Bisimulations in the Join-Calculus *

Cédric Fournet † Cosimo Laneve ‡

February 12, 1999

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 discuss the impact of the lexical scope discipline of the join-calculus on its extensional semantics. We propose several formulations of bisimulation and 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

†Microsoft Research, 1 Guildhall Street, Cambridge, U.K. `fournet@microsoft.com`

‡Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy. `laneve@cs.unibo.it`

Contents

1 Introduction

The join-calculus is a recent formalism for modeling mobile systems [13, 15]. 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 scope discipline over the channel names that appear in processes: names can be sent and received, but their input 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 [15, 17]. 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 [16]. In this paper, we show that locality strongly affects the treatment of bisimulation, and provides a semantics simpler than other proposals in the literature.[1]

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 focus on elementary forms of interaction between the process and its environment. 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 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 when 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 [20].

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 are especially convenient, because co-inductive proofs only need to consider single reduction steps instead of traces [24]. *Barbed bisimilarity* has been proposed in [27] as a uniform basis to define sensible behavioral equivalences on different process calculi. This is the approach chosen for the join calculus in [13, 15], 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 [13] have to analyze in details how contexts and pro-

[1] The term locality is a bit overloaded in the literature; here, names are locally defined inasmuch as no external definitions may interfere; this is the original meaning of locality in the chemical semantics of Banatre *et al.* [4], and in other papers on the join-calculus.

cesses get intertwined, and to partition their reductions accordingly. This makes routine or automated checking of these properties problematic.

In this paper, we introduce a labeled operational model for the join calculus and we equip it with the standard *weak bisimulation* [24]. We illustrate the use of purely co-inductive proof techniques on labeled transitions by establishing a series of useful equations in the join-calculus. We also consider alternate definitions of this equivalence and relate weak bisimulation to barbed congruence, showing that the difference between the two semantics 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) [13]. This operational semantics only accounts for the internal evolution of processes of the form $\mathsf{def}\ D\ \mathsf{in}\ P$, by describing how *name definitions* D can be used to synchronize and receive messages sent on these names in P. In order to shape labeled bisimulation upon the join-calculus, we propose a refined operational semantics—the *open* RCHAM— that explicitly models interactions with the environment. Via these interactions, the environment can get acquainted with local names of the process when they are emitted on free names—the *extrusions*—and can later emit on these names—the *intrusions* ("intrusion" has another technical meaning in [26]). 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 of the join-calculus are marked with their extruded names as soon as extrusion occurs. In turn, intrusions are allowed on extruded names only. 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. Informally, extruded names are handled as constants in the input interface of the process.

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; it also contains a message $x\langle v\rangle$. The name x is marked as extruded; the environment can therefore send messages on x to trigger copies of P. However, the environment cannot interfere with the definition of x; in particular, the message $x\langle v\rangle$ cannot be consumed by the environment, and as a consequence, for any observational equivalence, we should have the equation :

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

Weak bisimilation is obtained by applying the standard definition to the open RCHAM. The largest weak bisimulation—weak bisimilarity—is closed by renaming, is a congruence over contexts of the open calculus, and suitably abstracts from the actual structure of the definitions. In particular, weak bisimilarity is insensitive to message buffering, as can be expected from an asynchronous semantics.

The open RCHAM allows the intrusion of messages on previously extruded names, independently of the actual receiving capabilities of the process. This hides the presence of delayed messages from the environment. When the process is known, it is immediate to check whether there is a transition that can consume these messages. Unfortunately, intrusions of messages that are not directly consumed yield extraneous processes that have to be considered in bisimulation proofs.

To reduce the size of our model, we modify 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, thus significantly augmenting the blocking capability of processes. Asynchronous bisimulation explicitly models asynchrony by relaxing the bisimulation requirement for the intrusions of messages. We prove that the largest weak bisimulation and asynchronous bisimulation coincide, which validates the use of asynchronous bisimulations in proofs.

The last characterization of weak bisimulation is given in terms of barbed bisimulation. The barbed congruence of [13] is strictly coarser than weak bisimulation, because contexts of barbed congruence have no internal name-testing capability, while labels of weak bisimulation separate 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 then coincides with weak bisimulation for all processes. This result improves on previous partial results for other mobile process calculi, its proof relies on the technique presented in [14].

Besides their use as proof techniques, our labeled semantics yield a better understanding of multiway synchronization in the join-calculus, and provide a basis for comparing the join-calculus to other process calculi—usually equipped with labeled transition systems and bisimulations—and in particular to the asynchronous π-calculus [10]. 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 such as $\mathsf{0}$ and $x(u).\overline{x}\langle u\rangle$; several remedies are considered in [18, 3]. Our bisimulations for the join-calculus embed similar accommodations, but they yield simpler semantics, mostly because locality constrains interaction with the environment, thus pruning the number of transitions to consider, and also rules out processes where the same message can be both received by the context and consumed by an internal reduction.

The paper is organized as follows. In Section 2 we introduce our language and its model. In Section 3 we define weak bisimulation and study its basic properties. In Section 4 we introduce asynchronous bisimulation and prove that it coincides with weak bisimulation. In Section 5 we give examples of bisimilar processes. In Section 6 we consider standard variants of our equivalences (strong bisimulations, ground bisimulations). In Section 7 we study the impact of name-testing and relate our bisimulations to barbed congruence. In Section 8 we discuss related work for the asynchronous π-calculus. We conclude in Section 9.

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 [13]. We refer to previous works for a discussion of the join-calculus as an adequate model of distributed programming [13, 15, 12].

2.1 Syntax and Scopes

Let $\mathcal{N}$ be a countable set of *names* ranged over by $x, y, u, v, \ldots$; tuples of names are written $u_i{}^{i \in 1..p}$ or simply $\widetilde{u}$. Names can carry fixed-length messages. We assume a recursive sorting discipline on names that avoids arity mismatch, in the style of [25], and consider only well-sorted terms. We refer to other works for a detailed treatment of sorts and arities in the join-calculus [12, 16].

The syntax of the open join-calculus includes open processes A, processes P, definitions D, and join-patterns J defined as follows:

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

A process P can be the inert process $\mathsf{0}$, a message $x\langle \widetilde{u} \rangle$ sent on a name x that carries a tuple of names $\widetilde{u}$, a parallel composition of processes, or a defining process $\mathsf{def}\ D\ \mathsf{in}\ P$ where D defines names that are local to P. 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, these message can be replaced by a copy of the guarded process P where the formal parameters are replaced by the contents of the messages. The only binders of the calculus are join-patterns, but the scope of names appearing in a join-pattern depends on their position: the scope of received names is the guarded process; the scope of names carrying messages is the main process of the definition and, recursively, all guarded processes of the definition.

An open process A is like a process except that it has open definitions at top-level instead of local ones. The open definition $\mathsf{def}_S\ D\ \mathsf{in}\ P$ exhibits a subset S of names defined by D that are visible from the environment: the *extruded names*. We may omit the index set S when it is empty (no extruded names so far) and identify open definitions $\mathsf{def}_\emptyset\ D\ \mathsf{in}\ P$ with local definitions $\mathsf{def}\ D\ \mathsf{in}\ P$. We also shorten $J_1 \triangleright P_1 \wedge \cdots \wedge J_n \triangleright P_n$ into $\bigwedge_{i=1}^n J_i \triangleright P_i$ and $A_1 \,|\, \ldots \,|\, A_n$ into $\prod_{i=1}^n A_i$.

Received names $\mathsf{rv}[J]$, defined names $\mathsf{dv}[D]$, free names $\mathsf{fv}[A]$ or $\mathsf{fv}[D]$ and extruded names $\mathsf{xv}[A]$ are defined by structural induction as follows. As usual, $\uplus$ means disjoint union.

$$
\begin{array}{rcl@{\qquad}rcl}
J:\ \ \mathsf{rv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{u \in \widetilde{v}\} & \mathsf{rv}[J \,|\, J'] & \stackrel{\text{def}}{=} & \mathsf{rv}[J] \uplus \mathsf{rv}[J'] \\
\mathsf{dv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{x\} & \mathsf{dv}[J \,|\, J'] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] \uplus \mathsf{dv}[J'] \\
D: \mathsf{dv}[J \triangleright P] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] & \mathsf{dv}[D \wedge D'] & \stackrel{\text{def}}{=} & \mathsf{dv}[D] \cup \mathsf{dv}[D'] \\
\mathsf{fv}[J \triangleright P] & \stackrel{\text{def}}{=} & \mathsf{dv}[J] \cup (\mathsf{fv}[P] \setminus \mathsf{rv}[J]) & \mathsf{fv}[D \wedge D'] & \stackrel{\text{def}}{=} & \mathsf{fv}[D] \cup \mathsf{fv}[D']
\end{array}
$$

$$
\begin{array}{rcl@{\qquad}rcl}
A:\ \ \mathsf{fv}[A \,|\, A'] & \stackrel{\text{def}}{=} & (\mathsf{fv}[A] \setminus \mathsf{xv}[A']) \cup (\mathsf{fv}[A'] \setminus \mathsf{xv}[A]) & \mathsf{fv}[0] & \stackrel{\text{def}}{=} & \emptyset \\
\mathsf{xv}[A \,|\, A'] & \stackrel{\text{def}}{=} & \mathsf{xv}[A] \uplus \mathsf{xv}[A'] & \mathsf{xv}[0] & \stackrel{\text{def}}{=} & \emptyset \\
\mathsf{fv}[\mathsf{def}_S\, D\ \mathsf{in}\ A] & \stackrel{\text{def}}{=} & (\mathsf{fv}[A] \cup (\mathsf{fv}[D] \setminus \mathsf{xv}[A])) \setminus \mathsf{dv}[D] & \mathsf{fv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \{x, \widetilde{v}\} \\
\mathsf{xv}[\mathsf{def}_S\, D\ \mathsf{in}\ A] & \stackrel{\text{def}}{=} & S \uplus \mathsf{xv}[A] & \mathsf{xv}[x\langle\widetilde{v}\rangle] & \stackrel{\text{def}}{=} & \emptyset
\end{array}
$$

In the whole paper, we assume that all processes meet the following well-formed conditions. (These conditions extend those of [13], and will be preserved by all transitions.)

1. Join-patterns are linear: a variable may appear at most once in every join pattern. Linearity of received variables rules out name-matching. Linearity of defined variables does not affect expressiveness (cf. [12]) but it makes technical developments simpler.

2. Sets of names extruded by different open sub-processes are disjoint—informally these names are independently defined.

3. Open definitions $\mathsf{def}_S\ D\ \mathsf{in}\ P$ always define their extruded names ($S \subseteq \mathsf{dv}[D]$).

We partition names in processes as follows: *visible names* are either *free names* $\mathsf{fv}[A]$ or *extruded names* $\mathsf{xv}[A]$; the other names are called *local names*. As usual, we shall identify α-equivalent terms, i.e. terms that are equal up to a renaming of local names.

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 messages in.

DEFINITION 1 *Let A be an open process; a* global renaming *on A is a partial function from names to names with finite domain that is injective on* $\mathsf{xv}[A]$*.*

We let σ, σ' range over global renamings. Global renamings operate on processes by substituting their visible names. As usual, the substitution may require α-conversion on local names to avoid clashes. Global renamings may map free names onto previously free names, fresh names, or extruded names, but always map extruded names to distinct extruded names.

2.2 Open chemistry

We first explain the dynamic role played by the index S in definitions through an example. 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 and two free names y, z. The message $z\langle x\rangle$ can be consumed by the environment, thus exporting x:

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

(Structural equivalence is omitted for this example.) Once x is known by 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:

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

In order to model the interactions with the environment, we extend the RCHAM model defined in [13]. 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.

DEFINITION 2 *Let the functions* $\mathsf{dv}[\cdot], \mathsf{fv}[\cdot], \mathsf{xv}[\cdot]$ *be extended to multisets of terms componentwise.* Open chemical solutions*, ranged over by $\mathcal{S}, \mathcal{T}, \ldots$, are triples $(\mathcal{D}, S, \mathcal{A})$, written $\mathcal{D} \vdash_S \mathcal{A}$, where*

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

Str-Null	$\vdash_S \mathsf{0}$	$\rightleftharpoons$	$\vdash_S$
Str-Par	$\vdash_S A \mid A'$	$\rightleftharpoons$	$\vdash_S A,\ A'$
Str-And	$D \wedge D' \vdash_S$	$\rightleftharpoons$	$D,\ D' \vdash_S$
Str-Def	$\vdash_S \mathsf{def}_{S'}\ D \mathsf{\ in\ } A$	$\rightleftharpoons$	$D\sigma \vdash_{S \uplus S'} A\sigma$
Red	$J \triangleright P \vdash_S J\rho$	$\longrightarrow$	$J \triangleright P \vdash_S P\rho$
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	$\vdash_{S\cup\{x\}}$	$\xrightarrow{x\langle v_i{}^{i\in 1..p}\rangle}$	$\vdash_{S\cup\{x\}} x\langle v_i{}^{i\in 1..p}\rangle$

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

Str-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 $S' = \{v_i \mid i \in 1..p\} \cap (\mathsf{dv}[\mathcal{D}] \setminus S)$;
Int	$v_i^{i\in 1..p}$ are either free, or fresh, or extruded.

