

An Extensional Spatial Logic for Mobile Processes

Daniel Hirschhoff

LIP – ENS Lyon, France

Abstract. Existing spatial logics for concurrency are intensional, in the sense that they induce an equivalence that coincides with structural congruence. In this work, we study a contextual spatial logic for the π -calculus, which lacks the spatial operators to observe emptiness, parallel composition and restriction, and only has composition adjunct and hiding. We show that the induced logical equivalence coincides with strong early bisimilarity. The proof of completeness involves the definition of non-trivial formulas, including characteristic formulas for restriction-free processes up to bisimilarity. This result allows us to isolate the extensional core of spatial logics, decomposing spatial logics into a part that counts (given by the intensional operators) and a part that observes (given by their adjuncts). We also study how enriching the core extensional spatial logic with intensional operators affects its separative power.

1 Introduction

Spatial logics extend classical logic with constructions to reason about the structure of the underlying model (when applied to concurrent systems, the models are processes). The additional connectives belong to two families. *Intensional operators* allow one to inspect the structure of the model. A formula $\mathcal{A}_1|\mathcal{A}_2$ is satisfied whenever we can split the structure into two parts satisfying the corresponding subformula \mathcal{A}_i , $i = 1, 2$. In presence of restriction in the underlying model, a structure P satisfies formula $n\mathbb{R}\mathcal{A}$ if we can write P as $(\nu n)P'$ with P' satisfying \mathcal{A} . Finally, formula 0 is only satisfied by the empty structure. Connectives $|$ and \mathbb{R} come with adjunct operators, called guarantee (\triangleright) and hiding (\odot) respectively, that allow one to extend the structure being observed. In this sense, these can be called *contextual operators*. P satisfies $\mathcal{A}_1 \triangleright \mathcal{A}_2$ whenever the spatial composition (using $|$) of P with any structure satisfying \mathcal{A}_1 satisfies \mathcal{A}_2 , and P satisfies $\mathcal{A} \odot n$ if $(\nu n)P$ satisfies \mathcal{A} .

Previous studies have demonstrated that in existing spatial logics, the intensional character prevails. In the static case, where spatial logics are used to reason about semi-structured data [CG01a], or about memory along the execution of a program that manipulates pointers [Rey02], the guarantee operator is eliminable, in the sense that every formula involving \triangleright can be replaced by an equivalent formula that does not make use of \triangleright [Loz03,Loz04,DGG04]. In spatial logics for concurrency [CG00,CC01], that also include a temporal modality, this is not the case. However, the equivalence on processes induced by the logic

coincides with structural congruence, a very fine grained relation on processes — much finer in particular than behavioural equivalence [San01,HLS02,CL04]. This situation is in contrast with standard modal logics for concurrency like the Hennessy-Milner (HM for short) logic [MPW93], for which logical equivalence is known to coincide with bisimilarity.

Technically, the ability for spatial logic to capture structural congruence on processes is based on two aspects of its expressiveness. The first aspect is the ability to *count*, i.e., to express arithmetical properties about the number of substructures exhibited by a given system. The second aspect is the definability of modalities à la Hennessy-Milner within the logic, i.e., one is able to capture parts of the behaviour of processes. This has been shown in [San01,HLS02], and further studied in [HLS03], using a logic with a restricted set of operators, and applying it to both the Ambient calculus and the π -calculus (modality formulas are also derived in [CL04]). In [HLS03], in particular, the derivability of modality formulas for the π -calculus and for Mobile Ambients heavily relies on the use of intensional operators, in conjunction with guarantee: $|$ and 0 are used to isolate some kind of elementary components of interaction (called ‘threads’), while the revelation operator makes it possible to test the free names of a process, and to deduce behavioural properties.

In this work, we renounce to the intensional connectives, and study the resulting contextual spatial logic, called \mathcal{L} . \mathcal{L} only has spatial composition adjunct (\triangleright), revelation adjunct (\odot), a simple temporal modality (\diamond), and an operator \mathcal{N} for fresh name quantification. We apply \mathcal{L} to reason about the π -calculus, and we show *extensionality* of the logic, in the sense that \mathcal{L} induces the same separative power as strong early bisimilarity (and thus as Hennessy-Milner logic). This result suggests that the two families of operators in spatial logics serve different purposes: while intensional operators allow one to count (as illustrated by the study in [DLM04], where it is shown that a particular static spatial logic, in which \triangleright is eliminable, characterises Presburger arithmetic), we show that contextual operators are enough to bring extensionality.

To establish our main result, we exploit the characterisation of strong bisimilarity (written \sim) in terms of barbed equivalence (written \simeq). The elementary observations available in \mathcal{L} are indeed close to the definition of \simeq . However, technically, we still need to define a way to perform *instantaneous* observations (to detect barbs) in \mathcal{L} , which is a priori not obvious given the definition of the logic. We are only able to define formulas for barbs when imposing a bound on the size of processes, but this is enough for our purposes. Another aspect of the expressive power we need in order to capture \simeq is the ability to let two processes ‘pass the same tests’. This is achieved by defining characteristic formulas for restriction-free processes up to \sim . These formulas exploit the constructions for barbs, and are relatively concise thanks to some specific properties of bisimilarity on the calculus without restriction. As hinted above, due to the absence of intensional operators, our constructions depart from the formulas for modalities defined in related works [San01,HLS02,HLS03,CL04].

While we use \simeq in order to show that logical equivalence for \mathcal{L} coincides with \sim , the argument does not follow the classical proof that \simeq is included in \sim , and we instead use the ideas we just sketched. We briefly study also \mathcal{L}^\dagger , an adaptation of \mathcal{L} that is closer to the observations given in \simeq (detecting barbs is primitive in \mathcal{L}^\dagger). We show that \mathcal{L}^\dagger is also an extensional logic.

Having isolated a core extensional spatial logic, we may wonder what lies between \mathcal{L} and full spatial logics for concurrency. To address this question, we establish some results about the expressive and separative power we obtain when enriching \mathcal{L} with (some) intensional operators. These results suggest that from the point of view of separability, the most powerful intensional operator is \textcircled{R} .

Outline. We introduce the calculus and the logic we study in Section 2. Formulas for (some of the) π -calculus modalities and to characterise bisimilarity classes of restriction-free processes are presented in Section 3. In Section 4, we exploit these constructions to prove that \mathcal{L} is extensional. Section 5 is devoted to the discussion of variants and enrichments of \mathcal{L} , and we conclude in Section 6.

