Deciding Safety Properties in Infinite-State Pi-Calculus via Behavioural Types*

Lucia Acciai and Michele Boreale

Dipartimento di Sistemi e Informatica
Università di Firenze
{lacciai,boreale}@dsi.unifi.it

Abstract. In the pi-calculus, we consider decidability of certain safety properties expressed in a simple spatial logic. We first introduce a behavioural type system that, given a 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, we then prove that, for an interesting fragment of the considered logic, satisfiability ($T \models \phi$) is decidable for types. As a consequence of logical equivalence between types and processes, we obtain decidability of this fragment of the logic for all well-typed pi-processes.

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

1 Introduction

In recent years, *spatial logic* [6] and *behavioural type systems* [10,7,1] have gained attention as useful tools for the analysis of concurrent systems described in process calculi. Spatial logics are well suited to express properties related to concurrency and distribution, thanks to a combination of spatial and dynamic connectives. An example is the property expressing race-freedom on some channel a: "it is never the case that there are two concurrent outputs ready at channel a". behavioural type systems are used in order to obtain abstract representation of message-passing systems and simplify their analysis. In Igarashi and Kobayashi's work on generic type systems [10], 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 satisfied by a type is also satisfied by processes that inhabit that type.

In [1], we have combined ideas from spatial logics and behavioural type system into a single framework. Like in [10], the language of processes we consider is the pi-calculus, while types are ccs terms. Differently from [10], though, types of [1] account for both the behavioural *and* the spatial structure of processes. This fact allows one to establish a precise correspondence between processes and their types. This correspondence makes it possible to prove type soundness theorems holding for fairly general classes of properties, not only safety invariants, although this enhancement comes at some price in terms of flexibility of the type system w.r.t. [10]. A prominent feature of [1] is that *structural congruence* is used as a subtyping relation.

* Research partly supported by the EU within the FET-GC2 initiative, project SENSORIA.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 31–42, 2009.

A driving motivation in all the mentioned works is being able to combine type- 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: a type. Moving from processes to types certainly implies a gain in simplicity in terms of reasoning [10,1]. Unfortunately, in [11], undecidability of behavioural type systems using the simulation preorder as a sub-typing relation has been proven. The result suggests that any "reasonable" instances of the generic system of [10] based on simulation preorders might turn out to be undecidable. We may hope the situation is better for our system in [1], because this system adopts structural congruence as a subtyping relation, that, for the considered languages, is easily seen to be decidable.

In the present paper, our goal is to show decidability of a fragment of Spatial Logic over a pi-calculus with replication, introduced in Section 2. The fragment in question is expressive enough to capture interesting safety invariants. We achieve our goal in two steps. In the first one, we devise a behavioural type system whose purpose is, basically, to extract behavioural ccs types T out of given processes P. The types extracted this way are logically equivalent to the original processes. This part of the work is based on behavioural type techniques similar to those discussed in [1] and is reported in Section 3.

In the second one (Section 6), we show that it is actually decidable whether a ccs type T satisfies a formula in the fragment introduced in Section 4. This part, which is largely independent from the first one, heavily relies on the technique of *well-structured transition systems* (WSTS) introduced by Finkel and Schnoebelen [8] and overviewed in Section 5. Our result generalizes a previous result by Busi et al. [4], who had proven decidability in ccs with replication of *weak barbs*. As a corollary of the logical correspondence given by the type system, decidability of the considered logic carries over to well-typed pi-processes.

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 crucial. In particular, the WSTS technique does not apply to pi-calculus directly. The technical reason is that there is no upper bound on the nesting depth of restrictions in pi-terms as they evolve, a fact that prevents the definition of a syntax-based WQO in pi-calculus. Instead, there is such a bound for ccs.

2 Processes

The language we consider is a synchronous polyadic pi-calculus [13] with guarded summations and replications. We presuppose a countable set of *names* $\mathcal{N}$ and let $a, b, \ldots, x, \ldots$ range over names. Processes $P, Q, R, \ldots$ are defined by the grammar below

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

where $\tilde{b}$ is a tuple of names and $\mathsf{t} = (\tilde{x} : \tilde{\mathsf{t}}')T$ is a *channel type* where: $(\tilde{x} : \tilde{\mathsf{t}})$ is a binder with scope T; $\tilde{x}$ and $\tilde{\mathsf{t}}$ represent the formal parameters and types of objects carried by the channel; T is a process type (see Section 3) prescribing a usage of those parameters. The calculus is equipped with standard notions of free and bound names ($\mathrm{fn}(\cdot)$, $\mathrm{bn}(\cdot)$). Notice that we let $\mathrm{fn}((\nu b : \mathsf{t})P) = (\mathrm{fn}(P) \cup \mathrm{fn}(\mathsf{t})) \setminus \{b\}$ and that terms are identified up to alpha-equivalence, defined as usual. To prevent arity mismatch, we will only consider well-sorted terms in some fixed sorting system (see e.g. [13]), and call $\mathcal{P}$ the resulting set of *processes*.