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

Definitions of free and extruded names are lifted to solutions as follows:

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

When no ambiguity arises, we identify the term A and the open solution $\emptyset \vdash_\emptyset A$. We also let $\mathcal{S} \mid P$ abbreviate the open solution $\mathcal{D} \vdash_S \mathcal{A},\ P$, for all open solutions $\mathcal{S} = \mathcal{D} \vdash_S \mathcal{A}$ and processes P such that $\mathsf{fv}[P] \cap \mathsf{dv}[\mathcal{D}] \subseteq S$.

The chemical rules for the open RCHAM are given in Table 1; they define families of transitions between open solutions $\rightharpoonup$, $\leftharpoondown$, $\rightarrow$, and $\xrightarrow{\alpha}$ where α ranges over labels of the form $S\overline{x}\langle\widetilde{v}\rangle$ and $x\langle\widetilde{v}\rangle$. By convention, each chemical rule mentions only the processes and the definitions that are involved in the transition, and the transition applies to every chemical solution $\mathcal{S}$ that contains them.

Let us comment on the rules. The rules Str-Null, Str-Par, and Str-And make parallel composition of processes and conjunction of definitions associative and commutative. The rule Str-Def enforces a lexical scoping discipline with scope-extrusion. These structural rules define inverse transitions $\rightharpoonup$ and $\leftharpoondown$. The single reduction rule Red says when messages can be consumed and replaced with a copy of a guarded process within a definition. These first five rules are those of the RCHAM, except for rule Str-Def that performs some extra bookkeeping on extruded names.

The last two rules model interaction with the context. According to rule Ext, messages emitted on free names can be received by the environment; these messages can export defined names that were not previously known to the environment, thus causing the scope of their definitions to be opened. This is made explicit by the set S' that appears in the label of the transition $\xrightarrow{S'\overline{x}\langle\widetilde{v}\rangle}$. Names in S' are distinct from any name that appears in the interface before the transition; once these names have been extruded they cannot be α-converted anymore, and behave like constants. Our rule resembles the Open rule for restriction in the π-calculus [26], with 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 rule Int enables the intrusion of messages on extruded names only. It can be viewed as a disciplined version of one of the two Input rules proposed by Honda and Tokoro for the asynchronous π-calculus, where intrusion is enabled for all messages [18].

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

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

The original rcham of [13] can be recovered as the restriction of the open rcham that operates on chemical solutions with no extruded variables, and that does not use the rules Ext and Int.

Remark 3 *In its original presentation [13, 12], the join-calculus is equipped with auxiliary labeled transition systems. Their intent, however, is quite different from ours. These transitions systems are used to give an alternative, syntactic description of chemical reduction steps; transitions are labeled by whole reaction rules; these large labels are discarded when the transition applies within a defining process that contains the same reaction rule.*

In contrast, our open rcham *provides an extensional semantics of the calculus; labels are much simpler than definitions, and silent steps are not defined as a combination of labeled transitions and hiding.*

2.3 Basic properties

To conclude the section, we introduce convenient notations and state some elementary facts about terms and their transitions.

We let $\rightleftharpoons$ be the symmetric relation $\rightharpoonup \cup \leftharpoondown$ that gathers all structural moves—its transitive closure $\rightleftharpoons^*$ is called *structural equivalence*—and let $\Rrightarrow$ be the relation $\rightleftharpoons \cup \rightarrow$ that gathers all silent moves.

The following property is inherited from the rcham of [13]. It provides two different kinds of normal forms for open solutions.

Proposition 4 *Every open chemical solution is structurally equivalent to:*

- *a solution that contains only simple reaction rules and messages, of the form*

$$\{\ldots, J_j \triangleright P_j, \ldots\} \vdash_S \{\ldots, x_i\langle\widetilde{u_{ik}}\rangle, \ldots\}$$

  and which is unique up-to α-conversion (a fully-heated solution*).*

- *and to a solution that contains a single open process, of the form*

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

  (a cooled-down solution*).*

The multiset of messages can be further partitioned into three parts: messages on free names, messages on extruded names, and messages on locally-defined names. Informally, all messages on free names are available to the context, and could be exported at once.

Given a set of names S' and a fully heated solution $\mathcal{S} = \mathcal{D} \vdash_{S \uplus S'} \mathcal{M}$, the *restriction* of $\mathcal{S}$ on S is written $\mathcal{S} \setminus S$ and defined as follows:

$$(\mathcal{D} \vdash_{S \uplus S'} \mathcal{M}) \setminus S' \quad \stackrel{\text{def}}{=} \quad \mathcal{D} \vdash_S \mathcal{M}$$

The restriction operator $\setminus S$ is partially defined on all solutions and open processes by applying the restriction to the structurally-equivalent fully-heated solution, provided that every name in S is extruded by the solution.

The next proposition gathers basic facts about interfaces of terms, global renamings and structural equivalence. Its proof is a straightforward consequence of the definitions.

Proposition 5 *Let $\mathcal{S}$ be an open solution.*

1. $\mathsf{xv}[\mathcal{S}]$ *and* $\mathsf{fv}[\mathcal{S}]$ *are disjoint sets.*
2. *Let σ be a global renaming for $\mathcal{S}$. Then* $\mathsf{fv}[\mathcal{S}\sigma] = \mathsf{fv}[\mathcal{S}]\sigma \setminus \mathsf{xv}[\mathcal{S}]\sigma$*.*
3. *Let $\mathcal{S} \rightleftharpoons^* \mathcal{S}'$. Then*
   (a) $\mathsf{fv}[\mathcal{S}] = \mathsf{fv}[\mathcal{S}']$ *and* $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{S}']$*;*
   (b) $\mathcal{S} \setminus S \rightleftharpoons^* \mathcal{S}' \setminus S$*.*
4. *Let $\mathcal{S} \rightarrow \mathcal{S}'$. Then*
   (a) $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{S}']$ *and* $\mathsf{fv}[\mathcal{S}'] \subseteq \mathsf{fv}[\mathcal{S}]$*;*
   (b) $\mathcal{S} \setminus S \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}' \setminus S$*.*

The next proposition relates extrusions and intrusions to internal moves.

Proposition 6 *Let $\mathcal{S}$ be an open solution.*

1. *If $\mathcal{S} \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$, then $\mathcal{S} \rightleftharpoons^* (\mathcal{S}' \mid x\langle\widetilde{v}\rangle) \setminus S$.*

   If $\mathcal{S} \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \Rrightarrow^* \mathcal{S}'$, then $\mathcal{S} \Rrightarrow^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$.

2. *$\mathcal{S} \xrightarrow{x\langle\widetilde{v}\rangle} \mathcal{S}'$ if and only if $x \in \mathsf{xv}[\mathcal{S}]$ and $\mathcal{S}' = \mathcal{S} \mid x\langle\widetilde{v}\rangle$.*

   If $\mathcal{S} \Rrightarrow^* \xrightarrow{x\langle\widetilde{v}\rangle} \mathcal{S}'$, then $\mathcal{S} \xrightarrow{x\langle\widetilde{v}\rangle} \Rrightarrow^* \mathcal{S}'$.

There is a close correspondence between transitions in a chemical solution $\mathcal{S}$ and transitions in $\mathcal{S}\sigma$, where σ is a global renaming for $\mathcal{S}$. The next lemma details this correspondence.

Lemma 7 *Let $\mathcal{S}$ be an open solution and σ be a global renaming for $\mathcal{S}$.*

1. *$\mathcal{S} \rightleftharpoons \mathcal{S}'$ if and only if $\mathcal{S}\sigma \rightleftharpoons \mathcal{S}'\sigma$.*

2. *If $\mathcal{S} \rightarrow \mathcal{T}$ then $\mathcal{S}\sigma \rightarrow \mathcal{T}\sigma$.*

   If $\mathcal{S}\sigma \rightarrow \mathcal{T}$ then $\mathcal{S} \rightleftharpoons^* \xrightarrow{S_1\overline{x_1}\langle\widetilde{v}_1\rangle} \xrightarrow{y_1\langle\widetilde{v}_1\rangle} \dots \xrightarrow{S_n\overline{x_n}\langle\widetilde{v}_n\rangle} \xrightarrow{y_n\langle\widetilde{v}_n\rangle} \rightleftharpoons^* \rightarrow \mathcal{S}'$ where $(\mathcal{S}' \setminus \bigcup_{i=1}^n S_i)\sigma \rightleftharpoons^* \mathcal{T}$ and $x_i\sigma = y_i\sigma$ for $i = 1 \dots n$.

3. *If $\mathcal{S} \xrightarrow{x\langle\widetilde{v}\rangle} \mathcal{T}$ then $\mathcal{S}\sigma \xrightarrow{(x\langle\widetilde{v}\rangle)\sigma} \mathcal{T}\sigma$.*

   If $\mathcal{S}\sigma \xrightarrow{y\langle\widetilde{w}\rangle} \mathcal{T}$ then $\mathcal{S} \xrightarrow{x\langle\widetilde{v}\rangle} \mathcal{S}'$ where $\mathcal{S}'\sigma = \mathcal{T}$ and $(x\langle\widetilde{v}\rangle)\sigma = y\langle\widetilde{w}\rangle$.

4. *Let S be a set of names such that σ does not operate or range over S.*

   If $\mathcal{S} \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \mathcal{T}$ and $x\sigma \in \mathsf{fv}[\mathcal{S}\sigma]$, then $\mathcal{S}\sigma \xrightarrow{S\overline{x\sigma}\langle\widetilde{v}\sigma\rangle} \mathcal{T}\sigma$.

   If $\mathcal{S}\sigma \xrightarrow{S\overline{y}\langle\widetilde{w}\rangle} \mathcal{T}$, then $\mathcal{S} \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \mathcal{S}'$ where $\mathcal{S}'\sigma = \mathcal{T}$ and $(x\langle\widetilde{v}\rangle)\sigma = y\langle\widetilde{w}\rangle$.

Proof: We show in detail only the second part of Case 2, which is the less obvious. The other cases are similar but easier. If $\mathcal{S}\sigma \rightarrow \mathcal{T}$, then by definition of rule Red (and after α-conversion) $\mathcal{S}$ and $\mathcal{T}$ must be of the form

$$\begin{array}{rcl} \mathcal{S} & = & \mathcal{D}, J \triangleright P \vdash_S \mathcal{P}, M \\ \mathcal{T} & = & \mathcal{D}\sigma, J\sigma \triangleright P\sigma \vdash_{S\sigma} \mathcal{P}\sigma, P\sigma\rho \end{array}$$

where M is a parallel composition of messages and ρ is the substitution used in Red to consume the messages $M\sigma = J\sigma\rho$ in $\mathcal{S}\sigma$.

We partition messages in M according to the names carrying the messages. By definition of rule red every such name in $M\sigma$ is defined in $J\sigma$, but before renaming these names may be either free or defined in J: in the case they are free, they are mapped to names defined in $J\sigma$ and extruded in $S\sigma$. Let M'' be the parallel composition of messages in M sent on names defined in J, let n be the number of messages in M sent on free names, and assume that these messages are of the form $x_i\langle\widetilde{v}_i\rangle$. Let also y_i be the unique names in $\mathsf{dv}[J]$ such that $x_i\sigma = y_i\sigma$. For $i = 0 \ldots n$ we write

$$\mathcal{S}_i \quad \stackrel{\text{def}}{=} \quad \mathcal{D}, J \triangleright P \vdash_{S \cup \bigcup_{j=1}^{i-1} S_i} \mathcal{P}, M'', y_1\langle\widetilde{v}_1\rangle, \ldots, y_{i-1}\langle\widetilde{v}_{i-1}\rangle, x_i\langle\widetilde{v}_i\rangle, \ldots, x_n\langle\widetilde{v}_n\rangle$$

where the sets S_i are successively defined as the names in $\widetilde{v}_i$ that are not in the interface of $\mathcal{S}_{i-1}$. Using the structural rule STR-PAR we have $\mathcal{S} \rightleftharpoons^* \mathcal{S}_0$. Applying the transition rules EXT and INT we have $\mathcal{S}_i \xrightarrow{S_i\overline{x_i}\langle\widetilde{v}_i\rangle} \xrightarrow{y_i\langle\widetilde{v}_i\rangle} \mathcal{S}_{i+1}$. Using the structural rule STR-PAR we assemble in $\mathcal{S}_n$ a parallel composition of messages $M' = J\rho'$ where ρ' operates on $\mathsf{rv}[J]$, $M'\sigma = M\sigma$, and $\rho'\sigma = \sigma\rho$. Applying RED the process $P\rho'$ can then be substituted for M', yielding the solution $\mathcal{S}'$ of the lemma. $\Box$

Intrusions and extrusions always occur on disjoint sets of names, but this property is not closed by global renaming. The next lemma describes how intrusions and extrusions may cancel one another after renaming.