2 Preliminaries

2.1 The π -calculus

The finite synchronous π -calculus is introduced using an infinite set of names, ranged over using a, b, \dots, m, n, \dots . Processes, ranged over using P, Q, R, \dots , are defined by the following syntax:

$$P ::= \mathbf{0} \mid P_1|P_2 \mid (\nu n)P \mid m(n).P \mid \overline{m}(n).P.$$

Trailing occurrences of $\mathbf{0}$ will often be omitted. Name n is bound in an input-prefixed term $m(n).P$, and in a restricted term $(\nu n)P$. A name that is not bound is free, and $\text{fn}(P)$ will denote the set of free names of P . We write $P_{\{n \leftarrow m\}}$ for the process resulting from the capture-avoiding replacement of n with m in P .

Actions of the labelled transition system, ranged over with μ , are defined by the following syntax (notice the presence of free input):

$$\mu ::= mn \mid \overline{m}(n) \mid \overline{m}(n) \mid \tau.$$

Given an action μ , we define its names ($\text{n}(\mu)$), free names ($\text{fn}(\mu)$) and bound names ($\text{bn}(\mu)$) as usual. Figure 1 presents the transition rules that define the operational semantics of the π -calculus (symmetrical versions of rules involving parallel composition are omitted). We write $P \xrightarrow{\overline{m}(n)} P'$ whenever $P \xrightarrow{\overline{m}(n)} P'$ or $P \xrightarrow{\overline{m}(n)} P'$.

Structural congruence, \equiv , is the least equivalence relation that is a congruence and that satisfies the rules of Figure 2. Given a (possibly empty) sequence of names $\tilde{n} = n_1, \dots, n_k$, $(\nu \tilde{n})P$ will stand for $(\nu n_1) \dots (\nu n_k)P$. We will also implicitly reason up to permutation of consecutive restrictions, thus treating \tilde{n} as a set of names.

$$\begin{array}{c}
\overline{m}\langle n \rangle . P \xrightarrow{\overline{m}\langle n \rangle} P \quad m(n) . P \xrightarrow{ma} P_{\{n \leftarrow a\}} \quad \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q} \text{bn}(\mu) \cap \text{fn}(Q) = \emptyset \\
\\
\frac{P \xrightarrow{\mu} P'}{(\nu n) P \xrightarrow{\mu} (\nu n) P'} \quad n \notin \text{n}(\mu) \quad \frac{P \xrightarrow{\overline{m}\langle n \rangle} P'}{(\nu n) P \xrightarrow{\overline{m}\langle n \rangle} P'} \quad m \neq n \\
\\
\frac{P \xrightarrow{\overline{m}\langle n \rangle} P' \quad Q \xrightarrow{mn} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \quad \frac{P \xrightarrow{\overline{m}\langle n \rangle} P' \quad Q \xrightarrow{mn} Q'}{P|Q \xrightarrow{\tau} (\nu n) (P'|Q')} \quad n \notin \text{fn}(Q)
\end{array}$$

Fig. 1. Early operational semantics

$$\begin{array}{c}
P|\mathbf{0} \equiv P \quad P|Q \equiv Q|P \quad P|(Q|R) \equiv (P|Q)|R \quad (\nu n) \mathbf{0} \equiv \mathbf{0} \\
(\nu n)(\nu m) P \equiv (\nu m)(\nu n) P \quad P|(\nu n) Q \equiv (\nu n)(P|Q) \text{ if } n \notin \text{fn}(P)
\end{array}$$

Fig. 2. Structural congruence

The *public π -calculus* consists in the set of restriction-free processes. We shall also call P a *public process* whenever $P \equiv Q$ for some Q in the public π -calculus. Given a process P , we write $\text{size}(P)$ for the number of prefixes of P . By definition, if $P \equiv Q$, then $\text{size}(P) = \text{size}(Q)$. A process P is an *atom* if $\text{size}(P) = 1$.

We define some basic observations, usually called barbs, as follows: we write $P \downarrow_n$ (resp. $P \downarrow_{\overline{n}}$) whenever $P \equiv (\nu \tilde{m}) (n(a).P_1|P_2)$ (resp. $P \equiv (\nu \tilde{m}) (\overline{n}(a).P_1|P_2)$) for some n, a, \tilde{m}, P_1 and P_2 such that $n \notin \tilde{m}$.

We shall write relation composition using juxtaposition, and the negation of a relation \simeq will be written $\not\simeq$. We do not give the usual definition of reduction, and instead equivalently (see [SW01]) set $P \longrightarrow Q \stackrel{\text{def}}{=} P \xrightarrow{\tau} \equiv Q$.

2.2 Behavioural Relations

Definition 1 (Behavioural equivalences).

- Strong bisimilarity, \sim , is the greatest symmetrical relation such that whenever $P \sim Q$ and $P \xrightarrow{\mu} P'$, there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$.
- Strong barbed bisimilarity, \simeq , is the greatest symmetrical relation such that whenever $P \simeq Q$:
 - (i) For any n , $(P \downarrow_n \text{ iff } Q \downarrow_n)$ and $(P \downarrow_{\overline{n}} \text{ iff } Q \downarrow_{\overline{n}})$.
 - (ii) For any P' s.t. $P \longrightarrow P'$, there exists Q' s.t. $Q \longrightarrow Q'$ and $P' \simeq Q'$.
- P and Q are strong barbed equivalent, written $P \simeq Q$, iff for any process R , $P|R \simeq Q|R$.

In the sequel, we shall often omit the word ‘strong’ when mentioning these equivalences. The labelled transition system-based and reduction-based presentations for behavioural equivalence coincide, as expressed by the following result.

Theorem 1 ([SW01]). $P \sim Q$ iff $P \simeq Q$.

We shall need the following results about behavioural equivalence.

Proposition 1. Define \sim_s like \sim except that for actions μ of the form mn , when comparing two processes P and Q , we only consider names n belonging to the (finite) set $\text{fn}(P) \cup \text{fn}(Q) \cup \{d\}$, where $d \notin \text{fn}(P) \cup \text{fn}(Q)$. Then $\sim = \sim_s$.

Lemma 1. $\sim \subseteq \simeq$, and $\sim \simeq \sim \subseteq \sim$.

Lemma 2. Given a process P , we have the following:

1. There exist names \tilde{n} and a public process P_0 such that $P \sim (\nu \tilde{n}) P_0$.
2. If $\text{size}(P) < 2 * k$ for some integer k , then P cannot perform a sequence of reductions of length equal to k .

Proof. The first result follows from the two laws $P|(\nu n)Q \sim (\nu n)(P|Q)$ (when $n \notin \text{fn}(P)$) and $\alpha.(\nu n)P \sim (\nu n)\alpha.P$ where α is a prefix of the form $m(m')$ or $\bar{m}(m')$ with $n \notin \{m, m'\}$. \square

Note that the results of this lemma hold because we work in a finite calculus.

The following lemma shows that on the public π -calculus, bisimilarity is a quite discriminating relation.