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

$$(\nu y)\mathbf{0} \equiv \mathbf{0} \quad (P|Q)|R \equiv P|(Q|R) \quad P|Q \equiv Q|P \quad P|\mathbf{0} \equiv P \quad (\nu x:\mathsf{t})P|Q \equiv (\nu x:\mathsf{t})(P|Q) \text{ if } \tilde{x} \notin \mathrm{fn}(Q)$$

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

$$(\textsc{com})\frac{\alpha_l = a(\tilde{x}) \quad \alpha'_n = \overline{a}\langle\tilde{b}\rangle \quad l \in I \quad n \in J}{\sum_{i\in I}\alpha_i.P_i | \sum_{j\in J}\alpha'_j.Q_j \rightarrow P_l[\tilde{b}/\tilde{x}]|Q_n} \quad (\textsc{tau})\frac{j \in I \quad \alpha_j = \tau}{\sum_{i\in I}\alpha_i.P_i \rightarrow P_j} \quad (\textsc{res})\frac{P \rightarrow P'}{(\nu x:\mathsf{t})P \rightarrow (\nu x:\mathsf{t})P'}$$

$$(\textsc{rep})\frac{\alpha_n = \overline{a}\langle\tilde{b}\rangle \quad n \in J}{!a(\tilde{x}).P | \sum_{j\in J}\alpha_j.Q_j \rightarrow !a(\tilde{x}).P | P[\tilde{b}/\tilde{x}]|Q_n} \quad (\textsc{par})\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \quad (\textsc{struct})\frac{\begin{array}{c}P \equiv Q \\ Q \rightarrow Q' \quad Q' \equiv P'\end{array}}{P \rightarrow P'}$$

In the following, we write $\mathbf{0}$ for the empty summation, 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 $(\nu\tilde{b}:\tilde{\mathsf{t}})P$, 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. Concerning Table 1, note that we have dropped the law $(\nu x:\mathsf{t})(\nu y:\mathsf{t}')P = (\nu y:\mathsf{t}')(\nu x:\mathsf{t})P$, which allows one to swap restrictions: the reason is that swapping t and t', which may contain free names, would require unpleasant side conditions. The rules in Table 2 are standard.

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

3 Type System

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

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

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 for processes P and their types T scope extrusion, hence structural congruence, works in the same manner in both P and in T (see [1] for more details). 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 unnecessary. We will often denote guarded summations and replications by the letters $G, F, \ldots$. Notions of free and bound names ($\mathrm{fn}(\cdot)$ and $\mathrm{bn}(\cdot)$), alpha-equivalence, structural congruence and reduction for types parallel those of processes.

Typing rules. 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*: a finite partial map from names to channel types. We write $\Gamma \vdash a : \mathsf{t}$ if

Table 3. Typing rules for the local system

$$\text{(T-Inp)}\ \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathfrak{t}})T \quad \mathrm{fn}(\tilde{\mathfrak{t}}) \cup \mathrm{fn}(T) \setminus \tilde{x} = \mathfrak{a} \quad \Gamma, \tilde{x} : \tilde{\mathfrak{t}} \vdash P : T|T' \quad \tilde{x} \notin \mathrm{fn}(T')}{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T'} \qquad \text{(T-Out)}\ \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathfrak{t}})T \quad \Gamma \vdash \tilde{b} : \tilde{\mathfrak{t}} \quad \Gamma \vdash P : S}{\Gamma \vdash \overline{a}\langle \tilde{b} \rangle.P : \overline{a}.(T[\tilde{b}/\tilde{x}]\,|\,S)}$$

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

$$\text{(T-Sum)}\ \frac{|I| \neq 1 \quad \forall i \in I :\ \Gamma \vdash \alpha_i.P_i : \mu_i.T_i}{\Gamma \vdash \sum_{i \in I} \alpha_i.P_i : \sum_{i \in I} \mu_i.T_i} \qquad \text{(T-Rep)}\ \frac{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T}{\Gamma \vdash\, !a(\tilde{x}).P :\, !a^{\mathfrak{a}}.T} \qquad \text{(T-Tau)}\ \frac{\Gamma \vdash P : T}{\Gamma \vdash \tau.P : \tau.T}$$

$a \in \mathrm{dom}(\Gamma)$ and $\Gamma(a) = \mathfrak{t}$. We say that a context is *well-formed* if whenever $\Gamma \vdash a : (\tilde{x} : \tilde{\mathfrak{t}})T$ then $\mathrm{fn}(T, \tilde{\mathfrak{t}}) \subseteq \tilde{x} \cup \mathrm{dom}(\Gamma)$. 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 a little help from a context Γ prescribing channel usage. See [2] for further details. In the following we say that a process P *is* Γ*-well-typed* if $\Gamma \vdash P : T$ for some $T \in \mathcal{T}$.