LEMMA 8 *Suppose* $\mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \Rrightarrow^* \xrightarrow{y\langle\widetilde{v}\rangle} \mathcal{S}'$, *and let* σ *be a global renaming for* $\mathcal{S}$ *such that* $\sigma(x) = y$ *and* $y \notin S$. *Then we have* $\mathcal{S}\sigma \Rrightarrow^* (\mathcal{S}' \setminus S)\sigma$.

Proof: By applying Proposition 6, for some solution $\mathcal{T}$ we have

$$\mathcal{S} \rightleftharpoons^* (\mathcal{T} \mid x\langle\widetilde{v}\rangle) \setminus S \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \xrightarrow{y\langle\widetilde{v}\rangle} \mathcal{T} \mid y\langle\widetilde{v}\rangle \Rrightarrow^* \mathcal{S}'$$

We carry over these transitions to $\mathcal{S}\sigma$ as follows:

- By applying Lemma 7(1) we have $\mathcal{S}\sigma \rightleftharpoons^* ((\mathcal{T} \mid x\langle\widetilde{v}\rangle) \setminus S)\sigma$.
- By hypothesis, we have $((\mathcal{T} \mid x\langle\widetilde{v}\rangle) \setminus S)\sigma = ((\mathcal{T} \mid y\langle\widetilde{v}\rangle) \setminus S)\sigma$.
- By Proposition 5(3,4) the restriction operator $\setminus S$ commutes with $\Rrightarrow^*$, so $(\mathcal{T} \mid y\langle\widetilde{v}\rangle)\setminus S \Rrightarrow^* \mathcal{S}'\setminus S$ and, by applying Lemma 7(1,2), $((\mathcal{T} \mid y\langle\widetilde{v}\rangle)\setminus S)\sigma \Rrightarrow^* (\mathcal{S}' \setminus S)\sigma$. $\Box$

3 Weak bisimulation

The join-calculus has been "opened" to support the standard notion of bisimulation [24]. This section defines weak bisimulation for open solutions and investigates its properties.

DEFINITION 9 *A relation* $\mathcal{R}$ *on open solutions is a* weak simulation *if, whenever* $\mathcal{S}\ \mathcal{R}\ \mathcal{T}$, *we have*

- *if* $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \Rrightarrow^* \mathcal{T}'$ *and* $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}'$;

- *if* $\mathcal{S} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \Rrightarrow^* \xrightarrow{\alpha} \Rrightarrow^* \mathcal{T}'$ *and* $\mathcal{S}' \mathcal{R} \mathcal{T}'$,
  for all labels α *of shape* $x\langle\widetilde{v}\rangle$ *or* $S\overline{x}\langle\widetilde{v}\rangle$ *such that* $\mathsf{fv}[\mathcal{T}] \cap S = \emptyset$.

A relation $\mathcal{R}$ *is a* weak bisimulation *when both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are weak simulations. We let* $\approx$ *be the largest weak bisimulation.*

As is usual with name mobility, the simulation clause for extrusion does not consider labels that clash with the free names of $\mathcal{T}$. Without this condition, the extrusion of names on one side that are already in the interface on the other side could not be simulated. This standard technicality does not affect the intuitive discriminating power of bisimulation, because names in S can be α-converted before the extrusion.

Following our conventions on processes as solutions, we shall write $A \approx B$ instead of $(\emptyset \vdash_\emptyset A) \approx (\emptyset \vdash_\emptyset B)$. It is possible to tell whether two processes are weakly bisimilar by reasoning on their synchronization tree [24], without the need to exhibit discriminating contexts. For example, $x\langle u\rangle \not\approx x\langle v\rangle$ because each process performs an extrusion with distinct 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 $\{z\}\overline{x}\langle z\rangle$). Examples of processes that are weakly bisimilar are presented in Section 5. For the moment we focus on the basic properties of weak bisimulation.

Remark 10 *Rule* Int *makes weak bisimulation sensitive to the input interface* $\mathsf{xv}[\mathcal{S}]$ *of the solution* $\mathcal{S}$ *(*$\mathcal{S} \approx \mathcal{T}$ *implies* $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$*).*

Proof: Assume $x \in \mathsf{xv}[\mathcal{S}]$ and $\mathcal{S} \rightharpoonup^* \mathcal{S}'$ where $\mathcal{S}'$ is the fully-heated solution obtained by Proposition 4. We have $x \in \mathsf{xv}[\mathcal{S}']$ by Proposition 5(3), so rule Int is enabled in $\mathcal{S}'$ and $\mathcal{S} \rightleftharpoons^* \xrightarrow{x\langle\widetilde{v}\rangle}$ for all fresh names $\widetilde{v}$. The simulation of these transitions yields $\mathcal{T} \Rrightarrow^* \mathcal{T}' \xrightarrow{x\langle\widetilde{v}\rangle} \Rrightarrow^*$, hence $x \in \mathsf{xv}[\mathcal{T}']$ and $x \in \mathsf{xv}[\mathcal{T}]$. □

3.1 Up-to proof techniques

A whole range of "up to proof techniques" are available to reduce the size of the relation to exhibit when proving bisimilarities [24, 27, 32]. The lemma below establishes that our definition of weak bisimulation is robust with respect to reasoning up to structural equivalence, restriction of the input interface, 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 11 *Let* $\mathcal{R}$ *be a relation that satisfies all the bisimulation clauses of Definition 9 after replacing the requirement "*$\mathcal{S}' \mathcal{R} \mathcal{T}'$*" with one of the following weaker requirements:*

1. *(up to structural equivalence)* $\mathcal{S}' \rightleftharpoons^* \mathcal{R} \rightleftharpoons^* \mathcal{T}'$.

2. *(up to weak bisimulation on the right)* $\mathcal{S}' \ \mathcal{R}\!\approx \mathcal{T}'$.

3. *(up to restriction) there are a set of names S and two solutions $\mathcal{S}''$ and $\mathcal{T}''$ such that $\mathcal{S}' \rightleftharpoons^* \mathcal{S}'' \setminus S$, $\mathcal{S}'' \ \mathcal{R} \ \mathcal{T}''$, and $\mathcal{T}' \rightleftharpoons^* \mathcal{T}'' \setminus S$.*

Then we have $\mathcal{R} \subseteq \approx$.

Proof: In each case, we exhibit a bisimulation that contains $\mathcal{R}$:

1. The definition of weak bisimulation already takes into account structural rearrangements, so $\rightleftharpoons^* \mathcal{R} \rightleftharpoons^*$ is clearly a weak bisimulation.

2. The relation $\approx\!\mathcal{R}\!\approx$ is a weak bisimulation; this directly follows from the transitivity of $\approx$. (However, the lemma would not hold if we used $\approx\!\mathcal{R}\!\approx$ instead of $\mathcal{R}\!\approx$ as an even weaker requirement.)

3. We prove that the relation

$$\mathcal{R}\setminus \ \stackrel{\text{def}}{=} \ \left\{(\mathcal{S}\setminus S, \mathcal{T}\setminus S) \mid \mathcal{S} \ \mathcal{R} \ \mathcal{T} \text{ and both restrictions are defined } \right\}$$

is a weak bisimulation up to structural equivalence. All intrusions and internal reductions that are enabled after applying the restriction are also enabled before; and additional internal reductions $\Rrightarrow$ are preserved by the restriction, so we can obtain the required simulation properties by reporting series of transitions obtained by simulation before restriction. We detail the case of extrusions:

Assume $\mathcal{S} \ \mathcal{R} \ \mathcal{T}$ and $\mathcal{S} \setminus S \rightleftharpoons^* \xrightarrow{S'\overline{x}\langle\widetilde{v}\rangle} \mathcal{S}'$. For some $\mathcal{S}''$ we have $\mathcal{S} \rightleftharpoons^* \xrightarrow{(S'\setminus S)\overline{x}\langle\widetilde{v}\rangle} \mathcal{S}''$ and $\mathcal{S}' \rightleftharpoons^* \mathcal{S}'' \setminus (S \setminus S')$. By simulation hypothesis, there is a solution $\mathcal{T}''$ such that $\mathcal{T} \Rrightarrow^* \xrightarrow{(S'\setminus S)\overline{x}\langle\widetilde{v}\rangle} \Rrightarrow^* \mathcal{T}''$, so $\mathcal{T} \setminus S \Rrightarrow^* \xrightarrow{S'\overline{x}\langle\widetilde{v}\rangle} \Rrightarrow^* \mathcal{T}'' \setminus (S \setminus S')$. We let $\mathcal{T}'$ be $\mathcal{T}'' \setminus (S \setminus S')$ and finally obtain $\mathcal{S}' \ \mathcal{R} \setminus \mathcal{T}'$. $\square$

3.2 Renaming and Congruence properties

Weak bisimulation is a congruence. This is not very surprising since our calculus has no operator for external choice. Nevertheless, the proof is not trivial.

As usual, a context of the open join-calculus is an open process with a single process placeholder $[\cdot]$. We write $C[\cdot]$ for a context and $C[P]$ for the process obtained by substituting P for the placeholder. Whenever we apply a context, we implicitly assume that the resulting open process is well-formed. We sometimes use a simpler set of contexts that may bind some names of the interface of a process, but cannot prevent its execution: *evaluation contexts* $C[\,\cdot\,]$ are defined by the following grammar, up to structural equivalence.

$$C[\,\cdot\,] \quad \stackrel{\text{def}}{=} \quad [\,\cdot\,] \mid C[\,\cdot\,]|A \mid \mathsf{def}_S\ D\ \mathsf{in}\ C[\,\cdot\,]$$

We first establish that weak bisimulation is closed by application of evaluation contexts. The full congruence property will require additional properties with respect to global renamings (Lemma 13).

LEMMA 12 *Weak bisimulation is closed by application of evaluation contexts.*

Proof: We show that the relation

$$\mathcal{R} \quad \stackrel{\text{def}}{=} \quad \{(C[A],\, C[B]) \mid A \approx B\}$$

is a weak bisimulation up to restriction and structural rearrangement.

Transitions that does not depend on the process placed in the context $C[\,\cdot\,]$ are in clear correspondence on both sides of $\mathcal{R}$, and yield new pairs of related processes. Likewise, transitions in A or B that are not affected by the context $C[\,\cdot\,]$ are simulated in B or A by hypothesis, yielding new related processes. This leaves us with two cases:

- *Some messages of $C[\,\cdot\,]$ are consumed by a reaction rule of A (or B).* Let $x_1\langle\widetilde{v}_1\rangle \,|\ldots|\, x_n\langle\widetilde{v}_n\rangle$ be these messages. We must have $x_1, \ldots, x_n \in \mathsf{xv}[A]$ and, by applying Proposition 6, the internal move $C[A] \rightleftharpoons^*\rightarrow\rightleftharpoons^* E$ can be described as

$$\begin{array}{ccc} A & \rightleftharpoons^* \xrightarrow{x_1\langle\widetilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v}_n\rangle} \rightleftharpoons^* \rightarrow \rightleftharpoons^* & A' \\ C[\,\cdot\,] & \rightleftharpoons^* & C'[x_1\langle\widetilde{v}_1\rangle \,|\ldots|\, x_n\langle\widetilde{v}_n\rangle \mid \cdot\,] \end{array}$$

  where the latter relation is a structural equivalence that does not depend on the process in the context $C[\,\cdot\,]$, and where $E \rightleftharpoons^* C'[A']$. Since $A \approx B$, the first series of transitions can be simulated by B as

$$B \quad \rightleftharpoons\!\!\!\!\!\rightarrow^* \xrightarrow{x_1\langle\widetilde{v}_1\rangle} \cdots \rightleftharpoons\!\!\!\!\!\rightarrow^* \xrightarrow{x_n\langle\widetilde{v}_n\rangle} \rightleftharpoons\!\!\!\!\!\rightarrow^* \quad B'$$

  with $A' \approx B'$. By applying Proposition 6 again, we obtain $C[B] \rightleftharpoons\!\!\!\!\!\rightarrow^* C'[B']$, and a pair of related processes $C'[A'] \,\mathcal{R}\, C'[B']$.

