Bisimulations in the Join-Calculus *

Michele Boreale † Cédric Fournet ‡ Cosimo Laneve §

October 27, 1997

Abstract

We develop a theory of bisimulation in the join-calculus. We introduce a refined operational model that makes interactions with the environment explicit, and we discuss the impact of the lexical discipline of the join-calculus on its extensional semantics. We propose several formulations of bisimulation and we establish that all formulations yield the same equivalence. We prove that this equivalence is finer than barbed congruence, but that both relations coincide in the presence of name-testing.

*This work is partly supported by the ESPRIT CONFER-2 WG-21836

†Dipartimento di Scienze dell'Informazione, Università "La Sapienza," Via Salaria 113, 00198 Roma, Italy

‡INRIA Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France.

§Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy

Contents

1 Introduction

The join-calculus is a recent formalism for modeling mobile systems [4, 5]. Its main motivation is to relate two crucial issues in concurrency: distributed implementation and formal semantics. To this end, the join-calculus enforces a strict lexical discipline over the channel names that appear in processes: names can be sent and received, but their communication capabilities cannot be affected by the receivers. This is the *locality property*.

Locality yields a simpler distributed model, because the communication primitives of the calculus can be directly implemented via standard primitives of asynchronous systems [5, 7]. It also plays a prominent role in the design of implicit type systems for the join-calculus, because all contravariant occurrences of names are static [6]. In this paper, we show that locality strongly affects the treatment of bisimulation, and provides a semantics simpler than other proposals in the literature.

In order to reason about distributed processes, we must equip the join-calculus with semantics that have both a sensible discriminating power and some convenient proof techniques. The usual approach in concurrency is to isolate the process from its environment and to focus on their interaction, whose simplest form is the message.

Formally, this is achieved through the definition of two notions that are common to most process calculi: a *reduction relation* $\rightarrow$ that represents internal evolution and an *observation predicate* $\Downarrow_a$ that detects the ability of interacting at a given channel. Based on these two notions, numerous observational semantics can then be defined. As regards discriminating power, it would be adequate to test the observation predicate under all possible contexts; indeed, the resulting *trace-based semantics* are relevant to many practical issues in distributed programming, especially as one is not interested in the branching structure of processes. In order to establish trace-based equivalences, however, one must cope with quantification over both contexts and traces, which makes the proofs particularly difficult [10].

To tackle these problems, it is convenient to consider behavioral equivalences that are finer and easier to check, possibly using them as indirect proof techniques for coarser semantics. Among these, bisimulation-based semantics [12] are especially convenient, because co-inductive proofs need only to consider single reduction steps instead of traces. *Barbed bisimilarity* has been proposed in [15] as a uniform basis to define sensible behavioral equivalences on different process calculi. This is the approach chosen for the join calculus in [4, 5], where *barbed congruence* is defined as the coarsest congruence that is a barbed bisimulation. Yet, checking that two processes are barbed congruent still requires explicit quantification over all contexts. Indeed, most proofs of [4] have to analyze in details how contexts and processes get intertwined, and to partition their reductions accordingly. This makes routine or automated checking of these properties problematic.

In this paper, we introduce a new operational model for the join calculus, and we equip it with the standard *weak bisimulation* [12], which enjoys the usual, purely co-inductive proof technique on labeled transitions. Then, we exhibit numerous sensible equalities between join-calculus processes that are contained in weak bisimulation, and we consider alternate definitions of this equivalence. Finally we relate weak bisimulation and barbed congruence and establish that the difference between the two is precisely due to the absence of a name-matching operator. A more precise account of our work follows.

The original semantics of the join-calculus is based on the *reflexive chemical abstract machine* (RCHAM) [4], where only internal evolution of processes is taken into account. In order to shape labeled bisimulation upon join-calculus, we first propose the open RCHAM, a refined operational semantics that explicitly models interaction with the environment. Via these interactions, the environment can get acquainted with local names of the process that are emitted at free names — the *extrusions* — and can later emit on these local names — the *intrusions* ("intrusion" has another technical meaning in [14]). Due to the principle of locality, intrusions and extrusions occur on disjoint sets of names. To keep track of the environment's acquaintance with local names, definitions — the technical device for defining the computational steps of RCHAM's— are marked with extruded names as soon as these extrusions happen. In turn, intrusions are allowed on these extruded names. The resulting open syntax for the join-calculus has processes of the form $\mathsf{def}_S\ D\ \mathsf{in}\ P$, where S is a set of names defined by D and extruded to the environment. This also induces a refined chemical machine, called the open RCHAM.

Let us illustrate what locality means in our setting. The process $\mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}$ $x\langle v\rangle$ defines a name x such that, whenever a message $x\langle u\rangle$ is received, a fresh copy of process P is started. Notice that x is marked as extruded; the environment can therefore send messages on x to trigger copies of process P. However, the environment cannot interfere with the definition of x; in particular, the message $x\langle v\rangle$ is invisible from the environment, and a sensible equivalence $\approx$ would identify the two processes that follow.

$$\mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}\ x\langle v\rangle \quad \approx \quad \mathsf{def}_{\{x\}}\ x\langle u\rangle \triangleright P\ \mathsf{in}\ P\{^v/_u\}$$

Weak bisimulation is obtained by applying the standard definition to the open RCHAM. Weak bisimilarity is stable under renaming, is a congruence over contexts of the open calculus, and it suitably abstracts from the actual structure of the join-patterns. For instance, it is insensitive to buffering, as is expected for an asynchronous semantics. Unfortunately, the open RCHAM allows intrusions that do not affect the behavior of the process. While only intrusions on extruded names need to be considered, this is unsatisfactory: the receiving capabilities for all the extruded names are syntactically available from the join-patterns of the process. Yet, weak bisimulation does not take them into account.

To reduce the size of our model, we therefore alter the open RCHAM and equip it with an alternative equivalence called *asynchronous bisimulation*. The new chemical machine restricts intrusions to sets of messages that immediately trigger a reaction rule; this technical device significantly augments the blocking capability of processes. Asynchronous bisimulation allows the intrusion of several messages at the same time, instead of single ones. We prove that weak bisimulation and asynchronous bisimulation coincide, which validates the use of asynchronous bisimulation in proofs.

The last characterization of weak/asynchronous bisimulation is given in terms of barbed equivalence. The barbed congruence of [4] is strictly coarser than weak bisimulation, because contexts of barbed congruence have no internal name-testing capability, while labels of weak bisimulation separates distinct names that exhibit the same behavior. This is actually the only difference between the two equivalences. To establish this, we augment the join-calculus with a name-matching operator and we show that (a variant of) barbed congruence coincides with weak bisimulation over image-finite processes.

Beyond its use as a proof technique, our semantics yields a better understanding of the join-calculus, and provides a basis for comparing it with other calculi, which are usually equipped with weak bisimulations, and especially with the asynchronous π-calculus [3]. In these calculi, asynchrony means that message outputs have no continuation, and thus that there is no direct way to detect that a message has been received. Noticeably, the usual weak bisimulation of the π-calculus has too much discriminating power, and separates processes with the same behavior (e.g., $\mathsf{0} \not\approx x(u).\overline{x}\langle u\rangle$); several remedies are considered in [8, 1]. Our bisimulations for the join-calculus make use of similar accommodations, but they yield simpler semantics, mostly because locality constrains interaction with the environment. This prunes the number of transitions to consider, and rules out processes such as $x(u).\overline{x}\langle u\rangle$ where x is used by the environment both for input and output.

The rest of the paper is organized as follows. In Section 2 we introduce our language and its model. In Section 3 we define weak bisimulation in this setting, we give examples, and we study its basic properties. In Section 4 we introduce asynchronous bisimulation, and we prove that it coincides with weak bisimulation. In Section 5 we relate our bisimulations to barbed congruence, and study the impact of name-testing. In Section 7 we discuss related work for the asynchronous π-calculus. We conclude in Section 8.

2 The Open Join-Calculus

In this section, we define the open join-calculus and its operational semantics as extensions of the join-calculus and the RCHAM of [4].

2.1 Syntax and Scopes

Let $\mathcal{V}$ be a countable set of *names* ranged over by $x, y, u, v, \cdots$. Tuples of names are noted ${u_i}^{i \in 1..p}$ or simply $\widetilde{u}$. Names can carry fixed-length messages, and we assume a recursive sorting discipline on names that avoids arity mismatch, in the style of [13]. We consider only well-sorted terms, and we assume an infinite number of names of each sort. We omit the sorting discipline, as it is not important to understand the contents of the paper.

The open join-calculus processes A and the join-calculus processes P, the definitions

D, and the *join-patterns* J are the terms defined as follows:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
A & \stackrel{\text{def}}{=} & \mathsf{0} & P & \stackrel{\text{def}}{=} & \mathsf{0} \\
 & & x\langle {u_i}^{i\in 1..p}\rangle & & & x\langle {u_i}^{i\in 1..p}\rangle \\
 & & A \mid A & & & P \mid P \\
 & & \mathsf{def}_S\ D\ \mathsf{in}\ A & & & \mathsf{def}\ D\ \mathsf{in}\ P \\[1ex]
D & \stackrel{\text{def}}{=} & J \triangleright P & J & \stackrel{\text{def}}{=} & x\langle {u_i}^{i\in 1..p}\rangle \\
 & & D \wedge D & & & J \mid J
\end{array}
$$

A process P can be the inert process $\mathsf{0}$, a message sent on a name x that carries a tuple of names $\widetilde{u}$, a parallel composition of processes, or the definition of local names and the process within their scope; a definition D is a conjunction of reaction rules $J \triangleright P$ that associate join-patterns J to guarded processes P; the intended meaning is that, whenever messages match the pattern J, the guarded process P can be triggered. An open process A is like a process, except that it has open definitions at top-level instead of local ones. The open definition $\mathsf{def}_S\ D\ \mathsf{in}\ \ldots$ exhibits a subset S of names defined by D that are visible from the environment: the *extruded names*. When omitted, the index set S is empty (no extruded names so far). We identify such open definitions $\mathsf{def}_\emptyset\ D\ \mathsf{in}\ P$ with local definitions $\mathsf{def}\ D\ \mathsf{in}\ P$.

Definition 2.1 *We partition names in terms as follows:* visible names *are either* free names $\mathsf{fv}[A]$ *or* extruded names $\mathsf{xv}[A]$*;* local names *are either* received names $\mathsf{rv}[J]$ *or* defined names $\mathsf{dv}[J], \mathsf{dv}[D]$.

$$
\begin{array}{lrcl@{\qquad}rcl}
J: & \mathsf{rv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{u \in \widetilde{v}\} & \mathsf{dv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{x\} \\
 & \mathsf{rv}[J \mid J'] & \stackrel{\text{def}}{=} & \mathsf{rv}[J] \uplus \mathsf{rv}[J'] & \mathsf{dv}[J \mid J'] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] \uplus \mathsf{dv}[J'] \\[2ex]
D: & \mathsf{fv}[J \triangleright P] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] \cup (\mathsf{fv}[P] \backslash \mathsf{rv}[J]) & \mathsf{dv}[J \triangleright P] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] \\
 & \mathsf{fv}[D \wedge D'] & \stackrel{\text{def}}{=} & \mathsf{fv}[D] \cup \mathsf{fv}[D'] & \mathsf{dv}[D \wedge D'] & \stackrel{\text{def}}{=} & \mathsf{dv}[D] \cup \mathsf{dv}[D'] \\[2ex]
P, A: & \mathsf{fv}[\mathsf{0}] & \stackrel{\text{def}}{=} & \emptyset & \mathsf{xv}[\mathsf{0}] & \stackrel{\text{def}}{=} & \emptyset \\
 & \mathsf{fv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{x\} \cup \{u \in \widetilde{v}\} & \mathsf{xv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \emptyset \\
 & \mathsf{fv}[A \mid A'] & \stackrel{\text{def}}{=} & (\mathsf{fv}[A] \backslash \mathsf{xv}[A']) \cup (\mathsf{fv}[A'] \backslash \mathsf{xv}[A]) & \mathsf{xv}[A \mid A'] & \stackrel{\text{def}}{=} & \mathsf{xv}[A] \uplus \mathsf{xv}[A'] \\
 & \mathsf{fv}[\mathsf{def}_S\, D\ \mathsf{in}\, A] & \stackrel{\text{def}}{=} & (\mathsf{fv}[A] \cup \mathsf{fv}[D]) \backslash \mathsf{dv}[D] & \mathsf{xv}[\mathsf{def}_S\, D\ \mathsf{in}\, A] & \stackrel{\text{def}}{=} & S \uplus \mathsf{xv}[A]
\end{array}
$$

The above definition deserves further comments. In particular, note that the extruded names of two processes in parallel are required to be disjoint, because they are defined by different definitions. For the same reason, for every open subprocess $\mathsf{def}_S\ D\ \mathsf{in}\ A$, the extruded names S are disjoint from the extruded names in A ($S \cap \mathsf{xv}[A] = \emptyset$). Finally, to simplify the labels of asynchronous bisimulation (see Section 4), we constrain definitions to *linear* join patterns ($\mathsf{dv}[J] \cap \mathsf{dv}[J'] = \emptyset$). This constraint doesn't affect the expressivity of the calculus.