Results. This paragraph introduces the main properties of the type system. Theorem 1 and 2 guarantee the reduction-based correspondence between processes and the corresponding 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. Finally, Proposition 2 guarantees decidability of $\vdash$.

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 $\alpha ::= a \mid \overline{a}$, implies $T \searrow_\alpha$; 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 $\Gamma, \tilde{a} : \tilde{\mathfrak{t}} \vdash R : S$; 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 $i = 1, 2$; vice-versa for T and P.*

Proposition 2. *Let Γ be a context. It is decidable whether P is Γ-well-typed.*

4 Shallow Logic and Type-Process Correspondence

The logic for the pi-calculus we introduce below can be regarded as a fragment of Caires and Cardelli's Spatial Logic [6]. In [1] we have christened this fragment *Shallow Logic*, as it allows us 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 allows one to "look underneath" prefixes.

Definitions. The set $\mathcal{F}$ of *Shallow Logic* formulae $\phi, \psi, \ldots$ is given by the grammar $\phi ::= \mathbf{T} \mid a \mid \overline{a} \mid \phi|\phi \mid \neg\phi \mid \mathrm{H}^*\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid \diamond^* \phi$, where $a \in \mathcal{N}$.

The set of logical operators includes spatial ($a, \overline{a}, |, \mathrm{H}^*$) as well as dynamic ($\langle a\rangle, \Diamond^*$) connectives, beside the usual boolean connectives, including a constant $\mathbf{T}$ for "true". We have included both disjunction and conjunction to present more smoothly "monotone" properties, that is, properties whose satisfaction is preserved when adding "more structure" to terms. The names of a formula ϕ, written $\mathrm{n}(\phi)$, are defined as expected. The interpretation of $\mathcal{F}$ over processes and types is given below.

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

We let $\mathcal{U}$ be the set including all processes and all types. We write $A \models \phi$ if $A \in [\![\phi]\!]$, where $A \in \mathcal{U}$. Connectives and spatial modalities are interpreted as usual. Concerning the dynamic part, $\langle a\rangle\phi$ 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. This can also be written as $\Box^*\neg\psi$, where $\Box^* = \neg\Diamond^*\neg$.

Example 1. The following formulae define properties depending on generic names, a and l. $\mathit{NoRace}(a) \triangleq \neg\Diamond^*\mathrm{H}^*(\overline{a}|\overline{a})$ says that it will never be the case that there are two concurrent outputs competing for synchronization on a. $\mathit{Linear}(a) \triangleq \neg\Diamond^*\langle a\rangle\Diamond^*\langle a\rangle$ says that it is never the case that a is used more than once in a computation. In $\mathit{Lock}(a,l) \triangleq \neg\Diamond^*\mathrm{H}^*(l|\langle a\rangle)$, a represents a shared resource and l a lock: this formula says that it is never the case that the resource a is acquired in the presence of l, that is, without prior acquisition of the lock.

Logical correspondence between processes and types. 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.

Theorem 3 (type-process correspondence). *Suppose $\Gamma \vdash P : T$. Let ϕ be any formula. Then $P \models \phi$ if and only if $T \models \phi$.*

This correspondence can be enhanced by the next result, saying that, under certain circumstances, 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", by means of the $\downarrow_{\tilde{x}}$ operator (see [2] for the details), the free

names of the type that are not found in the formula. Moreover, if this masking produces a top-level sub-term in the type with no free name, this term can be safely discarded.

Proposition 3. *(a) Suppose* $\Gamma \vdash P : T$ *and let* ϕ *be an anti-monotone formula with* $\mathrm{n}(\phi) \subseteq \tilde{x}$*. 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* $T \models \phi$*.*

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

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

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

5 A Well-Structured Transition System for Behavioural Types

Background. We review below some background material about well-structured transition systems [8] and well quasi-ordering over trees and forests.

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*) if for any sequence of elements of* S*,* $(s_i)_{i\geq 0}$*, there are* i *and* j*, with* $i < j$*, s.t.* $s_i \preceq s_j$*.*

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

Definition 3 (WSTS, [8]). *A* well-structured transition system *(*WSTS *for short) is a pair* $\mathcal{W} = (\preceq, Tr)$ *where: (a)* $Tr = (S, \rightarrow)$ *is a finitely-branching transition system, and (b)* $\preceq$ *is a* WQO *over* S *that is compatible with* $\rightarrow$*; that is: whenever* $s_1 \preceq s_2$ *and* $s_1 \rightarrow s_1'$ *then there is* s_2' *such that* $s_2 \rightarrow s_2'$ *and* $s_1' \preceq s_2'$*.*

Otherwise said, a WSTS is a finitely-branching transition system equipped with a WQO that is a simulation relation. Let 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 $\{s \in S \mid s' \preceq s \text{ for some } s' \in I\}$. The set $\uparrow \{s\}$ will be 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, $\mathrm{Pred}(I)$, be the set $\{s \in S \mid s \rightarrow s' \text{ for some } s' \in I\}$ and the set of *predecessors* of I, $\mathrm{Pred}^*(I)$, be $\{s \in S \mid s \rightarrow^* s' \text{ for some } s' \in I\}$. We say $\mathcal{W}$ has an *(effective) pred-basis* if there is a (computable) function $\mathrm{pb}(\cdot) : S \rightarrow 2^S$ such that for each $s \in S$, $\mathrm{pb}(s)$ is a finite basis of $\uparrow \mathrm{Pred}(\uparrow s)$.

