Distrib. Comput. (2010) 23:61–83
DOI 10.1007/s00446-010-0112-6

A spatial equational logic for the applied π-calculus

Étienne Lozes · Jules Villard

Received: 11 December 2008 / Accepted: 31 May 2010 / Published online: 13 July 2010

Abstract Spatial logics have been proposed to reason locally and modularly on algebraic models of distributed systems. In this paper we define the spatial equational logic $\mathsf{A}\pi\mathsf{L}$ whose models are processes of the applied π-calculus. This extension of the π-calculus allows term manipulation and records communications as aliases in a frame, thus augmenting the predefined underlying equational theory. Our logic allows one 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 $\mathsf{A}\pi\mathsf{L}$, and show in particular that the whole logic induces a coarser equivalence than structural congruence. We give characteristic formulae for some of these equivalences and for static equivalence. Going further into the exploration of $\mathsf{A}\pi\mathsf{L}$'s expressivity, we also show that it can eliminate standard term quantification.

Keywords Spatial Logic · Applied pi-calculus

1 Introduction

1.1 Spatial logics

Spatial logics have been proposed to reason locally and modularly on algebraic models of distributed systems such as ambients [14] or π-calculus [8]. In a broad sense, spatial logics may also include separation logic [26] and tree logics [11], although these logics deal with static models. Two essential connectives in these logics are spatial conjunction and adjunct. The spatial conjunction $A * B$ expresses that property A hold on one part of the system, and property B hold on another, disjoint, part of the system. Disjointness plays a crucial role in separation logic to express non-aliasing properties, and in π-calculus spatial logic to express race-freedom properties, to cite a few examples. Adjunct $A \mathrel{-\!\!*} B$ expresses that, once placed in an environment satisfying A, the system will satisfy B. While spatial conjunction supports a form of *local reasoning* and bottom-up specification, the adjunct accounts for a certain form of *contextual reasoning*. Contextual reasoning, that is ensuring some properties of a system provided its environment enjoys some other properties, is the key idea of many specification techniques, such as rely/guarantee reasoning. Bisimilarity also is a form of contextual reasoning (although in a weaker sense), as it establishes statements of the form "the system will behave in a certain way under any environment."

1.2 Cryptographic protocols

Contextual reasoning is very important in cryptographic protocols. For instance, secrecy under a passive or active attacker is a certain safety property—the secret cannot be deduced—that the cryptographic protocol should guarantee in whatever environment it is run. Whether the environment is active or passive already is an example of conditions one may wish to impose on the environment.

In the applied π-calculus, this distinction is emphasized by introducing two different equivalences: static equivalence and bisimilarity. However, cryptographic properties often need to be more precise about what is assumed from the

Extended abstract originally published in the proceedings of CONCUR'08 conference [22].

É. Lozes (✉) · J. Villard
LSV, ENS Cachan, CNRS – 61 av. du pdt Wilson,
94230 Cachan, France
e-mail: lozes@lsv.ens-cachan.fr

J. Villard
e-mail: villard@lsv.ens-cachan.fr

environment. For instance, private channels or secret keys are usually modeled by name restriction in applied π-calculus, but the frontier between what is public and what is private is only meaningful when the attacker is the environment and the protocol is "fair." For some protocols the separation is not so clear. To give an example, in electronic voting protocols, coercion-resistance is a secrecy property of the vote administrator, for whom the environment involves two agents with different abilities and objectives: a voter, who needs to follow a given vote protocol, and a coercer who forces her to reveal information that can prove that she voted in a certain way. Coercion-resistance turned out to be hard to reduce to standard secrecy, and required the introduction of a new simulation relation [12].

This form of complex contextual reasoning seems to underline the limitations of which specifications can be expressed using bisimulation. The fact that adjunct allows to express complex contextual reasoning in a natural manner motivated us to introduce a spatial logic for the applied π-calculus.

1.3 Spatial logics and standard process equivalences

Spatial logics, and similarly modal logics, may provide a more flexible and manageable alternative than process equivalences. In an ideal world, the logical equivalence coincides with a well-identified process equivalence $\approx$, and the logic admits characteristic formulae. Then, checking whether $P \approx Q$ reduces to checking that P satisfies the characteristic formula of Q, or that the characteristic formulae of P and Q are equivalent.

Spatial logics live in such an ideal world for the π-calculus, the ambient calculus, the static ambient calculus and the memory model of separation logic, for various kinds of equivalences. The spatial conjunction is the cause of a very intensional discriminating power for spatial logics [16,17] as logical equivalence coincides with structural congruence. Dropping this connective and reasoning only with adjunct $-\!\!*$ usually yields extensional equivalences such as barb equivalence [15]. Albeit quite intuitive, these results depend on the nature of the process model they deal with, and should be treated with care: the intensional equivalence might be much coarser than structural congruence, and logical equivalence is not a congruence in the general case, even in the presence of adjunct.

1.4 A spatial equational logic

In this paper we investigate a spatial equational logic for the applied π-calculus [2], an extension of the π-calculus [24] where processes may communicate terms through channels. These terms are tested for equality using an equational theory, which is global, as well as local axioms placed in a *frame*, which act as a record of what has been sent to the environment so far.

For example, the frame $\Phi = \{^{\mathrm{enc}(s,y)}/_{x}\} \mid \{^{\mathrm{pk}(n)}/_{y}\}$, that we will use as an example throughout this paper, augments the equational theory with the knowledge that x is an alias for a message encrypted with a public key, itself aliased by y. To reflect the private nature of s and n, these names will be hidden, and we will write $\nu n, s.\, \Phi$.

More precisely, each term sent to the environment is stored in an *active substitution* that acts as a local alias that assigns a variable to this term. For instance, $\nu n, s.\, \Phi$ can be obtained from a process $\nu n.\, \{^{\mathrm{pk}(n)}/_{y}\} \mid \nu s.\, \bar{a}\langle \mathrm{enc}(s, y)\rangle.P$ by sending $\mathrm{enc}(s, y)$ to an environment listening on channel a, thus reducing to $\nu n, s.\, \Phi \mid P$.

This peculiar aspect of the calculus raises new questions with respect to standard expressiveness issues. First, it is not clear how this logic should be designed in order to live in the same ideal world as spatial logics for π-calculus. Second, one could hope to characterize the static equivalence for frames introduced in the applied π-calculus with a static fragment of the logic. This objective is very appealing, since static equivalence plays a central role in cryptographic properties. Third, we may wonder what is the correct notion of separation for frames.

1.5 Contributions

Our first contribution is the characterization of the logical equivalences of the static, static extensional, and dynamic fragments. Static fragments are obtained forbidding the use of the temporal modality, and the static extensional fragment is obtained by furthermore forbidding the use of the spatial conjunction. These 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 equivalence coincides with static equivalence.

It should be underlined that, to our knowledge, this is the first logical characterization of static equivalence that is independent from the equational theory: no constraint is put on the equational theory. All the constructions involved are rather simple compared to other logical characterizations of frame equivalence [19] for specific classes of equational theories. Moreover, characteristic formulae are derivable for both intensional and extensional equivalences.

Our second contribution deals with the logical equivalence of the dynamic intensional fragment. Surprisingly, we show that the logic cannot distinguish messages with similar information content. As a consequence, this equivalence is coarser than mere structural congruence. Moreover, we show that it is not a congruence, due to the possible introduction of noise in communications that the logic may not detect. This noticeably complicates the techniques to obtain an axiomatization

of this equivalence. We point out some admissible axioms for logical equivalence and prove this axiomatization complete for the equational theory of finite trees.

The following table summarizes our results (last two lines) with respect to previous expressiveness results:

Calculus	Extensional fragment	Full logic
Mobile Ambients	unknown	structural congruence [18]
Static Ambients	unknown	structural congruence [10]
CCS	unknown	structural congruence + renaming [9]
π-calculus	barb congruence [15]	structural congruence [16]
Frames	static equivalence	structural congruence
Applied π-calculus	unknown	structural congruence + noise

Our third contribution is a quantifier elimination technique that shows that the standard term quantifiers $\exists t.\, A$ can be mimicked by spatial connectives, which gives one more example of a spatial logic more expressive than first-order logic.

1.6 Decision procedures

To be of practical interest, the ideal world mentioned above should include a decision procedure for the model-checking or for the satisfiability problems of the logic. These problems are known to be hard for spatial logics, even for very simple calculi such as CCS without recursion [9]. However, some recent work introduces a new technique based on well-structured transition systems [3]. In this work, processes are typed, and only a positive fragment of the spatial logic can be checked. Adapting these ideas to the applied π-calculus is far out of the scope of the present work, but may be promising.

Deciding our logic is hard for another reason: as static equivalence is already undecidable [7], so is the model-checking problem for the static fragment of our logic. However, a lot of work has been put into obtaining decision procedures for static equivalence against some restricted classes of equational theories [1]. Similarly, the `ProVerif` tool [5] can decide restricted forms of bisimilarity, which has practical benefits to prove secrecy properties. One of our initial motivations was to investigate whether the spatial logic for frames could introduce new decidable classes of static equivalence. As already mentioned in a technical report [28], we can decide a fragment of the spatial logics for frames using tree automata techniques. The proof relies on some conditions on the equational theories, which unfortunately excluded lots of interesting equational theories commonly used in cryptographic examples. Recently [20], we investigated further this approach, and observed that even small relaxations of the conditions we put on the equational theory already yield undecidability. Thus we tend to believe that the fragment of logic handled by tree automata technique is not the most interesting one, and some useful and decidable fragments of our spatial logic still need to be identified.

1.7 Proof techniques for logical characterization

The goal of this work is to introduce a spatial logic that captures interesting logical equivalence, and this goal drives some of our choices in the design of the logic, considering that all spatial logics present in the literature differ on their syntax and semantics: some include revelation operator whereas others do not, some include least fixed points constructs whereas others do not, *etc*. We do not claim that our choices are the only relevant ones, but we try to comment on them whenever possible. However, adapting existing proof techniques for the characterization of logical equivalences is far from trivial, and depends highly on these choices.

There are at least two approaches for establishing that a logical equivalence $=_L$ coincides with a process equivalence $\doteq$ (extensional or intensional), provided $=_L$ is already such that $\doteq \subseteq =_L$: the first approach consists in defining, for each process P, a characteristic formula ϕ_P such that the models of ϕ_P exactly are the processes Q such that $P \doteq Q$. This is the method used for instance for CCS [9]. The result obtained from this method is stronger than a pure characterization of $=_L$, thus this method may fail even if the logic does characterize $\doteq$. The second method, due to Sangiorgi, brings to mind the proof that logical equivalence in Hennessy-Milner logic coincides with barb congruence. It goes through an intermediate equivalence $\approx_{int}$, called intensional bisimilarity [18], which can be proved (hopefully easily) to coincide with $\doteq$. Then the characterization of logical equivalence boils down to two properties. The first one, called soundness by Sangiorgi [27], states that $\approx_{int} \subseteq =_L$, and the second one, completeness, states that $=_L \subseteq \approx_{int}$. In the proof of Sangiorgi [27], and in its extension to infinitary ambients [18], soundness is rather easy to prove, since $\approx_{int}$ is almost the Ehrenfeucht-Fraïssé game of the logic. The rules of the adjuncts in the Ehrenfeucht-Fraïssé game are however omitted in the definition of $\approx_{int}$; for instance, the rule for $\triangleright$ would be: *if* $P_1 \approx_{int} Q_1$, *then for all* P_2 *there is* Q_2 *such that* $P_1 \mid P_2 \approx_{int} Q_1 \mid Q_2$ *and* $P_2 \approx_{int} Q_2$. Instead, Sangiorgi establishes that $\approx_{int}$ is a congruence, which in particular implies that the rules of adjuncts are admissible. The completeness proof of Sangiorgi goes through two steps: first, $\approx_{int}$ is identified with the ω-limit $\bigcap_{i \geq 0} \approx_i$ of its finite approximants $\approx_i$; second $\not\approx_i \subseteq \neq_L$ is proved for all $i \geq 0$ by induction, considering formulae that express the conditions of $\approx_{int}$. A key property for both steps of the completeness proof is that, for fixed P and Q, $P \approx_{i+1} Q$ holds if and only if only $P' \approx_i Q'$ for a finite set of such pairs P', Q' that depend only on P and Q, with some additional conditions. This property may not hold, for instance if the calculus is not image-finite.

We rely on the first method for characterizing the static equivalences, as characteristic formulae are rather easy to define in this case. However, for the logical equivalence of the whole logic, neither are we able to define characteristic formulae for $=_L$, nor do we establish that intensional bisimilarity coincides with $=_L$. We instead combine the two methods mentioned before: we introduce an intensional bisimilarity and establish that (1) $\equiv' \subseteq \approx_{int}$, where $\equiv'$ is the extended structural equivalence we aim at characterizing, (2) $\approx_{int}$ is sound, or in other words $\approx_{int} \subseteq =_L$, which holds only if the rules of the adjuncts are included in the definition of $\approx_{int}$, and (3) there exist characteristic formulae for $\equiv'$ when the equational theory over terms is the one of finite trees. Since steps (1) and (2) of the proof do not depend on the equational theory, we are able to prove that $=_L$ is not a congruence whatever the equational theory. This is what makes the rules for the adjuncts of the definition of $\approx_{int}$ mandatory. Their presence has important consequences on the completeness proof. Although we do not consider infinitary processes, $\approx_{int}$ cannot be proved to be the ω-limit $\bigcap_{i\geq 0} \approx_i$ of its finite approximants. There are proofs that intensional bisimilarity coincides with the limit of its approximants even with image-infiniteness: either using local characteristic formulae [18], or using well-ordering arguments [4]. However, it seems challenging to adapt these proofs to our setting. Instead, we let the question of whether $\approx_{int} = \bigcap_{i\geq 0} \approx_i$ open, as we do not need to answer it to characterize logical equivalence with our approach.

1.8 Structure of the paper

In Sect. 2 we collect all the necessary background on the applied π-calculus, and define our process compositions $*$ and $\|$. Section 3 introduces $\mathsf{A}\pi\mathsf{L}$. Sections 4 and 5 present the characterizations of the logical equivalences for the static and dynamic fragments respectively. Section 6 establishes the quantifier elimination result. A table summing up the auxiliary formulae used throughout this paper can be found in the appendix.

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*. This lets the user decide for example 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, where $ar(f)$ is the arity of f, $x \in \mathcal{V}$ and $a \in \mathcal{N}$:

$$M, N ::= x \mid a \mid f(M_1, \ldots, M_{ar(f)})$$

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" that may belong either to $\mathcal{N}$ or to $\mathcal{V}$. We will write M, N for terms. $fn(M)$ and $fv(M)$ respectively denote the sets of free names and free variables of M, defined as usual, and $fnv(M) \triangleq fn(M) \cup fv(M)$.

These terms come equipped with an equivalence relation $\mathcal{E}$ called an equational theory on Σ, where membership of a pair (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 $M[x \leftarrow N_1] = M[x \leftarrow N_2]$). In particular, syntactic equality on terms is an equational theory that satisfies these conditions, called the theory of finite trees in this paper.

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 which account for the dynamic part, and the *extended* ones, also simply referred to as "processes" which extend the former with a *static* part. Note that replication $!P^p$ is not part of our setting.

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

Our grammar differs from the original one in that it allows two kinds of communications that do 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. Another solution would be to define a type system for the terms, as in the original applied π-calculus; although this would not change our results, we prefer to avoid the complications this would lead to.

The set of free names (resp. variables) of a process P is defined as usual and written $fn(P)$ (resp. $fv(P)$), with $fn(\{^M/_x\}) \triangleq fn(M)$ (resp. $fv(\{^M/_x\}) \triangleq \{x\} \cup fv(M)$), and

with both restrictions and both inputs being binders. We write $fnv(P)$ for the set $fn(P) \cup fv(P)$. We may also write *e.g.* $fn(P, M, N)$ for $fn(P) \cup fn(M) \cup fn(N)$.

Compositions of active substitutions of the form $\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_n}/_{x_n}\}$ will be written $\{^{\mathbf{M}}/_{\mathbf{x}}\}$, and referred to using σ, τ. Depending on the context, we will write $\mathbf{x}$ either for the vector $x_1, \ldots, x_n$ or for the associated set $\{x_1, \ldots, x_n\}$, where $n = |\mathbf{x}|$. Trailing $\mathbf{0}$'s in processes will often be omitted, as well as null *else* branches in conditionals.

From now on, and as usual, we will only consider extended processes whose active substitutions are cycle-free, and we will always assume that there is at most one active substitution for each variable, and *exactly one* if the variable is restricted.

