# Partial Order Reduction for Verification of Spatial Properties of Pi-Calculus Processes

Reynald Affeldt [a,1] Naoki Kobayashi [b,2]

[a] *Department of Computer Science, University of Tokyo, Tokyo, Japan*

[b] *Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan*

**Abstract**

Mechanical tools have recently been developed that enable computer-aided verification of spatial properties of concurrent systems. To be practical, these tools are expected to deal with the state-space explosion problem. In order to alleviate this problem, we develop partial order reduction for verification of spatial properties of pi-calculus processes. The main issue is that spatial logics are very expressive and some spatial formulas prevent partial order reduction. After discussing this issue, we propose a restricted spatial logic such that partial order reduction holds. Our approach relies on exploiting partially confluent communications and on identifying invisible communications in the pi-calculus, for which we propose a simple syntactic criterion.

*Key words:* Partial order reduction, spatial logics, pi-calculus

## 1 Introduction

Spatial logics [3,4,5,6] have been drawing much attention as specification languages for concurrent systems. They can express, among others, properties of structure of concurrent systems, for example whether or not a concurrent system is composed of two or more identifiable subsystems.

Recently, efforts have been made to construct tools for computer-aided verification of spatial-logic specifications of concurrent systems. Vieira and Caires have been developing a model checker for automatic verification of finite-control concurrent systems written in a nominal $\pi$-calculus and specified using a rich spatial logic [16]. The authors of the present paper have been developing a library for interactive verification of concurrent systems written

[1] Email: affeldt@yl.is.s.u-tokyo.ac.jp

[2] Email: kobayasi@kb.cs.titech.ac.jp

in an applied version of the $\pi$-calculus using a spatial logic [1]. Like all verification tools for concurrent systems, these tools must deal with the state-space explosion problem.

In this paper, we study how to alleviate the state-space explosion problem for computer-aided verification of spatial logic specifications of concurrent programs. Our approach is to enable partial order reduction by exploiting partial confluence properties of $\pi$-calculus processes. We briefly explain the basic idea of partial order reduction. Let us consider some satisfaction relation $\models$ between the states of some reduction system and some set of formulas. The basic idea of partial order reduction is to exploit reductions $P \rightarrow P'$ such that, for some formula $\phi$, $P \models \phi \Leftrightarrow P' \models \phi$. In such situations, in order to verify whether $P \models \phi$, one can choose to perform the reduction $P \rightarrow P'$ (even if there are other possible reductions) and check whether $P' \models \phi$. In this paper, our goal is to find appropriate conditions for the formula $\phi$ and the reduction $P \rightarrow P'$ in the case where $\phi$ is a formula of the spatial logic and $P \rightarrow P'$ is a reduction of the $\pi$-calculus. The same question has already been addressed for usual temporal logics such as LTL and CTL* (for Kripke structures), but not for spatial logics. In particular, the existence of expressive spatial formulas makes this question difficult. In addition, the target language being the $\pi$-calculus also makes it non-trivial to find an appropriate syntactic condition for $P \rightarrow P'$. In usual model checkers like Spin, $P \rightarrow P'$ is just a transition caused by access to a local variable [10], but in the $\pi$-calculus, all the computations are communications.

Our contributions can be summarized as follows:

(i) We identify a set of spatial formulas that prevent partial order reduction.

(ii) We define a syntactic notion of invisible communication and we introduce a restricted spatial logic such that invisible communications cannot be observed by the set of spatial formulas.

(iii) We show that invisibility and partial confluence of communications is a sufficient criterion to enable sound partial order reduction for this spatial logic.

**Outline** In Sect. 2, we show informally with an example that the knowledge of partially confluent communications enables partial order reduction for verification of spatial properties of $\pi$-calculus processes. In Sect. 3, we discuss spatial formulas that prevent partial order reduction. In Sect. 4, we introduce the *TSL* logic, we define partially confluent and invisible communications and state our main theorem, namely that partially confluent invisible communications enable partial order reduction. In Sect. 5, we outline the proof of the main theorem. In Sect. 6, we discuss potential extensions to the *TSL* logic.

**Notation and Vocabulary** We use a version of the $\pi$-calculus [13] standard enough not to require a detailed introduction. For the sake of completeness, we give the complete description in Appendix A. It can be skipped on a first reading provided the reader is aware of the following points: (1) we use a subset of the $\pi$-calculus where the choice operator is removed and the replication operator is restricted to input processes, (2) we use a reduction operational semantics.

## 2 Motivating Example

We are interested in verifying $\pi$-calculus processes against spatial formulas. In this section, we show informally that partially confluent communications simplify such verifications.

Let us consider the following $\pi$-calculus process:

$$P = \underbrace{\overline{c}.\overline{e}.f|c}_{\text{process } C} \mid \underbrace{\overline{d}.e.f|d}_{\text{process } D}$$

(In this paper, we write $\overline{c}.P$ instead of $\overline{c}\langle v \rangle.P$ when the name $v$ is not relevant, similarly we write $c.P$ instead of $c(x).P$ when the name $x$ does not appear in $P$.)

The process $P$ consists of the parallel composition of several (sub)processes. The process $\overline{c}.\overline{e}.f$ is ready to send some data along the name $c$ (intuitively, a channel of communication) and will behave as the process $\overline{e}.f$ after emission. The process $c$ waits for input along the name $c$. The name $c$ is only known of the process $C$; the name $d$ is only known of the process $D$; both process $C$ and process $D$ share the names $e$ and $f$: they use $e$ to perform a hand-shake and will both seek access to some resource available along name $f$.

Let us assume that we want to verify that there is no race condition along the name $f$ (this is actually wrong). Put formally, we want to verify that there is no execution such that the spatial formula $\phi = f()|f()$ is eventually true (a process satisfies $\phi_1|\phi_2$ if it consists of a process satisfying $\phi_1$ and a process satisfying $\phi_2$, a process satisfies $f()$[3] if it consists of an input process that waits for data along the name $f$). The motivation for such a verification may be for instance that the resource along $f$ expects processes $C$ and $D$ to perform inputs in some predetermined order.

