A Spatial Equational Logic for the Applied π-calculus

Jules Villard[†] Étienne Lozes[†] Ralf Treinen[‡]

† LSV, ENS Cachan, CNRS
61 av. du pdt Wilson, 94230 Cachan, France
{lozes,villard}@lsv.ens-cachan.fr

‡ PPS, CNRS UMR 7126 & Université Paris Diderot
Case 7014, 75205 Paris cedex 13, France
treinen@pps.jussieu.fr

Abstract

Spatial logics have been proposed to reason locally and modularly on algebraic models of distributed systems. In this paper we investigate a spatial equational logic (AπL) whose models are processes of the applied π-calculus, an extension of the π-calculus allowing term manipulation modulo a predefined equational theory, and wherein communications are recorded as active substitutions in a frame. Our logic allows us to reason locally either on frames or on processes, thanks to static and dynamic spatial operators. We study the logical equivalences induced by various relevant fragments of AπL, and show in particular that the whole logic induces a coarser equivalence than structural congruence. We give characteristic formulae for this new equivalence as well as for static equivalence on frames. Going further into the exploration of AπL's expressivity, we also show that it can eliminate standard term quantification, and that the model-checking problem for the adjunct-free fragment of AπL can be reduced to satisfiability of a purely first-order logic of a term algebra.

1 Introduction

A spatial equational logic. Spatial logics [10, 4] have been proposed to reason locally and modularly on algebraic models of distributed systems.

They share a lot of similarities with BI and Separation Logic [19] which both have resource sharing and local reasoning as central concepts. Two essential connectives in these logics are parallel composition and adjunct. The parallel composition $A \mid B$, which corresponds to the separating conjunction $A * B$, asserts that A and B hold on disjoint components, whereas the adjunct $A \triangleright B$, which corresponds to the *magic wand* $A \multimap B$, states that when a component verifying A is plugged in then B holds for the resulting system. In this paper we investigate a spatial equational logic: term equality $M = N$ is defined by an equational theory $\mathcal{E}$ and a frame ϕ, *i.e.* a finite set of local axioms, and $*$ splits the frame into smaller sets of local axioms. For instance, $(f(x) = c) * (f(x) \neq c)$ states that some subset of the local axioms implies $f(x) = c$ and that $f(x) = c$ does not hold in $\mathcal{E}$, whereas $(f(x) \neq c) \multimap \perp$ states that $f(x) = c$ is a consequence of $\mathcal{E}$ without any local axiom.

This spatial equational logic, written AπL below, naturally arises as the spatial logic of the applied π-calculus. Applied π-calculus (Aπ) [1] is an extension of π-calculus [17] where processes may communicate terms through channels. One peculiar aspect of Aπ is that the processes keep the history of their past communications in *active substitutions*, written $\{^{M}/_{x}\}$, acting as local axioms that extend the equational theory. For instance, a process $\nu n.\, \bar{a}\langle f(n)\rangle.P$ may evolve into $\nu n.\, P \mid \{^{f(n)}/_{x}\}$ for some fresh x. This allows any context aware of x to use it as an alias for $f(n)$, even if n is a restricted name that is not directly visible to the context. In a cryptographic interpretation, one might then be interested in knowing whether or not n can be reconstructed from the public knowledge of $f(n)$, which might be the case if for instance $\forall t. g(f(t)) = t$ holds in $\mathcal{E}$.

Deducibility can be quite simply formalized in AπL: in the previous example, we will say that some restricted name n is deducible in the frame $\phi = \nu n.\, \{^{f(n)}/_{x}\}$ if $\exists t.\, \mathrm{H}n.\, (t = n)$ holds in ϕ. In this formula, $\exists$ is the standard term quantifier: $\exists t.\, A$ holds if there is a capture-avoiding term M such that $A[t \leftarrow M]$ holds, and H is the hidden name quantifier [4]: $\mathrm{H}n.\, A$ holds for $\nu n.\, P$ if A holds for P.

Motivations. In AπL, the active and static aspects of Aπ are addressed by two distinct families of connectives: $\emptyset, *, \multimap$ for frames, and $0, ¦, \triangleright, \Diamond$ for plain processes. The static part of the logic allows one to treat the frame as a shared knowledge on which one may reason locally. For instance, $(x = M) * ((x = M') \multimap \mathsf{Attack})$ expresses that a process becomes vulnerable if an attacker picks an emitted message M and replaces it by M'. The dynamic part of the logic allows one to reason about the execution in isolation of some partners of a given protocol, or in a context which abides by some policy of the protocol: formulae $\mathsf{Client} ¦ \mathsf{Server}$, or $\mathsf{Client} \rhd \mathsf{Attack}$, would describe a protocol with a client and a server, or a server that might be attacked by a context acting as the specification of a honest client. The semantics of $A ¦ B$ requires some attention: every part of the process is ran with its own copy of the frame, while some nonces might be shared between one side of the plain process and the frame, but not between the two sides of the plain process; for instance, $\nu n.\,(\phi \mid \mathit{Server}(n) \mid \mathit{Client})$ will satisfy $\mathsf{Server} ¦ \mathsf{Client}$ if $\nu n.\,(\phi \mid \mathit{Server}(n))$ satisfies Server and $\nu n..(\phi \mid \mathit{Client})$ satisfies Client.

In the context of process algebrae, spatial logics raise the question of what spatial equivalence is defined by means of logical equivalence for several fragments of the logic. The fragments considered are said to be static if they do not include the (strong) time modality, and extensional if the intensional connectives $|, \mathrm{H}$ are dropped and the logic only keeps adjuncts. Each of these fragments defines a logical equivalence, which provides alternative characterizations of some operational or syntactic equivalences. For instance, in the case of π-calculus, logical equivalence is structural congruence [12, 8], whereas extensional logical equivalence is barbed-congruence [11].

Spatial logics introduce many new connectives, which may give them a somewhat baroque aspect. Several expressiveness results propose how to reduce these logics, *e.g.* with adjuncts or quantifiers elimination procedures [15, 5], but which may not be directly adaptable to any model.

Contributions. Our first contribution is the characterization of the logical equivalences of the static, static extensional, and dynamic intensional fragments. Static fragments turn out to play similar roles as in the case of the π-calculus: static intensional equivalence is proved to coincide with structural congruence for frames, whereas static extensional logical equivalence coincides with static equivalence, the observational equivalence for frames [1]. In both cases, characteristic formulae are derivable. However, logical equivalence for the dynamic intensional fragment is proved to be coarser than $\equiv$, albeit finer than standard observational equivalence. We point some admissible axioms for logical equivalence that show that our logic defines a strictly coarser equivalence than $\equiv$, and prove this axiomatization complete at least for the equational theory of finite trees.

Our second contribution is a quantifier elimination technique which shows that standard term quantifiers $\exists t.\,A$ can be mimicked by other connectives. The proof is based on an embedding of a valuation into the process that is similar to the technique used to eliminate name quantification in the π-calculus [5].

Our third contribution is a reduction of the model-checking problem of the adjunct-free fragment of AπL to the satisfiability problem of a first-order logic deprived of all spatial connectives. Indeed, the target logic is a first-order logic of a term algebra featuring several equivalence relations; we show that the first-order theory of this structure is decidable under some assumptions on the equational theory.

Related work. Blanchet *et al.* have developed a decision procedure for a finer notion than barbed-congruence, and have implemented it in the tool ProVerif [2]. Several contributions were made by Kremer *et al.* about the formalization of non-trivial cryptographic protocols and advanced security properties, such as receipt freeness and coercion resistance in electronic voting protocols [14]. In these cases, the security property is not mere resistance to an either active or passive attacker, since the attacker is constrained and has to follow some specification. We believe that this kind of problems is a challenging issue for applications of spatial logics.

Spatial logics for process algebrae with explicit resources have been first studied by Pym [18]. The idea of distributing assertions about knowledge in space using spatial logics has been explored by Mardare [16]. Hüttel *et al.* gave a logical characterization and characteristic formulae for static equivalence [13], but had to make some assumptions on the equational theory.

Structure of the paper. In Section 2, we collect all the necessary background on Aπ, and define our process compositions $*$ and ¦. Section 3 introduces AπL; Sections 4 and 5 present the characterizations of the logical equivalences for the static and dynamic fragments respectively. Section 6 establishes the quantifiers elimination property, and Section 7 explains the model-checking technique by reduction to a simple first-order logic.

Acknowledgments. We acknowledge Jean Goubault-Larrecq, Steve Kremer and Stéphanie Delaune for valuable discussions.

2 Applied π-calculus

2.1 Terms

The grammar of applied π-calculus processes relies on the definition of a set of *terms* along with an *equational theory*, thus letting the user decide which cryptographic primitives the calculus will use. The set of terms is constructed using disjoint infinite sets $\mathcal{V}$ and $\mathcal{N}$ of respectively variables and names, and a finite *signature* Σ, which is a set of functions, each with its arity (constants have arity 0). Its grammar is as follows ($ar(f)$ is the arity of f):

$M, N ::=$	terms
$\quad x, y, z, \ldots$	variables
$\quad a, b, c, m, n, s, \ldots$	names
$\quad f(M_1, \ldots, M_{ar(f)})$	function application

Notations 2.1 We will use the letters a, b, c, n, m, s to refer to elements of $\mathcal{N}$, x, y, z for elements of $\mathcal{V}$ and u, v, w for "meta-variables" which may belong either to $\mathcal{N}$ or $\mathcal{V}$. We will write M, N for terms.

These terms are equipped with an equivalence relation $\mathcal{E}$ called an equational theory on Σ, where membership of a couple (M, N) of terms is written $\mathcal{E} \vdash M = N$, or simply $M = N$ if $\mathcal{E}$ is clear from context. This relation must be closed under substitution of terms for variables or names ($M_1 = M_2$ implies $M_1[u{\leftarrow}N] = M_2[u{\leftarrow}N]$) and context application ($N_1 = N_2$ implies $M[x{\leftarrow}N_1] = M[x{\leftarrow}N_2]$). $\mathit{fn}(M)$ and $\mathit{fv}(M)$ are respectively the sets of free names and free variables of M, defined as usual, and $\mathit{fnv}(M) \triangleq \mathit{fn}(M) \cup \mathit{fv}(M)$. A term with no free variables is said to be *ground*.

2.2 Processes

Applied π-calculus extends the standard π-calculus with primitives for term manipulation, namely *active substitutions* and term communications. The grammar of processes is split into two levels: the *plain* processes, written $P^p, Q^p, \ldots$ which account for the dynamic part, and the *extended* ones, also referred to simply as "processes" and written $P, Q, \ldots$ which extend the former with a *static* part. It is presented in Figure 1. Note that replication $!P^p$ is not part of our setting.

This grammar allows communications of two kinds, that may not interfere: communications of names behave as in the standard π-calculus whereas communications of terms may interact with active substitutions and conditionals. Names are thus allowed to serve both as channels through which communications may occur and as atoms on which to build terms, without the need to rely on a type system to ensure the validity of communications, as it is the case in the original applied π-calculus.

From now on, we will only consider extended processes whose active substitutions are cycle-free. Furthermore, we will always assume that there is at most one active substitution for each variable, and *exactly one* if the variable is restricted. A process following these constraints will be called well-formed, all others being ill-formed.

Figure 1 Grammar of processes

$P^p, Q^p ::=$	plain processes
$\quad \mathbf{0}$	null process
$\quad P^p \mid Q^p$	parallel composition
$\quad \nu a. P^p$	name restriction
$\quad a^{ch}(n).P^p$	name input
$\quad \bar{a}^{ch}\langle n\rangle.P^p$	name output
$\quad a(x).P^p$	term input
$\quad \bar{a}\langle M\rangle.P^p$	term output
$\quad \mathit{if}\ M = N\ \mathit{then}\ P^p\ \mathit{else}\ Q^p$	conditional
$P, Q ::=$	(extended) processes
$\quad P^p$	plain process
$\quad \{^M/_x\}$	active substitution
$\quad P \mid Q$	parallel composition
$\quad \nu a. P$	name restriction
$\quad \nu x. P$	variable restriction

Notations 2.2 The set of free names (resp. variables) of a process P is defined as usual and written $\mathit{fn}(P)$ (resp. $\mathit{fv}(P)$), with $\mathit{fn}(\{^M/_x\}) \triangleq \mathit{fn}(M)$ (resp. $\mathit{fv}(\{^M/_x\}) \triangleq \{x\} \cup \mathit{fv}(M)$), and with both restrictions and both inputs being binders. We write $\mathit{fnv}(P)$ for the set $\mathit{fn}(P) \cup \mathit{fv}(P)$.

Compositions of active substitutions of the form $\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_n}/_{x_n}\}$ will be written $\{^{M_1 \ldots M_n}/_{x_1 \ldots x_n}\}$, or $\{^{\widetilde{M}}/_{\tilde{x}}\}$, and referred to using σ, τ. Trailing $\mathbf{0}$'s in processes will often be omitted, as well as null *else* branches in conditionals.

2.3 Operational semantics

The structural congruence relation $\equiv$ identifies processes that can be obtained one from another by mere rewriting. It is the smallest equivalence relation on well-formed extended processes that is stable by α-conversion on both names and variables and by application of contexts, and that satisfies the rules described in Figure 2. Here, a context is an extended process with a hole instead of a plain process. This hole can be filled with any extended process provided the resulting extended process is well-formed.

According to the rules for active substitutions, two structurally congruent processes may not have the same set of free names or variables. Thus, we define the closures $\overline{\mathit{fn}}(P), \overline{\mathit{fv}}(P), \overline{\mathit{fnv}}(P)$ of these sets up to structural congru-

Figure 2 Rules for structural congruence

$$
\begin{array}{ll}
\textsc{Par-}\mathbf{0} & P \equiv P \mid \mathbf{0} \\
\textsc{Par-A} & P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\
\textsc{Par-C} & P \mid Q \equiv Q \mid P
\end{array}
$$

$$
\begin{array}{ll}
\textsc{New-}\mathbf{0} & \nu n.\, \mathbf{0} \equiv \mathbf{0} \\
\textsc{New-C} & \nu u.\, \nu v.\, P \equiv \nu v.\, \nu u.\, P \\
\textsc{New-Par} & P \mid \nu u.\, Q \equiv \nu u.\, (P \mid Q) \\
 & \qquad\qquad \text{if } u \notin \mathit{fnv}(P)
\end{array}
$$

$$
\begin{array}{ll}
\textsc{Alias} & \nu x.\, (\{^M/_x\} \mid P) \equiv P[x \leftarrow M] \\
\textsc{Subst} & \{^M/_x\} \mid P^p \equiv \{^M/_x\} \mid P^p[x \leftarrow M] \\
\textsc{Rewrite} & \{^M/_x\} \equiv \{^N/_x\} \text{ if } \mathcal{E} \vdash M = N
\end{array}
$$

ence, as well as the corresponding sets for terms:

$$\overline{\mathit{fn}}(P) \triangleq \bigcap_{Q \equiv P} \mathit{fn}(Q) \qquad \overline{\mathit{fv}}(P) \triangleq \bigcap_{Q \equiv P} \mathit{fv}(Q) \qquad \overline{\mathit{fnv}}(P) \triangleq \overline{\mathit{fn}}(P) \cup \overline{\mathit{fv}}(P)$$

$$\overline{\mathit{fn}}(M) \triangleq \bigcap_{N = M} \mathit{fn}(N) \qquad \overline{\mathit{fv}}(M) \triangleq \bigcap_{N = M} \mathit{fv}(N) \qquad \overline{\mathit{fnv}}(M) \triangleq \overline{\mathit{fn}}(M) \cup \overline{\mathit{fv}}(M)$$

It is worth mentioning that rules Alias and Subst are not the ones from the original applied π-calculus, which are:

$$
\begin{array}{ll}
\textsc{Alias'} & \nu x.\, \{^M/_x\} \equiv \mathbf{0} \\
\textsc{Subst'} & \{^M/_x\} \mid P \equiv \{^M/_x\} \mid P[x \leftarrow M]
\end{array}
$$

This is to allow active substitutions to apply to other active substitutions only if their domain contains only restricted variables, and only apply to plain processes otherwise. As such, the rule Alias is still valid in our setting, whereas Subst' is restricted to plain processes only. This does not change neither the reduction relation presented below, nor the fact that one can always rewrite an extended process to get rid of its variable restrictions. Indeed, a process P can always be rewritten into a set of restricted names $\tilde{n}$, a public composition of active substitutions σ and a public plain process Q: $P \equiv \nu\tilde{n}.\, (\sigma \mid Q)$.

