A Theory of Bisimulation for the π-calculus[*]

Davide Sangiorgi
Laboratory for Foundations of Computer Science
Department of Computer Science, University of Edinburgh
The King's Buildings, Edinburgh EH9 3JZ, U.K.

May 1993

Abstract

We study a new formulation of bisimulation for the π-calculus [MPW92], which we have called *open bisimulation* ($\sim$). In contrast with the previously known bisimilarity equivalences, $\sim$ is preserved by *all* π-calculus operators, including input prefix. The differences among all these equivalences already appear in the sublanguage without name restrictions: Here the definition of $\sim$ can be factorised into a "standard" part which, modulo the different syntax of actions, is the CCS bisimulation, and a part specific to the π-calculus, which requires name instantiation. Attractive features of $\sim$ are: a simple axiomatisation (of the finite terms), with a completeness proof which leads to the construction of *minimal canonical representatives* for the equivalence classes of $\sim$; an "efficient" characterisation, based on a modified transition system. This characterisation seems promising for the development of automated-verification tools and also shows the call-by-need flavour of $\sim$. Although in the paper we stick to π-calculus, the issues developed may be relevant to value-passing calculi in general.

Work supported by the ESPRIT BRA project 6454 "CONFER".

---

[*] A preliminary version of this article can be found in the proceedings of CONCUR '93.

1 Introduction

Recent developments in the theory of process algebras have led to the formalisation of calculi for mobile processes, i.e. concurrent systems whose communication topology may change dynamically. Among these calculi, Milner, Parrow and Walker's π-calculus [MPW92] is the prototypical one.

In π-calculus there are two entities, *names* and *processes*. Processes interact with each other by exchanging names. This confers mobility to the calculus as well as great expressiveness, for data values [MPW92], λ-calculus [Mil92b] and higher-order processes [San92] can be modeled as special π-calculus processes. Indeed, the main novelty of π-calculus w.r.t. its predecessor CCS [Mil89] is the focus on *name instantiation*, which shows up both in the transition semantics and in the definition of the behavioural equivalences. The purpose of this paper is to add to the understanding of *bisimulation* ([Par81, Mil83]) in the framework of calculi for mobile processes. However, although best suited for these calculi, the ideas we develop may be relevant for value-passing calculi in general.

In the process algebra community, bisimulation is probably the most common mathematical idea utilised to set the behavioural equivalence on processes. Bisimulation has been extensively studied in CCS, where there is general consensus on how it should be defined: approximately, two processes P and Q are bisimilar if

$$P \xrightarrow{\alpha} P' \text{ implies } Q \xrightarrow{\alpha} Q', \text{ for some } Q' \text{ bisimilar to } P' \qquad (*)$$

and the vice versa, on the possible transitions by Q. Here α represents an action and $P \xrightarrow{\alpha} P'$ means "P can evolve to P' by performing the action α". We shall call *ground bisimulation* a bisimilarity relation defined in this way. Obviously, a variation of the calculus may affect the syntax of the actions in $(*)$: for instance, in CCS, actions purely express synchronisations, whereas in π-calculus they may involve communication of names and, moreover, syntactic identity between actions is taken modulo alpha conversion.

But in π-calculus ground bisimulation is too simplistic: It completely ignores name instantiations. The most unfortunate consequence is that the relation is not preserved by parallel composition. To see why, take the processes

$$P \equiv a(x).\,[x = b]\overline{b}b \qquad\qquad Q \equiv a(x).\,\mathbf{0}$$

built from the input prefix $a(x)$, the output prefix $\overline{b}b$, the inactive process $\mathbf{0}$ and the matching $[x = b]$, to be read as "`if` $x = b$ `then`". The processes P and Q are ground bisimilar, since after consuming the initial action $a(x)$ they can do nothing. But they exhibit a different behaviour when run in parallel with $\overline{a}b$, since the interaction between P and $\overline{a}b$ sets the matching in P to true and activates the action $\overline{b}b$. To remedy this kind of problem, Milner, Parrow and Walker in [MPW92] adopts a clause different from $(*)$ for inputs, namely

$$P \xrightarrow{a(x)} P' \text{ implies } Q \xrightarrow{a(x)} Q', \text{ for some } Q' \text{ s.t. for all } y,\ P'\{y/x\} \text{ and } Q'\{y/x\} \text{ are bisimilar.} \qquad (**)$$

The relation obtained is called *late* bisimulation because the choice of the value y with which to instantiate the bound name x is done later than the choice of the derivative Q'. As pointed out in [MPW92], this order can be reversed, yielding the (coarser) *early* bisimulation. Late and early bisimulation are the bisimilarity relations used in the works on π-calculus so far appeared. However, not even these equivalences are preserved by all π-calculus operators. This time the failure, less dramatic, is on input prefix. Again, the reason has to do with name instantiation, and can be illustrated with the same P and Q of the example above: $[x = b]\overline{b}b$ and $\mathbf{0}$ cannot perform actions and therefore are late and early bisimilar; but P and Q, obtained by prefixing $a(x)$, are not, as demonstrated using b as input value. The consequence of this failure is that one has to introduce separately the induced late/early substitutive congruences, which may make heavy to reason about such equivalences.

Here, we propose a different formulation of bisimulation for π-calculus, called *open bisimulation* and written $\sim$. This relation is a full congruence, and strictly finer than the previously mentioned equivalences. In the definition of $\sim$, name instantiation plays a central role: In the late and early equivalences name instantiation only appears in the input clause and, in the congruences, before the initial step; in $\sim$ it becomes integral part of the recursive clause of bisimilarity. The appellative "open" was precisely chosen to emphasise that in this bisimulation, at least in absence of name restrictions, no definitive constraint on the equality of names is assumed but, instead, (free) names can be identified at any time; that is to say, names are treated as *program variables.* Indeed, in the restriction-free sublanguage open bisimulation can also be derived from the *"classical"* ground bisimulation by adding the *specific* π-calculus requirement of closure under name instantiation. Thus, the processes $[x = b]\overline{b}b$ and $\mathbf{0}$ are distinguished because non ground bisimilar under the substitution which identifies x and b.

The clause of bisimilarity in the definition of $\sim$ uses quantification over substitutions. The drawback is the increase in the size of the relations needed to define a bisimulation. Luckily, this problem can be relieved. By constructing a transition system specialised for $\sim$ (in the spirit of Hennessy and Lin's *symbolic* transition system [HL92]) we have derived an "efficient" characterisation of $\sim$, in which any quantification on substitutions is avoided. This characterisation seems promising for the development of automated-verification tools. We also expect that its average complexity be substantially lower than those of the late/early equivalences, since the latter do have a quantification over substitutions in the input clause, like in $(**)$. Moreover, this characterisation of $\sim$ is useful to understand better its intrinsic meaning: For instance, it shows that the instantiation of the parameter of an input should happen "only when needed", in a way which resembles the call-by-need semantics of λ-calculus [Wad71].

An attractive property of $\sim$ is the simple axiomatisation (for finite terms). The treatment of matching, the π-calculus conditional construct, is of technical interest. In the equational theories in [MPW92, Hen91, BD92] a conditional construct is discharged upon evaluation of its boolean condition; but this is not sound if the equivalence being axiomatised, like $\sim$, is preserved by name instantiation. In this case one has to add axioms for syntactic manipulations of conditionals (see for instance the axiomatisations in [PS93] of late and early congruences). Remarkably, with open bisimulation four simple axioms for matching suffice. A spin-off of the axiomatisation is the construction of *canonical representatives* for the equivalence classes of $\sim$. As far as we know, this is the first such result for value-passing calculi. Moreover, in our case the representatives selected are *minimal* w.r.t. to the length of the defining expressions.

Further motivation for the study of open bisimulation might come from Robin Milner's work on *action structures* [Mil92a]. These are proposed as canonical algebraic structures which underly concrete models of concurrency, like Petri Nets, event structures and process calculi. It appears that in the action structures for process calculi the bisimulation relations recovered are close to open bisimulation; but the details remain to be worked out.

We first develop our ideas on the sublanguage of π-calculus without restriction: This because the differences between $\sim$ and the previously known bisimulation-based equivalences already appear in this smaller language, and our ideas can be expressed more clearly. This sublanguage is also interesting because its semantical treatment is very close to the one of an ordinary CCS-like value-passing calculus. Later, we consider how to deal with *distinctions* and the restriction operator. Distinctions [MPW92] allow us to forbid certain identifications of names. The π-calculus does not distinguish among ports, variables and constants: They are all just *names*, which only differ in the way we instantiate them; for instance, variables can be instantiated, or identified, with any name, whereas constants with none. Thus in π-calculus a 'constanthood' is expressed as a special case of distinction. Distinctions shall also help us to deal with restriction. Indeed, a restriction, in common with a constant, declares that a name should not be confused with any other known name.

Related work will be discussed along the way. The notation and, in part, the structure of the paper follow [PS93]. The syntax and the operational semantics of the restriction-free π-calculus are presented in Section 2. We introduce $\sim$ in Section 3; we compare $\sim$ with the previously known bisimilarities and we prove some basic properties of it. In Section 4 we show a sound and complete axiomatisation of $\sim$ on finite terms. In Section 5 we exhibit the efficient characterisation of $\sim$. In Sections 6 and 7 we discuss how to generalise the results in the previous sections to handle distinctions and the restriction operator.

2 The π-calculus: syntax and transition system

We use P, Q, R to range over processes, and $a, b, x, y, \dots$ to range over names. For the moment, we leave the operator of restriction out from the syntax of π-calculus; we will introduce it in Section 7, after having examined distinctions. Thus, the class of processes is given by the following grammar:

$$P \;::\; \mathbf{0} \;\mid\; \alpha.\,P \;\mid\; [a=b]P \;\mid\; P_1 \mid P_2 \;\mid\; P_1 + P_2 \;\mid !P$$

The prefix α can represent an input, an output, or a silent move:

$$\alpha \quad :: \quad a(b) \;\mid\; \overline{a}b \;\mid\; \tau$$

We assign parallel composition and sum the lowest precedence among the operators. $\mathbf{0}$ is the inactive process. An input-prefixed process $a(b).\,P$ waits for a name c to be sent along a and then behaves like $P\{^c\!/\!_b\}$, where $\{^c\!/\!_b\}$ is the substitution of b with c. An output-prefixed process $\overline{a}b.\,P$ sends b along a and then continues like P. The τ-prefixed process $\tau.\,P$ is capable of evolving to P without interacting with the environment. The *matching* construct $[a = b]P$ is used to test for syntactic equality of names; it behaves like P if a and b are the same name, as $\mathbf{0}$ otherwise. Sum and parallel composition are used, as in CCS, to express non-determinism and to run two processes in parallel. Finally, the replication $!\,P$ represents an unbounded number of copies of P in parallel and allows us to describe processes with infinite behaviour.

An input $a(b).\,P$ represents a formal binder for the occurrences of the name b in P. A name which is not bound is *free*. The definitions of free and bound names are standard. We write $fn(P)$ and $fn(\alpha)$ for the set of free names of P and α; we use $bn(P)$ and $bn(\alpha)$ for their bound names; and $n(P)$, $n(\alpha)$ for the union of their free and bound names. Substitution and alpha conversion are defined in the expected way, with renaming possibly involved to avoid capture of names. We use σ and ρ to range over substitutions; $\sigma(a)$ is the name onto which σ maps a. As syntactic form, substitutions have precedence over the operators of the language. *We shall identify processes or actions which differ only on the bound names.* Therefore the symbol $\equiv$ means "syntactic identity modulo alpha conversion"; this will often avoid some tedious side conditions, especially in the definitions of bisimulations.