Naive verification of $P$ leads to the exhaustive enumeration of all execution paths, and in general this approach is impractical because it leads to the state-space explosion problem.

In comparison, the knowledge of partially confluent communications enables efficient verification. Informally, a communication is partially confluent when it commutes with all other communications. Because the communication along $c$ in our example is partially confluent, the state-space of $P$ can be represented as follows (we represent communication with arrows and annotate

[3] The formula $f()$ is actually an abbreviation for the formula $f()\top$ defined later.

them with the name used for communication):

$$\begin{tikzcd}
& P = \overline{c}.\overline{e}.f|c|\overline{d}.e.f|d \arrow[dl, "c"'] \arrow[dr, "d"] & \\
P' = \overline{e}.f|\overline{d}.e.f|d \arrow[dr, "d"'] & & P'' = \overline{c}.\overline{e}.f|c|e.f \arrow[dl, "c"] \\
& Q = \overline{e}.f|e.f \arrow[d, "e"] & \\
& f|f &
\end{tikzcd}$$

Since both possible execution paths lead to the same state, it is intuitively obvious that the verification of $P$ can be reduced to the verification of, say, $P'$. Although this is true for the verification of the formula $\phi$, this is wrong for the verification of the formula $c()\top \wedge e()\top$ (where the conjunction has its usual meaning) because the latter actually holds of $P''$.

In this paper, we will show for a spatial logic that the partial order reduction above is valid as long as the communication along $c$ is (1) partially confluent and (2) invisible (intuitively, the same spatial formulas hold of $P$ and $P'$, and $P''$ and $Q$).

## 3 Partial Order Reduction and Spatial Formulas

For LTL and CTL*, partial order reduction is sound for the fragment without the "next" formula (see for instance [7]). So, a natural question is: Is partial order reduction sound for the standard spatial logic without the "next" formula? The answer is no: as discussed below, there are many other formulas of the spatial logic that prevent partial order reduction.

**Problem with the Zero Formula** The zero formula of spatial logics is defined as follows:

$$P \models 0 \text{ iff } P \equiv 0$$

Using the zero formula, it is possible to write formulas to count the number of non-zero subprocesses (this is observed for instance in [9]). For example, formulas below hold respectively of processes with one, two, or three non-zero subprocesses:

$$\begin{aligned}
\mathbf{1} &\overset{def}{=} \neg 0 \wedge \neg(\neg 0|\neg 0) \\
\mathbf{2} &\overset{def}{=} (\neg 0|\neg 0) \wedge \neg(\neg 0|\neg 0|\neg 0) \\
\mathbf{3} &\overset{def}{=} (\neg 0|\neg 0|\neg 0) \wedge \neg(\neg 0|\neg 0|\neg 0|\neg 0)
\end{aligned}$$

These formulas prevent partial order reduction. For instance, the problem of verifying $P$ cannot be reduced to the problem of verifying $P'$ in the following example because it would let us conclude that **3** must be eventually true:

$$\begin{tikzcd}
 & P = c|\overline{c}.\overline{e}|d|\overline{d} \not\models \mathbf{3} \arrow[dl, "c"'] \arrow[dr, "d"] & \\
P' = \overline{e}|d|\overline{d} \models \mathbf{3} \arrow[dr, "d"'] & & c|\overline{c}.\overline{e} \not\models \mathbf{3} \arrow[dl, "c"] \\
 & \overline{e} \not\models \mathbf{3} &
\end{tikzcd}$$

**Problem with the Input/Output Formulas** There are several alternative definitions for input/output formulas (see [2,3], or [6] where the formula for ambient locations can be compared with the output formula). For instance:

$$\begin{aligned}
P &\models c()\phi \quad \text{iff} \quad P \equiv c(x).Q \text{ and for all } v,\ Q\{v/x\} \models \phi \\
P &\models \overline{c}\langle v\rangle\phi \quad \text{iff} \quad P \equiv \overline{c}\langle v\rangle.Q \text{ and } Q \models \phi
\end{aligned}$$

These definitions are problematic because they can be used to implicitly test for the absence of actions. For example, they prevent partial order reduction for the following verification ($\top$ is a formula true of any process):

$$\begin{tikzcd}
 & \overline{c}|c|\overline{d}|d \not\models \overline{d}\langle\rangle\top|d()\top \arrow[dl, "c"'] \arrow[dr, "d"] & \\
\overline{d}|d \models \overline{d}\langle\rangle\top|d()\top \arrow[dr, "d"'] & & \overline{c}|c \not\models \overline{d}\langle\rangle\top|d()\top \arrow[dl, "c"] \\
 & 0 \not\models \overline{d}\langle\rangle\top|d()\top &
\end{tikzcd}$$

**Problem with the Temporal Modality** To compensate for the loss of expressiveness due to the removal of the "next" temporal modality, we can replace it by its weak version (also defined in [9]):

$$P \models \Diamond\phi \text{ iff there exists } P' \text{ such that } P \rightarrow^* P' \text{ and } P' \models \phi$$

There is still a problem: mixed use of this temporal modality and the composition formula of spatial logics. For example, partial order reduction is not sound for the following process:

$$\begin{tikzcd}
d.\,((c.\overline{e}|\overline{c}) \mid (c.\overline{e}|\overline{c})) \mid \overline{d} \not\models \Diamond\overline{e}\langle\rangle\top|\Diamond\overline{e}\langle\rangle\top \arrow[d, "d"] \\
(c.\overline{e}|\overline{c}) \mid (c.\overline{e}|\overline{c}) \models \Diamond\overline{e}\langle\rangle\top|\Diamond\overline{e}\langle\rangle\top
\end{tikzcd}$$

# 4 The *TSL* Logic and Partial Order Reduction

As we have seen in the previous section, partial order reduction is not sound for the full spatial logic. In this section, we first introduce a restricted fragment (which we call the *TSL* logic) of the spatial logic, for which partial order reduction will be shown to hold.