Finally, the reduction rules are presented in Figure 3. Internal reduction $\rightarrow$ is the smallest relation that satisfies those rules and that is closed by structural congruence and by application of evaluation contexts, that is to say contexts where the hole is in place of an extended process instead of a plain one.

Figure 3 Reduction rules

COMM-T	$\bar{a}\langle x\rangle.P \mid a(x).Q \rightarrow P \mid Q$
COMM-C	$\bar{a}^{ch}\langle m\rangle.P \mid a^{ch}(n).Q \rightarrow P \mid Q[n\leftarrow m]$
THEN	$\mathit{if}\ M = M\ \mathit{then}\ P\ \mathit{else}\ Q \rightarrow P$
ELSE	$\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q \rightarrow Q$
	(when M and N are ground and $\mathcal{E} \nvdash M = N$)

2.4 Frames

A *frame* is an extended process built up from active substitutions and the null process only. The *domain* $dom(\phi)$ of a frame ϕ is the set of variables upon which the active substitutions of ϕ act. The frame $\phi(P)$ of a process P is P in which every plain process embedded in P is set to $\mathbf{0}$. Similarly, the plain process $(P)^p$ associated with P is obtained by mapping every substitution over non-restricted variables to $\mathbf{0}$. Frames behave consistently w.r.t. structural congruence, as expressed by the following lemma:

Lemma 2.3 *If $P \equiv Q$ then $\phi(P) \equiv \phi(Q)$.*

Proof Suppose a proof of $P \equiv Q$. A proof of $\phi(P) \equiv \phi(Q)$ can be obtained by setting every plain process embedded into P or Q to $\mathbf{0}$ into the proof, and then suppressing the portions of the proof tree that mention them (and which are now irrelevant). □

However, it does not hold that $P \equiv Q$ implies $(P)^p \equiv (Q)^p$: considering $P = \{^y/_x\} \mid \bar{a}\langle x\rangle$ and $Q = \{^y/_x\} \mid \bar{a}\langle y\rangle$, $P \equiv Q$ holds but not $\bar{a}\langle x\rangle \equiv \bar{a}\langle y\rangle$.

Definition 2.4 (Closed frame) *ϕ is* closed *when $fv(\phi) \subseteq dom(\phi)$.*

Definition 2.5 *We say that two terms M and N are equal in the frame ϕ, and write $\phi \vdash M = N$ when there exists a set of names $\tilde{n}$ and a substitution σ (i.e. a public frame) such that $\phi \equiv \nu\tilde{n}.\sigma$, $M\sigma = N\sigma$ and $\tilde{n} \cap fn(M, N) = \emptyset$.*

We say that two terms are equal in the process P, and write $P \vdash M = N$, if they are equal in $\phi(P)$.

Definition 2.6 *Two closed frames ϕ and ψ are said to be* statically equivalent*, written $\phi \approx_s \psi$, when $dom(\phi) = dom(\psi)$ and, for all terms M and N, $\phi \vdash M = N$ if and only if $\psi \vdash M = N$.*

Two processes are statically equivalent, written $P \approx_s Q$, when their frames are.

2.5 Additional operators

When splitting up a process P into a parallel composition of two subprocesses $P_1 \mid P_2$, two very different operations are performed, as both the dynamic and the static part of the process are split up into two extended processes. This does not match our intuition of local reasoning for processes: consider a protocol $\nu n, n'. \left(\{^{\mathrm{ck}(n,n')}/_{x}\} \mid A(x,n) \mid B(x,n')\right)$ where Abelard and Héloïse share a compound key $\mathrm{ck}(n,n')$ generated from a nonce n of A and n' of B. Then a specification of the form $A \mid B$ would not work since this process is atomic. To overcome this situation we have broken down the parallel composition into two finer-grained operators: the first one, $\mid$, keeps the same frame while splitting up the plain process and conversely the second one, $*$, keeps the same plain process while splitting up the frame.

Because the strict separation between the frame and the plain process of an extended process might not be syntactically obvious, these two operations need to be defined up to structural congruence. For this purpose, we first define them for a restricted class of processes for which composition using "$\mid$" or "$*$" is obvious, and then extend the definitions to every couple of processes which may be rewritten into two such processes.

Definition 2.7 *Given ϕ a frame, P_1, P_2 plain processes, and $\tilde{n}_1, \tilde{n}_2$ such that $\{\tilde{n}_1\} \cap fn(P_2) = \{\tilde{n}_2\} \cap fn(P_1) = \emptyset$, we let:*

$$\nu\tilde{n}_1. ((\nu\tilde{n}_2. \phi) \mid P_1) \mid \nu\tilde{n}_2. ((\nu\tilde{n}_1. \phi) \mid P_2) \triangleq \nu\tilde{n}_1\tilde{n}_2. (\phi \mid P_1 \mid P_2).$$

We write $P \leftrightarrow P_1 \mid P_2$ if there are P', P_1', P_2' such that $P \equiv P'$, $P_1 \equiv P_1'$, $P_2 \equiv P_2'$, and $P' = P_1' \mid P_2'$.

Definition 2.8 *Given $\phi_1, \phi_2, P, \tilde{n}_1, \tilde{n}_2$ such that $\{\tilde{n}_1\} \cap fn(\phi_2) = \{\tilde{n}_2\} \cap fn(\phi_1) = \emptyset$ and $\phi_1 \mid \phi_2$ is well-formed, we let:*

$$\nu\tilde{n}_1. (\phi_1 \mid \nu\tilde{n}_2. P) * \nu\tilde{n}_2. (\phi_2 \mid \nu\tilde{n}_1. P) \triangleq \nu\tilde{n}_1\tilde{n}_2. (\phi_1 \mid \phi_2 \mid P).$$

We write $P \leftrightarrow P_1 * P_2$ if there are P', P_1', P_2' such that $P \equiv P'$, $P_1 \equiv P_1'$, $P_2 \equiv P_2'$, and $P' = P_1' * P_2'$.

Remark 2.9 Every process P is structurally congruent to one built up using frame ($*$) and plain process ($\mid$) compositions in lieu of the parallel one ($\mid$).

Remark 2.10 Formally, $P \leftrightarrow P_1 * P_2$ is a ternary relation, and for some P_1, P_2, one may have $P \leftrightarrow P_1 * P_2$, $P' \leftrightarrow P_1 * P_2$ for some non congruent P, P'.

Ternary relations also arise in the relational models of BI, or in context logics. Albeit not a composition law, $*$ projects as a composition on frames: $\phi(P * Q) \equiv \phi(P) \mid \phi(Q)$. Though not composition laws, $*$ and $\mid$ still enjoy some nice properties. First, $*$ projects as a composition on frames: $\phi(P * Q) \equiv \phi(P) \mid \phi(Q)$. Second, $\mid$ is a composition law for a restricted class of equational theories (see appendix). We say that an equational theory $\mathcal{E}$ is *dually symmetric* if for all terms M, for all name permutations σ such that $\mathcal{E} \vdash M\sigma = M$, and for all n, $\mathcal{E} \vdash M[n \leftrightarrow \sigma(n)] = M$. If $\mathcal{E}$ is dually symmetric, then $\mid$ defines a composition law:

Proposition 2.11 *Let $\mathcal{E}$ be a dually symmetric equational theory. If P, Q, P', Q' are extended processes such that $P \equiv P'$, $Q \equiv Q'$, and such that $P \mid Q$, $P' \mid Q'$ are defined, then $P \mid Q \equiv P' \mid Q'$.*

The proof of this proposition can be found in Appendix A. The dual symmetry property holds for instance for the free algebra theory, for the theory of commutative monoids, but it does not hold for a theory with pairing and an AC operator $\oplus$. We expect it should still be a reasonable hypothesis for some case studies.

3 A spatial logic for the applied π-calculus

3.1 Syntax and semantics

We assume an infinite set $\mathcal{TV}$ of *term variables*, distinct from $\mathcal{V}$, ranged over by $t, t', \ldots$, and we write U, V for terms that possibly contain these term variables. We call $\mathcal{L}_{\text{spat}}$ the set of formulae defined by the following grammar:

$$\begin{aligned} A, B ::= {} & U = V \mid \neg A \mid A \wedge B \mid \exists t.\, A \mid \Diamond A \mid \text{H}u.\, A \mid ©u \mid \mathbf{0} \mid A \mid B \mid \emptyset \mid A * B \\ & \mid A \oslash u \mid A \rhd B \mid A \twoheadrightarrow\!\!* B \end{aligned}$$

$U = V$ is the equality of terms w.r.t. the current frame; negation and conjunction are classical; $\exists t.\, A$ is term quantification and $\Diamond A$ is the strong reduction modality. $\text{H}n.\, A$ is the hidden name quantification and $\text{H}x.\, A$ the hidden variable quantification. We use the same operator for both of these, as well as for the © and $\oslash$ operators, because our convention on naming lets us do so unambiguously. $©u$ means u is free in the process; $\mathbf{0}$ is the null plain process, $A \mid B$ is plain process composition; $\emptyset$ is the empty frame; $A * B$ is frame composition; $A \oslash u$ is hiding. $A \rhd B$ and $A \twoheadrightarrow\!\!* B$ are guarantee operators and are the adjuncts of (respectively) $A \mid B$ and $A * B$.

Notations 3.1 Fragments of the above logics will be defined using the notation $\mathcal{L} \setminus \mathcal{O}$ or $\mathcal{L} \cup \mathcal{O}$, where $\mathcal{L}$ is an already defined logical fragment and $\mathcal{O}$ a set of operators. $\mathcal{L} \setminus \mathcal{O}$ is the fragment $\mathcal{L}$ restricted to operators not in $\mathcal{O}$ and $\mathcal{L} \cup \mathcal{O}$ is $\mathcal{L}$ augmented with operators of $\mathcal{O}$.

There are two main fragments to consider when studying spatial logics: the extensional one, which lets one observe a process via its interactions with some (possibly constrained) environment, and the intensional one, which lets one explore the very structure of the process.

Here, the intensional fragment corresponds to the whole logic $\mathcal{L}_{\text{spat}}$. To define the extensional fragment, we make use of the fresh name and variable quantifications written $\text{И}u.\, A$, which can be defined using other operators of the logic, as shown in Figure 5 at the end of this section[1]: $\mathcal{L}_{\text{ext}} \triangleq \mathcal{L}_{\text{spat}} \setminus \{\text{H}, ©, \mathbf{0}, \shortmid\!\shortmid, *\} \cup \{\text{И}\}$.

Moreover, we define static fragments of the intensional and extensional fragments as: $\mathcal{L}_i^{\text{stat}} \triangleq \mathcal{L}_i \setminus \{\mathbf{0}, \shortmid\!\shortmid, \rhd, \diamondsuit, ©\}$ for i in $\{\text{int}, \text{ext}\}$.

The operators' semantics, close to the one defined by Caires and Cardelli for the π-calculus [4], is given by a satisfaction relation described in Figure 4 with judgements of the form $P, v \vDash A$ between a process P, a spatial formula A and a valuation v. Valuations assign terms $\widetilde{M}$ to all the free variables $\tilde{t}$ of the formula and are written $\{\widetilde{t \to M}\}$ when $v(t_i) = M_i$ for all $i \in \{1 \ldots n\}$. We write $v\{t \to M\}$ for the valuation v whose domain has been extended to t with $v(t) = M$. Finally, when A is closed and the valuation is empty, judgements are written $P \vDash A$.

Lemma 3.2 *The satisfaction relation is stable under structural congruence: $P \vDash A$ and $P \equiv Q$ implies $Q \vDash A$.*

Lemma 3.3 *For every formula A of $\mathcal{L}^{stat}$, $P \vDash A$ iff $\phi(P) \vDash A$.*

Boolean operators are assumed to bind more tightly than compositions and adjunctions, which in turn bind more tightly than every other operator.

Derived connectives $\forall t$, $\vee$, $\Rightarrow$ and $U \neq V$ are defined as usual, as well as the sets of free names and free variables of a formula A, written $fn(A)$ and $fv(A)$.

3.2 Examples

Some basic formulae, which will be useful in the following sections, are shown in Figure 5. Their meanings for a process P is that there must exist a process

[1] It is a known fact, which holds also for our logic, that spatial logics may be defined equivalently using either И and ® or H and ©. We chose the latter form to make capture-avoidance w.r.t. term quantification immediate.

Figure 4 Satisfaction relation

$$
\begin{array}{lcl}
P, v \vDash U = V & \Leftrightarrow & P \vdash Uv = Vv \\
P, v \vDash \neg A & \Leftrightarrow & P, v \nvDash A \\
P, v \vDash A_1 \wedge A_2 & \Leftrightarrow & P, v \vDash A_1 \text{ and } P, v \vDash A_2 \\
P, v \vDash \exists t.\, A & \Leftrightarrow & \exists M.\, P, (v\{t \rightarrow M\}) \vDash A \\
P, v \vDash \Diamond A & \Leftrightarrow & \exists P'.\, P \rightarrow P' \text{ and } P', v \vDash A \\
P, v \vDash \mathrm{H}u.\, A & \Leftrightarrow & \exists u' \notin \mathit{fnv}(P, v, A).\, \exists P'.\, P \equiv \nu u'.\, P' \text{ and } P', v \vDash A[u \leftarrow u'] \\
P, v \vDash \copyright u & \Leftrightarrow & u \in \overline{\mathit{fnv}}(P) \\
P, v \vDash \mathbf{0} & \Leftrightarrow & (P)^p \equiv \mathbf{0} \\
P, v \vDash A_1 \mathop{\mid} A_2 & \Leftrightarrow & \exists P_1, P_2.\, P \equiv P_1 \mathop{\mid} P_2,\ P_1, v \vDash A_1 \text{ and } P_2, v \vDash A_2 \\
P, v \vDash \emptyset & \Leftrightarrow & \phi(P) \equiv \mathbf{0} \\
P, v \vDash A_1 * A_2 & \Leftrightarrow & \exists P_1, P_2.\, P \equiv P_1 * P_2,\ P_1, v \vDash A_1 \text{ and } P_2, v \vDash A_2 \\
P, v \vDash A \oslash n & \Leftrightarrow & \nu n.\, P, v \vDash A \\
P, v \vDash A \oslash x & \Leftrightarrow & x \in \mathit{dom}(P) \text{ and } \nu x.\, P, v \vDash A \\
P, v \vDash A \rhd B & \Leftrightarrow & \forall Q, R.\, (R \leftrightarrow P \mathop{\mid} Q \text{ and } Q, v \vDash A) \text{ implies } R, v \vDash B \\
P, v \vDash A \multimap\!\!* \, B & \Leftrightarrow & \forall Q, R.\, (R \leftrightarrow P * Q \text{ and } Q, v \vDash A) \text{ implies } R, v \vDash B
\end{array}
$$

Figure 5 Basic formulae

$$\top \triangleq \mathbf{0} \vee \neg\mathbf{0} \qquad \bot \triangleq \neg\top \qquad A[B] \triangleq (A \wedge \mathbf{0}) \mathop{\mid} ((B \wedge \emptyset) * \top)$$

$$A \blacktriangleright B \triangleq \neg(A \rhd \neg B) \qquad A \,\bar{\multimap}\!\!*\, B \triangleq \neg(A \multimap\!\!* \neg B) \qquad \mathbf{1} \triangleq \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mathop{\mid} \neg\mathbf{0})$$

$$\mathbb{I} \triangleq \neg\emptyset \wedge \neg(\neg\emptyset * \neg\emptyset) \qquad \mathsf{public} \triangleq \neg\mathrm{H}n.\, \copyright n \qquad \mathsf{single} \triangleq \mathbf{1} \wedge \emptyset \wedge \mathsf{public}$$

$$\text{И}n.\, A \triangleq \mathrm{H}n.\, A \wedge \neg\copyright n \qquad \text{И}x.\, A \triangleq \mathrm{H}x.\, (\mathbb{I} \wedge \mathsf{public}) * (A \wedge \neg\copyright x)$$

$Q \equiv P$ such that:

- $\top$: nothing is required; $\bot$ is always false;
- $A[B]$: $\phi(Q)$ verifies A and $(Q)^p$ verifies B;
- $A \blacktriangleright B$ (this is the dual of $\triangleright$): there are Q', R such that $R \leftrightarrow Q \mid Q'$, $Q' \vDash A$ and $R \vDash B$, and similarly for $-\!\!\circledast$;
- $\mathbf{1}$ (resp. $\mathbb{I}$): $(Q)^p$ (resp. $\phi(Q)$) is not null and cannot be divided into two non-null processes;
- public: Q has no bound name: $\forall n, Q'. Q \equiv \nu n. Q' \Rightarrow n \notin \overline{fn}(Q')$;
- single: Q is guarded, either by a communication or by a conditional construct.
- $\text{И}u. A$: there exists $u' \notin fnv(P, A)$ such that $P \vDash A[u \leftarrow u']$;

As already mentioned, one may formalize some standard secrecy properties in AπL. For instance, $\exists t. \forall t'. \mathrm{H}n. (t = \mathrm{pk}(n) \ \wedge \ t' \neq \mathrm{sk}(n))$ asserts that some public key is publicly available whereas its associated private key is secret.

