilde{s}]$ and $C \preceq C'$.*

6. A well-structured transition system for behavioural types

We first review below some background material on well-structured transition systems [15]. We then introduce a well-structured transition system for our behavioural types.

6.1. Background

Recall that a transition system is a pair $Tr = (S, \to)$, where S is the set of states and $\to \subseteq S \times S$ is the transition relation. Tr is *finitely-branching* if for each $s \in S$ the set of *successors* $\{s' | s \to s'\}$ is finite.

Definition 4 (WSTS, [15]). *A* well-structured transition system *(WSTS for short) is a pair $\mathcal{W} = (\preceq, Tr)$ where:*

(a) *$Tr = (S, \to)$ is a finitely-branching transition system, and*

(b) *$\preceq$ is a WQO over S that is compatible with $\to$; that is: whenever $s_1 \preceq s_2$ and $s_1 \to s_1'$ then there is s_2' such that $s_2 \to^* s_2'$ and $s_1' \preceq s_2'$.*

Otherwise said, a WSTS is a finitely-branching transition system equipped with a WQO that is a weak simulation relation. Let $\mathcal{W} = (\preceq, Tr)$ be a transition system equipped with a QO $\preceq$. Let $I \subseteq S$ be a set of states. We let the *upward closure* of I, written $\uparrow I$, be $\{s \in S |\ s' \preceq s$ for some $s' \in I\}$. The set $\uparrow \{s\}$ will be often abbreviated as $\uparrow s$. A *basis* of (an upward-closed) set $Y \subseteq S$ is a set I such that $Y = \uparrow I$. We let the *immediate predecessors* of I, written $\mathrm{Pred}(I)$, be the set $\{s \in S |\ s \to s'$ for some $s' \in I\}$ and the set of *predecessors* of I, written $\mathrm{Pred}^*(I)$, be $\{s \in S |\ s \to^* s'$ for some $s' \in I\}$. We say $\mathcal{W}$ has an *(effective) pred-basis* if there is a (computable) function $\mathrm{pb}(\cdot) : S \to 2^S$ such that for each $s \in S$, $\mathrm{pb}(s)$ is a finite basis of $\uparrow \mathrm{Pred}(\uparrow s)$.

Proposition 4 (Finkel and Schnoebelen [15]). *Let $\mathcal{W}$ be a WSTS such that: (a) $\preceq$ is decidable, and (b) $\mathcal{W}$ has an effective pred-basis. Then there is a computable function that, for any finite $I \subseteq S$, yields a finite basis of* $\mathrm{Pred}^*(\uparrow I)$.

The above proposition entails decidability of a number of reachability-related problems in WSTS's (see [15]). Indeed, saying that the set I is reachable from a given state s is equivalent to saying that $s \in \mathrm{Pred}^*(\uparrow I)$: this can be decided, if one has at hand a finite basis B for $\mathrm{Pred}^*(\uparrow I)$, by just checking whether $s \succeq s'$ for some $s' \in B$.

6.2. A WSTS *for behavioural types*

In this subsection, we show that our types can be represented as ordered forests. This representation is convenient for endowing them with a WSTS structure.

Let $(X_i)_{i\geq 1}$ be an infinite set of *variables* disjoint from $\mathcal{N}$ and consider the grammar of types in Section 3, augmented with the clause $T ::= X$, where X ranges over variables. Let $\mathfrak{T}$ be the set of terms generated by this grammar – by "term" we mean here a proper term, *not* an alpha-equivalence class of terms – that respect the following conditions: each variable occurs at most once in a term and only in the scope of restrictions and/or parallel compositions. E.g. $(\nu a^{\mathfrak{a}})(X_1|a^{\mathfrak{a}}.\overline{b}.\overline{c})|X_2$ is in $\mathfrak{T}$, while $\overline{a}.X_1$ is not. In what follows, we will let $C, C', \ldots$ range over $\mathfrak{T}$, reserving the letters $S, T, \ldots$ for the subset of closed terms, that is types.

The definitions and results of Subsection 5.1 apply here, by setting the set of labels thus

$$L = \{(\nu a^{\mathfrak{a}}), G, X \,|\, a \in \mathcal{N}, \mathfrak{a} \subseteq_{\mathrm{fin}} \mathcal{N}, G \text{ guarded summation or replication}\}\,.$$

More explicitly, each context C can be seen as a forest $\mathcal{F}_C$, having restrictions $(\nu a^{\mathfrak{a}})$ as internal labels and either guarded summations/replications G or variables X as leaves, and parallel composition $|$ interpreted as concatenation. That is, each C is mapped to a forest $\mathcal{F}_C$ as follows:

$$\mathcal{F}_X = (X, \varepsilon) \qquad \mathcal{F}_G = (G, \varepsilon) \qquad \mathcal{F}_{T|S} = \mathcal{F}_T \cdot \mathcal{F}_S \qquad \mathcal{F}_{(\nu a^{\mathfrak{a}})T} = ((\nu a^{\mathfrak{a}}), \mathcal{F}_T)\,.$$

Note that $\mathcal{F}_{\mathbf{0}} = \mathcal{F}_{\sum_{i\in\emptyset}\mu_i.T_i} = (\mathbf{0}, \varepsilon)$. It is worth to stress that L can be partitioned into two sets: one containing labels like $(\nu a^{\mathfrak{a}})$, which are only used to annotate internal nodes, and the other containing labels like G and X, which are only used to annotate leaves.

Example 5. Consider the context $C = \overline{b}.f^{\mathfrak{f}} \,|\, (\nu a^{\mathfrak{a}})\big(b^{\mathfrak{b}}.\overline{a}\,|\,X_1\,|\,(\nu c^{\mathfrak{c}})(\overline{b}\,|\,\overline{a}.(\nu d^{\mathfrak{d}})\overline{d}\,)\big)$. The forest $\mathcal{F}_C$ is depicted below.

$$\begin{tikzcd}
\overline{b}.f^{\mathfrak{f}} & & (\nu a^{\mathfrak{a}}) \arrow[dl] \arrow[d] \arrow[dr] & & \\
& b^{\mathfrak{b}}.\overline{a} & X_1 & (\nu c^{\mathfrak{c}}) \arrow[dl] \arrow[dr] & \\
& & \overline{b} & & \overline{a}.(\nu d^{\mathfrak{d}})\overline{d}
\end{tikzcd}$$

Clearly, the QO introduced in Definition 3 is inherited by $\mathfrak{T}$, that is, we can set: $C \preceq C'$ iff $\mathcal{F}_C \preceq \mathcal{F}_{C'}$.

To make the rooted-tree embedding a *well* QO, though, we have to restrict ourselves to some subset of $\mathfrak{T}$ with bounded height and finite set of labels, so as to meet the conditions of Theorem 5. This set will be obtained by tailoring out of $\mathfrak{T}$ a set of all forests that, roughly, only mention terms that occur in a given initial closed term T. To this purpose, we introduce a few more additional notations directly on terms. Given a C, let us write $\mathrm{dp}(C)$ for the maximal *nesting depth of restrictions* in C, defined thus (max over an empty set yields 0):

$$\mathrm{dp}(X) = 0 \quad \mathrm{dp}(\textstyle\sum_{i\in I}\mu_i.C_i) = \max_{i\in I}\mathrm{dp}(C_i) \quad \mathrm{dp}(!a^{\mathfrak{a}}.C) = \mathrm{dp}(C)$$
$$\mathrm{dp}(C_1|C_2) = \max\{\mathrm{dp}(C_1), \mathrm{dp}(C_2)\} \quad \mathrm{dp}((\nu a^{\mathfrak{a}})C) = 1 + \mathrm{dp}(C)\,.$$

In the example above, $\mathrm{dp}(C) = 3$. We denote by $\mathrm{sub}(C)$ the set of variables, summations and replications that occur as subterms of C:

$$\mathrm{sub}(!a.T) = \{!a.T\} \cup \mathrm{sub}(T) \qquad \mathrm{sub}((\nu a)T) = \mathrm{sub}(T) \qquad \mathrm{sub}(X) = \{X\}$$
$$\mathrm{sub}(T|S) = \mathrm{sub}(T) \cup \mathrm{sub}(S) \qquad \mathrm{sub}(\textstyle\sum_{i\in I}\mu_i.T_i) = \{\sum_i \mu_i.T_i\} \cup \bigcup_{i\in I}\mathrm{sub}(T_i)\,.$$

The set $\mathrm{sub}(C)$ is of course finite. We denote by $\mathrm{res}(C)$ the set of restrictions $(\nu a^{\mathfrak{a}})$ occurring in C. The set of terms we are interested in is defined below.

Definition 5 ($\mathfrak{T}_T[\tilde{X}]$ **).** *Let T be a type and $\tilde{X}$ be a finite set of variables. Then*

$$\mathfrak{T}_T[\tilde{X}] \triangleq \left\{ C \in \mathfrak{T} \,\middle|\, \partial C \subseteq \mathrm{sub}(T) \cup \tilde{X},\ \mathrm{res}(C) \subseteq \mathrm{res}(T),\ \mathrm{dp}(C) \le \mathrm{dp}(T) \right\}.$$

In the following, we abbreviate $\mathfrak{T}_T[\tilde{X}]$ as $\mathfrak{T}_T$ when $\tilde{X} = \emptyset$. Consider now the rooted-tree embedding $\preceq$ (Definition 3), we have the following result.