Definition 1 (Well-formed processes) A process P is said to be well-formed if the following conditions are satisfied:

1. If $\{^{M_1}/_{x_1}\},\ldots,\{^{M_n}/_{x_n}\}$ are the active substitutions that appear in P, then the oriented graph whose vertices are these active substitutions, and where there is an edge from $\{^{M_i}/_{x_i}\}$ to $\{^{M_j}/_{x_j}\}$ iff $x_i \in fv(M_j)$ is acyclic.
2. There is at most one active substitution for each variable.
3. There is *exactly* one active substitution for each *restricted* variable.

2.3 Operational semantics

The structural congruence relation $\equiv$ identifies processes that can be obtained one from another by mere rewriting. It is called congruence as it is closed by context applications.

Definition 2 (Context, evaluation context) A context (resp. an evaluation context) is an extended process with a hole in place of a plain (resp. extended) process.

This hole can be filled with any extended process that makes the resulting extended process well-formed.

Definition 3 (Structural congruence) Structural congruence 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 following rules:

PAR- 0	P	$\equiv$	$P \mid \mathbf{0}$
PAR- A	$P \mid (Q \mid R)$	$\equiv$	$(P \mid Q) \mid R$
PAR- C	$P \mid Q$	$\equiv$	$Q \mid P$
NEW- 0	$\nu u.\, \mathbf{0}$	$\equiv$	$\mathbf{0}$
NEW- NEW	$\nu u.\, \nu v.\, P$	$\equiv$	$\nu v.\, \nu u.\, P$
NEW- PAR	$P \mid \nu u.\, Q$	$\equiv$	$\nu u.\, (P \mid Q)$ when $u \notin fnv(P)$
ALIAS	$\nu x.\, (\{^{M}/_{x}\} \mid P)$	$\equiv$	$P[x \leftarrow M]$
SUBST	$\{^{M}/_{x}\} \mid P^{p}$	$\equiv$	$\{^{M}/_{x}\} \mid P^{p}[x \leftarrow M]$
REWRITE	$\{^{M}/_{x}\}$	$\equiv$	$\{^{N}/_{x}\}$ if $\mathcal{E} \vdash M = N$

The original structural congruence [2] is slightly different from ours:

- It is closed by application of evaluation contexts instead of arbitrary contexts. This makes inductive characterization of processes up to $\equiv$ impossible (see Sect. 5).
- The original rules ALIAS' and SUBST' are as follows:

ALIAS'	$\nu x.\, \{^{M}/_{x}\} \equiv \mathbf{0}$
SUBST'	$\{^{M}/_{x}\} \mid P \equiv \{^{M}/_{x}\} \mid P[x \leftarrow M]$

With our own ALIAS and SUBST rules, active substitutions may affect other active substitutions only if their domain is a restricted variable. They may 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 makes our quantifier elimination technique easier, as it dramatically limits the interferences between active substitutions. In the absence of adjuncts, it could have been useful (although not necessary) for characterizing intensional bisimilarity as a limit of its finite approximants (as described in the introduction), since our notion of structural congruence allows only finitely many ways of splitting a process whereas, for the original structural congruence and for a fixed P, there may be infinitely many non-congruent pairs of processes P_1, P_2 such that $P \equiv P_1 \mid P_2$. Aside from these technical considerations, more informal ones pressed us to adopt this definition:

- Our definition does not change the behavior of processes, as both the reduction rules and the static equivalence definitions, presented below, stay the same. Moreover, as it was originally the case, each process P can be written as $\nu \mathbf{n}.\, (\sigma \mid Q^{p})$ for some set of restricted names $\mathbf{n}$, some parallel composition of active substitutions σ and some public plain process Q^{p}.
- We claim that the results presented in this paper would hold with either notion of structural congruence without a lot of changes, except for quantifier elimination.
- It could moreover be argued that our definition better supports the idea of active substitutions being viewed as logging information about past communications than it was the case with the standard congruence: consider a process that sends two messages M and N, with $N = f(M)$, resulting in the process $P' \mid \{^{M}/_{x}\} \mid \{^{N}/_{y}\}$. Consider similarly a process that sends M and the "recipe" $f(x)$ to produce N, ending in $P' \mid \{^{M}/_{x}\} \mid \{^{f(x)}/_{y}\}$. Then $\equiv$ distinguishes between these two processes, whereas the original structural congruence does not.

Due to the REWRITE rule, two structurally congruent processes may not have the same set of free names or variables. Thus, we define the closures of these sets up to structural

congruence $\overline{fn}(P), \overline{fv}(P), \overline{fnv}(P)$ and the corresponding sets for terms:

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

Similarly, a process will be considered plain even in the presence of active substitutions over *hidden* variables. Doing so makes the class of plain processes stable by structural congruence, as the ALIAS rule (as well as ALIAS', for that matter) may introduce such restricted active substitutions.

Finally, let us recall the definition of internal reduction:

Definition 4 (Reduction) Internal reduction $\rightarrow$ is the smallest relation that is closed by structural congruence and by application of evaluation contexts and that satisfies the rules below:

COMM- T	$\bar{a}\langle x\rangle.P^p \mid a(x).Q^p \rightarrow P^p \mid Q^p$
COMM- C	$\bar{a}^{ch}\langle m\rangle.P^p \mid a^{ch}(n).Q^p \rightarrow P^p \mid Q^p[n \leftarrow m]$
THEN	$\mathit{if}\ M = M\ \mathit{then}\ P^p\ \mathit{else}\ Q^p \rightarrow P^p$
ELSE	$\mathit{if}\ M = N\ \mathit{then}\ P^p\ \mathit{else}\ Q^p \rightarrow Q^p$ (when $fv(M, N) = \emptyset$ and $\mathcal{E} \not\vdash M = N$)

Remark 1 With these rules, and as explained by Abadì and Fournet [2], the communication of a term M happens as follow:

$$\begin{aligned}
&\bar{a}\langle M\rangle.P^p \mid a(x).Q^p \\
&\quad \equiv \nu x.\,(\{^M/_x\} \mid \bar{a}\langle x\rangle.P^p) \mid a(x).Q^p && \text{by ALIAS} \\
&\quad \equiv \nu x.\,(\{^M/_x\} \mid \bar{a}\langle x\rangle.P^p \mid a(x).Q^p) && \text{by NEW- PAR} \\
&\quad \rightarrow \nu x.\,(\{^M/_x\} \mid P^p \mid Q^p) && \text{by COMM- T} \\
&\quad \equiv P^p \mid Q^p[x \leftarrow M] && \text{by ALIAS}
\end{aligned}$$

One obtains thus that $\bar{a}\langle M\rangle.P^p \mid a(x).Q^p \rightarrow P^p \mid Q^p[x \leftarrow M]$ as expected.

Remark 2 Whenever $\mathcal{E} \vdash M = N$, we have $\bar{a}\langle M\rangle \equiv \bar{a}\langle N\rangle$ using ALIAS and REWRITE.

2.4 Frames

A *frame* is an extended process built up solely from active substitutions and the null process, using parallel composition and name and variable restrictions. In other words, it is a process with no plain parts except for $\mathbf{0}$. The frame $\phi(P)$ of a process P is P in which every embedded plain process is set to $\mathbf{0}$. The *domain* $dom(\phi)$ of a frame ϕ (resp. $dom(P)$ of a process P) is the set of variables upon which the active substitutions of ϕ (resp. $\phi(P)$) act.

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 1 *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 simply by setting every plain process embedded into P or Q to $\mathbf{0}$ into the proof, and by suppressing branches that prove two guarded plain processes to be congruent, as the equality to prove will be $\mathbf{0} \equiv \mathbf{0}$. □

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 $\bar{a}\langle x\rangle \equiv \bar{a}\langle y\rangle$.

The following three definitions are standard in the applied π-calculus.

Definition 5 (Closed frame, process, and closing context) A frame ϕ (resp. a process P) is *closed* when $fv(\phi) \subseteq dom(\phi)$ (resp. $fv(P) \subseteq dom(P)$). An evaluation context $C[\cdot]$ *closes* the frame ϕ (resp. the process P) when $C[\phi]$ (resp. $C[P]$) is both well-formed and closed.

Definition 6 (Term equality) Two terms M and N are equal in the frame ϕ, written $\phi \vdash M = N$ when there exists a set of names $\mathbf{n}$ and a substitution σ (*i.e.* a public frame) such that $\phi \equiv \nu\mathbf{n}.\,\sigma$, $M\sigma = N\sigma$ and $\mathbf{n} \cap fn(M, N) = \emptyset$. Two terms are equal in the process P when they are equal in $\phi(P)$.

Definition 7 (Static equivalence) Two closed frames ϕ and ψ are *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 when their frames are.

As it is, static equivalence is a congruence on closed frames, but not on non-closed ones. For instance, with $\langle \cdot, \cdot \rangle$ being the pairing operation and π_1 the first projection, if we let $\phi = \nu n.\,\{^{\mathrm{dec}(\mathrm{enc}(\langle 1,n\rangle,1),y)}/_x\}$ and $\psi = \nu n.\,\{^{\mathrm{dec}(\mathrm{enc}(\langle 1,n\rangle,1),z)}/_x\}$ then $\phi \approx_s \psi$, but $\pi_1(x) = 1$ holds in the frame $\{^1/_y\} \mid \phi$ and not in $\{^1/_y\} \mid \psi$.

To overcome this issue, and later be able to write formulae characterizing static equivalence for both closed and non-closed frames, we introduce *strong* static equivalence $\approx_s^s$.

Definition 8 (Strong static equivalence) Strong static equivalence $\approx_s^s$ is the largest equivalence relation included in $\approx_s$ and closed by application of closing evaluation contexts.

One can note that strong static equivalence is closed by application of arbitrary evaluation contexts (and not just closing ones), and that it coincides with static equivalence on closed frames.

2.5 Additional operators

One of the core operators of spatial logics is the spatial conjunction $|$, which allows local reasoning. A formula of the form $A \mid B$ is satisfied by a process which can be split in a *disjoint* manner into a parallel composition of two processes such that one of them satisfies the formula A and the other satisfies B. Here, disjointness is read as "not sharing any secret name." In this section, we introduce two finer-grained meanings for "disjointness" that we feel are a better fit for the applied π-calculus, by the mean of two novel operators on processes.

Consider a protocol $F = \nu n, n'.(\{^{\mathrm{ck}(n,n')}/_x\} \mid A(x,n) \mid H(x,n'))$ where Abelard and Héloïse share a compound key $\mathrm{ck}(n,n')$ generated from a nonce n of $A(x,n)$ and n' of $H(x,n')$. $A(x,n)$ and $H(x,n')$ are plain processes where the variable x and names n, n' appear free. Then we cannot write a specification of the form $\mathsf{A} \mid \mathsf{H}$ for this process, as the active substitution on x binds the two plain processes together under the restriction $\nu n, n'$. To overcome this limitation, we introduce a new operator $\mathbin{\text{¦}}$ which allows us to separate processes in a subtle way: the frame of the process is copied on both sides, but the plain part of the process is split in two pieces, provided they do not share private names. The same issue is raised when we try to cut into the frame, so we define $*$ that splits the frame into disjoint parts w.r.t. a common plain process.

Another justification for having two different kinds of "knives" for applied π-calculus processes is that, when splitting up a process P into a parallel composition of two subprocesses $P_1 \mid P_2$, two very different operations are performed on a conceptual level, as both the dynamic (plain) and the static (frame) part of the process are split into two extended processes.

Finally, and perhaps more importantly, as we aim at characterizing static equivalence, we want to be able to isolate a fragment of our logic that acts as a spatial logic for *frames*, *i.e.* such that every formula of this fragment is satisfied by some process P if and only if it is satisfied by $\phi(P)$ (this is what is expressed later on by Lemma 6). This calls for a spatial conjunction that can split solely the frame of a process.

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 $\mathbin{\text{¦}}$ or $*$ is obvious, and then extend the definitions to every pair of processes which may be rewritten into two such processes.

Definition 9 (Process and frame compositions) Given frames ϕ, ϕ_1, ϕ_2, plain processes P^p, P_1^p, P_2^p and names $\mathbf{n_1}, \mathbf{n_2}$ such that $\mathbf{n_1} \cap fn(\phi_2, P_2^p) = \mathbf{n_2} \cap fn(\phi_1, P_1^p) = \emptyset$, we let

$$\nu\mathbf{n_1}.((\nu\mathbf{n_2}.\phi) \mid P_1^p) \mathbin{\text{¦}} \nu\mathbf{n_2}.((\nu\mathbf{n_1}.\phi) \mid P_2^p) \triangleq \nu\mathbf{n_1}\mathbf{n_2}.(\phi \mid P_1^p \mid P_2^p)$$
$$\nu\mathbf{n_1}.(\phi_1 \mid \nu\mathbf{n_2}.P^p) * \nu\mathbf{n_2}.(\phi_2 \mid \nu\mathbf{n_1}.P^p) \triangleq \nu\mathbf{n_1}\mathbf{n_2}.(\phi_1 \mid \phi_2 \mid P^p).$$

In the following, for $\dagger$ in $\{\mathbin{\text{¦}}, *\}$, we write $P \leftrightarrow P_1 \dagger P_2$ if there are P', P_1', P_2' such that $P \equiv P'$, $P_1 \equiv P_1'$, $P_2 \equiv P_2'$ and $P' = P_1' \dagger P_2'$.

For example, the protocol F above can be written as

$$F = (\nu n.(\nu n'.(\{^{\mathrm{ck}(n,n')}/_x\}) \mid A(x,n)) \mathbin{\text{¦}} (\nu n'.(\nu n.(\{^{\mathrm{ck}(n,n')}/_x\}) \mid H(x,n')).$$

Remark 3 Formally, $P \leftrightarrow P_1 * P_2$ is a ternary relation and, for some P_1, P_2, one may have $P \leftrightarrow P_1 * P_2$ and $P' \leftrightarrow P_1 * P_2$ for some non congruent P, P'. Albeit not a composition law, $*$ projects as a composition law on frames: $\phi(P * Q) \equiv \phi(P) \mid \phi(Q)$. Ternary relations also arise in the relational models of BI, or in context logics.

Finally, let us state some useful properties of these new operators.

Lemma 2 *For all P, P_1, P_2, if $P \leftrightarrow P_1 \mathbin{\text{¦}} P_2$ then $\phi(P) \equiv \phi(P_1) \equiv \phi(P_2)$.*

Proof This is an immediate consequence of Lemma 1. □

Lemma 3 *For any process P, the following hold:*

1. $P \leftrightarrow \phi(P) \mathbin{\text{¦}} P$
2. $P \leftrightarrow (P)^p * P$
3. $\forall Q.\, Q \equiv P \Rightarrow P \leftrightarrow (Q)^p * P$
4. $\forall Q.\, P \leftrightarrow \phi(P) \mathbin{\text{¦}} Q \Rightarrow Q \equiv P$
5. $\forall Q, Q'.\, Q' \equiv P \wedge P \leftrightarrow (Q')^p * Q \Rightarrow Q \equiv P$
6. $P \leftrightarrow P_1 * P_2 \wedge \phi(P_1) \equiv \mathbf{0} \Rightarrow \exists Q \equiv P.\, (Q)^p \equiv P_1$

Proof 1. We write $P \equiv \nu\mathbf{n}.(\sigma \mid Q^p)$. Then we have $\phi(P) \equiv \nu\mathbf{n}.(\sigma \mid \mathbf{0})$, and $\nu\mathbf{n}.(\sigma \mid Q^p) \mathbin{\text{¦}} \nu\mathbf{n}.(\sigma \mid \mathbf{0}) = \nu\mathbf{n}.(\sigma \mid Q^p \mid \mathbf{0}) \equiv P$, *i.e.* $P \leftrightarrow \phi(P) \mathbin{\text{¦}} P$.