We first introduce the *TSL* logic. We then formally define partial confluence and invisible communications. We finally state our main theorem, namely that partially confluent invisible communications enable partial order reduction.

## *4.1 The TSL Logic*

The *TSL* logic is a spatial logic, restricted in such a way that it enables partial order reduction. Its definition takes into account the issues discussed in the previous section: there is no zero formula, the semantics of input/output formulas is defined appropriately, the usual temporal modality is replaced with its weak version (noted EF instead of $\Diamond$), arbitrary mixing of spatial and temporal formulas is prevented by distinguishing state and temporal formulas.

The *TSL* logic consists of state formulas and temporal formulas:

**Definition 4.1** The set of *state formulas* (noted *SL*) is given by the following grammar:

$$\phi ::= \underbrace{\top \mid \neg\phi \mid \phi_1 \vee \phi_2}_{\text{propositional formulas}} \mid \underbrace{c()\phi \mid \overline{c}\langle\rangle\phi \mid (\phi_1|\phi_2)}_{\text{spatial formulas}}$$

Its semantics is defined as follows ($\nu y_{1,\dots,n}$ is an abbreviation for $\nu y_1.\cdots.\nu y_n$):

$P \models \top$ always true

$P \models \neg\phi$ iff not $P \models \phi$

$P \models \phi_1 \vee \phi_2$ iff $P \models \phi_1$ or $P \models \phi_2$

$P \models c()\phi$ iff there exist $T, R, y_1, \dots, y_n$ such that $P \equiv \nu y_{1,\dots,n}.(c(x).T|R)$ with $c \notin y_1, \dots, y_n$ and for all $v$ such that $v \notin y_1, \dots, y_n$, $\nu y_{1,\dots,n}.(T\{v/x\}|R) \models \phi$

$P \models \overline{c}\langle\rangle\phi$ iff there exist $U, R, y_1, \dots, y_n, v$ such that $P \equiv \nu y_{1,\dots,n}.(\overline{c}\langle v\rangle.U|R)$ with $c \notin y_1, \dots, y_n$ and $\nu y_{1,\dots,n}.(U|R) \models \phi$

$P \models \phi_1|\phi_2$ iff there exist $R_1, R_2, y_1, \dots, y_n$ such that $P \equiv \nu y_{1,\dots,n}.(R_1|R_2)$ with $\nu y_{1,\dots,n}.R_1 \models \phi_1$ and $\nu y_{1,\dots,n}.R_2 \models \phi_2$

**Lemma 4.2 (Structural congruence preserves state-formulas validity)** *Let $P$ and $Q$ be two processes such that $P \equiv Q$. Then, for any state formula $\phi$, we have $P \models \phi \Leftrightarrow Q \models \phi$.*

**Definition 4.3** Let $S$ be a set of state formulas (ranged over by $\phi$) whose satisfaction relation is noted $\models$. The set of *temporal formulas* for $S$ (noted $T(S)$) is given by the following grammar:

$$f ::= \phi \mid \neg^t f \mid f_1 \vee^t f_2 \mid \text{EF } f \mid \text{AF } f$$

Its semantics is defined as follows. A *path* is a possibly infinite sequence of processes such that each process is obtained by a communication from the previous one. A path if *full* either if it is infinite, or if it is finite and the last process cannot be reduced. We write $p_i$ for the $i$th process of a path $p$.

$$\begin{array}{lll}
P \models^t \phi & \text{iff} & P \models \phi \\
P \models^t \neg^t f & \text{iff} & \text{not } P \models^t f \\
P \models^t f_1 \vee^t f_2 & \text{iff} & P \models^t f_1 \text{ or } P \models^t f_2 \\
P \models^t \text{EF } f & \text{iff} & \text{there exists a path } p \text{ such that } p_1 = P \text{ and } p_k \models^t f \text{ for some } k \\
P \models^t \text{AF } f & \text{iff} & \text{for any full path } p \text{ such that } p_1 = P,\ p_k \models^t f \text{ for some } k
\end{array}$$

**Definition 4.4** The *TSL* logic is the logic defined by $T(SL)$.

### *4.2 Partial Confluence and Invisible Communications*

In the following, we decorate $\pi$-calculus reductions with labels. A labeled reduction is written $P \xrightarrow{l,S} Q$ where $l$ is a name or a special label $\epsilon$ and $S$ is a set of names. Informally, $l$ is the name used for communication or the special label $\epsilon$ for internal communication, and names in $S$ are the names that are revealed by the communication (labels are ranged over by $\alpha, \beta$, see Appendix A for a formal definition). For instance:

$$c|\overline{c}.(d.\overline{e}\langle v\rangle|\overline{c}) \xrightarrow{c,\{c,d\}} d.\overline{e}\langle v\rangle|\overline{c}$$

We first formally define partial confluence.

**Definition 4.5** The set of *partially confluent* communications is the largest set $S$ such that for any $(P, \alpha, Q) \in S$ we have:

(i) $P \xrightarrow{\alpha} Q$, and

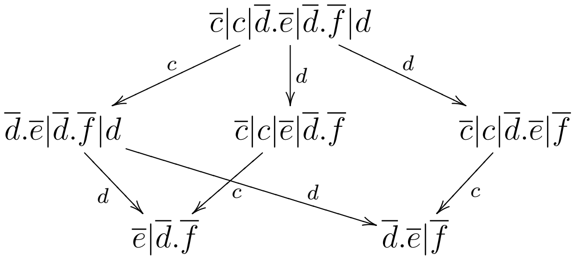
(ii) if $P \to P'$, then either:

  (a) $Q \equiv P'$, or

  (b) there exists $Q'$ such that $Q \to Q'$ and $(P', \alpha, Q') \in S$.