- *Some messages of A (or B) are consumed by a reaction rule of $C[\,\cdot\,]$.* Let $x_1\langle\widetilde{v}_1\rangle \,|\ldots|\, x_n\langle\widetilde{v}_n\rangle$ be these messages. We have $x_1, \ldots, x_n \in \mathsf{fv}[A]$ and, by applying Proposition 6, the internal move $C[A] \rightleftharpoons^*\rightarrow\rightleftharpoons^* E$ can be described as

$$\begin{array}{ccc} A & \rightleftharpoons^* \xrightarrow{S_1\overline{x_1}\langle\widetilde{v}_1\rangle} \cdots \xrightarrow{S_n\overline{x_n}\langle\widetilde{v}_n\rangle} \rightleftharpoons^* & A' \\ C[x_1\langle\widetilde{v}_1\rangle \,|\ldots|\, x_n\langle\widetilde{v}_n\rangle \mid \cdot\,] & \rightleftharpoons^* \rightarrow \rightleftharpoons^* & C'[\,\cdot\,] \end{array}$$

where the second series applies for all processes within the context that extrude every name in $S \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} S_i$, and where $E = C'[A'] \setminus S$. Since $A \approx B$, the first series can be simulated by B as

$$B \quad \rightleftharpoons^* \xrightarrow{S_1 \overline{x_1} \langle \widetilde{v}_1 \rangle} \cdots \rightleftharpoons^* \xrightarrow{S_n \overline{x_n} \langle \widetilde{v}_n \rangle} \rightleftharpoons^* \quad B'$$

with $A' \approx B'$. By applying Proposition 6 we can defer the extrusions, perform all the internal steps of B within the context $C[\,\cdot\,]$, then perform the internal step from $C[\,\cdot\,]$ to $C'[\,\cdot\,]$. We obtain $C[B] \rightleftharpoons^* C'[B'] \setminus S$ and again we have related processes up to restriction. $\square$

The next lemma is the key to prove that our bisimulation is a full congruence. We also provide a more general lemma that enables us to use global renamings on the fly to simplify bisimulation proofs:

Lemma 13 *Weak bisimulation is closed by global renaming.*

Lemma 14 (bisimulation up to global renaming) *A relation $\mathcal{R}$ is a weak bisimulation up to global renaming when it meets the requirements of Definition 9 after replacing the requirement "$\mathcal{S}' \; \mathcal{R} \; \mathcal{T}'$" with "there is a global renaming σ such that $\mathcal{S}' = \mathcal{S}''\sigma$, $\mathcal{S}'' \; \mathcal{R} \; \mathcal{T}''$, and $\mathcal{T}''\sigma = \mathcal{T}'$. Then $\mathcal{R} \subseteq \approx$.*

Proof: Let $\mathcal{R}$ be a weak bisimulation up to global renaming. We prove that its closure for all valid global renamings

$$\mathcal{R}_\Sigma \quad \stackrel{\text{def}}{=} \quad \{(\mathcal{S}\sigma, \mathcal{T}\sigma) \mid \mathcal{S} \; \mathcal{R} \; \mathcal{T} \text{ and } \sigma \text{ global renaming for } \mathcal{S} \text{ and } \mathcal{T}\}$$

is a weak bisimulation up to restriction. The proof relies on the analysis of the effects of a global renaming on every chemical rule, as described in Lemma 7. Let $\mathcal{S}\sigma \; \mathcal{R}_\Sigma \; \mathcal{T}\sigma$.

Intrusions Suppose $\mathcal{S}\sigma \rightleftharpoons^* \xrightarrow{x\langle \widetilde{v} \rangle} \rightleftharpoons^* \mathcal{S}_1$. By Lemma 7(3, second part), we have $\mathcal{S} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{S}'$ for some α such that $\alpha\sigma = x\langle \widetilde{v} \rangle$ and $\mathcal{S}'$ such that $\mathcal{S}_1 = \mathcal{S}'\sigma$. Since $\mathcal{S} \; \mathcal{R} \; \mathcal{T}$, for some σ' we also have $\mathcal{T} \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* \mathcal{T}''\sigma'$ with $\mathcal{S}' = \mathcal{S}''\sigma'$ and $\mathcal{S}'' \; \mathcal{R} \; \mathcal{T}''$. By Lemma 7(3, first part), this transition yields back $\mathcal{T}\sigma \rightleftharpoons^* \xrightarrow{x\langle \widetilde{v} \rangle} \rightleftharpoons^* (\mathcal{T}''\sigma')\sigma$, and the pair $\mathcal{S}_1 = (\mathcal{S}''\sigma')\sigma \; \mathcal{R}_\Sigma \; (\mathcal{T}''\sigma')\sigma$ meets the bisimulation clause for extrusion.

Extrusions Transitions $\mathcal{S}\sigma \rightleftharpoons^* \xrightarrow{S\overline{x}\langle \widetilde{v} \rangle} \rightleftharpoons^* \mathcal{S}_1$ are handled in the same manner.

Internal moves Suppose $\mathcal{S}\sigma \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}_1$. By Lemma 7(2), we obtain a series of extrusion-intrusion pairs followed by a silent move:

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

where $n \geq 0$ and, for all $i \leq n$, $y_i \in \mathsf{xv}[\mathcal{S}]$ and $x_i\sigma = y_i\sigma$. Let $S \stackrel{\text{def}}{=} \bigcup_{i=1\ldots n} S_i)$. Since $\mathcal{S}\ \mathcal{R}\ \mathcal{T}$, and by composing bisimulation diagrams up to global renamings for each transition above, there is a compound global renaming σ', $\mathcal{S}''$, and $\mathcal{T}''$ such that $\mathcal{S}''\ \mathcal{R}\ \mathcal{T}''$, $\mathcal{S}''\sigma' = \mathcal{S}'$, and

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

To translate these transitions under the global renaming σ, we repeatedly apply Proposition 5, Lemma 8, and Lemma 7(2):

$$\mathcal{T}\sigma \rightleftharpoons^* (\mathcal{T}' \setminus S)\sigma = (\mathcal{T}''\sigma')\sigma \setminus S$$

Since $\mathcal{S}''\ \mathcal{R}\ \mathcal{T}''$, we have $\mathcal{S}_1 = \mathcal{S}'\sigma = (\mathcal{S}''\sigma')\sigma\ \mathcal{R}_\Sigma\ (\mathcal{T}''\sigma')\sigma$, so this pair under the restriction $\setminus S$ meets the bisimulation clause for internal reductions up to restriction. $\square$

Proof of Lemma 13: We reuse the above proof in the special case $\mathcal{R} = \approx$, and conclude by the double inclusion $\approx_\Sigma \subseteq \approx \subseteq \approx_\Sigma$. $\square$

The congruence property follows from these results; its proof is standard.

THEOREM 15 *Weak bisimulation is a congruence.*

Proof: We first establish that $\approx$ is closed by application of contexts of the form $\mathsf{def}_S\ J \triangleright [\,\cdot\,]\ \mathsf{in}\ A$. Let us show that the relation

$$\mathcal{R} \stackrel{\text{def}}{=} \left\{ \Big((\mathcal{D}, J \triangleright P \vdash_S \mathcal{A}), (\mathcal{D}, J \triangleright Q \vdash_S \mathcal{A}) \Big) \mid P \approx Q \right\}$$

is a weak bisimulation up to weak bisimulation on the right. All transitions are in direct correspondence, except for reduction steps that consume messages in $\mathcal{A}$ to trigger the reaction rule $J \triangleright P$ or $J \triangleright Q$.

Let $\mathcal{S}\ \mathcal{R}\ \mathcal{T}$, let $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}_1$ be a reduction that consumes messages $J\sigma$ in $\mathcal{A}$, and assume $\mathcal{A} \rightleftharpoons^* \mathsf{def}_{S'}\ D\ \mathsf{in}\ \mathcal{A}' \mid J\sigma$. We have the reductions

$$\begin{array}{lll} \mathcal{S} & \rightleftharpoons^* \rightarrow \rightleftharpoons^* & \mathcal{D}, D, J \triangleright P \vdash_{S \uplus S'} \mathcal{A}' \mid P\sigma \rightleftharpoons^* \mathcal{S}_1 \\ \mathcal{T} & \rightleftharpoons^* \rightarrow \rightleftharpoons^* & \mathcal{D}, D, J \triangleright Q \vdash_{S \uplus S'} \mathcal{A}' \mid Q\sigma \stackrel{\text{def}}{=} \mathcal{T}_1 \end{array}$$

By definition of $\mathcal{R}$ we have $P \approx Q$, by Lemma 13 $P\sigma \approx Q\sigma$, and thus by Lemma 12 $\mathcal{T}_1 \approx (\mathcal{D}, D, J \triangleright Q \vdash_{S \uplus S'} \mathcal{A}' \mid P\sigma)$. Composing these relations, we have $\mathcal{S}_1 \rightleftharpoons^* \mathcal{R} \approx \mathcal{T}_1$.

We obtain the congruence theorem by structural induction on every context of the open join-calculus, using the above closure property or Lemma 12 at each step. $\square$

4 Asynchronous bisimulation

In order to prove that two processes are bisimilar, a large candidate bisimulation can be a nuisance, as it requires the analysis of numerous transition cases. Unfortunately, weak bisimulations on open chemical solutions are typically rather large. For example, a process with an extruded name has infinitely many derivatives, even if no "real" computation is ever performed. Consider the equivalence:

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

These two processes are bisimilar because their join-pattern cannot be triggered, no matter of the messages the environment may send on x. Still, one is confronted to infinite models on both 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 an 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 after an extrusion $\{x\}\overline{z}\langle x\rangle$, hence trivially bisimilar.

If we apply this refinement with the same labels for input as before, 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 inputs two messages $x\langle\rangle$ and $y\langle\rangle$, then performs a silent step that consumes these two messages 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 become inert too. This suggests the use of *join*-inputs on x and y in new transitions such as

$$x\langle\rangle\,|\,y\langle\rangle\,|\,z\langle\rangle \triangleright P \vdash_{\{x,y\}} z\langle\rangle \xrightarrow{x\langle\rangle\,|\,y\langle\rangle} x\langle\rangle\,|\,y\langle\rangle\,|\,z\langle\rangle \triangleright P \vdash_{\{x,y\}} P$$

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, by replacing the intrusion rule. Unlike the previous rule INT, the new rule INT-J permits intrusion of messages only if they are immediately used to trigger a process in a definition. This is formalized by allowing labels M' that are parallel compositions of messages. If the solution contains a complementary process M such that the combination $M\,|\,M'$ matches the join-pattern of 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. The rule RED is not part of the new chemical semantics because it is subsumed by rule INT-J in the

Rules Str-(Null,Par,And,Def) and 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: $J\rho \rightleftharpoons^* M \mid M'$, $\mathsf{dom}(\rho) = \mathsf{rv}[J]$, $\mathsf{dv}[M'] \subseteq S$, $\mathsf{rv}[M']$ free, fresh or extruded.

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

case $M' = \mathsf{0}$. (Nonetheless, we maintain the distinction between internal moves and proper input moves in the following discussion.)

The same chemical solution now has two different models: for instance, $x\langle\rangle|y\langle\rangle \triangleright P \vdash_{\{x\}}$ has no transition in the J-open RCHAM, while it has infinite series of transitions $\xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \xrightarrow{x\langle\rangle} \cdots$ in the open RCHAM. In the sequel, we shall keep the symbol $\xrightarrow{\alpha}$ for the open RCHAM and use $\xrightarrow{\alpha}_J$ for the J-open RCHAM; we may drop the subscript J when 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 16 *Let $\mathcal{S}$ be an open solution.*

1. *If $\mathcal{S} \xrightarrow{x_1\langle\widetilde{v}_1\rangle \mid \cdots \mid x_n\langle\widetilde{v}_n\rangle}_J \mathcal{S}'$, then $\mathcal{S} \xrightarrow{x_i\langle\widetilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v}_n\rangle} \rightleftharpoons^* \rightarrow \mathcal{S}'$.*
2. *If $\mathcal{S} \mid x\langle\widetilde{u}\rangle \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* \mathcal{T}$ and $x \in \mathsf{xv}[\mathcal{S}]$, then*
   (a) *either $\mathcal{S} \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* \mathcal{S}'$ with $\mathcal{S}' \mid x\langle\widetilde{u}\rangle \rightleftharpoons^* \mathcal{T}$;*
   (b) *or $\mathcal{S} \rightleftharpoons^* \xrightarrow{M \mid x\langle\widetilde{u}\rangle}_J \rightleftharpoons^* \mathcal{T}$.*

All other transitions coincide for the two chemical machines.

Proof: 1. The side-conditions on names that may appear on intrusion labels are the same for the two machines, so we can use rule Int to intrude every message $x_i\langle\widetilde{v}_i\rangle$ of the compound label $x_1\langle\widetilde{v}_1\rangle \mid \cdots \mid x_n\langle\widetilde{v}_n\rangle$ one at a time. Once this is done, we can use the structural rule Str-Par to assemble the parallel composition of messages that matches the join-pattern used in Int-J, and perform a Red transition that uses the same reaction rule to consume these messages and triggers the same process as Int-J.

2. The choice between the two cases depends on whether the message $x\langle\widetilde{u}\rangle$ is consumed by Int-J.

   (a) For any series of reductions $\mathcal{S} \mid x\langle\widetilde{u}\rangle \rightleftharpoons^* \xrightarrow{M} \mathcal{T}$ such that the intrusion does not consume the message $x\langle\widetilde{u}\rangle$, any preliminary structural step that operates on a process containing this message either commutes with the intrusion or is reverted before the intrusion, so we can build another series $\mathcal{S} \mid x\langle\widetilde{u}\rangle \Rightarrow^* \xrightarrow{M} \mathcal{S}' \mid x\langle\widetilde{u}\rangle \Rightarrow^* \mathcal{T}$ in which every step from $\mathcal{S}$ to $\mathcal{S}'$ applies independently of the presence of $x\langle\widetilde{u}\rangle$. This construction provides the required solution $\mathcal{S}'$.