2. This is a special case of 3.

3. Let $P' \equiv P$. It is sufficient to prove that there are $\mathbf{n}$, ϕ and Q^p such that $P \equiv \nu\mathbf{n}.\phi \mid Q^p$ with $\nu\mathbf{n}.\, Q^p \equiv (P')^p$, which is done by structural induction on P':
   - If P' is a plain process, we can take $\phi = \mathbf{0}$ and $\mathbf{n}$ empty.
   - If P' is an active substitution $\{^M/_x\}$ then we can take $Q = \mathbf{0}$ and $\mathbf{n}$ empty.
   - If $P' = P_1 \mid P_2$, the induction hypothesis gives us $P' \equiv (\nu\mathbf{n}_1.(\phi_1 \mid Q_1)) \mid (\nu\mathbf{n}_2.(\phi_2 \mid Q_2))$ with $\nu\mathbf{n}_i.\, Q_i \equiv (P_i)^p$. We can always assume $\mathbf{n}_1 \cap \mathbf{n}_2 = \emptyset$ up to additional steps of α-conversion, so that $P \equiv \nu\mathbf{n}_1\mathbf{n}_2.((\phi_1 \mid \phi_2) \mid (Q_1 \mid Q_2))$. Since $(P')^p = (P_1)^p \mid (P_2)^p \equiv \nu\mathbf{n}_1\mathbf{n}_2.(Q_1 \mid Q_2)$, we can conclude by taking $\mathbf{n} = \mathbf{n}_1\mathbf{n}_2$, $\phi = \phi_1 \mid \phi_2$ and $Q = Q_1 \mid Q_2$.

– If $P' = \nu n.\, P''$ then the induction hypothesis gives us that $P'' \equiv \nu \mathbf{n}'.\, (\phi' \mid Q')$ with $\nu \mathbf{n}'.\, Q' \equiv (P')^p$. It then follows that $P \equiv \nu n \mathbf{n}'.\, (\phi' \mid Q')$ with $(P')^p = \nu n.\, (P'')^p \equiv \nu n \mathbf{n}'.\, Q'$ which lets us conclude.
– If $P' = \nu x.\, P'$ then the induction hypothesis gives us that $P'' \equiv \nu \mathbf{n}'.\, (\phi' \mid Q')$ with $\nu \mathbf{n}'.\, Q' \equiv (P')^p$ and $x \in dom(\phi')$, from which we can assume without loss of generality that $\phi' = \{^M/_x\} \mid \phi''$ for some term M. It then follows that $P \equiv \nu x \mathbf{n}'.\, (\{^M/_x\} \mid \phi'' \mid Q')$ with $(P')^p = \nu x \mathbf{n}'.\, (\{^M/_x\} \mid Q')$ which allows us to conclude.

4. If $P \leftrightarrow \phi(P) \mid Q$ then there are $\mathbf{n}_1$, $\mathbf{n}_2$, ϕ, P_1^p and P_2^p such that:

$$\begin{aligned} P &\equiv \nu \mathbf{n}_1 \mathbf{n}_2.\, (\phi \mid P_1^p \mid P_2^p) \\ \phi(P) &\equiv \nu \mathbf{n}_1.\, ((\nu \mathbf{n}_2.\, \phi) \mid P_1^p) \\ Q &\equiv \nu \mathbf{n}_2.\, ((\nu \mathbf{n}_1.\, \phi) \mid P_2^p) \end{aligned}$$

Since $\phi(P)$ is a frame, $P_1^p \equiv \mathbf{0}$, so $P \equiv \nu \mathbf{n}_1 \mathbf{n}_2.\, (\phi \mid P_2^p) \equiv Q$.

5. If $P \leftrightarrow (Q')^p * Q$ then there are $\mathbf{n}_1$, $\mathbf{n}_2$, ϕ_1, ϕ_2 and P' such that:

$$\begin{aligned} P &\equiv \nu \mathbf{n}_1 \mathbf{n}_2.\, (\phi_1 \mid \phi_2 \mid P') \\ (Q')^p &\equiv \nu \mathbf{n}_1.\, (\phi_1 \mid (\nu \mathbf{n}_2.\, P')) \\ Q &\equiv \nu \mathbf{n}_2.\, (\phi_2 \mid (\nu \mathbf{n}_1.\, P')) \end{aligned}$$

Since $(Q')^p$ is a plain process, $\phi_1 \equiv \mathbf{0}$, so $P \equiv \nu \mathbf{n}_1 \mathbf{n}_2.\, (\phi_2 \mid P') \equiv Q$.

6. Assume $P \leftrightarrow P_1 * P_2 \wedge \phi(P_1) = \emptyset$. There are $Q \equiv P$ and $\mathbf{n}_1$, $\mathbf{n}_2$, Q'^p, ϕ_1, ϕ_2 such that $\nu \mathbf{n}_1.\, (\phi_1 \mid \nu \mathbf{n}_2.\, Q'^p) \equiv P_1$, $\nu \mathbf{n}_2.\, (\phi_2 \mid \nu \mathbf{n}_1.\, Q'^p) \equiv P_2$ and $Q = \nu \mathbf{n}_1 \mathbf{n}_2.\, (\phi_1 \mid \phi_2 \mid Q'^p)$. As $\phi(P_1) \equiv \mathbf{0}$, $\phi_1 \equiv \mathbf{0}$ so $(Q)^p = \nu \mathbf{n}_1 \mathbf{n}_2.\, Q'^p \equiv \nu \mathbf{n}_1.\, (\phi_1 \mid \nu \mathbf{n}_2.\, Q'^p) \equiv P_1$, which allows us to conclude. □

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 $\mathcal{V}$ and $\mathcal{N}$, ranged over with $t, t', \ldots$, and we write U, V for terms that may mention these term variables. We call $A\pi L$ the set of formulae defined by the following grammar:

$$\begin{aligned} A, B ::= {} & U = V \mid \neg A \mid A \wedge B \mid \Diamond A \mid \exists t.\, A \mid \mathit{\Nu} u.\, A \mid \mathsf{H} u.\, A \\ & \mid A \oslash u \mid \copyright u \mid \mathbf{0} \mid A \mid B \mid A \rhd B \mid \emptyset \mid A * B \mid A \mathrel{-\!\!*} B \end{aligned}$$

$U = V$ is the equality of terms w.r.t. the current frame, negation and conjunction are classical, and $\Diamond A$ is the strong reduction modality. $\exists t.\, A$ is term quantification and $\mathit{\Nu} n.\, A$ and $\mathit{\Nu} x.\, A$ are respectively the fresh name and fresh variable quantifications. We use the same operator for both of these, as well as for the H, $\copyright$ and $\oslash$ operators, because our convention on namings allows us to do so unambiguously. $\mathsf{H} u.\, A$ is the hidden name or variable quantification; $\copyright u$ means that u appears free in the process; $A \oslash u$ is hiding of name or variable u. $\mathbf{0}$ (resp. $\emptyset$) denotes the null plain process (resp. the empty frame) and $A \mid B$ (reps. $A * B$) plain process (resp. frame) composition. $A \rhd B$ (resp. $A \mathrel{-\!\!*} B$) is a guarantee operator and the adjunct of $A \mid B$ (resp. $A * B$).

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

Fig. 1 Satisfaction relation

The operators' semantics, close to the one defined by Caires and Cardelli for the π-calculus [8], is given by a satisfaction relation described in Figure 1 whose judgments are of the form $P, v \vDash A$ between a process P, a spatial formula A and a valuation v. Valuations assign terms $\mathbf{M}$ to all the free term variables $\mathbf{t}$ of the formula and are written $\{\mathbf{t} \to \mathbf{M}\}$ when $|\mathbf{t}| = |\mathbf{M}| = n$ and $v(t_i) = M_i$ 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, judgments are written $P \vDash A$.

Boolean operators are assumed to bind more tightly than compositions and adjunctions, which in turn bind more tightly than every other operator. Derived connectives $\forall t$, $\vee$, $\Leftrightarrow$ and $U \neq V$ are defined as usual, and so are the sets of free names and free variables of a formula A, written $fn(A)$ and $fv(A)$.

Satisfaction of a formula A by a process P and a valuation v should remain unchanged should some names or variables be swapped in A, P and v, as expressed by the following lemma. Swapping of two names or variables is defined in the obvious way and needs not to care about capture avoidance; it is written for example $P[m \leftrightarrow n]$ for the swapping of m and n in P.

Lemma 4 *For all P, v and A and names or variables u, u', the following holds:*

$$P, v \vDash A \text{ iff } P[u \leftrightarrow u'], v[u \leftrightarrow u'] \vDash A[u \leftrightarrow u'].$$

Proof The proof is done by induction on the formula A. It is enough to prove one implication because the swapping operation is an involution.

– $U = V$:

$$\begin{aligned}
&P, v \vDash U = V\\
&\Rightarrow P \vdash Uv = Vv\\
&\Rightarrow P[u \leftrightarrow u'] \vdash Uv[u \leftrightarrow u'] = Vv[u \leftrightarrow u'] \quad \text{By invariance of } \mathcal{E}\\
&\Rightarrow P[u \leftrightarrow u'], v[u \leftrightarrow u'] \vDash (U = V)[u \leftrightarrow u'] \quad \text{Because } u, u' \notin dom(v)
\end{aligned}$$

– $\neg A$, $A_1 \wedge A_2$, $\Diamond A$, $\exists t.\, A$: straightforward.
– $\mathcal{N}w.\, A$:

$$\begin{aligned}
&P, v \vDash \mathcal{N}w.\, A\\
&\Rightarrow \exists w' \notin fnv(P, v, A).\, P, v \vDash A[w \leftarrow w']\\
&\Rightarrow \exists w' \notin fnv(P, v, A).\, P[u \leftrightarrow u'],\\
&\quad v[u \leftrightarrow u'] \vDash A[w \leftarrow w'][u \leftrightarrow u']\\
&\qquad \textit{By induction hypothesis}
\end{aligned}$$

The only case that requires attention is when $w' = u'$. Then $u' \notin fnv(P, v, A)$ so $u \notin fnv(P[u \leftrightarrow u'], v[u \leftrightarrow u'], A[u \leftrightarrow u'])$ and $A[w \leftarrow w'][u \leftrightarrow u'] = A[u \leftrightarrow u'][w \leftarrow u]$. This leads us to:

$$\begin{aligned}
&P, v \vDash \mathcal{N}w.\, A\\
&\Rightarrow P[u \leftrightarrow u'], v[u \leftrightarrow u'] \vDash A[u \leftrightarrow u'][w \leftarrow u]\\
&\Rightarrow \exists w' \notin fnv(P, v, A).\, P[u \leftrightarrow u'], v[u \leftrightarrow u']\\
&\quad \vDash A[u \leftrightarrow u'][w \leftarrow w'] \quad \textit{By taking} w' = u\\
&\Rightarrow P[u \leftrightarrow u'], v[u \leftrightarrow u'] \vDash (\mathcal{N}w.\, A)[u \leftrightarrow u']
\end{aligned}$$

– $\mathsf{H}w.\, A$: similar to the previous case.
– The remaining cases are straightforward. □

Let us now comment on the choice of the logical operators we consider in our spatial logic with respect to other possible choices adopted in the literature.

Variable revelation We introduce a revelation operator for restricted variables. Hidden name revelation was sometimes considered unpleasant in early days of spatial logics, and hidden variable revelation can be considered unpleasant as well, with even more reasons, as hidden variables "do not exist properly", in the sense that any process is congruent to a process without restricted variables. Moreover, this operator will not be essential in the characterization of static equivalence. The reason for considering it is its help in the characterization of logical equivalence of the full logic: so as to express in the logic that a process sends a message M, the only solution we found is to first "reveal" a variable x and thus an active substitution $\{^{M}/_{x}\}$, and then express that the process sends x. This mimics the reduction steps of such a process, as detailed in Remark 1. The same trick is used to characterize branching tests. Despite our efforts, we do not have a clear representation of logical equivalence without hidden variable revelation. On the other hand, variable revelation is still not expressive enough to characterize structural congruence when the equational theory over terms is not the one of finite trees. Thus, there might be other choices than ours that would yield a more satisfactory framework, although we do not have any particular idea about what they could be.

The case for multiple spatial conjunctions; absence of fixed-points We introduce two distinct spatial conjunctions, one of the motivations mentioned earlier being that we may split processes even if they share private names. Another approach for solving this issue could be to keep only one spatial conjunction $|$, as in other spatial logics, and rely on operators that reveal arbitrarily many restricted names, such as the operator H^* introduced by Acciai and Boreale [3], or to include fixed point constructs in the logic, thus allowing to define H^*A as $\mu X.A \vee Hn.X$, with $n \notin fn(A)$. Fixed points complicate a bit the characterization of logical equivalence in terms of Ehrenfeucht-Fraïssé games, and thus in terms of intensional bisimilarity. Moreover, their use seemed rather limited to us, except for this specific purpose. Hence we do not consider them in our setting, nor do we consider the H^*A construct.

However, one could wonder about the impact that our choice of having two spatial conjunctions has on the characterization of the logical equivalences. The alternative would be to have a single spatial conjunction $|$, a null process predicate $\mathbf{0}$ and a single adjunct $\triangleright$, that would account for both the plain part and the extended part. We claim that most of the results presented in this paper would carry on to this setting, with very little modifications. In particular, this choice is not essential for the characterization of the logical equivalence of the full logic. However, and as stressed in Section 2.5, distinguishing $*$ and $-\!\!*$ from the usual $|$ and $\triangleright$ yields a logical characterization of static equivalence for *all* processes, and not processes restricted to frames, as highlighted by Lemmas 6 and 7. These lemmas would not hold with a single spatial conjunction connectives, because a static fragment that would contain one of $\mathbf{0}$, $|$ or $\triangleright$ would also constrain the plain part. Choosing to have distinct operators to treat the frame and the plain part of processes thus allows us to have a more elegant presentation in that respect.

3.2 Derived formulae

The formulae below will be useful in the following sections:

$$\top \triangleq \mathbf{0} \vee \neg\mathbf{0} \qquad \bot \triangleq \neg\top \qquad A[B] \triangleq (A \wedge \mathbf{0}) \mid ((B \wedge \emptyset) * \top)$$
$$A \blacktriangleright B \triangleq \neg(A \triangleright \neg B) \qquad A \mathrel{-\!\!\circledast} B \triangleq \neg(A \mathrel{-\!\!*} \neg B)$$
$$\mathbf{1} \triangleq \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0}) \qquad \mathbb{I} \triangleq \neg\emptyset \wedge \neg(\neg\emptyset * \neg\emptyset)$$
$$\mathsf{public} \triangleq \neg\mathsf{H}n.\, \copyright n \qquad \mathsf{single} \triangleq \mathbf{1} \wedge \emptyset \wedge \mathsf{public}$$

Proposition 1 *The formulae above are verified by a process P iff there exists a process $Q \equiv P$ such that:*

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

Proof

1. $\top$ and $\bot$: trivial;
2. If $P \vDash A[B]$ then, from the formula, there are P_1 and P_2 such that $P \leftrightarrow P_1 \upharpoonright P_2$, $P_1 \vDash A \wedge \mathbf{0}$ and $P_2 \vDash (B \wedge \emptyset) * \top$. Lemma 2 gives us that $P_1 \equiv \phi(P)$, hence $\phi(P) \vDash A$. As $P_1 \equiv \phi(P)$, Lemma 3(4) applies and $P_2 \equiv P$, so there exist P_1', P_2' such that $P \leftrightarrow P_1' * P_2'$ and $P_1' \vDash B \wedge \emptyset$. Since the frame of P_1' is empty, Lemma 3(6) applies, so there is $Q \equiv P$ such that $(Q)^p \equiv P_1'$, hence $(Q)^p \vDash B$. Moreover, $\phi(Q) \equiv \phi(P)$ (Lemma 1), so $\phi(Q) \vDash A$.
   Conversely, let Q be such that $Q \equiv P$, $\phi(Q) \vDash A$ and $(Q)^p \vDash B$. Lemma 3(2) entails $Q \leftrightarrow (Q)^p * Q$ so $Q \vDash (B \wedge \emptyset) * \top$ and Lemma 3(1) entails $Q \leftrightarrow \phi(Q) \upharpoonright Q$ so $Q \vDash (A \wedge \mathbf{0}) \upharpoonright ((B \wedge \emptyset) * \top)$.
3. 4. and 5. Straightforward.
6. Straightforward case analysis. □

It is also worth mentioning that the fresh quantifier and the "empty frame" predicates can be expressed using the other operators of the logic.