For example, the communication along $c$ in the process $P = \overline{c}|c|\overline{d}.\overline{e}|\overline{d}.\overline{f}|d$

is partially confluent because we have the state-space below:


$\overline{c}|c|\overline{d}.\overline{e}|\overline{d}.\overline{f}|d$
c
d
d
$\overline{d}.\overline{e}|\overline{d}.\overline{f}|d$
$\overline{c}|c|\overline{e}|\overline{d}.\overline{f}$
$\overline{c}|c|\overline{d}.\overline{e}|\overline{f}$
d
c
d
c
$\overline{e}|\overline{d}.\overline{f}$
$\overline{d}.\overline{e}|\overline{f}$

The property of partial confluence is for instance enjoyed by linearized names [12] and $\omega$-receptive names [15].

We now define the notion of invisible communication. For any formula $f$, let $fn(f)$ be the set of names in $f$.

**Definition 4.6** For any state formula $\phi$, the communication $P \xrightarrow{c,S} P'$ is *invisible* with respect to $\phi$ if $(\{c\} \cup S) \cap fn(\phi) = \emptyset$.

For instance, the communication $c|\overline{c}.(d.\overline{e}\langle v\rangle|\overline{c}) \xrightarrow{c,\{c,d\}} d.\overline{e}\langle v\rangle|\overline{c}$ is invisible with respect to the formula $\overline{e}\langle\rangle\top$ because $fn(\overline{e}\langle\rangle\top) = \{e\}$ and $\{c,d\}\cap\{e\} = \emptyset$.

We conclude this section with an important lemma stating when invisible communications cannot be observed by the state formulas. Note that this lemma holds thanks to the restrictions on the state formulas; as discussed in Sect. 3, it is not true for spatial logics in general.

**Lemma 4.7 (Invisible Communications cannot be Observed)** *Let $\phi$ be a state formula. If $P \xrightarrow{c,S} P'$ is invisible with respect to $\phi$, then we have $P \models \phi \Leftrightarrow P' \models \phi$.*

**Proof.** See Appendix B. □

### 4.3 Main Theorem

We finally state our main theorem, namely that partially confluent invisible communications enable partial order reduction:

**Theorem 4.8 (Partial Order Reduction)** *Let the communication $P \xrightarrow{\alpha} Q$ be partially confluent and invisible with respect to a set $S$ of state formulas. Then, for any temporal formula $f \in T(S)$, we have $Q \models^t f \Leftrightarrow P \models^t f$.*

**Proof.** Obtained directly as a corollary of Lemma 5.3 by using Lemma 5.2. See next section. □

We show how to use this theorem on an example. Let us consider the following process:

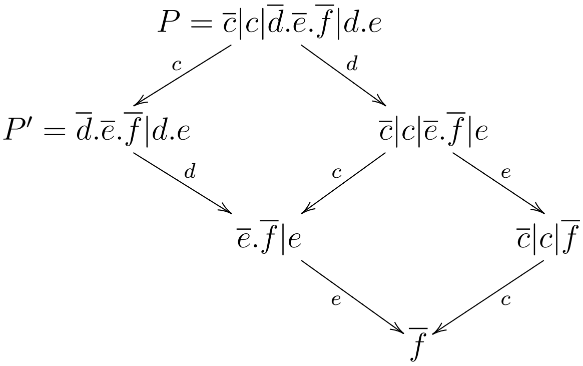
$$P = \overline{c}|c|\overline{d}.\overline{e}.\overline{f}|d.e$$

Let us assume that we want to verify that $P$ must eventually perform some output along the name $f$. Put formally, we want to verify whether $P \models^t \text{ AF } \overline{f}\langle\rangle\top$ holds.

Since the communication along name $c$ is partially confluent and invisible with respect to the formula $\overline{f}\langle\rangle\top$, by the theorem above, this verification is equivalent to the verification of $P' \models^t \text{AF } \overline{f}\langle\rangle\top$ with:

$$P' = \overline{d}.\overline{e}.\overline{f}|d.e$$

The latter verification is simpler because $P'$ is deterministic. This simplification is better appreciated by examining the state-space of $P$:


$P = \overline{c}|c|\overline{d}.\overline{e}.\overline{f}|d.e$
$c$
$d$
$P' = \overline{d}.\overline{e}.\overline{f}|d.e$
$\overline{c}|c|\overline{e}.\overline{f}|e$
$d$
$c$
$e$
$\overline{e}.\overline{f}|e$
$\overline{c}|c|\overline{f}$
$e$
$c$
$\overline{f}$

We observe that the theorem allows us to restrict verification to the path $P, P', \ldots$

## 5 Proof of the Main Theorem

We first introduce the notion of $S$-preserving bisimulation, where $S$ is a set of state formulas (as defined in Sect. 4.1). We then show that if $P \xrightarrow{\alpha} Q$ is partially confluent and invisible with respect to the formulas in $S$, then $P$ and $Q$ are $S$-preserving bisimilar (Lemma 5.2). Finally, we show that $S$-preserving bisimilar processes satisfy the same temporal formulas in $T(S)$ (as defined in Sect. 4.1) (Lemma 5.3). The main theorem (Theorem 4.8) is an immediate corollary of those lemmas.

**Definition 5.1** Let $S$ be a set of state formulas. A binary relation $R$ on processes is an *$S$-preserving bisimulation* if whenever $(P, Q) \in R$:

(i) if $P \rightarrow P'$, then there exist $Q_1, ..., Q_n$ $(n \geq 1)$ and $i$ $(1 \leq i \leq n)$ such that $Q = Q_1 \rightarrow \ldots \rightarrow Q_n$ and $(P, Q_1), \ldots, (P, Q_{i-1}), (P', Q_i), \ldots, (P', Q_n) \in R$,

(ii) if $Q \rightarrow Q'$, then there exist $P_1, \ldots, P_n$ $(n \geq 1)$ and $i$ $(1 \leq i \leq n)$ such that $P = P_1 \rightarrow \ldots \rightarrow P_n$ and $(P_1, Q), \ldots, (P_{i-1}, Q), (P_i, Q'), \ldots, (P_n, Q') \in R$, and

(iii) for any state formula $\phi \in S$, $P \models \phi$ if and only if $Q \models \phi$.