(b) When $x\langle\widetilde{u}\rangle$ is consumed, by definition of INT-J, the transitions of the lemma can be decomposed into

$$\mathcal{S} \Rightarrow^* \mathcal{D} \vdash_S \mathcal{P}, N \mid x\langle\widetilde{u}\rangle \xrightarrow{M} \mathcal{T}' \Rightarrow^* \mathcal{T}$$

where $N \mid x\langle\widetilde{u}\rangle$ is the process consumed by INT-J. We also have the transition $\mathcal{D} \vdash_S \mathcal{P}, N \xrightarrow{M \mid x\langle\widetilde{u}\rangle} \mathcal{T}'$, and as described above we can eliminate preliminary structural steps that affect the message $x\langle\widetilde{u}\rangle$. $\square$

REMARK 17 *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

Next we adapt the definition of weak bisimulation (Definition 9) to the new J-open RCHAM. Consider for instance the two processes:

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

(where $a \notin \mathsf{fv}[R]$). With the initial open RCHAM, the processes P and Q are weakly bisimilar, but with the new J-open RCHAM this is not true anymore, because after emitting on z the process P can input on $x\langle\rangle$ while the process 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 retain an asynchronous semantics—in particular to avoid detection of the structure of join patterns—weak bisimulation must be weakened further on. At least, we should consider that a process simulates a INT-J transition even if it does not immediately consume all of its messages.

Definition 18 *A relation $\mathcal{R}$ on open solutions is an* asynchronous simulation *if, whenever $\mathcal{S}\ \mathcal{R}\ \mathcal{T}$, we have*

- $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$;
- *if* $\mathcal{S} \rightleftharpoons^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \Rrightarrow^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \Rrightarrow^* \mathcal{T}'$ *and* $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}'$ *for all labels* $S\overline{x}\langle\widetilde{v}\rangle$ *such that* $\mathsf{fv}[\mathcal{T}] \cap S = \emptyset$;
- *if* $\mathcal{S} \rightleftharpoons^* \xrightarrow{M} \rightleftharpoons^* \mathcal{S}'$, *then* $\mathcal{T} \mid M \Rrightarrow^* \mathcal{T}'$ *and* $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}'$.

A relation $\mathcal{R}$ is an asynchronous bisimulation *when both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are asynchronous simulations. We let $\approx_a$ be the largest asynchronous bisimulation.*

In the definition of asynchronous bisimulation, it is necessary to require that related solutions have the same extruded names. Otherwise, Remark 10 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 $\vdash_\emptyset$.

Remark 19 *Another choice for Definition 18 would require that every join-intrusion be simulated by a series of join-intrusions, followed by the addition of any remaining messages in parallel to the solution (instead of adding a parallel composition and performing internal steps afterward). 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* $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}' \mid M'$.

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. It is necessary to allow several intrusion steps; otherwise the equivalence becomes strictly finer.

Next we prove that asynchronous bisimilarity and weak bisimilarity coincide. As a first step we need the following:

Proposition 20 *If $\mathcal{S} \approx_a \mathcal{T}$ and $x \in \mathsf{xv}[\mathcal{S}]$, then $\mathcal{S} \mid x\langle\widetilde{v}\rangle \approx_a \mathcal{T} \mid x\langle\widetilde{v}\rangle$.*

Proof: We prove that the relation

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

is an asynchronous bisimulation up to structural equivalence. (This proof technique is valid by definition of asynchronous bisimulation.) We consider only intrusions, all other cases being immediate. Let $\mathcal{S} \mid x\langle\widetilde{v}\rangle \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* \mathcal{S}'$. By Proposition 16, one of the following holds:

(a) $\mathcal{S} \rightleftharpoons^* \xrightarrow{M}_J \rightleftharpoons^* \mathcal{S}''$ with $\mathcal{S}' \rightleftharpoons^* \mathcal{S}'' \mid x\langle\widetilde{v}\rangle$.

Since $\mathcal{S} \approx_a \mathcal{T}$, there is a solution $\mathcal{T}''$ such that $\mathcal{T} \mid M \Rrightarrow^* \mathcal{T}''$ and $\mathcal{S}'' \approx_a \mathcal{T}''$. The same steps apply in a solution that contains the additional message $x\langle\widetilde{v}\rangle$, so $(\mathcal{T} \mid M) \mid x\langle\widetilde{v}\rangle \Rrightarrow^* \mathcal{T}'' \mid x\langle\widetilde{v}\rangle$ with $\mathcal{S}'' \mid x\langle\widetilde{v}\rangle \; \mathcal{R} \; \mathcal{T}'' \mid x\langle\widetilde{v}\rangle$, and thus $\mathcal{S}' \rightleftharpoons^* \mathcal{R} \; \mathcal{T}'' \mid x\langle\widetilde{v}\rangle$.

(b) $\mathcal{S} \rightleftharpoons^* \xrightarrow{M \mid x\langle\widetilde{v}\rangle}_J \rightleftharpoons^* \mathcal{S}'$. Since $\mathcal{S} \approx_a \mathcal{T}$, there is a solution $\mathcal{T}''$ such that $\mathcal{T} \mid (M \mid x\langle\widetilde{v}\rangle) \Rrightarrow^* \mathcal{T}'$ and $\mathcal{S}' \; \mathcal{R} \; \mathcal{T}'$, as required. □

THEOREM 21 $\approx_a = \approx$.

Proof: We prove the inclusions $\approx \subseteq \approx_a$ and $\approx_a \subseteq \approx$ by standard bisimulation proof techniques. We consider only the intrusions, as all other transitions are identical.

Weak bisimulation is an asynchronous bisimulation. Suppose $\mathcal{S} \approx \mathcal{T}$. According to Proposition 16(1), for every join-intrusion $\mathcal{S} \xrightarrow{M} \mathcal{S}'$ with label $M = x_1\langle\widetilde{v}_1\rangle \mid \cdots \mid x_n\langle\widetilde{v}_n\rangle$, in the J-open machine, there is a series of transitions in the open machine that leads to the same solution $\mathcal{S}'$:

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

By weak bisimulation, hypothesis, we obtain a mixed series of internal moves and simple intrusions from $\mathcal{T}$

$$\mathcal{T} \Rrightarrow^* \xrightarrow{x_1\langle\widetilde{v}_1\rangle} \Rrightarrow^* \cdots \Rrightarrow^* \xrightarrow{x_n\langle\widetilde{v}_n\rangle} \Rrightarrow^* \mathcal{T}'$$

with $\mathcal{S}' \approx \mathcal{T}'$. In the open machine we can always perform intrusions before internal reductions: By iterating Proposition 6(2) we get

$$\mathcal{T} \xrightarrow{x_1\langle\widetilde{v}_1\rangle} \cdots \xrightarrow{x_n\langle\widetilde{v}_n\rangle} \rightleftharpoons^* \mathcal{T} \mid M \Rrightarrow^* \mathcal{T}'$$

and we obtain the join-intrusion requirement of Definition 18 for the series of reduction $\mathcal{T} \mid M \Rrightarrow^* \mathcal{T}'$ and the equivalence $\mathcal{S}' \approx \mathcal{T}'$.

Asynchronous bisimulation is a weak bisimulation. Suppose $\mathcal{S} \approx_a \mathcal{T}$ and $\mathcal{S} \rightleftharpoons^* \xrightarrow{x\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{S}'$. By definition of rule INT, $x \in \mathsf{xv}[\mathcal{S}]$ and $\mathcal{S}' \rightleftharpoons^* \mathcal{S} \mid x\langle\widetilde{v}\rangle$. By asynchronous bisimulation hypothesis, $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$, thus $x \in \mathsf{xv}[\mathcal{T}]$ and $\mathcal{T} \rightleftharpoons^* \xrightarrow{x\langle\widetilde{v}\rangle} \rightleftharpoons^* \mathcal{T} \mid x\langle\widetilde{v}\rangle$. We conclude by Proposition 20. □

4.3 Bisimulation up to evaluation context

As an illustration, we show that $P \approx_a Q$ where P and Q are the processes at the beginning of this section. Both P and Q perform the same extrusion, therefore it suffices to prove

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

To this end, we state the bisimulation up to evaluation context proof technique for asynchronous bisimulations. This technique is formalized in [32], and provides an effective tool for establishing equivalences. (This technique does not directly apply to weak bisimulation, because intrusions could always be cancelled by discarding the intruded messages. For instance the two processes $\mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright a\langle\rangle\ \mathsf{in}\ 0$ and $\mathsf{def}_{\{x\}}\ x\langle\rangle \triangleright b\langle\rangle\ \mathsf{in}\ 0$ are not bisimilar, but they are bisimilar up to parallel composition, because the context $x\langle\rangle \,\|[\,\cdot\,]$ can be discarded after intrusion but before the guarded processes are triggered.)

Lemma 22 (up to evaluation contexts) *Let $\mathcal{R}$ be a relation on open processes that meets the definition of asynchronous bisimulation where the two resulting solutions are not directly related by $\mathcal{R}$, but instead structurally equivalent to open processes $C[A]$ and $C[B]$ where $C[\,\cdot\,]$ is an evaluation context and $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}'$. We have $\mathcal{R} \subseteq \approx_a$.*

Proof: Let $\mathcal{R}'$ be the relation that contains all pairs of well-formed processes $(C[A], C[B])$ such that $A\ \mathcal{R}\ B$. We show that $\mathcal{R}'$ is an asynchronous bisimulation up to restriction and structural equivalence, and remark that $\mathcal{R} \subseteq \mathcal{R}' \subseteq \approx_a$.

Every transition that involves only $C[\,\cdot\,]$ is enabled on both sides of $\mathcal{R}'$ (provided that there is no clash with names free in A or B), and yields processes still related by $\mathcal{R}'$. Every intrusion in $C[A]$ on names extruded by A is rendered as a intrusion in A possibly with more messages on the label. Every internal step that consumes messages in A using a rule of $C[\,\cdot\,]$ can be rendered as a series of extrusions from A, which can be simulated from B. Using a restriction, the two resulting processes become related by $\mathcal{R}'$. (extrusions from A on free variables of $C[A]$ are handled similarly). $\square$

Proof of the example: Let $A_n \stackrel{\text{def}}{=} A \mid \prod_1^n a\langle\rangle$ and $B_n \stackrel{\text{def}}{=} B \mid \prod_1^n x\langle\rangle$. We prove that the relation $\mathcal{R} \stackrel{\text{def}}{=} \{(A_n, B_n) \mid n \geq 0\}$ is an asynchronous bisimulation up to evaluation context and structural equivalence.

All the initial transitions of the J-open RCHAM for A_n and B_n are intrusions on x and y:

1. $A_n \rightleftharpoons^* \xrightarrow{x\langle\rangle} \rightleftharpoons^* A_{n+1}$,
2. $B_n \rightleftharpoons^* \xrightarrow{x\langle\rangle \mid y\langle\rangle} \rightleftharpoons^* R \mid B_n$,
3. $A_{n+1} \rightleftharpoons^* \xrightarrow{y\langle\rangle} \rightleftharpoons^* R \mid A_n$,
4. $B_{n+1} \rightleftharpoons^* \xrightarrow{y\langle\rangle} \rightleftharpoons^* R \mid B_n$.

Intrusions 3 and 4 lead to processes in $\mathcal{R}$ up to the context $R\,|\,[\,\cdot\,]$. Intrusions 1 do not correspond to any intrusion in B_n, but nonetheless the asynchronous bisimulation requirement can be met by adding the message $x\langle\rangle$ in parallel to B_n; we obtain two related processes at rank $n+1$. Intrusions 2 do not correspond to single intrusions in A_n, but the bisimulation requirement can be met after adding the two messages in parallel to A_n, since two internal steps can first consume the message $x\langle\rangle$ and release a message $a\langle\rangle$ instead, then consume $a\langle\rangle\,|\,y\langle\rangle$ and trigger R. □

5 Examples

Several equalities turn out to be useful for reasoning about complex processes. We present in this section a collection of simple bisimilarities, together with their proofs and applications. We refer to [12] for more complex applications of labeled bisimulation proofs.

In the join-calculus, whenever a name x is defined by a single clause of the form $x\langle\widetilde{y}\rangle \triangleright P$, the outcome of any message sent on x is determined, independently of the context:

Example 23 (Deterministic reductions)

$$\mathsf{def}_{\{x\}}\; x\langle\widetilde{y}\rangle \triangleright P \;\mathsf{in}\; x\langle\widetilde{u}\rangle \quad\approx\quad P\{^{\widetilde{u}}/_{\widetilde{y}}\} \,|\, \mathsf{def}_{\{x\}}\; x\langle\widetilde{y}\rangle \triangleright P \;\mathsf{in}\; 0$$

This simple equation is especially useful, as it handles most of the reduction steps that occur in standard deterministic encodings, such as encodings of functions or simple data structures.

