Comparing the expressiveness of the -calculus and CCS

Comparing the expressiveness of the π-calculus and CCS

Rob van Glabbeek[1,2] ✉

[1] Data61, CSIRO, Sydney, Australia
[2] School of Comp. Sc. and Engineering, Univ. of New South Wales, Sydney, Australia
rvg@cs.stanford.edu

Abstract. This paper shows that the π-calculus with implicit matching is no more expressive than CCS$_\gamma$, a variant of CCS in which the result of a synchronisation of two actions is itself an action subject to relabelling or restriction, rather than the silent action τ. This is done by exhibiting a compositional translation from the π-calculus with implicit matching to CCS$_\gamma$ that is valid up to strong barbed bisimilarity.
The full π-calculus can be similarly expressed in CCS$_\gamma$ enriched with the triggering operation of Meije.
I also show that these results cannot be recreated with CCS in the rôle of CCS$_\gamma$, not even up to reduction equivalence, and not even for the asynchronous π-calculus without restriction or replication.
Finally I observe that CCS cannot be encoded in the π-calculus.

1 Introduction

The π-calculus [23,24,22,33] has been advertised as an "extension to the process algebra CCS" [23] adding mobility. It is widely believed that the π-calculus has features that cannot be expressed in CCS, or other *immobile* process calculi—so called in [27]—such as ACP and CSP.

"the π-calculus has a much greater expressiveness than CCS"
[Sangiorgi [32]]

"Mobility – of whatever kind – is important in modern computing. It was not present in CCS or CSP, [...] but [...] the π-*calculus* [...] takes mobility of linkage as a primitive notion." [Milner [22]]

The present paper investigates this belief by formally comparing the expressive power of the π-calculus and immobile process calculi.

Following [10,11] I define one process calculus to be at least as expressive as another up to a semantic equivalence $\sim$ iff there exists a so-called *valid translation* up to $\sim$ from the other to the one. Validity entails compositionality, and requires that each translated expression is $\sim$-equivalent to its original. This concept is parametrised by the choice of a semantic equivalence that is meaningful for both the source and the target language. Any language is as expressive as any other up to the universal relation, whereas almost no two languages are equally expressive up to the identity relation. The equivalence $\sim$ up to which a translation is valid is a measure for the quality of the translation, and thereby for the degree in which the source language can be expressed in the target.

Robert de Simone [34] showed that a wide class of process calculi, including CCS [20], CSP [6], ACP [4] and SCCS [18], are expressible up to strong bisimilarity in MEIJE [1]. In [8] I sharpened this result by eliminating the crucial rôle played by unguarded recursion in De Simone's translation, now taking aprACP$_R$ as the target language. Here aprACP$_R$ is a fragment of the language ACP of [4], enriched with relational relabelling, and using action prefixing instead of general sequential composition. It differs from CCS only in its more versatile communication format, allowing multiway synchronisation instead of merely handshaking, in the absence of a special action τ, and in the relational nature of the relabelling operator. The class of languages that can be translated to MEIJE and aprACP$_R$ are the ones whose structural operational semantics fits a format due to [34], now known as the *De Simone* format. They can be considered the "immobile process calculi" alluded to above. The π-calculus does not fit into this class—its operational semantics is not in De Simone format.

To compare the expressiveness of mobile and immobile process calculi I first of all need to select a suitable semantic equivalence that is meaningful for both kinds of languages. A canonical choice is *strong barbed bisimilarity* [26,33]. Strong barbed bisimilarity is not a congruence for either CCS or the π-calculus, but it is used as a semantic basis for defining suitable congruences on languages [26,33]. For CCS, the familiar notion of *strong bisimilarity* [19] arises as the congruence closure of strong barbed bisimilarity. For the π-calculus, the congruence closure of strong barbed bisimilarity yields the notion of *strong early congruence*, called *strong full bisimilarity* in [33]. In general, whatever its characterisation in a particular calculus, *strong barbed congruence* is the name of the congruence closure of strong barbed bisimilarity, and a default choice for a semantic equivalence [33].

My first research goal was to find out if there exists a translation from the π-calculus to CCS that is valid up to strong barbed bisimilarity. The answer is negative. In fact, no compositional translation of the π-calculus to CCS is possible, even when weakening the equivalence up to which it should be valid from strong barbed bisimilarity to strong reduction equivalence, and even when restricting the source language to the asynchronous π-calculus [5] without restriction and replication. This disproves a result of [3].