Proposition 5. *For any T and $\tilde{X}$, the relation $\preceq$ is a* WQO *over $\mathfrak{T}_T[\tilde{X}]$.*

PROOF: The forests in $\mathfrak{T}_T[\tilde{X}]$ have bounded height: indeed, for any $C \in \mathfrak{T}_T[\tilde{X}]$, we have $\mathrm{h}(C) = \mathrm{dp}(C) \le \mathrm{dp}(T)$. Moreover, they are built using a finite set of labels: $L = \tilde{X} \cup \mathrm{res}(T) \cup \mathrm{sub}(T)$. Hence $\mathfrak{T}_T[\tilde{X}] \subseteq \mathfrak{F}^{(k)}$, with $k = \mathrm{dp}(T)$. Theorem 5 ensures then that $\preceq$ is a WQO over $\mathfrak{F}^{(k)}$, hence over $\mathfrak{T}_T[\tilde{X}]$. □

We want to show now that $\mathfrak{T}_T$ can be endowed with WSTS structure. In what follows, we shall consider the standard CCS transition relation over closed terms [25], denoted here by $\overset{\mu}{\longmapsto}$ (see AppendixA); in particular, we shall write $\overset{\tau}{\longmapsto}$ as $\mapsto$. The relation $\mapsto$ is preferable to $\to$ in the present context, because it avoids alpha-renaming, structural congruence and is finitely branching for the considered fragment of CCS. In Section 7, we will show that $\mapsto$ is equivalent to $\to$ for the purpose of defining the satisfaction relation $S \models \phi$.

The set of closed terms $\mathfrak{T}_T$ enjoys the properties stated in the lemmas below, which can be easily inferred by induction on the structure of the term. Note in particular that, by Lemma 7, the restriction nesting depth of any term is *not* increased by $\mapsto$. This is a crucial property that does not hold in the pi-calculus. E.g. in the pi-term below, the restriction nesting depth grows indefinitely. Reductions that increase the depth by one are alternated to reductions that keeps the nesting depth unchanged (type annotations omitted and communicating prefixes underlined):

$$\begin{array}{l}
(\nu b_1)\,\underline{\overline{a}\langle b_1\rangle} \,|\, \underline{!a(y)}.(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, !c(x).\overline{a}\langle x\rangle \;\to \\
\quad (\nu b_1)\,(\nu b_2)(b_1.b_2 \,|\, \underline{\overline{c}\langle b_2\rangle}) \,|\, !a(y).(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, \underline{!c(x)}.\overline{a}\langle x\rangle \;\to \\
\quad (\nu b_1)(\nu b_2)(b_1.b_2 \,|\, \underline{\overline{a}\langle b_2\rangle}) \,|\, \underline{!a(y)}.(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, !c(x).\overline{a}\langle x\rangle \;\to \\
\quad (\nu b_1)\,(\nu b_2)(b_1.b_2 \,|\, (\nu b_3)(b_2.b_3 \,|\, \underline{\overline{c}\langle b_3\rangle})) \,|\, !a(y).(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, \underline{!c(x)}.\overline{a}\langle x\rangle \;\to^* \\
\quad (\nu b_1)\,(\nu b_2)(b_1.b_2 \,|\, (\nu b_3)(b_2.b_3 \,|\, \cdots (\nu b_{n+1})(b_n.b_{n+1} \,|\, \overline{c}\langle b_{n+1}\rangle)\cdots)) \\
\qquad |\, !a(y).(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, !c(x).\overline{a}\langle x\rangle \;.
\end{array}$$

Lemma 7. *Assume* $S \overset{\mu}{\longmapsto} S'$. *Then* $\mathrm{dp}(S) \geq \mathrm{dp}(S')$ *and* $\mathrm{res}(S) \supseteq \mathrm{res}(S')$.

It is then easy to prove that the set $\mathfrak{T}_T$ is closed with respect to $\overset{\mu}{\longmapsto}$.

Lemma 8. *(1)* $T \in \mathfrak{T}_T$. *(2) For any* $S \in \mathfrak{T}_T$ *and* S', $S \overset{\mu}{\longmapsto} S'$ *implies that* $S' \in \mathfrak{T}_T$.

PROOF: Part (1) follows by definition of $\mathfrak{T}_T$ (notice that $\partial T \subseteq \mathrm{sub}(T)$).

Concerning part (2), the proof proceeds by induction on the derivation of $S \overset{\mu}{\longmapsto} S'$ and by distinguishing the last transition rule applied. The base cases of prefixes are easy to prove by applying Lemma 7 and by noting that $\mathrm{sub}(S) \supseteq \partial S$ for any S. In the other cases the proof proceeds by applying the inductive hypothesis.

Consider e.g. the case $S = (\nu a)U \overset{\mu}{\longmapsto} (\nu a)U' = S'$. $S \in \mathfrak{T}_T$ implies $U \in \mathfrak{T}_T$ by definition. Therefore, by applying the induction hypothesis, we get $U' \in \mathfrak{T}_T$ and, by Lemma 7, $\mathrm{dp}(U') \leq \mathrm{dp}(U)$ and $\mathrm{res}(U') \subseteq \mathrm{res}(U)$ hence $\mathrm{dp}(S') \leq \mathrm{dp}(S)$ and $\mathrm{res}(S') \subseteq \mathrm{res}(S)$ and $S' \in \mathfrak{T}_T$. □

Lemma 9. *Suppose that either* $C[X_1,X_2] \preceq C'[X_1,X_2]$ *or* $C[X_1,X_2] \succeq C'[X_1,X_2]$ *holds true. If* $G_1|G_2 \overset{\mu}{\longmapsto} S_1|S_2$ *and* $C[G_1,G_2] \overset{\mu}{\longmapsto} C[S_1,S_2]$ *then* $C'[G_1,G_2] \overset{\mu}{\longmapsto} C'[S_1,S_2]$.

Proposition 6. *The relation $\preceq$ is a simulation relation over $\mathfrak{T}_T$ wrt $\mapsto$.*

PROOF: Assume $S \preceq T$ and $S \mapsto S'$. Let G_1, G_2 be the generic sums which this reduction originates from, i.e. assume, for some C, S_1 and S_2, that $S = C[G_1, G_2] \mapsto C[S_1, S_2] = S'$ and that $G_1|G_2 \mapsto S_1|S_2$.

By Lemma 6 (3), there is C' such that $T = C'[G_1, G_2]$, with $C \preceq C'$. Hence, by Lemma 9, $T = C'[G_1, G_2] \mapsto C'[S_1, S_2] \stackrel{\triangle}{=} T'$. Moreover, by Lemma 6 (1) $T' = C'[S_1, S_2] \succeq C[S_1, S_2] = S'$. $\square$

As a consequence of Proposition 5 and 6 above we get the wanted result.

Corollary 1. *For any T, let Tr be the transition system $(\mathfrak{T}_T, \overset{\tau}{\longmapsto})$. Then $\mathcal{W}_T \stackrel{\triangle}{=} (\preceq, Tr)$ is a* WSTS.

Remark 2. Consider the labelled version of the reduction relation, $\overset{\langle\lambda\rangle}{\longmapsto}$, $\lambda ::= a|\varepsilon$. For any a, Proposition 6 still holds if considering the transition system given by $\overset{\langle a\rangle}{\longmapsto}$, rather than $\mapsto$. As a consequence, $(\mathfrak{T}_T, \overset{\langle a\rangle}{\longmapsto})$ is a WSTS too for any a.

Concerning the decidability of $\mathcal{W}_T$, we note that: (a) the WQO $\preceq$ is decidable, indeed its very inductive definition yields a decision algorithm; (b) the transition relation $\overset{\mu}{\longmapsto}$ is decidable for the fragment of CCS that corresponds to our language of types.

7. Decidability

In this section, we prove the decidability of an interesting monotone fragment of Shallow Logic, applying Proposition 4 to $\mathcal{W}_T$. The WQO $\preceq$ has already seen to be decidable. In order to be able to apply this proposition, we have to fulfill obligation (b), that is, to show that $\mathcal{W}_T$ has an effective pred-basis. Moreover, we have to show that each denotation $[\![\phi]\!]$, under certain conditions on ϕ, can be presented via an effectively computable finite basis: this will play the role of "I" in Proposition 4. We face these tasks in the next two subsections.

7.1. Pred-basis

Informally, the pred basis function, $\mathrm{pb}_T(S)$, works in two steps. First, all decompositions of S as $S = C[\tilde{U}]$, with $|\tilde{U}| = 0, 1$ or 2, are considered – there are finitely many of them. Then, out of each C, all contexts C' are considered that can be built by inserting 1 or 2 extra holes into C. Again, there are finitely many such contexts. The contexts C' are then filled with ground leaves, in such a way that the resulting terms posses a reduction to S, up to $\succeq$.