Proof: Let A be the open process $\mathsf{def}_{\{x\}}\; x\langle\widetilde{y}\rangle \triangleright P \;\mathsf{in}\; 0$, and let $\mathcal{R}$ be the singleton relation $\{(A\,|\,x\langle\widetilde{u}\rangle, A\,|\,P\{^{\widetilde{u}}/_{\widetilde{y}})\}$. We prove that the reflexive closure of $\mathcal{R}$ is an asynchronous bisimulation up to context and structural equivalence. The process $A\,|\,x\langle\widetilde{u}\rangle$ has two transitions: (1) the internal step

$$A\,|\,x\langle\widetilde{u}\rangle \quad \rightleftharpoons^*\rightarrow\rightleftharpoons^* \quad A\,|\,P\{^{\widetilde{u}}/_{\widetilde{y}}\}$$

yields the related process; all transitions of $A\,|\,P\{^{\widetilde{u}}/_{\widetilde{y}}\}$ can thus be simulated by $A\,|\,x\langle\widetilde{u}\rangle$ by composing this internal step and the same transitions, yielding identical processes. (2) The intrusions

$$A\,|\,x\langle\widetilde{u}\rangle \quad \rightleftharpoons^*\xrightarrow{x\langle\widetilde{v}\rangle}\rightleftharpoons^* \quad A\,|\,x\langle\widetilde{u}\rangle\,|\,P\{^{\widetilde{v}}/_{\widetilde{y}}\}$$

are simulated by identical intrusions on the other side; the pair of resulting is obtained from $\mathcal{R}$ by applying the context $[\,\cdot\,]\,|\,P\{^{\widetilde{v}}/_{\widetilde{y}}\}$. □

The following examples show that weak bisimulation is largely insensitive to the shape of join-patterns. To begin with, some straightforward equations allows one to get rid of redundant rules in definitions:

Example 24 (Redundant definitions)

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

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

Similarly, definitions can be simplified whenever some of their locally defined names do not occur anywhere else.

Next we consider the introduction of intermediate buffers; as one would expect in an asynchronous calculus, our semantics is not sensitive to such buffers, either before a join-synchronization or after it.

Example 25 (Buffering before synchronization)

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

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.

Example 26 (Buffering after synchronization)

$$\mathsf{def}_S\ J \triangleright P \wedge D \ \mathsf{in}\ A \quad \approx \quad \mathsf{def}_S\ J \triangleright x\langle\widetilde{v}\rangle \wedge x\langle\widetilde{v}\rangle \triangleright P \wedge D \ \mathsf{in}\ A$$

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

These simple properties of asynchrony can be obtained as corollaries of Example 23, and also directly established. For instance:

Proof of Example 26: Let $\mathcal{R}$ be the relation that contains all pairs of open processes (A_1, A_2) of the form

$$\begin{array}{lcl} A_1 & \stackrel{\text{def}}{=} & \mathsf{def}_S\ J \triangleright P \wedge D \ \mathsf{in}\ A \mid \prod_{\rho\in U} P\rho \\ A_2 & \stackrel{\text{def}}{=} & \mathsf{def}_S\ J \triangleright x\langle\widetilde{v}\rangle \wedge x\langle\widetilde{v}\rangle \triangleright P \wedge D \ \mathsf{in}\ A \mid \prod_{\rho\in U} x\langle\widetilde{v}\rho\rangle \end{array}$$

where U is a finite multiset of substitutions of domain $\widetilde{v}$. We check that $\mathcal{R}$ is a weak bisimulation up to structural equivalence, hence the desired equation in the case U is empty. We detail only transitions that affect U.

Transitions that involve processes $P\rho$ in A_1 are simulated from A_2 by silent steps that consumes the corresponding message $x\langle\widetilde{v}\rho\rangle$ to trigger identical processes $P\rho$ followed by the same transitions. Transitions that consume messages $x\langle\widetilde{v}\rho\rangle$ in A_2 are simulated from A_1 with no transition. In both cases the resulting processes are still related, for a smaller multiset U. decreases. Transitions that use the join-patterns $J \triangleright P$ in A_1 or $J \triangleright x\langle\widetilde{v}\rangle$ in A_2 respectively are handled by adding the substitution used by rule Red to U. $\square$

Note that the buffering of partial join-patterns would not usually preserve bisimulation, because it affects the branching structure of processes. 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. (This well-known problem of *gradual commitment* is further discussed in [12], where some variant of coupled simulations is used to relate the two above processes.)

Another standard property of bisimulation is that it is not possible to observe the internal state of processes. As observed in [13] for the weaker barbed congruence, we have

Example 27 (Adjunction of internal state)

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

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, which 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. To this aim 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 congruence of weak bisimulation, this validates a compilation scheme for the join-calculus that would replace every process with an equivalent process with simpler binders (either receptions or immediate definitions).

6 Related semantics

We briefly relate our labeled equivalences to other notions of bisimulations: barbed bisimulation congruence, strong labeled bisimulations, and weak ground bisimulations.

6.1 Barbed congruence

The main semantics of the join-calculus in [13] is *barbed congruence*. This equivalence was initially studied for the π-calculus as a "reduction-based equivalence" in [27]. We recall its definition below, and refer to the mentioned papers for discussion.

DEFINITION 28 *The* output barb $\Downarrow_x$ *is a predicate over open* RCHAMs *that tests for the potential emission of messages on a free name* x*:*

$$\mathcal{S} \Downarrow_x \quad \stackrel{def}{=} \quad \mathcal{S} \rightleftharpoons\!\!\!\Rightarrow^* C[x\langle\widetilde{v}\rangle] \text{ where } C[\,\cdot\,] \text{ is an evaluation context and } x \text{ is free.}$$

A relation $\mathcal{R}$ *on open solutions is a* barbed simulation *if, whenever* $\mathcal{S}\ \mathcal{R}\ \mathcal{T}$*, we have*

- $\mathsf{xv}[\mathcal{S}] = \mathsf{xv}[\mathcal{T}]$*;*
- *if* $\mathcal{S} \Downarrow_x$ *then* $\mathcal{T} \Downarrow_x$*;*
- *if* $\mathcal{S} \rightleftharpoons^* \rightarrow \rightleftharpoons^* \mathcal{S}'$ *then* $\mathcal{T} \rightleftharpoons\!\!\!\Rightarrow^* \mathcal{T}'$ *and* $\mathcal{S}'\ \mathcal{R}\ \mathcal{T}'$*.*

A relation $\mathcal{R}$ *is a* barbed bisimulation *if both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are barbed simulations. We let* $\dot{\approx}_b$ *be the largest barbed bisimulation over open solutions.*

A barbed congruence *is a barbed bisimulation that is closed by application of open join-calculus contexts. We 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 $\mathsf{def}\ x\langle u\rangle \triangleright u\langle\rangle\ \mathsf{in}\ [\,\cdot\,]$ separates them. Barbed congruence has a finer discriminating power. It yields an interesting observational equivalence, studied in detail in [12].

Since barbed congruence considers only internal steps, the introduction of open processes in the join-calculus provides a convenient notation, but no further equalities among processes. Besides, the use of open contexts does not provide additional discriminative power. Actually, every barbed congruence that relates two open processes can also be expressed as the barbed congruence of two processes with no extruded names, in which all the previously extruded names are explicitly sent to the context using auxiliary messages [12, Chapter 5].

6.2 Weak bisimulation versus barbed congruence

By definition of rule EXT, we have $\mathcal{S} \Downarrow_x$ if and only if $\mathcal{S} \rightleftharpoons\!\!\!\Rightarrow^* \xrightarrow{S\overline{x}\langle\widetilde{v}\rangle} \mathcal{S}'$. Weak bisimulation is thus a barbed bisimulation, and also a congruence (Theorem 15), hence weak bisimulation is 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. On the contrary weak bisimulation separates these two processes because their respective extrusion labels reveal that z is free and u is extruded. Since the contexts of the open join-calculus cannot identify names in messages, more powerful contexts will be required to reconcile the two semantics (see Section 7).

6.3 The strong case

In some cases, weak bisimulation can be replaced by a simpler and finer equivalence that is also sensitive to the number of internal steps. This *strong bisimulation* is obtained by substituting structural rearrangement only ($\rightleftharpoons^*$) for arbitrary series of internal reductions ($\rightleftharpoons\!\!\rightarrow^*$) in our definitions of equivalence. Strong bisimulation is somehow inconsistent with an asynchronous semantics, and is often too discriminant; for instance it allows one to detect the structure of join patterns. Nevertheless it may be put to use as an auxiliary equivalence relation in the proofs of more interesting equations. For example, weak bisimulation up to strong bisimulation (or one of its variations [35, 12]) often yields simpler proofs. We omit the formal treatment of strong bisimulation. Most of our results on weak bisimulation carry over to the strong case, with similar, simpler proofs.

Strong bisimulation can sometimes be characterized by an equational theory. In [3] for instance, this is the case for finite π-calculus processes. To carry over the join-calculus the same programme, one should define the purport of "finite" by cutting out recursive behaviours. This is not easy, since in the join-calculus recursion is combined into the definition operator, together with other crucial features (the creation of new names and the input of messages). Even more difficult seems the axiomatization in the weak case; this would requires a (very) deep analysis outside the scope of this paper.

6.4 Ground bisimulations

As first observed in the π-calculus [18, 3, 9], asynchrony brings another interesting property as regards the number of transitions to consider: the ground variant of bisimulations coincide with the basic ones. In early-style semantics, ground bisimulation is obtained by restricting the intrusions to labels that convey fresh, pairwise-distinct names. This is important to reduce the size of the models.

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" partial join-pattern when using asynchronous bisimulation.

Proposition 29 *Let $\approx_g$ and $\approx_{a,g}$ be the bisimulations obtained by restricting intrusion labels to labels of the form $x\langle\widetilde{v}\rangle$ in Definitions 9 and to labels of the form $x_1\langle\widetilde{v_1}\rangle \mid \ldots \mid x_m\langle\widetilde{v_m}\rangle$ in Definition 18, where the names $\widetilde{v}$, $\widetilde{v_i}$ are pairwise distinct fresh names with respect to the initial chemical solutions. 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 7). We prove $\approx_g \;=\; \approx$. (We obtain $\approx_g \;=\; \approx_{a,g}$ with the same proof of as for Theorem 21.) By definition $\approx \;\subseteq\; \approx_g$. To establish the converse inclusion, we prove that $\approx_g$ is a weak bisimulation up to global renaming (Lemma 13).

Let $\mathcal{S} \approx_g \mathcal{T}$. The only problematic transition is an intrusion whose label contain free and/or extruded names:

$$\mathcal{S} \quad \rightleftharpoons^* \xrightarrow{x\langle\widetilde{w}\rangle} \rightleftharpoons^* \quad \mathcal{S}'$$

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. The ground bisimulation clause now applies and yield transitions from $\mathcal{T}$, and by Lemma 7(1,3) these transitions carry over to $\mathcal{T}\sigma$:

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

□

7 The discriminating power of matching

Our comparison with barbed congruence in Section 6.2 shows 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.

In this section we reconcile the two equivalences by enriching the contexts that are considered in barbed congruence.

7.1 Name matching

In this section only, we extend the syntax of the join-calculus with a name matching operator, in the same style as [26]. [2]

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

[2] We consider this extension as a standard technical trick. Indeed, the expressiveness provided by the name-matching operator is not especially useful in an asynchronous process calculus, in particular when it is used as the core of a distributed programming language. On the contrary, name-matching breaks interesting observational laws.

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

$$\text{MATCH} \qquad \vdash [x=x]\, A \;\rightarrow\; \vdash A$$

A technical drawback of name matching is that global renamings do not preserve weak bisimulation anymore: Lemma 13 becomes false in the extended calculus. For instance $\mathsf{0} \approx [x=y]\, x\langle\rangle$, while after applying the renaming $\{^x/_y\}$, $\mathsf{0} \not\approx [x=x]\, x\langle\rangle$. Accordingly, bisimulation is not a congruence anymore. For instance the context $C[\cdot] \stackrel{\text{def}}{=} \mathsf{def}\ z\langle x, y\rangle \triangleright [\cdot]\ \mathsf{in}\ z\langle u, u\rangle$ separates $\mathsf{0}$ and $[x=y]\, x\langle\rangle$.

In order to retain the full congruence property, we may consider the coarsest equivalence contained into $\approx$ and closed under global renaming. It is possible to prove, in the same style of Theorem 15, that this equivalence is indeed a congruence; however this equivalence is not a bisimulation.

Alternatively, we consider equivalences that are not full congruences. The next lemma expresses that weak bisimilarity retains the congruence property for all evaluation contexts (Lemma 12)—the same proof applies unchanged in the presence of matching.

LEMMA 30 $\approx$ *is a congruence for all evaluation contexts.*

We adapt our definition of barbed congruence accordingly:

DEFINITION 31 Barbed equivalence $\approx_{be}$ *is the largest barbed bisimulation in the open join-calculus with matching that is closed by application of all evaluation contexts of the plain join-calculus with matching.*

Barbed equivalence now separates $x\langle z\rangle$ from $\mathsf{def}\ u\langle v\rangle \triangleright z\langle v\rangle\ \mathsf{in}\ x\langle u\rangle$ by using the context $\mathsf{def}\ x\langle y\rangle \triangleright [y=z]\, a\langle\rangle\ \mathsf{in}\ [\cdot]$, and weak bisimulation is still clearly finer. In the next section we focus on the converse property. (As a corollary of Theorem 32, it turns out that barbed equivalence is also closed by application of evaluation contexts with extruded names.)