In this paper, all processes are considered modulo α-conversion on *bound names*, namely received names and defined non-extruded names. We also require that every open process be *well-formed*, i.e. that every nested open definition $\mathsf{def}_S\ D\ \mathsf{in}\ P$ define its extruded names ($S \subseteq \mathsf{dv}[D]$).

The *interface* of an open process A consists of two disjoint sets of names: free names $\mathsf{fv}[A]$ used in A to send messages out, and extruded names $\mathsf{xv}[A]$ used by the environment to send message in.

DEFINITION 2.2 *A* global renaming *on open processes substitutes free, extruded or fresh names for free names, and substitutes distinct fresh names for extruded names.*

2.2 Open chemistry

Let us introduce the operational semantics of the open join-calculus through an example. It will also explain the dynamic role played by the index S in a definition. Consider the process $\mathsf{def}_\emptyset\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ z\langle x\rangle$. The interface contains no extruded name so far, and two free names y, z. The message $z\langle x\rangle$ can be consumed by the environment, thus exporting x.

$$\mathsf{def}_\emptyset\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ z\langle x\rangle \quad \xrightarrow{\{x\}\overline{z}\langle x\rangle} \quad \mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ 0\ .$$

Once x is known from the environment, it cannot be considered local anymore — the environment can emit on x —, but is not free either — the environment cannot modify or extend its definition. A new transition is enabled:

$$\mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ 0 \quad \xrightarrow{x\langle\rangle} \quad \mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ x\langle\rangle$$

Now the process can input some more messages on x, and also perform the two transitions below to emit on y:

$$\begin{array}{lcl} \mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ x\langle\rangle & \longrightarrow & \mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ y\langle\rangle \\ & \xrightarrow{\{\}\overline{y}\langle\rangle} & \mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright y\langle\rangle\ \mathsf{in}\ 0 \end{array}$$

In order to model the interactions with the environment, we extend the RCHAM model defined in [4]. This is accomplished by means of two dual operations: *extrusion* of defined names on free names, and *intrusion* of messages sent on previously extruded names. The explicit bookkeeping of extruded names is performed by suitably augmenting the chemical solutions.

Formally, let the functions $\mathsf{dv}[\cdot], \mathsf{fv}[\cdot], \mathsf{xv}[\cdot]$ be extended to multisets of terms in the expected way. *Open chemical solutions*, ranged over by $\mathcal{S}, \mathcal{T}, \ldots$, are triples $(\mathcal{R}, S, \mathcal{A})$, written $\mathcal{R} \Vdash_S \mathcal{A}$, where

- $\mathcal{A}$ is a multiset of open processes with disjoint sets of extruded names.
- $\mathcal{R}$ is a multiset of definitions such that $\mathsf{dv}[\mathcal{R}] \cap \mathsf{xv}[\mathcal{A}] = \emptyset$
- $S \subseteq \mathsf{dv}[\mathcal{R}]$ is a set of extruded names.

(S-Nil)	$\Vdash \mathsf{0}$	$\rightleftharpoons$	$\Vdash$
(S-Par)	$\Vdash A \mid A'$	$\rightleftharpoons$	$\Vdash A,\, A'$
(S-And)	$D \wedge D \Vdash$	$\rightleftharpoons$	$D,\, D' \Vdash$
(S-Def)	$\Vdash_S \mathsf{def}_{S'}\, D \mathsf{\ in\ } A$	$\rightleftharpoons$	$D\sigma \Vdash_{S \uplus S'} A\sigma$
(Red)	$J \triangleright P \Vdash J\rho$	$\longrightarrow$	$J \triangleright P \Vdash P\rho$
(Ext)	$\Vdash_S x\langle v_i{}^{i\in 1..p}\rangle$	$\xrightarrow{S'\overline{x}\langle v_i{}^{i\in 1..p}\rangle}$	$\Vdash_{S \cup S'}$
(Int)	$\Vdash_{\{x\}\cup S}$	$\xrightarrow{x\langle v_i{}^{i\in 1..p}\rangle}$	$\Vdash_{\{x\}\cup S} x\langle v_i{}^{i\in 1..p}\rangle$

Side conditions on the reacting solution $\mathcal{S} = (\mathcal{R} \Vdash_S \mathcal{A})$:

(S-Def)	σ replaces $\mathsf{dv}[D] \setminus S'$ with distinct fresh names.
(Red)	ρ substitutes names for $\mathsf{rv}(J)$;
(Ext)	x is free, and $S' = \{v_i \mid i \in 1..p\} \cap (\mathsf{dv}[\mathcal{R}] \setminus S)$;
(Int)	$v_i^{i\in 1..p}$ are either free, or fresh, or extruded.

Table 1: The open RCHAM of the open join-calculus

The definitions of free and extruded names are lifted to solutions component-wise.

$$\begin{aligned}\mathsf{fv}[\mathcal{R} \Vdash_S \mathcal{A}] &\stackrel{\text{def}}{=} (\mathsf{fv}[\mathcal{R}] \cup (\mathsf{fv}[\mathcal{A}]) \setminus \mathsf{dv}[\mathcal{R}])\\ \mathsf{xv}[\mathcal{R} \Vdash_S \mathcal{A}] &\stackrel{\text{def}}{=} S \uplus \mathsf{xv}[\mathcal{A}]\end{aligned}$$

The restriction on extruded names is partially defined as

$$(\mathcal{R} \Vdash_{S \uplus S'} \mathcal{A}) \setminus S' \stackrel{\text{def}}{=} \mathcal{R} \Vdash_S \mathcal{A}$$

The chemical rules for the open RCHAM are given in Table 1; they define families of transitions $\rightharpoonup$, $\leftharpoondown$, $\rightarrow$, $\xrightarrow{\alpha}$, $\alpha \in \{S\overline{x}\langle\widetilde{v}\rangle, x\langle\widetilde{v}\rangle\}$ between open solutions. As usual, each chemical rule mentions only the processes and the definitions that are involved in it, and the transition applies to every chemical solution $\mathcal{S}$ whose multisets contain these processes and definitions.

Let us comment on the rules. The first five rules are those of the RCHAM. The structural rules that define the inverse transitions $\rightharpoonup$ and $\leftharpoondown$ are unchanged, except for Rule (S-Def) that performs some extra bookkeeping on extruded names. The reduction rule (Red) says when a message can trigger a process in a definition. In the sequel, $\rightleftharpoons$ is the symmetric relation $\rightharpoonup \cup \leftharpoondown$ that contains all structural moves, its transitive closure $\rightleftharpoons^*$ is called structural equivalence; $\Rrightarrow$ is the relation $\rightleftharpoons \cup \rightarrow$ that gathers all silent moves.

The two last rules model interaction with the context. The extrusion rule (Ext) consumes messages sent on free names; these messages can contain defined names not previously known to the environment, thus causing the scope of their definitions to be opened; this is made explicit in the set of extruded names S' that appears on the transition label. Our rule resembles very much the Open rule for restriction in the π-calculus [14]. There is an important restriction due to locality: messages are either emitted on free names, to be consumed by (EXT), or on names defined in the open solution, to be consumed by (RED).

The intrusion rule (INT) can be viewed as a disciplined version of one of the Input rule for the asynchronous π-calculus proposed by Honda and Tokoro [8]: the latter allows intrusion of arbitrary messages at any stage, whereas our rule allows intrusion of messages only on defined-extruded names.

NOTATION 2.3 *Let $\mathcal{S}$ be the open solution $\mathcal{R} \Vdash_S \mathcal{A}$ and $M = \prod x\langle\widetilde{v}\rangle$ be a parallel composition of messages that can be intruded using Rule* (INT)*. Then $\mathcal{S} \mid M$ is the open solution $\mathcal{R} \Vdash_S \mathcal{A}, M$.*

Formally, the original RCHAM of [4] is the restriction of our open RCHAM that operates on chemical solutions with no extruded variables, and that use every chemical rule except (EXT) and (INT). In particular, we retain the following property of the RCHAM: z

REMARK 2.4 *Every open chemical solution is structurally equivalent*

- *to a fully-heated solution that contains only simple reaction rules and messages (unique up-to alpha-conversion)*

$$\{\cdots J_j \triangleright P_j, \cdots\} \Vdash_S \{\cdots x_i\langle\widetilde{u_{ik}}\rangle, \cdots\}$$

- *and to the corresponding solution that contains a single, cooled-down open process*

$$\emptyset \Vdash_\emptyset \left\{\mathsf{def}_S \bigwedge J_j \triangleright P_j \text{ in} \prod x_i\langle\widetilde{u_{ik}}\rangle\right\}$$

When applied to open solutions, the structural rules capture the intended meaning of extruded names: messages sent on extruded names can be moved inside or outside of their defining process. Assuming x has been extruded ($x \in S \cup S' \cup \mathsf{xv}[A]$), and modulo α-conversion to avoid clashes with the arguments ($\{\widetilde{v}\} \cap \mathsf{dv}[D] \subseteq S'$), we have

$$\Vdash_S \; x\langle\widetilde{v}\rangle \mid \mathsf{def}_{S'} \; D \text{ in } A \quad \rightleftharpoons^* \quad \Vdash_S \; \mathsf{def}_{S'} \; D \text{ in } (x\langle\widetilde{v}\rangle \mid A)$$

REMARK 2.5 *In its original presentation [4], the join-calculus is equipped with an auxiliary labeled transition system. Its intent, however, is quite different from ours. It is used to give an alternative, syntactic description of chemical reduction steps. In contrast, our open* RCHAM *provides an extensional semantics of the calculus; the labels are largely independent of the definitions, and we do not attempt to define silent steps as matching labeled transitions plus hiding.*

2.3 Barbed Congruence

The semantics of the join-calculus in [4] is *barbed congruence*; this equivalence was initially studied for the π-calculus as "reduction-based equivalence" in [9]. We recall the definition, and refer to these papers for discussion.

DEFINITION 2.6 *The* output barb $\Downarrow_x$ *is a predicate over open* RCHAM*s that tests for the potential emission on the free name* x*:*

$$\mathcal{S} \Downarrow_x \quad \stackrel{\text{def}}{=} \quad \exists S, \widetilde{v} \text{ such that } \mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle}$$

A symmetric relation Φ *over open* RCHAM*s is a* barbed bisimulation *if whenever* $\mathcal{S}\ \Phi\ \mathcal{T}$ *the following holds:*

- *If* $\mathcal{S} \Downarrow_x$ *then* $\mathcal{T} \Downarrow_x$*;*
- *If* $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \rightleftharpoons^* \mathcal{T}'$ *and* $\mathcal{S}'\ \Phi\ \mathcal{T}'$*.*

Let $\dot{\approx}_b$ *be the largest barbed bisimulation. As usual, this induces an equivalence on open terms, also noted* $\dot{\approx}_b$*:* $A \dot{\approx}_b A' \stackrel{\text{def}}{=} \emptyset \Vdash_\emptyset \{A\} \dot{\approx}_b \emptyset \Vdash_\emptyset \{A'\}$*.*

A barbed congruence *is a barbed bisimulation that is also a congruence for open join-calculus contexts. Let* $\approx_b$ *be the largest barbed congruence.*

Barbed bisimulation is far from being a congruence. For instance, $x\langle a\rangle \dot{\approx}_b x\langle b\rangle$ but $x\langle a\rangle \not\approx_b x\langle b\rangle$ because the context $\mathsf{def}\ x\langle u\rangle \triangleright u\langle\rangle\ \mathsf{in}\ [\,]$ separates them. Barbed congruence has a better discriminating power, and does not depend on the particular definition of barbs we use; indeed, we could as well detect emissions only on free names of arity 0, as is done in [4], or even on a single message $\mathsf{test}\langle\rangle$, and still obtain the same equivalence using the congruence property.

3 Weak Bisimulation

The join-calculus has been "opened" to support the standard notion of bisimulation for process calculi [12]. In this section, we define this equivalence for open solutions and we investigate its properties.

DEFINITION 3.1 *A symmetric relation* Φ *over open solutions is a* weak bisimulation *if, whenever* $\mathcal{S}\ \Phi\ \mathcal{T}$*, then*

- *if* $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \rightleftharpoons^* \mathcal{T}'$ *and* $\mathcal{S}'\ \Phi\ \mathcal{T}'$*;*
- *if* $\mathcal{S} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{T}'$ *and* $\mathcal{S}'\ \Phi\ \mathcal{T}'$*,*
  for all labels α *of shape* $x\langle\widetilde{v}\rangle$ *or* $S\overline{x}\langle\widetilde{v}\rangle$ *such that names in* S *are fresh in* $\mathcal{T}$*.*

Let $\approx$ *be the largest weak bisimulation. We shall write* $A \approx B$ *for* $(\emptyset \Vdash A) \approx (\emptyset \Vdash B)$*.*

As an immediate consequence of its definition, weak bisimulation is an equivalence relation, and we can tell whether two processes are bisimilar by reasoning on their synchronization tree, without the need to exhibit discriminating contexts. For example, $x\langle u\rangle \not\approx x\langle v\rangle$ because they perform emissions with different labels. Likewise, $x\langle y\rangle \not\approx \mathsf{def}\ z\langle u\rangle \triangleright y\langle u\rangle\ \mathsf{in}\ x\langle z\rangle$ because the first process emits a free name (label $\overline{x}\langle y\rangle$) while the latter emits a local name that gets extruded (label $\{z\}\overline{x}\langle z\rangle$). The following properties should convince the reader that $\approx$ is a reasonable semantics for the join-calculus.