My next research goal was to find out if there is a translation from the π-calculus to any other immobile process calculus, and if yes, to keep the target language as close as possible to CCS. Here the answer turned out to be positive. How close the target language can be kept to CCS depends on which version of the π-calculus I take as source language. My first choice was the original π-calculus, as presented in [23,24], as it is at least as expressive as its competitors. It turns out, however, that the matching operator $[x{=}y]P$ of [23,24] is the source of a complication. The book [33] merely allows matching to occur as part of action prefixing, as in $[x{=}y]u(z).P$ or $[x{=}y]\bar{u}v.P$. I call this *implicit matching*. Matching was introduced in [23,24] to facilitate complete equational axiomatisations of the π-calculus, and [33] shows that for that purpose implicit matching is sufficient.

To obtain a valid translation from the π-calculus with implicit matching (henceforth called π_{IM}) to an upgraded variant of CCS, the only upgrade needed is to turn the result of a synchronisation of two actions into a visible action, subject to relabelling or restriction, rather than the silent action τ. I call this variant CCS$_\gamma$, where γ is a commutative partial binary communication function, just like in ACP [4]. CCS$_\gamma$ is a fragment of aprACP$_R$, which also carries a parameter γ. If $\gamma(a,b)=c$, this means that an a-action of one component in a parallel composition may synchronise with a b-action of another component, into a c-action; if $\gamma(a,b)$ is undefined, the actions a and b do not synchronise. CCS can be seen as the instance of CCS$_\gamma$ with $\gamma(\bar{a},a)=\tau$, and γ undefined for other pairs of actions. But as target language for my translation I will need another choice of the parameter γ.

An important feature of ACP, which greatly contributes to its expressiveness, is multiway synchronisation. This is achieved by allowing an action $\gamma(a,b)$ to synchronise with an action c into $\gamma(\gamma(a,b),c)$. This feature is not needed for the target language of my translations. So I require that $\gamma(\gamma(a,b),c)$ is always undefined.

To obtain a valid translation from the full π-calculus, with an explicit matching operator, I need to further upgrade CCS$_\gamma$ with the *triggering* operator of Meije, which allows a relabelling of the first action of its argument only.

By a general result of [11], the validity up to strong barbed bisimilarity of my translation from π_{IM} to CCS$_\gamma$ (and from π to CCS$_\gamma^{\mathrm{trig}}$) implies that it is even valid up to an equivalence on their disjoint union that on π coincides with strong barbed congruence, or strong early congruence, and on CCS$_\gamma^{\mathrm{trig}}$ is the congruence closure of strong barbed bisimilarity under translated contexts. The latter is strictly coarser than strong bisimilarity, which is the congruence closure of strong barbed bisimilarity under all CCS$_\gamma^{\mathrm{trig}}$ contexts.

Having established that π_{IM} can be expressed in CCS$_\gamma$, the possibility remains that the two languages are equally expressive. This, however, is not the case. There does not exists a valid translation (up to any reasonable equivalence) from CCS—thus neither from CCS$_\gamma$—to the π-calculus, even when disallowing the infinite sum of CCS, as well as unguarded recursion. This is a trivial consequence of the power of the CCS renaming operator, which cannot be mimicked in the π-calculus. Using a simple renaming operator that is as finite as the successor function on the natural numbers, CCS, even without infinite sum and unguarded recursion, allows the specification of a process with infinitely many weak barbs, whereas this is fundamentally impossible in the π-calculus.

2 CCS

CCS [19] is parametrised with a sets $\mathcal{K}$ of *agent identifiers* and $\mathscr{A}$ of *visible actions*. The set $\overline{\mathscr{A}}$ of *co-actions* is $\overline{\mathscr{A}} := \{\bar{a} \mid a \in \mathscr{A}\}$, and $\mathscr{L} := \mathscr{A} \cup \overline{\mathscr{A}}$ is the set of *labels*. The function $\bar{\cdot}$ is extended to $\mathscr{L}$ by declaring $\bar{\bar{a}} = a$. Finally, $Act := \mathscr{L} \uplus \{\tau\}$ is the set of *actions*. Below, a, b, c, ... range over $\mathscr{L}$ and α, β over Act. A *relabelling* is a function $f\colon \mathscr{L} \to \mathscr{L}$ satisfying $f(\bar{a}) = \overline{f(a)}$; it extends