Lemma 3. Given two public processes P and Q , if $P \sim Q$, then $\text{fn}(P) = \text{fn}(Q)$, $\text{size}(P) = \text{size}(Q)$ and moreover P and Q have the same number of input (resp. output) prefixes. In particular, for P public, $P \sim \mathbf{0}$ implies $P \equiv \mathbf{0}$.

2.3 The Logic

Formulas of \mathcal{L} , the contextual spatial logic, are ranged over using $\mathcal{A}, \mathcal{B}, \dots$, and are given by the following grammar:

$$\mathcal{A} ::= \top \mid \neg \mathcal{A} \mid \mathcal{A}_1 \wedge \mathcal{A}_2 \mid \diamond \mathcal{A} \mid \mathcal{A}_1 \triangleright \mathcal{A}_2 \mid \mathcal{A} \otimes n \mid \forall n. \mathcal{A}.$$

Name n is bound in $\forall n. \mathcal{A}$, and we let $\text{fn}(\mathcal{A})$ stand for the set of free names of \mathcal{A} . $\mathcal{A}_{\{n \leftarrow m\}}$ (resp. $\mathcal{A}_{\{n \leftrightarrow m\}}$) stands for the formula obtained by replacing (resp. permuting) all occurrences of n with m (resp. and m) in \mathcal{A} .

Definition 2 (Satisfaction in \mathcal{L} , logical equivalence). The judgement $P \models \mathcal{A}$, saying that process P satisfies formula \mathcal{A} , is defined as follows:

- $P \models \top$ always;
- $P \models \neg \mathcal{A}$ iff not $P \models \mathcal{A}$ (also written $P \not\models \mathcal{A}$);
- $P \models \mathcal{A}_1 \wedge \mathcal{A}_2$ iff $P \models \mathcal{A}_1$ and $P \models \mathcal{A}_2$;
- $P \models \diamond \mathcal{A}$ iff there exists P' s.t. $P \longrightarrow P'$ and $P' \models \mathcal{A}$;

- $P \models \mathcal{A}_1 \triangleright \mathcal{A}_2$ iff for any Q s.t. $Q \models \mathcal{A}_1$, $P|Q \models \mathcal{A}_2$;
- $P \models \mathcal{A} \otimes n$ iff $(\nu n)P \models \mathcal{A}$.
- $P \models \forall n.\mathcal{A}$ iff for any m s.t. $m \notin \text{fn}(P)$ and $m \notin \text{fn}(\mathcal{A})$, $P \models \mathcal{A}_{\{n \leftrightarrow m\}}$.

P and Q are logically equivalent, written $P =_{\mathcal{L}} Q$, iff for any formula \mathcal{A} , $P \models \mathcal{A}$ iff $Q \models \mathcal{A}$.

We will also make use in our constructions of the following derived formulas:

$$\begin{aligned} \perp &\stackrel{\text{def}}{=} \neg\top & \mathcal{A}_1 \vee \mathcal{A}_2 &\stackrel{\text{def}}{=} \neg(\neg\mathcal{A}_1 \wedge \neg\mathcal{A}_2) & \Box\mathcal{A} &\stackrel{\text{def}}{=} \neg\Diamond\neg\mathcal{A} \\ \mathcal{A}_1 \blacktriangleright \mathcal{A}_2 &\stackrel{\text{def}}{=} \neg(\mathcal{A}_1 \triangleright \neg\mathcal{A}_2) \end{aligned}$$

The interpretation of \perp , \vee and \Box ('always') is standard. $P \models \mathcal{A}_1 \blacktriangleright \mathcal{A}_2$ iff there exists Q s.t. $Q \models \mathcal{A}_1$ and $P|Q \models \mathcal{A}_2$. Operators \triangleright and \blacktriangleright are right associative, and we define the following abbreviation: $\mathcal{A}_{\triangleright}^1 \mathcal{B} \stackrel{\text{def}}{=} \mathcal{A} \triangleright \mathcal{B}$, and $\mathcal{A}_{\triangleright}^{k+1} \mathcal{B} \stackrel{\text{def}}{=} \mathcal{A} \triangleright (\mathcal{A}_{\triangleright}^k \mathcal{B})$. We also set $\Diamond^1 \mathcal{A} \stackrel{\text{def}}{=} \Diamond \mathcal{A}$ and $\Diamond^{k+1} \mathcal{A} \stackrel{\text{def}}{=} \Diamond(\Diamond^k \mathcal{A})$. A process P satisfies $\mathcal{A}_{\triangleright}^k \mathcal{B}$ iff P , put in parallel with k processes satisfying \mathcal{A} , satisfies \mathcal{B} . $P \models \Diamond^k \mathcal{A}$ iff P can perform k reductions and then satisfy \mathcal{A} .

Proposition 2. *If $P \models \mathcal{A}$, then for any Q , $P \equiv Q$ implies $Q \models \mathcal{A}$.*

This result implies that $\equiv \subseteq =_{\mathcal{L}}$, and that for any P and \mathcal{A} , $P \models \Diamond \mathcal{A}$ iff $P \models \langle \tau \rangle.\mathcal{A}$, where $\langle \tau \rangle$ is the HM modality corresponding to $\xrightarrow{\tau}$ [MPW93].

3 Expressiveness of the Logic

3.1 Auxiliary Formulas – Characterising Basic Processes

We start by some technical constructions to capture elementary π -calculus terms.

$$\begin{aligned} \text{nil} &\stackrel{\text{def}}{=} \Box\perp \wedge (\Box\perp \triangleright \Box\perp) & \llbracket 1 \rrbracket &\stackrel{\text{def}}{=} \Diamond\top & \llbracket k+1 \rrbracket &\stackrel{\text{def}}{=} \Diamond\top \wedge \Box\llbracket k \rrbracket \\ \text{atom} &\stackrel{\text{def}}{=} \Box\perp \wedge (\top \blacktriangleright \Diamond\text{nil}) & \text{atom}(n) &\stackrel{\text{def}}{=} \text{atom} \wedge (\text{nil} \otimes n) \\ *(n) &\stackrel{\text{def}}{=} \forall a. (\text{atom}(n) \blacktriangleright \Diamond \text{atom}(a)) & \bar{n}\langle m \rangle &\stackrel{\text{def}}{=} *(n) \triangleright \Diamond \text{atom}(m) \\ n(-) &\stackrel{\text{def}}{=} \bar{n}\langle n \rangle \triangleright \Diamond \text{nil} & \text{duo}(n, m) &\stackrel{\text{def}}{=} (\text{atom}(m) \triangleright \Box\perp) \wedge (\bar{n}\langle n \rangle \triangleright \Diamond \bar{m}\langle m \rangle) \\ \text{testOut}(n, m) &\stackrel{\text{def}}{=} \forall a. \text{atom}(n) \blacktriangleright \Diamond \text{duo}(a, m) \\ \text{testIn}(n, m, a) &\stackrel{\text{def}}{=} (\text{atom}(a) \triangleright \Box\perp) \wedge (*(n) \triangleright \Diamond (\text{atom}(m) \blacktriangleright \Diamond \bar{a}\langle a \rangle)) \end{aligned}$$