3.1 Bisimilarity and chemical interface

The set of extruded names $\mathsf{xv}[\mathcal{S}]$ is the input-interface of solution $\mathcal{S}$, and the shape of rule (INT) makes weak bisimulation sensitive to it: $\mathcal{S} \approx \mathcal{T}$ implies $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$

To check this property, let us consider the partition of open RCHAMs induced by $\mathsf{xv}[\cdot]$. The resulting classes are stable for all chemical rules except (EXT), and in particular, any solution is structurally equivalent to a fully-heated solution with the same extruded variables; by definition of (INT), a fully-heated solution can perform a transition with label $x\langle\tilde{v}\rangle$ if and only if $x \in \mathsf{xv}[\mathcal{S}]$.

Conversely, rule (EXT) defines the only transitions that can separate solutions with the same extruded names. Obviously, open solutions whose derivatives never perform (EXT) are bisimilar if and only if they have the same extruded names to begin with, and this provides some "garbage-collection" properties such as $\mathsf{fv}[P] = \emptyset$ implies $P \approx 0$.

3.2 Internal choice

We define a derived operator for *internal choice*, which we use in the sequel. We set

$$\bigoplus_{i\in[1..n]} P_i \stackrel{\text{def}}{=} \mathsf{def} \bigwedge_{i\in[1..n]} c\langle\rangle \triangleright P_i\ \mathsf{in}\ c\langle\rangle$$

where c is a fresh name, and we abbreviate $\bigoplus_{i\in[1..2]} P_i$ as $P_1 \oplus P_2$. Up-to structural equivalence, the initial transitions of this internal sum are exactly $\bigoplus_i P_i \Rightarrow^* \rightarrow \Rightarrow^* P_i \mid Q$, where Q is the remaining definition with no more message $c\langle\rangle$, and thus $Q \approx \mathbf{0}$. We retrieve the usual axioms of internal choice from the definition; for instance we have $(P \oplus Q) \oplus Q \approx P \oplus Q$.

3.3 Equivalent definitions

The following examples show that weak bisimulation is largely insensitive to the shape of join-patterns.

redundant rules

$$\mathsf{def}_S\ D \wedge D'\ \mathsf{in}\ A \quad\approx\quad \mathsf{def}_S\ D \wedge D \wedge D'\ \mathsf{in}\ A \tag{1}$$

$$\begin{array}{c} \mathsf{def}_S\ J \triangleright P \wedge J' \triangleright P' \\ \wedge\ D'\ \mathsf{in}\ A \end{array} \quad\approx\quad \begin{array}{c} \mathsf{def}_S\ J \triangleright P \wedge J' \triangleright P' \\ \wedge\ J \,|\, J' \triangleright P \,|\, P' \wedge D'\ \mathsf{in}\ A \end{array} \tag{2}$$

buffering of incoming messages Weak bisimulation is insensitive to buffering, as one expects from an asynchronous semantics. For instance we have:

$$\mathsf{def}_S\ D \ \mathsf{in}\ A \quad \approx \quad \mathsf{def}_S\ x\langle\widetilde{u}\rangle \triangleright x'\langle\widetilde{u}\rangle \wedge D' \ \mathsf{in}\ A \tag{3}$$

where $x \in \mathsf{dv}[D]$, x' is a fresh name, and D' is obtained from D by substituting $x'\langle\widetilde{v}\rangle$ for $x\langle\widetilde{v}\rangle$ in every join-pattern. We also have:

$$\mathsf{def}_S\ J \triangleright P \wedge D \ \mathsf{in}\ A \quad \approx \quad \mathsf{def}_S\ J \triangleright x\langle\widetilde{\mathsf{rv}[J]}\rangle \wedge x\langle\widetilde{\mathsf{rv}[J]}\rangle \triangleright P \wedge D \ \mathsf{in}\ A \tag{4}$$

where x is a fresh name and $\widetilde{\mathsf{rv}[J]}$ abbreviates a tuple that conveys the names in $\mathsf{rv}[J]$.

These properties of asynchrony are easy to prove (it suffices to consider the relation that contains processes that differ only because of a buffering reduction, which is included in a bisimulation, or equivalently, to use the bisimulation up-to expansion proof technique of [20]). Notice, however, that buffering of partial join-patterns would not usually preserve bisimulation, because of the usual problem of gradual commitment. For instance, the commitment to one message on x separates the following processes.

$$\begin{array}{lll} & \mathsf{def}_{\{z\}}\ x\langle u\rangle \,|\, y\langle\rangle \,|\, z\langle v\rangle \triangleright P & \mathsf{in}\ x\langle 1\rangle \,|\, x\langle 2\rangle \,|\, y\langle\rangle \\ \not\approx & \mathsf{def}_{\{z\}}\ x\langle u\rangle \,|\, y\langle\rangle \triangleright t\langle u\rangle \wedge t\langle u\rangle \,|\, z\langle v\rangle \triangleright P & \mathsf{in}\ x\langle 1\rangle \,|\, x\langle 2\rangle \,|\, y\langle\rangle \end{array}$$

While these processes have the same output traces, the latter can reduce to $\mathsf{def}_{\{z\}}\ x\langle u\rangle \,|\, y\langle\rangle \triangleright t\langle u\rangle \wedge t\langle u\rangle \,|\, z\langle v\rangle \triangleright P \ \mathsf{in}\ t\langle 1\rangle \,|\, x\langle 2\rangle$, and this internal step cannot be simulated by the former process. Some variant of coupled simulation would be adequate to identify such definitions [17].

divergence It is well-known that weak bisimulation is insensitive to divergence; for instance $\mathbf{0} \approx \mathsf{def}\ x\langle\rangle \triangleright x\langle\rangle \ \mathsf{in}\ x\langle\rangle$. Likewise, any reversible synchronization is invisible; for instance, we can temporarily grab arbitrary messages J within a definition D

$$\mathsf{def}_S\ D \ \mathsf{in}\ A \quad \approx \quad \mathsf{def}_S\ J \triangleright J' \wedge J' \triangleright J \wedge D \ \mathsf{in}\ A \tag{5}$$

where $\mathsf{dv}[J] \subseteq \mathsf{dv}[D]$, $\mathsf{dv}[J']$ fresh, and $\mathsf{rv}[J'] = \mathsf{rv}[J]$.

adjunction of internal state Another standard property of bisimulation is that it is not possible to observe the internal state of processes. As mentioned in [4] for the weaker barbed congruence, we have

$$\mathsf{def}_S\ J \triangleright P \ \mathsf{in}\ A \quad \approx \quad \mathsf{def}_S\ J \,|\, s\langle\widetilde{v}\rangle \triangleright P \,|\, s\langle\widetilde{v}\rangle \ \mathsf{in}\ A \,|\, s\langle\widetilde{v}\rangle \tag{6}$$

where s is a fresh name and $\{\widetilde{v}\} \cap \mathsf{rv}[J] = \emptyset$. Indeed, there is always one available message $s\langle\widetilde{v}\rangle$ that conveys the same names, that were initially bound in a process *within the same scope*. This equality suffices to prove interesting properties with regards to our scoping rules: if we take $\{\widetilde{v}\} = \mathsf{dv}[J] \cup \{s\}$, then all occurrences of names of $\mathsf{dv}[J]$ that appeared in P — we call them recursive occurrences — are now bound as received variables. Up-to bisimulation, we can therefore eliminate recursion from every definition.

If we take $\{\widetilde{v}\} = (\mathsf{fv}[P] \cup \{s\}) \setminus \mathsf{rv}[J]$, then all names in P are now bound as received variables, which is reminiscent of lambda-lifting in the λ-calculus. In combination with the forthcoming congruence properties, this validates a compilation scheme for the join-calculus that would replace every process by an equivalent process with simpler binders (either receptions or immediate definitions).

3.4 Renaming and Congruence properties

The rest of the section is devoted to proving that weak bisimulation is indeed a congruence. This is not very surprising since our calculus has no operator for external choice. Nevertheless the proof is not completely standard and some interesting properties are required. In order to establish that weak bisimulation is persistent under global renaming, we first study the correspondence between transitions in a chemical solution $\mathcal{S}$ and transitions in $\mathcal{S}\sigma$, where σ is a global renaming.

LEMMA 3.2 *Let $\mathcal{S}$ be an open solution and σ be a global renaming for $\mathcal{S}$.*

1. $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}$ *implies* $\mathcal{S}\sigma \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}\sigma$*; conversely,* $\mathcal{S}\sigma \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}$ *implies* $\mathcal{S} \rightleftharpoons^* \xrightarrow{S_1\overline{x_1}\langle\widetilde{v}_1\rangle} \xrightarrow{y_1\langle\widetilde{v}_1\rangle} \cdots \xrightarrow{S_n\overline{x_n}\langle\widetilde{v}_n\rangle} \xrightarrow{y_n\langle\widetilde{v}_n\rangle} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ *where* $n \geq 0$*, for all* $i \leq n$ $\sigma(x_i) = \sigma(y_i)$*, and* $\mathcal{T} = \mathcal{S}' \setminus (\bigcup_{i=1\ldots n} S_i)$*;*
2. $\mathcal{S} \rightleftharpoons^* \xrightarrow{x\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{T}$ *if and only if* $\mathcal{S}\sigma \rightleftharpoons^* \xrightarrow{(x\langle\widetilde{v}\rangle)\sigma} \rightleftharpoons^* \mathcal{T}\sigma$*;*
3. *if* $\sigma(x)$ *is free,* $\mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{T}$ *implies* $\mathcal{S}\sigma \rightleftharpoons^* \xrightarrow{(S\overline{x}\langle\widetilde{v}\rangle)\sigma} \rightleftharpoons^* \mathcal{T}\sigma$*;*
   conversely, $\mathcal{S}\sigma \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{T}$ *implies* $\mathcal{S} \rightleftharpoons^* \xrightarrow{S'\overline{x'}\langle\widetilde{v}'\rangle} \rightleftharpoons^* \mathcal{S}'$*,*
   where $\mathcal{S}'\sigma \rightleftharpoons^* \mathcal{T}$ *and* $(S'\overline{x'}\langle\widetilde{v}'\rangle)\sigma = S\overline{x}\langle\widetilde{v}\rangle$*.*

The statements in the above lemma are natural, except perhaps for the last part of Case 1. This case occurs when an internal step consumes messages on extruded names $\sigma(x_i)$ where x_i are free names in the initial solution $\mathcal{S}$. For every such message that is consumed in the internal step, this is mimicked as an extrusion on x_i followed by an intrusion on y_i.

Proof: 1. We demonstrate only the converse statement ($\mathcal{S}\sigma \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}$). Let $M\sigma$ be the multiset of emissions that is consumed by the reaction rule that is used for this silent move. Up-to the reordering of the messages, we can assume that

$$M = x_1\langle\widetilde{v}_1\rangle \mid \cdots \mid x_n\langle\widetilde{v}_n\rangle \mid x_{n+1}\langle\widetilde{v}_{n+1}\rangle \mid \cdots \mid x_m\langle\widetilde{v}_m\rangle$$

where the names $x_i, i \leq n$ are free and the names $x_j, j > n$ are defined. In particular we can extrude the first n messages one by one using Rule (EXT). For every $i = 1 \ldots n$, the silent move after substituting requires that $\sigma(x)$ be an extruded name. This is possible only inasmuch as $\sigma^{-1}(\sigma(x))$ contains exactly one extruded name y_i, which can be used before substituting to re-intrude the same message. Once these $2n$ transitions have been completed, we use structural equivalence to re-assemble a multiset of messages

$$M' = y_1\langle\widetilde{v}_1\rangle \mid \cdots \mid y_n\langle\widetilde{v}_n\rangle \mid x_{n+1}\langle\widetilde{v}_{n+1}\rangle \mid \cdots \mid x_m\langle\widetilde{v}_m\rangle$$

such that $M'\sigma = M\sigma$, and that M' match the join-pattern of the same reaction rule before renaming. Once the reaction has occurred, we apply σ and obtain the same multisets of processes and definitions. Notice, however, that some of the conveyed names $S_i \subseteq \widetilde{v}_i$ that have been unduly extruded to the context in the round trip, and remain marked as extruded: the solution before applying the global renaming is more open than needed.

2. This is immediate because extruded variables are renamed to distinct fresh names.

3. The first half is obvious. For the second part remark that, by definition of global renaming, $\mathcal{S}\sigma$ may extrude along $\sigma(x)$, provided the counter-image of $\sigma(x)$ is a set of free variables in $\mathcal{S}$, hence the sentence.