Definition 6 (pred-basis). *Let T be a type and let $S \in \mathfrak{T}_T$. Let C, C' below range over $\mathfrak{T}_T[X_1, X_2]$. We define:*

$$\mathrm{pb}_T(S) \stackrel{\triangle}{=} \bigcup_{S=C[\tilde{U}]} \left\{ C'[\tilde{G}] \in \mathfrak{T}_T \,|C' \succeq C, \partial^g C' = \partial^g C, \tilde{G} \subseteq \mathrm{sub}(T),\, C'[\tilde{G}] \mapsto\succeq S \right\}.$$

The construction of $\mathrm{pb}_T(S)$ is effective. In particular, given C, there are finitely many ways of inserting one or two extra holes to C, resulting into a $C' \succeq C$, and they can all be considered in turn. In what follows we let $\mathrm{Pred}_T(\cdot)$ stand for $\mathrm{Pred}(\cdot) \cap \mathfrak{T}_T$.

Theorem 6. *Suppose $T \in \mathcal{T}$. Then for any $S \in \mathfrak{T}_T$, $\uparrow \mathrm{pb}_T(S) = \uparrow \mathrm{Pred}_T(\uparrow S)$. Moreover, $\mathrm{pb}_T(\cdot)$ is effective.*

PROOF: Effectiveness has already been discussed above.

By construction, $\uparrow \mathrm{pb}_T(S) \subseteq \uparrow \mathrm{Pred}_T(\uparrow S)$. Let us examine the other inclusion. Suppose first $V \mapsto\succeq S$, we show that there is $U \in \mathrm{pb}_T(S)$ s.t. $V \succeq U$: this will be sufficient to prove also the most general case $V \succeq\mapsto\succeq S$, since $\mathcal{W}_T$ is a WSTS.

Assume that the reduction in V originates from two communicating prefixes – the τ-prefix case is easier. That is, assume $V = C[G_1, G_2] \mapsto C[S_1, S_2] \succeq S$, where $G_1|G_2 \mapsto S_1|S_2$.

By Lemma 6, $S = C''[\tilde{S}']$, with $C \succeq C''$ and $(S_1, S_2) \succeq \tilde{S}'$. By Lemma 5, it is possible to build out of C'' a 2-holes context $C' \in \mathfrak{T}_T[X_1, X_2]$ such that $C \succeq C' \succeq C''$. Take $U = C'[G_1, G_2]$; notice that $U \preceq V$ by Lemma 6. Now, $U \in \mathrm{pb}_T(S)$. Indeed, by Lemma 5, $\partial^o C' = \partial^o C$, $\partial^g C'' = \partial^g C'$ and $C'' \preceq C'$. Notice also that $\{G_1, G_2\} \subseteq \mathrm{sub}(T)$ and by $C' \preceq C$ and Lemma 9, $U \mapsto C'[S_1, S_2] \succeq S$. □

We can extend pb_T to finite sets $I \subseteq \mathfrak{T}_T$, by setting $\mathrm{pb}_T(I) \stackrel{\triangle}{=} \bigcup_{S \in I} \mathrm{pb}_T(S)$. By doing so, we obtain the following corollary, which says that $\mathcal{W}_T$ has an effective pred-basis.

Corollary 2. *There is a computable function $\mathrm{pb}_T(\cdot)$ such that, for any finite $I \subseteq \mathfrak{T}_T$, $\uparrow \mathrm{pb}_T(I) = \uparrow \mathrm{Pred}_T(\uparrow I)$.*

Remark 3. Consider the labelled version of the reduction relation, $\overset{\langle\lambda\rangle}{\longmapsto}$, $\lambda ::= a|\varepsilon$. For any fixed label $\langle a \rangle$, Corollary 2 still holds if considering the transition system given by $\overset{\langle a\rangle}{\longmapsto}$, rather than $\mapsto$. We shall name $\mathrm{pb}_T^{\langle a\rangle}(\cdot)$ the corresponding pred-basis function.

Applying Proposition 4, we get the result we were after.

Corollary 3. *There exists a computable function $\mathrm{pb}^*_T(\cdot)$ such that, for any finite set $I \subseteq \mathfrak{T}_T$, $\mathrm{pb}^*_T(I)$ is a finite basis of $\mathrm{Pred}^*_T(\uparrow I)$.*

7.2. Finite bases for plain formulae

We first show that, for certain formulae ϕ, the satisfaction relation $S \models \phi$ can be defined relying solely on $\mapsto$ and on context decomposition, with no reference to structural congruence and $\to$. We then proceed by showing that the denotation of a formula is upward-closed, hence it makes sense to look for a finite basis of it. We do so in Proposition 7 and Proposition 8 below. Both results are valid only for plain monotone formulae, as defined below.

Definition 7 (plain formulae). *The set of* plain *formulae contains all formulae $\phi, \phi', \ldots$ that can be generated by the following grammar:*

$$\begin{aligned}\psi &::= \boldsymbol{T} \mid a \mid \overline{a} \mid \mathrm{H}^*(\psi_1|\psi_2) \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle a \rangle \psi \\ \phi &::= \psi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid \Diamond^* \phi\,.\end{aligned}$$

For the sake of readability, from now on we will abbreviate formulae of the form $\mathrm{H}^*(\psi_1|\psi_2)$ as $\psi_1 \,\hat{|}\, \psi_2$. Note that

$$[\![\psi_1 \,\hat{|}\, \psi_2]\!] = \{S : S \equiv (\nu\tilde{x})(S_1|S_2),\ \tilde{x}\#\mathrm{n}(\psi_1 \,\hat{|}\, \psi_2),\ S_i \in [\![\psi_i]\!],\ i = 1,2\}\,.$$

In essence, plain formulae are formulae where $\Diamond^*$ cannot occur underneath H^* while $|$ are always H^*-guarded.[2] We briefly explain the rationale behind the two restrictions of plain formulae. We want that checking $T \models \mathrm{H}^*\phi$ be essentially equivalent to checking satisfiability of ϕ against the parallel composition of T's leaves. This is clearly not the case if $\Diamond^*$ does appear underneath H^*: indeed, putting the leaves of T in parallel may give rise to interactions not possible in T, merely because certain names that in T are kept distinct by restrictions are now free and in a position to communicate. As an example, suppose $T = (\nu a)(a.c) \,|\, (\nu a)(\overline{a}.d)$ and $\phi = \mathrm{H}^*\Diamond^* c$. Clearly $T \not\models \phi$ while ∂T does: $a.c \,|\, \overline{a}.d \models \phi$. Note that the crucial point here is that we want to check satisfiability while disallowing alpha-equivalence.

The second constraint is necessary in order to guarantee the upward-closure of the denotation of formulae. Indeed, in the absence of this constraint, insertion of new leaves in a term might create new connections that can prevent the splitting of the new term, as shown in the following example. Consider $T \stackrel{\triangle}{=} (\nu c,d)(a.c \,|\, b.d)$ and $S \stackrel{\triangle}{=} (\nu c,d)(a.c \,|\, b.d \,|\, c.d)$; clearly $T \preceq S$ and $T \models a|b$, however $S \not\models a|b$.

Given two sequences $\tilde{\ell}$ and $\tilde{\ell}'$, in what follows we let $\tilde{\ell} * \tilde{\ell}'$ denote a generic element of the shuffle of $\tilde{\ell}$ and $\tilde{\ell}'$.

[2]The condition of H^*-guardedness of parallel composition was mistakenly omitted in the short version of this paper [3].

The following proposition gives an alternative characterization of formulae satisfaction in terms of either the shape or the behaviour of the terms. The proof is reported in AppendixB.

Proposition 7. *Assume $S \in \mathfrak{T}_T$, ϕ plain and monotone and* $\mathrm{bn}(T) \cap \mathrm{n}(\phi) = \emptyset$. *Then we have the following equivalences, where $\tilde{G}, \tilde{G}_1, \tilde{G}_2$ are sequences of leaves assumed to be included in* $\mathrm{sub}(T)$*:*

$$
\begin{array}{ll}
S \models a & \textit{iff } \exists G \in \partial S :\ G \searrow_a \\
S \models \overline{a} & \textit{iff } \exists G \in \partial S :\ G \searrow_{\overline{a}} \\
S \models \psi_1 \,\text{⚲}\, \psi_2 & \textit{iff } \partial S = \tilde{G} * \tilde{F} \textit{ with } \tilde{G} \models \psi_1 \textit{ and } \tilde{F} \models \psi_2 \\
S \models \langle a \rangle \phi & \textit{iff } \exists U : S \overset{\langle a \rangle}{\longmapsto} U \textit{ and } U \models \phi \\
S \models \Diamond^* \phi & \textit{iff } \exists U : S \mapsto^* U \textit{ and } U \models \phi .
\end{array}
$$

By definition of $\mathcal{F}_T$, i.e. by the partitioning of nodes into internal ones and leaves, and by definition of $\preceq$ (rooted tree embedding), we get the following result.