Proposition 4 ([8]). *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* $I \subseteq S$*, returns a finite basis of* $\mathrm{Pred}^*(\uparrow I)$*.*

The above proposition entails decidability of a number of reachability-related problems in WSTS's (see [8]). 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 $s \geq s'$ for some $s' \in B$.

We will also rely upon some definitions and results on trees. Let L be a set. We define *ordered forests* $\mathcal{F}, \mathcal{G}, ...$ 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 $\mathcal{F}_1, ..., \mathcal{F}_k$ are forests ($k \geq 0$) then the sequence $(a_1, \mathcal{F}_1) \cdots (a_k, \mathcal{F}_k)$, with $a_i \in L$, is a forest with roots $a_1, ..., a_k$. A forest of the form $(a, \mathcal{F})$ is called an *(ordered, rooted) tree*. A tree of the form (a, ϵ) is called a *leaf*. The multiset of leaves occurring in $\mathcal{F}$ is denoted by $\mathrm{L}(\mathcal{F})$, while the corresponding set is denoted $\mathrm{l}(\mathcal{F})$. 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, defined as expected; the height of a leaf is 0. We will often use the familiar pictorial representation of trees and forests. The following theorem provides us with a WQO on forests, hence on trees, called *rooted tree embedding*. One can think of this WQO as saying that $\mathcal{F}_1 \preceq \mathcal{F}_2$ if $\mathcal{F}_1$ can be mapped into a sub-forest of $\mathcal{F}_2$, provided that the mapping respects the roots of $\mathcal{F}_1$. The proof of the theorem can be given relying on a result on WQO on sequences due to Higman [9] (see [4] for a similar proof); or even generalizing the Kruskal tree theorem [12] to forests, again via Higman's lemma.

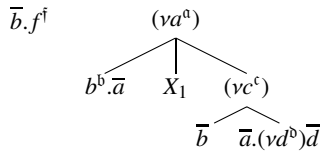
Theorem 4 (rooted tree embedding). *Let $\mathfrak{F}$ be the set of all forests with labels in a certain nonempty set. Consider the following* QO *$\preceq$ over $\mathfrak{F}$: $(a_1, \mathcal{F}_1) \cdots (a_k, \mathcal{F}_k) \preceq (b_1, \mathcal{G}_1) \cdots (b_h, \mathcal{G}_h)$ iff there are distinct indices $1 \leq i_1 < \cdots < i_k \leq h$ s.t. for each j, $1 \leq j \leq k$, $a_j = b_{i_j}$ and $\mathcal{F}_i \preceq \mathcal{G}_{i_j}$. Let $\mathfrak{G} \subseteq \mathfrak{F}$ be such that: there is a finite bound on the height of the forests in $\mathfrak{G}$ and there is a finite L s.t. the labels of all forests in $\mathfrak{G}$ are included in L. Then $\preceq$ is a* WQO *on* $\mathfrak{G}$.

A WSTS *for behavioural types.* Let $(X_i)_{i \geq 1}$ be an infinite sequence 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 – where each variable occurs at most once in a term and only in the scope of restrictions 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 other words, we are considering open terms representing *static contexts*, with the variables X_i acting as the "holes". We let C range over $\mathfrak{T}$, reserving the letters S, T for the subset of closed terms (types) and will sometimes write $C[\tilde{X}]$ to indicate that C's variables are *exactly* $\tilde{X} = (X_{i_1}, ..., X_{i_k})$. 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_j} with T_j in $C[\tilde{X}]$.

Each term C can be seen as a forest $\mathcal{F}_C$, with restrictions $(\nu a^{\mathfrak{a}})$ as internal labels and either guarded summations/replications G or variables X_i as leaves, and parallel composition | interpreted as concatenation[1], as shown in the following example.

Example 3. Consider the term $C = \overline{b}.f^{\mathfrak{f}} | (\nu a^{\mathfrak{a}})(b^{\mathfrak{b}}.\overline{a} | X_1 | (\nu c^{\mathfrak{c}})(\overline{b} | \overline{a}.(\nu d^{\mathfrak{d}})\overline{d}))$. The forest $\mathcal{F}_C$ associated to C is depicted below.

[1] More formally, each C is mapped to a forest $\mathcal{F}_C$ as follows: $\mathcal{F}_{X_i} = (X_i, \epsilon)$, $\mathcal{F}_G = (G, \epsilon)$, $\mathcal{F}_{T|S} = \mathcal{F}_T \cdot \mathcal{F}_S$ and $\mathcal{F}_{(\nu a^{\mathfrak{a}})T} = ((\nu a^{\mathfrak{a}}), \mathcal{F}_T)$.