We briefly comment on these formulas. Using the strong interpretation of operator \diamond , we can capture the class of processes that are bisimilar to an atom: formula **atom** says that these are processes that necessitate the addition of a context in order to evolve in one step of reduction to a process satisfying **0**. We then distinguish between the input and output polarity by exploiting the ability for a process to interact on a received fresh channel, in formula $\ast(n)$. This formula is then used to derive formulas to characterise the \sim -class of some processes of sizes 1, 2 and 3. The following lemma states this formally:

Lemma 4. *The above formulas have the following interpretation:*

- $P \models \mathbf{nil}$ iff $P \sim \mathbf{0}$.
- $P \models \llbracket k \rrbracket$ iff there is no reduction sequence $P = P_0 \longrightarrow P_1 \longrightarrow \dots \longrightarrow P_i$, with $i < k$, such that P_i cannot perform any \longrightarrow -transition.
- $P \models \mathbf{atom}(n)$ iff $P \sim n(m)$ or $P \sim \bar{n}(m)$, for some m ; for formula **atom**, name n is not fixed.
- $P \models \ast(n)$ iff $P \sim n(x).\bar{x}(y)$ or $P \sim n(x).x(y)$ for some y .
- $P \models \bar{n}(m)$ iff $P \sim \bar{n}(m)$, and $P \models n(_)$ iff $P \sim n(x)$ for some x .
- When $n \neq m$, $P \models \mathbf{duo}(n, m)$ iff $P \sim n(x).\bar{m}(m)$ for some $x \neq m$.
- $P \models \mathbf{testOut}(n, m)$ iff $P \sim n(x).x(y).\bar{m}(m)$ for some y .
- $P \models \mathbf{testIn}(n, m, a)$ iff $P \sim \bar{n}(m).\bar{a}(a)$ when $a \neq n$ and $a \neq m$.

Proof (Sketch). The interpretation of formulas **nil**, $\llbracket k \rrbracket$, **atom** and **atom**(n) is easy. We sketch the proof of some of the other cases. We say that P is \sim -atomic if $P \models \mathbf{atom}$.

Suppose $P \models \ast(n)$, and take a s.t. $a \neq n$ and $a \notin \text{fn}(P)$. We put P in presence of a process Q bisimilar to $\bar{n}(m)$ or to $n(m)$ for some m . Since $P|Q$ reduces to a process that admits a as a free name and $a \notin \text{fn}(P)$, $a \in \text{fn}(Q)$, and thus $Q \sim \bar{n}(a)$ because $a \neq n$. The reduction is necessarily an interaction between P and Q , since it leads to a term satisfying **atom**(a), and $n \neq a$. Hence $P \xrightarrow{n(a)} P'$ for some P' , and $P'|0 \models \mathbf{atom}(a)$. As $a \notin \text{fn}(P)$, we can conclude.

Suppose $P \models \bar{n}(m)$. We put P in presence of a process Q satisfying $\ast(n)$; the reduction step must be an interaction between P and Q , and P is \sim -atomic, otherwise we could not reach a \sim -atomic process. Necessarily $P \sim \bar{n}(m)$, because otherwise it could not react with Q and lead to a process satisfying **atom**(m).

Suppose $P \models \mathbf{duo}(n, m)$, and $n \neq m$. P cannot interact at m and can receive n at n , leading to a process which is bisimilar to $\bar{m}(m)$. This is enough to conclude.

Suppose $P \models \mathbf{testOut}(n, m)$, $a \notin \text{fn}(P)$, $a \neq n$, and $a \neq m$. We reason like in the case of $\ast(n)$ to deduce that P is put in presence with a process bisimilar to $\bar{n}(a)$. This implies that $P \xrightarrow{n(a)} P' \models \mathbf{duo}(a, m)$, which allows us to conclude. \square

3.2 Detecting Barbs

Although \mathcal{L} allows us to put a process in an arbitrary context built using parallel composition and restriction, what is missing to capture behavioural equivalence

is the ability to perform *instantaneous* observations. We achieve this by introducing formulas to characterise barbs, i.e., the possibility for a term to offer an interaction. We have not been able to define such formulas in the general case. Instead, our constructions depend on the size of the tested process, and are thus parametric over a natural number $k \geq 1$:

$$\begin{aligned} \text{pol}^{(k)}(a) &\stackrel{\text{def}}{=} a(-)_\triangleright^k \diamond^k \text{nil} \\ \text{prefpol}^{(k)}(n, a) &\stackrel{\text{def}}{=} (\text{atom}(a) \triangleright \square \perp) \wedge (\bar{n}\langle n \rangle \triangleright \diamond \text{pol}^{(k)}(a)) \\ \downarrow_{\bar{n}}^{(k)} &\stackrel{\text{def}}{=} \mathcal{I}a.\mathcal{I}b. (\text{duo}(n, a) \triangleright \text{prefpol}^{(k)}(a, b) \triangleright a(-) \triangleright b(-)_\triangleright^k \diamond \diamond \llbracket k \rrbracket) \\ \downarrow_n^{(k)} &\stackrel{\text{def}}{=} \mathcal{I}a.\mathcal{I}b. (\text{testIn}(n, n, a) \triangleright \text{prefpol}^{(k)}(a, b) \triangleright a(-) \triangleright b(-)_\triangleright^k \diamond \diamond \llbracket k \rrbracket) \end{aligned}$$

Lemma 5. *Given a process P and an integer k , we have:*

- $P \models \text{pol}^{(k)}(a)$ iff P is bisimilar to a term of the form $(\nu \tilde{c}) (\bar{a}\langle b_1 \rangle \mid \dots \mid \bar{a}\langle b_k \rangle)$, for some names b_1, \dots, b_k , $\tilde{c} \subseteq \{b_1, \dots, b_k\}$ and $a \notin \tilde{c}$.
- $P \models \text{prefpol}^{(k)}(n, a)$ iff $n \neq a$ and $P \sim n(x).P'$ with $P' \models \text{pol}^k(a)$.
- $P \models \downarrow_{\bar{n}}^{(k)}$ iff $P \xrightarrow{\bar{n}\langle m \rangle} P'$ for some m, P' , when $2 * k - 1 > \text{size}(P)$.
- $P \models \downarrow_n^{(k)}$ iff $P \xrightarrow{n\langle m \rangle} P'$, for some m, P' , when $2 * k - 1 > \text{size}(P)$.