The *labelled* transition system for the processes of the language is reported in Table 1. We have omitted the symmetric versions of the rules sum, par and com. We work up-to alpha conversion on processes also in the transition system; that is, alpha convertible agents are deemed to have the same transitions. Formally, this means that we implicitly assume the rule

$$\frac{P' \xrightarrow{\alpha} P'' \qquad P \text{ and } P' \text{ alpha convertible}}{P \xrightarrow{\alpha} P''}$$

Sometimes we shall abbreviate $\alpha.\,\mathbf{0}$ to α, and $P \xrightarrow{\tau} P'$ to $P \longrightarrow P'$. We refer to [MPW92] for more detailed discussions on the operators and their meaning.

3 Open bisimulation

3.1 From late to open bisimulation

Before introducing open bisimulation, we present the bisimilarity equivalences for π-calculus so far investigated in the literature. If $\mathcal{S}$ is a process relation, then $P \,\mathcal{S}\, Q$ means $(P, Q) \in \mathcal{S}$.

Definition 3.1 *A relation $\mathcal{S}$ on processes is a* late simulation *if $P \,\mathcal{S}\, Q$ implies*

pre:	$\alpha.P \xrightarrow{\alpha} P$	sum:	$\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$
rep:	$\dfrac{P \mid\, !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$	par:	$\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\quad bn(\alpha) \cap fn(Q) = \emptyset$
com:	$\dfrac{P \xrightarrow{\overline{a}y} P' \qquad Q \xrightarrow{a(x)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{y/x\}}$	match:	$\dfrac{P \xrightarrow{\alpha} P'}{[a=a]P \xrightarrow{\alpha} P'}$

Table 1: π-calculus transition system

- *If $P \xrightarrow{a(x)} P'$, then Q' exists s.t. $Q \xrightarrow{a(x)} Q'$ and for each name b, $P'\{b/x\}\ \mathcal{S}\ Q'\{b/x\}$.*
- *If $P \xrightarrow{\alpha} P'$ for $\alpha = \overline{a}b$ or $\alpha = \tau$, then Q' exists s.t. $Q \xrightarrow{\alpha} Q'$ and $P'\ \mathcal{S}\ Q'$.*

$\mathcal{S}$ is a late bisimulation *if $\mathcal{S}$ and $\mathcal{S}^{-1}$ are late simulations. Two processes P and Q are* late bisimilar*, written $P \stackrel{\cdot}{\sim}_L Q$, if $P\ \mathcal{S}\ Q$, for some late bisimulation $\mathcal{S}$.*

In the definition of *early bisimulation*, written $\stackrel{\cdot}{\sim}_E$, the order of the quantifiers in clause (1) is reversed, i.e. one requires that "for each name b, a process Q' exists s.t. $Q \xrightarrow{a(x)} Q'$ and $P'\{b/x\} \stackrel{\cdot}{\sim}_L Q'\{b/x\}$. As shown in Section 1, neither $\stackrel{\cdot}{\sim}_L$ nor $\stackrel{\cdot}{\sim}_E$ are preserved by input prefix; the full congruences are obtained using name instantiation. Thus two processes P and Q are *late congruent* (resp. *early congruent*), written $\sim_L$ (resp. $\sim_E$) if $P\sigma \stackrel{\cdot}{\sim}_L Q\sigma$ (resp. $P\sigma \stackrel{\cdot}{\sim}_E Q\sigma$), for every substitution σ.

In *open bisimulation*, which we are going to introduce, name instantiation is moved inside the definition of bisimulation; as a consequence, the clauses (1) and (2) of the previous definition collapse into one.

Definition 3.2 *A relation $\mathcal{S}$ on processes is an* open simulation *if $P\ \mathcal{S}\ Q$ implies, for every σ:*

- *whenever $P\sigma \xrightarrow{\alpha} P'$ then Q' exists s.t. $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\ \mathcal{S}\ Q'$.*

$\mathcal{S}$ is an open bisimulation *if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are open simulations. The processes P and Q are* open bisimilar*, written $P \sim Q$, if $P\ \mathcal{S}\ Q$, for some open bisimulation $\mathcal{S}$.*

It is shown in [MPW91] that both the late and the early congruences are finer than the corresponding bisimulations and that, in turn, the two late equivalences are finer than the corresponding early ones. It is easy to see that an open bisimulation is a late bisimulation; moreover, since open bisimulation is closed under name instantiation (we shall show this in Proposition 3.9) it is also contained in late congruence. The following example shows that the inclusion is strict.

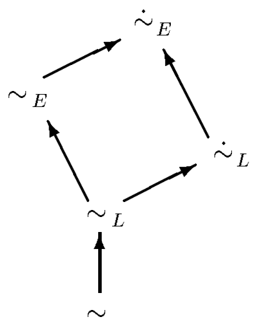


Figure 1: The spectrum of the bisimilarity equivalences

Example 3.3 *We give an example which can be used in any CCS-like language with value-passing (for π-calculus we could make it even simpler by omitting the first action $c(a)$). We write $R \longrightarrow$ to mean that $R \longrightarrow R'$, for some R'. Define*

$$\begin{array}{rcl} P & \stackrel{def}{=} & c(a).(\tau.\tau+\tau) \\ Q & \stackrel{def}{=} & c(a).(\tau.\tau+\tau+\tau.[a=b]\tau) \end{array}$$

It holds that $P \sim_L Q$: for every σ, after the initial input action $Q\sigma$'s third summand becomes equal to $P\sigma$'s first summand or to $P\sigma$'s second summand, depending on whether a is instantiated to b or not. By contrast, in $\sim$ the instantiation of a can be delayed until a used — echoing a call-by-need style. Consequently, the actions $Q \xrightarrow{c(a)} \longrightarrow [a=b]\tau$ are not matched neither by $P \xrightarrow{c(a)} \longrightarrow \tau$, nor by $P \xrightarrow{c(a)} \longrightarrow \mathbf{0}$, since in the former case $\tau \longrightarrow$ but $[a=b]\tau \not\longrightarrow$, and in the latter case $([a=b]\tau)\{{}^a\!/\!_b\} \longrightarrow$ but $\mathbf{0}\{{}^a\!/\!_b\} \not\longrightarrow$.

Figure 1 summarises the relationship among the equivalences considered, an arrow meaning strict inclusion.

3.2 The weak version

The relations that we have seen so far are called *strong* because the internal moves of the processes, i.e. the τ-actions, are given the same weight as the visible actions. If one wishes to abstract from the internal details of the processes, then one adopts a *weak* behavioural equivalence. Let $\Longrightarrow$ be $\longrightarrow^*$, i.e. the reflexive and transitive closure of $\longrightarrow$, and $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ be $\Longrightarrow$ if $\alpha = \tau$ and $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$ if $\alpha \neq \tau$. *Weak open bisimulation*, written $\approx$, is obtained, as usual, by replacing the arrow $Q\sigma \xrightarrow{\alpha} Q'$ with $Q\sigma \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ in the Definition 3.2 of the strong bisimulation. It is straightforward to see that:

Proposition 3.4 *The relations $\sim$ and $\approx$ are equivalences.* $\square$

Early bisimulation is extended to the weak case in a similar way. The extension of late bisimulation is more delicate: in the input clause of Definition 3.1, the arrow $Q \xrightarrow{a(x)} Q'$ has to be replaced by $Q \Longrightarrow\xrightarrow{a(x)} Q'$ rather than (the expected) $Q \overset{a(x)}{\Longrightarrow} Q'$, for otherwise the resulting relation $\dot{\approx}_L{}'$ is not transitive; the counterexample below is a variant of the original one due to Robin Milner:

Example 3.5 *Let $\alpha \neq \tau$ and define:*

$$\begin{array}{lclclcl} P_1 & \overset{def}{=} & c(a).\,[a=b]\alpha \;+ & c(a).\,(\tau.\,[a=b]\alpha & + & \tau.\,\alpha & + \;\; \tau) \\ P_2 & \overset{def}{=} & & c(a).\,(\tau.\,[a=b]\alpha & + & \tau.\,\alpha & + \;\; \tau) \\ P_3 & \overset{def}{=} & & c(a).\,(& & \tau.\,\alpha & + \;\; \tau) \end{array}$$

It holds that $P_1 \dot{\approx}_L{}' P_2 \dot{\approx}_L{}' P_3$. However $P_1 \not\dot{\approx}_L{}' P_3$: The action $P_1 \xrightarrow{c(a)} [a=b]\alpha$ cannot be matched by P_3: If it chooses $P_3 \overset{c(a)}{\Longrightarrow} \tau.\,\alpha + \tau$, then $[a=b]\alpha \not\dot{\approx}_L{}' \tau.\,\alpha + \tau$; similarly, if it chooses $P_3 \overset{c(a)}{\Longrightarrow} \alpha$, then $[a=b]\alpha \not\dot{\approx}_L{}' \alpha$; finally, if it chooses $P_3 \overset{c(a)}{\Longrightarrow} \mathbf{0}$, then $([a=b]\alpha)\{b/a\} \not\dot{\approx}_L{}' \mathbf{0}\{b/a\}$.

Another way of making weak late bisimulation transitive is to replace not only $Q \xrightarrow{a(x)} Q'$ with $Q \overset{a(x)}{\Longrightarrow} Q'$, but also $P \xrightarrow{a(x)} P'$ with $P \overset{a(x)}{\Longrightarrow} P'$ (in the example above, the equality between P_2 and P_3 breaks down). But this would yield a clause too expensive to check in practice: The set of derivatives of P under $\overset{\alpha}{\Longrightarrow}$ can be much bigger — even sometimes infinite — than the set under $\xrightarrow{\alpha}$; moreover, certain derivatives would be considered more than once (for instance, if $P \xrightarrow{\tau} P' \overset{\alpha}{\Longrightarrow} P''$, then P'' would be examined both as an $\overset{\alpha}{\Longrightarrow}$ derivative of P and as an $\overset{\alpha}{\Longrightarrow}$ derivative of P'). For this reason, it is important to keep the "strong" arrow $P \xrightarrow{a(x)} P'$.

The smooth weak extension may be the indication of a greater stableness of open bisimulation w.r.t. late bisimulation. Indeed, a possible interpretation of open bisimulation is as a correction of late bisimulation — the "very late" approach of open bisimulation to the instantiation of names already appeared in Example 3.3, and it will be made clearer in Section 5. In the remainder of the paper we stick to strong open bisimulation.

3.3 A useful proof technique

Our next result shows that the verification of open bisimilarities can be split into two parts: on the one hand there is *ground bisimulation* which, modulo the different syntax of actions, is bisimulation as defined in CCS; on the other hand there is *name instantiation*, that is the most distinctive notion of π-calculus w.r.t. CCS.