Table 1. Structural operational semantics of CCS

$$\alpha.P \xrightarrow{\alpha} P \qquad \frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i\in I} P_i \xrightarrow{\alpha} P'_j}\ (j \in I)$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \frac{P \xrightarrow{a} P',\ Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L}\ (\alpha \notin L \cup \bar{L}) \qquad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]} \qquad \frac{P \xrightarrow{\alpha} P}{A \xrightarrow{\alpha} P}\ (A \overset{\text{def}}{=} P)$$

to *Act* by $f(\tau) := \tau$. The class $\mathrm{T}_{\mathrm{CCS}}$ of CCS *terms*, *expressions*, *processes* or *agents* is the smallest class[1] including:

$\alpha.P$	for $\alpha \in \mathit{Act}$ and $P \in \mathrm{T}_{\mathrm{CCS}}$	*prefixing*
$\sum_{i\in I} P_i$	for I an index set and $P_i \in \mathrm{T}_{\mathrm{CCS}}$	*choice*
$P\|Q$	for $P, Q \in \mathrm{T}_{\mathrm{CCS}}$	*parallel composition*
$P\backslash L$	for $L \subseteq \mathscr{L}$ and $P \in \mathrm{T}_{\mathrm{CCS}}$	*restriction*
$P[f]$	for f a relabelling and $P \in \mathrm{T}_{\mathrm{CCS}}$	*relabelling*
A	for $A \in \mathcal{K}$	*recursion.*

One writes $P_1 + P_2$ for $\sum_{i\in I} P_i$ when $I = \{1,2\}$, and $\mathbf{0}$ when $I = \emptyset$. Each agent identifier $A \in \mathcal{K}$ comes with a unique *defining equation* of the form $A \overset{\text{def}}{=} P$, with $P \in \mathrm{T}_{\mathrm{CCS}}$. The semantics of CCS is given by the labelled transition relation $\rightarrow \subseteq \mathrm{T}_{\mathrm{CCS}} \times \mathit{Act} \times \mathrm{T}_{\mathrm{CCS}}$. The transitions $P \xrightarrow{\alpha} Q$ with $P, Q \in \mathrm{T}_{\mathrm{CCS}}$ and $\alpha \in \mathit{Act}$ are derived from the rules of Table 1.

Arguably, the most authentic version of CCS [20] features a recursion construct instead of agent identifiers. Since there exists a straightforward valid transition from the version of CCS presented here to the one from [20], the latter is at least as expressive. Therefore, when showing that a variant of CCS is at least as expressive as the π-calculus, I obtain a stronger result by using agent identifiers.

3 CCS$_\gamma$

CCS$_\gamma$ has four parameters: the same set $\mathcal{K}$ of *agent identifiers* as for CCS, an alphabet $\mathscr{A}$ of *visible actions*, with a subset $\mathscr{S} \subseteq \mathscr{A}$ of synchronisations[2], and a

[1] CCS [19,20] allows arbitrary index sets I in summations $\sum_{i\in I} P_i$. As a consequence, $\mathrm{T}_{\mathrm{CCS}}$ is a proper class rather than a set. Although this is unproblematic, many computer scientists prefer the class of terms to be a set. This can be achieved by choosing a cardinal κ and requiring the index sets I to satisfy $|I| < \kappa$. To enable my translation from the π-calculus to $\mathrm{CCS}_\gamma^{\mathrm{trig}}$, κ should exceed the size of the set of names used in the π-calculus.

[2] These have been added solely to prevent multiway synchronisation.

partial *communication function* $\gamma : (\mathscr{A}\backslash\mathscr{S})^2 \rightharpoonup \mathscr{S} \cup \{\tau\}$, which is commutative, i.e. $\gamma(a,b) = \gamma(b,a)$ and each side of this equation is defined just when the other side is. Compared to CCS there are no co-actions, so $Act := \mathscr{A} \uplus \{\tau\}$.

The syntax of CCS$_\gamma$ is the same as that of CCS, except that parallel composition is denoted $\|$ rather than $|$, following ACP [4,2]. This indicates a semantic difference: the rule for communication in the middle of Table 1 is for CCS$_\gamma$ replaced by

$$\frac{P \xrightarrow{a} P',\ Q \xrightarrow{b} Q'}{P\|Q \xrightarrow{c} P'\|Q'} \quad (\gamma(a,b) = c).$$

Moreover, relabelling operators $f : \mathscr{A} \rightarrow Act$ are allowed to rename visible actions into τ, but not vice versa.[3] They are required to satisfy $c \in \mathscr{S} \Rightarrow f(c) \in \mathscr{S} \cup \{\tau\}$. These are the only differences between CCS and CCS$_\gamma$.