7.2 Barbed equivalence is a weak bisimulation

In a process calculus with matching, the coincidence of weak bisimulation and barbed equivalence is not easy to prove or disprove; in the π-calculus for instance it was a long-standing problem [27], closed with both positive and negative results for slightly different definitions of equivalences [14].

Indeed, all direct proofs are difficult. To our knowledge, the only proof that the standard early bisimulation coincides with barbed equivalence appears in Sangiorgi's thesis, for both CCS and the monadic π-calculus [31]. The technique consists of building contexts that test all possible behaviors of a process under bisimulation. This techique requires infinite contexts with infinite numbers of free names and of recursive constants. Such contexts are otherwise never considered in congruence properties, and cannot be expressed using constructs such as guarded replication instead of parameterized recursive constants. In recent works, approximate results for variants of the π-calculus are obtained by using the same technique; since only finite contexts are available the coincidence is established only for image-finite processes (i.e., processes with a finite set of derivatives.)

As discussed in [14, 12], there are actually *two* ways of defining barbed equivalence. In "classical" barbed equivalences, a congruence property is required once, before checking that the two processes are bisimilar. In our definitions 28 and 31, however, congruence and bisimulation closure properties are required at the same time. This technical choice turns out to be essential to obtain a simple proof that weak bisimulation and barbed equivalence coincide. Instead of considering whole synchronization trees, we can focus on single transitions. For every labeled transition, we apply a specific evaluation context that "captures" the label, then disappears up to barbed equivalence. (The corresponding π-calculus proof appears in [14]; further discussion of barbed equivalences, including a proof that the two definitions of barbed equivalence yield the same equivalence appears in [12].)

THEOREM 32 *With a name matching operator, we have* $\approx_{be} = \approx$.

In the following proof we focus on $\approx_{be} \subseteq \approx$, since the converse inclusion follows by Definition 31 and Lemma 30. The problematic case is extrusion, because a context of the join-calculus must define a name in order to detect an output transition on that name; this case is handled by creating a permanent relay for all other messages on that name. Without additional care, however, this relay can be detected through the comparison of names. To tackle this problem, we use a family of contexts that separates two aspects of a name. For every name $x \in \mathcal{N}$, we let

$$R_x[\cdot] \quad \stackrel{\text{def}}{=} \quad \mathsf{def}\ x\langle\widetilde{y}\rangle \triangleright x'\langle\widetilde{y}\rangle\ \mathsf{in}\ v_x\langle x\rangle \,\|[\cdot]$$

where the length of $\widetilde{y}$ matches the arity of x. Assuming $x \in \mathsf{fv}[A]$, the process $R_x[A]$ uses x' as a free name instead of x, and forwards all messages from x to x'. The context should still be able to discriminate whether the process sends the name x or not by using name-matching; this extra capability is supplied by the auxiliary message $v_x\langle x\rangle$. The next proposition describes reductions within contexts $R_x[\,\cdot\,]$:

PROPOSITION 33 *If* $R_x[A] \rightleftharpoons\!\!\!\Rightarrow^* T$ *then for some* A' *we have* $A \rightleftharpoons\!\!\!\Rightarrow^* A'$ *and* $R_x[A'] \approx T$.

Proof: We prove by induction on n that if $R_x[A]\ (\rightleftharpoons^* \longrightarrow \rightleftharpoons^*)^n \approx T$ then $A \rightleftharpoons\!\!\!\Rightarrow^* A'$ and $R_x[A'] \approx T$. The base case is immediate; we distinguish two situations for the inductive case, according to the initial reduction step.

- If this step occurs in A, we apply the induction hypothesis.

- Otherwise this step uses the single rule of the context; it consumes a message $x\langle\widetilde{w}\rangle$ and replaces it with a message $x'\langle\widetilde{w}\rangle$; this step commutes with all subsequent reduction steps and preserves weak bisimulation (Example 23). We apply the induction hypothesis for all other steps and substitute weak bisimulation for the last step. $\Box$

Informally, the contexts $R_x[\,\cdot\,]$ above will be the residuals of contexts that test for labels of the form $\{x\}\overline{y}\langle x\rangle$. We state a lemma that captures the essential property of $R_x[\,\cdot\,]$.

LEMMA 34 (ACCOMODATING THE EXTRUSIONS) *For all open processes A, B such that $x \notin \mathsf{xv}[A] \cup \mathsf{xv}[B]$ and x', v_x are not in the interface of A and B, we have*

$$A \approx_{be} B \quad \textit{if and only if} \quad R_x[A] \approx_{be} R_x[B]$$

Proof: If $A \approx_{be} B$, then the congruence property of $\approx_{be}$ with respect to evaluation contexts immediately yields $R_x[A] \approx_{be} R_x[B]$.

To prove the converse implication, we let

$$\mathcal{R} \stackrel{\text{def}}{=} \left\{ (A,B) \mid R_x[A] \approx_{be} R_x[B] \begin{array}{l} x \notin \mathsf{xv}[A] \cup \mathsf{xv}[B] \\ \text{for some fresh names } x' \text{ and } v_x \end{array} \right\}$$

and establish that $\mathcal{R}$ is a barbed equivalence. Let $A \,\mathcal{R}\, B$.

1. $\mathcal{R}$ is a weak bisimulation for silent steps: if $A \rightleftharpoons^* \rightarrow \rightleftharpoons^* A'$, then $R_x[A] \rightleftharpoons^* \rightarrow \rightleftharpoons^* R_x[A']$. By hypothesis $R_x[A] \approx_{be} R_x[B]$ hence this reduction is simulated by reductions $R_x[B] \Rrightarrow^* T$ with $R_x[A'] \approx_{be} T$. By Proposition 33 we have $B \Rrightarrow^* B'$ and $R_x[B'] \approx_{be} T$. By transitivity $R_x[A'] \approx_{be} R_x[B']$ and thus $A' \,\mathcal{R}\, B'$.

2. $\mathcal{R}$ respects the barbs: if $A \Downarrow_y$ with $y \neq x$, then also $R_x[A] \Downarrow_y$ by using the same reductions; by hypothesis $R_x[B] \Downarrow_y$ and, since $y \notin \{v_x, x'\}$ we must have $B \Downarrow_y$. Similarly, if $A \Downarrow_x$ then $R_x[A] \Downarrow_{x'}$, $R_x[B] \Downarrow_{x'}$, and $B \Downarrow_x$.

3. $\mathcal{R}$ is closed by application of evaluation contexts of the plain join-calculus with matching. Let $C[\,\cdot\,] = \mathsf{def}\ \bigwedge_{i=1}^n J_i \triangleright T_i \ \mathsf{in}\ M \,\|[\,\cdot\,]$ be such a context. We show that $R_x[C[A]] \approx_{be} R_x[C[B]]$ by translating $C[\,\cdot\,]$ to another context $[\![C]\!][\,\cdot\,]$ that binds v_x, receives x on v_x, uses x everywhere except for the definition of x', and re-applies R_x on the outside. We distinguish two cases according to the scope of x:

   - in case C binds x ($x \in \bigcup_{i=1}^n \mathsf{dv}[J_i]$), we use the translation

$$\begin{array}{rcl} [\![C]\!][\,\cdot\,] & \stackrel{\text{def}}{=} & R_x[0] \mid \mathsf{def}\ \bigwedge_{i=1}^n J_i\{^{x'}/_x\} \mid \rho\langle x\rangle \triangleright T_i \mid \rho\langle x\rangle \\ & & \wedge\ v_x\langle x\rangle \triangleright \rho\langle x\rangle \mid M\ \mathsf{in}\ [\,\cdot\,] \end{array}$$

- otherwise, we use the translation

$$[\![C]\!][\,\cdot\,] \stackrel{\text{def}}{=} \begin{array}{l}\mathsf{def}\ w_x\langle y\rangle \triangleright v_x\langle y\rangle\ \mathsf{in}\\ \mathsf{def}\ \bigwedge_{i=1}^{n} J_i \,|\, \rho\langle x\rangle \triangleright T_i \,|\, \rho\langle x\rangle\\ \wedge\ v_x\langle x\rangle \triangleright \rho\langle x\rangle \,|\, w_x\langle x\rangle \,|\, M\ \mathsf{in}\ [\,\cdot\,]\end{array}$$

In each case, we establish $[\![C]\!][R_x[A]] \approx R_x[C[A]]$ by a standard bisimulation argument. We then use the inclusion $\approx\ \subseteq\ \approx_{be}$, apply the congruence property for $[\![C]\!][\,\cdot\,]$ to the hypothesis $R_x[A] \approx_{be} R_x[B]$, and obtain $C[A]\ \mathcal{R}\ C[B]$ by transitivity. □

Proof of Theorem 32: We establish that $\approx_{be}$ is a weak bisimulation. Let us assume that $A \approx_{be} B$. For every kind of transition $A \rightleftharpoons^* \xrightarrow{\alpha} \rightleftharpoons^* A'$ we use a context that specifically consumes this transition, then behaves as the trivial context $[\,\cdot\,]$.

internal step $\rightleftharpoons^* \rightarrow \rightleftharpoons^*$ This is subsumed by the weak bisimulation property of $\approx_{be}$.

intrusion $\rightleftharpoons^* \xrightarrow{x\langle\widetilde{y}\rangle} \rightleftharpoons^*$ Independently of the values $\widetilde{y}$, intrusion is enabled on x if and only if $x \in \mathsf{xv}[A]$, and we have $A \rightleftharpoons^* \xrightarrow{x\langle\widetilde{y}\rangle} \rightleftharpoons^* A' = A \,|\, x\langle\widetilde{y}\rangle$. We apply the congruence property of $\approx_{be}$ for the context $[\,\cdot\,] \,|\, x\langle\widetilde{y}\rangle$.

extrusion $\rightleftharpoons^* \xrightarrow{S\overline{x}\langle y_1,\ldots,y_n\rangle} \rightleftharpoons^*$ Let $m \in 0\ldots n$ be the cardinal of S. Without loss of generality, we assume that the freshly extruded names in S are the first arguments of the message ($S = \{y_1,\ldots,y_m\}$). We also assume that $S \cap (\mathsf{fv}[A] \cup \mathsf{fv}[B]) = \emptyset$. We use the congruence property for the context

$$\begin{array}{rcl}
T & \stackrel{\text{def}}{=} & \{(y_i, z_i) \mid m < i \leq n\}\\
& & \cup\ \{(z_i, z_j) \mid 0 \leq i < j \leq m \text{ and } y_i = y_j\}\\
F & \stackrel{\text{def}}{=} & (\mathsf{fv}[A] \cup \mathsf{fv}[B] \cup \mathsf{xv}[A] \cup \mathsf{xv}[B]) \times S\\
& & \cup\ \{(z_i, z_j) \mid 0 \leq i, j \leq m \text{ and } y_i \neq y_j\}\\
V & \stackrel{\text{def}}{=} & \bigcap_{(y,z)\in T} [y = z]done\langle\rangle \ \Big|\ \prod_{(y,z)\in F} [y = z]test\langle\rangle\\
E[\,\cdot\,] & \stackrel{\text{def}}{=} & \mathsf{def}\ x\langle\widetilde{z}\rangle \,|\, grab\langle\rangle \triangleright V\\
& & \wedge\ x\langle\widetilde{z}\rangle \,|\, done\langle\rangle \triangleright x'\langle\widetilde{z}\rangle \,|\, done\langle\rangle\\
& & \wedge\ once\langle\rangle \triangleright test\langle\rangle\\
& & \wedge\ once\langle\rangle \,|\, done\langle\rangle \triangleright done\langle\rangle\\
& & \mathsf{in}\ once\langle\rangle \,|\, grab\langle\rangle \,|\, v_x\langle x\rangle \,|[\,\cdot\,]
\end{array}$$

where the names $grab$, $once$, $done$, $test$, and v_x are fresh, and where the notation $\bigcap$ abbreviates a sequence of tests. Informally, $E[\,\cdot\,]$ receives a message on x, checks that its arguments match the expected label using the process V, then behaves like $R_x[\,\cdot\,]$. The parameters are bound to names $\widetilde{z}$, then compared to the names expected names on the label. The finite set $T \subset \mathcal{N} \times \mathcal{N}$ gathers all pairs of names that must coincide: previously-known names and repeated fresh names. The finite set $F \subset \mathcal{N} \times \mathcal{N}$ gathers all pairs of names that must be distinct: freshly-extruded names are distinct from any previously visible name, and pairwise distinct unless syntactically the same on the label.