$\square$

It is usually easier to prove that processes are bisimilar using relations much smaller than bisimulations, and a whole range of "up-to proof techniques" are available to reduce the size of the relation in use [12]. We now establish that our definition of weak bisimulation is robust with respect to reasonings up-to structural equivalence, restriction of the input interface, global renamings, and weak-bisimulation on the right. As usual, we derive the definitions of "weak bisimulation up-to" from the definition of weak bisimulation, and we use the resulting definitions as proof techniques.

LEMMA 3.3 *Let Φ be a relation that satisfies Definition 3.1 where the requirement $\mathcal{S}'\ \Phi\ \mathcal{T}'$ has been replaced by one of the following weaker requirements.*

1. $\mathcal{S}' \rightleftharpoons^* \Phi \rightleftharpoons^* \mathcal{T}'$

2. $\mathcal{S}'\ \Phi{\approx}\ \mathcal{T}'$

3. *there is a set of names S such that $\mathcal{S}'' \approx \mathcal{T}''$, $\mathcal{S}' = \mathcal{S}'' \setminus S$, and $\mathcal{T}' = \mathcal{T}'' \setminus S$.*

4. *there is a global renaming σ such that $\mathcal{S}'' \approx \mathcal{T}''$, $\mathcal{S}' = \mathcal{S}''\sigma$, and $\mathcal{T}' = \mathcal{T}''\sigma$.*

Then we have $\Phi \subseteq \approx$.

Proof: In each case, we exhibit a bisimulation that contains Φ:

1. The definition of weak bisimulation already takes into account structural rearrangements; as $\rightleftharpoons^*$ is symmetric, the statement is obvious.

2. The relation $\Phi{\approx}$ is a weak bisimulation; this directly follows from the transitivity of $\approx$. Note that this is distinct from weak-bisimulation up-to weak-bisimulation, as we only use $\approx$ on the right side.

3. The relation $\Phi\backslash \stackrel{\text{def}}{=} \{(\mathcal{S}\backslash S, \mathcal{T}\backslash S)$ when defined such that $\mathcal{S} \approx \mathcal{T}\}$ is a weak bisimulation. All transitions after restriction are allowed before, and thus the bisimulation properties after restriction are direct consequences of the same properties before restriction.

4. The relation $\Sigma(\Phi) \stackrel{\text{def}}{=} \{(\mathcal{S}\sigma, \mathcal{T}\sigma)$ such that $\mathcal{S}\ \Phi\ \mathcal{T}$ and σ global renaming $\}$ is a weak bisimulation up-to restriction. This directly rely on our case analysis of the effect of each kind of renaming on every chemical rule. We only show in detail the case of internal reduction, as the others are much easier.

   Let $\mathcal{S}\sigma \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'\sigma$. By Lemma 3.2(1) we obtain a series of extrusion-intrusion pairs followed by a silent move

   $$\mathcal{S} \rightleftharpoons^* \xrightarrow{S_1 \overline{x_1}\langle \widetilde{v}_1\rangle} \xrightarrow{y_1\langle \widetilde{v}_1\rangle} \cdots \xrightarrow{S_n \overline{x_n}\langle \widetilde{v}_n\rangle} \xrightarrow{y_n\langle \widetilde{v}_n\rangle} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}''$$

   We apply the bisimulation property to every transition in turn, and thus we obtain that there is a derivation leading to some $\mathcal{T}''\ \Phi\ \mathcal{S}''$:

   $$\mathcal{T} \Rrightarrow^* \xrightarrow{S_1 \overline{x_1}\langle \widetilde{v}_1\rangle} \mathcal{T}_1 \Rrightarrow^* \mathcal{T}_1' \xrightarrow{y_1\langle \widetilde{v}_1\rangle} \mathcal{T}_1'' \Rrightarrow^* \cdots \Rrightarrow^* \mathcal{T}_n' \xrightarrow{y_n\langle \widetilde{v}_n\rangle} \mathcal{T}_n'' \Rrightarrow^* \mathcal{T}''$$

   To go back under the global renaming σ, we study the action of σ on this series of transitions. Silent steps remain silent, but the extrusions cannot be performed anymore, so we keep the message in solution. We use the auxiliary notation $\mathcal{T}_i[M]$ to represent the chemical solution $\mathcal{T}_i$ with one extra message M in the multiset of processes. For instance we have $\mathcal{T}_i'' = \mathcal{T}_i'[y_i\langle \widetilde{v}_i\rangle]$.

   The transitions after the renaming become, for $S \stackrel{\text{def}}{=} \bigcup S_i$,

   $$\begin{array}{rcl} \mathcal{T}\sigma \Rrightarrow^* (\mathcal{T}_1[x_1\langle \widetilde{v}_1\rangle])\sigma\backslash S_1 & \Rrightarrow^* & (\mathcal{T}_1'[x_1\langle \widetilde{v}_1\rangle])\sigma\backslash S_1 \\ & = & (\mathcal{T}_1'[y_1\langle \widetilde{v}_1\rangle])\sigma\backslash S_1 \\ & = & \mathcal{T}_1''\sigma \Rrightarrow^* \cdots \Rrightarrow^* \ (\mathcal{T}_n'[x_n\langle \widetilde{v}_n\rangle])\sigma\backslash S \\ & & \quad = \ (\mathcal{T}_n'[y_n\langle \widetilde{v}_n\rangle])\sigma\backslash S \\ & & \quad = \ \mathcal{T}_n''\sigma\backslash S \Rrightarrow^* \mathcal{T}''\sigma\backslash S \end{array}$$

   that is, after aggregating the internal moves, $\mathcal{T}\sigma \Rrightarrow^* \mathcal{T}''\sigma \setminus S$. Therefore, we have $S' = \mathcal{S}''\sigma \setminus S$, $T' = \mathcal{T}''\sigma \setminus S$, and $(\mathcal{S}''\sigma, \mathcal{T}''\sigma)$ in our relation $\Sigma(\Phi)$.

In particular, if we take $\Phi = \approx$ we directly obtain that weak bisimulation is preserved by structural equivalence, restriction and global renaming. ▯

Corollary 3.4 *$\mathcal{S} \approx \mathcal{T}$ implies $\mathcal{S}\sigma \approx \mathcal{T}\sigma$ for every global renaming σ.*

The congruence property essentially relies on this proposition; its proof is standard.

Theorem 3.5 *Weak bisimulation is a congruence.*

Proof: We first prove that weak bisimulation is a congruence for contexts of the form $C[\cdot] \stackrel{\mathrm{def}}{=} \mathsf{def}_S\ D\ \mathsf{in}\ M \mid [\cdot]$, where M is a parallel composition of messages. Without loss of generality, we can assume that all the free variables of processes A in $C[A]$ that are bound by C are extruded variables, and then conclude using Lemma 3.3(3).

For a given pair of bisimilar open processes $A \approx B$, let σ be the inverse of a global renaming that substitutes pairwise distinct names that do not appear in $C[\cdot]$ for all the names in $\mathsf{fv}[A] \cup \mathsf{xv}[A] \cup \mathsf{fv}[B] \cup \mathsf{xv}[B]$. We still have $A' \approx B'$, where $A'\sigma = A$ and $B'\sigma = B$, and we can now rewrite $C[A] = C[A'\sigma] = (C[A'])\sigma \rightleftharpoons^* (C[\mathbf{0}] \mid A')\sigma$, and similarly $C[B] \rightleftharpoons^* (C[\mathbf{0}] \mid A')\sigma$. Using Lemma 3.3(4), we prove that the following relation Φ is a bisimulation up-to renaming:

$$\left\{ C \,|\, A,\ C \,|\, B \text{ such that } \begin{array}{l} A \approx B, \\ \mathsf{fv}[C], \mathsf{fv}[A] \cup \mathsf{fv}[B], \mathsf{xv}[C], \mathsf{xv}[A] \text{ pairwise disjoint} \end{array} \right\}$$

Notice that, before applying the renaming, the process and the context do not interact at all. Except for intrusions, every transitions on one side is independent of the other side, and the resulting pair of processes are still in Φ. Let now suppose that $C \mid A \rightleftharpoons^* \xrightarrow{x\langle \widetilde{w} \rangle} \rightleftharpoons^* C \mid A \mid x\langle \widetilde{w} \rangle$. The label of this intrusion can be rewritten as $(x\langle \widetilde{v} \rangle)\rho$ where the v's are distinct fresh names and ρ substitutes the actual intruded names $\widetilde{w}$ for the fresh names $\widetilde{v}$. But the intrusion with label $(x\langle \widetilde{v} \rangle)$ is also enabled for $C \mid A$ (in fact it is enabled on either for every $\widetilde{v}$), and can be simulated by $C \mid B$. If for instance $x \in \mathsf{xv}[A]$ we have

$$\begin{array}{lcl} C \mid A & \rightleftharpoons^* \xrightarrow{x\langle \widetilde{v} \rangle} \rightleftharpoons^* & C \mid A' \text{ such that } A'\rho = A \mid x\langle \widetilde{w} \rangle \\ C \mid B & \Rrightarrow^* \xrightarrow{x\langle \widetilde{v} \rangle} \Rrightarrow^* & C \mid B' \text{ such that } (C \mid A')\ \Phi\ (C \mid B') \\ C \mid B & \Rrightarrow^* \xrightarrow{x\langle \widetilde{v}\rho \rangle} \Rrightarrow^* & (C \mid B')\rho \end{array}$$

The same analysis applies for $x \in \mathsf{xv}[C]$, and thus Φ is a bisimulation up-to renaming.

It remains to prove that congruence for contexts as $\mathsf{def}_S\ D\ \mathsf{in}\ M \mid [\cdot]$ can be lifted to the full congruence. To this aim we prove that the following relation Φ is a bisimulation up-to weak-bisimulation-on-the-right:

$$\left\{ \Big((\mathcal{D}, J \triangleright P \Vdash_S \mathcal{A}), (\mathcal{D}, J \triangleright Q \Vdash_S \mathcal{A}) \Big) \text{ such that } P \approx Q \right\}$$

All transitions are identical, except for the reduction steps that consume some messages $J\sigma$ to trigger the reaction rule we isolated above. However, we can use our partial congruence result to match the requirement of Lemma 3.3(2):

$$\begin{array}{rcl} \mathcal{D}, J \triangleright P \Vdash_S J\sigma, \mathcal{A} & \rightarrow & \mathcal{D}, J \triangleright P \Vdash_S P\sigma, \mathcal{A} \\ & \Phi & \mathcal{D}, J \triangleright Q \Vdash_S P\sigma, \mathcal{A} \\ & \approx & \mathcal{D}, J \triangleright Q \Vdash_S Q\sigma, \mathcal{A} \end{array}$$

We finally prove the full congruence by structural induction on all contexts, using structural equivalence before applying the two previous results. $\square$

3.5 Weak Bisimulation versus Barbed Congruence

Weak bisimulation is a congruence and a barbed bisimulation, and thus a barbed congruence ($\approx\,\subset\,\approx_b$). This containment is strict, as can be seen from the paradigmatic example of barbed congruence:

$$x\langle z\rangle \;\approx_b\; \mathsf{def}\; u\langle v\rangle \triangleright z\langle v\rangle \;\mathsf{in}\; x\langle u\rangle\;.$$

That is, emitting a free name z is the same as emitting a bound name u that forwards all the messages it receives to z, and the two processes are indistinguishable, up-to an extra internal move for every use of z. These two processes are distinguished by weak bisimulation, however, because their respective extrusion labels reveal that z is free and u is extruded.

4 Asynchronous Bisimulation

In order to prove that two chemical solutions are bisimilar, the large number of multiset configurations to consider is a nuisance; for example, a process with an extruded name already has an infinite set of configurations, even if no "real" computation is ever performed. In order to check the obvious equivalence

$$\mathsf{def}\; x\langle u\rangle \,|\, y\langle v\rangle \triangleright P \;\mathsf{in}\; z\langle x\rangle \quad\approx\quad \mathsf{def}\; x\langle u\rangle \mid y\langle v\rangle \triangleright Q \;\mathsf{in}\; z\langle x\rangle$$

one is confronted to infinite models on either sides, with a distinct chemical solution for every multiset of messages that have been intruded on x so far. This flaw in our definition of bisimulation motivates the following alternative formulation.

We refine the open RCHAM by allowing inputs only when they trigger some guarded process. Thus, we augment the blocking ability of inputs. For example, the two processes above become inert, and trivially bisimilar.

If we apply our refinement with our previous labels for input, however, we obtain a dubious result. The solution $x\langle\rangle \,|\, y\langle\rangle \,|\, z\langle\rangle \triangleright P \Vdash_{\{x,y\}} z\langle\rangle$ can transit inasmuch as it first input the two messages $x\langle\rangle$ and $y\langle\rangle$, then perform a silent step that consumes them together with the local message $z\langle\rangle$ already in the solution. Yet, neither $x\langle\rangle$ nor $y\langle\rangle$ alone can trigger the process P, and therefore this solution would becomes inert too. This suggests the use of *join*-inputs on x and y in new reduction steps such that

$$x\langle\rangle \mid y\langle\rangle \mid z\langle\rangle \triangleright P \Vdash_{\{x,y\}} z\langle\rangle \xrightarrow{x\langle\rangle|y\langle\rangle} x\langle\rangle \mid y\langle\rangle \mid z\langle\rangle \triangleright P \Vdash_{\{x,y\}} P\;.$$

On the other hand, the solution $x\langle\rangle \,|\, y\langle\rangle \,|\, z\langle\rangle \triangleright P \Vdash_{\{x\}} z\langle\rangle$ is truly inert, since P can never be triggered. Here join-inputs have a greater blocking ability than atomic inputs, and our refinement suppresses all input transitions.