Proof (sketch). We focus on formula $\downarrow_{\bar{n}}$. We first show that if $P \xrightarrow{\bar{n}\langle m \rangle} P'$ for some m and P' , then $P \models \downarrow_{\bar{n}}$. When P is put in parallel with the processes specified by the formula for $\downarrow_{\bar{n}}$, we can observe the following two reaction steps: P can interact with the process satisfying $\text{duo}(n, a)$, thus liberating a process that can perform an output on a , which can in turn react with the term satisfying $\text{prefpol}^{(k)}(a, b)$, yielding a state where formula $\llbracket k \rrbracket$ is satisfied (thanks to communications on b).

Suppose now $P \models \downarrow_{\bar{n}}$ and $\text{size}(P) < 2 * k - 1$. The scenario described by formula $\downarrow_{\bar{n}}$ expresses a property of the reductions of a process of the form $T = P|Q_1|Q_2|Q_3|Q_4$, where the Q_i s are tester processes specified by the formula:

- Q_1 can perform an input at n followed by an output at a .
- Q_2 starts by performing an input on a and then (independently from the received value) is liable to do k outputs at b .
- Q_3 just performs an input at a , while Q_4 can perform k inputs at b .

First observe that, since a and b are fresh, process $Q_1|Q_2|Q_3|Q_4$ cannot reduce on its own, and can only perform an input at n .

Then, consider a reduction $T \longrightarrow T'$, and suppose that it comes from a reduction $P \longrightarrow P'$, the Q_i s remaining inactive. We show that formula $\diamond \llbracket k \rrbracket$ does not hold for T' . For this, we look for a term T'' s.t. $T' \longrightarrow T''$ and $T'' \models \llbracket k \rrbracket$.

1. Suppose the reduction to T'' results from P' performing a free output at n (the case where P' performs a bound output is treated similarly) and synchronising with Q_1 , then we obtain a process $T'' = P''|Q'_1|Q_2|Q_3|Q_4$, where $P' \xrightarrow{\bar{n}(m)} P''$ for some m and $Q'_1 \xrightarrow{\bar{a}(a)} \sim \mathbf{0}$. This entails that we can derive $T'' \longrightarrow \sim U = P''|Q_2|Q_4$ from an interaction between Q'_1 and Q_3 . As a result, process $Q_2|Q_4$ is stuck in all possible evolutions of U , since P'' does not know names a and b . So the only possible reductions of U are reductions resulting from P'' on its own. Since $\text{size}(P) < 2 * k - 1$, we can conclude using Lemma 2 that $T'' \not\models \llbracket k \rrbracket$.
2. Suppose the reduction to T'' results from P' performing a reduction step on its own, to a process P'' . Then there are two cases:
 - (a) Either there exists R s.t. $P'' \longrightarrow^* R$ and $R \xrightarrow{\bar{n}(m)} R'$ for some m and R' . In this case, we reason as above to show that $R|Q_1|Q_2|Q_3|Q_4 \longrightarrow R'|Q'_1|Q_2|Q_3|Q_4 \longrightarrow \sim R'|Q_2|Q_4$, and in the resulting state, process $Q_2|Q_4$ is stuck, which shows that $T'' \not\models \llbracket k \rrbracket$. The case where R does a bound output at n is treated similarly.
 - (b) Either such an R does not exist, in which case $Q_1|Q_2|Q_3|Q_4$ is frozen in all possible evolutions of P'' , and, since $\text{size}(P) < 2 * k - 1$, $T'' \not\models \llbracket k \rrbracket$.

So finally, there is no process T'' fulfilling the conditions stated above. This implies that no reduction of T involving only P can lead to a state where the formula is satisfied. So necessarily, P has to interact with $Q_1|Q_2|Q_3|Q_4$, which is possible only if P can perform an output at n .

The interpretation of formula \downarrow_n follows the same ideas, the testing process being specified using formula `testIn` instead of `duo`. \square

When clear from the context, we will omit the superscript (k) in $\downarrow_n^{(k)}$, $\downarrow_{\bar{n}}^{(k)}$; we will do so in particular when P is fixed (cf. the proof of Lemma 8).

3.3 Characteristic Formulas for Public Processes

As remarked above, \mathcal{L} has the modality $\langle \tau \rangle$, due to the presence of constructor \diamond . We further have derivability of the following modalities, that will be useful below to define characteristic formulas.

$$\begin{aligned} \langle mn \rangle^{(k)} . \mathcal{A} &\stackrel{\text{def}}{=} \forall a. \text{testIn}(m, n, a) \triangleright (\text{atom}(a) \blacktriangleright \diamond \diamond (\neg a^\dagger \wedge \mathcal{A})) \\ \langle \bar{m} \langle n \rangle \rangle^{(k)} . \mathcal{A} &\stackrel{\text{def}}{=} \forall a. \forall b. \text{testOut}(m, a) \triangleright \text{duo}(n, b) \triangleright \\ &\quad \diamond \diamond (a^\dagger \wedge b^\dagger \wedge (\text{atom}(a) \blacktriangleright \text{atom}(b) \blacktriangleright \diamond \diamond (\neg a^\dagger \wedge \neg b^\dagger \wedge \mathcal{A}))) \end{aligned}$$

Lemma 6 (Modality formulas). *The formulas above have the following interpretation, when $\text{size}(P) < 2 * k - 1$:*

$$- P \models \langle mn \rangle^{(k)} . \mathcal{A} \text{ iff } P \xrightarrow{mn} P' \text{ and } P' \models \mathcal{A} \text{ for some } P'.$$

– $P \models \langle \overline{m}\langle n \rangle \rangle^{(k)}. \mathcal{A}$ iff $P \xrightarrow{\overline{m}\langle n \rangle} P'$ and $P' \models \mathcal{A}$ for some P' .

Proof (sketch). In the case of $\langle mn \rangle^{(k)}. \mathcal{A}$, we put the candidate process in presence of a process bisimilar to $\overline{n}\langle m \rangle. \overline{a}\langle a \rangle | a(x)$, for a fresh. The remainder of the formula specifies that after two steps of reduction, the process must not exhibit a barb on a and must satisfy the continuation formula \mathcal{A} . This is possible only if a communication on a has happened, preceded by the output at n .

The formula for the free output modality follows similar ideas, the tester process being more complex due to the necessity to recognise two names in the prefix that is triggered. \square

Remark 1. An important property of our constructions is that at the end of the ‘experiment’, no garbage process is left, so that we can go on with the satisfaction of \mathcal{A} . This allows us to avoid using $|$ in formulas like is done e.g. in [HLS03].

Remark 2 (Bound output modality). Although we have no formal proof for this, we believe that we cannot define a formula for the bound output modality in general in \mathcal{L} . Intuitively, the reason is that in order to define a formula $\langle \overline{n}\langle m \rangle \rangle. \mathcal{A}$, we should be able to impose satisfaction of \mathcal{A} under the restriction binding the extruded name, which is not possible (see also the extensions of \mathcal{L} in 5.2).