Lemma 10. *$S \preceq T$ implies $\partial S \preceq \partial T$.*

As a consequence of the previous lemma we get that $S \preceq T$ implies $\partial S \subseteq \partial T$, where both ∂S and ∂T are seen as multisets. We are now ready to prove that the denotation of each plain and monotone formula is an upward-closed set.

Proposition 8. *Assume ϕ plain and monotone. Then $[\![\phi]\!]$ is upward-closed.*

PROOF: The proof proceeds by induction on the structure of ϕ. The cases $\phi ::= a \,|\, \overline{a} \,|\, \mathbf{T}$ and the boolean connectives are easy to prove. The cases $\phi ::= \Diamond^* \phi' \,|\, \langle a \rangle \phi'$ follow by Proposition 6 and Remark 2. Consider now $\phi = \psi_1 \,\text{⚲}\, \psi_2$ and any $S \models \phi$. By Proposition 7, we get that $\partial S = \tilde{G} * \tilde{F}$ with $\tilde{G} \models \psi_1$ and $\tilde{F} \models \psi_2$. Consider now any T such that $S \preceq T$. By Lemma 10, we get $\partial S \preceq \partial T$, hence $\tilde{G} \preceq \tilde{G}'$ and $\tilde{F} \preceq \tilde{F}'$ for suitable $\tilde{G}'$ and $\tilde{F}'$ such that $\partial T = \tilde{G}' * \tilde{F}'$. By applying the induction hypothesis to ψ_1 and ψ_2 we get $\tilde{G}' \models \psi_1$ and $\tilde{F}' \models \psi_2$ and, by Proposition 7,s it follows that $T \models \psi_1 \,\text{⚲}\, \psi_2$. $\square$

As discussed at the beginning of this section, in order to take advantage of Corollary 3, we have to show that each set $[\![\phi]\!]$, or, more accurately, each set $[\![\phi]\!]_T \overset{\triangle}{=} [\![\phi]\!] \cap \mathfrak{T}_T$, can be presented via an effectively computable finite basis in $\mathcal{W}_T$. We define this basis for ϕ below. This definition is by induction on the structure of ϕ. The $\Diamond^*$ and $\langle a \rangle$ cases take advantage of the pred-basis function (Definition 6), the other cases basically follow the corresponding cases of Proposition 7 or, in the case of $\vee$ and $\mathbf{T}$, the expected boolean interpretation. The only exception to this scheme is the $\wedge$ connective, which is nontrivial and will be commented below. Some more terminology first. Given an upward closed set

$I \subseteq \mathfrak{T}_T$, we denote by $\mathrm{Min}(I)$ an arbitrarily chosen set of representatives of the minimal elements of I, wrt the WQO $\preceq$, i.e. $\uparrow \mathrm{Min}(I) = I$.

In the following, we say that a context C is *pure* if $\partial^g C = \emptyset$ and we let D range over pure contexts, e.g. $D = (\nu a)(X_1|X_2)|X_3$ is pure.

Definition 8 (finite basis). *Let T be a type and ϕ be a plain and monotone formula, such that* $\mathrm{bn}(T) \cap \mathrm{n}(\phi) = \emptyset$. *The* finite basis $\mathrm{Fb}_T(\phi)$ *is inductively defined below, where $G, \tilde{G}, \tilde{G}_1$ and $\tilde{G}_2$ are sequences of leaves assumed to be included in* $\mathrm{sub}(T)$*:*

$$\begin{aligned}
\mathrm{Fb}_T(\boldsymbol{T}) &\triangleq \{D[G] \in \mathfrak{T}_T \mid G \in \mathrm{sub}(T)\}\\
\mathrm{Fb}_T(a) &\triangleq \{D[G] \in \mathfrak{T}_T \mid G \searrow_a\}\\
\mathrm{Fb}_T(\overline{a}) &\triangleq \{D[G] \in \mathfrak{T}_T \mid G \searrow_{\overline{a}}\}\\
\mathrm{Fb}_T(\psi_1 \,\text{?}\, \psi_2) &\triangleq \textstyle\bigcup_{S_1 \in \mathrm{Fb}_T(\psi_1), S_2 \in \mathrm{Fb}_T(\psi_2)} \left\{D[\partial S_1 * \partial S_2] \in \mathfrak{T}_T\right\}\\
\mathrm{Fb}_T(\phi_1 \vee \phi_2) &\triangleq \mathrm{Fb}_T(\phi_1) \cup \mathrm{Fb}_T(\phi_2)\\
\mathrm{Fb}_T(\phi_1 \wedge \phi_2) &\triangleq \mathrm{Min}([\![\phi_1]\!] \cap [\![\phi_2]\!])\\
\mathrm{Fb}_T(\langle a \rangle \phi) &\triangleq \mathrm{pb}_T^{\langle a \rangle}(\mathrm{Fb}_T(\phi))\\
\mathrm{Fb}_T(\Diamond^* \phi) &\triangleq \mathrm{pb}_T^{*}(\mathrm{Fb}_T(\phi))\,.
\end{aligned}$$

As required by Proposition 4, the basis defined above is finite. This follows from finiteness of the set $\mathrm{sub}(T)$ and of the number of 1-hole pure contexts $D[\cdot]$ one can build from a given T. Finiteness of the set $\mathrm{Min}([\![\phi_1]\!] \cap [\![\phi_2]\!])$ follows from Corollary 1 and Definition 2: if not, one would find an infinite sequence of pairwise incomparable elements, thus violating the condition of WQO. The proof of the following proposition can be found in AppendixC.

Proposition 9. *Consider T and ϕ like in Definition 8. Then* $\mathrm{Fb}_T(\phi)$ *is a finite basis for* $[\![\phi]\!]_T$*, that is* $\uparrow \mathrm{Fb}_T(\phi) = [\![\phi]\!]_T$.

In order to prove the decidability of the logic, it is necessary to guarantee the effectiveness of $\mathrm{Fb}_T(\phi)$. This is not an easy matter due to the case $\phi = \phi_1 \wedge \phi_2$, where effectiveness of the set $\mathrm{Min}([\![\phi_1]\!] \cap [\![\phi_2]\!])$ has to be made effective. In order to do that, we introduce the merge operator $|||$, which allows one to compute a finite over-approximation of Min by appropriately merging terms, seen as forests, drawn from $\mathrm{Fb}_T(\phi_1)$ and $\mathrm{Fb}_T(\phi_2)$.

For any forest $\mathcal{F} = (\ell_1, \mathcal{F}_1) \cdots (\ell_n, \mathcal{F}_n)$ we write $\mathcal{F}(\ell)$ for the sub-forest of $\mathcal{F}$ containing all trees with root ℓ. That is, $\mathcal{F}(\ell) = (\ell_{i_1}, \mathcal{F}_{i_1}) \cdots (\ell_{i_k}, \mathcal{F}_{i_k})$ where $1 \leq i_1 < \cdots < i_k \leq n$ and $\ell_{i_j} = \ell$ for $j = 1, \cdots, k$, $\ell_{i_m} \neq \ell$ for each $i_m \in \{1, \cdots, n\} \setminus \{i_1, \cdots, i_k\}$. The forest $\mathcal{F}_1 \,|||\, \mathcal{F}_2$ is defined as the set containing all terms obtained as combinations of the subtrees with common roots in $\mathcal{F}_1$ and $\mathcal{F}_2$. Formally, we have:

Definition 9 (forest merging). *Let $\mathcal{G}$ and $\mathcal{H}$ be forests. Then $\mathcal{G}\,|||\,\mathcal{H}$ is a set of forests defined as follows. $\mathcal{F} \in \mathcal{G}\,|||\,\mathcal{H}$ iff for each root ℓ of $\mathcal{F}$, letting $\mathcal{F}(\ell) = (\ell, \mathcal{F}_1)\cdots(\ell, \mathcal{F}_n)$, $\mathcal{G}(\ell) = (\ell, \mathcal{G}_1)\cdots(\ell, \mathcal{G}_m)$ and $\mathcal{H}(\ell) = (\ell, \mathcal{H}_1)\cdots(\ell, \mathcal{H}_k)$ one has*

1. $\min\{m, k\} \le n \le m + k$*;*
2. *for each* $1 \le i \le n$

$$\mathcal{F}_i \in \{\mathcal{G}_1, \cdots, \mathcal{G}_m, \mathcal{H}_1, \cdots \mathcal{H}_k\} \cup \bigcup_{j=1,\cdots,m,\ l=1,\cdots,k} \mathcal{G}_j\,|||\,\mathcal{H}_l \ .$$

Note that $\{\mathcal{G}, \mathcal{H}\} \subseteq \mathcal{G}\,|||\,\mathcal{H}$.