4 Strong barbed bisimilarity

The semantics of the π-calculus and CCS can be expressed by associating a labelled or a barbed transition system with these languages, with processes as states. Semantic equivalences are defined on the states of labelled or barbed transition systems, and thereby on π- and CCS processes.

Definition 1. A *labelled transition system* (LTS) is pair $(S, \rightarrow)$ with S a class (of *states*) and $\rightarrow\ \subseteq S \times A \times S$ a *transition relation*, for some suitable set of *actions* A.

I write $P \xrightarrow{\alpha} Q$ for $(P,\alpha,Q) \in \rightarrow$, $P \xrightarrow{\alpha}$ for $\exists Q.\ P \xrightarrow{\alpha} Q$, and $P \stackrel{\alpha}{\not\rightarrow}$ for its negation. The structural operational semantics of CCS presented before creates an LTS with as states all CCS processes and the transition relation derived from the operational rules, with $A := Act$.

Definition 2. A *strong bisimulation* is a symmetric relation $\mathscr{R}$ on the states of an LTS such that

– if $P \mathrel{\mathscr{R}} Q$ and $P \xrightarrow{\alpha} P'$ then $\exists Q'.\ Q \xrightarrow{\alpha} Q' \wedge P' \mathrel{\mathscr{R}} Q'$.

Processes P and Q are *strongly bisimilar*—notation $P \leftrightarroweq Q$—if $P \mathrel{\mathscr{R}} Q$ for some strong bisimulation $\mathscr{R}$.

As is well-known, $\leftrightarroweq$ is an equivalence relation, and a strong bisimulation itself. Through the operational semantics of CCS$_\gamma$, strong bisimilarity is defined on CCS$_\gamma$ processes.

Definition 3. A *barbed transition system* (BTS) is a triple $(S, \mapsto, \downarrow)$ with S a class (of *states*), $\mapsto\ \subseteq S \times S$ a *reduction relation*, and $\downarrow\ \subseteq S \times B$ an *observability predicate* for some suitable set of *barbs* B.

[3] Renaming into τ could already be done in CCS by means of parallel composition. Hence this feature in itself does not add extra expressiveness.

Table 2. The actions

α	Kind	$O(\alpha)$	$\mathrm{fn}(\alpha)$	$\mathrm{bn}(\alpha)$
$M\tau$	Silent	$-$	$\emptyset$	$\emptyset$
$M\bar{x}y$	Free output	$\bar{x}$	$\mathrm{n}(M) \cup \{x, y\}$	$\emptyset$
$M\bar{x}(y)$	Bound output	$\bar{x}$	$\mathrm{n}(M) \cup \{x\}$	$\{y\}$
Mxy	Free input	x	$\mathrm{n}(M) \cup \{x, y\}$	$\emptyset$
$Mx(y)$	Bound input	x	$\mathrm{n}(M) \cup \{x\}$	$\{y\}$

One writes $P{\downarrow_b}$ for $P \in S$ and $b \in B$ when $(P, b) \in {\downarrow}$. A BTS can be extracted from an LTS with $\tau \in A$, by means of a partial observation function $O\colon A \rightharpoonup B$. The states remain the same, the reductions are taken to be the transitions labelled τ (dropping the label in the BTS), and $P{\downarrow_b}$ holds exactly when there is a transition $P \xrightarrow{\alpha} Q$ with $O(\alpha) = b$.

In this paper I consider labelled transition systems whose actions $\alpha \in A$ are of the forms presented in Table 2. Here x and y are *names*, drawn from the disjoint union of two sets $\mathcal{Z}$ and $\mathcal{R}$ of *public* and *private* names, and M is a (possibly empty) *matching sequence*, a sequence of *matches* $[x{=}y]$ with $x, y \in \mathcal{Z} \uplus \mathcal{R}$ and $x \neq y$. The set of names occurring in M is denoted $\mathrm{n}(M)$. In Table 2, also the *free names* $\mathrm{fn}(\alpha)$ and *bound names* $\mathrm{bn}(\alpha)$ of an action α are defined. The set of *names* of α is $\mathrm{n}(\alpha) := \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$. Consequently, also the actions *Act* of my instantiation of CCS$_\gamma$ need to have the forms of Table 2. For the translation into barbed transition systems I take $B := \mathcal{Z} \cup \overline{\mathcal{Z}}$, where $\overline{\mathcal{Z}} := \{\bar{a} \mid a \in \mathcal{Z}\}$, and $O(\alpha)$ as indicated in Table 2, provided $M = \varepsilon$ and $O(\alpha) \in B$.