However, we can observe the ability for a process to perform a bound output:

Lemma 7. *Given P , n and k such that $\text{size}(P) < 2*k - 1$, there exists a formula $\langle \overline{n}\langle - \rangle \rangle^{(k)}$ such that $P \models \langle \overline{n}\langle - \rangle \rangle^{(k)}$ iff $P \xrightarrow{\overline{n}\langle m \rangle} P'$ for some m and P' .*

For lack of space, we do not present the proof of this result. The main idea is to express the fact that there is no way for a process coming from ‘outside the tested process’ to interact on the name received at n . We use for this some formulas whose interpretation contain a form of universal quantification on names (to give an idea, this is the case for example for formula **atom** introduced in 3.1).

The formulas given by Lemma 6 allow us to derive the following result.

Theorem 2 (Characterising public processes). *For any public process P , there exists a \mathcal{L} -formula F_P such that for any process Q , $Q \models F_P$ iff $P \sim Q$.*

Proof (Sketch). We exploit the characterisation of \sim in Proposition 1, as well as Lemma 3 to simplify the formulas we manipulate. We define F_P by induction over the size of the transition system for \sim_s generated by P , using nil for the bisimilarity class of $\mathbf{0}$. According to the result given in Proposition 1, we first pick n fresh names a_1, \dots, a_n , where n is the number of input prefixes in P , and define $\mathcal{N} = \text{fn}(P) \cup \{a_1, \dots, a_n\}$. We define \mathcal{M} as the following set of actions:

$$\mathcal{M} \stackrel{\text{def}}{=} \{mn, m \in \mathcal{N}, n \in \mathcal{N}\} \cup \{\overline{m}\langle n \rangle, m \in \mathcal{N}, n \in \mathcal{N}\}.$$

We also set $\text{Ac}(P) = \{\mu. \exists P'. P \xrightarrow{\mu} P'\}$, and, for a free input or output action μ , $P|_{\mu} = \{P'. P \xrightarrow{\mu} P'\}$. We then define:

$$F_P \stackrel{\text{def}}{=} \mathcal{I}a_1 \dots \mathcal{I}a_n. \bigwedge_{\mu \in \text{Ac}(P)} [\mu]^{(k)}. \left(\bigvee_{P' \in P|_{\mu}} F_{P'} \right) \wedge \bigwedge_{\mu \in \mathcal{M} \setminus \text{Ac}(P)} \neg \langle \mu \rangle^{(k)}. \top,$$

where $[\mu]^{(k)}.\mathcal{A}$ stands for $\neg\langle\mu\rangle^{(k)}.\neg\mathcal{A}$ and k satisfies $2 * k - 1 > \text{size}(P)$.

The construction of F_P is rather standard, and consists in describing the transitions a state can make by expressing the possible actions and their continuations as well as those actions that cannot be performed. \square

Remark 3. The characteristic formulas we define are valid for the whole calculus, and in particular they are also satisfied by all processes with restriction that are bisimilar to a public process. The logic also allows us to characterise some processes outside this class, as illustrated by the following formula

$$\langle\bar{a}(-)\rangle \ \wedge \ \forall r. (F_{a(x).x(y).\bar{r}(y)} \triangleright \diamond\bar{\diamond}\bar{r}(b)),$$

which captures the processes bisimilar to $(\nu c)(\bar{a}(c)|\bar{c}(b))$ (this is the case e.g. for $(\nu c)\bar{a}(c).\bar{c}(b)$). We have not been able to define characteristic formulas for the whole calculus, though, and do not believe that this would be feasible along the lines of the constructions presented above.

Remark 4 (On the role of \mathcal{I}). We could get rid of \mathcal{I} in the constructions we have presented. This is possible by defining a formula $\text{unactive}^{(k)}(n)$, that says that name n is not liable to be used in the first k interactions of a given process P provided n is not sent to P (we can easily express (dis)equality of names in \mathcal{L}). Note that this property is different from being a fresh name for P , as a process bisimilar to $\mathbf{0}$ can have free occurrences of names. Intuitively, to obtain extensionality (Theorem 4 below), we only need to be able to pick enough ‘unactive names’ to build the characteristic formula for a given public process. Without having checked formally that this is the case, we do believe that our main result can be proved in a logic without \mathcal{I} .

4 Extensionality

We now show that $=_{\mathcal{L}}$ coincides with \sim .

Theorem 3 (Behavioural implies logical). $P \sim Q$ implies $P =_{\mathcal{L}} Q$.

Proof (Sketch). We prove by structural induction on \mathcal{A} that whenever $P \sim Q$ and $P \models \mathcal{A}$, we have $Q \models \mathcal{A}$. The cases corresponding to the adjunct operators \triangleright and \odot follow from congruence properties of \sim w.r.t. parallel composition and restriction, respectively (see [SW01]). \square

Lemma 8 (Characteristic formulas for barbed bisimilarity). *Given a process P and a finite set of names \mathcal{N} such that $\text{fn}(P) \subseteq \mathcal{N}$, there exists a formula $\mathbb{B}_{\mathcal{N},P}$ such that for any Q such that $\text{fn}(Q) \subseteq \mathcal{N}$, $Q \models \mathbb{B}_{\mathcal{N},P}$ iff $P \simeq Q$.*

Proof. The formula is defined by induction on the size of P as follows:

$$\begin{aligned} \mathbb{B}_{\mathcal{N},P} \stackrel{\text{def}}{=} & \bigwedge_{n \in \mathcal{N}. P \downarrow_n} \downarrow_n \ \wedge \ \bigwedge_{n \in \mathcal{N}. P \downarrow_{\bar{n}}} \downarrow_{\bar{n}} \ \wedge \ \bigwedge_{n \in \mathcal{N}. P \downarrow_n} \neg \downarrow_n \ \wedge \ \bigwedge_{n \in \mathcal{N}. P \downarrow_{\bar{n}}} \neg \downarrow_{\bar{n}} \\ & \wedge \ \bigwedge_{P' \in \{P'. P \longrightarrow P'\}_{/ \equiv}} \diamond \mathbb{B}_{\mathcal{N},P'}. \end{aligned}$$

In this formula, $\mathcal{S}/_{\equiv}$ stands for the quotient of \mathcal{S} modulo \equiv . The above conjunction is finite because \mathcal{N} is finite and reduction is image-finite up to \equiv ([SW01]). The definition is well-formed because P' is smaller (in the sense of size) than P in the recursive calls.

By definition of \simeq , $Q \models \mathbb{B}_{\mathcal{N},P}$ iff $P \simeq Q$, as long as all free names of Q are inspected by formula $\mathbb{B}_{\mathcal{N},P}$, which is guaranteed by the condition $\text{fn}(Q) \subseteq \mathcal{N}$. \square

Theorem 4 (Logical implies behavioural). $P =_{\mathcal{L}} Q$ implies $P \sim Q$.