4.1 The J-open RCHAM

The J-open RCHAM is defined in Table 2, as a replacement of the intrusion rule. Unlike the previous rule (INT), the new rule (INT-J) permits intrusion of messages only if they

Rules (S-JOIN), (S-AND), (S-DEF), (EXT) are as in Table 1;
(INT-J) $\qquad J \triangleright P \Vdash_S M \xrightarrow{M'} J \triangleright P \Vdash_S P\rho$
Side condition: $J\rho \rightleftharpoons^* M \mid M'$, $\quad Dom(\rho) = \mathsf{rv}[J]$, $\mathsf{dv}[M'] \subseteq S$, $\quad \mathsf{rv}[M']$ free, fresh or extruded.

Table 2: The J-open RCHAM of the open join calculus

are immediately used to trigger a process in a definition. This is formalized by allowing labels M' that are multisets of messages, and by handling them as partial join-patterns: if the solution can supply a complementary join-pattern M such that the combination $M \mid M'$ matches the join-pattern in a reaction rule, then the transition occurs and triggers this reaction rule. As for (INT), we restrict intrusions in M' to messages on extruded names. Note that rule (RED) is now subsumed by rule (INT-J) where $M' = \mathsf{0}$, although we keep the distinction between internal moves and proper input moves in the following discussion.

The same chemical solution now has two different models: for instance, $x\langle\rangle \mid y\langle\rangle \triangleright P \Vdash_{\{x\}}$ has no transition in the J-open RCHAM, while it has infinitely many transitions in the open RCHAM:

$$x\langle\rangle \mid y\langle\rangle \triangleright P \Vdash_{\{x\}} \xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \cdots$$

In the sequel, we shall keep the symbol $\xrightarrow{\alpha}$ for the open RCHAM and use $\xrightarrow{\alpha}_J$ for the J-open RCHAM; the subscript J is dropped whenever no ambiguity can arise.

As a direct consequence of the definitions of the J-open semantics, we have the following relation between our two models:

PROPOSITION 4.1 *The intrusions of the* J-*open* RCHAM *and those of the open* RCHAM *are related as follows:*

1. $\mathcal{S} \xrightarrow{x_1\langle\widetilde{v_1}\rangle|\cdots|x_n\langle\widetilde{v_n}\rangle}_J \mathcal{S}'$ *implies* $\mathcal{S} \xrightarrow{x_i\langle\widetilde{v_1}\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v_n}\rangle} \rightleftharpoons^* \rightarrow \mathcal{S}'$.
2. *For any* A, *if* $A \mid x\langle\widetilde{u}\rangle \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* B$, *then either* $A \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* A'$ *with* $A' \mid x\langle\widetilde{u}\rangle \rightleftharpoons^* B$, *or* $A \rightleftharpoons^* \xrightarrow{M'}_J \rightleftharpoons^* B$ *with* $M' \rightleftharpoons^* M \mid x\langle\widetilde{u}\rangle$.

All other transitions are common to both chemical machines.

Proof: 1. The side-conditions on names that may appear on intrusion labels are the same for both machines; thus we can use Rule (INT) to intrude all messages one at a time. Once this is done, we use structural equivalence to assemble a complete join-pattern, and we perform a (RED) transition on the reaction rule that is used by the J-open RCHAM to perform the (INT-J) transition, which yield the same chemical solution.

2. The choice between the two possibilities is directed by the extra internal messages that are consumed by the (INT-J) transition. We check that if the message in not used, then the first statement hold, and that otherwise it must be part of the extra messages (within M in the definition of Rule (INT-J)).

$\square$

REMARK 4.2 *Intrusions and extrusions are naturally dual rules and could be used to define a transition system up-to the structural rules of the chemical machine and alpha-renaming. This approach would suggest the use of "join-extrusions" which aggregate join-patterns on the extrusions labels, as opposite to join-intrusions. As a proof technique, however, it seems preferable to keep extrusions simple, as anyway there is no involved synchronization.*

4.2 Asynchronous bisimulation

We now try to shape the definition of weak bisimulation (Definition 3.1) to the new J-open RCHAM. Consider for instance the two processes:

$$P \;=\; \mathsf{def}\ x\langle\rangle \triangleright a\langle\rangle \;\wedge\; a\langle\rangle \,|\, y\langle\rangle \triangleright R \ \mathsf{in}\ z\langle x, y\rangle \qquad\qquad Q \;=\; \mathsf{def}\ x\langle\rangle \,|\, y\langle\rangle \triangleright R \ \mathsf{in}\ z\langle x, y\rangle$$

With the previous underlying model (open RCHAM), P and Q are weakly bisimilar, but with the new one (J-open RCHAM) this is not true anymore, because after emitting on z, P can input on $x\langle\rangle$ while Q cannot. If we equip the J-open RCHAM with the weak bisimulation of Section 3, Q becomes inert after the extrusion on z because join inputs are not considered. But if we consider the weak bisimulation that uses join-input labels instead of single ones, Q can input on $x\langle\rangle \,|\, y\langle\rangle$ while P cannot, and P and Q are still separated. It turns out that weak bisimulation discriminates too much in the J-open RCHAM.

In order to avoid detection of the structure of join patterns, weak bisimulation must be weakened further on. At least, we must consider that a process simulate another one even if it does not immediately consume a join-input.

DEFINITION 4.3 *A symmetric relation Φ over* J*-open* RCHAM*s is an* asynchronous bisimulation *if, whenever $\mathcal{S}\ \Phi\ \mathcal{T}$, then*

- $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$
- *if $\mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$ and names in S are fresh in $\mathcal{T}$,*
  then $\mathcal{T} \Longrightarrow^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \Longrightarrow^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$;
- *if $\mathcal{S} \rightleftharpoons^* \xrightarrow{M} \rightleftharpoons^* \mathcal{S}'$, then $\mathcal{T} \mid M \Longrightarrow^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$.*

Let $\approx_a$ be the largest asynchronous bisimulation. We write $A \approx_a B$ for $(\emptyset \Vdash \{A\}) \approx_a (\emptyset \Vdash \{B\})$.

Let us show the equality $P \approx_a Q$, where P and Q are the processes at the beginning of this section. Both P and Q perform the same emission, therefore it suffices to prove

$$A = \mathsf{def}_{\{x,y\}}\ x\langle\rangle \triangleright a\langle\rangle \ \wedge\ a\langle\rangle \,|\, y\langle\rangle \triangleright R \ \mathsf{in}\ 0 \quad \approx_a \quad B = \mathsf{def}_{\{x,y\}}\ x\langle\rangle \,|\, y\langle\rangle \triangleright R \ \mathsf{in}\ 0$$

We observe $A \xrightarrow{x\langle\rangle} A'$, while $B \not\xrightarrow{x\langle\rangle}$. Nevertheless, it is possible to prove that $A' \approx_a \mathsf{def}_{\{x,y\}}\ x\langle\rangle \mid y\langle\rangle \triangleright R\ \mathsf{in}\ x\langle\rangle$. Hence, bisimilarity can be still achieved, by delaying the actual use of a message. Conversely, there is B' such that $B \xrightarrow{x\langle\rangle|y\langle\rangle} B'$, whilst this move cannot be mimicked by A. Again, one can prove that B' is asynchronous bisimilar to $\mathsf{def}_{\{x,y\}}\ x\langle\rangle \triangleright a\langle\rangle \wedge a\langle\rangle \mid y\langle\rangle \triangleright R\ \mathsf{in}\ R$, which is a derivative of $A \mid x\langle\rangle \mid y\langle\rangle$. All other transitions are the same, and we can conclude that $P \approx_a Q$.

REMARK 4.4 *In the definition of asynchronous bisimulation, it is necessary to require that related solutions have the same extruded names. Otherwise, Remark 3.1 would not carry over asynchronous bisimulation, as for instance the open deadlocked solution* $x\langle\rangle \,|\, y\langle\rangle \triangleright P \Vdash_{\{y\}}$ *would be equivalent to the empty solution* $\Vdash_{\emptyset}$.

PROPOSITION 4.5 $\mathcal{S} \approx_a \mathcal{T}$ *implies* $\mathcal{S} \mid x\langle\widetilde{v}\rangle \ \approx_a\ \mathcal{T} \mid x\langle\widetilde{v}\rangle$.

Proof: Foremost observe that asynchronous bisimulation up-to structural equivalence is a valid proof technique. Then we prove that the following relation is an asynchronous bisimulation:

$$\Phi \stackrel{\text{def}}{=} \approx_a \cup \{(\mathcal{S} \mid x\langle\widetilde{v}\rangle, \mathcal{T} \mid x\langle\widetilde{v}\rangle) \text{ such that } \mathcal{S} \approx_a \mathcal{T}\}$$

We only consider the case of intrusions, all other cases being immediate. Let $\mathcal{S} \mid x\langle\widetilde{v}\rangle \Rightarrow^{*} \xrightarrow{M}_{J} \Rightarrow^{*} \mathcal{S}'$. By Proposition 4.1, one of the following holds:

(*a*) $\mathcal{S} \Rightarrow^{*} \xrightarrow{M}_{J} \Rightarrow^{*} \mathcal{S}''$ with $\mathcal{S}' = \mathcal{S}'' \mid x\langle\widetilde{v}\rangle$. If $\mathcal{S} \approx_a \mathcal{T}$, then we have $\mathcal{T} \mid M \Rrightarrow^{*} \mathcal{T}' \approx_a \mathcal{S}'$, thus $\mathcal{T} \mid M \mid x\langle\widetilde{v}\rangle \Rrightarrow^{*} \mathcal{T}' \mid x\langle\widetilde{v}\rangle$ with $\mathcal{S}' \mid x\langle\widetilde{v}\rangle \ \Phi\ \mathcal{T}' \mid x\langle\widetilde{v}\rangle$.

(*b*) $\mathcal{S} \Rightarrow^{*} \xrightarrow{M|x\langle\widetilde{v}\rangle}_{J} \Rightarrow^{*} \mathcal{S}'$. If $\mathcal{S} \approx_a \mathcal{T}$, then we have $\mathcal{T} \mid M \mid x\langle\widetilde{v}\rangle \Rrightarrow^{*} \mathcal{T}' \approx_a \mathcal{S}'$, and in particular $\mathcal{S}' \ \Phi\ \mathcal{T}'$.

$\square$

COROLLARY 4.6 $\approx_a$ *is an equivalence relation.*

Proof: We establish that $\approx_a$ is transitive; to this end, we prove that the relation $\approx_a \approx_a$ is an asynchronous bisimulation. Again, the only interesting case is join-intrusion.

Let assume that $\mathcal{S}_0 \approx_a \mathcal{S} \approx_a \mathcal{T}$, and that $\mathcal{S}_0$ performs the join-intrusion labeled with the messages M. By Proposition 4.5 we know that $\mathcal{S} \,|\, M \approx_a \mathcal{T} \,|\, M$. By definition of asynchronous bisimulation, $\mathcal{S}_0 \Rightarrow^{*} \xrightarrow{M} \Rightarrow^{*} \mathcal{S}_0'$ implies $\mathcal{S} \,|\, M \Rrightarrow^{*} \mathcal{S}' \approx_a \mathcal{S}_0'$, and every trailing silent move can be matched on the other side: there is some solution $\mathcal{T}'$ such that $\mathcal{T} \mid M \Rrightarrow^{*} \mathcal{T}' \approx_a \mathcal{S}' \approx_a \mathcal{S}_0'$, and the pair$(\mathcal{S}_0', \mathcal{T}')$ remains within our candidate bisimulation. $\square$

Remark 4.7 *Another choice for Definition 4.3 would require intrusions to be matched by a sequence of join-intrusions on the right-hand-side, followed by the parallel composition of remaining messages, instead of a parallel composition followed by internal reductions. The alternative clause would be:*

- *if* $\mathcal{S} \rightleftharpoons^* \xrightarrow{M} \rightleftharpoons^* \mathcal{S}'$, *then* $\mathcal{T} \rightleftharpoons^* \xrightarrow{M_1} \rightleftharpoons^* \cdots \rightleftharpoons^* \xrightarrow{M_p} \rightleftharpoons^* \mathcal{T}'$ *where* $M \rightleftharpoons^* M_1 \mid \cdots \mid M_p \mid M'$, *and* $\mathcal{S}' \; \Phi \; \mathcal{T}' \mid M'$.

This alternate characterization is easily proven equivalent to the previous one, and it is slightly better for automated proofs, because it does not consider unrelated silent moves after join-intrusions. Note also that it is necessary to allow several steps; otherwise the equivalence becomes strictly finer.