Proposition 2 *The following logical equivalences hold:*

1. $\mathcal{V}n. A \Leftrightarrow \mathsf{H}n. A \wedge \neg\copyright n$
2. $\mathcal{V}x. A \Leftrightarrow \mathsf{H}x. (\mathbb{I} \wedge \mathsf{public}) * (A \wedge \neg\copyright x)$
3. $\emptyset \Leftrightarrow \top \triangleright ((\neg\Diamond\top) \mathrel{-\!\!*} \neg\Diamond\top)$

Proof

1. This is straightforward since the conditions $P \equiv \nu n. P'$ and $P' \vDash \neg\copyright n$ are equivalent to $P \equiv P'$. This uses (but is independent from) Lemma 5 to conclude.
2. The elimination of the fresh variable quantifier requires a bit more care, as variable revelation creates a new active substitution. We also have to make sure that this substitution cannot have had affected the rest of the process. The only possibility for that, as the variable must be fresh, is for it to have replaced the term of the substitution by the revealed variable in the plain part of the process, thus exposing an occurrence of the variable that would contradict $\neg\copyright x$.
3. The left-to-right direction is easy. For the converse, let P be a process whose frame is not empty. Then there is $x \in dom(P)$. Let us now consider a frame ϕ closing P, the process $Q = \mathit{if}\ x = y\ \mathit{then}\ \mathbf{0}$ for $y \notin fv(\phi, P)$, and a closed term M. Then $Q \mid \phi \mid \{^M/_y\} \vDash \neg\Diamond\top$ as the test in Q is still open in x, but $P \mid Q \mid \phi \mid \{^M/_y\} \nvDash \neg\Diamond\top$ as $P \mid \phi \mid \{^M/_y\}$ can "close" x and y and therefore force Q to reduce to its "else" branch (or its "then" branch in case the test is rendered true). □

Finally, as do the corresponding operators of Caires and Cardelli, the fresh and the hidden quantifiers have the Gabbay-Pitts property [13]:

Proposition 3 *For all name or variable* u*, formula* A*, process* P *and valuation* v*, the following propositions are equivalent:*

1. $P, v \vDash \mathcal{V}u. A$
2. $\exists u' \notin fnv(P, v, A). P, v \vDash A[u \leftarrow u']$
3. $\forall u' \notin fnv(P, v, A). P, v \vDash A[u \leftarrow u']$
4. $P, v \vDash \neg\mathcal{V}u. \neg A$

Proof

- (1) $\Leftrightarrow$ (2): By definition.
- (2) $\Rightarrow$ (3): If $P, v \vDash A[u \leftarrow u']$ for $u' \notin fnv(P, v, A)$ then, for any $u'' \notin fnv(P, v, A)$ Lemma 4 gives us $P[u' \leftrightarrow u''], v[u' \leftrightarrow u''] \vDash A[u \leftarrow u'][u' \leftrightarrow u'']$. As neither u' nor u'' appear free in P, v or A, we can conclude that $P, v \vDash A[u \leftarrow u'']$.
- (2) $\Leftarrow$ (3): Trivial.
- (3) $\Leftrightarrow$ (4): By definition. □

Proposition 4 *For all name or variable* u*, formula* A*, process* P *and valuation* v*, the following propositions are equivalent:*

1. $P, v \vDash \mathsf{H}u. A$
2. $\exists u' \notin fnv(P, v, A). \exists P'. P \equiv \nu u'. P'$ *and* $P', v \vDash A[u \leftarrow u']$
3. $\forall u' \notin fnv(P, v, A). \exists P'. P \equiv \nu u'. P'$ *and* $P', v \vDash A[u \leftarrow u']$

Proof Similar to the proof of Proposition 3. □

3.3 Cryptographic examples

We now propose, on a very basic example, some possible avenues for using the spatial logic to express cryptographic properties. As usual, we interpret the frame as the history of past communications: restricted names are nonces or secrets, and each active substitution holds the content of an emitted message. Recall the frame $\nu n, s. \Phi$ of the introduction, modeling a situation where an encrypted secret s had been transmitted using a published public key $\mathrm{pk}(n)$ (we assume here the equational theory axiom $\mathrm{dec}(\mathrm{enc}(x, \mathrm{pk}(y)), \mathrm{sk}(y)) = x$) and consider the frame $\nu n, s. \Phi \mid \phi = \nu n, s. \{^{\mathrm{enc}(s,y)}/_x\} \mid \{^{\mathrm{pk}(n)}/_y\} \mid \phi$ for an extra frame ϕ.

Following the definition of the applied π-calculus, we will say that the secret s is deducible from this frame if the formula $\mathsf{leak} \triangleq \exists t.\, x = \mathrm{enc}(t, y)$ holds; for instance, choosing $\phi = \{^{\mathrm{sk}(n)}/_{z}\}$ as an extra frame would yield such a leak, with $t = \mathrm{dec}(x, z)$ as witness. The formula $\exists t.\, \forall t'.\, \mathsf{H}n.\, (t = \mathrm{pk}(n) \wedge t' \neq \mathrm{sk}(n))$ asserts that the published key is indeed public, and that its associated private key is secret. An emitted message M represented by an active substitution $\{^{M}/_{z}\}$ in ϕ is part of the cause of a leak if the formula $(\neg\mathsf{leak}) \oslash z \wedge \mathsf{leak}$ holds.

One could also express static properties about authenticated sessions: in a protocol where each user is assigned a session identifier (here, a secret name) used in every subsequent communication, one may count the number of opened sessions with $*$-conjunctions of the $\mathbb{I}$ formula. Indeed, if every other nonce used by the protocol within a session is generated from the session identifier, each subframe verifying $\mathbb{I}$ will correspond to a different session.

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: the formula $\mathsf{Client} \nmid \mathsf{Server}$ would describe a protocol with a client and a server, and $\mathsf{Client} \triangleright \mathsf{Attack}$ would describe a server that might be attacked by a context that follows the specification of a genuine client.

As mentioned earlier, our initial motivation was also to provide a logic that may express some complex forms of contextual reasoning. Let us illustrate these ideas on the electronic vote protocol and the coercion-resistance property, following the definitions of Kremer et al. [12]. The vote protocol involves three kinds of agents: the organizers (administrator, key managers, and vote collectors), the voters, and the coercers. Voters share some keys with organizers that are unknown to the coercer. The coercion-resistance property is a property of the organizers protocol that says that whatever the coercer will force the voter to send (interactively), she can never be sure that the voter voted in a certain way. Receipt-freeness is defined likewise, except that the coercer is passive, and can be reduced to a static equivalence check. On the contrary, coercion-resistance cannot be reduced to a bisimilarity check, because here the voter plays against the coercer. Our expectation is that one may express coercion resistance as a check of the form

$$\mathsf{Administrator} \models \mathsf{Coercer}(\mathsf{a}) \triangleright (\mathsf{CoercedVoter} \blacktriangleright \mathsf{Claim}(a) \wedge \mathsf{Vote}(b)).$$

In this formula, the adjuncts may express the complex form of contextual reasoning that causes standard bisimulation technique to fail. Formula $\mathsf{Coercer}(a)$ expresses that the process is a coercer who wants a voter to vote a, $\mathsf{CoercedVoter}$ expresses that the process is a coerced voter, following both the vote protocol and the coercer protocol, $\mathsf{Claim}(a)$ expresses that the coercer-voter protocol indeed succeeds, and $\mathsf{Vote}(b)$ expresses that the administrator-voter protocol succeeds and that the voter voted b. This formula can be read in English as "whatever the coercer asks the voter to do, the voter has a strategy to satisfy both the coercer and the administrator while preserving his vote secret."

Finally, an important concept in cryptography in general and applied π-calculus in particular is deducibility of terms [1], and it can be expressed very naturally inside our logic. A term M is said to be deducible in a frame $\nu\mathbf{n}.\, \sigma$ which is written $\nu\mathbf{n}.\, \sigma \vdash M$ (the scope of the restriction extends to M: the notation should be read as $\nu\mathbf{n}.\, (\sigma \vdash M)$), when there exists a term N such that $fn(N) \cap \mathbf{n} = \emptyset$ and $N\sigma = M$. This can be expressed in our logic by the following formula, where $\mathbf{s}$ are the secret names occurring in M:

$$\exists t.\, \mathbf{s} \circledR\, t = M$$

As the term quantification is placed first, the guessed term cannot contain the revealed names. Unfortunately, some care is required when revealing the names $\mathbf{n}$: because of the α-conversions that may occur, the names we reveal are not necessarily the ones we intended to reveal. For instance, consider $F = \nu s, s'.\, \{^{\mathrm{pk}(s)}/_{x}\}\{^{\mathrm{pk}(s')}/_{y}\}\{^{s'}/_{z}\}$ and the question "is s deducible from F?" Then while the answer is "no", the formula $\exists t.\, \mathbf{s} \circledR\, t = s$ is true, because we can chose to swap s and s' by α-conversion and then take $t = z = s$ as witness.

To overcome this, we have to provide additional constraints on the names we reveal. If there is only one secret name in the frame, nothing is required. If the frame is F and we want to know whether s is deducible, we can use the formula $\exists t.\, s \circledR\, (t = s \wedge \mathrm{pk}(s) = x)$ which ensures that we are talking about the same s than before. A general recipe for solving the issue is to describe the whole frame $\nu\mathbf{n}.\, \{^{\mathbf{M}}/_{\mathbf{x}}\}$ within the formula:

$$\exists t.\, \mathbf{n} \circledR\, (t = M \wedge x_1 = M_1 \wedge \ldots \wedge x_n = M_n)$$

3.4 Fragments

We define two usual fragments of our logic: the extensional one, which allows one to observe a process via its interactions with some (possibly constrained) environment, and the intensional one, which allows one to explore the very structure of the process. We also distinguish between static operators, which only account for the frame, and dynamic ones, which account for the whole process. The four fragments are summed up by the table below, that defines which operators the formulae of each fragments may be composed of. The intensional fragment contains the extensional one and the dynamic one contains the static one; the static extensional fragment is thus common to all fragments, and the dynamic intensional one coincides with the whole logic.

	Dynamic	
	Static	
Extensional	$=, \neg, \wedge, \exists, \mathcal{N}, \multimap, \oslash$	$\Diamond, \triangleright$
Intensional	$\mathsf{H}, *, \emptyset$	$©, \mathbf{0}, \mathfrak{f}$

We write $A \pi L$ for the set of all formulae, L^{stat} for the static fragment, and $L^{\text{stat}}_{\text{ext}}$ for the static extensional fragment.

3.5 Logical equivalences and other process equivalences

The next two sections are devoted to the characterization of the expressiveness of every logical fragment introduced so far, and in particular to that of the logical equivalences they induce. A first step in that direction is to notice that the semantics of the formulae is preserved by structural congruence, by projection over the frame for static formulae, and by static equivalence for static extensional formulae. We finish this section by proving these properties.

Lemma 5 *$P \vDash A$ and $P \equiv Q$ implies $Q \vDash A$.*

Proof We prove this by induction on the structure of the formula.

- $U = V$: As structural congruence is included in static equivalence [2], $P \approx_s Q$, so $Q, v \vDash U = V$.
- $\mathcal{N}u.\, A$: If $P, v \vDash \mathcal{N}u.\, A$ then, by definition and Lemma 3, $\forall u' \notin fnv(P, v, A).\ P, v \vDash A[u \leftarrow u']$. In particular, we can pick $u'' \notin fnv(P, v, A) \cup fnv(Q)$ such that $P, v \vDash A[u \leftarrow u'']$. By induction hypothesis, $Q, v \vDash A[u \leftarrow u'']$, so $\exists u' \notin fnv(Q, v, A).\ Q, v \vDash A[u \leftarrow u']$, *i.e.* $Q, v \vDash \mathcal{N}u.\, A$.
- $\mathsf{H}u.\, A$:

$$\begin{aligned} P, v \vDash \mathsf{H}u.\, A &\iff \exists u' \notin fnv(P, v, A).\, \exists P'.\ P \equiv \nu u'.\, P' \\ &\qquad \text{and } P', v \vDash A[u \leftarrow u'] \\ &\iff \exists u' \notin fnv(P, v, A).\, \exists P'.\ Q \equiv \nu u'.\, P' \\ &\qquad \text{and } P', v \vDash A[u \leftarrow u'] \\ &\iff \exists u' \notin fnv(Q, v, A).\, \exists P'.\ Q \equiv \nu u'.\, P' \\ &\qquad \text{and } P', v \vDash A[u \leftarrow u'] \\ &\iff Q, v \vDash \mathsf{H}u.\, A \end{aligned}$$

The second equivalence uses $P \equiv Q$ and the third one uses Lemma 4.

All other cases are straightforward. □

Lemma 6 *For every formula A of L^{stat}, $P \vDash A$ iff $\phi(P) \vDash A$.*

Proof The proof is done by induction on the formula. All the cases are straightforward, except for frame composition and adjunct. Let us detail the case of $*$: $P \vDash A_1 * A_2$ if and only if there are P_1, P_2 such that $P \leftrightarrow P_1 * P_2$ and $P_i \vDash A_i$. By Lemma 5, we may assume $P \equiv P_1 * P_2$. By induction hypothesis, this is true if and only if $\phi(P_i) \vDash A_i$ and by definition of the frame composition, $\phi(P) \equiv \phi(P_1) \mid \phi(P_2)$ which lets us conclude. □

Lemma 7 *For every formula A of L^{stat}_{ext}, $\phi \approx^s_s \psi$ and $\phi \vDash A$ implies $\psi \vDash A$.*

Proof Let us prove by induction on the formula A that for all frames $\phi \approx^s_s \psi$ and all valuations v,

$\phi, v \vDash A$ implies $\psi, v \vDash A$.

- The cases of $\emptyset, \exists t.\, A, \neg A$ and $A_1 \wedge A_2$ are straightforward.
- If $\phi, v \vDash U = V$ then $\phi \vdash Uv = Vv$ so that by definition of static equivalence $\psi \vdash Uv = Vv$ and thus $\psi, v \vDash U = V$.
- If $\phi, v \vDash A \oslash x$, then $x \in dom(\phi)$ and $(\nu x.\, \phi), v \vDash A$. Since $\phi \approx^s_s \psi$, $x \in dom(\psi)$ and $\nu x.\, \phi \approx^s_s \nu x.\, \psi$ so by induction hypothesis $(\nu x.\, \psi), v \vDash A$, yielding $\psi, v \vDash A \oslash x$. The case for $A \oslash n$ is similar.
- If $\phi, v \vDash A \multimap B$ then for all ϕ', ϕ'', $\phi'' \leftrightarrow \phi * \phi'$ and $\phi', v \vDash A$ implies $\phi'', v \vDash B$. Since the plain parts are all null, this is equivalent to $\phi'' \equiv \phi \mid \phi'$, and thus the induction hypothesis applies for $\psi \mid \phi' \approx^s_s \phi \mid \phi'$, yielding $(\psi \mid \phi'), v \vDash B$ and then $\psi, v \vDash A \multimap B$. □

4 Spatial logic applied to frames

In this section, we establish that logical equivalences induced by the static fragments match static equivalences that have originally been proposed for the applied π-calculus: structural congruence for frames for the intensional fragment, and static equivalence for the extensional fragment.

4.1 Intensional characterization

The formula $\mathsf{Subst}(x = M)$ below characterizes processes of the form $\{^M/_x\}$, for a given x and a given M. The other two will be useful for our quantifier elimination procedure in Sect. 6.

$$\begin{aligned} \mathsf{public\_frame} &\triangleq \neg(\top * (\mathbb{I} \wedge \mathsf{H}x.\, \mathbb{I})) \\ \mathsf{Subst}(x) &\triangleq \mathsf{public\_frame} \wedge (\emptyset \oslash x) \\ \mathsf{Subst}(x = M) &\triangleq \emptyset \oslash x \wedge x = M \end{aligned}$$

Lemma 8 *For each process P, variable x and term M such that $\mathcal{E} \nvdash x = M$, we have:*

- $P \vDash \mathsf{public\_frame}$ *iff* $\phi(P) \equiv \{^{\mathbf{M}}/_{\mathbf{x}}\}$ *for some variables* $\mathbf{x}$ *and terms* $\mathbf{M}$*;*
- $P \vDash \mathsf{Subst}(x)$ *iff* $\phi(P) \equiv \{^N/_x\}$ *for some term* N*;*
- $P \vDash \mathsf{Subst}(x = M)$ *iff* $\phi(P) \equiv \{^M/_x\}$.

Proof Observe first that if $P \vDash \mathbb{I} \wedge \mathsf{H}x.\,\mathbb{I}$ then $P \vDash \mathbb{I}$, so either $\phi(P)$ is a single public active substitution or $\phi(P) \equiv \nu\mathbf{n}.\ \sigma$ where σ cannot be split into σ_1 and σ_2 that do not share names in $\mathbf{n}$. So proving the formula amounts to prove that a frame of the first form does not satisfy $\mathsf{H}x.\,\mathbb{I}$, whereas a frame of the latter form does.