Definition 4. A *strong barbed bisimulation* is a symmetric relation $\mathscr{R}$ on the states of a BTS such that

- if $P \mathrel{\mathscr{R}} Q$ and $P \longmapsto P'$ then $\exists Q'.\ Q \longmapsto Q' \wedge P' \mathrel{\mathscr{R}} Q'$
- and if $P \mathrel{\mathscr{R}} Q$ and $P{\downarrow_b}$ then also $Q{\downarrow_b}$.

Processes P and Q are *strongly barbed bisimilar*—notation $P \overset{\bullet}{\sim} Q$—if $P \mathrel{\mathscr{R}} Q$ for some strong barbed bisimulation $\mathscr{R}$.

Again, $\overset{\bullet}{\sim}$ is an equivalence relation, and a strong barbed bisimulation itself. Through the above definition, strong barbed bisimilarity is defined on all LTSs occurring in this paper, as well as on my instantiation of CCS$_\gamma$. It can also be used to compare processes from different LTSs, namely by taking their disjoint union.

5 The π-calculus

The π-calculus [23,24] is parametrised with an infinite set $\mathcal{N}$ of *names* and, for each $n \in \mathbb{N}$, a set of $\mathcal{K}_n$ of *agent identifiers* of arity n. The set T_π of π-calculus *terms*, *expressions*, *processes* or *agents* is the smallest set including:

$\mathbf{0}$		*inaction*
$\tau.P$	for $P \in \mathrm{T}_\pi$	*silent prefix*
$\bar{x}y.P$	for $x, y \in \mathcal{N}$ and $P \in \mathrm{T}_\pi$	*output prefix*
$x(y).P$	for $x, y \in \mathcal{N}$ and $P \in \mathrm{T}_\pi$	*input prefix*
$(\nu y)P$	for $y \in \mathcal{N}$ and $P \in \mathrm{T}_\pi$	*restriction*
$[x{=}y]P$	for $x, y \in \mathcal{N}$ and $P \in \mathrm{T}_\pi$	*match*
$P\|Q$	for $P, Q \in \mathrm{T}_\pi$	*parallel composition*
$P + Q$	for $P, Q \in \mathrm{T}_\pi$	*choice*
$A(y_1, ..., y_n)$	for $A \in \mathcal{K}_n$ and $y_i \in \mathcal{N}$	*defined agent*

The order of precedence among the operators is the order of the listing above. A process $\alpha.\mathbf{0}$ with $\alpha = \tau$ or $\bar{x}y$ or $x(y)$ is often written α.

$\mathrm{n}(P)$ denotes the set of all names occurring in a process P. An occurrence of a name y in a term is *bound* if it occurs in a subterm of the form $x(y).P$ or $(\nu y)P$; otherwise it is *free*. The set of names occurring free (resp. bound) in a process P is denoted $\mathrm{fn}(P)$ (resp. $\mathrm{bn}(P)$).

Each agent identifier $A \in \mathcal{K}_n$ is assumed to come with a unique *defining equation* of the form

$$A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$$

where the names x_i are all distinct and $\mathrm{fn}(P) \subseteq \{x_1, \ldots, x_n\}$.

The π-calculus with implicit matching (π_{IM}) drops the matching operator, instead allowing prefixes of the form $M\bar{x}y.P$, $Mx(y).P$ and $M\tau.P$, with M a matching sequence.

A *substitution* is a partial function $\sigma{:}\mathcal{N} \rightharpoonup \mathcal{N}$ such that $\mathcal{N}\backslash(\mathit{dom}(\sigma) \cup \mathit{range}(\sigma))$ is infinite. For $\vec{x} = (x_1, \ldots, x_n)$, $\vec{y} = (y_1, \ldots, y_n) \in \mathcal{N}^n$, $\{\vec{y}/\vec{x}\}$ denotes the substitution given by $\sigma(x_i) = y_i$ for $1 \leq i \leq n$. One writes $\{y/x\}$ when $n{=}1$.

For $x \in \mathcal{N}$, $x[\sigma]$ denotes $\sigma(x)$ if $x \in \mathit{dom}(\sigma)$ and x otherwise; $M[\sigma]$ is the result of changing each occurrence of a name x in M into $x[\sigma]$, while dropping resulting matches $[y{=}y]$.