The relations $\approx$ (over open RCHAMs) and $\approx_a$ (over J-open RCHAMs) do match; this is a direct consequence of Proposition 4.1.

Theorem 4.8 $\approx_a = \approx$.

Proof: We prove the inclusions $\approx \subseteq \approx_a$ and $\approx_a \subseteq \approx$; we only consider intrusions, as all other transitions are identical.

Weak bisimulation is an asynchronous bisimulation. Let us consider $\mathcal{S} \approx \mathcal{T}$ in the J-open machine. According to Proposition 4.1(1), for every join-intrusion by $\mathcal{S} \xrightarrow{M} \mathcal{S}'$ there is an equivalent series of transitions in the open machine that lead to the same machine S'.

$$\mathcal{S} \xrightarrow{x_i\langle\widetilde{v_1}\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v_n}\rangle} \rightleftharpoons^* \rightarrow \mathcal{S}'$$

By weak bisimulation, we obtain a mixed series of internal moves and simple intrusions:

$$\mathcal{S} \rightleftharpoons\!\!\Rightarrow^* \xrightarrow{x_i\langle\widetilde{v_1}\rangle} \rightleftharpoons\!\!\Rightarrow^* \cdots \rightleftharpoons\!\!\Rightarrow^* \xrightarrow{x_n\langle\widetilde{v_n}\rangle} \rightleftharpoons\!\!\Rightarrow^* \mathcal{S}'$$

Note, however, that in the open machine we can always perform all intrusions before internal reductions (the extruded variables being unchanged). Therefore we can exhibit the required transitions from $\mathcal{S} \mid M$.

$$\mathcal{S} \rightleftharpoons^* \xrightarrow{x_i\langle\widetilde{v_1}\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v_n}\rangle} \rightleftharpoons^* \mathcal{S} \mid M \rightleftharpoons\!\!\Rightarrow^* \mathcal{S}'$$

Asynchronous bisimulation is a weak bisimulation. Suppose that $\mathcal{S} \approx_a \mathcal{T}$, and that $\mathcal{S} \xrightarrow{x\langle\widetilde{v}\rangle} \mathcal{S}'$. Then, by definition of Rule (Int), $x \in \mathsf{xv}[\mathcal{S}]$ and $\mathcal{S}' = \mathcal{S} \mid x\langle\widetilde{u}\rangle$. By asynchronous bisimulation $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[T]$, thus $x \in \mathsf{xv}[T]$ and $\mathcal{T} \xrightarrow{x\langle\widetilde{c}\rangle} \mathcal{T} \mid x\langle\widetilde{v}\rangle$. We conclude by Proposition 4.5. ∎

4.3 Ground bisimulations

As first observed in the π-calculus, asynchrony brings another interesting property as regards the number of reductions to consider: the ground variant of bisimulations coincide with the basic ones. In early-style semantics, this variant is obtained by restricting the intrusions to labels that convey fresh, pairwise-distinct variables. This is important to reduce the size of the models, because these variables can be treated as constants, hence the name. We refer to [1] for a thorough discussion.

This property carries over the join-calculus, thus providing proof techniques with, for every chemical solution, exactly one intrusion per extruded name when using weak bisimulation, and one intrusion per "active" join-pattern when using asynchronous bisimulation.

PROPOSITION 4.9 *Let $\approx_g$ and $\approx_{a,g}$ be the bisimulations obtained by restricting intrusion labels in the definitions of $\approx$ and $\approx_a$ to $x\langle\widetilde{v}\rangle$ and $x_1\langle\widetilde{v_1}\rangle \mid \cdots \mid x_m\langle\widetilde{v_m}\rangle$, where the x, x_i are pairwise distinct, extruded names and the $\widetilde{v}$, $\widetilde{v_i}$ are pairwise distinct fresh names with regards to the initial chemical solutions. Then we have:*

$$\approx_g \,=\, \approx \,=\, \approx_a \,=\, \approx_{a,g}$$

Proof: This is an easy corollary of the stability of transitions through global renaming (Lemma 3.2); we prove $\approx_g \,=\, \approx$, and we obtain $\approx_g \,=\, \approx_{a,g}$ with the same proof than for Theorem 4.8. By definition $\approx \,\subseteq\, \approx_g$. To establish the converse inclusion, we remark that $\approx_g$ is a weak bisimulation up-to renaming. Let $\mathcal{S} \approx_g \mathcal{T}$. The problematic transition is intrusion with an arbitrary label, that may contain free and/or extruded names.

$$\mathcal{S} \xrightarrow{x\langle\widetilde{w}\rangle} \mathcal{S}'$$

The definition of $\approx_g$ tells nothing about such intrusions; nonetheless, the label can be written $x\langle\widetilde{v}\rho\rangle$ where the v_i's are distinct, fresh names, and another intrusion labeled $x\langle\widetilde{v}\rangle$ is enabled and handled by the ground definition. On the other side, we obtain a series of ground transitions, which are preserved by the renaming ρ:

$$\begin{array}{llll} \mathcal{T} & \Rrightarrow^* x\langle\widetilde{v}\rangle \Rrightarrow^* & \mathcal{T}' \approx_g \mathcal{S}' \\ \mathcal{T}\rho & \Rrightarrow^* x\langle\widetilde{w}\rangle \Rrightarrow^* & \mathcal{T}'\rho \end{array}$$

We conclude by Lemma 3.3(4). ∎

5 The discriminating power of matching

Our comparison with barbed congruence in Section 3.5 suggests that weak bisimulation is more discriminating because of name-testing. For instance, output labels distinguish newly extruded names from free names, while contexts of the barbed congruence cannot. Nevertheless, barbed congruence can easily be equipped with such a discriminating power, provided it can use richer contexts. To this end, we extend the syntax of the join-calculus with the standard name-matching prefix of [14].

$$A \stackrel{\text{def}}{=} \cdots \mid [x=y]\, A \qquad\qquad P \stackrel{\text{def}}{=} \cdots \mid [x=y]\, P$$

Accordingly, we extend our chemical machines with a new reduction rule.

$$(\textsc{Match}) \qquad\qquad \Vdash [x=x]\, A \rightarrow \Vdash A$$

Observe that barbed congruence can now use the context $C[\,] = \mathsf{def}\ x\langle y\rangle \triangleright [y=z]\, a\langle\rangle\ \mathsf{in}\ [\,]$ to separate $x\langle z\rangle$ from $\mathsf{def}\ u\langle v\rangle \triangleright z\langle v\rangle\ \mathsf{in}\ x\langle u\rangle$.

The usual drawback of name matching is that global renamings don't preserve weak bisimulation anymore: Corollary 3.4 is false in the extended calculus, and for instance $0 \approx [x=y]\, x\langle\rangle$, while after applying the renaming $\{^x/_y\}$, $0 \not\approx [x=x]\, x\langle\rangle$. Accordingly, bisimulation is not a congruence anymore; it is easy to check that the context $C[\,] \stackrel{\text{def}}{=} \mathsf{def}\ z\langle x, y\rangle \triangleright [\,]\ \mathsf{in}\ z\langle u, u\rangle$ separates the two processes above.

In order to retain the full congruence property, we can consider the coarsest equivalence contained into $\approx$ and closed under renaming. It is possible to prove, in the same style of Theorem 3.5, that this equivalence is indeed a congruence; however this equivalence is not a bisimulation.

Alternatively, we can consider equivalences that are not congruences. To this aim, we define the set of *evaluation contexts* $C[\,]$ with the following grammar, up-to $\rightleftharpoons^*$:

$$C[\,] \stackrel{\text{def}}{=} [\,] \ \mid\ C[\,]\,|\,P \ \mid\ \mathsf{def}\ D\ \mathsf{in}\ C[\,]\,.$$

Lemma 5.1 *$\approx$ is a congruence with respect to evaluation contexts.*

Proof: We prove that the relation

$$\Phi = \{(\mathcal{D} \Vdash_S \mathcal{P}, A),\ (\mathcal{D} \Vdash_S \mathcal{P}, B) \text{ such that } A \approx B\}$$

is a weak bisimulation up-to restriction and structural equivalence.

Unrelated silent steps, extrusions, and intrusions are clearly the same. The interesting case is a silent step $\mathcal{S} \rightarrow \mathcal{S}'$ that uses a rule in $\mathcal{D}$ and that consumes some messages sent by A. Let $x_i\langle\widetilde{v_i}\rangle$ be the messages of A that are used in this reduction. We must have:

$$A \rightleftharpoons^* \xrightarrow{S_1 \overline{x_1}\langle\widetilde{v_1}\rangle} \cdots \xrightarrow{S_n \overline{x_n}\langle\widetilde{v_n}\rangle} A'$$

By bisimulation, and after gathering all the extrusions at the end, we obtain:

$$B \Longrightarrow^* \xrightarrow{S_1 \overline{x_1}\langle\widetilde{v_1}\rangle} \cdots \xrightarrow{S_n \overline{x_n}\langle\widetilde{v_n}\rangle} B' \approx A'$$

We directly report these two series of transitions in context:

$$\mathcal{D} \Vdash_S, \mathcal{P}, A' \mid \prod x_i\langle\widetilde{v_i}\rangle \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}''$$

$$\mathcal{D} \Vdash_S, \mathcal{P}, B' \mid \prod x_i\langle\widetilde{v_i}\rangle \Longrightarrow^* \mathcal{T}'' \rightleftharpoons^* \Phi \rightleftharpoons^* \mathcal{S}''$$

Provided we erase the extraneous extruded names, these are exactly the transitions that we need to simulate the silent move on the right-hand-side:

$$\mathcal{D} \Vdash_S, \mathcal{P}, A \rightarrow \rightleftharpoons^* \mathcal{S}' \rightleftharpoons^* \mathcal{S}'' \setminus \bigcup S_i$$

$$\mathcal{D} \Vdash_S, \mathcal{P}, B \Longrightarrow^* \mathcal{T}' \rightleftharpoons^* \mathcal{T}'' \setminus \bigcup S_i \; .$$

□

We now define barbed equivalence $\dot{\approx}_{be}$ over open join-calculus processes with matching as follows:

$$A \dot{\approx}_{be} B \quad \stackrel{\text{def}}{=} \quad \text{for all evaluation contexts } C[\,],\ C[A] \dot{\approx}_b C[B]$$

We are going to study the coincidence of $\dot{\approx}_{be}$ and $\approx$. This holds at least for *image-finite processes*, namely processes A such that, for every configuration $\mathcal{S}$ yielded by evaluating A and for every label α, the set $\{\mathcal{S}' \mid \mathcal{S} \Longrightarrow^* \xrightarrow{\alpha} \Longrightarrow^* \mathcal{S}'\}$ is finite up to structural equivalence. The (open) join-calculus is not image finite; a simple image infinite process is:

$$\mathsf{def}\ c\langle\rangle \triangleright \Big(x\langle\rangle \oplus (x\langle\rangle \,|\, c\langle\rangle)\Big)\ \mathsf{in}\ c\langle\rangle$$

Bisimulations on image finite processes have an alternative characterization [12]. In particular, let $\mathcal{F}$ be the monotone operator over set of pairs of open reflexive solutions associated with the definition of asynchronous bisimulation, and define: $\stackrel{0}{\approx} = \{\mathcal{S}, \mathcal{T}\}$, $\stackrel{n+1}{\approx} = \mathcal{F}(\stackrel{n}{\approx})$ and $\stackrel{\omega}{\approx} = \bigcap_n \stackrel{n}{\approx}$. It follows from standard results in the theory of process calculi that the relations $\approx$ and $\stackrel{\omega}{\approx}$ coincide for image finite processes.

Therefore we can prove $\stackrel{\omega}{\approx} = \dot{\approx}_{be}$, on image finite processes. The troublesome containment is $\dot{\approx}_{be} \subseteq \stackrel{\omega}{\approx}$ since the other follows by Lemma 5.1. In [1] the proof of this containment for the π-calculus consists of providing a parametric set of contexts, called $R(n, L)$, such that $(L')(P \mid R(n, L)) \dot{\approx}_{be} (L')(Q \mid R(n, L))$ implies $P \stackrel{n}{\approx} Q$. Unfortunately, the contexts $R(n, L)$ add new definitions for the names they receive, and thus do not carry over the join-calculus. This can be fixed, however, because the part of $R(n, L)$ that invalidates locality are there to test communications on defined names that cannot be checked in the join-calculus. Another technical complication arise from the use of a polyadic calculus, as our contexts must take care of the sorts of names.

Theorem 5.2 *For image finite open processes,* $\dot{\approx}_{be} = \approx$*.*

Proof: The containment $\approx \subseteq \dot{\approx}_{be}$ follows by Lemma 5.1. Let us discuss the converse, which is more difficult. Since we consider only image-finite process, $\approx = \approx_\omega$, and we just have to prove that, for all n, $A \dot{\approx}_{be} A'$ implies $A \stackrel{n}{\approx} A'$.