Assume first that P is of the form $\{^M/_x\}$, and let us show that $P \nvDash \mathsf{H}x.\,\mathbb{I}$. Let P' be any arbitrary process such that $P \equiv \nu y.\ P'$. Then $P' \equiv \nu\mathbf{n}.\ (\{^{M'}/_x\} \mid \{^N/_y\})$ with $M'[y \leftarrow N] = M$, thus $P' \equiv \{^M/_x\} \mid \nu\mathbf{n}.\ \{^N/_y\}$, which shows that it does not satisfy $\mathbb{I}$.

Assume now that $\phi(P)$ is not public. Then there is n such that $\phi \equiv (\nu n.\ \phi_1)$, $n \in \overline{fn}(\phi_1)$ and $\nu n.\ \phi_1 \vDash \mathbb{I}$. Then, for $x \notin dom(P)$, we have $\nu n.\ \phi_1 \equiv \nu x, n.\ (\phi_1 \mid \{^n/_x\})$ so $\nu n.\ \phi_1 \vDash \mathsf{H}x.\,\mathbb{I}$. This illustrates one peculiar behavior of variable revelation: it may reveal a substitution under the scope of an arbitrary number of name restrictions.

The other two formulae are straightforward. Observe that the hide operator is used in conjunction with the $\emptyset$ predicate to state both $x \in dom(P)$ and $dom(P) \subseteq \{x\}$. □

Once this basic block is defined, one can easily build up a formula capturing processes in a certain structural congruence class, as expressed by the following theorem:

Theorem 1 (Characteristic formulae for frames) *For all frames ϕ there exists a formula F_ϕ in L^{stat} such that for all extended processes P, $P \vDash \mathsf{F}_\phi$ if and only if $\phi(P) \equiv \phi$.*

Proof One can build a frame $\phi' \equiv \phi$ constructed using frame compositions instead of parallel compositions. A characteristic formula for ϕ' can then be built up inductively in a straightforward manner, using the logical frame composition and name and variable revelations. □

For example, a characteristic formula for our running example's frame $\nu n, s.\ \Phi$ is

$$\mathsf{F}_{\nu n,s.\,\Phi} = (\mathsf{H}s.\,\mathsf{Subst}(x = \mathrm{enc}(s, y))) * (\mathsf{H}n.\,\mathsf{Subst}(y = \mathrm{pk}(n))).$$

Together with Lemma 5, this theorem gives a precise definition of logical equivalence induced by L^{stat} on frames:

Corollary 1 (Logical equivalence in L^{stat}) *For all extended processes P and Q, P and Q satisfy the same formulae of L^{stat} if and only if $\phi(P) \equiv \phi(Q)$.*

4.2 Extensional characterization

We will show in this section that logical equivalence for $L^{\mathrm{stat}}_{\mathrm{ext}}$, or extensional equivalence, coincides with strong static equivalence and that, given a closed frame ϕ, one can construct a formula $\mathsf{F}_\phi^{\approx_s}$ characterizing the equivalence class of ϕ. The right-to-left inclusion is given by Lemma 7. For its converse, let us first remark that one can characterize frames whose domains are $\mathbf{x}$ using the formula $\emptyset \oslash \mathbf{x}$. Then, one can define a characteristic formula $\mathsf{F}_\sigma^{\approx_s}$ for a public frame $\sigma = \{^{\mathbf{M}}/_{\mathbf{x}}\}$ of size n:

$$\mathsf{F}_\sigma^{\approx_s} \triangleq \emptyset \oslash \mathbf{x} \wedge \bigwedge_{i=1}^{n} x_i = M_i.$$

Let ϕ be a closed frame $\nu\mathbf{n}.\,\sigma$ with σ a public frame, and consider the formula

$$(\nu\mathbf{n}.\,\sigma)\ \mathsf{forces}\ U = V \triangleq \mathsf{F}_\sigma^{\approx_s} \mathrel{-\!\!*} \big((U = V) \oslash \mathbf{n}\big).$$

Then $\emptyset, v \vDash \phi\ \mathsf{forces}\ U = V$ if and only if $\phi, v \vDash U = V$. Moreover, one can internalize an assumption $\emptyset \vDash A$ in the logic: a process P satisfies $(\emptyset \wedge \neg A) \mathrel{-\!\!*} \bot$ if and only if $\emptyset \vDash A$. We may then derive characteristic formulae for static equivalence on closed frames:

$$\mathsf{F}_\phi^{\approx_s} \triangleq \emptyset \oslash \mathbf{x} \wedge \forall t, t'.\ \big((\emptyset \wedge \neg\phi\ \mathsf{forces}\ t = t') \mathrel{-\!\!*} \bot\big) \Leftrightarrow t = t'.$$

Theorem 2 (Formulae for static equivalence) *For all closed frames ϕ, ψ, $\psi \vDash \mathsf{F}_\phi^{\approx_s}$ if and only if $\phi \approx_s \psi$.*

Proof See discussion above. □

Using this theorem and Lemma 7, one concludes that two closed frames ϕ and ψ satisfy the same formulae of $L^{\mathrm{stat}}_{\mathrm{ext}}$ if and only if $\phi \approx_s \psi$, and this extends to the general case by using strong static equivalence:

Theorem 3 (Logical equivalence in $L^{\mathrm{stat}}_{\mathrm{ext}}$) *For all extended processes P and Q, P and Q satisfy the same formulae of L^{stat}_{ext} if and only if $P \approx_s^s Q$, and characteristic formulae for these equivalence classes are derivable.*

Proof Let us consider two non-closed, logically equivalent frames ϕ_1 and ϕ_2. Then, for any closing evaluation context $C \equiv \nu\mathbf{n}.\,(\cdot \mid \sigma)$, they should both satisfy the formula $\mathsf{F}_\sigma \mathrel{-\!\!*} \mathsf{F}_{C[\phi_1]}^{\approx_s} \oslash \mathbf{n}$. Thus, for all closing evaluation contexts C, $C[\phi_2] \vDash \mathsf{F}_{C[\phi_1]}^{\approx_s}$ so $C[\phi_2] \approx_s C[\phi_1]$. This shows $\phi_1 \approx_s^s \phi_2$, so logical equivalence on frames is indeed strong static equivalence on frames (and thus on processes, by Lemma 6). □

5 Logical characterization of processes

In this section we study the logical equivalence induced by the dynamic intensional fragment. More precisely, we write $P =_L Q$ if P and Q cannot be told apart by a $A\pi L$ formula, and look for a better understanding of $=_L$. We first introduce a notion of intensional bisimulation, and prove it sound with respect to $=_L$. We then establish that $=_L$ is not a congruence, which reinforces our choice of adding rules for adjuncts in the definition of intensional bisimulation. We then introduce a notion of relaxed structural congruence $\equiv'$, and show that

it is an intensional bisimulation. Finally, we establish several completeness results using characteristic formulae, either by restricting the equational theory, or by restricting the set of considered processes, which altogether entails that $=_L$ is $\equiv'$ in these particular cases.

5.1 Intensional bisimilarity and its soundness

We introduce a notion of intensional bisimulation that aims at characterizing $=_L$ by an Ehrenfeucht-Fraïssé game.

Definition 10 (Intensional bisimulation) A relation $\mathcal{R}$ is an intensional bisimulation if $\mathcal{R}$ is symmetric and the following assertions hold 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' such that $P \equiv \nu u.\, P'$ then there is Q' such that $Q \equiv \nu u.\, Q'$ and $P' \mathcal{R} Q'$;
5. for $\dagger \in \{*, |\}$, for all P_1, P_2, if $P \leftrightarrow P_1 \dagger P_2$, then there are Q_1, Q_2 such that $Q \leftrightarrow Q_1 \dagger Q_2$ and $P_i \mathcal{R} Q_i$;
6. for $\dagger \in \{*, |\}$, for all P_1, P', if $P_1 \leftrightarrow P \dagger P'$, then there are Q_1, Q' such that $Q_1 \leftrightarrow Q \dagger Q'$, $P' \mathcal{R} Q'$ and $P_1 \mathcal{R} Q_1$;
7. if there is P' such that $P \rightarrow P'$ then there is Q' such that $Q \rightarrow Q'$ and $P' \mathcal{R} Q'$;
8. $\nu u.\, P \mathcal{R} \nu u.\, Q$.

Let us stress the fact that no equivalents of conditions 6 and 8 occur in the original intensional bisimulation [27]. Fortunately, intensional bisimilarity was a congruence in that case and as a consequence, conditions 6 and 8 were admissible. Note moreover that conditions 6 and 8 do not entail that $\mathcal{R}$ is a congruence (even with $\dagger = |$).

Proposition 5 (Soundness) *Let $\mathcal{R}$ be an intensional bisimulation. 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 $(P, Q) \in \mathcal{R}$ then $Q, v \vDash A$.

- $A_1 \wedge A_2$: immediate by induction.
- $\neg A$: suppose $Q \vDash A$, then by induction (and symmetry of $\mathcal{R}$) $P \vDash A$: contradiction.
- $\emptyset$ or $U = V$: immediate by condition 1.
- $\mathbf{0}$: immediate by condition 2.
- $\copyright u$: immediate by condition 3.
- $\exists t.A$: then there is v' such that $P, v' \vDash A$ and the induction applies for A.
- $\Diamond A$: immediate by condition 7 and induction.
- $\mathsf{H} u.\, A$: immediate by condition 4 and induction.
- $A \oslash u$: immediate by condition 8 and induction.
- $A_1 \mid A_2$: then $P \leftrightarrow P_1 \mid P_2$ so by condition 5 there are Q_1, Q_2, such that $Q \leftrightarrow Q_1 \mid Q_2$ and $P_i \mathcal{R} Q_i$. We conclude by induction.
- $A_1 * A_2$: similar.
- $A_1 \rhd A_2$: let Q_1, Q_2 be such that $Q_2 \leftrightarrow Q \mid Q_1$ and $Q_1 \vDash A_1$. By condition 6 there are P_1, P_2 such that $P_i \mathcal{R} Q_i$. By induction $P_1 \vDash A_1$ and as $P \vDash A_1 \rhd A_2$, $P_2 \vDash A_2$. Another use of the induction hypothesis lets us conclude $Q_2 \vDash A_2$.
- $A_1 \mathrel{-\!\!*} A_2$: similar. □

5.2 Non-congruence of logical equivalence

We now give two examples of intensional bisimulations that show that logical equivalence is strictly coarser than structural congruence, and is not even a congruence in general. These bisimulations are based on the notion of shift functions. A unary function symbol f is called a shift function if there are unary function symbols $g_1, \ldots, g_n$ such that $\mathcal{E} \vdash f(\mathbf{g}(x)) = \mathbf{g}(f(x)) = x$. In the following, we assume some fixed shift function f (we will later on consider the case of the equational theory of trees, for which there is no such function). In cryptographic terms, M and $f(M)$ represent two different pieces of information that are deducible one from another by linear deduction, which explains why they may be indistinguishable for some notion of observer matching our logic.

Let a be some fixed channel name, f a shift function and $\mathbf{g}$ unary functions such that $\mathcal{E} \vdash f(\mathbf{g}(x)) = \mathbf{g}(f(x)) = x$. We consider a transformation $\mathsf{shift}_a^f(P)$ that intuitively shifts all term communications of P on channel a using function f. This can be viewed of as a reversible noise introduced globally on all communications over a.

Definition 11 (Shifted channels) The transformation shift_a^f is inductively defined as a morphism for all syntactic operators but term inputs and outputs on a, for which it is defined as follows:

$$\mathsf{shift}_a^f(\bar{a}\langle M\rangle.P) \triangleq \bar{a}\langle f(M)\rangle.\mathsf{shift}_a^f(P)$$
$$\mathsf{shift}_a^f(a(x).P) \triangleq a(x).\mathsf{shift}_a^f(P[x \leftarrow \mathbf{g}(x)]).$$

For instance, $\mathsf{shift}_a^f(\bar{a}\langle M\rangle \mid a(x).\bar{a}\langle h(x)\rangle) = \bar{a}\langle f(M)\rangle \mid a(x).\bar{a}\langle f(h(\mathbf{g}(x)))\rangle$.

Proposition 6 *The symmetric closure of $\mathcal{R} = \{(P, \mathsf{shift}_f^a(P)), P \in \mathcal{P}\}$ is an intensional bisimulation.*

To prove this, we first derive some technical lemmas.

Lemma 9 $(\mathsf{shift}_f^a(P))[x \leftarrow M] = \mathsf{shift}_f^a(P[x \leftarrow M])$.

Proof By induction on P:

- Homomorphic cases are straightforward;

– For outputs on a:

$$\begin{aligned}
&(\mathsf{shift}^a_f(\bar{a}\langle N\rangle.P))[x\leftarrow M]\\
&\quad = (\bar{a}\langle f(N)\rangle.\mathsf{shift}^a_f(P))[x\leftarrow M] && \text{by definition of shift}\\
&\quad = \bar{a}\langle f(N[x\leftarrow M])\rangle.((\mathsf{shift}^a_f(P))[x\leftarrow M])\\
&\quad = \bar{a}\langle f(N[x\leftarrow M])\rangle.(\mathsf{shift}^a_f(P[x\leftarrow M])) && \text{by induction}\\
&\quad = \mathsf{shift}^a_f((\bar{a}\langle N\rangle.P)[x\leftarrow M]) && \text{by definition of shift}
\end{aligned}$$

– For inputs on a (y is taken fresh for M, x):

$$\begin{aligned}
&(\mathsf{shift}^a_f(a(y).P))[x\leftarrow M]\\
&\quad = (a(y).\mathsf{shift}^a_f(P[y\leftarrow g(y)]))[x\leftarrow M] && \text{by definition of shift}\\
&\quad = a(y).((\mathsf{shift}^a_f(P[y\leftarrow g(y)]))[x\leftarrow M])\\
&\quad = a(y).\mathsf{shift}^a_f(P[y\leftarrow g(y)][x\leftarrow M]) && \text{by induction}\\
&\quad = a(y).\mathsf{shift}^a_f(P[x\leftarrow M][y\leftarrow g(y)])\\
&\quad = \mathsf{shift}^a_f((a(y).P)[x\leftarrow M]) && \text{by definition of shift}
\end{aligned}$$

□

Lemma 10 *If $P \equiv Q$, then* $\mathsf{shift}^a_f(P) \equiv \mathsf{shift}^a_f(Q)$.

Proof The reasoning is done by induction on the proof that $P \equiv Q$. The only cases that require some care are:

ALIAS $\nu x.(\{^M/_x\} \mid P) \equiv P[x\leftarrow M]$
SUBST $\{^M/_x\} \mid P^p \equiv \{^M/_x\} \mid P^p[x\leftarrow M]$

Let us detail the ALIAS case:

$$\begin{aligned}
&\mathsf{shift}^a_f(\nu x.(\{^M/_x\} \mid P))\\
&\quad = \nu x.(\{^M/_x\} \mid \mathsf{shift}^a_f(P)) && \text{by definition of shift}\\
&\quad \equiv (\mathsf{shift}^a_f(P))[x\leftarrow M]\\
&\quad = \mathsf{shift}^a_f(P[x\leftarrow M]) && \text{by Lemma 9}
\end{aligned}$$

The case for SUBST is similar. □

Lemma 11 $\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(P)) \equiv P$.

Proof By induction on P:

– Homomorphic cases are straightforward;
– For outputs on a:

$$\begin{aligned}
&\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(\bar{a}\langle N\rangle.P))\\
&\quad = \mathsf{shift}^a_f(\bar{a}\langle \mathbf{g}(N)\rangle.\mathsf{shift}^a_{\mathbf{g}}(P))\\
&\quad = \bar{a}\langle f(\mathbf{g}(N))\rangle.\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(P))\\
&\quad \equiv \bar{a}\langle N\rangle.\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(P))\\
&\quad \equiv \bar{a}\langle N\rangle.P \quad \text{by induction;}
\end{aligned}$$

– For inputs on a (y is taken fresh for M, x):

$$\begin{aligned}
&\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(a(x).P))\\
&= \mathsf{shift}^a_f(a(x).\mathsf{shift}^a_{\mathbf{g}}(P[x\leftarrow f(x)]))\\
&= a(x).\mathsf{shift}^a_f((\mathsf{shift}^a_{\mathbf{g}}(P[x\leftarrow f(x)]))[x\leftarrow \mathbf{g}(x)])\\
&= a(x).\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(P[x\leftarrow f(x)][x\leftarrow \mathbf{g}(x)]))\\
&\qquad\qquad \text{by Lemma 9}\\
&= a(x).\mathsf{shift}^a_f(\mathsf{shift}^a_{\mathbf{g}}(P))\\
&= a(x).P \qquad\qquad \text{by induction.}
\end{aligned}$$

□

Lemma 12 *For $\dagger$ in $\{*, \nmid\}$, if $P \leftrightarrow P_1 \dagger P_2$ then* $\mathsf{shift}^a_f(P) \leftrightarrow \mathsf{shift}^a_f(P_1) \dagger \mathsf{shift}^a_f(P_2)$.