Definition 3.6 *A relation $\mathcal{S}$ is* closed under a substitution σ *if $P \,\mathcal{S}\, Q$ implies $P\sigma \,\mathcal{S}\, Q\sigma$.*

Definition 3.7 *A relation $\mathcal{S}$ is a* ground simulation *if $P \,\mathcal{S}\, Q$ implies*

- *whenever $P \xrightarrow{\alpha} P'$ then Q' exists s.t. $Q \xrightarrow{\alpha} Q'$ and $P' \,\mathcal{S}\, Q'$.*

$\mathcal{S}$ is a ground bisimulation *if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are ground simulations.*

Proposition 3.8 *Suppose that a relation* $\mathcal{S}$

1. *is a ground bisimulation*
2. *is closed under all substitutions.*

Then $\mathcal{S}$ *is an open bisimulation.*

Proof: Easy. $\square$

The converse of Proposition 3.8 in general fails: An open bisimulation is a ground bisimulation but is not necessarily closed under all substitutions. An example is $\mathcal{S} = \{(\overline{x}y, \overline{x}y), (\mathbf{0}, \mathbf{0})\}$. To close $\mathcal{S}$, we need to add the instantiations of the "root" $(\overline{x}y, \overline{x}y)$. In the same way we can close any open bisimulation. Since the *largest* open bisimulation *is* closed, we have:

Proposition 3.9 $\sim$ *is the largest ground bisimulation which is closed under all substitutions.* $\square$

3.4 Congruence

We finish the section by looking at the congruence properties of $\sim$.

Proposition 3.10 $\sim$ *is a congruence relation.*

Proof: By showing that $\sim$ is preserved by each operator of the language. For input prefix, this holds because $\sim$ is closed under substitutions of names (Proposition 3.9). For the other operators, the technical details are similar to those in the proofs in [MPW92] of the congruence properties of late bisimulation. We only consider parallel composition. Define

$$\mathcal{S} = \{(P_1 \mid R, P_2 \mid R) \ : \ P_1 \sim P_2\}$$

The relation $\mathcal{S}$ is closed under all substitutions because $\sim$ is so; by Proposition 3.8 it is enough to prove that $\mathcal{S}$ is a ground bisimulation. For this one proceeds by a case analysis on the rule used to infer an action for $P_1 \mid R$. We only consider the rule com when P_1 performs an input, since the other are easier. We have

$$\frac{P_1 \xrightarrow{a(x)} P_1' \qquad R \xrightarrow{\overline{a}b} R'}{P_1 \mid R \xrightarrow{\tau} P_1'\{b/x\} \mid R'}$$

By definition of $\sim$, $P_2 \xrightarrow{a(x)} P_2' \sim P_1$. Therefore also $P_2 \mid R \xrightarrow{\tau} P_2\{b/x\} \mid R'$. Since $\sim$ is closed under substitution, we have $P_1'\{b/x\} \sim P_2\{b/x\}$, hence $P_1'\{b/x\} \mid R' \ \mathcal{S} \ P_2'\{b/x\} \mid R'$. $\square$

Therefore $\sim$ is both a bisimulation *and* a congruence. This reminds us of Montanari and Sassone's *dynamic bisimulation* [MS92]. The latter has been introduced in CCS to study systems which allow for dynamic reconfigurations. When bisimilar, such systems can be consistently substituted one for the other in any context and at any time of their life. We present here the strong version of dynamic bisimulation for the sublanguage of π-calculus which we are presently examining.

Definition 3.11 (dynamic bisimulation) Dynamic bisimulation *is the largest symmetric relation* $\mathcal{S}$ *s.t.* $P\,\mathcal{S}\,Q$ *implies, for every contexts* $C[\cdot]$*:*

- *if* $C[P] \xrightarrow{\alpha} P'$*, then* Q' *exists s.t.* $C[Q] \xrightarrow{\alpha} Q'$ *and* $P'\,\mathcal{S}\,Q'$*.*

In other words, dynamic bisimulation is the largest ground bisimulation which is also a congruence. Since the latter requirement is assured by the closure under substitutions of names and vice versa, by Proposition 3.9 we get:

Proposition 3.12 *Dynamic bisimulation and open bisimulation coincide.* ☐

However, the two relations may differ in the weak case, due to the usual congruence problems of sum: Open bisimulation is preserved by *guarded* sum but not by sum, whereas dynamic bisimulation, by definition, is always a congruence.

4 The axiomatisation

In this section we give a sound and complete axiomatisation for $\sim$ on finite terms, i.e. terms in which replication does not appear. We shall use M, N, L to stand for finite (and possibly empty) *match sequences*; thus, if N is $[a = b][c = d]$, then NP is an abbreviation for $[a = b][c = d]P$. A match sequence defines an equivalence relation on names. Given a match sequence M and the equivalence relation $\mathcal{R}_M$ associated to it, we denote by σ_M a special substitution which selects a representative out of each equivalence class of $\mathcal{R}_M$ and maps all names in the same class to their representative (there may be many ways of defining σ_M; we just pick one). We write $M \rhd N$ if M implies N, i.e. whenever the tests in M are true, then also the tests in N are true. Similarly, we write $M \lhd\rhd N$ if both $M \rhd N$ and $N \rhd M$ hold. We say that a substitution σ *satisfies* a match sequence M if for each a, b, it holds that $M \rhd [a = b]$ implies $\sigma(a) = \sigma(b)$, that is, the condition $M\sigma$ is true. We write $\sigma\rho$ for the composition of substitutions; therefore we have $P\sigma\rho \equiv (P\sigma)\rho$.

Lemma 4.1

1. $(N\alpha.\,P)\sigma_M \xrightarrow{\alpha\sigma_M} P\sigma_M$ *iff* $M \rhd N$ *(or, equivalently,* $NM \lhd\rhd M$*).*

2. *If* σ *satisfies* M*, then* $\sigma = \sigma_M\rho$*, for some* ρ*.*

3. *For each name* a*,* $\sigma_M(\sigma_M(a)) = \sigma_M(a)$*.*

4. *If* $M \lhd\rhd N$*, then for some injective substitution* ρ*, we have* $\sigma_M = \sigma_N\rho$*.* ☐

Alpha-conversion	**A**	If P and Q alpha-convertible then P		$=$	Q
Congruence	**C1**	If $P = Q$ then	$\alpha. P$	$=$	$\alpha. Q$
	C2	If $P = Q$ then	$P + R$	$=$	$Q + R$
	C3	If $P = Q$ then	$[x = y]P$	$=$	$[x = y]Q$
	C4	If $P = Q$ then	$P \mid R$	$=$	$Q \mid R$
Summation	**S1**		$P + \mathbf{0}$	$=$	P
	S2		$P + Q$	$=$	$Q + P$
	S3		$P + (Q + R)$	$=$	$(P + Q) + R$
Matching	**M1**	If $M \lhd\!\rhd N$ then	MP	$=$	NP
	M2		$[x = y]P$	$=$	$[x = y](P\{^x\!/\!_y\})$
	M3		$[x = y](P + Q)$	$=$	$[x = y]P + [x = y]Q$
	M4		$[x = y]P + P$	$=$	P
Expansion	**E**				

Assume $P \equiv \sum_i M_i\, \alpha_i.\, P_i$ and $Q \equiv \sum_j N_j\, \alpha_j.\, Q_j$ where no α_i (resp. β_i) binds a name free in Q (resp. P). Then infer:

$$P \mid Q = \sum_i M_i\, \alpha_i.\, (P_i \mid Q) + \sum_j N_j\, \alpha_j.\, (P \mid Q_j) + \sum_{\alpha_i \; opp \; \beta_j} M_i N_j [\, x_i \; = \; y_j \,]\, \tau.\, R_{ij}$$

where α_i *opp* β_j and R_{ij} are defined as follows

1. α_i is $\overline{x}_i u$ and β_j is $y_j(v)$; then R_{ij} is $P_i \mid Q_j\{^u\!/\!_v\}$;
2. the vice versa of (1);

Table 2: The axiom system

4.1 The axiom system

The axioms are in Table 2. **S1**-**S3** are the monoidal laws for sum; **C1**-**C4** are the congruence laws; **E** is the usual expansion law. The interesting laws are those involving matching. **M1** says that two semantically equivalent match sequences can be exchanged one for the other; **M2** shows that underneath a matching the tested names are indistinguishable; **M3** asserts the distributivity of matching over sum; **M4** is an absorption law certifying that the only possible effect of matching is to block the access to a process. If we prefer some more concrete and purely equational axioms, then **M1** can be replaced by the following five axioms which formalise the notion of equality on names:

$$\begin{aligned}
[x = y][x = y]P &= [x = y]P \\
[x = y][x = z]P &= [x = y][x = z][y = z]P \\
[x = y][z = u]P &= [z = u][x = y]P \\
[x = x]P &= P \\
[x = y]P &= [y = x]P
\end{aligned}$$

We call $\mathcal{E}$ the given axiom system. We write $\mathcal{E} \vdash P = Q$ if $P = Q$ can be inferred from $\mathcal{E}$ with equational reasoning; occasionally, we might omit $\mathcal{E}$, if clear. We also write $P \overset{\mathbf{Z}}{=} Q$ if the equality of P and Q is deduced from the the axiom (or rule) **Z** plus the axioms for congruence and equivalence.

The soundness of $\mathcal{E}$ is evident.

Proposition 4.2 (soundness of $\mathcal{E}$) *If $\mathcal{E} \vdash P = Q$, then $P \sim Q$.* ☐

The remainder of the section is devoted to the proof of completeness. After having presented some derived rules, we introduce the notion of normal form and show how to rewrite a process into normal form. We then refine normal forms to strong normal forms and demonstrate that equivalent strong normal forms are provably equal. A tree representation of the terms will allow us to visualise the latter part.

Lemma 4.3 *The following are derived rules in $\mathcal{E}$. (In the denomination of these axioms, "D" stands for "derived".)*

$$\begin{array}{lrcl}
\textbf{SD1} & P + P & = & P \\
\textbf{MD1} & [a = b]\mathbf{0} & = & \mathbf{0} \\
\textbf{MD2} & P + MP & = & P \\
\textbf{MD3} & MP & = & M(P\sigma_M)
\end{array}$$

Proof:

SD1: $P \overset{\mathbf{M4}}{=} [x = x]P + P \overset{\mathbf{M1}}{=} P + P$.

MD1: $\mathbf{0} \overset{\mathbf{M4}}{=} [a = b]\mathbf{0} + \mathbf{0} \overset{\mathbf{S1}}{=} [a = b]\mathbf{0}$.

MD2: Induction on the length of M. If the length is 0, then it is a special case of **SD1**. Otherwise M is of the form $[a=b]N$, and by induction

$$\vdash\ P + NP = P \tag{1}$$

Therefore we get

$$\vdash\ P \overset{(1)}{=} P + NP \overset{\mathbf{M4}}{=} P + NP + [a=b]NP \overset{(1)}{=} P + [a=b]NP$$

MD3: Take a name a and suppose $\sigma_M(a) = b$. Then, by definition of σ_M it holds that $\sigma_M(b) = b$ and $M \lhd\rhd M[b=a]$; hence

$$MP \overset{\mathbf{M1}}{=} M[b=a]P \overset{\mathbf{M2}}{=} M[b=a](P\{{}^{b}\!/_{a}\}) \overset{\mathbf{M1}}{=} M(P\{{}^{b}\!/_{a}\})$$

Repeating this for all names free in P gives **MD3**.