Proof. Suppose $P \not\sim Q$; by Theorem 1, $P \not\sim Q$, i.e., $P|R \not\sim Q|R$ for some R .

Write using Lemma 2 $R \sim (\nu \tilde{n})R_0$, where R_0 is public and $\tilde{n} \cap \text{fn}(P) = \tilde{n} \cap \text{fn}(Q) = \emptyset$. We have that $\sim \subseteq \simeq$, and hence $(\nu \tilde{n})(P|R_0) \not\sim (\nu \tilde{n})(Q|R_0)$.

Take $\mathcal{N} = \text{fn}(P) \cup \text{fn}(Q) \cup \text{fn}(R)$, and define $\mathcal{A} \stackrel{\text{def}}{=} \mathbb{B}_{\mathcal{N},(\nu \tilde{n})(P|R_0)} \odot \tilde{n}$. Observe that we have $P \models \mathbb{F}_{R_0} \triangleright \mathcal{A}$. Suppose now $Q \models \mathbb{F}_{R_0} \triangleright \mathcal{A}$, this means that for all R_1 such that $R_0 \sim R_1$, $(\nu \tilde{n})(Q|R_1) \models \mathcal{A}$, which entails by Lemma 8 that $(\nu \tilde{n})(P|R_0) \simeq (\nu \tilde{n})(Q|R_1)$. $R_0 \sim R_1$ implies $(\nu \tilde{n})(Q|R_1) \sim (\nu \tilde{n})(Q|R_0)$, and hence, since $\sim \subseteq \simeq$, and by transitivity of \simeq , we obtain $(\nu \tilde{n})(P|R_0) \simeq (\nu \tilde{n})(Q|R_0)$, a contradiction. So $Q \not\models \mathbb{F}_{R_0} \triangleright \mathcal{A}$, and finally $P \neq_{\mathcal{L}} Q$. \square

Note that the proof above exploits the two presentations of \sim : characteristic formulas for public processes are derived using the labelled transition system, while the overall structure of the proof follows the definition of \simeq .

5 Variants and Extensions of \mathcal{L}

5.1 Changing the Primitive Observation

The most tedious constructions in Section 3 are the formulas to detect barbs (cf. Theorem 5). We consider here a variant of \mathcal{L} , called \mathcal{L}^\dagger , in which we remove \odot and add a primitive formula n^\dagger , whose satisfaction is defined by $P \models n^\dagger$ iff $(P \downarrow_n$ or $P \downarrow_{\bar{n}})$. With respect to \mathcal{L} , \mathcal{L}^\dagger allows one to build less contexts, while providing the ability to perform instantaneous observations independently from the size of the tested process.

We first remark that the only place where \odot is used in the formulas presented in Section 3 is in the definition of $\text{atom}(n)$, that can be rewritten in \mathcal{L}^\dagger as follows:

$$\text{atom}(n) \stackrel{\text{def}}{=} \text{atom} \wedge n^\dagger.$$

To show that logic \mathcal{L}^\dagger induces an equivalence that also coincides with \sim , the completeness proof of Section 4 has to be adapted. The main point is to observe that testing for \simeq against *public* processes is enough to get the same discriminative power as \simeq , as expressed by the following lemma.

Lemma 9. *Define $P \simeq_p Q$ iff for any R public, $P|R \simeq Q|R$. Then $\simeq_p = \simeq$.*

This result allows us to replay the proof of Theorem 4 in the case of \mathcal{L}^\dagger without using \odot , and we have:

Theorem 5. *Two processes are logically equivalent for \mathcal{L}^\dagger iff they are bisimilar.*

5.2 Enriching \mathcal{L} with Intensional Operators

Our main result, given by Theorems 3 and 4, isolates the extensional subset of spatial logics for concurrency. To explore the spectrum between extensional and intensional spatial logics, we now consider enrichments of \mathcal{L} with (some of the) intensional operators 0 , $|$ and \mathbb{R} .

Observing emptiness. Formula 0 gives us an elementary form of observation: in $\mathcal{L} \cup \{0\}$ (using an obvious notation), we can detect garbage, in the sense that we can use 0 to separate for example processes $\mathbf{0}$ and $(\nu n)\bar{n}\langle m \rangle.P$ (which are bisimilar). As a consequence, we can define characteristic formulas for ‘minimal-size public processes’, i.e., characteristic formulas up to \sim that are satisfied only by public processes, by using 0 instead of nil in the constructions of Section 3.

Separating. In $\mathcal{L} \cup \{| \}$, we get a finer equivalence than \sim . In particular, we have:

Lemma 10. *In $\mathcal{L} \cup \{| \}$, logical equivalence on public terms coincides with \equiv .*

On the full calculus, it would be interesting to study the relationship with *distributed bisimulation* [CH89], a behavioural equivalence that is able to separate for example $\bar{m}\langle n \rangle.\bar{m}\langle n \rangle$ and $\bar{m}\langle n \rangle | \bar{m}\langle n \rangle$. In presence of restriction, though, the definition of distributed bisimulation is rather complex, even for CCS [Kie89].

Revealing names. The results in [HLS03] show that revelation brings a lot of expressiveness to the logic. In $\mathcal{L} \cup \{\mathbb{R}\}$, formula 0 is derivable, as well as a formula to test the free occurrence of a name at any depth in a process:

$$0 \stackrel{\text{def}}{=} \text{nil} \wedge \neg \mathcal{I}a. a\mathbb{R}\neg\text{nil} \qquad \text{free}(n) \stackrel{\text{def}}{=} \neg n\mathbb{R}\top$$

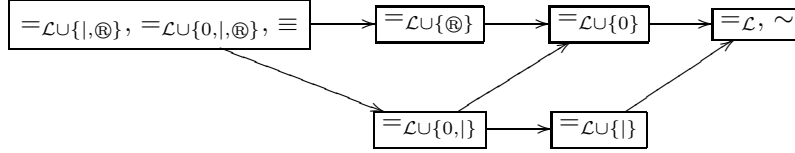
(the second formula is from [CG01b] — note that in the static case, $\text{free}(n)$ can be defined using only \odot [CG04]). Having the ability to observe under restrictions, we can define a modality formula for bound output:

Lemma 11. *In $\mathcal{L} \cup \{\mathbb{R}\}$, given a formula \mathcal{A} and a process P such that $\text{size}(P) < 2 * k - 1$, there exists a formula $\langle \bar{m}\langle n \rangle \rangle^{(k)}. \mathcal{A}$ such that $P \models \langle \bar{m}\langle n \rangle \rangle^{(k)}. \mathcal{A}$ iff $P \xrightarrow{\bar{m}\langle n \rangle} P'$ and $P' \models \mathcal{A}$ for some P' .*

We believe that characteristic formulas up to \sim for the whole calculus are definable in this enriched logic. \mathbb{R} actually gives us a greater precision. Indeed, the induced logical equivalence is rather fine-grained, but we have not been able to provide a precise characterisation of it. For example, $(\nu c) \bar{a}\langle c \rangle.\bar{c}\langle b \rangle$ can be separated from $(\nu c) (\bar{a}\langle c \rangle | \bar{c}\langle b \rangle)$, while $\bar{m}\langle n \rangle.\bar{m}\langle n \rangle$ and $\bar{m}\langle n \rangle | \bar{m}\langle n \rangle$ are equivalent in $\mathcal{L} \cup \{\mathbb{R}\}$.