To this purpose, we define a set of contexts

$$\mathsf{C}_n(\kappa_n^\sigma, F, \widetilde{e})[\,] \stackrel{\text{def}}{=} \mathsf{def}\ \mathsf{D}_n^F(\kappa_n^\sigma)\ \mathsf{in}\ [\,] \mid \kappa_n^\sigma\langle\widetilde{e}\rangle$$

where F is a set of names meant as a superset of the free names of the observed process, $\widetilde{e}$ is a tuple of names meant as the extruded names of the observed process, and κ_n^σ is a name meant as a trigger to enter an observing state with a tuple of extruded names of sort σ.

The definition of $\mathsf{D}_n^F(\kappa_n^\sigma)$ relies on a few notational shortenings: we shall write $\widetilde{e},\widetilde{u}\backslash\widetilde{v}$ for the tuple of names that contains first the names in $\widetilde{e}$, then the names in $\widetilde{u}$ that do not appear in $\widetilde{v}$; in this context, σ is the sort of $\widetilde{e}$ and σ' is the sort of the resulting tuple. We also use $[\widetilde{w=v}]$ instead of $[w_1 = v_1][w_2 = v_2]\cdots[w_m = v_m]$. We make use of the families of names κ_n^σ, $\mathtt{tau}_n$, $\mathtt{int}_n^{y\langle\widetilde{z}\rangle}$ and $\mathtt{ext}_n^{x,[\widetilde{w=v}]}$, which we always assume to be fresh and distinct.

$$
\begin{array}{rcl}
\mathsf{D}_n^F(\kappa_n^\sigma) & \stackrel{\mathrm{def}}{=} & \kappa_n^\sigma\langle\widetilde{e}\rangle \;\triangleright\; \kappa_{n-1}^\sigma\langle\widetilde{e}\rangle \mid \left(\mathtt{tau}_n\langle\rangle \oplus \displaystyle\bigoplus_{y\in\{\widetilde{e}\},\,\widetilde{z}\subseteq\{\widetilde{e}\}\cup F} \left(y\langle\widetilde{z}\rangle \mid \mathtt{int}_n^{y\langle\widetilde{z}\rangle}\langle\rangle\right)\right) \\
& \wedge & \displaystyle\bigwedge_{x\in F} x\langle\widetilde{u}\rangle \mid \kappa_n^\sigma\langle\widetilde{e}\rangle \triangleright \bigoplus_{\widetilde{w}\subseteq\widetilde{u},\,\widetilde{v}\subseteq\{\widetilde{e}\}\cup F\cup\widetilde{u}} [\widetilde{w=v}]\left(\mathtt{ext}_n^{x,[\widetilde{w=v}]}\langle\rangle \mid \kappa_{n-1}^{\sigma'}\langle\widetilde{e},\widetilde{u}\setminus\widetilde{v}\rangle\right) \\
& \wedge & \displaystyle\bigwedge_{\sigma'} \mathsf{D}_{n-1}^F(\kappa_{n-1}^{\sigma'}) \\
\mathsf{D}_0^F(\kappa^\sigma) & \stackrel{\mathrm{def}}{=} & \kappa^\sigma\langle\widetilde{u}\rangle \triangleright \mathbf{0}
\end{array}
$$

With an abuse of notation, we will write $C[\mathcal{S}]$, where $C[\;]$ is a context and $\mathcal{S}$ is a open RCHAM. With this we mean the agent $C[A]$ where $\mathcal{S} \rightleftharpoons^* (\emptyset \Vdash A)$. The statement we show is:

> *Let A and B have a same set of extruded names. Let h be the maximum arity of a channel in A and in B. For every n, for every $\widetilde{e} \supseteq \mathsf{xv}[\mathcal{S}]$, for every F, $F \supseteq \mathsf{fv}[A] \cup \mathsf{fv}[B]$ and containing enough fresh names,*
>
> $$\textit{if}\;\; \mathsf{C}_n(\kappa_n^\sigma, F, \widetilde{e})[A] \dot{\approx}_b \mathsf{C}_n(\kappa_n^\sigma, F, \widetilde{e})[B] \;\;\textit{then}\;\; A \stackrel{n}{\approx} B\,.$$

The proof proceeds by induction on n. Let us focus on the inductive step. We prove the above statement for $n = n' + 1$ by inspecting the three cases of transitions in context.

1. $A(\rightleftharpoons^*\rightarrow\rightleftharpoons^*)^*A'$. We must prove the existence of B' such that $B \Rrightarrow^* B'$ and $A' \stackrel{n'}{\approx} B'$. Observe that

$$\mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[A] \;\; \Rrightarrow^*\dot{\approx}_b \;\; \mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e})[B] \mid \mathtt{tau}_{n'+1}\langle\rangle$$

This move is mimicked by

$$
\begin{array}{rcl}
\mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B] & \rightarrow^*\rightleftharpoons^*\dot{\approx}_b & \mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e})[B] \mid \mathtt{tau}_{n'+1}\langle\rangle \\
& \Rrightarrow^* & \mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e})[B'] \mid \mathtt{tau}_{n'+1}\langle\rangle
\end{array}
$$

where B' is such that $\mathsf{C}_n(\kappa_n^\sigma, F, \widetilde{e})[A'] \dot{\approx}_b \mathsf{C}_n(\kappa_n^\sigma, F, \widetilde{e})[B']$. Hence, by inductive hypothesis, it follows that $A' \stackrel{n'}{\approx} B'$.

2. $A(\Rightarrow^* \xrightarrow{\Rightarrow^*})* \xrightarrow{S\overline{x}\langle\widetilde{u}\rangle} (\Rightarrow^* \xrightarrow{\Rightarrow^*}) * A'$. Let $\widetilde{w} = \widetilde{u} \setminus S$ and $\widetilde{e}' = \widetilde{e}, \widetilde{u} \setminus \widetilde{w}$. Observe that the extrusion causes exactly one barb to appear:

$$\mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[A] \quad \Longrightarrow^* \dot{\approx}_b \quad \mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e}')[A'] \mid \mathtt{ext}_{n'+1}^{x,\widetilde{[w=v]}}\langle\rangle$$

Now, since $\mathcal{S} \dot{\approx}_{be} \mathcal{S}'$, there exist a computation of $\mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B]$ that may be factored-out as follows:

$$\begin{array}{rl} \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B] & \Longrightarrow^* \quad \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[\mathcal{R} \Vdash_S \mathcal{M}, x\widetilde{u}] \\ & \longrightarrow \dot{\approx}_b \quad \mathsf{C}_{n'}(\kappa_{n'}^{\sigma'}, F, \widetilde{e}')[\mathcal{R} \Vdash_S \mathcal{M}] \mid \mathtt{ext}_{n'+1}^{x,\widetilde{[w=v]}}\langle\rangle \\ & \Longrightarrow^* \quad \mathsf{C}_{n'}(\kappa_{n'}^{\sigma'}, F, \widetilde{e}')[B'] \mid \mathtt{ext}_{n'+1}^{x,\widetilde{[w=v]}}\langle\rangle \end{array}$$

where B' is such that $\mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e'})[A'] \dot{\approx}_b \mathsf{C}_{n'}(\kappa_{n'}^{\sigma'}, F, \widetilde{e'})[B']$. Again, $A' \stackrel{n'}{\approx} B'$ follows by inductive hypothesis.

3. $A(\Rightarrow^* \rightarrow \Rightarrow^*)^* \xrightarrow{x\langle\widetilde{u}\rangle} (\Rightarrow^* \rightarrow \Rightarrow^*)^* A'$. By Lemma 3.2 we may safely restrict to consider fresh names in $\widetilde{u}$ which are in F but not in $\mathsf{fv}[\mathcal{S}]$. Notice that the intrusion causes the following derivation:

$$\begin{array}{rl} \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[A] & \Longrightarrow^* \quad \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[A \mid x\langle\widetilde{u}\rangle] \mid \mathtt{int}_{n'+1}^{x\langle\widetilde{u}\rangle} \\ & \Longrightarrow^* \dot{\approx}_b \quad \mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e})[A'] \mid \mathtt{int}_{n'+1}^{x\langle\widetilde{u}\rangle} \end{array}$$

By $A \dot{\approx}_{be} B$, there exist a computation of $\mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B]$ that may be factored-out as follows:

$$\begin{array}{rl} \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B] & \Longrightarrow^* \quad \mathsf{C}_{n'+1}(\kappa_{n'+1}^\sigma, F, \widetilde{e})[B \mid x\langle\widetilde{u}\rangle] \mid \mathtt{int}_{n'+1}^{x\langle\widetilde{u}\rangle} \\ & \Longrightarrow^* \dot{\approx}_b \quad \mathsf{C}_{n'}(\kappa_{n'}^{\sigma'}, F, \widetilde{e})[B'] \mid \mathtt{int}_{n'+1}^{x\langle\widetilde{u}\rangle} \end{array}$$

and B' is such that $\mathsf{C}_{n'}(\kappa_{n'}^\sigma, F, \widetilde{e})[A'] \dot{\approx}_b \mathsf{C}_{n'}(\kappa_{n'}^{\sigma'}, F, \widetilde{e})[B']$. Therefore $A' \stackrel{n'}{\approx} B'$ follows by inductive hypothesis. ∎

6 The strong case

Strong bisimulation is a finer equivalence than weak bisimulation, too precise in general but that can be profitably used in proofs (*cf.* weak bisimulation up-to strong bisimulation [20]). We define the strong counterparts of our two labeled equivalences, and we survey their main properties.

DEFINITION 6.1 *A symmetric relation Φ over open solutions is a* strong bisimulation *if, whenever $\mathcal{S}\ \Phi\ \mathcal{T}$, then*

- *if $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ then $\mathcal{T} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$;*
- *if $\mathcal{S} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{S}'$ then $\mathcal{T} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$,*
  for all labels α of shape $x\langle\widetilde{v}\rangle$ or $S\overline{x}\langle\widetilde{v}\rangle$ such that names in S are fresh in $\mathcal{T}$.

Let $\sim$ be the largest strong bisimulation. We shall write $A \sim B$ for $(\emptyset \Vdash A) \sim (\emptyset \Vdash B)$.

Strong bisimulation, unlike its weak variant, is sensitive to the structure of join patterns, and to the number of internal steps. A major advantage is that, in our setting, almost all transition steps commute, which makes strong bisimilarity very easy to establish. Most of the properties of weak bisimulation carry over to the strong case, with similar, simpler proofs.

PROPOSITION 6.2 *1. $\rightleftharpoons^*$ is a strong bisimulation;*

2. strong bisimulation is preserved by global renamings;

3. strong bisimulation is a congruence.

We also recall the associated up-to strong bisimulation proof techniques. A relation Φ is a weak (resp. strong) bisimulation up-to strong bisimulation if it satisfies the definition of weak (resp. strong) bisimulation where the requirement $\mathcal{S}'\ \Phi\ \mathcal{T}'$ on the solution that matches the transition is weakened into $\mathcal{S}' \sim\Phi\sim \mathcal{T}'$.

LEMMA 6.3 *If Φ is a strong bisimulation up-to strong bisimulation, then $\Phi \subseteq \sim$; if Φ is a weak bisimulation up-to strong bisimulation, then $\Phi \subseteq \approx$.*

Proof: Both follow by a simple diagram chasing. ▯

For the same reasons that lead us to define (weak) asynchronous bisimulation, we can also characterize strong bisimulation as an asynchronous equivalence on top of the J-open RCHAM.

DEFINITION 6.4 *A symmetric relation Φ over J-open RCHAMs is a* strong asynchronous bisimulation *if, whenever $\mathcal{S}\ \Phi\ \mathcal{T}$, then*

- $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$
- *if $\mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$ and names in S are fresh in $\mathcal{T}$, then $\mathcal{T} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$;*
- *if $\mathcal{S} \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* \mathcal{S}'$ then $\mathcal{T} \mid M \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{T}'$ and $\mathcal{S}'\ \Phi\ \mathcal{T}'$.*

Let $\sim_a$ be the largest strong asynchronous bisimulation. We shall write $A \sim_a B$ for $(\emptyset \Vdash A) \sim_a (\emptyset \Vdash B)$.

Strong bisimulation and strong asynchronous bisimulation coincide, for the same reasons that in the weak case. We omit the proofs.

PROPOSITION 6.5 $\sim_a = \sim$

7 A comparison with the π-calculus

The join-calculus can be considered as a disciplined version of the asynchronous π-calculus, where the syntax enforces locality in the usage of names. Still, both calculi have a lot in common, and our semantics largely draw upon the bisimulations developed for the π-calculus [14, 15, 20, 8, 1]. In this section, we relate our definitions to previous proposals in the literature, and we compare the equivalences obtained by applying similar definitions to both the join-calculus and the π-calculus.

In the sequel, we use the following grammar for asynchronous π-calculus processes.

$$P \ ::= \ \mathsf{0} \ | \ \overline{x}\langle v\rangle \ | \ x(y).P \ | \ \nu x.P \ | \ !P \ | \ P|P$$

We refer to [3] for the definition of its semantics. We recall the basic reduction step, which matches complementary pairs of emission and reception, and substitutes actual names for the variables in the receiving process:

$$\overline{x}\langle v\rangle \mid x(y).Q \longrightarrow Q\{^y/_v\}$$

Other transitions render intrusion or extrusion of messages, and their labels carry the same information than those we use in the open RCHAM. In particular output labels explicitly mention names that are being extruded.