Via this correspondence, we can identify terms with forests, and in what follows we shall not notationally distinguish between the two. In the following we will sometimes use a ground version of the function $\mathrm{L}(\cdot)$, written $\mathrm{GL}(\cdot)$, returning the multiset of *ground*, i.e. non-variables, leaves of a term. In the example above: $\mathrm{L}(C) = \{\!|\, \overline{b}.f^{\dagger},\ b^{b}.\overline{a},\ X_1,\ \overline{b},\ \overline{a}.(\nu d^{b})\overline{d} \,|\!\}$ and $\mathrm{GL}(C) = \{\!|\, \overline{b}.f^{\dagger},\ b^{b}.\overline{a},\ \overline{b},\ \overline{a}.(\nu d^{b})\overline{d} \,|\!\}$. The QO defined in the statement of Theorem 4 is also inherited by $\mathfrak{T}$, that is, we can set: $C \preceq C'$ iff $\mathcal{F}_C \preceq \mathcal{F}_{C'}$. To make this a *well* QO, we have to restrict ourselves to some subset of $\mathfrak{T}$ with bounded height and set of labels. This will be obtained by tailoring out of $\mathfrak{T}$ a superset of all terms that are reachable from 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):

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

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: this is of course a finite set. Finally, we denote by $\mathrm{res}(C)$ the set of restrictions (νa^{a}) occurring in C. The set of terms we are interested in is defined below.

Definition 4 ($\mathfrak{T}_T[\tilde{X}]$)**.** *Fix a type T and a set of variables $\tilde{X} = (X_{j_1}, ..., X_{j_k})$, then*

$$\mathfrak{T}_T[\tilde{X}] \triangleq \{\, C \in \mathfrak{T} \;\big|\; \mathrm{l}(C) \subseteq \mathrm{sub}(T) \cup \tilde{X},\ \mathrm{res}(C) \subseteq \mathrm{res}(T),\ \mathrm{dp}(C) \leq \mathrm{dp}(T) \,\}\,.$$

In the following, we abbreviate $\mathfrak{T}_T[\tilde{X}]$ as $\mathfrak{T}_T$ when $\tilde{X} = \epsilon$. Consider now the rooted-tree embedding $\preceq$ described above, 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. Terms in $\mathfrak{T}_T[\tilde{X}]$, by definition, have bounded height: indeed, for any $C \in \mathfrak{T}_T[\tilde{X}]$, we have $\mathrm{h}(C) \leq \mathrm{dp}(C) \leq \mathrm{dp}(T)$. Moreover, they are built using a finite set of labels: $\tilde{X} \cup \mathrm{res}(T) \cup \mathrm{sub}(T)$. Theorem 4 ensures then that $\preceq$ is a WQO 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 traditional ccs transition relation over closed terms, denoted here $\overset{\mu}{\mapsto}$; in particular, we shall write $\overset{\tau}{\mapsto}$ as $\mapsto$. The relation $\mapsto$ is preferable to $\rightarrow$ in the present context, because it avoids alpha-equivalence, structural congruence and is finitely branching for the considered fragment. In Section 6, we shall argue that $\mapsto$ is equivalent to $\rightarrow$ for the purpose of defining the satisfaction relation $S \models \phi$. The set $\mathfrak{T}_T$ of closed terms enjoys the following crucial properties, which can be easily inferred by induction on the structure of the term. Note in particular that, by the second property, the restriction nesting depth of any term is *not* increased by $\overset{\mu}{\mapsto}$. This is a crucial property that does not hold in the pi-calculus. E.g. (type annotations omitted):

$$(\nu b_1)\overline{a}\langle b_1\rangle \,|\, !a(y).(\nu b_2)(y.b_2\,|\overline{c}\langle b_2\rangle) \,|\, !c(x).\overline{a}\langle x\rangle \;\rightarrow^{*}$$
$$(\nu b_1)(\nu b_2)(b_1.b_2|(\nu b_3)(b_2.b_3|\cdots(\nu b_{n+1})(b_n.b_{n+1}|\overline{c}\langle b_{n+1}\rangle)\cdots))|!a(y).(\nu b_2)(y.b_2|\overline{c}\langle b_2\rangle)|!c(x).\overline{a}\langle x\rangle.$$

Proposition 6. *(1) For any $S \in \mathfrak{T}_T$ and S', $S \overset{\mu}{\mapsto} S'$ implies that $S' \in \mathfrak{T}_T$. (2) The relation $\preceq$ is a simulation relation over $\mathfrak{T}_T$. As a consequence: (3) For any T, let Tr be the transition system $(\mathfrak{T}_T, \overset{\tau}{\mapsto})$. Then $\mathcal{W}_T \triangleq (\preceq, Tr)$ is a* WSTS.