$P$ and $Q$ are *$S$-bisimilar*, written $P \approx_S Q$, if $(P, Q) \in R$ for some $S$-preserving bisimulation $R$.

**Lemma 5.2** *If $P \xrightarrow{\alpha} Q$ is an invisible communication with respect to a set $S$ of state formulas and if it is a partially confluent communication, then $P \approx_S Q$.*

**Proof.** Consider the relation $R \overset{def}{=} \equiv \cup \{(P, Q) \mid P \xrightarrow{\alpha} Q\}$. We show that $R$ is an $S$-preserving bisimulation. Consider $(P, Q) \in \{(P, Q) \mid P \xrightarrow{\alpha} Q\}$.

- Suppose that $P \longrightarrow P'$. Since $P \xrightarrow{\alpha} Q$ is partially confluent, then either (1) $P' \equiv Q$, in which case we can take $Q_2 = Q$, and $(P, Q), (P', Q_2) \in R$, or (2) there exists $Q'$ such that $Q \longrightarrow Q'$ and $P' \xrightarrow{\alpha} Q'$, in which case we can take $Q_2 = Q'$, and $(P, Q), (P', Q_2) \in R$.
- Suppose that $Q \longrightarrow Q'$. We have $P \xrightarrow{\alpha} Q \longrightarrow Q'$. Therefore we can take $P_2 = Q$ and $P_3 = Q'$, and $(P, Q), (P_2, Q), (P_3, Q') \in R$.
- Let $\phi$ be a state formula in $S$. Since $P \xrightarrow{\alpha} Q$ is an invisible communication with respect to the set $S$ of state formulas, by Lemma 4.7, we know that $P \models \phi$ if and only if $Q \models \phi$.

Thus, $P \approx_S Q$. □

**Lemma 5.3** *If $P \approx_S Q$, then, for any temporal formula $f \in T(S)$, we have $P \models^t f \Leftrightarrow Q \models^t f$.*

**Proof.** By induction on $f$:

- Case $f = \phi$ where $\phi \in S$, $f = \neg^t f'$, $f = f_1 \vee^t f_2$ are immediate.
- Case $f = \mathrm{EF}\ f'$.

  Case $\Rightarrow$. (The case $\Leftarrow$ is similar.) By assumption, there exists a path $p$ such that $p_1 = P$ and there exists $k$ such that $p_k \models^t f'$. Since $P \approx_S Q$, there exists a path $q$ such that $q_1 = Q$ and an index $j$ such that $p_k \approx_S q_j$. By the inductive hypothesis, $q_j \models^t f'$, which implies $Q \models^t \mathrm{EF}\ f'$.
- Case $f = \mathrm{AF}\ f'$.

  Case $\Rightarrow$. (The case $\Leftarrow$ is similar.) Let $q$ be a full path such that $q_1 = Q$. Since $P \approx_S Q$, there exists a full path $p$ such that $p_1 = P$ and such that for any $i$, there exists $j_i$ such that $p_i \approx_S q_{j_i}$. Since $P \models^t \mathrm{AF}\ f'$, there exists $k$ such that $p_k \models^t f'$. By the induction hypothesis, $q_{j_k} \models^t f'$. Thus, we have $Q \models^t \mathrm{AF}\ f'$.

□

## 6 Extensions

In this section, we discuss potential extensions to the *TSL* logic.

We can alleviate the restriction that prevents mixing of temporal and spatial formulas in *TSL*. As discussed in Sect. 3, we forbid arbitrary mixing of temporal and spatial formulas because it makes it difficult to define a satisfactory definition of invisibility. In the light of our development, it appears that we can allow a restricted form of mixing where temporal formulas can appear in composition formulas as long as they are guarded by temporal modalities and do not contain negation symbols, thus extending the set of temporal formulas as follows:

$$f ::= \ldots \mid \mathrm{EF}(f_1|f_2) \text{ where } f_1, f_2 \text{ do not contain negation symbols}$$

We conjecture that our approach can be applied to *TSL* extended with fairness conditions. This extension amounts to restrict path quantifiers to *fair paths* in the semantics of temporal formulas. A path is fair if there is no communication that is infinitely often enabled but never performed (this corresponds to the definition of *strong fairness*). For instance, we can add the following formula to *TSL*:

$$\begin{aligned} P \models^t \mathrm{AF}_{\mathit{fair}} f \quad &\text{iff} \quad \text{for any fair full path } p \text{ such that } p_1 = P, \\ &\quad\;\;\, \text{there exists } k \text{ such that } p_k \models^t f \end{aligned}$$

However, this extension requires refinement of the definition of label and of the definition of partial confluence. These refinements call for special care in the definition of the operational semantics of the $\pi$-calculus (similar to the development in [11]).

It is trivial to extend *TSL* with the adjunct of the composition formula defined as follows:

$$P \models \phi_1 \triangleright \phi_2 \text{ iff for any } Q \text{ such that } Q \models \phi_1, P|Q \models \phi_2 \text{ holds}$$

(where $\phi_0$ and $\phi_1$ are state formulas) but not useful. If temporal formulas are allowed to be used with the adjunct of composition, then the extension of *TSL* would require a change of the above definition. For illustration, consider the process $P = c.\overline{d}.\overline{e}|\overline{c}.d$. Provided we allow temporal formulas to be used with the adjunct, we have $P \models \top \triangleright \mathrm{AF}\ \overline{e}\langle\rangle\top$. Even though the communication along $c$ is partially confluent, this is no longer true if we compose $P$ with, say, $\overline{c}$.

## 7 Conclusion

In this paper, we introduced an approach to partial order reduction for verification of spatial properties of $\pi$-calculus processes. First, we discussed some spatial formulas that prevent partial order reduction. Second, we defined the *TSL* logic, which is a restricted spatial logic, and a notion of invisible communication for the $\pi$-calculus. Then, we showed that partially confluent invisible communications enables partial order reduction in *TSL*. Finally, we discussed potential extensions to our approach.