Example 6 (forest merging). *Suppose $L = \{a, b, c, d, e\}$ and consider the forests depicted below: $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ belong to the merging $\mathcal{G}\,|||\,\mathcal{H}$.*

$\mathcal{G}$
$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & & b \arrow[d] \\ b & & c \arrow[d] & a \\ & & d & \end{tikzcd}$$

$\mathcal{H}$
$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & & b \arrow[d] \\ b & & c \arrow[d] & e \\ & & e & \end{tikzcd}$$

$\mathcal{F}_1$
$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & & b \arrow[d] \\ b & & c \arrow[d] & e \\ & & d & \end{tikzcd}$$

$\mathcal{F}_2$
$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & & b \arrow[d] \\ b & & c \arrow[dl] \arrow[dr] & a \\ & d & & e \end{tikzcd}$$

$\mathcal{F}_3$
$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & & & b \arrow[dl] \arrow[dr] & \\ b & & c \arrow[dl] \arrow[dr] & a & & e \\ & d & & e & & \end{tikzcd}$$

The definition of $|||$ can be generalized to *sets* of forests $\mathfrak{A}$ and $\mathfrak{B}$ as follows

$$\mathfrak{A}\,|||\,\mathfrak{B} \stackrel{\triangle}{=} \bigcup_{\mathcal{G}\in\mathfrak{A},\ \mathcal{F}\in\mathfrak{B}} \mathcal{G}\,|||\,\mathcal{F} \ .$$

The following lemma will give us a way of computing $\mathrm{Fb}_T(\phi_1 \wedge \phi_2)$.

Lemma 11. *Let ϕ_1 and ϕ_2 be plain and monotone. Then*

$$[\![\phi_1 \wedge \phi_2]\!]_T = \uparrow \Big(\big(\mathrm{Fb}_T(\phi_1)\,|||\,\mathrm{Fb}_T(\phi_2)\big) \cap \uparrow \mathrm{Fb}_T(\phi_1) \cap \uparrow \mathrm{Fb}_T(\phi_2) \Big) \, .$$

PROOF: We prove the inclusion in both directions. The $\supseteq$-direction follows by definition of $[\![\phi_1 \wedge \phi_2]\!]_T$ and from Proposition 8 and 9. Consider the opposite inclusion. We prove the following: for any U, if there are S_1, S_2 s.t. $U \succeq S_i$ $(i = 1, 2)$ then there is $S \in S_1\,|||\,S_2$ s.t. $U \succeq S \succeq S_i$ $(i = 1, 2)$. This fact implies the thesis, because if $U \in [\![\phi_1 \wedge \phi_2]\!]_T$ then, by Proposition 9, there are $S_i \in \mathrm{Fb}_T(\phi_i)$ s.t. $U \succeq S_i$ (i=1,2); hence S belongs to the r.h.s. of the equality above.

So, assume $U \succeq S_1, S_2$. We want to prove that there is a $S \in \big((S_1\,|||\,S_2) \cap \uparrow S_1 \cap \uparrow S_2\big)$ such that $S \preceq U$. The proof proceeds by induction on the height of U seen as a forest. For a generic label ℓ, let

$$\begin{array}{rcl} U(\ell) & = & (\ell, U_1)\cdots(\ell, U_n) \\ S_1(\ell) & = & (\ell, T_1)\cdots(\ell, T_m) \\ S_2(\ell) & = & (\ell, V_1)\cdots(\ell, V_k) \end{array}$$

be the subforests with roots ℓ in U, S_1 and S_2. By definition of rooted-tree embedding (Definition 3), there are distinct indices $1 \leq i_1 < \cdots < i_m \leq n$ and $1 \leq l_1 < \cdots < l_k \leq n$ s.t. for each j, $1 \leq j \leq m$ it holds that $T_j \preceq U_{i_j}$ and for each j, $1 \leq j \leq k$ it holds that $V_j \preceq U_{l_j}$.

Let $f_1 : \{1, \cdots, m\} \to \{1, \cdots, n\}$ and $f_2 : \{1, \cdots, k\} \to \{1, \cdots, n\}$ such that $f_1(j) = i_j$ and $f_2(j) = l_j$. Let $u_1, \cdots, u_v$ be the indices in $\text{range}(f_1) \cup \text{range}(f_2)$. Note that $v \leq m + k$ by definition. Define S such that, for each ℓ, $S(\ell) = (\ell, S'_1) \cdots (\ell, S'_v)$, with the S'_j defined as follows, for each $j \in 1, \cdots, v$:

- if $i_j \in \text{range}(f_1) \setminus \text{range}(f_2)$ then take $S'_j = T_t$ if $f_1(t) = i_j$ (hence $S'_j \preceq U_{i_j}$);
- if $i_j \in \text{range}(f_2) \setminus \text{range}(f_1)$ then take $S'_j = V_t$ if $f_2(t) = i_j$ (hence $S'_j \preceq U_{i_j}$);
- if $i_j \in \text{range}(f_1) \cap \text{range}(f_2)$ then there are t and s such that $f_1(t) = i_j$ and $f_2(s) = i_j$. By applying the induction hypothesis to U_{i_j} we get that there is $B \in \big((T_t \,|||\, V_s) \cap \uparrow T_t \cap \uparrow V_s\big)$ such that $B \preceq U_{i_j}$. Choose $S'_j = B$.

By construction $S(\ell) \preceq U(\ell)$, for each label ℓ; therefore $S \preceq U$. Moreover, again by construction, $S \succeq S_1$, $S \succeq S_2$ and $S \in \big((S_1 \,|||\, S_2) \cap \uparrow S_1 \cap \uparrow S_2\big)$.

□

Lemma 12. *For any $T \in \mathcal{T}$, $\text{Fb}_T(\cdot)$ is computable.*

PROOF: This follows from effectiveness of the cases of the inductive definition of $\text{Fb}_T(\cdot)$. Effectiveness of the cases $\phi ::= \mathbf{T} \mid a \mid \overline{a}$ is given by finiteness of $\text{sub}(\cdot)$ and finiteness of the number of pure contexts D. In the cases $\phi_1 \vee \phi_2$ and $\psi_1 \,\text{⫯}\, \psi_2$ the proof relies on the inductive hypothesis. In the cases $\phi ::= \langle a \rangle \phi \mid \Diamond^* \phi$ the proof relies on Theorem 6, Corollary 2 and 3. Finally, in case $\phi_1 \wedge \phi_2$ the proof relies on Lemma 11, by virtue of which $\text{Min}([\![\phi_1]\!] \cap [\![\phi_2]\!])$ can be realized by the set

$$\begin{aligned}&(\text{Fb}_T(\phi_1) \,|||\, \text{Fb}_T(\phi_2)) \cap \uparrow \text{Fb}_T(\phi_1) \cap \uparrow \text{Fb}_T(\phi_2)\\ &\quad = \big\{S \in (\text{Fb}_T(\phi_1) \,|||\, \text{Fb}_T(\phi_2)) \;\big|\; S \succeq \text{Fb}_T(\phi_1),\ S \succeq \text{Fb}_T(\phi_2)\big\}\ .\end{aligned}$$

Effectiveness then follows from the inductive hypothesis applied to $\text{Fb}_T(\phi_1)$ and $\text{Fb}_T(\phi_2)$, and from computability of the rooted tree embedding $\preceq$. □

Theorem 7. *Consider T and ϕ like in Definition 8. Then $\text{Fb}_T(\phi)$ is a finite basis for $[\![\phi]\!]_T$, that is $\uparrow \text{Fb}_T(\phi) = [\![\phi]\!]_T$. Moreover, $\text{Fb}_T(\cdot)$ is computable.*

PROOF: By Proposition 9 and Lemma 12. □

By virtue of the above theorem, we can decide if $S \models \phi$, with $S \in \mathfrak{T}_T$, by checking if there is $U \in \text{Fb}_T(\phi)$ s.t. $S \succeq U$. Since $\preceq$ is decidable, this can be effectively carried out, and we obtain Corollary 4. Finally, Corollary 5 is a consequence of Proposition 2.

Corollary 4 (decidability on types). *Let ϕ be plain and monotone. It is decidable whether $T \models \phi$. Hence, decidability also holds for ϕ plain and anti-monotone.*

Corollary 5 (decidability on pi-processes). *Let Γ be a context. Given a Γ-well-typed pi-process P and ϕ plain and (anti-)monotone, it is decidable whether $P \models \phi$.*

Remark 4 (again on Kruskal's preorder). It should now be clear why Kruskal's preorder cannot be usefully employed as WQO for types. Indeed, by recalling the definition of $\preceq^{\mathrm{kr}}$ in Remark 1, we would for instance get that

$$\begin{tikzcd}
 & (\nu a^{\mathfrak{a}}) \arrow[dl] \arrow[dr] & & \preceq^{\mathrm{kr}} & & (\nu a^{\mathfrak{a}}) \arrow[dl] \arrow[dr] & \\
a.b.c & & \overline{a}.\overline{b}.\overline{c} & & a.b.c & & (\nu b^{\mathfrak{b}}) \arrow[d] \\
 & & & & & & \overline{a}.\overline{b}.\overline{c}
\end{tikzcd}$$

The term on the left satisfies the formula $\diamondsuit^* \langle c \rangle$, while this is not true for the term on the right.