4 Logical characterization of processes

In this section we give an axiomatization of the logical equivalence induced by the dynamic intensional fragment. We moreover prove it to be complete, with characteristic formulae, when the equational theory $\mathcal{E}$ is the one of finite trees.

We write $P =_L Q$ if P, Q cannot be discriminated by $\mathcal{L}_{\text{spat}}$ formulae. We say that a process P is non branching if it does not contain a subprocess

$$\textit{if } M = N \textit{ then } P \textit{ else } Q,$$

and we say it is in the π-calculus if moreover the only communications are channel communications.

4.1 Channel communications

Theorem 4.1 *For every π-calculus process P, there is a formula $\mathbf{F}_P \in \mathcal{L}_{spat}$ such that for every extended process Q, $Q \models \mathbf{F}_P$ if and only if $Q \equiv P$.*

The formula $\mathbf{F}_P$ is defined by induction on P. The induction step for channel communications calls on formulae $\mathsf{com}^{ch}.A$ such that P satisfies $\mathsf{com}^{ch}.A$ if P is $com^{ch}.P'$ and P' satisfies A. Figure 6 summarizes the construction.

Figure 6 Formulae for names communications

$$\mathsf{atom}(a,b) \triangleq ©a \wedge ©b \wedge (\mathsf{single} \blacktriangleright \Diamond \mathbf{0}) \qquad \mathsf{test}^{ch}(a,b) \triangleq \mathsf{atom}(a,b) \wedge \neg \mathrm{H}x.\, \top[©x]$$

$$\mathsf{in}^{ch}(a,b).A \triangleq \mathsf{single} \wedge \neg\Diamond\top \wedge Иb.\, \mathsf{test}^{ch}(a,b) \blacktriangleright \Diamond A$$

$$\mathsf{out}^{ch}(a,b).A \triangleq \mathsf{single} \wedge Иb'.\, (\mathsf{in}^{ch}(a,c).\mathsf{test}^{ch}(b',c) \triangleright \Diamond(\mathsf{test}^{ch}(b',b) \,¦\, A))$$

Formula $\mathsf{test}^{ch}(a,b)$ characterizes the processes of the form $\bar{a}\langle b\rangle$ or $\bar{b}\langle a\rangle$, and is then used to characterize communication primitives.

4.2 Term communications and conditionals

Characterizing term communications and conditionals is much harder due to the equational theory. For this section only, we first assume that we work with the equational theory of finite trees, and then discuss some intuitions on the general case.

The first widget we need is a formula $©^{\mathsf{s}}_{=}x$ that characterizes the processes where x does not occur as a strict subterm of one of the terms appearing in the plain part of the process. It is then quite simple to characterize processes $\bar{a}\langle x\rangle$. From this, we may also characterize $\bar{a}\langle M\rangle$ by revealing x in $\nu x.\, (\{^M/_x\} \mid \bar{a}\langle x\rangle)$, and from there all communications can be characterized. Figure 7 presents the whole construction.

Theorem 4.2 *Let $\mathcal{E}$ be the theory of finite trees. Then for every non-branching process P, there is a formula $\mathbf{F}_P \in \mathcal{L}_{spat}$ such that for all extended processes Q, $Q \models \mathbf{F}_P$ if and only if $Q \equiv P$.*

We can use the same technique to characterize the test of conditionals, as any conditional is congruent to $\nu x, y.\, (\{^M/_x\} \mid \{^N/_y\} \mid \mathit{if}\ x = y\ \mathit{then}\ P\ \mathit{else}\ Q)$. However, we need to be slightly more careful when we reveal variables x, y, since we do not want to let them appear in P, Q. For this, we need to characterize the variables that appear inside one of the branches of a conditional, but not inside the test itself: this is the purpose of formula $©^{\mathsf{br}}x$. We have that $P \vDash ©^{\mathsf{br}}x$ iff $P \equiv \mathit{if}\ M = N\ \mathit{then}\ P_1\ \mathit{else}\ P_2$ with $x \in \mathit{fv}(P) \setminus \mathit{fv}(M, N)$.

Figure 7 Formulae for terms communications

$$\copyright^{\mathsf{s}}_{=} x \triangleq \top[\copyright x] \wedge \neg \mathrm{H} z.\, (z \neq x \wedge \copyright x)[\copyright z]$$

$$\mathsf{out}(a, x) \triangleq \mathsf{single} \wedge \copyright a \wedge \copyright^{\mathsf{s}}_{=} x \wedge (\mathsf{single} \blacktriangleright \Diamond \mathbf{0})$$

$$\mathsf{in}(a, x).A \triangleq \mathsf{single} \wedge \neg\Diamond\top \wedge \text{И} x.\, \mathsf{out}(a, x) \triangleright \Diamond A$$

$$\mathsf{out}(a, M).A \;\triangleq \mathrm{H} x.\, (x = M) \left[\begin{array}{l} \text{И} b.\, \mathsf{in}(a, y).\mathsf{out}(b, y) \triangleright \\ \Diamond\big(\mathsf{out}(b, x) \mid (A \wedge \neg\copyright x)\big) \end{array} \right]$$

Figure 8 Formulae for conditionals

$$\mathsf{if} \triangleq \mathsf{single} \wedge \top \mathrel{\overline{-\!\!*}} \Diamond\top \qquad \copyright^{\mathsf{br}} x \triangleq \mathsf{if} \wedge \mathrm{H} z, z'.\, \top[(\neg\copyright x)[\top] \mathrel{\overline{-\!\!*}} \top[\Diamond\copyright x]]$$

$$\mathsf{if}(M = N, A, B) \triangleq \mathsf{if} \wedge \text{И} x, y.\, \mathsf{Subst}(x = M, y = N) \mathrel{-\!\!*}$$

$$\top \left[\begin{array}{l} \copyright^{\mathsf{s}}_{=} x \wedge \copyright^{\mathsf{s}}_{=} y \wedge \neg\copyright^{\mathsf{br}} x \wedge \neg\copyright^{\mathsf{br}} y \\ \wedge \mathsf{Subst}(x = y) \mathrel{-\!\!*} \top[\Diamond A] \\ \wedge \text{И} s, s'.\, \mathsf{Subst}(x = s, y = s') \mathrel{-\!\!*} \top[\Diamond B] \end{array} \right]$$

Let $\sim$ be the smallest equivalence on pairs of terms $M = N$ such that:

$$\begin{array}{lrcl} \textsc{Symmetry} & M = N & \sim & N = M \\ \textsc{Shift} & \vec{f}(M) = N & \sim & M = \vec{g}(N) \end{array}$$

where we use $\vec{f}(M)$ and $\vec{g}(N)$ to respectively abbreviate $f_1(f_2(\ldots f_n(M)\ldots))$ and $g_1(g_2(\ldots g_m(N)\ldots))$ for some unary function symbols f_i, g_i such that $\mathcal{E} \vdash \vec{f}(\vec{g}(x)) = \vec{g}(\vec{f}(x)) = x$. Note that such symbols do not exist in the theory of finite trees. Let $\equiv'$ be the smallest congruence extending $\equiv$ with the following axiom:

$$\textsc{Test} \quad \begin{array}{l} \mathit{if\ test} \\ \quad \mathit{then}\ P\ \mathit{else}\ Q \end{array} \quad \equiv' \quad \begin{array}{l} \mathit{if\ test}' \\ \quad \mathit{then}\ P\ \mathit{else}\ Q \end{array}$$

when $\mathit{test} \sim \mathit{test}'$. Then the following holds:

Theorem 4.3 *Let $\mathcal{E}$ be the theory of finite trees. Then for every process P there is a formula $\mathbf{F}_P \in \mathcal{L}$ such that for all $A\pi$ processes Q, $Q \models \mathbf{F}_P$ if and only if $Q \equiv' P$.*

Let us drop now our hypothesis on the equational theory. Then the following result can be established:

Theorem 4.4 $\equiv' \ \subseteq \ =_L$.

Corollary 4.5 *Let $\mathcal{E}$ be the theory of finite trees. Then $\equiv'$ is the same as $=_L$.*

We do not know if $\equiv'$ is a complete axiomatization of $=_L$ in the general case, but we presume it is not a congruence, as it is the case for some other spatial logics [5].

5 Elimination of term quantification

This section is devoted to the proof of the following theorem:

Theorem 5.1 (Term quantification elimination) *For every closed formula $A \in \mathcal{L}$, there is a formula $[\![A]\!] \in \mathcal{L} \setminus \{\exists\}$ such that $A \Leftrightarrow [\![A]\!]$ is valid.*

$[\![A]\!]$ is defined by structural induction on the formula A. It leaves most of A's structure unchanged, while replacing every subformula $\exists t.\, A'$ with a formula of the form $\mathrm{H}x.\, [\![A']\!]_{\{t \to x\}}$. Thus, it leaves the term quantification to H, which has to pick a term for the new active substitution on x it reveals. Further occurrences of t in the formula will have to be replaced by the variable used for this new substitution. Of course, to follow the semantics of $\exists t.\, A$ we also have to make sure that this substitution does not use any hidden name of the process or any substitutions created by the traversal of previous term quantifications. This builds up an *environment frame* placed alongside the actual process that records witnesses of term quantifications, but for which some maintenance work is needed during the translation of a formula. For instance, we will need to copy this environment on each side of a $*$ operator, and on the left-hand side of a $-\!\!*$.

To keep track of this environment frame, the translation will be of the form $[\![A]\!]_v$ where v is a valuation $\{t_1 \to x_1, \ldots, t_n \to x_n\}$ that lets previously encountered term variables point to their corresponding variables in the environment. The translation thus starts with an empty valuation: $[\![A]\!] \triangleq [\![A]\!]_\emptyset$, and the valuation grows up each time a term quantification is encountered. We write e for the environment $\{x_1 \to M_1, \ldots,\, x_n \to M_n\}$ corresponding to the environment frame $[\![e]\!] \triangleq \{^{\widetilde{M}}/_{\tilde{x}}\}$. Moreover, we only consider environments e and translations $[\![A]\!]_v$ where $fv(A, \widetilde{M}) \cap \tilde{x} = \emptyset$. Finally, when the domain of e matches the codomain of v, we write $e \circ v$ for the valuation $\{t_1 \to M_1, \ldots, t_n \to M_n\}$.

We are now ready to give the inductive lemma we want to prove on $[\![A]\!]_v$:

Lemma 5.2 (Inductive hypothesis) *$P \models [\![A]\!]_v$ if and only if there exists Q and e such that $P \equiv Q \,|\, [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$, and $Q, e \circ v \models A$.*

To meet the requirements of this lemma and make sure that P actually is the composition of a process Q and an environment frame corresponding to e, we first define a formula Φ_v that will have to be verified at every step of the translation:

$$\Phi_{\{\tilde{t} \to \tilde{x}\}} \triangleq \bigwedge_{x \in \tilde{x}} (\mathsf{Subst}(x) * \neg \copyright x)$$

Lemma 5.3 *For all processes P and valuations v, $P \models \Phi_v$ if and only if there exists a process Q and an environment e such that $P \equiv Q \,|\, [\![e]\!]$ and $fv(Q) \cap dom([\![e]\!]) = \emptyset$.*

The translation of all the operators of the logic can be found in Appendix C. Given below are the proof sketches for the translations of $\exists t.\, A$ and $A_1 * A_2$. Let us first give an immediate property of the translation:

Lemma 5.4 $fv([\![A]\!]_v) = fv(A, v)$.

The actual translation of term quantification is as follows, where $x_{n+1} \notin fv(A, v)$:

$$[\![\exists t.\, A]\!]_v \triangleq \Phi_v \wedge \mathrm{H} x_{n+1}.\, [\![A]\!]_{v\{t \to x_{n+1}\}}$$

This is just a matter of creating a fresh substitution, as the inductive hypothesis on $[\![A]\!]_{v\{t \to x_{n+1}\}}$ suffices to enforce the validity of the new environment frame.

The translation of frame composition needs the valuation frame to be copied in order for it to be present alongside both subprocesses. It is performed as follows:

$$[\![A_1 * A_2]\!]_v \triangleq \Phi_v \wedge И\tilde{x}'.\, (\Phi_{v'} \wedge \bigwedge_{x \in \tilde{x}} \neg ©x \wedge \mathsf{Subst}(\tilde{x}')) \mathrel{\vec{-\!\!*}} \begin{pmatrix} *_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' = x_i) * \top \\ \wedge [\![A_1]\!]_v * [\![A_2]\!]_{v'} \end{pmatrix}$$

The idea is to add a new environment frame over fresh variables $\tilde{x}'$. The left-hand side of $\vec{-\!\!*}$ ensures that this is a valid environment which does not make use of the variables of the previous environment. This is to avoid the possibility of creating active substitutions of the form $\{{}^{x}/_{x'}\}$ which would not make sense once we separate them from the first environment. The right-hand side makes sure that both environments are the same and distributes them over the interpretations of sub-formulae A_1 and A_2.

6 Decidability of the logic without adjuncts

In this section we investigate the decidability of the model-checking problem for frames. We do not consider the adjuncts $-\!\!*$, $\triangleright$, as they allow to internalize validity, and validity of the logic with the hidden-name quantifier is undecidable [7]. Hence we consider the fragment $\mathcal{L}_{\text{nadj}} \triangleq \mathcal{L}_{\text{int}}^{\text{stat}} \setminus \{\mathrm{H}^{\mathcal{V}}, -\!\!*\} \cup \{©\}$, where $\mathrm{H}^{\mathcal{V}}$ is the hidden variable quantifier. We first prove that the model-checking problem for $\mathcal{L}_{\text{nadj}}$ on frames reduces to the satisfiability problem for a light equational logic $\mathcal{L}_{\text{eq}}$, and then investigate the decidability of the satisfiability problem for $\mathcal{L}_{\text{eq}}$.