**Related Work** Our work is directly related to partial order reduction techniques for model checking. Partial order reduction techniques have been defined for several temporal logics (LTL-X [7], CTL*-X [8], the weak modal mu-calculus [14]). The main originality of our work lies in the application to spatial formulas (that requires adequate restrictions) and the application to the $\pi$-calculus (that requires an appropriate definition of invisibility).

The problem of finding partially confluent communications has been addressed several times in the literature on the $\pi$-calculus (linear types and linearized types [12], linear receptiveness and $\omega$-receptiveness [15]). Our work can be seen as an application of this work.

**Future Work** We plan to extend our work to enable mixing of temporal formulas with the adjunct of the composition formula. We also plan to investigate extension to other spatial formulas, in particular the revelation formula and quantifiers.

We plan to formalize the results of the present paper in Coq for the sake of completeness of the library in [1].

### Acknowledgments

This work is partially supported by a research project funded by Japanese Ministry of Education and Science's research program "e-Society."

## References

[1] Reynald Affeldt and Naoki Kobayashi. A Coq library for verification of concurrent programs. In *4th International Workshop on Logical Frameworks and Meta-Languages (LFM 2004), Cork, Ireland, July 4, 2004*, pages 66–83. Informal proceedings available at http://www.cs.yale.edu/~carsten/lfm04.

[2] Luís Caires. Behavioral and spatial observations in a logic for the pi-calculus. In *7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2004), Barcelona, Spain, March 29–April 2, 2004*, volume 2987 of *Lecture Notes in Computer Science*. Springer, Mar. 2004.

[3] Luís Caires and Luca Cardelli. A spatial logic for concurrency (part I). In Naoki Kobayashi and Benjamin C. Pierce, editors, *Theoretical Aspects of Computer Software (TACS 2001), Sendai, Japan*, number 2215 in Lecture Notes in Computer Science, pages 1–37. Springer, Oct. 2001.

[4] Luís Caires and Luca Cardelli. A spatial logic for concurrency (part II). In Lubos Brim, Petr Jancar, Mojmir Kretinsky, and Antonin Kucera, editors, *13th International Conference on Concurrency Theory (CONCUR 2002), Brno, Czech Republic*, number 2421 in Lecture Notes in Computer Science, pages 209–225. Springer, Aug. 2002.

[5] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: modal logics for mobile ambients. In *27th ACM SIGPLAN-SIGACT symposium on principles of*

*programming languages (POPL 2000), Boston, Massachusetts, USA, January 19–21, 2000*, pages 365–377. ACM Press, 2000.

[6] Luca Cardelli and Andrew D. Gordon. Logical properties of name restriction. In *5th International Conference on Typed Lambda Calculi and Applications (TLCA 2001), Krakow, Poland, May 2–5, 2001*, volume 2044 of *Lecture Notes in Computer Science*, pages 46–60. Springer, May 2001.

[7] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking.* MIT Press, 2000.

[8] Rob Gerth, Ruurd Kuiper, Doron Peled, and Wojciech Penczek. A partial order approach to branching time logic model checking. *Information and Computation*, 150(2):132–152, 1999.

[9] Daniel Hirschkoff, Étienne Lozes, and Davide Sangiorgi. Minimality results for the spatial logics. In *23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2003), Mumbai, India*, volume 2914 of *Lecture Notes in Computer Science*, pages 252–264. Springer, Dec. 2003.

[10] Gerard J. Holzmann. *The SPIN Model Checker, Primer and Reference Manual.* Addison Wesley Professional, 2004.

[11] Naoki Kobayashi. A type system for lock-free processes. *Information and Computation*, 177(2):122–159, Sep. 2002.

[12] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the Pi-Calculus. In *23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1996), St. Petersburg Beach, Florida, January 21–24, 1996*, pages 358–371. ACM Press, 1996.

[13] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, Sep. 1992.

[14] Y. S. Ramakrishna and Scott A. Smolka. Partial-order reduction in the weak modal mu-calculus. In *8th International Conference on Concurrency Theory (CONCUR 1997), Warsaw, Poland*, volume 1243 of *Lecture Notes in Computer Science*, pages 5–24. Springer, Jul. 1997.

[15] Davide Sangiorgi. The name discipline of uniform receptiveness (extended abstract). In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *24th International Colloquium on Automata, Languages and Programming (ICALP 1997), Bologna, Italy, July 7-11, 1997*, volume 1256 of *Lecture Notes in Computer Science*, pages 303–313. Springer, Jul. 1997.

[16] Hugo Vieira and Luís Caires. Spatial logic model checker user's guide (version 0.9). Technical report, Departamento de Informática, FCT/UNL, Dec. 2003. Revised March 2004.

## A Syntax and Semantics for the $\pi$-calculus

**Syntax** The $\pi$-calculus consists of two syntactic entities: names and processes. Processes use names to interact and can pass names to one another during interactions. In this paper, names are ranged over by[4] $x, y, c, d, e, f, v$ and processes are ranged over by $P, Q, R, T, U$. The syntax of processes is given by the following grammar:

$$P ::= \overline{c}\langle v\rangle.P \mid c(x).P \mid P|Q \mid !c(x).P \mid \nu x.P \mid 0$$

The output process $\overline{c}\langle v\rangle.P$ can send the name $v$ along the name $c$ and then behave as $P$. The input process $c(x).P$ can receive some name, say $v$, along $c$, and then behave as $P\{v/x\}$, that represents the process $P$ in which all the free occurrences of the name $x$ have been replaced with the name $v$. Parallel composition $P|Q$ makes it possible for processes to interact. The replicated input $!c(x).P$ behaves as infinitely many input processes in parallel. The restriction $\nu x.P$ indicates that the scope of the name $x$ is restricted to $P$. The process 0 represents termination.

We define $fn(P)$ to be the set of free names in the process $P$. The prefix $\nu x_1.\cdots.\nu x_n$ is abbreviated $\nu x_{1,\ldots,n}$. The process $c_1(x).P_1|c_2(x).P_2|\cdots$ is abbreviated $\Pi_i c_i(x).P_i$, and similarly for outputs and replicated inputs. When the name $v$ in $\overline{c}\langle v\rangle.P$ (resp. the name $x$ in $c(x).P$) is not relevant, we write instead $\overline{c}.P$ (resp. $c.P$). We omit trailing zeros; for instance, we write $\overline{c}$ instead of $\overline{c}.0$.