8. Conclusion and related work

We have proven the decidability of a fragment of Spatial Logic that includes interesting safety properties of a class of infinite-control pi-calculus processes. The proof relies heavily on both behavioural type systems [17, 3] and well-structured transition system techniques [15].

Implementation issues are not in the focus of the present paper. Whether a practical algorithm can be obtained or not from the theoretical discussion presented here is an interesting topic, that is left for future work. Complexity of the presented decision procedure is also left as an open problem.

Our proof of decidability for CCS generalizes the result in [5] that "weak" barbs $\diamondsuit^* a$ are decidable in CCS with replication. Variations and strengthening of these results have recently been obtained by Valencia et al. [28]. It is worthwhile to note that weak barbs are *not* decidable in the pi-calculus in general [4]. On the other hand, our results show that they are decidable restricting to well-typed pi-processes.

During our investigation, we have also considered the possibility of approximating CCS with Petri Nets, somehow along the lines of [21, 22, 23], where *structural stationary* pi-processes are mapped into finite nets. Unfortunately, this approach turned not to reconcile well with the needs of Spatial Logic. In particular, it appears that the spatial structure of terms determined by restrictions is hard to recover from the nets resulting from the translation.

Decidability of Shallow logic in a general setting of Spatial Transition Systems, i.e. transition systems where a monoidal structure on states accounts for the "spatial" dimension, has been recently studied in [1]. The negation-free fragment of the logic has been shown to be decidable and an alternative characterization of the logical preorder is given in terms of a weak simulation enriched with constraints on the spatiality of terms. This result has been obtained by applying forward reachability analyses techniques based on the *completion* of WSTS, recently introduced in [13, 14]. These techniques are essentially based on providing a finite representation of downward closed sets.

Also related to our approach are Yoshida's graph types [29], which extend Milner's sorting system with informations on the communication behaviors of terms expressed as graphs, and some recent proposals by Caires [6, 7], where a logical semantics approach to types for concurrency is pursued. Specifically, in [7], a notion of spatial-behavioural typing suitable to discipline concurrent interactions and resource usage in a distributed object calculus is defined. Types, that can be viewed as a fragment of a spatial logic for concurrency, express resource ownership. The proposed system guarantee the availability of services and (resource access) race freedom. Closest to our type system is [6], where a generic type system for the pi-calculus - parameterized on the subtyping relation - is proposed. In [8], Caires has proved that model-checking of bounded pi-calculus processes, and in particular of finite-control processes, is decidable. Note that the class of processes we have considered here includes properly finite-control ones.

Acknowledgments. We wish to thank Luis Caires, Roland Meyer and Gianluigi Zavattaro for stimulating discussions on the topics of the paper.

References

[1] Acciai, L., Boreale, M., and Zavattaro, G.: On the relationship between spatial logics and behavioral simulations. In Proc. of *FOSSACS'10*. *LNCS* 6014:146–160, (2010). Technical Report available at `http://gdn.dsi.unifi.it/~acciai/`.

[2] Acciai, L. and Boreale, M.: Deciding safety properties in infinite-state pi-calculus via behavioural types. In Proc. of *ICALP'09*. *LNCS* 5556:31–42 (2009)

[3] Acciai, L. and Boreale, M.: Spatial and behavioural Types in the pi-calculus. In Proc. of *CONCUR'08*. *LNCS* 5201:372–386 (2008). Full version in Information and Computation 208:1118–1153 (2010)

[4] Amadio, R., Meyssonnier, C.: On decidability of the control reachability problem in the asynchronous pi-calculus. Nordic Journal of Computing 9(2):70–101 (2002).

[5] Busi, N., Gabbrielli, M., Zavattaro, G.:Comparing recursion, replication, and iteration in process calculi. In Proc. of *ICALP 2004*. *LNCS* 3142:307–319 (2004)

[6] Caires, L.: Logical Semantics of Types for Concurrency. In Proc. of *CALCO'07*. *LNCS* 4624:16–35 (2007)

[7] Caires, L.: Spatial-behavioural Types, Distributed Services, and Resources. In Proc. of *TGC'06*. *LNCS* 4661:98–115 (2007)

[8] Caires, L.: behavioural and Spatial Observations in a Logic for the pi-Calculus. In Proc. of *FoSSaCS'04*. *LNCS* 2987:72–89 (2004)

[9] Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Inf. Comput. 186(2):194–235 (2003)

[10] Cardelli, L., Gordon, A.D.: Anytime, Anywhere: Modal Logics for Mobile Ambients. In Proc. of *POPL'00*, pp. 365–377 (2000)

[11] Chaki, S., Rajamani, S.K., Rehof, J.: Types as models: model checking message-passing programs. In Proc. of *POPL'02*, pp. 45–57 (2002)

[12] Engelfriet, J., Gelsema, Tj.: The decidability of structural congruence for replication restricted pi-calculus processes. Technical Report, LIACS 2004–07 (2004)

[13] Finkel, A. and Goubault-Larrecq, J.: Forward Analysis for WSTS, Part II: Complete WSTS. In Proc. of *ICALP'09*, LNCS 5556:188–199 (2009)

[14] Finkel, A. and Goubault-Larrecq, J.: Forward Analysis for WSTS, Part I: Completions. In Proc. of *STACS'09*, LIPIcs 3:433–444 (2009)

[15] Finkel, A. and Schnoebelen, Ph.: Well-Structured Transition Systems Everywhere! Theoretical Computer Science, 256(1-2):63–92 (2001)

[16] Higman, G.: Ordering by divisibility in abstract algebras. In Proc. of London Math. Soc., 2:326–366 (1952)

[17] Igarashi, A., Kobayashi, N.: A generic type system for the Pi-calculus. Theoretical Computer Science 311(1-3):121–163 (2004)

[18] Kobayashi, N., and Suto, T.: Undecidability of 2-Label BPP Equivalences and behavioural Type Systems for the Pi-Calculus. In Proc. of *ICALP 2007*. *LNCS* 4596:740–751 (2007)

[19] Kobayashi, N.: Type-based information flow analysis for the pi-calculus. Acta Inf. 42:291–347 (2005)

[20] Kruskal, J .B.: Well-quasi-ordering, the tree theorem, and Vázsonyi's conjecture. Trans. American Math. Soc. 95:210–225 (1960)

[21] Meyer, R.: A Theory of Structural Stationarity in the π-Calculus. Acta Informatica 46(2):87–137 (2009)

[22] Meyer, R.: *Structural Stationarity in the π-Calculus*. Ph.D: Thesis, Department of Computing Science, University of Oldenburg (2009)

[23] Meyer, R. and Gorrieri, R.: On the Relationship between π-Calculus and Finite Place/Transition Petri Nets. In Proc. of *CONCUR'09*. *LNCS* 5710:463–480 (2009)

[24] Milner, R.: The polyadic π-calculus: a tutorial. In Logic and Algebra of Spec., pp. 203–246 (1993)

[25] R. Milner. *Communication and Concurrency*, Prentice-Hall (1989)

[26] Sangiorgi, D.: The Name Discipline of Uniform Receptiveness. Theoretical Computer Science, 221(1-2):457–493 (1999)

[27] Shepherdson, C., Sturgis, J.E.: Computability of recursive functions. Journal of the ACM 10:217–255 (1963)

[28] Valencia, F., Aranda, J., Versari, C.: On the Expressiveness of Restriction in CCS with Replication. In Proc. of *FoSSaCS 2009*. LNCS 5504:242–256 (2009)

[29] Yoshida, N.: Graph types for Monadic Mobile Processes. In Proc. of *16th FST/TCS*. *LNCS* 1180:371–386 (1996)

$$(\text{TAU-T})\ \tau.T \overset{\tau}{\longmapsto} T \qquad (\text{OUT-T})\ \overline{a}.T \overset{\overline{a}}{\longmapsto} T \qquad (\text{INP-T})\ a.T \overset{a}{\longmapsto} T \qquad (\text{REP-T})\ !a.T \overset{a}{\longmapsto} !a.T\,|\,T$$

$$(\text{COM-T})\ \frac{T \overset{a}{\longmapsto} T' \quad S \overset{\overline{a}}{\longmapsto} S'}{T|S \overset{\tau}{\longmapsto} T'|S'} \qquad (\text{SUM-T})\ \frac{j \in I \quad \mu_j.T_j \overset{\mu_j}{\longmapsto} T_j}{\sum_{i\in I}\mu_i.T_i \overset{\mu_j}{\longmapsto} T_j} \qquad (\text{RES-T})\ \frac{T \overset{\mu}{\longmapsto} T' \quad a\#\text{n}(\mu)}{(\nu a^{\alpha})T \overset{\mu}{\longmapsto} (\nu a^{\alpha})T'}$$

$$(\text{PAR}_l\text{-T})\ \frac{T \overset{\mu}{\longmapsto} T'}{T|S \overset{\mu}{\longmapsto} T'|S} \qquad (\text{PAR}_r\text{-T})\ \frac{T \overset{\mu}{\longmapsto} T'}{S|T \overset{\mu}{\longmapsto} S|T'}$$

Table A.6: Transition relation $\overset{\mu}{\longmapsto}$ on types.