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

6 Decidability

Decidability of a fragment of Shallow Logic relies on 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, show that $\mathcal{W}_T$ has an effective pred-basis. Moreover, we have to show that each denotation $[\![\phi]\!]$ can be presented via an effectively computable finite basis playing the role of "I" in the proposition.

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 built that have the same ground leaves as C, but possibly more holes, up to 2. 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 $\succeq$. In what follows, we shall also admit as a possible context C the 0-hole forest ϵ, which gives rise only to the decomposition $S = \epsilon[S]$.

Definition 5 (pred-basis). *Let T be a type, $S \in \mathfrak{T}_T$ and C, C' range over $\mathfrak{T}_T[X_1, X_2]$.*

$$\mathrm{pb}_T(S) \triangleq \bigcup_{S=C[\tilde{U}]} \{C'[\tilde{G}] \in \mathfrak{T}_T \mid C' \succeq C, \mathrm{GL}(C') = G\mathrm{L}(C), \tilde{G} \subseteq \mathrm{sub}(T),\ C'[\tilde{G}] \mapsto\succeq S\}$$

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

Theorem 5. *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. (Outline) Effectiveness has already been discussed. Moreover, 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 accommodate 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 $V = C[G_1, G_2] \mapsto C[S_1, S_2] \succeq S$. It is then easy to prove that $S = C''[\tilde{S}']$, with $C \succeq C''$ and $(S_1, S_2) \succeq \tilde{S}'$. It is possible to build out of C'' a 2-holes context $C' \in \mathfrak{T}_T[X_1, X_2]$ s.t. $C \succeq C' \succeq C''$. Take $U = C'[G_1, G_2]$.

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

Corollary 1. *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 1. Consider the labelled version of the reduction relation, $\overset{\langle\lambda\rangle}{\longmapsto}$, $\lambda ::= a|\epsilon$. For any fixed label $\langle a\rangle$, Corollary 1 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.

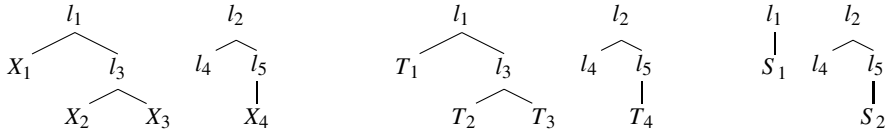
Applying Proposition 4, we get the result we were after.

Corollary 2. *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* $\mathrm{Pred}_T^*(\uparrow I)$.

Finite bases for plain formulae. Our first task is showing that, for certain formulae ϕ, the satisfaction relation $S \models \phi$ can be defined relying solely on $\mapsto$ and on context decomposition, in particular, with no reference to structural congruence and $\rightarrow$. In the proposition below, we show that this is indeed possible for *plain* formulae.

Definition 6 (plain formulae). *We say a formula* ϕ *is* plain *if it does not contain* $\Diamond^*$ *underneath* H^*.

Let us say a context C is *pure* if $\mathrm{l}(C) \subseteq (X_i)_{i\geq 1}$; we let D range over pure contexts; e.g. $D = (\nu a)(X_1|X_2)|X_3$ is pure. Given a context $C[\tilde{X}, \tilde{Y}]$ and two sequences $\tilde{G}$, $\tilde{F}$ s.t. $|\tilde{X}| = |\tilde{G}|$ and $|\tilde{Y}| = |\tilde{F}|$, we say C *links* $\tilde{G}$ *and* $\tilde{F}$ if there are an internal node (νa^{a}) of C seen as a forest, $X_i \in \tilde{X}$ and $Y_j \in \tilde{Y}$ such that both X_i and Y_j are in the scope of this node, and $a \in \mathrm{fn}(G_i) \cap \mathrm{fn}(F_j)$. Given a sequence $\tilde{G}$, we denote by $\prod \tilde{G}$ the parallel composition of the terms in $\tilde{G}$, in some arbitrary order. Given a term $C[\tilde{T}]$ and $\tilde{S} \preceq \tilde{T}$, fix any injection $f : \{1,\ldots,k\} \to \{1,\ldots,|\tilde{T}|\}$ ($k = |\tilde{S}|$) such that $S_j \preceq T_{f(j)}$, for $1 \leq f(1) < \cdots < f(k) \leq |\tilde{T}|$. We write $C[\tilde{S} \triangleleft_f \tilde{T}]$ for the closed term obtained from $C[{}^{S_j}/_{X_{f(j)}}]_{j=1,\cdots,k}$ by pruning all sub-trees having only variables as leaves. In the following we will write $C[\tilde{S} \triangleleft \tilde{T}]$ for $C[\tilde{S} \triangleleft_f \tilde{T}]$, 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 $f(2) = 4$. C, $C[\tilde{T}]$ and $C[\tilde{S} \triangleleft_f \tilde{T}]$ are depicted below.



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 assumed to be included in* $\mathrm{sub}(T)$.