$\Box$

Definition 4.4 (normal form) *A process P is in* normal form *(nf in short) if*

$$P \equiv \textstyle\sum_{i\in I} M_i\alpha_i.\, P_i$$

where each P_i is in normal form.

Lemma 4.5 *For each process P there is a term H in nf s.t. $\mathcal{E} \vdash P = H$.*

Proof: By induction on the structure of P. Suppose $P \equiv [x=y]P_1$ and $\vdash\ P_1 = H$: Then $\vdash\ P = [x=y]H$; now, either use **M3** to distribute the matching over the summands of H or, if $H \equiv \mathbf{0}$, use **MD1** to reduce P to $\mathbf{0}$. Suppose $P \equiv P_1 \mid P_2$ and $\vdash\ P_1 = H_1$, $\vdash\ P_2 = H_2$: Then $\vdash\ P = H_1 \mid H_2$, and $H_1 \mid H_2$ can be put into nf by repeatedly applying **E**. Finally, the case when $P \equiv \alpha.\, Q$ or $P \equiv P_1 + P_2$ just requires use of induction. $\Box$

4.2 The tree representation

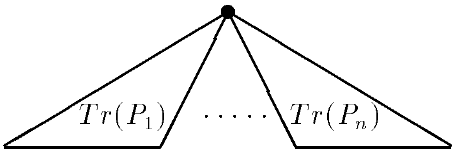
Processes in nf can be represented as *conditional trees*; these are labelled trees in which the access to a branch is controlled by a condition. Two examples are shown in figure 2; they are the trees of the terms $[a=b]\,a(x).\,([x=b]\,\overline{x}c + \overline{a}b)$ and $a(x).\,(\overline{b}c + [x=a][a=b]\,\overline{x}x)$, respectively.

The tree representation has been a very useful form of denotational semantics for CCS-like process algebras; often techniques for manipulating terms have been described, or even defined, directly on trees [BK85, DKV91]. Up-to-date no tree representation for π-calculus terms has been proposed. We think that part of the reason is in the behavioural equivalences used: Usually these equivalences are not preserved by input prefix and therefore have a rather complicated inference rule for it; this makes arduous to prove process equivalences proceeding by term-rewriting transformations on some form of tree. By contrast, we found conditional trees helpful to visualise our reasoning with open bisimulation. In the next section, we shall use them to explain the phases of our completeness proof.

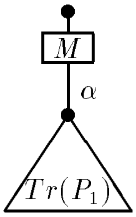
Figure 2: Examples of conditional trees

Definition 4.6 (conditional trees) *The* tree *of a term P in nf, written $Tr(P)$, is defined by induction on the structure of P as follows:*

1. *If $P \equiv \sum_{i=1}^{n} P_i$ then $Tr(P)$ is obtained by joining the roots of the trees of P_1, ..., P_n:*



2. *If $P \equiv M\alpha.P_1$ then $Tr(P)$ is*



Note that by clause (1), we have $Tr(\mathbf{0}) = \bullet$. We write $T_1 =_* T_2$ if T_1 and T_2 are equal modulo the following syntactic manipulations, for which the choice of a canonical form would require the definition of an ordering on the names and a standard representation of match sequences:

- alpha conversion,
- commutativity of consecutive branches,
- equality $\lhd\rhd$ between match sequences,
- identification of names equated by a match underneath the match itself.

These transformations are the direct correspondent of the axioms **A**, **S2**, **S3**, **M1**, and **M2**:

Lemma 4.7 *Let P and Q be nf's; then*

$$Tr(P) =_* Tr(Q) \textit{ iff } \{\mathbf{C1\text{-}4}, \mathbf{A}, \mathbf{S2}, \mathbf{S3}, \mathbf{M1}, \mathbf{M2}\} \vdash P = Q. \qquad \Box$$

It is convenient to work in terms of $=_*$ to avoid defining and respecting tedious technicalities like the ordering on names. With this motivation in mind, the reader should think of $=_*$ as syntactic equality between trees.

4.3 Strong normal forms and completeness of the axiom system

The next step in the proof of completeness for our axiom system is to rewrite a normal form into *strong normal form* (snf). The advantage of this is that equivalent processes have essentially the *same* snf. This unicity property fails for nf's. For instance, $[a = b]\alpha.(\overline{a}c + \overline{b}c)$ and $[a = b]\alpha.\overline{a}c$ are equivalent but structurally different nf's. Intuitively, a nf P is in snf if none of its non-trivial subterms is redundant; thus if $P_1 + P_2$ is a subterm of P then both P_1 and P_2 must contribute to the behaviour of P.

Definition 4.8 *A term P is in* strong normal form *(snf in short) if*

$$P \equiv \sum_{i \in I} M_i \alpha_i.\, P_i \qquad \textit{where}$$

1. *for all i, $bn(\alpha_i) \cap fn(P) = \emptyset$ and $P_i \sigma_{M_i}$ is in snf;*
2. *if $i \neq j$ then $M_i \alpha_i.\, P_i \not\succ M_i \alpha_i.\, P_i + M_j \alpha_j.\, P_j$.*

The reader might want to verify that in clause (2) it does not make a difference to have a composite term (i.e. a sum) rather than the "single" term $M_i \alpha_i.\, P_i$.

Lemma 4.9 *Suppose that P is in snf and ρ is an injective substitution; then also $P\sigma$ is in snf.* □

The next two lemmas are the core of the completeness proof. They show that equivalent snf's are provably equal and that a nf can be transformed into snf. In the assertion of the first result all axioms needed are explicitly declared, in order to show that equivalence of snf's is the same as the equality $=_*$ of trees; this will be formalised in Theorem 4.13 and will give us canonical representatives for the equivalence classes of $\sim$.

Lemma 4.10 *If P and Q are snf's and $P \sim Q$, then* $\{\mathbf{C1\text{-}4}, \mathbf{A}, \mathbf{S2}, \mathbf{S3}, \mathbf{M1}, \mathbf{M2}\} \vdash P = Q$.

Proof: We shall also use the axiom **MD3** since, as shown in Lemma 4.3, this is derivable from **C1-4**, **M1** and **M2**. We proceed by induction on the depth of $P + Q$. If the depth is 0, then $P \equiv Q \equiv \mathbf{0}$. Suppose the depth is non-zero. We show that each summand of P can be transformed into a summand of Q; then $P = Q$ can be derived from the commutativity and associativity of sum, namely axioms **S2** and **S3** (note that we do not need **S1**: It cannot be that different summands of P are transformed into the same summand of Q otherwise, by the soundness of our axiom system, the second clause in the definition of snf would be violated).

If $M\alpha.P_1$ is a summand of P, we have

$$P\sigma_M \xrightarrow{\alpha\sigma_M} P_1\sigma_M$$

Let $N\beta.Q_1$ be the summand of Q used by $Q\sigma_M$ to match this move. Then $M \rhd N$ (Lemma 4.1(1)) and $Q\sigma_M \xrightarrow{\beta\sigma_M} Q_1\sigma_M$ with

$$\alpha\sigma_M \equiv \beta\sigma_M \tag{2}$$

$$P_1\sigma_M \sim Q_1\sigma_M \tag{3}$$

By definition of snf, $P_1\sigma_M$ and $Q_1\sigma_N$ are snf's. Let us show how we could finish the proof if we knew that not only $M \rhd N$, but also

$$N \rhd M \tag{4}$$

holds: If $M \lhd\rhd N$, then by Lemma 4.1(4), $\sigma_M = \sigma_N\rho$, for some injective ρ, and by Lemma 4.9, $Q_1\sigma_M \equiv Q_1\sigma_N\rho$ is in snf. Therefore, from (3) and the induction assumption of the lemma, we can derive

$$P_1\sigma_M = Q_1\sigma_M \tag{5}$$

which gives

$$\begin{gathered} N\beta.Q_1 \overset{\mathbf{M1}}{=} M\beta.Q_1 \overset{\mathbf{MD3}}{=} M(\beta\sigma_M).(Q_1\sigma_M) \overset{(2)}{=} \\ M(\alpha\sigma_M).(Q_1\sigma_M) \overset{(5)}{=} M(\alpha\sigma_M).(P_1\sigma_M) \overset{\mathbf{MD3}}{=} M\alpha.P_1 \end{gathered}$$

Thus it remains to prove (4). Suppose $N \not\rhd M$. We know that $P \sim Q$; consider the transition

$$Q\sigma_N \xrightarrow{\beta\sigma_N} Q_1\sigma_N$$

The process $P\sigma_N$ cannot match it using the summand $(M\alpha.P_1)\sigma_N$, because of the assumption $N \not\rhd M$ (Lemma 4.1(1)). Therefore another summand of P, say $L\gamma.P_2$, has to be used and it holds that

$$\sigma_N \text{ satisfies } L \tag{6}$$

and $(L\gamma.P_2)\sigma_N \xrightarrow{\gamma\sigma_N} P_2\sigma_N$ with

$$\gamma\sigma_N \equiv \beta\sigma_N \tag{7}$$

$$P_2\sigma_N \sim Q_1\sigma_N \tag{8}$$

But now we get into contradiction with the hypothesis that P is in snf, since we can show that

$$M\alpha.P_1 + L\gamma.P_2 \sim L\gamma.P_2 \tag{9}$$

We prove (9) following the definition of $\sim$. Consider a generic substitution σ. The interesting case is when σ allows us to use $M\alpha.P_1$ to infer an action:

$$(M\alpha.P_1 + L\gamma.P_2)\sigma \xrightarrow{\alpha\sigma} P_1\sigma \tag{10}$$

For this to be possible, σ must satisfies M, i.e. by Lemma 4.1(2) for some σ'

$$\sigma = \sigma_M \sigma' \tag{11}$$

Similarly, since $M \rhd N$ and therefore σ_M satisfies N, for some ρ

$$\sigma_M = \sigma_N \rho \tag{12}$$

This and (11) give

$$\sigma = \sigma_N \rho \sigma' \tag{13}$$

We shall use these informations on the substitutions to show that $L\gamma.\, P_2$ can match the transition in (10). First, from (13) and (6) we deduce that σ satisfies L; hence $(L\gamma.\, P_2)\sigma$ can do:

$$(L\gamma.\, P_2)\sigma \xrightarrow{\gamma\sigma} P_2\sigma$$

Then we want to show that $\gamma\sigma \equiv \alpha\sigma$ and $P_2\sigma \sim P_1\sigma$. This would conclude (9) and finish the proof of the lemma. For the former, we have:

$$\gamma\sigma \overset{(13)}{=} \gamma\sigma_N\rho\sigma' \overset{(7)}{=} \beta\sigma_N\rho\sigma' \overset{(12)}{=} \beta\sigma_M\sigma' \overset{(2)}{=} \alpha\sigma_M\sigma' \overset{(11)}{=} \alpha\sigma$$

To derive the latter, we start from (8), namely $P_2\sigma_N \sim Q_1\sigma_N$. Since $\sim$ is closed under substitutions (Proposition 3.9), and by (12) $\sigma_M = \sigma_N\rho$, we have

$$P_2\sigma_M \equiv P_2\sigma_N\rho \sim Q_1\sigma_N\rho \equiv Q_1\sigma_M$$

which together with (3) yields

$$P_2\sigma_M \sim P_1\sigma_M$$

Finally, similarly as above, but this time using (11),

$$P_2\sigma \equiv P_2\sigma_M\sigma' \sim P_1\sigma_M\sigma' \equiv P_1\sigma \qquad \square$$

Now we can show how to transform a nf into snf. If P is in nf, then the *length* of P, written $\texttt{len}(P)$, is the number of prefixes appearing in P.