**Structural Congruence** Structural congruence relates pairs of processes that only differ by spatial rearrangements. It facilitates the definition of the operational semantics of the $\pi$-calculus and is at the basis of the definition of spatial formulas. Two processes $P$ and $Q$ are structurally congruent when $P \equiv Q$ can be inferred from the following rules:

| | | | |
|---|---|---|---|
| $P \equiv P\|0$ | zero | $P \equiv P$ | refl |
| $P\|Q \equiv Q\|P$ | comm | $P \equiv Q \Rightarrow Q \equiv P$ | sym |
| $P\|(Q\|R) \equiv (P\|Q)\|R$ | assoc | $P \equiv Q \land Q \equiv R \Rightarrow P \equiv R$ | trans |
| $\nu x.\nu y.P \equiv \nu y.\nu x.P$ | swap | $!c(x).P \equiv !c(x).P \mid c(x).P$ | rep |
| $\nu x.0 \equiv 0$ | reszero | $!c(x).P \equiv !c(x).P \mid !c(x).P$ | rep2 |
| $\nu x.(P\|Q) \equiv P\|\nu x.Q \; (x \notin fn(P))$ | extrusion | | |

**Operational Semantics** The operational semantics of the $\pi$-calculus is defined inductively by the following rules. The use of labels and the use of

[4] We tend to use $x, y$ for names used as binders, $c, d, e, f$ for names used as channels of communication, and $v$ for names used as transmitted values.

the $\mathsf{guards}$ predicate in the rule for communication are not standard. In this paper, labels are ranged over by $\alpha, \beta$.

$$\frac{\mathsf{guards}(P|Q\{v/x\}) = S}{\overline{c}\langle v\rangle.P|c(x).Q \xrightarrow{\{c,S\}} P|Q\{v/x\}}\ \mathsf{com}$$

$$\frac{P \xrightarrow{\alpha} Q}{\nu x.P \xrightarrow{\alpha\backslash x} \nu x.Q}\ \mathsf{res} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ \mathsf{par} \qquad \frac{Q \xrightarrow{\alpha} Q' \quad P \equiv Q \quad P' \equiv Q'}{P \xrightarrow{\alpha} P'}\ \mathsf{struct}$$

where $\mathsf{guards}$ is defined inductively as follows:

$$\begin{array}{ll} \mathsf{guards}(\overline{c}\langle v\rangle.P) = \{c\} & \mathsf{guards}(P|Q) = \mathsf{guards}(P) \cup \mathsf{guards}(Q) \\ \mathsf{guards}(c(x).P) = \{c\} & \mathsf{guards}(\nu x.P) = \mathsf{guards}(P) - \{x\} \\ \mathsf{guards}(!c(x).P) = \{c\} & \mathsf{guards}(0) = \emptyset \end{array}$$

and $\alpha\backslash x$ is defined as follows:

$$\begin{aligned} (y,S)\backslash x &= \begin{cases} (y, S - \{x\}) \text{ if } y \neq x \\ (\epsilon, S - \{x\}) \text{ if } y = x \end{cases} \\ (\epsilon, S)\backslash x &= (\epsilon, S - \{x\}) \end{aligned}$$

where $\epsilon$ is a special label for denoting an internal name.

We write $P \longrightarrow Q$ when $P \xrightarrow{\alpha} Q$ and $\alpha$ is not relevant. We write $\longrightarrow^+$ for the transitive closure of $\longrightarrow$, and $\longrightarrow^*$ for the reflexive transitive closure of $\longrightarrow$.

## B Proof of Lemma 4.7

We first prove an intermediate lemma. This intermediate lemma shows that, if a state formula $\phi$ holds of a process, then we can remove input/output subprocesses whose input/output name is not a free name of $\phi$ without affecting validity.

We introduce the $\mathsf{rem\_inout}$ function that removes the input/output processes that use certain names for input/output. Formally:

$$\begin{array}{lll}
\multicolumn{3}{l}{\mathsf{rem\_inout}\,(S,P) = \text{Case } P \text{ of}} \\
\quad \mid \overline{c}\langle v\rangle.Q & \rightarrow & \text{if } c \in S \text{ then } 0 \text{ else } P \\
\quad \mid c(x).Q & \rightarrow & \text{if } c \in S \text{ then } 0 \text{ else } P \\
\quad \mid !c(x).Q & \rightarrow & \text{if } c \in S \text{ then } 0 \text{ else } P \\
\quad \mid P|Q & \rightarrow & \mathsf{rem\_inout}\,(S,P) \mid \mathsf{rem\_inout}\,(S,Q) \\
\quad \mid \nu x.P & \rightarrow & \nu x.(\mathsf{rem\_inout}\,(S,P)) \text{ assuming } x \notin S \text{ by convention of bound names} \\
\quad \mid \_ & \rightarrow & P
\end{array}$$

**Lemma B.1 (Removal of in/out-processes preserves structural congruence)** *Let $P$ and $Q$ be two processes such that $P \equiv Q$. For any set of names $S$, we have $\mathsf{rem\_inout}\,(S,P) \equiv \mathsf{rem\_inout}\,(S,Q)$.*

**Lemma B.2 (Removal of in/out-processes inversion)** *For any processes $P$ and $Q$ such that $\mathsf{rem\_inout}\,(\{c\},P) = \nu y_{1,\ldots,n}.Q$, there exist processes $U_i, T_j$ and names $v_i$ such that $P \equiv \nu y_{1,\ldots,n}.(Q \mid \Pi_i \overline{c}\langle v_i\rangle.U_i \mid \Pi_j c(x).T_j \mid \Pi_k !c(x).T_k)$.*

**Lemma B.3 (Removal of unrelated in/out-processes preserves validity)** *Let $P$ be a process and $\phi$ be a state formula. For any name $c$ such that $c \notin fn(\phi)$, we have $P \models \phi \Leftrightarrow \mathsf{rem\_inout}\,(\{c\},P) \models \phi$.*

**Proof.** By induction on $\phi$:

- Cases $\phi = \top$, $\phi = \neg\phi'$, $\phi = \phi_1 \vee \phi_2$ are immediate.
- Case $\phi = d()\phi'$. (The case for $\phi = \overline{d}\langle\rangle\phi'$ is similar.)