Proof Straightforward by definition of $*, \nmid$. □

Lemma 13 *If $P \to P'$ then* $\mathsf{shift}^a_f(P) \to \mathsf{shift}^a_f(P')$.

Proof The proof is by case analysis on the contracted redex:

– Channel communication: $P \equiv \nu\mathbf{n}.\, a^{ch}(n).P_1 \mid \bar{a}^{ch}\langle b\rangle.P_2 \mid P_3$, and $P' \equiv \nu\mathbf{n}.\, P_1[n\leftarrow b] \mid P_2 \mid P_3$. Straightforward from definition of shift^a_f.
– Branching test: $P \equiv \nu\mathbf{n}.\, \mathit{if}\ M = N\ \mathit{then}\ P_1\ \mathit{else}\ P_2 \mid P_3$ and $P' \equiv \nu\mathbf{n}.\, P_i \mid P_3$ for $i \in \{1, 2\}$. Straightforward (note that shifting does not affect the test $M = N$).
– Term communication: similarly to channel communication, $P \equiv \nu\mathbf{n}.\, P_3 \mid a(x).P_1 \mid \bar{a}\langle M\rangle.P_2$ and $P' \equiv \nu\mathbf{n}.\, P_1[x\leftarrow M] \mid P_2 \mid P_3$. Some care is then necessary in the application of shift^a_f:

$$\begin{aligned}
\mathsf{shift}^a_f(P) &\equiv \nu\mathbf{n}.\, a(x).\mathsf{shift}^a_f(P_1[x\leftarrow g(x)])\\
&\quad \mid \bar{a}\langle f(M)\rangle.\mathsf{shift}^a_f(P_2) \mid \mathsf{shift}^a_f(P_3)\\
&\to \nu\mathbf{n}.\, (\mathsf{shift}^a_f(P_1[x\leftarrow g(x)]))[x\leftarrow f(M)] \mid\\
&\quad \mathsf{shift}^a_f(P_2) \mid \mathsf{shift}^a_f(P_3)\\
&= \nu\mathbf{n}.\, \mathsf{shift}^a_f(P_1[x\leftarrow g(x)][x\leftarrow f(M)]) \mid\\
&\quad \mathsf{shift}^a_f(P_2) \mid \mathsf{shift}^a_f(P_3)\\
&= \nu\mathbf{n}.\, \mathsf{shift}^a_f(P_1[x\leftarrow M]) \mid\\
&\quad \mathsf{shift}^a_f(P_2) \mid \mathsf{shift}^a_f(P_3)\\
&\equiv \mathsf{shift}^a_f(P')
\end{aligned}$$

□

We are now ready to prove Proposition 6.

Proof (of Proposition 6) Let $\mathcal{R} = \{(P, \mathsf{shift}^a_f(P)),\ P \in \mathcal{P}\}$, and let us prove that the symmetric closure of $\mathcal{R}$ is an intensional bisimulation. Let (P, Q) be some pair in $\mathcal{R}$:

1. $\phi(P) \equiv \phi(Q)$: by definition, shifting does not affect the frame.
2. If $P^p \equiv \mathbf{0}$ then $Q^p \equiv \mathbf{0}$: straightforward.
3. If $u \in \overline{fnv}(P)$ then $u \in \overline{fnv}(Q)$: straightforward.
4. If there is P' such that $P \equiv \nu u.\, P'$ then there is Q' such that $Q \equiv \nu u.\, Q'$ and $P'\ \mathcal{R}\ Q'$: either $P = \mathsf{shift}^a_f(Q)$ or $Q = \mathsf{shift}^a_f(P)$. Assume first $Q = \mathsf{shift}^a_f(P)$. Set $Q' = \mathsf{shift}^a_f(P')$, then by $P \equiv \nu u.\, P'$ and Lemma 10, $Q \equiv \nu u.\, Q'$, which ends the proof. Assume now $P = \mathsf{shift}^a_f(Q)$; then $Q = \mathsf{shift}^a_{\mathbf{g}}(P)$ by Lemma 11, so setting $Q' = \mathsf{shift}^a_{\mathbf{g}}(P')$ entails both $Q \equiv \nu u.\, Q'$ and $P' = \mathsf{shift}^a_f(Q')$, which ends the proof.
5. For $\dagger \in \{*, \nmid\}$, for all P_1, P_2, if $P \leftrightarrow P_1 \dagger P_2$, then there are Q_1, Q_2 such that $Q \leftrightarrow Q_1 \dagger Q_2$ and $P_i\ \mathcal{R}\ Q_i$: either

$P = \mathsf{shift}^a_f(Q)$ or $Q = \mathsf{shift}^a_f(P)$. Assume first $Q = \mathsf{shift}^a_f(P)$. Set $Q_i = \mathsf{shift}^a_f(P_i)$, then by $P \equiv P_1 \dagger P_2$ and Lemma 12, one gets $Q \equiv Q_1 \dagger Q_2$, which ends the proof. The other case is treated similarly using Lemma 11.

6. For $\dagger \in \{*, |\}$, for all P_1, P', if $P_1 \leftrightarrow P \dagger P'$ then there are Q_1, Q' such that $Q_1 \leftrightarrow Q \dagger Q'$, $P' \mathcal{R} Q'$ and $P_1 \mathcal{R} Q_1$: the proof is similar to previous case.
7. If there is P' such that $P \rightarrow P'$ then there is Q' such that $Q \rightarrow Q'$ and $P' \mathcal{R} Q'$: straightforward by Lemma 13.
8. $\nu u.\, P \mathcal{R} \nu u.\, Q$: straightforward. □

Propositions 5 and 6 have some quite unexpected consequences on logical equivalence. First, it entails the following logical equivalences:

$$\bar{a}\langle 0\rangle =_L \bar{a}\langle f(0)\rangle \quad (1)$$

$$\bar{a}\langle 0\rangle \mid \bar{b}\langle 0\rangle =_L \bar{a}\langle f(0)\rangle \mid \bar{b}\langle 0\rangle \quad (2)$$

But the noise introduced on a channel should affect *all* of its communications, as it could otherwise be observed; in particular, it can be proved that:

$$\bar{a}\langle 0\rangle \mid \bar{a}\langle 0\rangle \neq_L \bar{a}\langle f(0)\rangle \mid \bar{a}\langle 0\rangle \quad (3)$$

which, together with (1) shows that $=_L$ is not a congruence.

Proposition 7 (Non-congruence) *Logical equivalence is not a congruence.*

Proof We prove that logical equivalence cannot be a congruence in the general case. In order to do so, we consider an equational theory with a shift function f. We introduce below a formula that will discriminate between the processes $P = \bar{a}\langle f(a)\rangle \mid \bar{a}\langle a\rangle$ and $Q = \bar{a}\langle a\rangle \mid \bar{a}\langle a\rangle$. If $=_L$ was to be a congruence P and Q would have to satisfy the same formulae as Proposition 6 shows that $\bar{a}\langle f(a)\rangle =_L \bar{a}\langle a\rangle$.

For this purpose, we build a formula that plays the following scenario: place a process that listens on a and then branches depending on the input term. With P, there is such a process that may take either branches depending on what output on a it interacted with, but with Q the same branch will be taken in both cases.

Let us first characterize outputs of terms:

$$\mathsf{output} \triangleq \mathsf{single} \wedge (\mathsf{single} \blacktriangleright \Diamond\mathbf{0}) \wedge \mathcal{V}x.\, (\mathsf{Subst}(x) \multimap\!\circledast \top[\copyright x])$$

Lemma 14 *For all processes P, $P \vDash \mathsf{output}$ iff there exists M and a such that $P \equiv \bar{a}\langle M\rangle$.*

Proof This formula is true of a process P if P is guarded and can, together with another guarded process, reduce to the null process. Thus, it has to be a communication of some sort or a conditional; but if P was a conditional, it could not reduce to null when in parallel with another guarded process (there would still be one of the processes left), so P has to be a communication, followed by $\mathbf{0}$. The last part of the formula states that one may unapply some active substitution in P, so P has to be an output of term. The converse is immediate. □

The following formula characterizes a single input followed by a conditional that depends on the result of the input:

$$\mathsf{test\_input} \triangleq \mathsf{single} \wedge (\mathsf{output} \blacktriangleright \Diamond\Diamond\mathbf{1}) \wedge (\mathsf{output} \blacktriangleright \Diamond\Diamond\mathbf{0})$$

Lemma 15 *For all processes P, $P \vDash \mathsf{test\_input}$ iff there exists a and a test $test(x)$ such that there exist terms M and N that respectively validate and invalidate $test(x)$, and either $P \equiv a(x).\mathit{if}\ test(x)\ \mathit{then}\ Q\ \mathit{else}\ \mathbf{0}$ or $P \equiv a(x).\ \mathit{if}\ test(x)\ \mathit{then}\ \mathbf{0}\ \mathit{else}\ Q$ for some Q satisfying $\mathbf{1}$.*

Proof If P satisfies test_input then P is guarded and, together with an output process, may perform two reduction steps resulting in the null process or a guarded process. If the first reduction step was performed inside P in the first case, then it would be the same in the second case. The second reduction step will then have to involve both processes in order to be able to yield the null process, but then the continuation of P will have to satisfy both $\mathbf{1}$ and $\mathbf{0}$ which is impossible.

Hence, the only solution is for P to be a conditional guarded by an input, and the test has to depend on the result of the input given the two different behaviors that one may observe depending on the output atom placed alongside P. The converse is immediate. □

We are now ready to give a formula that may discriminate between P and Q:

$$A \triangleq \mathsf{test\_input} \blacktriangleright ((\mathbf{1} \nmid \Diamond\Diamond\mathbf{1}) \wedge (\mathbf{1} \nmid \Diamond\Diamond\mathbf{0}))$$

We have $P \vDash A$: for $R = a(x).\mathit{if}\ x = a\ \mathit{then}\ R'$ for some process $R' \vDash \mathbf{1}$, it holds that $R \vDash \mathsf{test\_input}$, $\bar{a}\langle a\rangle \mid R \vDash \Diamond\Diamond\mathbf{1}$ and $\bar{a}\langle f(a)\rangle \mid R \vDash \Diamond\Diamond\mathbf{0}$. On the other hand, no process satisfying test_input may produce two different reduction sequences when placed alongside Q as it outputs twice the same term. Hence, $Q \nvDash A$ and logical equivalence is not a congruence. □

Such a phenomenon was already observed for the spatial logic of CCS [9], where $=_L$ coincided with structural congruence up to injective renaming. Non-congruence makes the proof of the converse of Proposition 5 much harder than the Howe-like method used e.g. by Sangiorgi [27], even in the very simple case of CCS. Indeed, a global quantification over the shift function used for each channel should be expressed at the logical level, which calls on for a quantifiers elimination result; despite some progress in that direction (see next section), we did not succeed in using them to prove that $=_L$ is an intensional bisimulation.

5.3 Relaxed structural congruence

Let now $\sim$ be the smallest equivalence on tests $M = N$ such that:

$$\text{SYMMETRY}\quad M = N \sim N = M$$
$$\text{SHIFT}\quad M = N \sim f(M) = f(N)$$

where f is a shift function. Let moreover $\equiv'$ be the smallest congruence extending $\equiv$ with the following axiom:

$$\text{TEST}\quad \begin{array}{l}\textit{if } test \\ \textit{then } P^p \textit{ else } Q^p\end{array} \equiv' \begin{array}{l}\textit{if } test' \\ \textit{then } P^p \textit{ else } Q^p\end{array}$$

when $test \sim test'$. Then the following result can be established:

Proposition 8 (Soundness of $\equiv'$) *$\equiv'$ is an intensional bisimulation.*

To prove this, let us first derive some lemmas. We write $P \sim P'$ for the smallest congruence that satisfies the axiom TEST only, so $\equiv'$ is $(\equiv \cup \sim)^*$.

Lemma 16 *If $P \sim Q$, then $P\sigma \sim Q\sigma$ for all public frames σ (resp. for all name substitutions).*

Proof Straightforward from the definition of $\sim$. □

Lemma 17 *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: $P[x \leftarrow M]$ must contain a test $\mathbf{f}(N_1) = N_2$ that is replaced by $N_1 = \mathbf{g}(N_2)$ in Q. If M is a subterm $\mathbf{f}_1(N_1)$ of $\mathbf{f}(N_1)$ one may possibly have the test $\mathbf{f}_2(x) = N_2$ in P, with $\mathbf{f} = \mathbf{f}_2\mathbf{f}_1$ (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 $N_1 = \mathbf{g}(N_2)$ of Q that is replaced by $\mathbf{g}(\mathbf{f}_2(\mathbf{f}_1(N_1))) = \mathbf{g}(N_2)$. Then $Q_0 = Q'[x \leftarrow M]$ where Q' contains the test $\mathbf{g}(\mathbf{f}_2(x)) = \mathbf{g}(N_2)$, so $P \sim Q'$. □

Lemma 18 *$P \equiv\sim Q$ 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 16 and 17 are used. □

We can now prove the Proposition 8.

Proof (of Proposition 8) Since $\equiv$ is an intensional bisimulation, it is sufficient to prove that $\sim$ is an intensional bisimulation up to $\equiv$. It is straightforward that $\sim$ satisfies conditions 1, 2, 3, and 8. It is also straightforward that it satisfies 4 when u is a name. By Lemma 16, it also satisfies 7. Let us detail the last cases: condition 4 for variables, and conditions 5 and 6.

4. Assume $P \sim Q$ and $P \equiv \nu x.\, P'$. Then $P' \equiv \nu \mathbf{n}.\, \{^M/_x\} \mid P''$ with $P''[x \leftarrow M] \equiv\sim Q$. By Lemma 17, there is Q'' such that $Q \equiv \nu x.\, \nu \mathbf{n}.\, (\{^M/_x\} \mid Q'')$ and $P'' \sim Q''$. Then $P' \sim Q'$ for $Q' = \{^M/_x\} \mid Q''$, which shows condition 4.
5. Assume $P \sim Q$ and $P \leftrightarrow P_1 \mid P_2$. Then by Lemma 18 there is Q', ϕ, P_1', P_2' such that $Q \equiv Q'$ and $Q' \sim \nu \mathbf{n_1}, \mathbf{n_2}.\, \phi \mid P_1' \mid P_2'$. By definition of $\sim$ there is then Q_1', Q_2' such that $Q' = \nu \mathbf{n_1}, n_2.\, \phi \mid Q_1' \mid Q_2'$ and $P_i' \sim Q_i'$, which shows condition 5 for $\parallel$. The $*$ case is analogous.
6. Assume $P \sim Q$ and $P_1 \leftrightarrow P \mid P'$. Then $P_1 \sim Q \mid P'$, which ends the proof by taking $Q_1 = P_1$. □

5.4 Completeness and characteristic formulas

As mentioned earlier, we did not succeed in proving a completeness result in the general case. However, we can characterize logical equivalence in at least two cases:

- when the processes are restricted to those of the π-calculus;
- when the equational theory is the one of finite trees.

The proof relies on defining characteristic formulas for $\equiv'$, which has been proved to be included in $=_L$ above. Some parts of the proof do not use the hypothesis that $\mathcal{E}$ is the equational theory of finite trees, hence the hypothesis is explicitly mentioned for the results that depend on it.

5.4.1 Formulae for π-calculus processes

Consider the formulae of Fig. 2. 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.