6.1 From spatial to equational

The target logic $\mathcal{L}_{\text{eq}}$ we want to compile spatial formulae into is described below, and its semantics is given by the satisfaction relation between valuations and formulae of Figure 9. As we will see later on, satisfiability of this logic is decidable when we restrict the form of $U{=_\sigma}V$.

$$\begin{aligned} F ::= {} & U{=_\sigma}V \mid n \in \mathit{fn}(t) \mid x \in \mathit{fv}(t) \\ \mid {} & F \wedge F \mid \neg F \mid \exists t.\, F \mid \text{И} n.\, F \end{aligned}$$

Figure 9 Satisfaction relation

$$\begin{array}{lcl} v \vDash_{\mathrm{eq}} U{=_\sigma}V & \Leftrightarrow & \mathcal{E}, \sigma \vdash Uv = Vv \\ v \vDash_{\mathrm{eq}} n \in \mathit{fn}(t) & \Leftrightarrow & n \in \mathit{fn}(v(t)) \\ v \vDash_{\mathrm{eq}} x \in \mathit{fv}(t) & \Leftrightarrow & x \in \mathit{fv}(v(t)) \\ v \vDash_{\mathrm{eq}} F_1 \wedge F_2 & \Leftrightarrow & v \vDash_{\mathrm{eq}} F_1 \text{ and } v \vDash_{\mathrm{eq}} F_2 \\ v \vDash_{\mathrm{eq}} \neg F & \Leftrightarrow & v \nvDash_F F \\ v \vDash_{\mathrm{eq}} \exists t.\, F & \Leftrightarrow & \exists M.\, v\{t {\to} M\} \vDash_{\mathrm{eq}} F \\ v \vDash_{\mathrm{eq}} \text{И} n.\, F & \Leftrightarrow & \exists n' \notin \mathit{fn}(v, F).\, v \vDash_{\mathrm{eq}} F[n {\leftarrow} n'] \end{array}$$

We will now present our reduction, and establish the following theorem:

Theorem 6.1 *For every frame ϕ and formula A of $\mathcal{L}_{nadj}$ there is a formula $(\!|\phi \vDash A|\!)$ of $\mathcal{L}_{eq}$ such that:*

$$\phi \vDash A \textit{ if and only if } (\!|\phi \vDash A|\!) \textit{ is valid.}$$

Notations 6.2 We write $\tilde{t} = \tilde{t}'$ for $\bigwedge_{i=1}^{n} t_i = t'_i$ and $\mathit{codom}(\sigma)$ for the codomain of σ. Arities are implicitly supposed to match: for instance, in $\exists\tilde{t}.\, \tilde{t} = \mathit{codom}(\sigma)$, $\tilde{t}$'s size is implicitly chosen to match the size of $\mathit{codom}(\sigma)$. Finally, $(\emptyset \vdash U = V)$ may be written $U = V$.

Let $\phi \vDash A$ be a model-checking instance, with ϕ a frame and A a formula of $\mathcal{L}_{\mathrm{nadj}}$. We first have to rewrite ϕ to consist of a set of restricted names $\tilde{n}$ and a public frame σ. The translation then starts with $(\!|\phi \vDash A|\!) \triangleq (\!|\tilde{n}, \sigma, A|\!)$.

The translation is done by induction on the formula A. We maintain a list $\tilde{h}$ of hidden names that will be updated whenever we go through a hiding or a revelation operator. The inductive property we want to prove on the translation, and of which Theorem 6.1 is a direct consequence is as follows:

Lemma 6.3 (Inductive hypothesis) *For all v, σ, A and $\tilde{h}$:*

$$v \vDash (\!|\tilde{h}, \sigma, A|\!) \textit{ if and only if } (\nu\tilde{h}.\, \sigma), v \vDash A.$$

Let us now give the translation of all operators.

The translation of ©u follows its semantics and thus is quite straightforward. It is defined as $\perp$ when $u \in \tilde{h}$, $\top$ when $u \in \mathit{dom}(\sigma)$, and as shown below otherwise, fnv being fn when u is a name and fv when u is a variable.

$$(\!|\tilde{h}, \sigma, ©u|\!) \triangleq \forall\tilde{t}.\, \tilde{t} = \mathit{codom}(\sigma) \Rightarrow u \in \mathit{fnv}(\tilde{t})$$

When hiding a variable $x \in dom(\sigma)$, we need to apply the corresponding substitution to the rest of the frame and then throw the substitution on x away. Thus, if $x \notin dom(\sigma)$, then $(\!|\tilde{h}, \sigma, A \oslash x|\!) \triangleq \bot$, and otherwise we let:

$$(\!|\tilde{h}, \sigma \,|\{^{M}/_{x}\}, A \oslash x|\!) \triangleq (\!|\tilde{h}, \sigma[x \leftarrow M], A|\!)$$

Hiding a name consists merely of adding h to the set of hidden names, and term quantification is left as-is, since the semantics of $\exists t$ for AπL and $\mathcal{L}_{\mathrm{eq}}$ are the same. As we know all the hidden names of the frame, we can treat name revelation as a disjunction over those names, plus one fresh extra name to model the fact that we can reveal "fake" hidden names:

$$(\!|\tilde{h}, \sigma, A \oslash h|\!) \triangleq (\!|(\tilde{h}, h), \sigma, A|\!) \qquad\qquad (\!|\tilde{h}, \sigma, \exists t.\, A|\!) \triangleq \exists t.\, (\!|\tilde{h}, \sigma, A|\!)$$

$$(\!|\tilde{h}, \sigma, \mathrm{H}n.\, A|\!) \triangleq \text{И}h'.\, \bigvee_{h \in \tilde{h}, h'} (\!|(\tilde{h} \setminus h'), \sigma[h \leftarrow h'], A[n \leftarrow h']|\!)$$

To translate equality judgements, one has to take care of the hidden names of σ, as $\mathcal{L}_{\mathrm{eq}}$ only allows public frames in its grammar. To overcome this situation, we first replace the names $\tilde{h}$ of σ that should be restricted with fresh names $\tilde{h}'$ such that $\tilde{h}' \cap fn(\sigma, U, V) = \emptyset$. It is easy to check that these fresh names behave like hidden names for the equality test.

$$(\!|\tilde{h}, \sigma, U = V|\!) \triangleq \text{И}\tilde{h}'.\, U {=_{\sigma[\tilde{h} \leftarrow \tilde{h}']}} V$$

To translate $*$, we need to be able to state that the set of hidden names appearing in two subframes are disjoint one from another up to rewriting of terms using the equational theory. This is achieved by defining the operator $\widetilde{U} \bot^{\tilde{h}} \widetilde{V}$ below which states that two sets of names $\widetilde{U}$ and $\widetilde{V}$ may be rewritten not to share names in $\tilde{h}$:

$$\begin{aligned} \widetilde{U} \bot^{\tilde{h}} \widetilde{V} \triangleq\, & \exists \widetilde{U}', \widetilde{V}'.\, \widetilde{U}\widetilde{V} = \widetilde{U}'\widetilde{V}' \\ & \wedge \textstyle\bigwedge_{h \in \tilde{h}} (h \in fn(\widetilde{U}') \Rightarrow h \notin fn(\widetilde{V}')) \end{aligned}$$

The translation of frame composition then only needs to quantify over all valid frame compositions. Here, $\{\sigma_1, \sigma_2\}$ range over 2-partitions of σ considered as the set of its active substitutions.

$$(\!|\tilde{h}, \sigma, A_1 * A_2|\!) \triangleq \bigvee_{\sigma_1 \,|\, \sigma_2 \equiv \sigma} \begin{pmatrix} codom(\sigma_1) \bot^{\tilde{h}} codom(\sigma_2) \\ \wedge (\!|\tilde{h}, \sigma_1, A_1|\!) \wedge (\!|\tilde{h}, \sigma_2, A_2|\!) \end{pmatrix}$$

Finally, $(\!|\tilde{h}, \sigma, \neg A|\!) \triangleq \neg (\!|\tilde{h}, \sigma, A|\!)$, $(\!|\tilde{h}, \sigma, A_1 \wedge A_2|\!) \triangleq (\!|\tilde{h}, \sigma, A_1|\!) \wedge (\!|\tilde{h}, \sigma, A_2|\!)$, $(\!|\tilde{h}, \sigma, \emptyset|\!) \triangleq \top$ if $\sigma = \emptyset$ and $\bot$ otherwise.

6.2 Towards decidability: a first result

The translation of the previous subsection reduces the model checking problem for $\mathcal{L}_{\mathrm{nadj}}$ with respect to frames to the satisfiability of $\mathcal{L}_{\mathrm{eq}}$. We expect this reduction may help to obtain some decidability result.

As a first step in that direction, let us mention the decidability of the fragment of $\mathcal{L}_{\mathrm{nadj}}$ defined by restricting judgements $U{=_\sigma}V$ to the cases $t{=_\sigma}t'$ or $t{=_\sigma}M$. This fragment already allows to express the deducibility properties mentioned along the paper.

Since the previous translation only produces equality judgements of these forms, the model-checking problem of $\mathcal{L}_{\mathrm{nadj}}$ reduces to the satisfiability of $\mathcal{L}_{\mathrm{eq}}$ with the same restrictions on judgements $U{=_\sigma}V$. Note that we may also drop off the $\text{И}n$ quantifier in the satisfiability problem for closed formulae.

The decidability of the model-checking problem in restricted $\mathcal{L}_{\mathrm{nadj}}$ is thus a consequence of the decidability of the satisfiability problem for a first-order formula obtained with $\wedge$, $\vee$, $\neg$, $\forall t$, $\exists t$ and from the predicates $t_1{=_\sigma}t_2$ stating that the values of t_1 and t_2 are equal modulo the equational theory $\mathcal{E} \cup \sigma$, $t = M$ stating that the value of t is M, and $M \leq t$ stating that M is a subterm of the value of t, and where quantifiers vary over ground terms.

Decidability of this theory can be shown using standard tree automata techniques [6] using similar techniques as Dauchet and Tison [9], provided that the equational theory $\mathcal{E}$ can be represented by a term rewrite system R that is confluent, left-linear, and whose rules are all of the form $f(g(x_1, \ldots, x_n)) \to x_i$. An example of such an axiom system is $\{\pi_1\langle t_1, t_2\rangle = t_1, \pi_2\langle t_1, t_2\rangle = t_2\}$.

In particular the rewrite system is, as a consequence of the last condition, terminating. Confluence and termination assure that we can work with normal forms modulo R, left-linearity yields that the set of normal forms is regular, and the last condition ensures that the automaton recognizing the relation $t_1{=_\sigma}t_2$ has only finitely many states. The details are given in Appendix D.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, 2001.

[2] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *LICS 2005*, pages 331–340, June 2005.

[3] A. Blumensath and E. Grädel. Automatic structures. In *LICS 2000*, pages 51–62, Santa Barbara, CA, June 2000. IEEE Computer Society.

[4] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Journal of Information and Computation 186(2)*, 186(2), 2003.
[5] L. Caires and É. Lozes. Elimination of quantifiers and undecidability in spatial logics for concurrency. In Ph. Gardner and N. Yoshida, editors, *CONCUR'04)*, volume 3170 of *LNCS*, pages 240–257, London, UK, Aug. 2004. Springer.
[6] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.
[7] G. Conforti and G. Ghelli. Decidability of freshness, undecidability of revelation. *Proc. of Foundations of Software Science and Computation Structures*, 2004.
[8] H. D., L. E., and Sangiorgi.D. On the expressiveness of the ambient logic. *Logical Methods in Computer Science*, 2(2), 2006.
[9] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *LICS'90*, pages 242–248. IEEE Computer Society, 1990.
[10] A. Gordon and L. Cardelli. Anytime, anywhere: Modal logics for mobile ambients. In A. Press, editor, *POPL 2000*, pages 365–377, 2000.
[11] D. Hirschkoff. An extensional spatial logic for mobile processes. In *CONCUR'02*, volume 3252 of *LNCS*. Springer Verlag, 2002.
[12] D. Hirschkoff, É. Lozes, and D. Sangiorgi. Minimality results for spatial logics. In P. K. Pandya and J. Radhakrishnan, editors, *Proceedings of the 23rd Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, volume 2914 of *LNCS*, pages 252–264, Mumbai, India, Dec. 2003. Springer.
[13] H. Hüttel and M. D. Pedersen. A logical characterisation of static equivalence. *Electron. Notes Theor. Comput. Sci.*, 173:139–157, 2007.
[14] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In M. Sagiv, editor, *ESOP'05*, volume 3444 of *LNCS*, pages 186–200, Edinburgh, U.K., Apr. 2005. Springer.
[15] É. Lozes. Adjuncts elimination in the static ambient logic. In F. Corradini and U. Nestmann, editors, *Proceedings of the 10th International Workshop on Expressiveness in Concurrency (EXPRESS'03)*, volume 96 of *Electronic Notes in Theoretical Computer Science*, pages 51–72, Marseilles, France, June 2004. Elsevier Science Publishers.
[16] R. Mardare. Observing distributed computation. a dynamic-epistemic approach. In *CALCO*, volume 4624 of *LNCS*, pages 379–393. Springer, 2007.
[17] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1):1–40, 1992.
[18] D. Pym and C. Tofts. A Calculus and logic of resources and processes. *Formal Aspects of Computing*, 18(4):495–517, 2006.

[19] J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings Seventeenth Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, Los Alamitos, California, 2002. IEEE Computer Society.

A Technical lemmas about applied pi-calculus

We first collect some useful results on structural congruence in section A.1, and then give the proof of Proposition 2.11 in Section A.2.

Conventions. A process is said to be public if it does not make use of name restriction. In this appendix, except when stated otherwise, $P, P', Q, Q', \ldots$ will always stand for public plain processes, and $\phi, \psi, \ldots$ for public frames. Extended processes will be explicitly written $P^e, Q^p, \ldots$ We write $P\phi$ the plain process in which all the active substitutions of ϕ have been applied. The support of a name permutation σ is the set of names n such that $\sigma(n) \neq n$.

A.1 Technical lemmas about structural congruence

Lemma A.1 *If $P \mid \phi \equiv P' \mid \phi'$, then $\phi \equiv \phi'$.*

Lemma A.2 *If $P \mid \phi \equiv P' \mid \phi'$, then $P\phi \equiv P'\phi'$.*

Lemma A.3 *If $P \mid \phi \equiv P' \mid \phi'$, and $Q \mid \phi \equiv Q' \mid \phi'$, then $P \mid Q \mid \phi \equiv P' \mid Q' \mid \phi'$.*

Definition A.4 *An extended process P^e is said to be without fake names if $fn(P^e) = \overline{fn}(P^e)$.*

Lemma A.5 *For every extended process P, there is P' without fake names such that $P \equiv P'$.*

Lemma A.6 *For all P, ϕ, P', ϕ' without fake names, for all sets of names $\tilde{n}, \tilde{n}'$, the two conditions below are equivalent:*

- $\nu\tilde{n}.\,(\phi \mid P) \equiv \nu\tilde{n}'.\,(\phi' \mid P')$
- *there is a bijection $\sigma : \tilde{n} \rightarrow \tilde{n}'$ such that $\phi\sigma \mid P\sigma \equiv \phi' \mid P'$.*

A.2 On dually symmetric theories

In this section, we prove that $\between$ is a composition law when the equational theory is dually symmetric. Before defining this notion, let us explain why $*$ is not composition law and why dual symmetry is required for $\between$:

- $*$ may never be a composition law for processes, although it is always one for frames. Let $\phi = \{^a/_x\}$, $\phi_1 = \{^b/_x\}$, $\phi_2 = \{^c/_x\}$, and $P = a.b.c \mid b.c.a \mid c.a.b$. Let $P_1 \triangleq \nu a, b, c. (\phi_1 \mid P)$, $P_2 \triangleq \nu a, b, c. (\phi_2 \mid P)$, and $P' \triangleq \nu a, b, c. P$. Then $P_1 \equiv P_2$, but $P_1 * P' \not\equiv P_2 * P'$.