$S \models \langle a\rangle\phi$ *iff* $\exists U : S \overset{\langle a\rangle}{\longmapsto} U$ *and* $U \models \phi$ $\qquad$ $S \models \Diamond^*\phi$ *iff* $\exists U : S \mapsto^* U$ *and* $U \models \phi$

$S \models a$ *iff* $\exists D, \tilde{G}, U : S = D[\tilde{G}]$ *and for some* $G \in \tilde{G}$: $G = !a^{\mathrm{a}}.U$ *or* $a^{\mathrm{a}}.U$ *is a summand of* G

$S \models \overline{a}$ *iff* $\exists D, \tilde{G}, U : S = D[\tilde{G}]$ *and for some* $G \in \tilde{G}$: $\overline{a}.U$ *is a summand of* G

$S \models \mathrm{H}^*\phi$ *iff* $\exists D, \tilde{G} : S = D[\tilde{G}]$ *and* $\prod \tilde{G} \models \phi$

$S \models \phi_1|\phi_2$ *iff* $\exists D, \tilde{G}_1, \tilde{G}_2 : S = D[\tilde{G}_1, \tilde{G}_2]$, D *not linking* $\tilde{G}_1$ *and* $\tilde{G}_2$, $D[\tilde{G}_i \triangleleft \tilde{G}_1, \tilde{G}_2] \models \phi_i$ $(i = 1,2)$

As discussed at the beginning of this section, in order to take advantage of Corollary 2, we have to show that each set $[[\phi]]$, or, more accurately, each set $[[\phi]]_T \triangleq [[\phi]] \cap \mathfrak{T}_T$, can be presented via an effectively computable finite basis in $\mathcal{W}_T$. We define this basis below, by induction on the structure of ϕ: the $\Diamond^*$ and $\langle a\rangle$ cases take advantage of the pred-basis functions defined in the last paragraph, the other cases basically follow the corresponding cases of the previous proposition 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 a set $I \subseteq \mathfrak{T}_T$, we denote by minimal($I$) the set of minimal elements in I, w.r.t. the WQO $\preceq$. For any ordered sequence $\tilde{G}$, we denote by $\langle \tilde{G} \rangle$ the multiset obtained if ignoring order.

Definition 7 (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 assumed to be included in* $\mathrm{sub}(T)$.

$$\mathrm{Fb}_T(a) \triangleq \{D[G] \in \mathfrak{T}_T \mid G = !a^{\mathfrak{a}}.U \text{ or } a^{\mathfrak{a}}.U \text{ is a summand of } G, \text{ for some } U\}$$

$$\mathrm{Fb}_T(\overline{a}) \triangleq \{D[G] \in \mathfrak{T}_T \mid \overline{a}.U \text{ is a summand of } G, \text{ for some } U\}$$

$$\mathrm{Fb}_T(\phi_1|\phi_2) \triangleq \bigcup_{S_1 \in \mathrm{Fb}_T(\phi_1), S_2 \in \mathrm{Fb}_T(\phi_2)} \{D[\tilde{G}_1, \tilde{G}_2] \in \mathfrak{T}_T \mid \text{for } i = 1,2 : \langle \tilde{G}_i \rangle = \mathrm{L}(S_i), D \text{ does not link } \tilde{G}_1 \text{ and } \tilde{G}_2 \text{ and } D[\tilde{G}_i \triangleleft \tilde{G}_1, \tilde{G}_2] \succeq S_i\}$$

$$\mathrm{Fb}_T(\mathrm{H}^*\phi) \triangleq \bigcup_{S \in \mathrm{Fb}_T(\phi)} \{D[\tilde{G}] \in \mathfrak{T}_T \mid \langle \tilde{G} \rangle = \mathrm{L}(S)\} \qquad \mathrm{Fb}_T(\mathbf{T}) \triangleq \{D[G] \in \mathfrak{T}_T \mid G \in \mathrm{sub}(T)\}$$

$$\mathrm{Fb}_T(\phi_1 \vee \phi_2) \triangleq \mathrm{Fb}_T(\phi_1) \cup \mathrm{Fb}_T(\phi_2) \qquad \mathrm{Fb}_T(\langle a \rangle \phi) \triangleq \mathrm{pb}_T^{\langle a \rangle}(\mathrm{Fb}_T(\phi))$$

$$\mathrm{Fb}_T(\phi_1 \wedge \phi_2) \triangleq \mathrm{minimal}([\![\phi_1]\!] \cap [\![\phi_2]\!]) \qquad \mathrm{Fb}_T(\Diamond^* \phi) \triangleq \mathrm{pb}_T^*(\mathrm{Fb}_T(\phi))$$

Note that minimal($[\![\phi_1]\!] \cap [\![\phi_2]\!]$) is finite: if not, one would find an infinite sequence of pairwise incomparable elements, thus violating the condition of WQO.