Lemma 19 *$P \vDash \mathsf{atom}(a, b)$ iff one of the conditions 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. □

Lemma 20 *$P \vDash \mathsf{test}^{ch}(a, b)$ iff P satisfies conditions 1 or 2 of Lemma 19.*

Fig. 2 Formulae for name communications

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

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

$$\mathsf{out}^{ch}(a,b).A \triangleq \mathsf{single} \wedge \mathcal{U}b'.\,(\mathsf{in}^{ch}(a,c).\mathsf{test}^{ch}(b',c) \blacktriangleright \Diamond(\mathsf{test}^{ch}(b',b) \mathbin{!} A))$$

Fig. 3 Formulae for terms communications

$$\textcircled{c}^{\mathsf{p}}_{=}x \triangleq \top[\textcircled{c}x] \wedge \neg \mathsf{H}z.\,(z \neq x \wedge \textcircled{c}x)[\textcircled{c}z] \qquad \mathsf{out}(a,x) \triangleq \mathsf{single} \wedge \textcircled{c}a \wedge \textcircled{c}^{\mathsf{p}}_{=}x \wedge (\mathsf{single} \blacktriangleright \Diamond \mathbf{0})$$

$$\mathsf{in}(a,x).A \triangleq \mathsf{single} \wedge \neg\Diamond\top \wedge \mathcal{U}x.\,(\mathsf{out}(a,x) \rhd \Diamond A)$$

$$\mathsf{out}(a,M).A \triangleq \mathsf{H}x.\,(x = M)\left[\mathcal{U}b.\,\mathsf{in}(a,y).\mathsf{out}(b,y) \rhd \Diamond\left(\mathsf{out}(b,x) \mathbin{!} (A \wedge \neg\textcircled{c}x)\right)\right]$$

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

Lemma 21 *$P \vDash \mathsf{in}^{ch}(a,b).A$ iff there is P' such that $P \equiv a^{ch}(b').P'$ and $P' \vDash A[b \leftarrow b']$ for some $b' \notin fn(A)$.*

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

Lemma 22 *$P \vDash \mathsf{out}^{ch}(a,b).A$ iff there is P' such that $P \equiv \bar{a}^{ch}\langle b\rangle.P'$ and $P' \vDash A$.*

Proof We have:

$P \vDash \mathsf{out}^{ch}(a,b).A$
- $\Leftrightarrow$ P is single and there exists Q such that
  $P \mid a^{ch}(c).\bar{d}^{ch}\langle e\rangle \to Q$ and $Q \vDash \mathsf{test}^{ch}(b',b) \mathbin{!} A$
  for some $\{d,e\} = \{c,b'\}$ such that $c, b' \notin fnP$
- $\Leftrightarrow$ $P \equiv \bar{a}^{ch}\langle f\rangle.P'$ for some f, P' and
  $P' \mid (\bar{d}^{ch}\langle e\rangle)[c \leftarrow f] \vDash \mathsf{test}^{ch}(b',b) \mathbin{!} A$ for some
  $\{d,e\} = \{c,b'\}$ and $c, b' \notin fn(P)$
- $\Leftrightarrow$ $P \equiv \bar{a}^{ch}\langle b\rangle.P'$ and $P' \vDash A$.

We call a plain process a π-calculus process if it does not contain any term communication or branching. Then, a direct consequence of the two previous lemmas is this first characterization of logical equivalence:

Theorem 4 *For every π-calculus process P, there is a formula $\mathsf{F}_P \in A\pi L$ such that for every extended process Q, $Q \vDash \mathsf{F}_P$ if and only if $Q \equiv P$.*

Proof By induction over P (see also proof of Theorem 5). □

This result and the characteristic formulae shown in Fig. 2 are very similar to what was already known in the case of the π-calculus [16].

5.4.2 Formulae under the empty equational theory

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

Theorem 5 *Let $\mathcal{E}$ be the theory of finite trees. Then for every process P there is a formula $\mathsf{F}_P \in A\pi L$ such that for all processes Q, $Q \vDash \mathsf{F}_P$ if and only if $Q \equiv' P$. In particular, $\equiv'$ is the same as $=_L$.*

Characterizing term communications The first widget we need is a formula $\textcircled{c}^{\mathsf{p}}_{=}x$ 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 point on, 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 3 presents the whole construction.

Lemma 23 *$P \vDash \textcircled{c}^{\mathsf{p}}_{=}x$ 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:

$P \vDash \textcircled{c}^{\mathsf{p}}_{=}x$
- $\Leftrightarrow$ $x \in \overline{fn}(P)$ 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 24 *$P \vDash \mathsf{out}(a,x)$ 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
  $P \mid Q \to \mathbf{0}$, $a \in fn(P)$, $x \in \overline{fv}(P)$,
  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
- $\Leftrightarrow$ $P \equiv \bar{a}\langle x\rangle$

□

Fig. 4 Formulae for conditionals

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

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

Lemma 25 *$P \vDash \mathsf{in}(a,x).A$ iff there is P', x' such that $P \equiv a(x').P'$, $x' \notin fv(A)$, and $P' \vDash A[x \leftarrow x']$.*

Proof Straightforward from previous lemma. □

Lemma 26 *$P \vDash \mathsf{out}(a,M).A$ iff there is P' such that $P \equiv \bar{a}\langle M\rangle.P'$ and $P' \vDash A$.*

Proof Straightforward from previous lemma. □

To summarize, the following result result holds for non-branching processes:

Theorem 6 *Let $\mathcal{E}$ be the theory of finite trees. Then for every non-branching process P, there is a formula $\mathsf{F}_P \in A\pi L$ such that for all extended processes Q, $Q \models \mathsf{F}_P$ if and only if $Q \equiv P$.*

Proof By induction over P (see also proof of Theorem 5). □

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

Lemma 27 *$P \vDash \mathsf{if}$ iff there is M, N, P_1, P_2 such that $P \equiv \mathit{if}\ M = N\ \mathit{then}\ P_1\ \mathit{else}\ P_2$.*

Proof Conditional is the only construct that can perform one step of reduction on its own. Moreover, it can always do so when the process is closed. □

Lemma 28 *$P \vDash \copyright^{\mathsf{br}} x$ iff there is M, N, P_1, P_2 such that $P \equiv \mathit{if}\ M = N\ \mathit{then}\ P_1\ \mathit{else}\ P_2$, $x \in \overline{fv}(P_1, P_2)$ and $x \notin \overline{fv}(M, N)$.*

Proof If $P \vDash \copyright^{\mathsf{br}} x$, then $P \equiv \mathit{if}\ M = N\ \mathit{then}\ P_1\ \mathit{else}\ P_2$ and there is ϕ with $x \notin fv(\phi)$ such that $\phi \mid P \vDash \Diamond\copyright x$, that is $\phi \mid P_1 \vDash \copyright x$ or $\phi \mid P_2 \vDash \copyright x$. The result follows in either case, since $x \notin fv(\phi)$.

The converse is obtained by revealing the substitution $\{^M/_z\}$ and adding with $-\!\circledast$ the frame $\{^N/_z\}$ if $x \in \overline{fv}(P_1)$, thus making the test true, or $\{^n/_z\} \mid \phi'$ where ϕ' closes N and n is a fresh name if $x \in \overline{fv}(P_2)$, thus making the test false. □

Lemma 29 *$P \vDash \mathsf{if}(M = N, A, B)$ iff there are M, N, P_1, P_2 such that $P \equiv \mathit{if}\ test\ \mathit{then}\ P_1\ \mathit{else}\ P_2$ with $test \in \{M = N, N = M\}$, $P_1 \vDash A$ and $P_2 \vDash B$.*

Proof Straightforward. □

Proof of Theorem 5. Assume first that P is a plain process. We define F_P by induction on P:

$$\mathsf{F}_{\upsilon P} \triangleq \mathsf{H}u.\mathsf{F}_P \qquad \mathsf{F}_{\mathbf{0}} \triangleq \emptyset \wedge \mathbf{0}$$
$$\mathsf{F}_{P_1 \mid P_2} \triangleq \mathsf{F}_{P_1} \nmid \mathsf{F}_{P_2}$$
$$\mathsf{F}_{\mathit{if}\ M=N\ \mathit{then}\ P_1\ \mathit{else}\ P_2} \triangleq \mathsf{if}(M = N, \mathsf{F}_{P_1}, \mathsf{F}_{P_2})$$
$$\mathsf{F}a^{ch}(b).P \triangleq \mathsf{in}^{ch}(a,b).\mathsf{F}_P \qquad \mathsf{F}_{\bar{a}^{ch}\langle b\rangle.P} \triangleq \mathsf{out}^{ch}(a,b).\mathsf{F}_P$$
$$\mathsf{F}a(x).P \triangleq \mathsf{in}^{ch}(a,x).\mathsf{F}_P \qquad \mathsf{F}_{\bar{a}\langle M\rangle.P} \triangleq \mathsf{out}(a,M).\mathsf{F}_P$$

Assume now P is a process with a frame. Then P is always congruent to a process of the form $\nu\mathbf{n}.\,\phi\,|(P)^p$, for which a characteristic formula is $\mathsf{H}\mathbf{n}.\,\mathsf{F}_\phi[\mathsf{F}_{(P)^p}]$, which ends the proof. □

6 Elimination of term quantification

6.1 Intuitions

This section is devoted to the construction of a translation of any formula A of $A\pi L$ into a logically equivalent one that does not make use of term quantification, thus proving the following theorem:

Theorem 7 (Term quantification elimination) *For every closed formula $A \in \mathsf{A}\pi\mathsf{L}$, there is a formula $[\![A]\!] \in \mathsf{A}\pi\mathsf{L}\setminus\{\exists\}$ 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, but replaces every subformula $\exists t.\,A'$ with a formula of the form $\mathsf{H}x.\,[\![A']\!]_{\{t\to x\}}$. Hence, the H quantifier is in charge of picking a term M for the new active substitution $\{^M/_x\}$ it reveals, thus mimicking term quantification. Further occurrences of t in the formula are then replaced by x. The successive variable revelations inductively build up an *environment frame* placed alongside the actual process that records witnesses of term quantifications. This environment frame calls for special care during the translation of a formula. For instance, we need to copy it on either sides of a $*$ operator, and on the left-hand side of a $-\!\!*$. Moreover, to follow the semantics of $\exists t.\,A$, one has to make sure that this substitution is not created under hidden

names of the process and does not mention other substitutions belonging to the environment frame.

To keep track of this, the translation will have to be of the form $[\![A]\!]_v$ where v is a valuation $\{\mathbf{t}{\to}\mathbf{x}\}$ that associates previously encountered term variables to their corresponding variables in the domain of the environment frame. The translation thus starts with an empty valuation: $[\![A]\!] \triangleq [\![A]\!]_\emptyset$, and the valuation grows each time a term quantification is encountered. We write e for the environment $\{\mathbf{x}{\to}\mathbf{M}\}$ corresponding to the environment frame $[\![e]\!] \triangleq \{^{\mathbf{M}}/_{\mathbf{x}}\}$. Moreover, we will only consider *well-formed* environments e and translations $[\![A]\!]_v$ where $fv(A, \mathbf{M}) \cap \mathbf{x} = \emptyset$. Finally, when the domain of e matches the codomain of v, we write $e \circ v$ for the valuation $\{\mathbf{t}{\to}\mathbf{M}\}$.

Before going through the technicalities of the proof, let us first give the proof sketches for the translations of $\exists t.\, A$ and $A_1 * A_2$. The formula Φ_v enforces, at each step of the translation, the wellformedness of the environment frame and will be defined later on. The actual translation of term quantification is as follows, where $x_{n+1} \notin fv(A, v)$:

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

It merely creates a fresh substitution, as the inductive hypothesis on $[\![A]\!]_{v\{t\to x_{n+1}\}}$ will suffice to enforce the well-formedness of the new environment frame.

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

$$[\![A_1 * A_2]\!]_v \triangleq \Phi_v \wedge \mathit{И}\mathbf{x'}.\, (\Phi_{v'} \wedge \bigwedge_{x\in\mathbf{x}} \neg\copyright x \wedge \mathsf{Subst}(\mathbf{x'}))$$
$$—\circledast \begin{pmatrix} *_{i=1}^{n} (\mathsf{Subst}(x_i, x_i') \wedge x_i' = x_i) * \top \\ \wedge [\![A_1]\!]_v * [\![A_2]\!]_{v'} \end{pmatrix}.$$

The idea is to add a new environment frame over fresh variables $\mathbf{x'}$. The left-hand side of —$\circledast$ ensures that this is a valid environment which does not make use of the variables of the previous environment. This avoids 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 translations of subformulae A_1 and A_2.

6.2 Technical lemmas

Let us first enunciate three lemmas:

Lemma 30 *For all processes Q, Q' such that $fv(Q, Q') \cap \{\mathbf{x}\} = \emptyset$, if $Q \,|\{^{M_1\ldots M_n}/_{x_1\ldots x_n}\} \equiv Q' \,|\{^{M'_1\ldots M'_n}/_{x_1\ldots x_n}\}$ for some terms $\mathbf{M}, \mathbf{M'}$, then $\mathbf{M} = \mathbf{M'}$ and $Q \equiv Q'$.*

Proof $Q \equiv Q'$ is immediate since the processes do not mention variables in $\mathbf{x}$. According to structural congruence rules for public frames, $\mathbf{M}$ and $\mathbf{M'}$ have to be equal term by term. □

Lemma 31 *If $P \equiv \sigma \mid P'$ for some public frame σ and extended process P' such that $fv(P') \cap dom(\sigma) = \emptyset$, and there exists Q, R such that $R \leftrightarrow P \nmid Q$, then $Q \equiv \sigma \mid Q'$ and $R \equiv \sigma \mid R'$ for some Q' and R', and $R' \leftrightarrow P' \nmid Q'$.*

Proof As $R \leftrightarrow P \nmid Q$ and $P \equiv \sigma \mid P'$, there exists $\mathbf{n_1}, \mathbf{n_2}, \phi, P_1$ and P_2 such that $\mathbf{n_1} \cap fn(P_2) = \mathbf{n_2} \cap fn(P_1) = \emptyset$, $\mathbf{n_1 n_2} \cap fn(\sigma) = \emptyset$ and:

$$P \equiv \nu\mathbf{n_1}.\, (\nu\mathbf{n_2}.\, (\sigma \mid \phi) \mid P_1)$$
$$Q \equiv \nu\mathbf{n_2}.\, (\nu\mathbf{n_1}.\, (\sigma \mid \phi) \mid P_2)$$
$$R \equiv \nu\mathbf{n_1 n_2}.\, (\sigma \mid \phi \mid P_1 \mid P_2)$$

It follows that there are Q' and R' such that $Q \equiv \sigma \mid Q'$ and $R \equiv \sigma \mid R'$. Taking σ out of the processes above gives us:

$$P' \equiv \nu\mathbf{n_1}.\, (\nu\mathbf{n_2}.\, \phi \mid P_1)$$
$$Q' \equiv \nu\mathbf{n_2}.\, (\nu\mathbf{n_1}.\, \phi \mid P_2)$$
$$R' \equiv \nu\mathbf{n_1 n_2}.\, (\phi \mid P_1 \mid P_2)$$

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

Lemma 32 *For every public frame ϕ and processes P, Q, R, $R \leftrightarrow P \nmid Q$ if and only if $(\phi \mid R) \leftrightarrow (\phi \mid P) \nmid (\phi \mid Q)$.*

Proof This is straightforward. □

6.3 The proof

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

Lemma 33 (Induction hypothesis) *$P \vDash [\![A]\!]_v$ if and only if there exists Q and e such that $P \equiv Q \,|[\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$, and $Q, e \circ v \vDash A$.*

To meet the requirements of this lemma and make sure that P is indeed 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:

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

Lemma 34 *For all processes P and valuations v, $P \vDash \Phi_v$ if and only if there exists a process Q and an environment e such that $P \equiv Q \,|[\![e]\!], fv(Q) \cap dom([\![e]\!]) = \emptyset$ and the domain of e matches the codomain of v.*

Proof If $P \vDash \Phi_{\{\mathbf{t}\to\mathbf{x}\}}$ then for all $x \in \mathbf{x}$ there are Q_x and M_x such that $P \equiv \{^{M_x}/_x\} \mid Q_x$ and $x \notin fv(Q_x, M_x)$. So there is a frame $\phi = \{^{M_1\ldots M_n}/_{x_1\ldots x_n}\}$ with $x_i \notin fv(M_j)$ for $(i, j) \in \{1, \ldots, n\}$ and a process Q with $fv(Q) \cap \mathbf{x} = \emptyset$ such that $P \equiv \phi \mid Q$. We can conclude with $e = \{\mathbf{x}{\to}\mathbf{M}\}$. The converse is straightforward. □

Fig. 5 Elimination of term quantification

$$[\![\mathbf{0}]\!]_v \triangleq \Phi_v \wedge \mathbf{0} \qquad [\![\emptyset]\!]_v \triangleq \Phi_v \wedge \mathsf{Subst}(\mathbf{x}) \qquad [\![A_1 \wedge A_2]\!]_v \triangleq \Phi_v \wedge [\![A_1]\!]_v \wedge [\![A_2]\!]_v \qquad [\![\neg A]\!]_v \triangleq \Phi_v \wedge \neg [\![A]\!]_v$$

$$[\![\Diamond A]\!]_v \triangleq \Phi_v \wedge \Diamond [\![A]\!]_v \qquad [\![A_1 \mid A_2]\!]_v \triangleq \Phi_v \wedge ([\![A_1]\!]_v \mid [\![A_2]\!]_v) \qquad [\![A \blacktriangleright B]\!] \triangleq \Phi_v \wedge [\![A]\!]_v \blacktriangleright [\![B]\!]_v$$

$$[\![\mathsf{H}x.A]\!]_v \triangleq \Phi_v \wedge \mathsf{H}x'.[\![A[x \leftarrow x']]\!]_v \quad (x' \notin fv(v)) \qquad [\![\mathsf{H}n.A]\!]_v \triangleq \Phi_v \wedge \mathsf{H}n.[\![A]\!]_v$$

$$[\![©u]\!]_v \triangleq \Phi_v \wedge \mathsf{Subst}(\mathbf{x}) * ((\bigwedge_{x \in \{\mathbf{x}\}} \neg ©x) \wedge ©u) \qquad [\![\exists t.A]\!]_v \triangleq \Phi_v \wedge \mathsf{H}x_{n+1}.[\![A]\!]_{v\{t \rightarrow x_{n+1}\}}$$

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

$$[\![A \multimap\!\circledast B]\!]_v \triangleq \Phi_v \wedge \mathsf{H}\mathbf{x}'.(\bigwedge_{x \in \mathbf{x}} \neg ©x \wedge [\![A]\!]_{v'}) \multimap\!\circledast \begin{pmatrix} \top * (*_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' \neq x_i)) \\ \wedge ([\![B]\!]_v \wedge \bigwedge_{x' \in \mathbf{x}'} \neg ©x') * \mathsf{Subst}(\mathbf{x}') \end{pmatrix}$$

We are now ready to present the inductive translation $[\![A]\!]_v$ shown on Figure 5. We assume that formulae are written using $\blacktriangleright$ and $\multimap\!\circledast$ in lieu of $\rhd$ and $\multimap$ as it is equivalent, albeit easier to do so. Except for the ones above and $\multimap\!\circledast$ (which works the same way as $*$), the meaning of all the cases should be quite straightforward.