  Case $\Rightarrow$. By assumption, there exist $T, R, y_1, \ldots, y_n$ such that $P \equiv \nu y_{1,\ldots,n}.(d(x).T|R)$ with $d \notin y_1, \ldots, y_n$, and for all $v$ such that $v \notin y_1, \ldots, y_n$, we have $\nu y_{1,\ldots,n}.(T\{v/x\}|R) \models \phi'$. Since $c \notin fn(\phi)$, we have $c \neq d$. Therefore, there exists $R'$ such that $\mathsf{rem\_inout}\,(\{c\},P) \equiv \nu y_{1,\ldots,n}.(d(x).T|R')$. Since $c \notin fn(\phi)$, we have $c \notin fn(\phi')$. Therefore, for all $v$ such that $v \notin y_1, \ldots, y_n$, we have $\mathsf{rem\_inout}\,(\{c\}, \nu y_{1,\ldots,n}.(T\{v/x\}|R)) \models \phi'$, by the inductive hypothesis. We conclude that $\mathsf{rem\_inout}\,(\{c\},P) \models d()\phi'$.

  Case $\Leftarrow$. By assumption, there exist $T, R, y_1, \ldots, y_n$ such that $\mathsf{rem\_inout}\,(\{c\},P) \equiv \nu y_{1,\ldots,n}.(d(x).T|R)$ with $d \notin y_1, \ldots, y_n$, and for all $v$ such that $v \notin y_1, \ldots, y_n$, we have $\nu y_{1,\ldots,n}.(T\{v/x\}|R) \models \phi'$. Since $c \notin fn(\phi)$, we have $c \neq d$. By Lemma B.2, there exists $R'$ such that $P \equiv \nu y_{1,\ldots,n}.(d(x).T|R')$. Since $c \notin fn(\phi)$, we have $c \notin fn(\phi')$. Therefore, by the inductive hypothesis, for all $v$ such that $v \notin y_1, \ldots, y_n$, we have $\nu y_{1,\ldots,n}.(T\{v/x\}|R') \models \phi'$. We conclude that $P \models d()\phi'$.
- Case $\phi = \phi_1|\phi_2$.

  Case $\Rightarrow$. By assumption, there exist $P_1, P_2$ such that $P \equiv \nu y_{1,\ldots,n}.(P_1|P_2)$ with $\nu y_{1,\ldots,n}.P_1 \models \phi_1$ and $\nu y_{1,\ldots,n}.P_2 \models \phi_2$. By Lemma B.1, we know that

$\mathsf{rem\_inout}\,(\{c\}, P) \models \phi_1|\phi_2$ is equivalent to $\mathsf{rem\_inout}\,(\{c\}, \nu y_{1,\ldots,n}.(P_1|P_2)) \models \phi_1|\phi_2$. By definition of $\mathsf{rem\_inout}$, we have $\mathsf{rem\_inout}\,(\{c\}, \nu y_{1,\ldots,n}.(P_1|P_2)) = \nu y_{1,\ldots,n}.(\mathsf{rem\_inout}\,(\{c\}, P_1)\,|\mathsf{rem\_inout}\,(\{c\}, P_2))$. By the inductive hypothesis, $\nu y_{1,\ldots,n}.(\mathsf{rem\_inout}\,(\{c\}, P_1)) \models \phi_1$ and $\nu y_{1,\ldots,n}.(\mathsf{rem\_inout}\,(\{c\}, P_2)) \models \phi_2$. Therefore, $\mathsf{rem\_inout}\,(\{c\}, P) \models \phi_1|\phi_2$.

Case $\Leftarrow$. By assumption, there exist $P_1, P_2$ such that $\mathsf{rem\_inout}\,(\{c\}, P) \equiv \nu y_{1,\ldots,n}.(P_1|P_2)$ with $\nu y_{1,\ldots,n}.P_1 \models \phi_1$ and $\nu y_{1,\ldots,n}.P_2 \models \phi_2$. By Lemma B.2, there exist processes $U_i, T_j$ and names $v_i$ such that $P \equiv \nu y_{1,\ldots,n}.(P_1 \mid P_2 \mid \Pi_i \bar{c}\langle v_i\rangle.U_i \mid \Pi_j c(x).T_j \mid \Pi_k !c(x).T_k)$. Since we have $\nu y_{1,\ldots,n}.(\mathsf{rem\_inout}\,(\{c\}, P_1 \mid \Pi_i \bar{c}\langle v_i\rangle.U_i \mid \Pi_j c(x).T_j \mid \Pi_k !c(x).T_k)) \models \phi_1$, then we also have $\nu y_{1,\ldots,n}.(P_1 \mid \Pi_i \bar{c}\langle v_i\rangle.U_i \mid \Pi_j c(x).T_j \mid \Pi_k !c(x).T_k) \models \phi_1$ by the inductive hypothesis. Therefore $P \models \phi_1|\phi_2$.

□

Proof of Lemma 4.7:

**Proof.** By Lemma B.3, we have $P \models \phi \Leftrightarrow \mathsf{rem\_inout}\,((\{c\} \cup S), P) \models \phi$, and $P' \models \phi \Leftrightarrow \mathsf{rem\_inout}\,((\{c\} \cup S), P') \models \phi$. We conclude by observing that $\mathsf{rem\_inout}\,((\{c\} \cup S), P) = \mathsf{rem\_inout}\,((\{c\} \cup S), P')$ because $P \xrightarrow{c,S} P'$. □