Theorem 6. *Consider T and ϕ like in Definition 7. Then* $\mathrm{Fb}_T(\phi)$ *is a finite basis for $[\![\phi]\!]_T$, that is* $\uparrow \mathrm{Fb}_T(\phi) = [\![\phi]\!]_T$. *Moreover,* $\mathrm{Fb}_T(\cdot)$ *is computable.*

Proof. (Outline) The first part of the statement is quite easy, indeed one inclusion, $\uparrow \mathrm{Fb}_T(\phi) \subseteq [\![\phi]\!]_T$, is valid by construction, while the opposite direction is proved by induction on ϕ, relying on the characterization of $\models$ provided by Proposition 7 for the spatial and dynamic connectives, the boolean ones being trivial to handle. Proving that $\mathrm{Fb}_T(\cdot)$ is computable is more difficult, because of the clause for conjunction. This is accommodated by introducing an effective operator $\|$ that over-approximates the "minimal" operator: $\mathrm{Fb}_T(\phi_1) \,\|\, \mathrm{Fb}_T(\phi_2) \supseteq \mathrm{minimal}([\![\phi_1]\!]_T \cap [\![\phi_2]\!]_T)$. The operator $\|$ produces a finite set of terms by appropriately merging terms, seen as forests, drawn from $\mathrm{Fb}_T(\phi_1)$ and $\mathrm{Fb}_T(\phi_2)$. We refer the reader to [2] for the details.

By virtue of the above theorem, we can decide if $S \models \phi$, with $S \in \mathfrak{T}_T$, by checking if there is $U \in \mathrm{Fb}_T(\phi)$ s.t. $S \succeq U$: since $\preceq$ is decidable, this can be effectively carried out, and we obtain Corollary 3. Finally, Corollary 4 is a consequence of Proposition 2.

Corollary 3 (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 4 (decidability on pi-processes). *Let Γ be a context. Given a Γ-well-typed P and ϕ plain and (anti-)monotone, it is decidable whether $P \models \phi$.*

7 Conclusion and Related Work

We have proven the decidability of a fragment of Spatial Logic that includes interesting safety properties for a class of infinite-control pi-processes. The proof relies heavily on both behavioural type systems [10,7,1] and well-structured transition system techniques [8]. Implementation issues are not in the focus of this paper. Whether a practical

algorithm may be obtained or not from the theoretical discussion presented here is an interesting topic, that is left for future work.

Our proof of decidability generalizes the result in [4] that "weak" barbs $\diamond^* a$ are decidable in ccs with replication. Variations and strengthening of these results have recently been obtained by Valencia et al. [14]. It is worth to notice that weak barbs are *not* decidable in the pi-calculus, [3]. On the other hand, our results show that they become decidable when restricting to well-typed pi-processes.

Also related to our approach is [5], where Caires proves that model-checking Spatial Logic formulae for bounded pi-calculus processes, and in particular finite-control processes, is decidable. Note that the class of processes we have considered here properly includes bounded processes.

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

References

1. Acciai, L., Boreale, M.: Spatial and behavioral types in the pi-calculus. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 372–386. Springer, Heidelberg (2008) (Full version submitted, 2009)
2. Acciai, L., Boreale, M.: Deciding safety properties in infinite-state pi-calculus via behavioural types. Extended version, `http://gdn.dsi.unifi.it/~acciai/papers/decFull.pdf`
3. 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)
4. Busi, N., Gabbrielli, M., Zavattaro, G.: Comparing recursion, replication, and iteration in process calculi. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 307–319. Springer, Heidelberg (2004)
5. Caires, L.: Behavioural and Spatial Observations in a Logic for the pi-Calculus. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 72–89. Springer, Heidelberg (2004)
6. Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Inf. Comput. 186(2), 194–235 (2003)
7. Chaki, S., Rajamani, S.K., Rehof, J.: Types as models: model checking message-passing programs. In: Proc. of POPL 2002, pp. 45–57 (2002)
8. Finkel, A., Schnoebelen, P.: Well-Structured Transition Systems Everywhere! Theoretical Computer Science 256(1-2), 63–92 (2001)
9. Higman, G.: Ordering by divisibility in abstract algebras. Proc. London Math. Soc. 2, 326–366 (1952)
10. Igarashi, A., Kobayashi, N.: A generic type system for the Pi-calculus. Theoretical Computer Science 311(1-3), 121–163 (2004)
11. Kobayashi, N., Suto, T.: Undecidability of 2-Label BPP Equivalences and behavioural Type Systems for the Pi-Calculus. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 740–751. Springer, Heidelberg (2007)
12. Kruskal, J.B.: Well-quasi-ordering, the tree theorem, and Vázsonyi's conjecture. Trans. American Math. Soc. 95, 210–225 (1960)
13. Milner, R.: The polyadic π-calculus: a tutorial. In: Logic and Algebra of Spec., pp. 203–246 (1993)
14. Valencia, F., Aranda, J., Versari, C.: On the Expressive Power of Restriction and Priorities in CCS with Replication. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 242–256. Springer, Heidelberg (2009)