Lemma 4.11 *If P is a nf, then there is a snf H s.t. $\mathcal{E} \vdash P = H$; moreover $fn(H) \subseteq fn(P)$ and $\texttt{len}(H) \le \texttt{len}(P)$.*

Proof: By induction on the depth of P. If the depth is 0 then $P \equiv \mathbf{0}$ and the lemma is trivially true. Suppose now the depth is non-zero. Applying **MD3** on the summands $M_i\alpha_i.\, P_i$ of P, we get the process

$$\sum_{i \in I} M_i(\alpha_i\sigma_{M_i}).\, (P_i\sigma_{M_i})$$

Using the inductive assumption of the lemma, each $P_i\sigma_{M_i}$ can be put into a snf P_i', thus obtaining the process

$$P' \equiv \sum_{i\in I} M_i(\alpha_i\sigma_{M_i}).\, P_i'$$

Moreover, since

$$fn(P_i') \subseteq fn(P_i\sigma_{M_i}). \tag{14}$$

and $\texttt{len}(P_i') \leq \texttt{len}(P_i\sigma_{M_i})$, it also holds that

$$fn(P') \subseteq fn(P) \;\text{ and }\; \texttt{len}(P') \leq \texttt{len}(P) \tag{15}$$

Each summand $M_i(\alpha_i\sigma_{M_i}).\, P_i'$ is in snf if clause (1) of the definition of snf is satisfied. For this, we demonstrate that $P_i'\sigma_{M_i} \equiv P_i'$: Let $a \in fn(P_i')$; by (14) we also have $a \in fn(P_i\sigma_{M_i})$ and since σ_{M_i} is idempotent (Lemma 4.1(3)), $\sigma_{M_i}(a) = a$.

Thus, we are left with having to fulfill the second clause of the definition of snf for P'. Suppose that Q and R are summands of P' with

$$Q \sim Q + R \tag{16}$$

We shall show that

$$\vdash Q = Q + R \tag{17}$$

i.e. the axioms allow us to eliminate the summand R. The same strategy can be used to eliminate any other redundant summand of P', thus ending up with a process which is in snf and, using (15), has no more free names than P and a length not greater than the one of P, as required by the assertion of the lemma. Suppose $R \equiv M\alpha.\, R_1$; the equality (17) can be derived from

$$\vdash MQ = R \tag{18}$$

as follows:

$$Q \overset{\mathbf{MD2}}{=} Q + MQ \overset{(18)}{=} Q + R$$

Thus we have reduced ourselves to having to prove (18). Take the substitution σ_M corresponding to M. From (16) and Proposition 3.9

$$Q\sigma_M \sim Q\sigma_M + R\sigma_M \tag{19}$$

Using the summand $R\sigma_M$ we can infer

$$Q\sigma_M + R\sigma_M \xrightarrow{\alpha\sigma_M} R_1\sigma_M.$$

But because of (19), we know that $Q\sigma_M$ can match this move. Therefore, if $Q \equiv N\beta.\, Q_1$, then σ_M must satisfy N, and we have $Q\sigma_M \xrightarrow{\beta\sigma_M} Q_1\sigma_M$ with

$$\beta\sigma_M \equiv \alpha\sigma_M \tag{20}$$

$$Q_1\sigma_M \sim R_1\sigma_M \tag{21}$$

Moreover, by Lemma 4.1(1),

$$NM \lhd\rhd M \tag{22}$$

Now, from (22) and (20) we get

$$MQ \equiv MN\beta.\,Q_1 \overset{\mathbf{M1}}{=} M\beta.\,Q_1 \overset{\mathbf{MD3}}{=} M(\beta\sigma_M).\,(Q_1\sigma_M) \equiv M(\alpha\sigma_M).\,(Q_1\sigma_M) \tag{23}$$

We know that $R_1\sigma_M$ is in snf, and by (21), it is equivalent to the nf $Q_1\sigma_M$[1]. Using the inductive assumption of the lemma and the soundness of our axioms, $Q_1\sigma_M$ can be transformed into an equivalent snf which, by Lemma 4.10 is provably equal to $R_1\sigma_M$; thus we can infer

$$\vdash\ Q_1\sigma_M = R_1\sigma_M \tag{24}$$

Finally, the two previous equalities give

$$\vdash\ MQ \overset{(23)}{=} M(\alpha\sigma_M).\,(Q_1\sigma_M) \overset{(24)}{=} M(\alpha\sigma_M).\,(R_1\sigma_M) \overset{\mathbf{MD3}}{=} M\alpha.\,R_1$$

which is precisely (18).

□

Corollary 4.12 (soundness and completeness of $\mathcal{E}$) *It holds that* $P \sim Q$ *iff* $\mathcal{E} \vdash P = Q$.

Proof: The soundness has been considered in Lemma 4.2. For completeness, suppose $P \sim Q$. By Lemmas 4.5 and 4.11 there are snf's H and K such that $\mathcal{E} \vdash P = H$ and $\mathcal{E} \vdash Q = K$. By the soundness of $\mathcal{E}$ and transitivity of $\sim$, we have $H \sim K$; therefore by Lemma 4.10, we also have $\mathcal{E} \vdash H = K$. □

The proofs that we have given say more than the completeness of our axiom system. They also show how to put a process into a *canonical* form, namely the snf, and that these canonical forms are essentially unique for each equivalence class of $\sim$; the latter because Lemma 4.10, together with Lemma 4.7, gives:

Theorem 4.13 (canonical representatives)
If P *and* Q *are snf's then* $P \sim Q$ *iff* $Tr(P) =_* Tr(Q)$. □

Moreover the canonical representatives are *minimal* w.r.t. the dimension of the associated conditional tree.

Theorem 4.14 (minimality of the canonical representatives) *Suppose* P *is a snf,* Q *is a nf and* $P \sim Q$*; then* $\mathtt{len}(P) \le \mathtt{len}(Q)$.

---

[1] $Q_1\sigma_N$ is in snf, but $Q_1\sigma_M$ in general is not.

PROOF: Lemma 4.11 shows that that the transformation to snf does not increase the length a nf. By Lemma 4.10, snf's are "structurally" unique for each equivalence class: Hence the assertion of the theorem follows. $\square$

The minimal canonical representatives are somehow the "best" or "simplest" processes in the respective equivalence classes. It would be interesting to see whether the algorithm which can be extracted from the completeness proof is expressible as a non-divergent term-rewriting system. Note that on the tree representatives, such an algorithm trivially yields a fully abstract semantics for the finite terms of the language, in which a term is mapped onto its canonical tree.

5 An efficient characterisation

The definition of $\sim$ involves at each step a universal quantification over substitutions, which in practice may represent a rather heavy requirement. We shall show in this section a more efficient method to check process bisimilarities. This is based on the definition of a transition system specialised for $\sim$. The actions in this system are of the form $P \overset{(M,\alpha)}{\leadsto} P'$. Intuitively, M collects the conditions indispensable for the action α to fire. In other words, M defines the "minimal" substitution σ_M which would allow P to use α; Examples of transitions are

$$[a = b]\, \alpha.\, P \overset{([a=b],\alpha)}{\leadsto} P \quad \text{and} \quad \big([a = b]\, \overline{c}a.\, P_1\big) \mid \big(d(x).\, P_2\big) \overset{([a=b][c=d],\tau)}{\leadsto} P_1 \mid P_2\{a/x\}$$

The new transition system is presented in Table 3. Its composite actions (M, α) are ranged over by μ. The definitions of substitution and bound names for them are as expected: $(M, \alpha)\sigma$ is $(M\sigma, \alpha\sigma)$ and $bn((M, \alpha))$ is $bn(\alpha)$. We extend the notation $M \rhd N$ to substitutions and write $\sigma \rhd \rho$ if σ equates more than ρ, i.e. $\rho(a) = \rho(b)$ implies $\sigma(a) = \sigma(b)$. Note that, in Table 3, if $P \overset{(M,\alpha)}{\leadsto} P'$, then no name bound in α appears in M. The bisimulation $\asymp$ which we define on the new transition system and which we shall prove to coincide with $\sim$ is nearly a ground bisimulation.

Definition 5.1 *A binary relation $\mathcal{S}$ on processes is a* $\asymp$-simulation *if $P\mathcal{S}Q$ implies*

- *whenever $P \overset{(M,\alpha)}{\leadsto} P'$ then N, β and Q' exist s.t. $Q \overset{(N,\beta)}{\leadsto} Q'$ and*
  - $M \rhd N$,
  - $\alpha\sigma_M \equiv \beta\sigma_M$,
  - $P'\sigma_M \;\mathcal{S}\; Q'\sigma_M$

$\mathcal{S}$ is a $\asymp$-bisimulation *is both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are* $\asymp$-simulations. *Two processes P and Q are* $\asymp$-bisimilar, *written $P \asymp Q$, if $P\mathcal{S}Q$ for some $\asymp$-bisimulation $\mathcal{S}$.*

Intuitively, the above clause ensures that, in the ordinary transition system, the move $P\sigma_M \xrightarrow{\alpha\sigma_M} P'\sigma_M$ is matched by the move $Q\sigma_M \xrightarrow{\beta\sigma_M} Q'\sigma_M$. The better efficiency of $\asymp$ over $\sim$ is evident, for $\asymp$ only forces those instantiations of names which are strictly necessary to ensure equivalence.

$$\textsf{op-pre:}\quad \alpha.\,P \overset{(\emptyset,\alpha)}{\leadsto} P \qquad\qquad \textsf{op-sum:}\quad \frac{P \overset{\mu}{\leadsto} P'}{P+Q \overset{\mu}{\leadsto} P'}$$

$$\textsf{op-rep:}\quad \frac{P \mid !\,P \overset{\mu}{\leadsto} P'}{!\,P \overset{\mu}{\leadsto} P'} \qquad\qquad \textsf{op-par:}\quad \frac{P \overset{\mu}{\leadsto} P'}{P \mid Q \overset{\mu}{\leadsto} P' \mid Q}\ \ bn(\mu) \cap fn(Q) = \emptyset$$

$$\textsf{op-com:}\quad \frac{P \overset{(M,\overline{a}y)}{\leadsto} P' \qquad Q \overset{(N,b(x))}{\leadsto} Q'}{P \mid Q \overset{(L,\tau)}{\leadsto} P' \mid Q'\{y/x\}} \qquad \text{where}\quad L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{otherwise} \end{cases}$$

$$\textsf{op-match:}\frac{P \overset{(M,\alpha)}{\leadsto} P'}{[a=b]P \overset{(N,\alpha)}{\leadsto} P'} \qquad \text{where}\quad N = \begin{cases} M[a=b] & \text{if } a \neq b \text{ and } a, b \notin bn(\alpha) \\ M & \text{if } a = b \end{cases}$$

Table 3: The transition system specialised for open bisimulation

---

We also expect that $\asymp$ be easier to verify than $\sim_L$ or $\sim_E$. In the input clause, $\sim_L$ and $\sim_E$ require the instantiation of the bound name with all names free in the examined processes plus a fresh name. As the number of input increases, this may cause an explosion of the number of process pairs to check. In contrast, with $\asymp$ it is enough to instantiate the bound name of the input with a *single* fresh name; any other instantiation only occurs when — and if — it is *needed* to perform a communication along that name. Indeed, this is the essence of the call-by-need nature of open bisimulation. We leave for future work a precise analysis of the complexity of $\asymp$ and the comparison with the complexity of the other equivalences.