- One could replay this counter-example with $\between$ in lieu of $*$ just by taking π-calculus as the term algebra and π-calculus structural congruence as the equational theory.

Definition A.7 *We say that a theory is dually symmetric if*

1. *for all terms M, for all name permutations σ such that $\mathcal{E} \vdash M\sigma = M$, and for all n, $\mathcal{E} \vdash M[n \leftrightarrow \sigma(n)] = M$*

2. *for all ϕ, M, N, if $\phi \vdash M = N$ and $\phi \equiv \phi[n \leftrightarrow n']$, then $\phi \vdash M[n \leftarrow n'] = N[n \leftarrow n']$.*

In particular, the following facts hold:

- if $\phi\sigma \equiv \phi$, then $\phi[n \leftrightarrow \sigma(n)] \equiv \phi$;

- if σ_1, σ_2 are name permutations with disjoint supports and $\phi\sigma_1\sigma_2 \equiv \phi$, then $\phi\sigma_1 \equiv \phi\sigma_2 \equiv \phi$.

Lemma A.8 *Assume $\phi \vdash M = M'$, $\phi[n_1 \leftrightarrow n_2] \equiv \phi$, and $n_1 \in fn(M) \setminus fn(M')$, $n_2 \notin fn(M)$. Then $\phi \vdash M = M[n_1 \leftrightarrow n_2]$.*

Proof $\phi \vdash M = N$ and $\phi \equiv \phi[n \leftrightarrow n']$ implies $\phi \vdash M[n \leftarrow n'] = N[n \leftarrow n']$ since $\mathcal{E}$ is dually symmetric. □

Lemma A.9 *Assume $\phi \mid P \equiv \phi \mid P'$, $\phi[n_1 \leftrightarrow n_2] \equiv \phi$, and $n_1 \in fn(P) \setminus fn(P')$, $n_2 \notin fn(P)$. Then $P \mid \phi \equiv P[n_1 \leftrightarrow n_2]\phi$.*

Proof This is a direct consequence of Lemma A.8. □

Let $P_1, P'_1, P_2, P'_2, \phi, \phi'$ be some fixed processes and frames without fake names, and $\tilde{n}_1, \tilde{n}_2, \tilde{n}'_1, \tilde{n}'_2$ be sets of names such that:

$$\tilde{n}_1 \cap \tilde{n}_2 = \tilde{n}'_1 \cap \tilde{n}'_2 = \emptyset; \tag{1}$$

$$\mathit{fn}(P_i) \subseteq \tilde{n}_i,\ \mathit{fn}(P'_i) \subseteq \tilde{n}'_i,\ i = 1, 2. \tag{2}$$

Definition A.10 *We call* solver *a pair of name permutations* (σ, τ) *such that* $\sigma, \tau : (\tilde{n}_1 \cup \tilde{n}_2) \rightarrow (\tilde{n}'_1 \cup \tilde{n}'_2)$*, and:*

$$P\sigma \mid \phi\sigma \;\equiv\; P' \mid \phi' \tag{3}$$

$$Q\tau \mid \phi\tau \;\equiv\; Q' \mid \phi' \tag{4}$$

Lemma A.11 *If there is a solver* (σ, τ)*, then there is a solver* $(\sigma_1\sigma_2, \tau_1\tau_2)$ *such that* $\sigma_i, \tau_i : \tilde{n}_i \rightarrow \tilde{n}'_i$*.*

Proof We note $\mathit{conflict}(\sigma, \tau)$ the set of names $\sigma(\tilde{n}_1) \cap \tau(\tilde{n}_2)$. We need to show that we may take σ, τ such that $\mathit{conflict}(\sigma, \tau) = \emptyset$. We reason by induction. Assume $n' \in \mathit{conflict}(\sigma, \tau)$. We want to define a solver σ', τ' such that $\mathit{conflict}(\sigma', \tau') \subsetneq \mathit{conflict}(\sigma, \tau)$. Let n_1, n_2 be such that $\sigma(n_1) = \tau(n_2) = n'$. Pick $m_1 \in \tilde{n}_1, m_2 \in \tilde{n}_2$ such that the orbit under $\sigma\tau^{-1}$ of n_1 is $n_1, n_2, .., m_2, m_1, ..$, and let $m' = \sigma(m_2) = \tau(m_1)$. Note that $n' \neq m'$ since σ is injective. Note also that both $([n_1 \leftrightarrow m_2]; \sigma), \tau$ and $\sigma, ([n_2 \leftrightarrow m_1]; \tau)$ have a conflict set equal to $\mathit{conflict}(\sigma, \tau) \setminus \{n\}$. In the remaining, our aim is to show that one of these two possibilities gives a solver.

We first observe some symmetries between these names. By (3), (4) and Lemma A.1, $\phi\sigma \equiv \phi' \equiv \phi\tau$, so $\phi\sigma\tau^{-1} \equiv \phi$. By dual symmetry, we deduce that:

$$\begin{array}{ll} (a) & \phi \equiv \phi[n_1 \leftrightarrow m_2] \equiv \phi[n_2 \leftrightarrow m_1] \\ (b) & \qquad\quad \phi' \equiv \phi'[n' \leftrightarrow m'] \end{array}$$

We then reason by case analysis:

- Assume $n_1 \notin \mathit{fn}(P)$. We then set $\sigma' = [n_1 \leftrightarrow m_2]; \sigma$ and $\tau' = \tau$. Then $\mathit{conflict}(\sigma', \tau') = \mathit{conflict}(\sigma, \tau) \setminus \{n'\}$. Moreover, $P[n_1 \leftrightarrow m_2] = P$ since $n_1 \notin \mathit{fn}(P)$, that is $P\sigma' = P\sigma$. Then by (a), (3) holds for σ', so σ', τ' is a solver.

- Assume $n_2 \notin \mathit{fn}(Q)$. By a symmetric argument, we show that $\sigma, [m_1 \leftrightarrow n_2]\tau$ is a solver.

- Assume $n_1 \in \mathit{fn}(P_1)$ and $n_2 \in \mathit{fn}(P_2)$. By (1) and (2), either $n' \notin \mathit{fn}(P)$ or $n' \notin \mathit{fn}(Q)$. By symmetry, we may assume that $n' \notin \mathit{fn}(P')$. Then $P \mid \phi \equiv P'\sigma^{-1} \mid \phi$ by (3), $\phi \equiv \phi[n_1 \leftrightarrow m_2]$ by (a), $n_1 \in \mathit{fn}(P) \setminus \mathit{fn}(P'\sigma^{-1})$

by previous hypothesis, and $m_2 \notin fn(P)$ by (1, 2), so we may apply Lemma A.9, which gives $P \mid \phi \equiv P[n_1 \leftrightarrow m_2] \mid \phi$. We conclude the proof with the same argument used for the case $n_1 \notin fn(P)$.

□

Proof of Proposition 2.11 Assume P, Q, P', Q satisfy the hypothesis. Let $P_1, P_1', P_2, P_2', \phi, \phi'$ be some fixed processes and frames, and $\tilde{n}_1, \tilde{n}_2, \tilde{n}_1', \tilde{n}_2'$ be sets of names such that (1) and (2) hold, and $P = \nu\tilde{n}_1.\,(P_1 \mid \nu\tilde{n}_2.\,\phi)$, $Q = \nu\tilde{n}_2.\,(P_2 \mid \nu\tilde{n}_1.\,\phi)$, $P = \nu\tilde{n}_1'.\,(P_1' \mid \nu\tilde{n}_2'.\,\phi')$, $Q = \nu\tilde{n}_2'.\,(P_2' \mid \nu\tilde{n}_1'.\,\phi')$. We want to show that:

$$\nu\tilde{n}_1, \tilde{n}_2.\,(P_1 \mid P_2 \mid \phi) \equiv \nu\tilde{n}_1', \tilde{n}_2'.\,(P_1' \mid P_2' \mid \phi').$$

First, by Lemma A.5, we may assume without loss of generality that all processes are without fake names. Then, by Lemma A.6, we have some solver (σ, τ). By Lemma A.11, we may assume that $\sigma = \sigma_1\sigma_2$ and $\tau = \tau_1\tau_2$. (3) and (4) then boil down to:

$$\begin{array}{ll} (3') & P\sigma_1 \mid \phi\sigma_1\sigma_2 \equiv P' \mid \phi' \\ (4') & Q\tau_2 \mid \phi\tau_1\tau_2 \equiv Q' \mid \phi'. \end{array}$$

By Lemma A.1, we get $\phi\sigma_1\sigma_2 \equiv \phi' \equiv \phi\tau_1\tau_2$, so $\phi\sigma_1\tau_1^{-1}\sigma_2\tau_2^{-1} \equiv \phi$, and by dual symmetry $\phi\sigma_1\tau_1^{-1} \equiv \phi\sigma_2\tau_2^{-1} \equiv \phi$. Let $\upsilon = \sigma_1\tau_2$. Then $\phi\upsilon \equiv \phi\sigma \equiv \phi\tau$, and:

$$\begin{array}{ll} (3'') & P\upsilon \mid \phi\upsilon \equiv P' \mid \phi' \\ (4'') & Q\upsilon \mid \phi\upsilon \equiv Q' \mid \phi'. \end{array}$$

Now, by Lemma A.3, $P\upsilon \mid Q\upsilon \mid \phi\upsilon \equiv P' \mid Q' \mid \phi'$, which ends the proof. □

B Characterization of logical equivalence

In this section, we prove the characterization of logical equivalence on the dynamic intensional fragment.

B.1 Characterizing channel communications

Lemma B.1 *$P \vDash \mathsf{atom}(a, b)$ (i.e. ©$a \wedge$ ©$b \wedge (\mathsf{single} \blacktriangleright \Diamond\mathbf{0})$) iff one of the condition below is satisfied:*

1. *$P \equiv \bar{a}^{ch}\langle b\rangle$;*

2. *$P \equiv \bar{b}^{ch}\langle a\rangle$;*

3. *$P \equiv \bar{a}\langle M\rangle$ and $b \in \overline{fn}(M)$;*

4. *$P \equiv \bar{b}\langle M\rangle$ and $a \in \overline{fn}(M)$;*

Proof Straightforward. $\square$

Lemma B.2 *$P \models \mathsf{test}^{ch}(a,b)$ (i.e. $\mathsf{atom}(a,b) \wedge \neg \mathrm{H}x.\, \top[©x]$) iff P satisfies conditions 1 or 2 of Lemma B.1*

Proof P satisfies 3 or 4, iff it is congruent to $(\nu x)(\{^M/_x\} \mid \bar{c}\langle x\rangle)$ for some $c \in \{a,b\}$. $\square$

Lemma B.3 *$P \models \mathsf{in}^{ch}(a,b).A$ (i.e. $\mathsf{single} \wedge \neg\Diamond\top \wedge Иb.\, \mathsf{test}^{ch}(a,b) \blacktriangleright \Diamond A$) iff there is P' such that $P \equiv a^{ch}(b')P'$, and $P' \models A[b \leftarrow b']$ for some $b' \notin fn(A)$.*

Proof Assume $P \models \mathsf{in}^{ch}(a,b).A$. Then P is single and deadlock, but $P \mid \bar{c}^{ch}\langle d\rangle$ reduces. So P must be an input. Moreover, $\{c,d\} = \{a,b'\}$ with $b' \notin fnP$, so $a = c$ and $b' = d$. The other implication is straightforward. $\square$

Lemma B.4 *$P \models \mathsf{out}^{ch}(a,b).A$ (i.e.*

$$\mathsf{single} \wedge Иb'.\, (\mathsf{in}^{ch}(a,c).\mathsf{test}^{ch}(b',c) \triangleright \Diamond(\mathsf{test}^{ch}(b',b) \mathbin{¦} A))$$

) iff there is P' such that $P \equiv \bar{a}^{ch}\langle b\rangle.P'$, and $P' \models A$ for some $b' \notin fn(A)$.

Proof We have:

$$\begin{array}{ll}
 & P \models \mathsf{out}^{ch}(a,b).A \\
\Leftrightarrow & P \text{ is single} \\
 & \text{and } P \mid a^{ch}(c).\bar{d}^{ch}\langle e\rangle \rightarrow\ \models\ \mathsf{test}^{ch}(b',b) \mathbin{¦} A \\
 & \text{for some } \{d,e\} = \{c,b'\} \text{ and } c,b' \notin fnP \\
\Leftrightarrow & P \equiv \bar{a}^{ch}\langle f\rangle.P' \text{ for some } f, P' \\
 & \text{and } P' \mid (\bar{d}^{ch}\langle e\rangle)[c \leftarrow f]\ \models\ \mathsf{test}^{ch}(b',b) \mathbin{¦} A \\
 & \text{for some } \{d,e\} = \{c,b'\} \text{ and } c,b' \notin fnP \\
\Leftrightarrow & P \equiv \bar{a}^{ch}\langle b\rangle.P' \text{ and } P' \models A.
\end{array}$$

$\square$

B.2 Characterizing term communications in the theory of finite trees

In this section and the following, we assume that $\mathcal{E}$ is the equational theory of finite trees.

Lemma B.5 *$P \models ©^{s}_{=}x$ (i.e. $\top[©x] \wedge \neg\mathrm{H}z.\, (z \neq x \wedge ©x)[©z]$) iff $x \in \overline{fn}(P)$ and for all terms M appearing in P, either $x \notin \overline{fv}(M)$, or $M = x$.*

Proof We have:

$P \vDash ©^{s}_{=} x$

$\Leftrightarrow$ $x \in \overline{fn}(P)$ and there is no M, z, P' such that $P \equiv (\nu z)\{^M/_z\} \mid P'$, $M \neq x$, $x \in \overline{fv}(M)$, and $z \in \overline{fv}(P')$

$\Leftrightarrow$ $x \in \overline{fn}(P)$ and there is no M appearing in P such that $M \neq x$ and $x \in \overline{fv}(M)$.

□

Lemma B.6 *$P \vDash \mathsf{out}(a, x)$ (i.e. $\mathsf{single} \wedge ©a \wedge ©^{s}_{=} x \wedge (\mathsf{single} \blacktriangleright \Diamond \mathbf{0})$) iff $P \equiv \bar{a}\langle x \rangle$*

Proof We have:

$P \vDash \mathsf{out}(a, x)$

$\Leftrightarrow$ P is single, there is Q single such that $P \mid Q \rightarrow \mathbf{0}$, $a \in fn(P)$, $x \in \overline{fv}(P)$, and x is not a strict subterm

$\Leftrightarrow$ $P \equiv com.\mathbf{0}$ for some communication primitive, $a \in fn(P)$, $x \in \overline{fv}(P)$, and x is not a strict subterm

$\Leftrightarrow$ $P \equiv \bar{a}\langle x \rangle$

□

Lemma B.7 *$P \vDash \mathsf{in}(a, x).A$ (i.e. $\mathsf{single} \wedge \neg\Diamond\top \wedge И x.\, \mathsf{out}(a, x) \rhd \Diamond A$) iff there is P', x' such that $P \equiv a(x').P'$, $x' \notin fv(A)$, and $P' \vDash A[x \leftarrow x']$.*

Lemma B.8 *$P \vDash \mathsf{out}(a, M).A$,*
i.e. $\mathrm{H}x..(x = M) \left[\begin{array}{l} Иb.\, \mathsf{in}(a, y).\mathsf{out}(b, y) \rhd \\ \Diamond\big(\mathsf{out}(b, x) ¦ (A \wedge \neg © x)\big) \end{array} \right]$
iff there is P' such that $P \equiv \bar{a}\langle M \rangle.P'$ and $P' \vDash A$.