For a substitution σ, the process $P\sigma$ is obtained from $P \in \mathrm{T}_\pi$ by simultaneous substitution, for all $x \in \mathit{dom}(\sigma)$, of $x[\sigma]$ for all free occurrences of x in P, with change of bound names to avoid name capture. A formal inductive definition is:

$$\begin{array}{rcl}
\mathbf{0}\sigma &=& \mathbf{0} \\
(M\tau.P)\sigma &=& M[\sigma]\tau.(P\sigma) \\
(M\bar{x}y.P)\sigma &=& M[\sigma]\overline{x[\sigma]}y[\sigma].(P\sigma) \\
(Mx(y).P)\sigma &=& M[\sigma]x[\sigma](z).(P\{z/y\}\sigma) \\
((\nu y)P)\sigma &=& (\nu z)(P\{z/y\}\sigma) \\
([x{=}y]P)\sigma &=& [x[\sigma]{=}y[\sigma]](P\sigma) \\
(P|Q)\sigma &=& (P\sigma)|(Q\sigma) \\
(P+Q)\sigma &=& (P\sigma)+(Q\sigma) \\
A(\vec{y})\sigma &=& A(\vec{y}[\sigma])
\end{array}$$

where z is chosen outside $\mathrm{fn}((\nu y)P) \cup \mathit{dom}(\sigma) \cup \mathit{range}(\sigma)$; in case $y \notin \mathit{dom}(\sigma) \cup \mathit{range}(\sigma)$ one always picks $z := y$.

A *congruence* is an equivalence relation $\sim$ on T_π such that $P \sim Q$ implies $\tau.P \sim \tau.Q$, $\bar{x}y.P \sim \bar{x}y.Q$, $x(y).P \sim x(y).Q$, $(\nu y)P \sim (\nu y)Q$, $[x{=}y]P \sim [x{=}y]Q$,

$P|U \sim Q|U$, $U|P \sim U|Q$, $P+U \sim Q+U$ and $U+P \sim U+Q$. Let $\equiv$ be the smallest congruence on T_π allowing renaming of bound names, i.e., that satisfies $x(y).P \equiv x(z).(P\{^z\!/\!_y\})$ and $(\nu y)P \equiv (\nu z)(P\{^z\!/\!_y\})$ for any $z \notin \mathrm{fn}((\nu y)P)$. If $P \equiv Q$, then Q is obtained from P by means of *α-conversion*. Due to the choice of z above, substitution is precisely defined only up to α-conversion.

Note that $P \equiv Q$ implies that $\mathrm{fn}(P) = \mathrm{fn}(Q)$, and also that $P\sigma \equiv Q\sigma$ for any substitution σ.

6 The semantics of the π-calculus

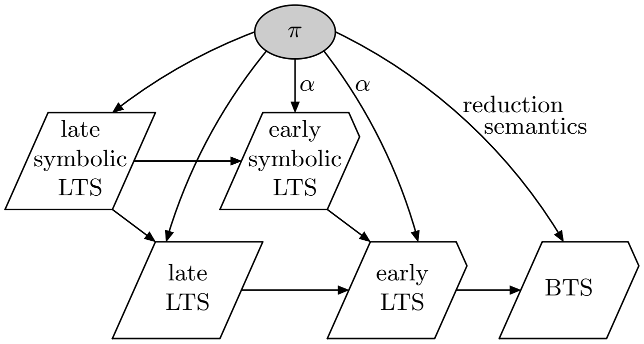


Fig. 1. Semantics of the π-calculus

Whereas CCS has only one operational semantics, the π-calculus is equipped with at least five, as indicated in Figure 1. The *late* operational semantics stems from [24], the origin of the π-calculus. It is given by the action rules of Table 3. These rules generate a labelled transition system in which the states are the π-calculus processes and the transitions are labelled with the actions τ, $\bar{x}y$, $\bar{x}(y)$ and $x(y)$ of Table 2 (always with M the empty string). Here I take $\mathscr{Z} := \mathscr{N}$ and $\mathcal{R} := \emptyset$. For π_{IM}, rule **match** is omitted. A process $[x{=}y]\alpha.P$ has no outgoing transitions, similar to $\mathbf{0}$.

In [24] the *late* and *early* bisimulation semantics of the π-calculus were proposed.