Now we are ready to embark into the proof that $\asymp$ and $\sim$ coincide, which will occupy us until the end of the section. For this proof, the main lemmas are Lemma 5.4, where we show that $\asymp$ is closed under substitution, and Lemmas 5.5, 5.7, where we establish the operational correspondence between the ordinary and the new transition systems. In all lemmas below, by alpha conversion it is assumed that a substitution σ acts as an identity on the bound names of the processes and keeps the separation between bound and free names. Therefore, in $P\sigma \xrightarrow{a(x)\sigma} P'\sigma$, the free occurrences of x in P' are not modified by σ and $fn(P\sigma) \cap \{x\} = \emptyset$.

Lemma 5.2

1. *If* $P \overset{(M,\alpha)}{\leadsto} Q$, *then* $P\sigma \overset{(M',\alpha')}{\leadsto} Q'$, *with* $M' \lhd\rhd M\sigma$, $\alpha' \equiv \alpha\sigma$ *and* $Q' \equiv Q\sigma$*;*
2. *the converse, i.e. if* $P\sigma \overset{(M',\alpha')}{\leadsto} Q'$, *then* $P \overset{(M,\alpha)}{\leadsto} Q$, *with* $M' \lhd\rhd M\sigma$, $\alpha' \equiv \alpha\sigma$ *and* $Q' \equiv Q\sigma$.

Proof: Straightforward transition induction. □

Lemma 5.3

1. *If $M \rhd N$ then $M\sigma \rhd N\sigma$;*
2. *If $M \rhd N$ then $\sigma_M \rhd \sigma_N$;*
3. *If $\sigma \rhd \sigma'$ then ρ exists s.t. $\sigma = \sigma'\rho$.*

PROOF: Straightforward from the definitions of $\rhd$ on conditions and substitutions. □

Lemma 5.4 *$P \asymp Q$ implies $P\sigma \asymp Q\sigma$*

PROOF: We show that

$$\mathcal{S} = \bigcup_{\sigma} \{(P\sigma, Q\sigma) \,:\, P \asymp Q\}$$

is a $\asymp$-bisimulation. By Lemma 5.2(2), modulo $\lhd\rhd$ between match sequences, an action by $P\sigma$ can be written as $P\sigma \overset{(M,\alpha)\sigma}{\rightsquigarrow} P'\sigma$ for some M, α and P' s.t. $P \overset{(M,\alpha)}{\rightsquigarrow} P'$. Since $P \asymp Q$, we have $Q \overset{(N,\beta)}{\rightsquigarrow} Q'$ with

$$M \rhd N \tag{25}$$

$$\alpha\sigma_M \equiv \beta\sigma_M \tag{26}$$

$$P'\sigma_M \asymp Q'\sigma_M \tag{27}$$

From Lemma 5.2(1), modulo $\lhd\rhd$ between match sequences, $Q\sigma \overset{(N,\beta)\sigma}{\rightsquigarrow} Q'\sigma$. This gives a counterpart to the above transition of $P\sigma$ if

$$M\sigma \rhd N\sigma \tag{28}$$

$$\alpha\sigma\sigma_{M\sigma} \equiv \beta\sigma\sigma_{M\sigma} \tag{29}$$

$$P'\sigma\sigma_{M\sigma} \;\mathcal{S}\; Q'\sigma\sigma_{M\sigma} \tag{30}$$

Now, (28) follows from (25) and Lemma 5.3(1). For (29) and (30) we first express $\sigma\sigma_{M\sigma}$ in terms of σ_M. The implication

$$\sigma\sigma_{M\sigma} \rhd \sigma_M \tag{31}$$

holds if for any a, b, $\sigma_M(a) = \sigma_M(b)$ implies $\sigma_{M\sigma}(\sigma(a)) = \sigma_{M\sigma}(\sigma(b))$. Now, if $\sigma_M(a) = \sigma_M(b)$, then, by definition of σ_M, $M \rhd [a = b]$; therefore also $M\sigma \rhd [\sigma(a) = \sigma(b)]$, from which, by definition of $\sigma_{M\sigma}$, we have $\sigma_{M\sigma}(\sigma(a)) = \sigma_{M\sigma}(\sigma(b))$. Having showed (31), we can apply Lemma 5.3(3) and obtain $\sigma\sigma_{M\sigma} = \sigma_M\rho$, for some ρ. Finally, using this decomposition, (29) and (30) follow from (26) and (27). □

Lemma 5.5 *If $P\sigma_M \xrightarrow{\alpha} P''$ then $P \overset{(N,\beta)}{\rightsquigarrow} P'$ with*

- *$M \rhd N$;*
- *$\alpha \equiv \beta\sigma_M$;*
- *$P'' \equiv P'\sigma_M$.*

Proof: By transition induction. The hardest case is **com**. We leave the others to the reader. Consider the derivation

$$\frac{P_1\sigma_M \xrightarrow{\overline{a}y} P_1'' \qquad P_2\sigma_M \xrightarrow{a(x)} P_2''}{(P_1 \mid P_2)\sigma_M \xrightarrow{\tau} (P_1'' \mid P_2''\{y/x\})}$$

From the inductive assumption, we get

$$P_1 \overset{(N_1,\overline{b}c)}{\rightsquigarrow} P_1' \qquad\qquad P_2 \overset{(N_2,d(x))}{\rightsquigarrow} P_2' \tag{32}$$

with

$$M \rhd N_1, \qquad \overline{a}y \equiv (\overline{b}c)\sigma_M \qquad P_1'' \equiv P_1'\sigma_M \tag{33}$$

$$M \rhd N_2, \qquad a(x) \equiv (d(x))\sigma_M \qquad P_2'' \equiv P_2'\sigma_M \tag{34}$$

Now, from (32) and the rule **op-com**, we infer (we suppose $b \neq d$, the case $b = d$ is similar)

$$P_1 \mid P_2 \overset{(N_1N_2[b=d],\tau)}{\rightsquigarrow} P_1' \mid P_2'\{c/x\}$$

Finally, we obtain $(P_1' \mid P_2'\{c/x\})\sigma_M \equiv P_1'' \mid P_2''\{y/x\}$ and $M \rhd N_1N_2[b = d]$ from (33) and (34). For the former, we have:

$$(P_1' \mid P_2'\{c/x\})\sigma_M \equiv (P_1'\sigma_M \mid P_2'\sigma_M\{\sigma_M(c)/x\}) \equiv P_1'' \mid P_2''\{y/x\}$$

(since x is a bound name, we can assume that no name free in P_1' and $P_2'\{c/x\}$ is mapped by σ_M onto x). Now the latter: We have $M \rhd [b = d]$, since $\sigma_M(b) = \sigma_M(d) = a$; this, together with $M \rhd N_1$ and $M \rhd N_2$ gives $M \rhd N_1N_2[b = d]$, as required. ☐

Lemma 5.6 *If* $P \xrightarrow{\alpha} P'$ *then* $P\sigma \xrightarrow{\alpha\sigma} P'\sigma$.

Proof: By transition induction (see [MPW92]). ☐

Lemma 5.7 *If* $P \overset{(M,\alpha)}{\rightsquigarrow} P'$ *then* $P\sigma_M \xrightarrow{\alpha\sigma_M} P'\sigma_M$.

Proof: Another transition induction. The interesting rules are **op-com** and **op-match**. Consider the rule **op-com**, and suppose $a \neq b$; we have:

$$\frac{P_1 \overset{(M_1,\overline{a}y)}{\rightsquigarrow} P_1' \qquad P_2 \overset{(M_2,b(x))}{\rightsquigarrow} P_2'}{P_1 \mid P_2 \overset{(M_1M_2[a=b],\tau)}{\rightsquigarrow} P_1' \mid P_2'\{y/x\}}$$

Using Lemma 5.3 we can decompose $\sigma_{M_1M_2[a=b]}$ as $\sigma_{M_1}\rho_1$ and $\sigma_{M_2}\rho_2$, for some ρ_1, ρ_2. Moreover, by induction,

$$P_1\sigma_{M_1} \xrightarrow{(\overline{a}y)\sigma_{M_1}} P_1'\sigma_{M_1} \qquad\qquad P_2\sigma_{M_2} \xrightarrow{(b(x))\sigma_{M_2}} P_2'\sigma_{M_2}$$

Now we apply the above decomposition of $\sigma_{M_1M_2[a=b]}$ and Lemma 5.6 to infer with the rule **com**

$$\frac{P_1\sigma_{M_1}\rho_1 \xrightarrow{(\overline{a}y)\sigma_{M_1}\rho_1} P_1'\sigma_{M_1}\rho_1 \qquad P_2\sigma_{M_1}\rho_2 \xrightarrow{(b(x))\sigma_{M_2}\rho_2} P_2'\sigma_{M_2}\rho_2}{(P_1 \mid P_2)\sigma_{M_1M_2[a=b]} \xrightarrow{\tau} (P_1' \mid P_2'\{y/x\})\sigma_{M_1M_2[a=b]}}$$

The case of the rule **op-match** can be solved using Lemmas 5.3 and 5.6 in a similar way. ☐

Theorem 5.8 *$\sim$ implies $\asymp$.*

PROOF: We show that

$$\mathcal{S} = \{(P,Q) \,:\, P \sim Q\}$$

is a $\asymp$-bisimulation. Suppose $P \overset{(M,\alpha)}{\rightsquigarrow} P'$. By Lemma 5.7,

$$P\sigma_M \xrightarrow{\alpha\sigma_M} P'\sigma_M$$

From $P \sim Q$ and Proposition 3.9, also $P\sigma_M \sim Q\sigma_M$, hence

$$Q\sigma_M \xrightarrow{\beta} Q''$$

with

$$\alpha\sigma_M \equiv \beta \qquad\qquad P'\sigma_M \sim Q'' \tag{35}$$

By Lemma 5.5, $Q \overset{(N,\gamma)}{\rightsquigarrow} Q'$ with

$$M \rhd N\,, \qquad \gamma\sigma_M \equiv \beta\,, \qquad Q'\sigma_M \equiv Q'' \tag{36}$$

Thus, from (35) and (36), we have $Q \overset{(N,\gamma)}{\rightsquigarrow} Q'$ with $M \rhd N$, $\gamma\sigma_M \equiv \alpha\sigma_M$, and $(P'\sigma_M, Q'\sigma_M) \in \mathcal{S}$, which is what required in the definition of $\asymp$- bisimulation. $\Box$

Theorem 5.9 *$\asymp$ implies $\sim$.*

PROOF: The relation

$$\mathcal{S} = \{(P,Q) \,:\, P \asymp Q\}$$

is a $\sim$-bisimulation. By Lemma 5.4, $\mathcal{S}$ is closed under substitutions; by Proposition 3.8 we have only to show that $\mathcal{S}$ is a ground bisimulation. Suppose $P \xrightarrow{\alpha} P'$. By Lemma 5.5, $P \overset{(\emptyset,\alpha)}{\rightsquigarrow} P'$. Since $P \asymp Q$, we also have $Q \overset{(\emptyset,\alpha)}{\rightsquigarrow} Q' \asymp P'$. From Lemma 5.7, $Q \xrightarrow{\alpha} Q'$, which closes up the bisimulation. $\Box$

Transition systems and bisimulations similar to those defined in this section (and with similar "efficiency" motivations in mind) have been considered by Hennessy and Lin [HL92] and Amadio [Ama92, section 5]. Hennessy and Lin work in CCS with value-passing and their *symbolic bisimulations* intend to capture the standard bisimulations of this calculus (rather than a new one as we do in this paper); our $\asymp$ appears simpler than their symbolic bisimulations since in the clause of bisimilarity of the latter an action by a process may be matched by a *set* of actions (as opposite to a *single* action) from the other process. Amadio works in a subcalculus of ours, on which the relation he introduces is similar to our $\asymp$; however, the theory of such a relation is not developed.