B.3 Characterizing conditionals

Lemma B.9 *$P \vDash \mathsf{if}$ (i.e. $\mathsf{single} \wedge \top \twoheadrightarrow \Diamond\top$) iff there is M, N, P_1, P_2 such that $P \equiv$ if $M = N$ then P_1 else P_2.*

Lemma B.10 *$P \vDash ©^{\mathsf{br}} x$ (i.e. $\mathsf{if} \wedge \mathrm{H}z, z'.\, \top[(\neg © x)[\top] \twoheadrightarrow \top[\Diamond © x]]$) iff there is M, N, P_1, P_2 such that $P \equiv$ if $M = N$ then P_1 else P_2 and $x \notin fv(M, N)$.*

Proof Let P, N, N', P_1, P_2 be such that $P \equiv$ *if* $N = N'$ *then* P_1 *else* P_2, and z, z' some fresh variables. We have:

$P \vDash ©^{\mathsf{br}} x$

$\Leftrightarrow$ (1) there is M, M', P', ϕ s.t. $P \equiv (\nu z, z')\big(\{^M/_z\} \mid \{^{M'}/_{z'}\} \mid P'\big)$, $x \notin \overline{fv}(\phi)$, and $P' \mid \phi \rightarrow P''$ with $x \in fv(P'')$

Assume (1). Then there is $N_1, N_1', P_1', P_2', \phi$ s.t.

- $x \notin \overline{fv}(M, M')$,
- $P_1 \equiv P_1'[z, z' \leftarrow M, M']$, $P_2 \equiv P_2'[z, z' \leftarrow M, M']$,
- $N = N_1[z, z' \leftarrow M, M']$, $N' = N_1'[z, z' \leftarrow M, M']$,
- and either $\phi \vdash N_1 = N_1'$ with $x \in \overline{fv}(P_1'\phi)$, or $x \in \overline{fv}(P_1'\phi)$ with $N_1\phi, N_1'\phi$ ground and non equal.

Then clearly $x \in \overline{fv}(P_1, P_2)$. Conversely, if $x \in \overline{fv}(P_1, P_2)$, then (1) holds for $M = M'$, $N = N'$, $P' \equiv \mathit{if}\ z = z'\ \mathit{then}\ P_1\ \mathit{else}\ P_2$, and either $\phi = \{^{z'}/_{z}\}$ if $x \in \overline{fv}(P_1)$, or $\phi = \{^{n}/_{z}\} \mid \{^{n'}/_{z'}\}$ if $x \in \overline{fv}(P_2)$. $\square$

Lemma B.11 *$P \vDash \mathsf{if}(M = N, A, B)$, i.e.*

$$\mathsf{if} \wedge \text{И} x, y.\, \mathsf{Subst}(x = M, y = N) \twoheadrightarrow$$

$$\top \left[\begin{array}{l} ©^{s}_{=} x \wedge ©^{s}_{=} y \wedge \neg ©^{\mathsf{br}} x \wedge \neg ©^{\mathsf{br}} y \\ \wedge \mathsf{Subst}(x = y) \twoheadrightarrow \top[\Diamond A] \\ \wedge \text{И} s, s'.\, \mathsf{Subst}(x = s, y = s') \twoheadrightarrow \top[\Diamond B] \end{array} \right]$$

iff there is M, N, P_1, P_2 such that $P \equiv \mathit{if}\ \mathit{test}\ \mathit{then}\ P_1\ \mathit{else}\ P_2$, with $\mathit{test} \in \{\ M = N\ ,\ N = M\ \}$, and $P_1 \vDash A$, $P_2 \vDash B$.

B.4 Soundness of $\equiv'$

We note $P \sim P'$ for the smallest congruence that satisfies the axiom Test only. So $\equiv'$ is $(\equiv \cup \sim)^*$.

Definition B.12 *A relation $\mathcal{R}$ is an intensional bisimulation if $\mathcal{R}$ is symmetric and for all $(P, Q) \in \mathcal{R}$:*

1. *$\phi(P) \equiv \phi(Q)$*
2. *if $P^p \equiv \mathbf{0}$, then $Q^p \equiv \mathbf{0}$;*
3. *if $u \in \overline{fnv}(P)$, then $u \in \overline{fnv}(Q)$;*
4. *if there is P' s.t. $P \equiv (\nu u)P'$, then there is Q' s.t. $Q \equiv (\nu u)Q'$ and $P'\mathcal{R}Q'$;*
5. *if $P \equiv P_1 \mid P_2$, then there are Q_1, Q_2 such that $Q \equiv Q_1 \mid Q_2$ and $P_i\mathcal{R}Q_i$;*
6. *if there is P' s.t. $P \rightarrow P'$, then there is Q' s.t. $Q \rightarrow Q'$ and $P'\mathcal{R}Q'$.*

Lemma B.13 *If $\mathcal{R}$ is both an intensional bisimulation and a congruence, then $\mathcal{R} \subseteq =_L$*

Proof Let A be some formula. We prove by induction on A that for all P, Q, v, if $P, v \vDash A$ and $Q, v \nvDash A$, then $(P, Q) \notin \mathcal{R}$:

- $A = A_1 \wedge A_2$: then either A_1 or A_2 discriminates P and Q
- $A = \neg A'$: then by induction $(Q, P) \notin \mathcal{R}$, and by symmetry $(P, Q) \notin \mathcal{R}$
- $A = \emptyset$ or $A = \ U = V$: then $(P, Q) \notin \mathcal{R}$ by condition 1.
- $A = \mathbf{0}$: then $(P, Q) \notin \mathcal{R}$ by condition 2.
- $A = ©u$: then $(P, Q) \notin \mathcal{R}$ by condition 3.
- $A = \exists t.A$: then there is v' such that $P, v' \vDash A$ and $Q, v' \nvDash A$, and the induction applies for A.
- $A = \Diamond A'$: then $(P, Q) \notin \mathcal{R}$ by condition 6 and induction.
- $A = \mathrm{H}u.\, A'$: then $(P, Q) \notin \mathcal{R}$ by condition 4 and induction.
- $A = A' \oslash u$: then by induction $(\nu u.P, \nu u.Q) \notin \mathcal{R}$, and since $\mathcal{R}$ is supposed to be a congruence $(P, Q) \notin \mathcal{R}$.
- $A = A_1 \,\dagger\, A_2$: let $P_1, P_2, \phi, \tilde{n}_1, \tilde{n}_2$ be such that $P \equiv (\nu\tilde{n}_1, \tilde{n}2\big(P_1 \,|\, \phi \,|\, P_2))$, and $(\nu\tilde{n}_1, \tilde{n}_2)(P_i \,|\, \phi) \vDash A_i$. Assume by absurd $(P, Q) \in \mathcal{R}$. Then by conditions 4 and 5, there are Q', Q_1, Q_2 such that $\phi\mathcal{R}Q'$, $P_1\mathcal{R}Q_1$, and $P_2\mathcal{R}Q_2$. By condition 1 and 2, Q' is a frame, and by condition 3 Q splits as:
$$(\nu\tilde{n}_1, \tilde{n}_2)(Q' \,|\, Q_1) \,\dagger\, (\nu\tilde{n}_1, \tilde{n}_2)(Q' \,|\, Q_2).$$
Then by congruence of $\mathcal{R}$, $(\nu\tilde{n}_1, \tilde{n}_2)(P_i \,|\, \phi)\ \mathcal{R}\ (\nu\tilde{n}_1, \tilde{n}_2)(Q_i \,|\, Q')$, which contradicts the induction.
- $A = A_1 * A_2$: similar.
- $A = A_1 \triangleright A_2$: let R be some extended process such that $Q \,\dagger\, R \downarrow$. Then there are $Q', R', \phi, \tilde{n}_1, \tilde{n}_2$ s.t. $Q \equiv (\nu\tilde{n}_1)(Q' \,|\, \nu\tilde{n}_2.\phi)$ and $R \equiv (\nu\tilde{n}_2)(R' \,|\, \nu\tilde{n}_1.\phi)$. Assume by absurd (P, Q). Then by conditions 4 and 5 there are P', PP such that $P \equiv (\nu\tilde{n}_1, \tilde{n}_2)(PP \,|\, P')$, $P'\mathcal{R}Q'$ and $PP\mathcal{R}\phi$. By condition 1 and 2 $PP \equiv \phi$, and then by condition 3 $P \,\dagger\, R$ is defined. Moreover, by congruence of $\mathcal{R}$, we get $P \,\dagger\, R\ \mathcal{R}\ Q \,\dagger\, R$, which contradicts the induction hypothesis.
- $A = A_1 \twoheadrightarrow\!\!* A_2$: similar.

□

Lemma B.14 $\equiv$ *is an intensional bisimulation.*

Proof Straightforward. $\square$

Lemma B.15 *If $P \sim Q$, then $P\sigma \sim Q\sigma$ for all public frames σ (resp for all name substitutions).*

Lemma B.16 *If $P[x \leftarrow M] \sim Q$, then there is Q' such that $Q \equiv Q'[x \leftarrow M]$ and $P \sim Q'$.*

Proof We reason by case analysis on the rule of test replacement that is used in $P \sim Q$:

- SYMMETRY: straightforward.
- SHIFT: then there $P[x \leftarrow M]$ contains a test $\vec{f}(N_1) = N_2$ that is replaced by $N_1 = \vec{g}(N_2)$ in Q. If M is a subterm $\vec{f_1}(N_1)$ of $\vec{f}(N_1)$, one may possibly have the test $\vec{f_2}(x) = N_2$ in P, with $\vec{f} = \vec{f_2}\vec{f_1}$ (for simplicity, we omit to consider that M may also be a subterm of N_2). Then Q is congruent to the process Q_0 identical to Q except for the test $N_1 = \vec{g}(N_2)$ of Q that is replaced by $\vec{g}(\vec{f_2}(\vec{f_1}(N_1))) = g(N_2)$. Then $Q_0 = Q'[x \leftarrow M]$ where Q' contains the test $\vec{g}(\vec{f_2}(x)) = \vec{g}(N_2)$, so $P \sim Q'$.

$\square$

Lemma B.17 *$P \equiv\sim Q$ if and only if $P \sim\equiv Q$.*

Proof Test replacement commutes with all congruence rewriting. It is straightforward for most of the cases, but the cases of SUBST and REWRITE. For these two cases, Lemmas B.15 and B.15 are used. $\square$

Lemma B.18 $\sim$ *is an intensional bisimulation up to $\equiv$.*

Proof It is straightforward that $\sim$ satisfies conditions 1,2 and 3. It also is straightforward that it satisfies 4 for when u is a name. Thanks to Lemma B.15, it also satisfies 6. Let us detail the last cases: condition 4 for variables, and condition 5.

- Assume $P \sim Q$ and $P \equiv (\nu x)P'$. Then $P' \equiv \{^M/_x\} \mid P''$ with $P''[x \leftarrow M] \equiv\sim Q$. By Lemma B.16, there is Q'' such that $Q \equiv (\nu x)(\{^M/_x\} \mid Q''$ and $P'' \sim Q''$. Then $P' \sim Q'$ for $Q' = \{^M/_x\} \mid Q''$, which shows condition 4.
- Assume $P \sim Q$ and $P \equiv P_1 \mid P_2$. Then by Lemma B.17, there is Q' such that $Q \equiv Q'$ and $Q' \sim P_1 \mid P_2$. By definition of $\sim$, there is then Q_1, Q_2 such that $Q' = Q_1 \mid Q_2$ and $P_i \sim Q_i$, which shows condition 5.

$\square$

C Elimination of term quantifiers

Let us first enunciate two lemmas:

Lemma C.1 *For every extended processes Q, Q' such that $fv(Q,Q') \cap \{\tilde{x}\} = \emptyset$, if $Q \,|\, \{^{M_1 \dots M_n}/_{x_1 \dots x_n}\} \equiv Q' \,|\, \{^{M'_1 \dots M'_n}/_{x_1 \dots x_n}\}$ for some terms $\widetilde{M}, \widetilde{M'}$, then $\widetilde{M} = \widetilde{M'}$ and $Q \equiv Q'$.*

Proof $Q \equiv Q'$ is immediate since the processes do not mention variables in $\tilde{x}$.

According to structural congruence rules for public frames, $\widetilde{M}$ and $\widetilde{M'}$ have to be equal term by term. □

Lemma C.2 *If $P \equiv \sigma \,|\, P'$ for some public frame σ and extended process P' such that $fv(P') \cap dom(\sigma) = \emptyset$, and there exists Q, R such that $R \leftrightarrow P \,¦\, Q$, then $Q \equiv \sigma \,|\, Q'$ and $R \equiv \sigma \,|\, R'$ for some Q' and R', and $R' \leftrightarrow P' \,¦\, Q'$.*

Proof As $R \leftrightarrow P \,¦\, Q$ and $P \equiv \sigma \,|\, P'$, there exists $\tilde{n}_1, \tilde{n}_2, \phi, P_1$ and P_2 such that $\tilde{n}_1 \cap fn(P_2) = \tilde{n}_2 \cap fn(P_1) = \emptyset$, $\tilde{n}_1\tilde{n}_2 \cap fn(\sigma) = \emptyset$ and:

$$\begin{array}{rcl} P & \equiv & \nu\tilde{n}_1.\, (\nu\tilde{n}_2.\, (\sigma \,|\, \phi) \,|\, P_1) \\ Q & \equiv & \nu\tilde{n}_2.\, (\nu\tilde{n}_1.\, (\sigma \,|\, \phi) \,|\, P_2) \\ R & \equiv & \nu\tilde{n}_1\tilde{n}_2.\, (\sigma \,|\, \phi \,|\, P_1 \,|\, P_2) \end{array}$$

By Lemma 2.3, we deduce that there are Q' and R' such that $Q \equiv \sigma \,|\, Q'$ and $R \equiv \sigma \,|\, R'$. We may chose Q' and R' such that $fv(Q', R') \cap dom(\sigma)\emptyset$, and thanks to the fact that σ is public and that $\tilde{n}_1\tilde{n}_2 \cap dom(\sigma) = \emptyset$, we can assume $fv(P_1, P_2) \cap dom(\sigma) = \emptyset$ too and deduce that:

$$\begin{array}{rcl} P' & \equiv & \nu\tilde{n}_1.\, (\nu\tilde{n}_2.\, \phi \,|\, P_1) \\ Q' & \equiv & \nu\tilde{n}_2.\, (\nu\tilde{n}_1.\, \phi \,|\, P_2) \\ R' & \equiv & \nu\tilde{n}_1\tilde{n}_2.\, (\phi \,|\, P_1 \,|\, P_2) \end{array}$$

This shows that $R' \leftrightarrow P' \,¦\, Q'$. □

Lemma C.3 *For every public frame ϕ and processes P, Q, R, $R \leftrightarrow P \,¦\, Q$ if and only if $(\phi \,|\, R) \leftrightarrow (\phi \,|\, P) ¦ (\phi \,|\, Q)$.*

Proof This is straightforward. □

Definition C.4 *We write $e \equiv e'$ when $e = \{\tilde{x} \to \widetilde{M}\}$, $e' = \{\tilde{x} \to \tilde{M}'\}$ and for $i \in \{1 \dots n\}$, $M_i = M_i'$.*

We now prove Lemma 5.2 by induction on the formula A. We give the translation $[\![A]\!]_v$ along with the proof, and replace operators $\triangleright$ and $-\!\!*$ with their respective dual counterparts, as it is equivalent, albeit easier to do so.