AppendixA. Standard operational semantics of CCS

The standard operational semantics of CCS terms [24] is reported in Table A.6. Throughout the paper, $\overset{\tau}{\longmapsto}$ is usually abbreviated as $\mapsto$ and $\mapsto^*$ denotes the reflexive and transitive closure of $\mapsto$.

The correspondence between $\mapsto$ and $\rightarrow$ relies on the following lemmas that can be proven by induction on the derivation of $\mapsto$ and $\rightarrow$.

Lemma 13. *$S \rightarrow S'$ implies that there exists T' such that $S' \equiv T'$ and $S \mapsto T'$.*

Lemma 14. *$T \mapsto T'$ implies $T \rightarrow T'$.*

Both lemmas are applied in the proof of Proposition 7 to guarantee the correctness of the alternative characterization of $S \models \phi$ in the cases of formulae containing dynamic connectors (i.e. $\diamondsuit^*$ and $\langle a\rangle$).

AppendixB. Proof of Proposition 7

The proof is broken into several simple lemmas. It is necessary to introduce some more terminology. Let us say that a term T is in *head-normal form* if $T = (\nu\tilde{x})\tilde{G}$. The following lemma guarantees that any term T is structural congruent to one in head-normal form $(\nu\tilde{x})\tilde{G}$. The proof is straightforward by induction on the structure of T.

Lemma 15. *For any T there exist $\tilde{G}$ and distinct $\tilde{x}$ such that $\tilde{x}\#\text{fn}(\tilde{G})$ and $T \equiv (\nu\tilde{x})\tilde{G}$.*

Given any two structurally congruent terms, their frontiers are equivalent up to renaming of bound names. In virtue of the preceding lemma, this result can be applied to any term and the corresponding head-normal form.

Lemma 16. *For any T and S such that the bound names in S are pairwise distinct,* $\mathrm{fn}(S)\#\mathrm{bn}(S)$ *and $T \equiv S$ there exists an injective substitution σ such that* $\mathrm{dom}(\sigma) = \mathrm{bn}(S)$, $\mathrm{range}(\sigma) = \mathrm{bn}(T)$ *and $\partial T \equiv \partial S\sigma$.*

In the following we write $\mathrm{n}(\sigma)$ for the set $\mathrm{dom}(\sigma) \cup \mathrm{range}(\sigma)$. The following result guarantees that satisfiability of formulae is preserved by substitutions. The proof is straightforward by induction on the structure of the formula.

Lemma 17. *Suppose ϕ is monotone and ϕ does not contain $\Diamond^*$. Consider σ such that* $\mathrm{n}(\sigma)\#\mathrm{n}(\phi)$. *Then $S \models \phi$ if and only if $S\sigma \models \phi$.*

The following two lemmas will be necessary when dealing with the case $\phi = \psi_1 \,\mathring{|}\, \psi_2$ of Proposition 7. Lemma 19 will be applied in the proof of the *if* statement, while Lemma 18 in the *only if* case.

Lemma 18. *Suppose $\phi = \psi_1 \,\mathring{|}\, \psi_2$ is monotone and plain and* $\mathrm{bn}(S)\#\mathrm{n}(\phi)$. *Then $\partial S \models \phi$ implies $S \models \phi$.*

PROOF: $\partial S \models \phi$ implies $\partial S = \tilde{F}_1 * \tilde{F}_2$ with $\tilde{F}_i \models \psi_i$, $i = 1,2$, by definition. By Lemma 15, we have $S \equiv (\nu\tilde{x})\tilde{G}$, for some $\tilde{G}$ and $\tilde{x}$ satisfying the premise of Lemma 16. Therefore, $\partial S \equiv \tilde{G}\sigma$ for some σ such that $\mathrm{dom}(\sigma) \subseteq \tilde{x}$ and $\mathrm{range}(\sigma) \subseteq \mathrm{bn}(S)$ (Lemma 16). Without loss of generality, we assume $\tilde{x}\#\mathrm{n}(\phi)$. Therefore, by $\partial S = \tilde{F}_1 * \tilde{F}_2$, $\tilde{F}_i \models \psi_i$ and $\partial S \equiv \tilde{G}\sigma$ we get $\tilde{G}\sigma = \tilde{G}_1\sigma * \tilde{G}_2\sigma$ with $\tilde{F}_i \equiv \tilde{G}_i\sigma$ and $\tilde{G}_i\sigma \models \psi_i$. Moreover, by $\mathrm{n}(\sigma) \subseteq \mathrm{bn}(S) \cup \tilde{x}$ we get $\mathrm{n}(\sigma)\#\mathrm{n}(\phi)$, and by Lemma 17, $\tilde{G}_i \models \psi_i$, $i = 1,2$. Then, $S \models \psi_1 \,\mathring{|}\, \psi_2$ follows by $S \equiv (\nu\tilde{x})\tilde{G}$ and definition of $[\![\psi_1 \,\mathring{|}\, \psi_2]\!]$. $\square$

Lemma 19. *Suppose ϕ is plain and monotone, does not contain $\Diamond^*$ and* $\mathrm{bn}(S)\#\mathrm{n}(\phi)$. *If $S \models \phi$ then for any D and $\tilde{T}$ such that $S = D[\tilde{T}]$ it holds that $\tilde{T} \models \phi$.*

PROOF: The proof is by induction on the structure of the formula ϕ. The most interesting case is when $\phi = \psi_1 \,\mathring{|}\, \psi_2$. $S \models \phi$ implies $S \equiv (\nu\tilde{x})(S_1|S_2)$, with $\tilde{x}\#\mathrm{n}(\phi)$ and $S_i \models \psi_i$. Without loss of generality, we assume that $\tilde{x}\#\mathrm{fn}(S)$. $S = D[\tilde{T}]$ implies $\partial S = \partial D[\tilde{T}] \equiv (\partial S_1 * \partial S_2)\sigma$, for some σ such that $\mathrm{n}(\sigma)\#\mathrm{n}(\phi)$ (Lemma 16). Hence, $\partial\tilde{T} = \tilde{F}_1 * \tilde{F}_2$ with $\tilde{F}_i \equiv \partial S_i\sigma$, for $i = 1,2$. By applying the induction hypothesis to $S_i \models \psi_i$ we get that $\partial S_i \models \psi_i$, for $i = 1,2$. Therefore, by Lemma 17, $\partial S_i\sigma \models \psi_i$ and $\tilde{F}_i \models \psi_i$. This implies that $\partial\tilde{T} \models \phi$, hence, by Lemma 18, $\tilde{T} \models \phi$. $\square$

We are now ready to prove the correctness of the alternative definition of $S \models \phi$.

PROOF OF PROPOSITION 7: The proof is by induction on the structure of ϕ.

$\phi ::= \overline{a} \mid a$. In both cases, $(\Rightarrow)$ and $(\Leftarrow)$, the proof relies on the definition of $\searrow_x$.

$\phi = \psi_1 \mathring{|} \psi_2$. $(\Rightarrow)$. By Lemma 19 with $\tilde{T} = \partial S$. $(\Leftarrow)$. By Lemma 18.

$\phi ::= \langle a \rangle \phi' \mid \Diamond^* \phi'$. The proof relies on the correspondence between $\rightarrow$ and $\mapsto$. Consider e.g. the case $\phi = \Diamond^* \phi'$ and suppose that $S \models \phi$. By definition, $S \rightarrow^* S'$ with $S' \models \phi'$. By applying Lemma 13 at each step of the derivation $S \rightarrow^* S'$ we get that there exists $T \equiv S'$ such that $S \mapsto T$. Hence, by definition of $\models$ and by $S' \models \phi'$, we obtain $T \models \phi'$. In the opposite case, $S \mapsto T$ and $T \models \phi'$, Lemma 14 is applied to prove that $S \models \phi$.