The join-calculus and the asynchronous π-calculus have the same expressive power, at least up-to barbed congruence [4]. Their treatment of names, however, are quite different. In the open join-calculus, we partition the visible names of a process into names for extrusions (*free names*), which do not have a local definition, and names for intrusions and internal reduction (*extruded names*), which do have a local definition. As a result, interaction with the environment has more structure, and this significantly prunes the number of transitions that are enabled, even before introducing join-message labels.

In the π-calculus, on the contrary, a free name can always be used for intrusion, extrusion and internal reduction. Furthermore, a received name can be re-defined, as in $x(y).y(z).P$. In this respect, the barbed equivalence of Section 3.5 is illuminating. Consider the rough π-calculus encodings of the processes mentioned in its example, namely $\overline{x}\langle z\rangle$ and $\nu u.(!u(v).\overline{z}\langle v\rangle \mid \overline{x}\langle u\rangle)$. These two processes are not barbed congruent, and this can be established using a context that invalidates our locality property. For example, if we choose $C[\cdot] = \nu x, z.(x(a).a(u).\overline{u}\langle\rangle \mid \overline{z}\langle b\rangle \mid [\cdot])$, we have $C[\overline{x}\langle z\rangle] \Downarrow b$ and

$C[\nu u.(!u(v).\overline{z}\langle v\rangle \mid \overline{x}\langle u\rangle)] \not\Downarrow b$. As could be expected, locality restricts the discriminating power of observers.

We now compare the observational semantics for the open join calculus with previous proposals for the asynchronous π-calculus, notably [8] and [1]. In these works, a major issue is to accommodate the semantics inherited from the standard π-calculus [14] to asynchrony. As message-output is not a prefix anymore, emitters in contexts cannot detect whether a message is actually read. Technically, this leads to a special treatment of input actions, either in the definition of transitions or in the definition of bisimulation.

In [8], Honda and Tokoro take as observational semantics the standard notion of weak bisimulation. As a consequence, they are forced to change the intensional semantics. Let us discuss this issue with a paradigmatic example. Take for instance the π-calculus processes $\mathsf{0}$ and $x(u).\,\overline{x}\langle u\rangle$. In order to progress, the process $x(u).\,\overline{x}\langle u\rangle$ has to consume an emission $\overline{x}\langle v\rangle$, thus exhibiting a new emission $\overline{x}\langle v\rangle$. The same happens for the process $\mathsf{0}$ that "takes" $\overline{x}\langle v\rangle$ and "gives" the same $\overline{x}\langle v\rangle$ (just nothing is consumed or produced). Therefore $\mathsf{0}$ and $x(u).\,\overline{x}\langle u\rangle$ should be reasonably equated in an asynchronous regime: indeed, they are barbed congruent. However, standard bisimulation obviously discriminates between $\mathsf{0}$ and $x(u).\,\overline{x}\langle u\rangle$. To cope with this problem, Honda and Tokoro adopt an operational model where asynchrony of communication with the environment is rendered as the total receptiveness of the process. In other words, any message can be intruded from the environment at any moment, using extended structural equivalence in combination with the rule:

$$\text{(INPUT)} \qquad\qquad \mathsf{0} \xrightarrow{x\langle v\rangle} \overline{x}\langle v\rangle\ .$$

The intrusion rule of the open-RCHAM is reminiscent of this kind of operational semantics, with two important differences: (i) Our rule (INT) can be used only with names that have previously been extruded, and there are finitely many of them; conversely, their input rule immediately yields an infinite transition system; and (ii) extraneous inputs in the join calculus are not observable: if they are not used by the process, they can safely be garbage-collected, as we are not faced with the problem of contexts that attempt to read back their emissions.

In [1], Amadio *et al.* take the opposite approach. They keep the standard synchronous semantics, and they modify the notion of bisimulation. For two processes to be bisimilar, they do not require that every input on one side be necessarily simulated by an input of the other side. Rather, they introduce a *delay* clause in bisimulation, by which an input can be simulated by adding a new message in parallel on the other side. This is crucial, for example, for proving that $\mathsf{0}$ is bisimilar to $x(u).\,\overline{x}\langle u\rangle$. Their resulting asynchronous bisimulation offers three advantages: it eliminates the need for total receptiveness, it is consistent with external sum, and it relies on a widely-used semantics.

Our asynchronous bisimulation presented in Section 4 relies on similar motivations — handling asynchrony in the definition of bisimulation with a relaxed clause for intrusions —, but the clause we use is different from theirs: we deal with multiple intrusions, and we allow the simulating process to perform arbitrary internal moves after parallel composition with the intruded messages; in their terminology, this would place our equivalence between 1-bisimulation and asynchronous bisimulation. Notice, also, that the J-open RCHAM equipped with asynchronous bisimulation is not meant to be a standard semantics, but rather a technical device to reduce the branching of the underlying transition system. Arguably, the open RCHAM gives a more intuitive meaning to extruded names.

It is worth mentioning that the clause they use for intrusions in asynchronous bisimulation does not suitably carry over the join-calculus. In our syntax, this clause is

$$\text{if } P \rightleftharpoons^* \xrightarrow{x\langle\widetilde{u}\rangle} \rightleftharpoons^* P' \text{ then } \quad \begin{array}{rl} \textit{either} & Q \rightleftharpoons^* \xrightarrow{x\langle\widetilde{u}\rangle} \rightleftharpoons^* Q' \text{ and } P' \,\Phi\, Q' \\ \textit{or} & Q \rightleftharpoons^* Q' \text{ and } P' \,\Phi\, (Q' \mid x\langle\widetilde{u}\rangle) \end{array}$$

If we naively extend it to our J-open RCHAM model by allowing join-intrusions of several messages, we get in troubles. Indeed, problems arise when only some of the messages are needed on the right-hand-side to perform internal reductions. Take, for instance, the following equivalence:

$$\mathsf{def}_{\{x,y\}}\ x\langle z\rangle \mid y\langle\rangle \triangleright z\langle\rangle \ \mathsf{in}\ x\langle a\rangle \quad\approx\quad \mathsf{def}_{\{x,y\}}\ x\langle z\rangle \mid y'\langle\rangle \triangleright z\langle\rangle \wedge y\langle\rangle \triangleright y'\langle\rangle \ \mathsf{in}\ x\langle a\rangle$$

The two processes are weakly bisimilar, but the latter cannot simulate a double intrusion $x\langle b\rangle \mid y\langle\rangle$ that would commit on emitting the message $b\langle\rangle$ and not $a\langle\rangle$.

From a technical point of view, we often discuss the same, standard properties (stability through renaming, congruence, relation with barbed bisimulation); as we compare our work to the formal treatment of bisimulation in the π-calculus, we observe that locality makes most proofs simpler: it forbids the detection of messages carried on defined names, and reduces the number of cases to consider in the interaction between transitions and global renamings. Noticeably, it also rules out the subtle problems of input and output occurring on the same name. Most complications stem from stability of equivalence under global renamings, because renamings can create new redexes. In the synchronous π-calculus, bisimulations are not stable under substitutions, unless it is directly taken as part of the definition [18]; in the asynchronous π-calculus, the property holds because silent steps can always be split in two opposite transitions. In the join-calculus, global renamings may create new reductions only by substituting extruded variables for free variables, which cannot occur for regular (non-open) processes.

Weak bisimulation and asynchronous bisimulation coincide in the open join-calculus, for simple reasons. In contrast, the correspondence between the bisimulations of [8] and those of [1], discussed in details in the latter, is a delicate issue; it is unclear, for instance, whether both approaches yield the same relation in the weak case.

Independently, several notions of locality somehow related to the join-calculus have also been considered in a pure π-calculus setting. In [2], a fragment of the π-calculus is considered where received names cannot be used for input (i.e., cannot be redefined) and it is proven that this language can be compositionally compiled into a very simple, substitution-free calculus. The condition of transmitting only names with output capabilities is further strengthened into *uniform receptiveness* in [19] by also demanding that names are available in input-replicated form as soon as created. Uniform receptiveness is actually the locality property, which is formulated by syntactic means in join-calculus and by means of a typing system in [19] for the π-calculus.

8 Conclusions

In this paper we have developed the theory of bisimulation for the join-calculus. We started from the initial definitions of [4], which are purely based on internal reductions and observational equivalence; we provided extensions of the RCHAM that encompass intrusions and extrusions, in two different styles, and we studied for each style a relevant definition of bisimulation.

The open RCHAM communicates with the environment through single, asynchronous messages; it is associated with the standard (early weak) bisimulation. The J-open RCHAM features intrusions of several messages in a single step, but has a stronger blocking capability, as it demands that the intruded messages be immediately used (in combination with previous internal messages). The J-open model is associated with asynchronous bisimulation, a variant of bisimulation that relaxes the simulation requirement for intrusions with a clause that delays the actual use of messages, and thus recovers an asynchronous behavior. Asynchronous bisimulation, and more specifically its ground variant, is best suited for proofs, as it relies on the smallest synchronization trees. In contrast with barbed congruence, bisimulation is therefore easy to check, and should be amenable to automated verification by means of the existing algorithms (see, for instance, [21]).

As regards the comparison between the π-calculus and the join-calculus, there are full-abstract encodings up-to barbed congruence [4], but it would be interesting to know what is the status between the open join-calculus equipped with weak bisimulation and the asynchronous π-calculus.

The relation with barbed congruence, and the precise role of name-testing, deserve further investigation. Indeed, numerous useful properties of the join-calculus as a programming language are not directly captured by weak bisimulation, but they may be proved using first an encoding of name-testing, then weak bisimulation techniques. Finally, one may wonder whether our approach carries over the full distributed join-calculus of [5].

Acknowledgments

We thank Jean-Jacques Lévy, Uwe Nestmann, and especially Georges Gonthier for their fruitful comments.

References

[1] Roberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous π-calculus. RR 2913, INRIA, June 1996. A shorter version

appeared in [16].

[2] Michele Boreale. On the expressiveness of internal mobility in name-passing calculi. In Montanari and Sassone [16]. LNCS 1119.

[3] Gérard Boudol. Asynchrony and the π-calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis, 1992.

[4] Cédric Fournet and Georges Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of POPL '96*, pages 372–385. ACM, January 1996.

[5] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In Montanari and Sassone [16]. LNCS 1119.

[6] Cédric Fournet, Cosimo Laneve, Luc Maranget, and Didier Rémy. Implicit typing à la ML for the join-calculus. In Mazurkiewicz and Winkowski [11]. LNCS 1243.

[7] Cédric Fournet and Luc Maranget. The join-calculus language. Available electronically (`http://pauillac.inria.fr/join`), June 1997.

[8] Kohei Honda and Mario Tokoro. On asynchronous communication semantics. In P. Wegner M. Tokoro and O. Nierstrasz, editors, *Proceedings of the ECOOP '91 Workshop on Object-Based Concurrent Computing*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer-Verlag, 1992.

[9] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[10] Cosimo Laneve. May and must testing in the join-calculus. Technical Report UBLCS 96-04, University of Bologna, March 1996. Revised: May 1996.

[11] Antoni Mazurkiewicz and Jòzef Winkowski, editors. *Proceedings of the 8th International Conference on Concurrency Theory*, Warsaw, Poland, 1–4 July 1997. Springer. LNCS 1243.

[12] Robin Milner. *Communication and Concurrency.* Prentice Hall, New York, 1989.

[13] Robin Milner. The polyadic π-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification.* Springer Verlag, 1993.

[14] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes I and II. *Information and Computation*, 100:1 – 40 & 41 – 77, September 1992.

[15] Robin Milner and D. Sangiorgi. Barbed bisimulation. In *Proceedings of 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695, Vienna, 1992. Springer-Verlag.

[16] Ugo Montanari and Vladimiro Sassone, editors. *Proceedings of the 7th International Conference on Concurrency Theory*, Pisa, Italy, 26–29August 1996. Springer. LNCS 1119.

[17] Joachim Parrow and Peter Sjodin. Multiway synchronization verified with coupled simulation. In Rance Cleaveland, editor, *Third International Conference on Concurrency Theory (CONCUR '92, Stony Brook, NY)*, volume 630 of *Lecture Notes in Computer Science*, pages 518–533. Springer-Verlag, 1992.

[18] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996. Earlier version published as Report ECS-LFCS-93-270, University of Edinburgh. An extended abstract appeared in LNCS 715 (Proc. CONCUR'93).

[19] Davide Sangiorgi. The name discipline of uniform receptiveness. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Proceedings of ICALP '97*, volume 1256 of *Lecture Notes in Computer Science*, pages 303–313. Springer-Verlag, 1997. Full version as Technical Report, INRIA Sophia-Antipolis, December 1996.

[20] Davide Sangiorgi and Robin Milner. The problem of "weak bisimulation up to". In W. R. Cleaveland, editor, *Proceedings of CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1992.

[21] B. Victor and F. Moller. The mobility workbench – a tool for the π-calculus. In *CAV'94*, pages 428–440, 1994. LNCS 818.