- $[\![M = N]\!]_v \triangleq \Phi_v \wedge Mv = Nv$:

$$\begin{array}{rl} & P \models [\![M = N]\!]_v \\ \Leftrightarrow & P \equiv Q \,|\, [\![e]\!],\ fv(Q) \cap dom([\![e]\!]) = \emptyset \\ & \text{ and } P \models Mv = Nv \\ \Leftrightarrow & P \equiv Q \,|\, [\![e]\!],\ fv(Q) \cap dom([\![e]\!]) = \emptyset \\ & \text{ and } Q \models Mve = Nve \\ \Leftrightarrow & P \equiv Q \,|\, [\![e]\!],\ fv(Q) \cap dom([\![e]\!]) = \emptyset \\ & \text{ and } Q, (e \circ v) \models M = N \end{array}$$

  Indeed, $fv(Q, M, N) \cap dom([\![e]\!]) = \emptyset$.

- $[\![\mathbf{0}]\!]_v \triangleq \Phi_v \wedge \mathbf{0}$, $[\![\emptyset]\!]_v \triangleq \Phi_v \wedge \mathsf{Subst}(\tilde{x})$: Trivial.

- $[\![A_1 \wedge A_2]\!]_v \triangleq \Phi_v \wedge [\![A_1]\!]_v \wedge [\![A_2]\!]_v$: If $P \models [\![A]\!]_v$, then by induction there exist $Q_1 \,|\, [\![e_1]\!] \equiv P$ and $Q_2 \,|\, [\![e_2]\!] \equiv P$ such that $fv(Q_i) \cap dom([\![e_i]\!]) = \emptyset$ and $Q_i, (e_i \circ v) \models A_i$ ($i \in \{1, 2\}$). By Lemmas C.1 and 3.2, A_2 also holds for Q_1 and the desired result follows (the converse is immediate).

- $[\![\neg A]\!]_v \triangleq \Phi_v \wedge \neg [\![A]\!]_v$: If $P \models [\![\neg A]\!]_v$ it then holds not that $P \models [\![A]\!]_v$ so, by induction hypothesis, for all Q, e such that $P \equiv Q \,|\, [\![e]\!]$, $Q, (e \circ v) \not\models A$. As $P \models \Phi_v$, there exists such Q and e and $Q, (e \circ v) \models \neg A$, hence the result. The converse follows immediately from Lemma C.1.

- $[\![\Diamond A]\!]_v \triangleq \Phi_v \wedge \Diamond [\![A]\!]_v$: Straightforward.

- $[\![A_1 \,¦\, A_2]\!]_v \triangleq \Phi_v \wedge ([\![A_1]\!]_v \,¦\, [\![A_2]\!]_v)$: Straightforward.

- $[\![A \blacktriangleright B]\!] \triangleq \Phi_v \wedge [\![A]\!]_v \blacktriangleright [\![B]\!]_v$: If $P \models [\![A \blacktriangleright B]\!]_v$ then $P \equiv P' \,|\, [\![e]\!]$ for some P' and e such that $fv(P') \cap dom(e) = \emptyset$, and there exists Q, R such that $R \leftrightarrow P' ¦ Q$, $Q \models [\![A]\!]_v$ and $R \models [\![B]\!]_v$. By Lemma C.2 there are processes Q', R' such that $Q \equiv Q' \,|\, [\![e]\!]$, $R' \leftrightarrow P' ¦ Q'$ and $R \equiv R' \,|\, [\![e]\!]$. By induction hypothesis, $Q', (e \circ v) \models A$ and $R', (e \circ v) \models B$, so $P', (e \circ v) \models A \blacktriangleright B$.

  The converse is similar; it uses Lemma C.3 as a converse of Lemma C.2.

- $[\![\mathrm{H}x.\, A]\!]_v \triangleq \Phi_v \wedge \mathrm{H}x'.\, [\![A[x{\leftarrow}x']]\!]_v \quad (x' \notin fv(v))$: If $P \vDash [\![\mathrm{H}x.\, A]\!]_v$ then $P \equiv Q \,|\, [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $P \vDash \mathrm{H}x'.\, [\![A[x{\leftarrow}x']]\!]_v$. By definition of H, there exists $y \notin fv(Q, e, [\![A[x{\leftarrow}x']]\!]_v)$ and P' such that $P \equiv \nu y.\, P'$ and $P' \vDash [\![A[x{\leftarrow}x']]\!]_v[x'{\leftarrow}y]$, so $P' \vDash [\![A[x{\leftarrow}y]]\!]_v$. By induction hypothesis, there exists Q', e' such that $P' \equiv Q' \,|\, [\![e']\!]$ and $Q', (e' \circ v) \vDash A[x{\leftarrow}y]$. As $P \equiv \nu y.\, P'$, $e \equiv e'$ and $Q \equiv \nu y.\, Q'$, so $Q', (e \circ v) \vDash \mathrm{H}y.\, A[x{\leftarrow}y]$, and by α-conversion, $Q, (e \circ v) \vDash \mathrm{H}x.\, A$.

  Reciprocally, with the same notations:

  $$\begin{aligned} & Q, (e \circ v) \vDash \mathrm{H}x.\, A \Rightarrow Q', (e \circ v) \vDash A[x{\leftarrow}y] \\ \Rightarrow\; & P' \vDash [\![A[x{\leftarrow}y]]\!]_v \quad \Rightarrow P' \vDash [\![A[x{\leftarrow}x']]\!]_v[x'{\leftarrow}y] \\ \Rightarrow\; & P \vDash \mathrm{H}x'.\, [\![A[x{\leftarrow}x']]\!]_v \end{aligned}$$

- $[\![\mathrm{H}n.\, A]\!]_v \triangleq \Phi_v \wedge \mathrm{H}n.\, [\![A]\!]_v$: Same as above, albeit easier.

- $[\![©u]\!]_v \triangleq \Phi_v \wedge \mathsf{Subst}(\tilde{x}) * ((\bigwedge_{x \in \{\tilde{x}\}} \neg ©x) \wedge ©u)$: Let us prove the inductive case for $[\![©y]\!]_v$ for some variable y, the case where u is a name being similar. $P \vDash [\![©y]\!]_v$ if and only if $P \equiv Q \,|\, [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $P \vDash \mathsf{Subst}(\tilde{x}) * ((\bigwedge_{x \in \tilde{x}} \neg ©x) \wedge ©y)$. This is true if and only if $Q \vDash ©y$, the condition $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and the formula $\bigwedge_{x \in \tilde{x}} \neg ©x$ ensuring that active substitutions of the environment frame have not been applied, nor unapplied.

- $[\![\exists t.\, A]\!]_v \triangleq \Phi_v \wedge \mathrm{H}x_{n+1}.\, [\![A]\!]_{v\{t \to x_{n+1}\}}$: If $P \vDash [\![\exists t.\, A]\!]_v$ then one the one hand, by Lemma 5.3, there exist Q and e such that $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $P \equiv Q \,|\, [\![e]\!]$, and on the other hand, by definition of H and Lemma 5.4, for some $x \notin fv(P, A, v)$ there is P' such that $P \equiv \nu x.\, P'$ and $P' \vDash [\![A]\!]_{v\{t \to x\}}$. By induction hypothesis, $P' \equiv Q \,|\, [\![e]\!] \,|\, \{{}^{M}/_{x}\}$, so $Q, (e\{x{\to}M\} \circ v\{t{\to}x\} \vDash A$, so $Q, (e \circ v) \vDash \exists t.\, A$.

  Conversely, if $P \equiv Q \,|\, [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $Q, (e \circ v) \vDash \exists t.\, A$, then there exists M such that $Q, (e \circ v)\{t{\to}M\} \vDash A$. As variables in $dom([\![e]\!])$ are all free for Q,A, we can chose M such that $fv(M) \cap dom([\![e]\!]) = \emptyset$, and so $e\{x{\to}M\}$ is a valid environment and the induction hypothesis applies, so $Q \,|\, [\![e\{x{\to}M\}]\!] \vDash [\![A]\!]_{v \cup \{t \to x\}}$. This, together with Lemma 5.4, shows that $P \vDash \mathrm{H}x_{n+1}.\, [\![A]\!]_{v \cup \{t \to x_{n+1}\}}$, and by Lemma 5.3 $P \vDash [\![\exists t.\, A]\!]_v$.

In the following, we will write w for the valuation $\{t_1{\to}y_1, \ldots, t_n{\to}y_n\}$ and f for the environment $\{y_1{\to}N_1, \ldots, y_n{\to}N_n\}$.

- $[\![A_1 * A_2]\!]_v \triangleq \Phi_v \wedge \text{И}\tilde{x}'.$

$$(\bigwedge_{x\in\tilde{x}} \neg \copyright x \wedge \mathsf{Subst}(\tilde{x}') \wedge \Phi_{v'}) \\ \mathrel{-\!\!\circledast} \begin{pmatrix} \ast_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' = x_i) \ast \top \\ \wedge [\![A_1]\!]_v \ast [\![A_2]\!]_v' \end{pmatrix}$$

It is easy to check that $P \vDash [\![A_1 \ast A_2]\!]_v$ if and only if there are Q, $\widetilde{M}$, $\tilde{y}$ fresh and $\widetilde{N}$ such that $fv(M, N, Q) \cap \tilde{x}\tilde{y} = \emptyset$, for $i \in \{1, \ldots, n\}$, $\{^{M_i}/_{x_i}\} \,|\, \{^{N_i}/_{y_i}\} \vDash x_i = y_i$, and $Q \,|\, [\![e]\!] \,|\, [\![f]\!] \vDash [\![A_1]\!]_v \ast [\![A_2]\!]_w$. The former is equivalent to $M_i = N_i$ for all i, and we conclude from the latter and the induction hypothesis that $Q \equiv Q_1 \ast Q_2$ and $Q_i, (e \circ v) \vDash A_i$ (as $e \circ v = f \circ w$). Thus, $Q, (e \circ v) \vDash A_1 \ast A_2$.

Conversely, if $Q, (e \circ v) \vDash A_1 \ast A_2$, then $Q \equiv Q_1 \ast Q_2$ for some $Q_i, (e \circ v) \vDash A_i$. By induction hypothesis, $Q_1 \,|\, [\![e]\!] \vDash [\![A_1]\!]_v$ and $Q_2 \,|\, [\![f]\!] \vDash [\![A_2]\!]_w$. As $Q \,|\, [\![e]\!] \,|\, [\![f]\!] \equiv (Q_1 \,|\, [\![e]\!]) \ast (Q_2 \,|\, [\![f]\!])$, the desired result holds.

- $[\![A \mathrel{-\!\!\circledast} B]\!]_v \triangleq \Phi_v \wedge И\tilde{x}'.\, (\bigwedge_{x\in\tilde{x}} \neg \copyright x \wedge [\![A]\!]_{v'}) \\ \mathrel{-\!\!\circledast} \begin{pmatrix} \top \ast (\ast_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' \neq x_i)) \\ \wedge([\![B]\!]_v \wedge \bigwedge_{x'\in\tilde{x}'} \neg \copyright x') \ast \mathsf{Subst}(\tilde{x}') \end{pmatrix}$

  As in the cases of $\ast$ and $\blacktriangleright$, it should be immediate to check that $P \vDash [\![A \mathrel{-\!\!\circledast} B]\!]_v$ if and only if there are P',Q',R',$\widetilde{M}$,$\tilde{y}$ and $\widetilde{N}$ such that $R' \leftrightarrow P' \ast Q'$, $fv(M, N, P', Q', R') \cap \tilde{x}\tilde{y} = \emptyset$, $\widetilde{N} = \widetilde{M}$, $Q' \,|\, [\![f]\!] \vDash [\![A]\!]_w$ and $R' \,|\, [\![e]\!] \,|\, [\![f]\!] \vDash ([\![B]\!]_v \wedge \bigwedge_{y\in\tilde{y}} \neg \copyright y) \ast \mathsf{Subst}(\tilde{y})$, so $R' \,|\, [\![e]\!] \vDash [\![B]\!]_v$. With two applications of the induction hypothesis, we get $Q', (e \circ v) \vDash A$ and $R', (e \circ v) \vDash B$, yielding the result. Like for frame composition, the converse is straightforward.

D First-Order Theory of the Termalgebra

D.1 Eliminating the fresh quantifier

We detail here why it is sufficient for us to decide validity for formulae of $\mathcal{L}_{\text{eq}}$ without the fresh quantifier. For A, B some formulae in $\mathcal{L}_{\text{eq}}$, let $A \dashv\vdash B$ denotes that A, B are satisfied the same valuations v. Then we have:

$$\begin{array}{rcll} Иn.\, A & \dashv\vdash & Иm.\, A[n{\leftarrow}m] & (m \notin fn(A)) \\ (Иn.\, A) \;\wedge\; B & \dashv\vdash & Иn.\, (A \wedge B) & (n \notin fn(B)) \\ \neg Иn.\, A & \dashv\vdash & Иn.\, \neg A & \\ \exists t Иn.\, A & \dashv\vdash & Иn.\, \exists t. n \notin fn(t) \wedge A & \end{array}$$

We may then rewrite any formula into a formula that respects the Barendregt convention, and then pull all fresh quantifiers in prenex position. So any formula is equivalent to a formula of the form $\text{И}\tilde{n}.\, A$ where A is without $\text{И}n.\,.$. This is sufficient to eliminate all freshness conditions: if A is closed, e.g every t variable in A is bound by some $\exists$, then $\text{И}\tilde{n}.\, A$ is satisfiable if and only if A is satisfiable.

D.2 Tree-automatic structures

We recall the definition of tree automata [6].

Definition D.1 *Given a finite signature Σ, a (bottom-up)* tree automaton *A is given by (Q, F, Δ) where*

- *Q is a finite set of* states.
- *$F \subseteq Q$ is called the set of* accepting states.
- *Δ is a set of rewrite rules of the form $f(q_1, \ldots, q_n) \to q$ with $f \in \Sigma_n$, $q_1, \ldots, q_n, q \in Q$.*

The automaton A accepts *a tree t iff $t \to^* q \in F$ by the transition rules Δ. The* language *L_A is the set of all trees accepted by A.*

Example

The following tree automaton accepts the set of trees representing lists of integers, where integers are build with 0 and s, and lists with *nil* and *cons*:

- $\Sigma_0 = \{0, nil\}$, $\Sigma_1 = \{s\}$, $\Sigma_2 = \{cons\}$
- $Q = \{q_n, q_l\}$
- $F = \{q_l\}$
- $0 \to q_n$, $s(q_n) \to q_n$, $nil \to q_l$, $cons(q_n, q_l) \to q_l$

Tree automata enjoy (almost) all the nice properties of word automata, in particular closure under Boolean operation, decidability of the emptiness problem, determinisation, minimization [6].

The *convolution* operation defined below allows to code n-tuples of trees as trees over a signature of n-tuples. For this definition it is convenient to see a tree M as a pair (D_M, L_M) consisting of a *tree domain* D_M, that is a non-empty subset of $\mathbf{N}^*$ that is closed under prefix and left brother, and a *labeling function* $L_M \colon D_M \to \Sigma$ that is consistent with the arities of the symbols in Σ.

Definition D.2 *Let Σ be a signature with $\Box \notin \Sigma$. We define the signature $\Sigma_n^{\Box}$, for $n \geq 1$, as*

$$\Sigma_n^{\Box} = \{(f_1, \ldots, f_n) \mid f_i \in \Sigma \cup \{\Box\}, f_i \neq \Box \text{ for some } i\}$$

The arity of $[f_1 \ldots, f_n]$ in $\Sigma_n^{\Box}$ is the maximum of the arities of those f_i that are in Σ.