The proof of the induction hypothesis will make use of the following lemma:

Lemma 35 $fv([\![A]\!]_v) = fv(A, v)$.

Proof Immediate by induction on the formula A. □

We can now prove Lemma 33 by induction on the formula A. We write $e \equiv e'$ when $e = \{\mathbf{x} \rightarrow \mathbf{M}\}$, $e' = \{\mathbf{x} \rightarrow \mathbf{M}'\}$ and for $i \in \{1 \ldots n\}$, $M_i = M_i'$.

Proof of Lemma 33. – $[\![M = N]\!]_v \triangleq \Phi_v \wedge Mv = Nv$:

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

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

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

– $[\![A_1 \wedge A_2]\!]_v \triangleq \Phi_v \wedge [\![A_1]\!]_v \wedge [\![A_2]\!]_v$: If $P \vDash [\![A]\!]_v$, then by induction there exist $Q_1 \mid [\![e_1]\!] \equiv P$ and $Q_2 \mid [\![e_2]\!] \equiv P$ such that $fv(Q_i) \cap dom([\![e_i]\!]) = \emptyset$ and $Q_i, (e_i \circ v) \vDash A_i$ ($i \in \{1, 2\}$). By Lemmas 30 and 5, 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 $P \vDash [\![\neg A]\!]_v$ then $P \nvDash [\![A]\!]_v$ so, by induction hypothesis, for all Q, e such that $P \equiv Q \mid [\![e]\!]$, $Q, (e \circ v) \nvDash A$. As $P \vDash \Phi_v$, there exists such Q and e and $Q, (e \circ v) \vDash \neg A$, hence the result.
Reciprocally, if there are Q,e such that $P \equiv Q \mid [\![e]\!]$ and $Q, (e \circ v) \vDash \neg A$ with $fv(Q) \cap dom([\![e]\!]) = \emptyset$, then by Lemma 30, for all Q',e' such that $P \equiv Q' \mid [\![e']\!]$ and $fv(Q') \cap dom([\![e']\!]) = \emptyset$, $Q \equiv Q'$ and $e = e'$ so $Q', (e' \circ v) \vDash \neg A$. By induction hypothesis, $P \nvDash [\![A]\!]_v$ so $P \vDash [\![\neg A]\!]_v$.

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

– $[\![A_1 \mid A_2]\!]_v \triangleq \Phi_v \wedge ([\![A_1]\!]_v \mid [\![A_2]\!]_v)$: Straightforward.

– $[\![A \blacktriangleright B]\!] \triangleq \Phi_v \wedge [\![A]\!]_v \blacktriangleright [\![B]\!]_v$: If $P \vDash [\![A \blacktriangleright B]\!]_v$ then $P \equiv P' \mid [\![e]\!]$ for some P' and e such that $fv(P') \cap dom(e) = \emptyset$, and there exists Q, R such that $R \leftrightarrow P \mid Q$, $Q \vDash [\![A]\!]_v$ and $R \vDash [\![B]\!]_v$. By Lemma 31 there are processes Q', R' such that $Q \equiv Q' \mid [\![e]\!]$, $R' \leftrightarrow P' \mid Q'$ and $R \equiv R' \mid [\![e]\!]$. By induction hypothesis, Q', $(e \circ v) \vDash A$ and R', $(e \circ v) \vDash B$, so P', $(e \circ v) \vDash A \blacktriangleright B$.
The converse is similar; it uses Lemma 32 as a converse of Lemma 31.

– $[\![\mathsf{H}x.\,A]\!]_v \triangleq \Phi_v \wedge \mathsf{H}x'.[\![A[x \leftarrow x']]\!]_v \quad (x' \notin fv(v))$: If $P \vDash [\![\mathsf{H}x.\,A]\!]_v$ then $P \equiv Q \mid [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $P \vDash \mathsf{H}x'.[\![A[x \leftarrow x']]\!]_v$. By definition of H, there exists P' and $y \notin fv(Q, e, [\![A[x \leftarrow x']]\!]_v)$ 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 $P' \equiv Q' \mid [\![e']\!]$ 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 $Q', (e \circ v) \vDash \mathsf{H}y.\,A[x \leftarrow y]$, and by α-conversion, $Q, (e \circ v) \vDash \mathsf{H}x.\,A$.
Reciprocally, with the same notations:

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

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

– $[\![©u]\!]_v \triangleq \Phi_v \wedge \mathsf{Subst}(\mathbf{x}) * ((\bigwedge_{x \in \{\mathbf{x}\}} \neg ©x) \wedge ©u)$: 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 $P \equiv Q \mid [\![e]\!]$, $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and $P \vDash \mathsf{Subst}(\mathbf{x}) * ((\bigwedge_{x \in \mathbf{x}} \neg ©x) \wedge ©y)$. This is true if and only if $Q \vDash ©y$, the condition $fv(Q) \cap dom([\![e]\!]) = \emptyset$ and the formula $\bigwedge_{x \in \mathbf{x}} \neg ©x$ ensuring that active substitutions of the environment frame have not been applied, nor unapplied.

– $[\![\exists t.\, A]\!]_v \triangleq \Phi_v \wedge \mathsf{H}x_{n+1}.\, [\![A]\!]_{v\{t\to x_{n+1}\}}$: If $P \vDash [\![\exists t.\, A]\!]_v$ then one the one hand, by Lemma 34, 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 35, 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 $Q \,|\, [\![e\{x\to M\}]\!] \vDash [\![A]\!]_{v\cup\{t\to x\}}$. Together with Lemma 35, this shows $P \vDash \mathsf{H}x_{n+1}.\, [\![A]\!]_{v\cup\{t\to x_{n+1}\}}$, and by Lemma 34 $P \vDash [\![\exists t.\, A]\!]_v$.
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 $\{y_1\to N_1, \ldots, y_n\to N_n\}$.
– $[\![A_1 * A_2]\!]_v$

$$\triangleq \Phi_v \wedge \mathcal{N}\mathbf{x}'.\, \Big(\bigwedge_{x\in\mathbf{x}} \neg\copyright x \wedge \mathsf{Subst}(\mathbf{x}') \wedge \Phi_{v'}\Big) \;\text{—}\circledast\; \begin{pmatrix} *_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' = x_i) * \top \\ \wedge [\![A_1]\!]_v * [\![A_2]\!]'_v \end{pmatrix}$$

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

– $[\![A \text{—}\circledast B]\!]_v$

$$\triangleq \Phi_v \wedge \mathcal{N}\mathbf{x}'.\, \Big(\bigwedge_{x\in\mathbf{x}} \neg\copyright x \wedge [\![A]\!]_{v'}\Big) \;\text{—}\circledast\; \begin{pmatrix} \top * (*_{i=1}^{n}(\mathsf{Subst}(x_i, x_i') \wedge x_i' \neq x_i)) \\ \wedge ([\![B]\!]_v \wedge \bigwedge_{x'\in\mathbf{x}'} \neg\copyright x') * \mathsf{Subst}(\mathbf{x}') \end{pmatrix}$$

As in the cases of $*$ and $\blacktriangleright$, it should be immediate to check that $P \vDash [\![A \text{—}\circledast B]\!]_v$ if and only if there are $P', Q', R', \mathbf{M}, \mathbf{y}$ and $\mathbf{N}$ such that $R' \leftrightarrow P' * Q'$, $fv(M, N, P', Q', R') \cap \mathbf{xy} = \emptyset, \mathbf{N} = \mathbf{M}$, $Q' \,|\, [\![f]\!] \vDash [\![A]\!]_w$ and $R' \,|\, [\![e]\!] \,|\, [\![f]\!] \vDash ([\![B]\!]_v \wedge \bigwedge_{y\in\mathbf{y}} \neg\copyright y) * \mathsf{Subst}(\mathbf{y})$, 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. □

7 Conclusion

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` [6]. Spatial logics for process algebrae with explicit resources have first been studied by Pym [25]. The idea of distributing assertions about knowledge in space using spatial logics has been explored by Mardare [23]. More examples of applications of spatial connectives in cryptographic logics can be found in Kramer's thesis [21]. Hüttel *et al.* gave a logical characterization and characteristic formulae for static equivalence [19] for some classes of equational theories.

Extensions One natural way to extend the logic could be to consider a weak, several steps version of the $\Diamond$ modality. We conjecture that this would allow us to handle the full applied π-calculus with replication, as in the case of ambients [17].

Appendix

See Table 1.

Table 1 Table of auxiliary formulae

Notation	Intuitive meaning	Definition	First seen
$\top$	True	$\mathbf{0} \vee \neg\mathbf{0}$	p. 15
$\bot$	False	$\neg\top$	p. 15
$A[B]$	Frame satisfies A and plain part satisfies B	$(A \wedge \mathbf{0}) \restriction ((B \wedge \emptyset) * \top)$	p. 15
$A \blacktriangleright B$	Dual of $\rhd$	$\neg(A \rhd \neg B)$	p. 15
$A \text{—}\circledast B$	Dual of $\text{—}*$	$\neg(A \text{—}* \neg B)$	p. 15
$\mathbf{1}$	Indivisible plain part	$\neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \restriction \neg\mathbf{0})$	p. 15
$\mathbb{I}$	Indivisible frame	$\neg\emptyset \wedge \neg(\neg\emptyset * \neg\emptyset)$	p. 15
public	No secret name	$\neg\mathsf{H}n.\, \copyright n$	p. 15
single	Guarded plain process	$\mathbf{1} \wedge \emptyset \wedge$ public	p. 15
public_frame	No secret name (static version)	$\neg(\top * (\mathbb{I} \wedge \mathsf{H}x.\, \mathbb{I}))$	p. 20
$\mathsf{Subst}(x)$	Substitution on x	public_frame $\wedge\ (\emptyset \oslash x)$	p. 20
$\mathsf{Subst}(x = M)$	Substitution $\{^M/_x\}$	$\emptyset \oslash x \wedge x = M$	p. 20

References

1. Abadi, M., Cortier, V.: Deciding knowledge in security protocols under equational theories. Theor. Comput. Sci. **367**(1–2), 2–32 (2006)
2. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL'01, pp. 104–115 (2001)
3. Acciai, L., Boreale, M.: Deciding safety properties in infinite-state pi-calculus via behavioural types. In: Proceedings of ICALP'2009 Volume 5556 of Lecture Notes in Computer Science, pp. 31–42, (2009)
4. Acciai, L., Boreale, M., Zavattaro, G.: On the relationship between spatial logics and behavioral simulations. In: Proceedings of FOSSACS 2010 Volume 6014 of Lecture Notes in Computer Science, pp. 146–160, (2010)
5. Blanchet, B.: Automatic proof of strong secrecy for security protocols. In: IEEE Symposium on Security and Privacy, pp. 86–100, Oakland, California (2004)
6. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. In: LICS 2005, pp. 331–340 (2005)
7. Borgström, J.: Static equivalence is harder than knowledge. Electron. Notes Theor. Comput. Sci. **154**(3), 45–57 (2006)
8. Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). J. Inf. Comput. **186**(2), (2003)
9. Caires, L., Lozes, É.: Elimination of quantifiers and undecidability in spatial logics for concurrency. In: CONCUR Volume 3170 of LNCS, pp. 240–257. Springer, London (2004)
10. Calcagno, C., Cardelli, L., Gordon, A.D.: Deciding validity in a spatial logic for trees. In: Shao, Z., Lee, P. (eds.) TLDI, pp. 62–73. ACM, (2003)
11. Calcagno, C., Gardner, P., Zarfaty, U.: Context logic and tree update. In: Palsberg, J., Abadi, M. (eds.) POPL, pp. 271–282. ACM, (2005)
12. Delaune, S., Kremer, S., Ryan, M.D.: Coercion-resistance and receipt-freeness in electronic voting. In: Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06), pp. 28–39, Venice, Italy, July 2006. IEEE Computer Society Press (2006)
13. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects Comput. **13**(3), 341–363 (2002)
14. Gordon, A., Cardelli, L.: Anytime, anywhere: modal logics for mobile ambients. In: ACM Press editor (ed.) POPL 2000, pp. 365–377, (2000)
15. Hirschkoff, D.: An extensional spatial logic for mobile processes. In: CONCUR'02 Volume 3252 of LNCS. Springer, (2002)
16. Hirschkoff, D., Lozes, É., Sangiorgi, D.: Minimality results for spatial logics. In: FSTTCS'03 Volume 2914 of LNCS, Mumbai, India, pp. 252–264. Springer (2003)
17. Hirschkoff, D., Lozes, É., Sangiorgi, D.: On the expressiveness of the ambient logic. Logical Methods Comput. Sci. **2**(2) (2006)
18. Hirschkoff, D., Lozes, É., Sangiorgi, D.: On the expressiveness of the ambient logic. Logical Methods Comput. Sci. **4**(4) (2008)
19. Hüttel, H., Pedersen, M.D.: A logical characterisation of static equivalence. Electron. Notes Theor. Comput. Sci. **173**, 139–157 (2007)
20. Jacquemard, F., Lozes, E., Treinen, R., Villard, J.: First-order constraint systems with multiple congruence relations. (submitted)
21. Kramer, S.: Logical concepts in cryptography. PhD thesis, École Polytechnique Fédérale de Lausanne, (2007)
22. Lozes, É., Villard, J.: A spatial equational logic for the applied π-calculus. In: van Breugel, F., Chechik, M. (eds.) Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08) Volume 5201 of Lecture Notes in Computer Science, pp. 387–401, Toronto, Canada. Springer 2008
23. Mardare, R.: Observing distributed computation. A dynamic-epistemic approach. In: CALCO Volume 4624 of LNCS, pp. 379–393. Springer, (2007)
24. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, i. Inf. Comput. **100**(1), 1–40 (1992)
25. Pym, D.J., Tofts, C.M.N.: A Calculus and logic of resources and processes. Formal Aspects Comput. **18**(4), 495–517 (2006)
26. Reynolds, J.C.: Separation logic: a logic for shared mutable data structures. In 17th IEEE Symposium on Logic in Computer Science (LICS), pp. 55–74, (2002)
27. Sangiorgi, D.: Extensionality and intensionality of the ambient logics. In: POPL (2001)
28. Villard, J., Lozes, É., Treinen, R.: A spatial equational logic for the applied pi-calculus. Research report LSV-08-10, LSV, ENS Cachan, France. 44 pp. (2008)