6 Distinctions

In this section we outline the generalisation of the theory in the previous sections to deal with *distinctions* [MPW92]. A distinction expresses permanent inequalities on names, i.e. if (a, b) are in the distinction, then a must be kept separate from b. We use D to range over distinctions.

Definition 6.1 *A* distinction *is a finite symmetric and irreflexive relation on names. A substitution* σ respects *a distinction D if $(a, b) \in D$ implies $\sigma(a) \neq \sigma(b)$. Similarly, a match sequence M* respects *a distinction D if $(a, b) \in D$ implies $M \not\triangleright [a = b]$.*

The finiteness requirement on distinctions is imposed to ensure that there are always enough names available to permit alpha conversion to fresh names. We denote by $n(D)$ the set of names which are mentioned in D. Sometimes, in the expressions defining distinctions we shall avoid to give all symmetric pairs; for instance, we might define $D = \{(a, b)\}$ without recalling that also $(b, a) \in D$. We write $D - x$ for the distinction

$$\{(a, b) \ : \ (a, b) \in D \text{ and } a, b \neq x\}$$

and, if σ is a substitution which respects D, we write $D\sigma$ for the distinction

$$\{(\sigma(a), \sigma(b)) \ : \ (a, b) \in D\}.$$

To allow for distinctions, the definitions of the bisimulations use a set of relations; each of them is indexed by a different distinction. In the statements in this and in the next section, a name declared *fresh* is supposed to be different from any other name appearing in the objects of the statement, like processes or distinctions; this will avoid ambiguity on the consistency of the distinctions we define.

Definition 6.2 *The set $\mathcal{S} = \{\mathcal{S}_D\}_D$ of process relations is an* indexed open simulation *if for each $\mathcal{S}_D$ and for each σ which respects D, $P \, \mathcal{S}_D \, Q$ implies*

- *whenever $P\sigma \xrightarrow{\alpha} P'$ with $bn(\alpha)$ fresh, then Q' exists s.t. $Q\sigma \xrightarrow{\alpha} Q'$ and $P' \, \mathcal{S}_{D\sigma} \, Q'$.*

$\mathcal{S}$ is an indexed open bisimulation *if both $\{\mathcal{S}_D\}_D$ and $\{\mathcal{S}_D^{-1}\}_D$ are indexed open simulations. The processes P and Q are* open D-bisimilar, *written $P \sim_D Q$, if there is an indexed open bisimulation $\mathcal{S}$ s.t. $P \, \mathcal{S}_D \, Q$.*

Note that $\sim$ corresponds to the special case of $\sim_D$ in which D is empty. The class $\{ \sim_D \}_D$ ordered by relation inclusion, almost gives rise to a lattice: $\sim$ is the bottom element, and two elements $\sim_D$, $\sim_{D'}$ have $\sim_{D \cap D'}$ as their greatest lower bound and $\sim_{D \cup D'}$ as their least upper bound; there is no top element though, due to the fact that we only allow finite distinctions. A relation $\sim_D$ enjoys a weak form of congruence w.r.t. input prefix, namely

$$\text{if } P \sim_{D-x} Q \text{ then } a(x).\, P \sim_D a(x).\, Q \tag{37}$$

The other operators of the language preserve $\sim_D$. Therefore, $\sim_D$ is a full congruence under the assumption that the names of D are not used as bound names in the process expressions.

The axiom system for $\sim_D$ is obtained by adding two simple axioms to the system $\mathcal{E}$ for $\sim$ (we take equalities $P = Q$ in $\mathcal{E}$ as abbreviations for $P =_\emptyset Q$):

$$\begin{array}{lrrcl}
\mathbf{P} & \text{From } (a,b) \in D \text{ infer} & [a=b]\,P & =_D & \mathbf{0} \\
\mathbf{W} & \text{From } P =_{D'} Q \text{ and } D' \subseteq D \text{ infer} & P & =_D & Q
\end{array}$$

P stands for "Pruning", **W** for "Weakening". Let $\mathcal{D}$ be $\mathcal{E} \cup \{\mathbf{P}, \mathbf{W}\}$. The soundness of $\mathcal{E}$ has been considered in Section 4; the soundness of **P** is self-evident. The soundness of **W** comes form the following lemma:

Lemma 6.3 *If $D' \subseteq D$, then $P \sim_{D'} Q$ implies $P \sim_D Q$.*

Proof: The requirements in the definition of $\sim_{D'}$ are a superset of those in the definition of $\sim_D$. □

Definition 6.4 *A term P is in* strong normal form on D, *(D-snf in short) if*

$$P \equiv \sum_{i \in I} M_i \alpha_i P_i \qquad \textit{and}$$

1. *for all i, M_i respects D,*
2. *for all i, $bn(\alpha_i)$ is fresh and $P_i\sigma_{M_i}$ is in $D\sigma_{M_i}$-snf,*
3. *if $i \neq j$, then $M_i\alpha_i P_i \not\sim_D M_i\alpha_i P_i + M_j\alpha_j P_j$.*

The snf's of Definition 4.8 are a special case of D-snf's, in which $D = \emptyset$. Note also that if P is in D-snf, then P is also in D'-snf, for $D' \subseteq D$ (clause (3) holds because of Lemma 6.3).

The following lemma, together with Lemma 4.5 — to put a process into nf — and axiom **W** — to lift each equality $=$ up to $=_D$ — gives the completeness of $\mathcal{D}$ for $\sim_D$.

Lemma 6.5

1. *If P and Q are D-snf's and $P \sim_D Q$, then $\{\mathbf{C1\text{-}4}, \mathbf{A}, \mathbf{S2}, \mathbf{S3}, \mathbf{M1}, \mathbf{M2}\} \vdash P = Q$.*
2. *If P is a nf, then there is a D-snf H s.t. $\mathcal{D} \vdash P =_D H$; moreover $fn(H) \subseteq fn(P)$ and $\texttt{len}(H) \leq \texttt{len}(P)$.*

Proof: Completely analogous to the proofs of Lemmas 4.10 and 4.11. Be birefly summarise the proofs. Both of them proceed by induction on the maximal depth of the given processes. For the assertion (1), one shows that each summand $M\alpha.\,P_1$ of P can be transformed into a summand of Q, as follows. Given the action

$$(M\alpha.\,P_1)\sigma_M \xrightarrow{\alpha\sigma_M} P_1\sigma_M,$$

from the definition of $\sim_D$ one finds a summand $N\beta.Q_1$ of Q with $M \rhd N$ which can match the previous action. For this, the first clause in the definition of D-snf is needed to ensure that σ_M respects D. Then, reasoning by absurd and exploiting the third clause in the definition of D-snf, one infers that also $N \rhd M$ must hold. Finally, the condition $M \lhd\rhd N$ and the inductive assumption are used to equate $N\beta.Q_1$ to $M\alpha.P_1$.

We turn now to the inductive part of the assertion (2) of the lemma. For the first clause of the definition of D-snf, one uses **P** (possibly in conjunction with **M1**) and **MD1**, **S1** to remove all subterms which at the top have a condition inconsistent with D, and **W** to weaken $=$ with $=_D$. The other clauses can be dealt with as in Lemma 4.11. We only recall the proof schema for the third clause. Suppose that M_i and M_j respect D (we can assume this because of the first clause dealt with above), and $M_i\alpha_iP_i \sim_D M_i\alpha_iP_i + M_j\alpha_jP_j$. This holds if $M_j \rhd M_i$ and $M_jM_i\alpha_iP_i \sim_D M_j\alpha_jP_j$. Using the inductive assumption, the terms $M_j\alpha_jP_j$ and $M_jM_i\alpha_iP_i$ can be put into D-snf, and by the assertion (1) of the lemma the resulting D-snf's can be equated each other. Hence we can conclude that

$$\mathcal{D} \vdash M_i\alpha_iP_i =_D M_i\alpha_iP_i + M_jM_i\alpha_iP_i =_D M_i\alpha_iP_i + M_j\alpha_iP_i =_D M_i\alpha_iP_i + M_j\alpha_jP_j$$

All other redundant summands of P can be canceled in a similar way. □

Lemma 6.5(1) also shows that the D-snf's, such as the snf's, are essentially *unique*. Therefore, on D-snf's $\sim_D$ and $\sim$ coincide. This property breaks if we remove clause (3) in Definition 6.4, i.e. we consider normal forms rather strong normal forms. For instance, the processes

$$P \overset{def}{=} \tau.[c=d]\tau \qquad\qquad Q \overset{def}{=} P + [a=c][b=d]\tau$$

satisfy clauses (1) and (2) of Definition 6.4 and are in the relation $\sim_{\{(a,b)\}}$; but they are not in the relation $\sim$.

The other results presented for $\sim$ in the previous sections can be generalised to $\sim_D$ with the expected modifications in the assertions and in the proofs. As example, we consider the new version of Proposition 3.8 and the definition of the efficient characterisation $\asymp_D$. For the former, we have:

- Suppose that $\mathcal{S} = \{\mathcal{S}_D\}_D$ is a class of relations which are ground bisimulations and closed under respectful substitutions (i.e. $P\,\mathcal{S}_D\,Q$ and σ respects D implies $P\sigma\,\mathcal{S}_{D\sigma}\,Q\sigma$). Then $\mathcal{S}$ is an open indexed bisimulation.

Moreover $\{\sim_D\}_D$ represents the class of the largest relations which satisfy the above property. Also the definition of $\asymp_D$ uses a set $\{\mathcal{S}_D\}_D$ of process relations indexed by distinctions; the requirement on the bisimilarity is that $P\,\mathcal{S}_D\,Q$ implies

- whenever $P \overset{(M,\alpha)}{\leadsto} P'$ with $bn(\alpha)$ fresh, and M respects D, then N, β and Q' exist s.t. $Q \overset{(N,\beta)}{\leadsto} Q'$ and
  - $M \rhd N$,

- $\alpha\sigma_M \equiv \beta\sigma_M$,
- $P'\sigma_M \; \mathcal{S}_{D\sigma_M} \; Q'\sigma_M$.