□

AppendixC. Proof of Proposition 9

Given a term $C[\tilde{X}]$ and two sequences of terms $\tilde{T}$ and $\tilde{S}$ such that $|\tilde{T}| = |\tilde{X}|$ and $\tilde{S} \preceq \tilde{T}$, fix any injection $f : \{1, \ldots, |\tilde{S}|\} \rightarrow \{1, \ldots, |\tilde{T}|\}$ such that $S_j \preceq T_{f(j)}$, for $1 \leq f(1) < \cdots < f(j) < \cdots < f(|\tilde{S}|) \leq |\tilde{T}|$. We write $C[\tilde{S} \lhd_f \tilde{T}]$ for the closed term obtained from $C[{}^{S_j}/_{X_{f(j)}}]_{j=1,\cdots,|\tilde{S}|}$ by pruning all sub-trees having only variables as leaves. In the following we will write $C[\tilde{S} \lhd \tilde{T}]$ for $C[\tilde{S} \lhd_f \tilde{T}]$, when f is the identity. As an example, take $\tilde{T} = T_1, T_2, T_3, T_4$, $\tilde{S} = S_1, S_2$ and suppose $f(1) = 1$ and $f(2) = 4$ (that is $S_1 \preceq T_1$ and $S_2 \preceq T_4$). Then $C[X_1, X_2, X_3, X_4]$, $C[\tilde{T}]$ and $C[\tilde{S} \lhd_f \tilde{T}]$ are depicted below from left to right.

$$\begin{tikzcd}
{} & l_1 \arrow[dl] \arrow[dr] & & & l_2 \arrow[dl] \arrow[dr] & \\
X_1 & & l_3 \arrow[dl] \arrow[dr] & l_4 & & l_5 \arrow[d] \\
& X_2 & & X_3 & & X_4
\end{tikzcd}
\qquad
\begin{tikzcd}
{} & l_1 \arrow[dl] \arrow[dr] & & & l_2 \arrow[dl] \arrow[dr] & \\
T_1 & & l_3 \arrow[dl] \arrow[dr] & l_4 & & l_5 \arrow[d] \\
& T_2 & & T_3 & & T_4
\end{tikzcd}
\qquad
\begin{tikzcd}
{} & l_1 \arrow[dl] & & l_2 \arrow[dl] \arrow[dr] & \\
S_1 & & l_4 & & l_5 \arrow[d] \\
& & & & S_2
\end{tikzcd}$$

The following lemma says that $C[\tilde{S} \lhd_f \tilde{T}]$ is embedded into $C[\tilde{T}]$. The proof is an easy application of Lemma 4.

Lemma 20. *$C[\tilde{S} \lhd_f \tilde{T}] \preceq C[\tilde{T}]$ for any injection f such that $1 \leq f(1) < \cdots < f(i) < \cdots < f(n) \leq |\tilde{T}|$ and $S_i \preceq T_{f(i)}$, for $1 \leq i \leq |\tilde{S}|$.*

The following result is a consequence of the correspondence between "$\rightarrow$" and "$\mapsto$" (see Lemma 13 and Lemma 14 in AppendixA).

Lemma 21. *(1) $T \in [\![\langle a \rangle \phi]\!]$ if and only if $T \overset{\langle a \rangle}{\longmapsto} S$ with $S \in [\![\phi]\!]$. (2) $T \in [\![\Diamond^* \phi]\!]$ if and only if $T \mapsto^* S$ with $S \in [\![\phi]\!]$.*

As a consequence of Theorem 6, Corollary 3 and the lemmas above, we can guarantee the expected result. That is, $\mathrm{Fb}_T(\cdot)$ is a (finite) basis for $[\![\cdot]\!]_T$.

PROOF OF PROPOSITION 9: The proof proceeds by showing that each term in $\mathrm{Fb}_T(\phi)$ satisfies ϕ, and that given any term $U \in [\![\phi]\!]_T$ there is a $S \in \mathrm{Fb}_T(\phi)$ such that $S \preceq U$. Together with Proposition 8, this guarantees that $\uparrow \mathrm{Fb}_T(\phi) = [\![\phi]\!]_T$.

The proof is by induction on the structure of the formula ϕ and proceeds by distinguishing the following cases:

$\phi = \mathbf{T}$**.** For each $S \in \mathrm{Fb}_T(\mathbf{T})$ it holds that $S \in [\![\mathbf{T}]\!]_T$, by definition.

Take any $U \in [\![\mathbf{T}]\!]_T$ and take any leaf $G \in \partial U \subseteq \mathrm{sub}(T)$. By Lemma 20, if $U = D[\tilde{G}]$ then $D[G \lhd \tilde{G}] \preceq U$ and $D[G \lhd \tilde{G}] \in \mathrm{Fb}_T(\mathbf{T})$ by definition (recall that $G \in \mathrm{sub}(T)$ because $U \in \mathfrak{T}_T$).

$\phi = a$**.** For each $S \in \mathrm{Fb}_T(a)$ it holds that $S \searrow_a$, and $S \in [\![a]\!]_T$, by definition.

Take any $U \in [\![a]\!]_T$. By Proposition 7, there is $G \in \partial U$ such that either $G \searrow_a$. Now, $D[G \lhd \partial U] \preceq D[\partial U] = U$ by Lemma 20 and $D[G \lhd \partial U] \in \mathrm{Fb}_T(a)$ by definition (recall that $G \in \mathrm{sub}(T)$ because $U \in \mathfrak{T}_T$).

$\phi = \overline{a}$**.** The proof is similar to the previous case.

$\phi = \phi_1 \vee \phi_2$**.** By applying the induction hypothesis to ϕ_1 and ϕ_2 we get that there exist finite basis $\mathrm{Fb}_T(\phi_1)$ and $\mathrm{Fb}_T(\phi_2)$ of $[\![\phi_1]\!]_T$ and $[\![\phi_2]\!]_T$. Therefore, for any $S \in \mathrm{Fb}_T(\phi_1 \vee \phi_2)$ either $S \in [\![\phi_1]\!]_T$ or $S \in [\![\phi_2]\!]_T$, hence $S \in [\![\phi_1 \vee \phi_2]\!]_T$ by definition.

Consider any $U \in [\![\phi_1 \vee \phi_2]\!]_T$. By definition, either $U \in [\![\phi_1]\!]_T$ or $U \in [\![\phi_2]\!]_T$. Suppose $U \in [\![\phi_1]\!]_T$. Therefore, there exists $B \in \mathrm{Fb}_T(\phi_1) \subseteq \mathrm{Fb}_T(\phi_1 \vee \phi_2)$ such that $B \preceq U$. Similar comments in case $U \in [\![\phi_2]\!]_T$.

$\phi = \phi_1 \wedge \phi_2$**.** Take any $S \in \mathrm{Fb}_T(\phi_1 \wedge \phi_2)$, by definition $S \in [\![\phi_1]\!]_T \cap [\![\phi_2]\!]_T$, hence $\mathrm{Fb}_T(\phi_1 \wedge \phi_2) \subseteq [\![\phi_1 \wedge \phi_2]\!]_T$.

Consider any $U \in [\![\phi_1 \wedge \phi_2]\!]_T$. By definition of $\mathrm{Fb}_T(\phi_1 \wedge \phi_2)$ either $U \in \mathrm{Fb}_T(\phi_1 \wedge \phi_2)$ or there is a minimal element T of $[\![\phi_1]\!]_T \cap [\![\phi_2]\!]_T$ such that $T \preceq U$.

$\phi = \psi_1 \overset{\circ}{|} \psi_2$**.** By applying the induction hypothesis to ψ_1 and ψ_2 we get that there are two finite basis $\mathrm{Fb}_T(\psi_1)$ and $\mathrm{Fb}_T(\psi_2)$ of $[\![\psi_1]\!]_T$ and $[\![\psi_2]\!]_T$.

Consider now any $S \in \mathrm{Fb}_T(\psi_1 \overset{\circ}{|} \psi_2)$. By definition, $S = D[\partial S_1 * \partial S_2]$, for some $S_i \in \mathrm{Fb}_T(\psi_i)$, $i = 1, 2$. By induction, $S_i \models \psi_i$ and by Lemma 19, $\partial S_i \models \psi_i$, for $i = 1, 2$. Hence, by Proposition 7, $S \in [\![\psi_1 \overset{\circ}{|} \psi_2]\!]_T$.

Consider any $U \in [\![\psi_1 \mathbin{\overset{\circ}{|}} \psi_2]\!]_T$. By Proposition 7, $\partial U = \tilde{G}_1 * \tilde{G}_2$ with $\tilde{G}_i \in [\![\psi_i]\!]$, for $i = 1,2$. By applying the inductive hypothesis, we get that there are $T_i \in \mathrm{Fb}_T(\psi_i)$, for $i = 1,2$, such that $T_i \preceq \tilde{G}_i$. Let $\tilde{F}_i = \partial T_i$. By $T_i \preceq \tilde{G}_i$ and Lemma 10, we get $\tilde{F}_i \preceq \tilde{G}_i$. Hence, by Lemma 20, for some injection f and sequence $\tilde{F}_1 * \tilde{F}_2$ in the shuffle of $\tilde{F}_1$ and $\tilde{F}_2$

$$D[\tilde{F}_1 * \tilde{F}_2 \lhd_f \tilde{G}_1 * \tilde{G}_2] \preceq D[\partial U] = U$$

and $D[\tilde{F}_1 * \tilde{F}_2 \lhd_f \tilde{G}_1 * \tilde{G}_2] \in \mathrm{Fb}_T(\psi_1 \mathbin{\overset{\circ}{|}} \psi_2)$ by definition.

$\phi = \langle a \rangle \phi'$. $\mathrm{Fb}_T(\langle a \rangle \phi')$ is a basis of $[\![\phi]\!]_T$ by Corollary 2.

$\phi = \Diamond^* \phi'$. $\mathrm{Fb}_T(\phi)$ is a basis of $[\![\phi]\!]_T$ by Corollary 3.

$\square$