- if $A \Rightarrow^* \xrightarrow{S\overline{x}\langle y_1,\ldots,y_n\rangle} \Rightarrow^* A'$, then $E[A] \Rightarrow^* \approx R_x[A']$.
  We use the series of $n-m+2$ reductions that consumes $x\langle\widetilde{y}\rangle \mid \mathit{grab}\langle\rangle$, passes the series of positive tests T in V—thus releasing the message $\mathit{done}\langle\rangle$—then consumes $\mathit{once}\langle\rangle \mid \mathit{done}\langle\rangle$—thus removing the barb $\Downarrow_{test}$. Using structural rearrangement, what remains of the context can be written
  $$\begin{aligned} E'[\,\cdot\,] \stackrel{\text{def}}{=}\; & \mathsf{def}\ x\langle\widetilde{z}\rangle \mid \mathit{done}\langle\rangle \triangleright x'\langle\widetilde{z}\rangle \mid \mathit{done}\langle\rangle \wedge D\ \mathsf{in} \\ & \mathit{done}\langle\rangle \mid v_x\langle x\rangle \parallel [\,\cdot\,] \mid \textstyle\prod_{(y,z)\in F}([y=z]\mathit{test}\langle\rangle) \end{aligned}$$
  where D and $\prod_{(y,z)\in F}([y=z]\mathit{test}\langle\rangle)$ are inert. In particular we have $E'[A'] \approx R_x[A']$ for any A' where the names grab, once, done, and v_x are fresh.
- if $E[B] \Rightarrow^* U$ and $U \not\Downarrow_{test}$, then $B \Rightarrow^* \xrightarrow{S\overline{x}\langle y_1,\ldots,y_n\rangle} \Rightarrow^* B'$ with $R_x[B'] \approx_{be} U$.
  In order to get rid of the barb $\Downarrow_{test}$, every step in the series of derivations detailed above is required to emit the message $\mathit{done}\langle\rangle$. Let $x\langle\widetilde{z}\rangle$ be the first message received by $E[\,\cdot\,]$. We decompose the reductions leading to U as follows. Before the first reception on x, all reductions are internal to B; after this receptions, all reductions are either internal steps in the derivative of B, internal steps to $E[\,\cdot\,]$ exactly as described above, or further receptions on x in $E[\,\cdot\,]$. These latter receptions are deterministic and do not affect barbed congruence.
  Since the barb $\Downarrow_{test}$ disappears, every comparison in the series on the first line of V has succeeded, which ensures $y_i = z_i$ for $i = m+1\ldots n$. Besides, no comparison in the parallel composition on the second line of V may succeed, as this would reintroduce a $\mathit{test}\langle\rangle$ message, hence the names $z_1,\ldots,z_m$ are all fresh names and we can perform α-conversion before the reduction to enforce $y_i = z_i$ for $i = 1\ldots m$.
  We write $B \Rightarrow^* C[x\langle\widetilde{y}\rangle]$ for the series of reductions internal to B, and we choose B' such that $C[x\langle\widetilde{y}\rangle] \Rightarrow^* \xrightarrow{S\overline{x}\langle y_1,\ldots,y_n\rangle} \Rightarrow^* B'$.

Let us assume that $A \approx_{be} B$ and $A \Rightarrow^* \xrightarrow{S\overline{x}\langle y_1,\ldots,y_n\rangle} \Rightarrow^* A'$. By congruence property $E[A] \approx_{be} E[B]$. By barbed equivalence, the reductions $E[A] \rightleftharpoons^* E'[A']$ must be simulated by some reductions $E[B] \rightleftharpoons^* U$ with $E'[A'] \approx_{be} U$. Since $E'[A] \not\Downarrow_{test}$, $U \not\Downarrow_{test}$ and thus $R_x[A'] \approx_{be} R_x[B']$. By Lemma 34, this entails $A' \approx_{be} B'$, which closes the required bisimulation diagram for extrusions. $\square$

Recently Merro and Sangiorgi have proposed an approach to reconcile labeled bisimulation and barbed congruence, in the absence of name matching, for a new typed variant of the asynchonous π-calculus very similar to the join-calculus [23]. In their definition of labeled bisimulation, the extrusion clause requires that either the extruded name be the same on both sides of the bisimulation or that they can be made indistinguishable by adding a relaying process on the fly. They prove that in their calculus this coarser bisimulation does coincide with barbed equivalence for image-finite processes.

8 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 [26, 27, 35, 18, 3]. 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.

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

We refer to [10] 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:

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

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 [13]. Their treatments 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 less transitions, 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 6.2 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 the context

$$C[\cdot] \quad \stackrel{\text{def}}{=} \quad \nu x, z.(x(a).a(u).\overline{u}\langle\rangle \mid \overline{z}\langle b\rangle \mid [\cdot])$$

we obtain that $C[\overline{x}\langle z\rangle] \Downarrow_b$ 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 [18] and [3]. In these works, a major issue is to accommodate the semantics inherited from the standard π-calculus [26] 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 [18], Honda and Tokoro take as observational semantics the standard notion of weak bisimulation. As a consequence, they are forced to change the intensional semantics. 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:

$$\textsc{Input} \qquad\qquad \mathsf{0} \xrightarrow{x\langle v\rangle} \overline{x}\langle v\rangle$$

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 [3], 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

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

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:

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

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 global 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 sometimes 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 at the same name. Most complications stem from stability of equivalence under global renamings, because these renamings can create new redexes. In the synchronous π-calculus, bisimulations are not closed under substitutions, unless it is directly taken as part of the definition [33]; in the asynchronous π-calculus, the property holds because silent steps can always be split in two complementary and independent transitions. In the join-calculus, global renamings may create new reductions only by substituting extruded names for free names, which cannot occur for plain (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 [18] and those of [3], 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] the π-calculus is equipped with a typing system that bans multiple receptors of a same message. It is proved that this π-calculus with unique receptors is expressive enough to simulate the asynchronous π-calculus. In [6], 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 [34] 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 [34] for the π-calculus.

9 Conclusions

In this paper we have developed the theory of bisimulation for the join-calculus. We started from the initial definitions of [13], 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.

Asynchronous bisimulation, and more specifically its ground variant, is best suited for proofs, as it relies on smaller synchronization trees, but both labeled equivalences are much easier to check than barbed congruence, and should be amenable to automated verification by means of the existing algorithms (see, for instance, [36] and [28]).

As regards the comparison between the π-calculus and the join-calculus, there are fully abstract encodings up to barbed congruence [13] but these encodings are not fully abstract with respect to the stronger weak bisimulation. The reason is that "firewalls" used for protecting encodings from contexts after the translation use forwarders in both directions, and thus crucially rely on the absence of name-testing. In the absence of name-testing in the calculus, this seems unredeemable: we still cannot reveal an internal name to the outside; yet, there is no context that can maintain a one-to-one correspondence between internal names and external forwarders.

If we add full name comparison (match and mismatch) to both calculi, however, it should be possible to build a pair of fully abstract encodings between the resulting asynchronous π-calculus and join-calculus. In fact these encodings are probably simpler than the ones for barbed bisimulation, and the use of open terms of the join-calculus should make the proofs easier. The new translations are like the old ones as regards the basic part and the overall structure of the firewall. However, the firewall now maintains an association table between internal and external names, creating a forwarder and filling a new entry in the table whenever a new name crosses the firewall in either direction. This is the technique which is used in [1].

The precise role of name-testing deserve further investigation. The coincidence of weak bisimilarity and barbed equivalence only holds in the presence of matching, which invalidates useful process equalities. Thus, numerous useful properties of the join-calculus as a programming language are not directly captured by weak bisimulation. Instead, they may be proved by using first an encoding of name-testing, then weak bisimulation techniques. Of course, it would be interesting to find direct, purely co-inductive characterizations of barbed equivalence in the absence of matching. This issue is addressed in [8] in a π-calculus setting, and in [23] in an asynchronous calculus closely related to the join-calculus.

Finally, one may wonder whether our approach carries over the full distributed join-calculus of [15]. The technical device to define bisimulations in the present paper consists of marking transition with meaningful labels, and the information carried by labels can be extended to manifest other properties of interest. A direction has been recently pointed out by Amadio [2] and Hennessy and Riely [30] for coping with failures in distributed systems. Similar results should follow by suitably "opening" the distributed model developed in [15] according to the technique discussed in this paper.

Acknowledgments

Our preliminary results in collaboration with Michele Boreale appeared in [7]. We also thank Jean-Jacques Lévy, Uwe Nestmann, and especially Georges Gonthier for their fruitful comments.

References

[1] Martín Abadi, Cédric Fournet, and Georges Gonthier. Secure implementation of channel abstractions. In LICS '98 [19], pages 105–116.

[2] Roberto M. Amadio. An asynchronous model of locality, failure, and process mobility. In *COORDINATION'97*, volume 1282 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997. also Rapport Interne LIM February 1997, and INRIA Rapport de recherche 3109.

[3] Roberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998. Also INRIA Rapport de recherche 2913, June 1996. An extended abstract appeared in [29].

[4] Jean-Pierre Banâtre and Daniel Le Métayer. The Gamma model and its discipline of programming. *Science of Computer Programming*, 15:55–77, 1990.

[5] Eike Best, editor. *Proceedings of the 4th International Conference on Concurrency Theory (CONCUR '93)*, volume 715 of *Lecture Notes in Computer Science*, Hildesheim, Germany, 1993. Springer-Verlag.

[6] Michele Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theoretical Computer Science*, 195(2):205–226, 1998. An extended abstract appeared in [29], pages 163–178.

[7] Michele Boreale, Cédric Fournet, and Cosimo Laneve. Bisimulations in the join-calculus. In *Proceedings of PROCOMET '98*, pages 68–86. IFIP, Chapman and Hall, June 1998.

[8] Michele Boreale and Davide Sangiorgi. Bisimulation in name-passing calculi without matching. In LICS '98 [19].

[9] Michele Boreale and Davide Sangiorgi. Some congruence properties for π-calculus bisimilarities. *Theoretical Computer Science*, 198(1–2):159–176, 1998.

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

[11] Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors. *24th Colloquium on Automata, Languages and Programming (ICALP '97)*, volume 1256 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[12] Cédric Fournet. *The Join-Calculus: a Calculus for Distributed Mobile Programming*. PhD thesis, Ecole Polytechnique, Palaiseau, 1998. To appear.

[13] 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.

[14] Cédric Fournet and Georges Gonthier. A hierarchy of equivalences for asynchronous calculi (extended abstract). In Larsen et al. [21], pages 844–855.

[15] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In Montanari and Sassone [29], pages 406–421.

[16] Cédric Fournet, Cosimo Laneve, Luc Maranget, and Didier Rémy. Implicit typing à la ML for the join-calculus. In Mazurkiewicz and Winkowski [22], pages 196–212.

[17] Cédric Fournet and Luc Maranget. The join-calculus language (version 1.03 beta). Source distribution and documentation available from `http://join.inria.fr/`, June 1997.

[18] 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.

[19] IEEE. *Thirteenth Symposium on Logic in Computer Science (LICS '98, Indianapolis)*, June 1998.

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

[21] Kim Larsen, Sven Skyum, and Glynn Winskel, editors. *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP '98)*, volume 1443 of *Lecture Notes in Computer Science*, Aalborg, Denmark, July 1998. Springer-Verlag.

[22] Antoni Mazurkiewicz and Jòzef Winkowski, editors. *Proceedings of the 8th International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, Warsaw, Poland, 1–4 July 1997. Springer-Verlag.

[23] Massimo Merro and Davide Sangiorgi. On asynchrony in name-passing calculi. In Larsen et al. [21], pages 856–867.

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

[25] 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. Also appeared as technical report ECS–LFCS–91–180, University of Edinburgh, UK, 1991.

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

[27] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Proceedings of ICALP '92*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695, Vienna, 1992. Springer-Verlag.

[28] Ugo Montanari and Marco Pistore. Checking bisimilarity for finitary π-calculus. In Insup Lee and Scott A. Smolka, editors, *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR '95, Philadelphia)*, volume 962 of *Lecture Notes in Computer Science*, pages 42–56. Springer-Verlag, 1995.

[29] Ugo Montanari and Vladimiro Sassone, editors. *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR '96)*, volume 1119 of *Lecture Notes in Computer Science*, Pisa, Italy, 26–29 August 1996. Springer-Verlag.

[30] James Riely and Matthew Hennessy. Distributed processes and location failures. In Degano et al. [11], pages 471–481. Also Report 2/97, University of Sussex, Brighton, April 1997.

[31] Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* Ph.D. thesis, University of Edinburgh, May 1993. Available as Technical Report CST–99–93, Computer Science Department, University of Edinburgh.

[32] Davide Sangiorgi. On the bisimulation proof method. Revised version of Technical Report ECS–LFCS–94–299, University of Edinburgh, 1994. An extended abstract appears in Proc. of MFCS'95, LNCS 969, 1994. Available electronically at `ftp://ftp.dcs.ed.ac.uk/pub/sad/bis-proof.ps.Z`.

[33] 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 [5].

[34] Davide Sangiorgi. The name discipline of uniform receptiveness. In Degano et al. [11], pages 303–313. Also INRIA Rapport de recherche, December 1996.

[35] 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.

[36] Björn Victor and F. Moller. The mobility workbench – a tool for the π-calculus. In *CAV'94*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.