Alternatively, the condition "M respects D" can be moved inside the transition system by adding the premise $(a,b) \notin D$ in the rules **op-match** and **op-com**.

7 The restriction operator

The work in the previous section shows the central position of $\sim_D$ in the theory we are developing, with $\sim$ recovered for a special choice of distinction. Here we show how $\sim_D$ behaves with restriction, the π-calculus operator so far ignored. The restriction of the name x in the process P is written $(x)\,P$ (in some papers $\boldsymbol{\nu} x\; P$) and declares the name x local to P. The scope of x can however be widened, as a consequence of a communication of it. The output $(x)\overline{a}x.\,P$ is abbreviated to $\overline{a}(x).\,P$, and $\overline{a}(x)$ called *bound output*.

To take restriction into account, one has to add the appropriate rules for restriction in the transition and the axiom systems, and a clause for bound output in the definition of the bisimulations. We consider this below. These extensions require fairly simple adaptations to the previous results and proofs[1] about $\sim_D$; the only proof we discuss is the completeness of the axiom system, since it needs a little more thought.

The π-calculus transition system has three rules specific to restriction. Note that the prefixes on which α ranges must now include bound outputs as well. We have omitted the symmetric version of **close**.

$$\textsf{close:}\ \frac{P \xrightarrow{\overline{a}(x)} P' \qquad Q \xrightarrow{a(x)} Q'}{P \mid Q \xrightarrow{\tau} (x)\,(P' \mid Q')}$$

$$\textsf{res:}\frac{P \xrightarrow{\alpha} P'}{(b)\,P \xrightarrow{\alpha} (b)\,P'} \quad b \notin n(\alpha) \qquad\qquad \textsf{open :}\frac{P \xrightarrow{\overline{a}b} P'}{(b)\,P \xrightarrow{\overline{a}(b)} P'} \quad b \neq a$$

Now the corresponding rules in the transition system specialised for $\sim_D$; we write $n((M,\alpha))$ for $n(M) \cup n(\alpha)$.

$$\textsf{op-close:}\frac{P \overset{(M,\overline{a}(x))}{\rightsquigarrow} P' \qquad Q \overset{(N,b(x))}{\rightsquigarrow} Q'}{P \mid Q \overset{(L,\tau)}{\rightsquigarrow} (x)\,(P' \mid Q')} \qquad \text{where} \quad L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{otherwise} \end{cases}$$

$$\textsf{op-res:}\frac{P \overset{\mu}{\rightsquigarrow} P'}{(b)\,P \overset{\mu}{\rightsquigarrow} (b)\,P'} \quad b \notin n(\mu) \qquad\qquad \textsf{op-open :}\frac{P \overset{(M,\overline{a}b)}{\rightsquigarrow} P'}{(b)\,P \overset{(M,\overline{a}(b))}{\rightsquigarrow} P'} \quad b \notin n(M) \cup \{a\}$$

---

[1] In the proof of congruence for $\sim_D$ over parallel composition, the technique of bisimulation up-to restriction [MPW92] is needed.

Restriction	**CR**	$(x)P$	$=_{D-x}$	$(x)Q$	if $P =_D Q$
	R1	$(x)\mathbf{0}$	$=$	$\mathbf{0}$	
	R2	$(x)(y)P$	$=$	$(y)(x)P$	
	R3	$(x)(P+Q)$	$=$	$(x)P + (x)Q$	
	R4	$(x)\alpha.\,P$	$=$	$\alpha.\,(x)P$	if $x \notin n(\alpha)$
	R5	$(x)\alpha.\,P$	$=$	$\mathbf{0}$	if $\alpha \equiv \overline{x}y$ or $\alpha \equiv x(y)$
	R6	$(x)[y=z]P$	$=$	$[y=z](x)P$	if $x \neq y, x \neq z$

Table 4: The axioms for restriction

This enlarged transition system still has the property that if $P \overset{(M,\alpha)}{\rightsquigarrow} P'$ then no name bound in α occurs in M. Note in op-open the side condition $b \notin n(M)$: A restricted name is logically different from any known name and hence an equality in M involving this name is unsatisfiable. This same principle rules the clause for bound outputs in bisimulations. Thus, for $\sim_D$ we have:

- if $P\sigma \xrightarrow{\overline{a}(b)} P'$ with b fresh, then Q' exists s.t. $Q\sigma \xrightarrow{\overline{a}(b)} Q'$ and $P'\mathcal{S}_{D'}Q'$, for $D' = D\sigma \cup \{\{b\} \times fn(P\sigma, Q\sigma)\}$.

The distinction $D\sigma$ is augmented to D' to keep b separate from the names visible in $P\sigma$ and $Q\sigma$. It is worth pointing out that the adoption of a similar clause — i.e., with increment of the distinction — forces the use of a set of distinction-indexed relations also in the definition of ground bisimulation.

The axioms for restriction are reported in Table 7. **CR** is for congruence. Its soundness can be proved by exhibiting the appropriate bisimulation. Compare **CR** with the rule (37) for the congruence of $\sim_D$ w.r.t. input prefix: The opposite positioning of $D - x$ reveals the different nature of the two binding operators involved. **R1**–**R6** say how and when restriction distributes over the π-calculus operators. Moreover, in the formulation of the expansion law in Section 4, we have to take into account the case in which P and Q exchange a private name; this is done by adding the clauses for bound outputs in the definition of the terms R_{ij}'s, as in [MPW92] or [PS93]. We briefly discuss how to adapt the completeness proof for $\sim_D$ to the language extended with restriction. First note that the axiom

$$\mathbf{RD1}\quad (x)[x=y]P \;=\; \mathbf{0} \quad \text{if } x \neq y$$

is derivable: By pruning, $[x = y]P =_{\{(x,y)\}} \mathbf{0}$; using **CR**, we get $(x)[x = y]P = (x)\mathbf{0}$, whose second member can be equated to $\mathbf{0}$ with **R1**. In the clause (2) of Definition 6.4 of D-snf, we have to distinguish the case in which α_i is a bound output, since underneath it a distinction must be augmented. Thus we require:

if α_i is a bound output with bound name x fresh, then P_i is in D'-snf, for
$D' = D\sigma_{M_i} \cup \{\{x\} \times fn(P\sigma_{M_i})\}$.

The axioms for restriction are only employed in the derivation of D-snf's: They allow us to strengthen the D-snf underneath a restriction (**CR**) and to push a restriction inside a term (**R2**, **R3**, **R4**, **R6**) until either it disappears (**R1**, **R5**, **RD1**) or it gives rise to a bound output. Everything else in the completeness proof for $\mathcal{D}$ is left almost unchanged.

8 Conclusions

In this paper we have presented a new formulation of bisimulation for the π-calculus, called open bisimulation. Its difference from the previously known bisimulation-based equivalences is already present in the sublanguage without name restrictions: Here the definition of open bisimulation can be factorised into a "standard" part which, modulo the different syntax of actions, is the CCS bisimulation, and a part specific for π-calculus, which demands name instantiation. We have derived an efficient characterisation of open bisimulation, which also shows its call-by-need flavour. We have given a sound and complete axiomatisation on finite terms. There are a few interesting points which emerge from the completeness proof. Fairly simple transformations reconduct open bisimilarity between finite terms to syntactic equality on their trees. This, on the one hand, suggests that open bisimulation may represent the finest extensional equality on π-calculus terms one would like to impose. On the other hand, it has led us to the construction of canonical (and minimal) representatives for the equivalence classes of open bisimulation. This kind of result is unusual for value-passing process calculi. For instance, the axiomatisations of the π-calculus equivalences so far appeared in the literature fail to provide canonical representatives because, in some cases, they use normal forms parametrised upon the set of free names of the processes, in the other cases, they use head normal forms which apparently cannot be transformed into normal forms. We think that in π-calculus it would be difficult to obtain canonical representatives (for some reasonable form of behavioural equivalence) if *mismatching*, i.e. testing for inequality of names, is present in the calculus. Mismatching was excluded from the original syntax of π-calculus for no special theoretical or pragmatical motivation except the preservation of the following property, asserting that substitutions may only increase the action capabilities of an agent:

$$\text{if } P \xrightarrow{\alpha} P' \text{ then also } P\sigma \xrightarrow{\alpha\sigma} P'\sigma.$$

The study conducted in this paper highlights some possible advantages in only having matching: Indeed, the theory developed for open bisimulation — in particular the completeness proof of the axiomatisation — is strongly based on the above monotonicity property. The definition itself of open bisimulation does not make sense with mismatchings, since their appearance conflicts with the requirement of closure under all substitutions. We do not know at the present what is the most reasonable way to define open bisimulation on a calculus with mismatching.

We would like to investigate whether the verification of open bisimulation can be mechanised exploiting the efficient characterisation presented in Section 5. This would be important since at present there is a total lack of software tools to reason with calculi for mobile processes. The comparison between open bisimulation and the late/early equivalences should be developed. The axiomatisations presented here and in [PS93] might give us some guidance, although there is the obstacle that the calculus used in [PS93] also includes mismatching. Another direction is to look at applications in which π-calculus has been used, like the modeling of λ-calculus [Mil92b] and of higher-order calculi [San92], and see whether the adoption of open bisimulation affects the process equivalences obtained in these cases.

Acknowledgements

I am very grateful to Claudio Calvelli, Robin Milner and David N. Turner for inspiring discussions and suggestions.

References

[Ama92] R. Amadio. A uniform presentation of CHOCS and π-calculus. Rapport de recherche 1726, INRIA-Lorraine, Nancy, 1992.

[BD92] M. Boreale and R. De Nicola. Testing equivalence for mobile processes. In R. Cleaveland, editor, *3rd CONCUR*, volume 630 of *Lecture Notes in Computer Science*, pages 2–16. Springer Verlag, 1992.

[BK85] J.A. Bergstra and J.W. Klop. Algebra for communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[DKV91] P. Degano, S. Kasangian, and S. Vigna. Applications of the calculus of trees to process description languages. In *Proc. of the CTCS '91 Conference*, volume 530 of *Lecture Notes in Computer Science*, pages 281–301. Springer Verlag, 1991.

[Hen91] M. Hennessy. A model for the π-calculus. 91/08, Department of Computer Science, University of Sussex, 1991.

[HL92] M. Hennessy and H. Lin. Symbolic bisimulations. Revised Version of Technical Report TR 1/92, Department of Computer Science, University of Sussex, 1992.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:269–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mil92a] R. Milner. Action structures. Technical Report ECS–LFCS–92–249, LFCS, Dept. of Comp. Sci., Edinburgh Univ., December 1992.

[Mil92b] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[MPW91] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. In *2nd CONCUR*, volume 527 of *Lecture Notes in Computer Science*, pages 45–60. Springer Verlag, 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[MS92] U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, XVI(2):171–199, 1992.

[Par81] D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.

[PS93] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. Technical Report ECS–LFCS–93–262, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1993.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* CST–99–93, Department of Computer Science, University of Edinburgh, 1992. Also published as ECS–LFCS–93–266.

[Wad71] C. P. Wadsworth. *Semantics and pragmatics of the lambda calculus.* PhD thesis, University of Oxford, 1971.