The convolution $M_1 \otimes \ldots \otimes M_n$ is the tree M defined by

- $D_M = D_{M_1} \cup \ldots \cup D_{M_n}$
- $L_M(\pi) = [f_1, \ldots, f_n]$ *where* $f_i = L_{M_i}(\pi)$ *if* $\pi \in D_{M_i}$, *and* $f_i = \Box$ *otherwise.*

Projection is defined by $\pi_i(M_1 \otimes \ldots \otimes M_n) = M_i$.

Example

Let $\Sigma = \{h, f, a\}$, where a is a constant, f unary, and h binary. Then we have that

$$f(a) \otimes h(a, f(a)) = [f, h]([a, a], [\Box, f]([\Box, a]))$$

Now, one can define *tree-automatic representations* and *tree-automatic structures* analogously to the definition given in [3] for automata over finite words. This definition applies only to so-called *relational* structures, that is structures that have only predicates in their logical language and no constants or function symbols. This is not a restriction as constants or functions can always be expressed by predicates.

Definition D.3 *Let A be a structure over a relational signature with relation symbols $R_1, \ldots, R_n$. A* tree-automatic representation *of A is given by*

1. *a finite signature Σ,*
2. *a regular tree language $L_\delta \subseteq T(\Sigma)$,*
3. *an onto function $\nu \colon L_\delta \to A$,*
4. *a regular tree language $L_R \subseteq T(\Sigma_n^{\Box})$ for each relation symbol R of the signature of A, such that for all $x_1, \ldots, x_n \in L_\delta$:*

$$x_1 \otimes \ldots \otimes x_n \in L_R \text{ iff } (\nu(x_1), \ldots, \nu(x_n)) \in R^A$$

A structure is tree-automatic *if it has a tree-automatic representation.*

Theorem D.4 *The first-order theory of any tree-automatic structure is decidable.*

D.3 Decidability of the FO-theory of $\mathcal{A}_{\sigma,G,\mathcal{E}}$

The reader shouldn't be confused by the fact that the notation used in this section differs slightly from the one that is common in equational logic. We abide by the notational conventions of applied π-calculus used throughout this paper, that is t_i are first-order variables, M_i are terms, and x_i are considered constants here (they are variables of the applied π-calculus that are bound by frames, which do not exist in the first order theory).

Let σ be a set of ground equations of the form

$$\sigma = \{x_1 = M_1, \ldots, x_n = M_n\}$$

with the following properties:

- $x_i \neq x_j$ if $i \neq j$
- $x_i \notin \texttt{subterms}(M_j)$ for all $1 \leq i \leq n$ and $1 \leq j \leq n$

Let $G \subseteq T(\Sigma)$ be a finite set of ground terms, and let $\mathcal{E}$ be an equational theory that is represented by some term rewrite system R (that is: $\mathcal{E} \vdash M_1 = M_2$ iff $M_1 \downarrow_R = M_2 \downarrow_R$) with the properties

1. confluent,
2. left-linear,
3. all rules are of the form $f(g(x_1, \ldots, x_n)) \rightarrow x_i$

We define $D_\sigma = \{x_1, \ldots, x_n\}$. We assume w.l.o.g. that no symbol of D_σ appears in R, and that each M_i is in normal form with respect to R.

Definition D.5 *the language of $\mathcal{A}_{\sigma,G,\mathcal{E}}$ consists of*

- *a unary predicate $t = M$ for each $M \in G$;*
- *a unary predicate $M \leq t$ for each $M \in G$;*
- *for any subset $\tau \subseteq \sigma$ a binary relation $t_1 =_\tau t_2$*

The universe of the structure of $\mathcal{A}_{\sigma,G,\mathcal{E}}$ is $T(\Sigma)\downarrow_R$, that is the set of normal forms of $T(\Sigma)$ w.r.t. the term rewrite system R.

- $\mathcal{A}_{\sigma,G,\mathcal{E}}, v \models t = M$ *iff* $v(t) = M$*;*
- $\mathcal{A}_{\sigma,G,\mathcal{E}}, v \models M \leq t$ *iff* $M \in \textit{subterms}(v(t))$
- $\mathcal{A}_{\sigma,G,\mathcal{E}}, v \models t_1 =_\tau t_2$ *iff* $\mathcal{E} \cup \tau \vdash v(t_1) = v(t_2)$

Note that this structure, in contrast to the Herbrand structure, does not feature function symbols in the logical language. As a consequence, one can not express a ternary relation like $t_1 = f(t_2, t_3)$.

We can identify any subsystem $\{x_1 = M_1, \ldots, x_i = M_i\}$ with the replacement $[x_1 \mapsto M_1, \ldots, x_i \mapsto M_n]$. Then we obtain thanks to the cycle-freeness of σ (and hence, cycle-freeness of any of its subsystems) that

Proposition D.6 *Let $\tau \subseteq \sigma$. Then for all $N_1, N_2 \in T(\Sigma)$ $\tau \vdash N_1 = N_2$ iff $N_1\tau = N_2\tau$.*

Since no symbol of D_τ appears in R, we also obtain that

Proposition D.7 *Let $\tau \subseteq \sigma$. Then for all $N_1, N_2 \in T(\Sigma)$ $\mathcal{E} \cup \tau \vdash N_1 = N_2$ iff $N_1\tau \downarrow_R = N_2\tau \downarrow_R$.*

Lemma D.8 *The structure $\mathcal{A}_{\sigma,G,\mathcal{E}}$ is tree-automatic for any σ, G, and $\mathcal{E}$ with the restrictions given above.*

Proof We define $L_\delta = T(\Sigma) \downarrow_R$ (which is a regular language due to the fact that R is left-linear [6]), and $\nu(x) = x$ for any $x \in T(\Sigma)$.

It remains to define the automata for the three kinds of relations. The automaton for $t = M$ is as follows:

- States $Q_{=M} = \{q_s \mid s \in \texttt{subterms}(M)\}$
- Accepting states $F_{=M} = \{q_M\}$
- Transition rules $f(q_{M_1}, \ldots, q_{M_n}) \to q_{f(M_1,\ldots,M_n)}$ if all M_i are proper subterms of M

Proposition D.9 *For any $N \in L_\delta$, $N \to^* q_M$ iff $N = M$.*

The automaton for $M \leq t$ is an extension of the automaton for $t = M$:

- States $Q_{M\leq} = \{q_s \mid s \in \texttt{subterms}(M), s \neq M\} \cup \{q_*, q_{M\leq}\}$
- Accepting states $F_{M\leq} = \{q_{M\leq}\}$
- Transition rules:
  - $f(q_{M_1}, \ldots, q_{M_n}) \to q_{f(M_1,\ldots,M_n)}$ if all M_i are proper subterms of M and $M \neq f(M_1, \ldots, M_n)$
  - $f(q_{M_1}, \ldots, q_{M_n}) \to q_{\leq M}$
    if $M = f(M_1, \ldots, M_n)$
  - $f(q_*, \ldots, q_*) \to q_*$ for any $f \in \Sigma, n \geq 0$

– $f(q_*, \ldots, q_*, q_{M\leq}, q_*, \ldots, q_*) \to q_{M\leq}$ for any $f \in \Sigma$ and $n \geq 1$.

Proposition D.10 *For any $N \in L_\delta$,
$N \to^* q_{M\leq}$ iff $M \in \mathtt{subterms}(N)$.*

The idea of the automaton for $x =_\tau y$ is that it keeps in its finite memory track of the normalization of his input under the replacement τ and the term rewrite system R. Once one of the two input terms, after having applied τ and R, lies outside of S the automaton checks that the normalized values are the same and from then on verifies that he sees the same input symbols on both components. Let

$$S = \bigcup_{i=1}^{i=n} \mathtt{subterms}(M_i)$$

The automaton is defined as follows:

- States $Q_\tau =$
  $\{q_=\} \cup \{q_{N_1,N_2} \mid N_1, N_2 \in S \cup \{\Box\}\} \setminus \{q_{\Box,\Box}\}$
- Accepting states $F_\tau = \{q_=\}$
- Transition rules:
  - $(f, \Box)(q_{N_1,\Box}, \ldots, q_{N_n,\Box}) \to q_{N,\Box}$
    if $N = f(N_1, \ldots, N_n) \downarrow_R \in S$, $f \notin D_\tau$.
  - $(\Box, f)(q_{\Box,N_1}, \ldots, q_{\Box,N_n,}) \to q_{\Box,N}$
    if $N = f(N_1, \ldots, N_n) \downarrow_R \in S$, $f \notin D_\tau$.
  - $(f, g)(q_{N_1,K_1}, \ldots, q_{N_n,K_n}) \to q_{N,K}$
    if $N = f(N_1, \ldots, N_{\mathrm{ar}(f)}) \downarrow_R \in S$
    and $K = f(K_1, \ldots, K_{\mathrm{ar}(g)}) \downarrow_R \in S$
    and $f, g \notin D_\tau$.
  - $(f, g)(q_{N_1,K_1}, \ldots, q_{N_n,K_n}) \to q_=$
    if $f(N_1, \ldots, N_{\mathrm{ar}(f)}) \downarrow_R$
    $= g(K_1, \ldots, K_{\mathrm{ar}(g)}) \downarrow_R$
    and $f, g \notin D_\tau$.
  - $(f, f)(q_=, \ldots, q_=) \to q_=$ if $f \notin D_\tau$.
  - $(x, \Box) \to q_{M,\Box}$ if $(x = M) \in \tau$.
  - $(\Box, x) \to q_{\Box,M}$ if $(x = M) \in \tau$.
  - $(x_1, x_2) \to q_{M_1,M_2}$
    if $(x_1 = M_1), (x_2 = M_2) \in \tau$.

– $(f, x)(q_{N_1,\Box}, \ldots, q_{N_n,\Box}) \to f_{N,M}$
if $N = f(N_1, \ldots, N_n) \downarrow_R \in S$
and $(x = M) \in \tau$, $f \notin D_\tau$.

– $(x, f)(q_{\Box,N_1}, \ldots, q_{\Box,N_n}) \to f_{M,N}$
if $N = f(N_1, \ldots, N_n) \downarrow_R \in S$
and $(x = M) \in \tau$, $f \notin D_\tau$.

Proposition D.11 *For every $u \in T(\Sigma_2^\Box)$:*

- *If $u \to^* q_{N,\Box}$ then $N = \pi_1(u)\tau \downarrow_R$.*
- *If $u \to^* q_{\Box,N}$ then $N = \pi_2(u)\tau \downarrow_R$.*
- *If $u \to^* q_{N_1,N_2}$ then $N_1 = \pi_1(u)\tau \downarrow_R$ and $N_1 = (\pi_2(u)\tau) \downarrow_R$.*
- *If $u \to^* q_=$ then $\pi_1(u) \downarrow_R = \pi_2(u) \downarrow_R$*

This is shown very easily by induction on the tree u.

The following proposition states that the automaton accepts only pairs of trees that are congruent modulo $\mathcal{E} \cup \sigma$.

Proposition D.12 *For every $N_1, N_2 \in L_\delta$: If $N_1 \otimes N_2 \to^* q_=$ then $\mathcal{E} \cup \tau \vdash N_1 = N_2$.*

Proof By Proposition D.11, if $N_1 \otimes N_2 \to^* q_=$ then $(N_1\tau) \downarrow_R = (N_2\tau) \downarrow_R$. This implies by Proposition D.7 that $\mathcal{E} \cup \tau \vdash N_1 = N_2$. □

In order to show completeness of our automaton construction we define an auxiliary ground rewrite system:

$$\begin{array}{rcl} R' & = & \{f(N) \to f(N) \downarrow_R \mid N \in S, f(N) \downarrow_R \in S\} \\ & \cup & \{x \to M \mid (x = M) \in \tau\} \end{array}$$

This rewrite system describes the rewrite steps that are "hardwired" in the automaton.

Proposition D.13 *The system R' is confluent and terminating.*

Proof Termination is a consequence of the fact that rewriting by R or τ cannot introduce new occurrences of symbols in D_τ, and that R is terminating. Local confluence is a consequence of the fact that R is confluent, and no left-hand side of R matches a symbol in D_τ, and local confluence together with termination implies confluence. □

We recall that the right-hand sides of τ are assumed to be in normal form with respect to R.

Proposition D.14 *Let N be in normal from w.r.t. R. Then*

$$N\tau\downarrow_R = N\downarrow_{R'}$$

Proof Obviously if $N_1 \rightarrow^* N_2$ by R' then $N_1\tau \rightarrow^* N_2$ by R. Since R is confluent it is hence sufficient to show that $N\downarrow_{R'}$ is in normal form w.r.t. R.

Let us write $\tau = \{x_1 = M_1, \ldots, x_n = M_n\}$. First, observe that we can write N as $N[x_1, \ldots, x_n]$ when N is a context, and that $N\tau = N[M_1, \ldots, M_n]$. One shows easily by induction on the number of rewrite steps that if N, M_1, ..., M_n are in normal w.r.t. R, if $M_1, \ldots, M_n \in S$ and $N[M_1, \ldots, M_n] \rightarrow^*_{R'} u$ then u can be written in the form $N'[M'_1, \ldots, M'_m]$ where $N', M'_1, \ldots, M'_n$ are in normal form w.r.t. R and $M'_1, \ldots, M'_n \in S$. Any such term that is reducible by R is also reducible by R'. This is a consequence of the particular form of the rewrite rules of R: Suppose that $N', M'_1, \ldots, M'_n$ are in normal form w.r.t. R and that $N'[M'_1, \ldots, M'_n]$ is reducible by R. The redex cannot entirely lie in N' or entirely in one of the M'_i since these terms are assumed to be in normal form. Hence, there is are N'' and a $M''_1, \ldots, M''_k$ such that $M'_i = g(M''_1, \ldots, M''_k)$, $N'[M'_1, \ldots, M'_n] = N''[M'_1, \ldots, f(g(M''_1, \ldots, M''_l)), \ldots, M'_n]$ where the $M''_1, \ldots, M''_k \in S$ (since $M'_i \in S$ and S is closed under subterms). As a consequence, the term is reducible by R'. $\square$

Proposition D.15 *For every $N_1, N_2 \in L_\delta$: If $\mathcal{E} \cup \tau \vdash N_1 = N_2$ then $N_1 \otimes N_2 \rightarrow^* q_=$.*

Proof This is a consequence of Proposition D.14 since the automaton performs normalization with respect to the system R': $(N_1 \otimes N_2) \rightarrow^* q_{N_1\downarrow_{R'}, N_2\downarrow_{R'}}$ provided that $N_1\downarrow_{R'}, N_2\downarrow_{R'} \in S$. $\square$

This completes the proof of Lemma D.8. $\square$

Here is an example of the automaton construction for $t_1 =_\tau t_2$. Let the equational theory $\mathcal{E}$ be

$$\begin{array}{rcl} l(p(t_1, t_2)) & = & t_1 \\ r(p(t_1, t_2)) & = & t_2 \end{array}$$

and τ be

$$\begin{array}{rcl} x_1 & = & p(a, p(b, c)) \\ x_2 & = & p(c, b) \end{array}$$

We will show how the automaton will recognize that

$$f(d, r(r(x_1))) =_\tau f(d, l(x_2))$$

The execution steps of the automaton on the convolution of these two trees are

$$
\begin{aligned}
& [f,f]([d,d],[r,l]([r,x_2]([x_1,\square]))) \\
\rightarrow\ & [f,f](q_{=},[r,l]([r,x_2](q_{p(a,p(b,c)),\square}))) \\
\rightarrow\ & [f,f](q_{=},[r,l](q_{p(b,c),p(c,b)})) \\
\rightarrow\ & [f,f](q_{=},q_{=}) \\
\rightarrow\ & q_{=}
\end{aligned}
$$