Combining intensional observations. In $\mathcal{L} \cup \{0, |\}$, on the other hand, the logic separates the two latter processes while equating the first two. Finally, logic $\mathcal{L} \cup \{|\, \textcircled{R}\}$ is intensional: logical equivalence coincides with \equiv . This can be shown by adapting the proofs in [HLS02] using the constructions of the present paper.

The following graph sums up the observations made above. Vertices contain relations, and two relations situated on the same vertex coincide ($=_{\mathbb{L}}$ stands for the equality induced by the logic \mathbb{L}). An arrow between two edges represents strict inclusion between the corresponding relations, and unrelated vertices correspond to incomparable relations. More detailed explanations are given in [Hir04]



6 Conclusion

We have defined a spatial logic \mathcal{L} , and shown that the induced logical equivalence coincides with bisimilarity. We can remark that while HM logic and \mathcal{L} induce the same equivalence on processes, model-checking in \mathcal{L} seems a priori much more difficult, due to the presence of \triangleright . We can also remark that \mathcal{L} has a restricted set of operators, and is minimal in the sense of [Loz03]: getting rid of a connective of \mathcal{L} hinders the expressive and separative powers of the logic.

The observations provided in \mathcal{L}^\dagger suggest that this logic is to barbed equivalence what Hennessy-Milner logic is to bisimilarity. The constructions in 3.2 show that, to some extent, \mathcal{L} has the ability to express the observations of \mathcal{L}^\dagger . We do not see how the converse could hold, i.e. how hiding could be expressed within \mathcal{L}^\dagger . At least we believe there is no compositional encoding that could translate a formula of the form $\mathcal{A} \otimes n$ into \mathcal{L}^\dagger . A perhaps more interesting question would be to find out whether \otimes is eliminable in the logic resulting from the ‘union’ of \mathcal{L} and \mathcal{L}^\dagger , along the lines of adjunct elimination in [Loz03,Loz04,DGG04].

In contrast with existing spatial logics, that all include intensional operators, satisfaction in \mathcal{L} is defined with no direct reference to structural congruence (along these lines, the clause defining satisfaction for \triangleright in \mathcal{L} can be seen as specifying composition of behaviours). It would be interesting to look for a way to recover some of the separating power of existing spatial logics while keeping logical equivalence close to some existing equivalence. In another direction, we would like to see whether we can combine the ideas presented here with the constructions defined in [CL04], which would mean studying $\mathcal{L} \setminus \{\otimes\}$, a contextual logic with no reference to names. Logical equivalence is up to name permutation in the name-free logic of [CL04]: we believe that logical equivalence in $\mathcal{L} \setminus \{\otimes\}$ is bisimilarity up to name permutation.

Finally, our techniques do not apply directly if we adopt a weak interpretation for \diamond (which we use here to count, in some way): studying logical equivalence in a ‘weak version’ of \mathcal{L} represents a challenging question.

Acknowledgements. We would like to thank Étienne Lozes, Davide Sangiorgi and Luís Caires for inspiring discussions about the results presented in this paper. This work has been supported by european FET - Global Computing project PROFUNDIS and by the french ACI GEOCAL.

References

- [CC01] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). In *Proc. of TACS'01*, LNCS. Springer Verlag, 2001.
- [CG00] L. Cardelli and A. Gordon. Anytime, Anywhere, Modal Logics for Mobile Ambients. In *Proc. of POPL'00*, pages 365–377. ACM Press, 2000.
- [CG01a] L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In *Proc. of ESOP'01*, volume 2028 of LNCS, pages 1–22. Springer Verlag, 2001. invited paper.
- [CG01b] L. Cardelli and A. Gordon. Logical Properties of Name Restriction. In *Proc. of TLCA'01*, volume 2044 of LNCS. Springer Verlag, 2001.
- [CG04] G. Conforti and G. Ghelli. Decidability of Freshness, Undecidability of Revelation. In *Proc. of FOSSACS'04*, volume 2987 of LNCS, pages 105–120. Springer Verlag, 2004.
- [CH89] I. Castellani and M. Hennessy. Distributed Bisimulations. *J. ACM*, 36(4):887–911, 1989.
- [CL04] L. Caires and E. Lozes. Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency. this volume, 2004.
- [DGG04] A. Dawar, P. Gardner, and G. Ghelli. Games for the Ambient Logic. draft, 2004.
- [DLM04] S. Dal Zilio, D. Lugiez, and C. Meyssonnier. A Logic You Can Count on. In *Proc. of POPL 2004*. ACM Press, 2004.
- [Hir04] D. Hirschhoff. An Extensional Spatial Logic for Mobile Processes. Technical report, LIP - ENS Lyon, 2004. to appear.
- [HLS02] D. Hirschhoff, E. Lozes, and D. Sangiorgi. Separability, Expressiveness and Decidability in the Ambient Logic. In *Proc. of LICS'02*, pages 423–432. IEEE Computer Society, 2002.
- [HLS03] D. Hirschhoff, E. Lozes, and D. Sangiorgi. Minimality Results for the Spatial Logics. In *Proc. of FSTTCS'03*, volume 2914 of LNCS, pages 252–264. Springer Verlag, 2003.
- [Kie89] A. Kiehn. Distributed Bisimulations for Finite CCS. Technical Report 7/89, University of Sussex, 1989.
- [Loz03] E. Lozes. Adjunct Elimination in the Static Ambient Logic. In *Proc. of EXPRESS'03*, ENTCS. Elsevier, 2003.
- [Loz04] E. Lozes. Separation logic preserves the expressiveness of classical logic. In *Proc. of SPACE'04*, 2004.
- [MPW93] Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993.
- [Rey02] J. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proc. of LICS'02*. IEEE Computer Society, 2002.
- [San01] D. Sangiorgi. Extensionality and Intensionality of the Ambient Logic. In *Proc. of 28th POPL*, pages 4–17. ACM Press, 2001.
- [SW01] D. Sangiorgi and D. Walker. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.