Definition 5. A *late bisimulation* is a symmetric relation $\mathscr{R}$ on π-processes such that, whenever $P \mathrel{\mathscr{R}} Q$, α is either τ or $\bar{x}y$ and $z \notin \mathrm{n}(P) \cup \mathrm{n}(Q)$,

1. if $P \xrightarrow{\alpha} P'$ then $\exists Q'$ with $Q \xrightarrow{\alpha} Q'$ and $P' \mathrel{\mathscr{R}} Q'$,
2. if $P \xrightarrow{x(z)} P'$ then $\exists Q' \forall y.\ Q \xrightarrow{x(z)} Q' \wedge P'\{y/z\} \mathrel{\mathscr{R}} Q'\{y/z\}$,
3. if $P \xrightarrow{\bar{x}(z)} P'$ then $\exists Q'$ with $Q \xrightarrow{\bar{x}(z)} Q'$ and $P' \mathrel{\mathscr{R}} Q'$.

Processes P and Q are *late bisimilar*—notation $P \mathrel{\dot{\sim}_L} Q$—if $P \mathrel{\mathscr{R}} Q$ for some late bisimulation $\mathscr{R}$. They are *late congruent*—notation $P {\sim_L} Q$—if $P\{\vec{y}/\vec{x}\} \mathrel{\dot{\sim}_L} Q\{\vec{y}/\vec{x}\}$ for any substitution $\{\vec{y}/\vec{x}\}$.

Table 3. Late structural operational semantics of the π-calculus

tau: $\tau.P \xrightarrow{\tau} P$	**output**: $\bar{x}y.P \xrightarrow{\bar{x}y} P$	**input**: $x(y).P \xrightarrow{x(z)} P\{z/y\}\quad (z \notin \text{fn}((\nu y)P))$
sum: $\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	**match**: $\dfrac{P \xrightarrow{\alpha} P'}{[x{=}x]P \xrightarrow{\alpha} P'}$	**ide**: $\dfrac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'}\ (A(\vec{x}) \overset{\text{def}}{=} P)$
par: $\dfrac{P \xrightarrow{\alpha} P'}{P\vert Q \xrightarrow{\alpha} P'\vert Q}\ \begin{pmatrix}\text{bn}(\alpha)\cap\\ \text{fn}(Q)=\emptyset\end{pmatrix}$	**com**: $\dfrac{P \xrightarrow{\bar{x}y} P',\ Q \xrightarrow{x(z)} Q'}{P\vert Q \xrightarrow{\tau} P'\vert Q'\{y/z\}}$	**close**: $\dfrac{P \xrightarrow{\bar{x}(z)} P',\ Q \xrightarrow{x(z)} Q'}{P\vert Q \xrightarrow{\tau} (\nu z)(P'\vert Q')}$
res: $\dfrac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}\ (y \notin \text{n}(\alpha))$	**alpha-open**: $\dfrac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(z)} P'\{z/y\}}\ \begin{pmatrix}y \neq x\\ z \notin \text{fn}((\nu y)P')\end{pmatrix}$	

The rules **sum**, **par**, **com** and **close** additionally have symmetric forms, with the rôles of P and Q exchanged.

Early bisimilarity ($\dot{\sim}_E$) and congruence ($\sim_E$) are defined likewise, but with $\forall y \exists Q'$ instead of $\exists Q' \forall y$. In [24,33] it is shown that $\dot{\sim}_L$ and $\dot{\sim}_E$ are congruences for all operators of the π-calculus, except for the input prefix. $\sim_E$ and $\sim_L$ are congruence relations for the entire language; in fact they are the congruence closures of $\dot{\sim}_L$ and $\dot{\sim}_E$, respectively. By definition, $\dot{\sim}_L \subseteq \dot{\sim}_E$, and thus $\sim_L \subseteq \sim_E$.

Lemma 1 ([24]). Let $P \equiv Q$ and $\text{bn}(\alpha) \cap \text{n}(Q) = \emptyset$.
If $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\alpha} Q'$ for some Q' with $P' \equiv Q'$.

This implies that $\equiv$ is a late bisimulation, so that $\equiv\ \subset\ \sim_L$.

In [25] the *early* operational semantics of the π-calculus is proposed, presented in Table 4; it uses free input actions xy instead of bound inputs $x(y)$. This is also the semantics of [33]. The semantics in [25,33] requires us to identify processes modulo α-conversion before applying the operational rules. This is equivalent to adding rule **alpha** of Table 4.

A variant of the late operational semantics incorporating rule **alpha** is also possible. In this setting rule **alpha-open** can be simplified to **open**, and likewise **input** to $x(y).P \xrightarrow{x(y)} P$. By Lemma 1, the late operational semantics with **alpha** gives rise to the same notions of early and late bisimilarity as the late operational semantics without **alpha**; the addition of this rule is entirely optional. Interestingly, the rule **alpha** is not optional in the early operational semantics, not even when reinstating **alpha-open**.

Example 1. Let $P := \bar{x}y|(\nu y)(x(z))$. One has $(\nu y)(x(z)) \xrightarrow{x(z)}_L (\nu y)\mathbf{0}$ and thus $P \xrightarrow{\tau}_L \mathbf{0}|(\nu y)\mathbf{0}$ by **com**. However, $(\nu y)(x(z)) \xrightarrow{xy}_E (\nu y)\mathbf{0}$ is forbidden by the side condition of **res**, so in the early semantics without **alpha** P cannot make a τ-step. Rule **alpha** comes to the rescue here, as it allows $P \equiv \bar{x}y|(\nu w)(x(z)) \xrightarrow{\tau}_E \mathbf{0}|(\nu w)\mathbf{0}$.

Table 4. Early structural operational semantics of the π-calculus

tau: $\tau.P \xrightarrow{\tau} P$	**output**: $\bar{x}y.P \xrightarrow{\bar{x}y} P$	**early-input**: $x(y).P \xrightarrow{xz} P\{z/y\}$
sum: $\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	**match**: $\dfrac{P \xrightarrow{\alpha} P'}{[x{=}x]P \xrightarrow{\alpha} P'}$	**ide**: $\dfrac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'}$ $(A(\vec{x}) \overset{\text{def}}{=} P)$
par: $\dfrac{P \xrightarrow{\alpha} P'}{P\vert Q \xrightarrow{\alpha} P'\vert Q}$ $\left(\begin{matrix}\text{bn}(\alpha)\cap\\ \text{fn}(Q)=\emptyset\end{matrix}\right)$	**early-com**: $\dfrac{P \xrightarrow{\bar{x}y} P',\ Q \xrightarrow{xy} Q'}{P\vert Q \xrightarrow{\tau} P'\vert Q'}$	**early-close**: $\dfrac{P \xrightarrow{\bar{x}(z)} P',\ Q \xrightarrow{xz} Q'}{P\vert Q \xrightarrow{\tau} (\nu z)(P'\vert Q')}$ $\left(\begin{matrix}z \notin\\ \text{fn}(Q)\end{matrix}\right)$
res: $\dfrac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}$ $(y \notin \text{n}(\alpha))$	**open**: $\dfrac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(y)} P'}$ $(y \neq x)$	**alpha**: $\dfrac{P \equiv Q, \quad Q \xrightarrow{\alpha} Q'}{P \xrightarrow{\alpha} Q'}$

By the following lemma, the early transition relation $\longrightarrow_E$ is completely determined by the late transition relation $\longrightarrow_{\alpha L}$ with **alpha**:

Lemma 2 ([25]). Let $P \in \text{T}_\pi$ and β be τ, $\bar{x}y$ or $\bar{x}(y)$.

– $P \xrightarrow{\beta}_E Q$ iff $P \xrightarrow{\beta}_{\alpha L} Q$.
– $P \xrightarrow{xy}_E Q$ iff $P \xrightarrow{x(z)}_{\alpha L} R$ for some R, z with $Q \equiv R\{y/z\}$.

The early transition relations allow a more concise definition of early bisimilarity:

Proposition 1 ([25]). An *early bisimulation* is a symmetric relation $\mathscr{R}$ on T_π such that, whenever $P \mathrel{\mathscr{R}} Q$ and α is an action with $\text{bn}(\alpha) \cap (\text{n}(P) \cup \text{n}(Q)) = \emptyset$,

– if $P \xrightarrow{\alpha}_E P'$ then $\exists Q'$ with $Q \xrightarrow{\alpha}_E Q'$ and $P' \mathrel{\mathscr{R}} Q'$.

Processes P and Q are early bisimilar iff $P \mathrel{\mathscr{R}} Q$ for some early bisimulation $\mathscr{R}$.

Through the general method of Section 4, taking $\mathcal{Z} := \mathcal{N}$ and $\mathcal{R} := \emptyset$, a barbed transition system can be extracted from the late or early labelled transition system of the π-calculus; by Lemmas 1 and 2 the same BTS is obtained either way. This defines strong barbed bisimilarity $\overset{\bullet}{\sim}$ on T_π. The congruence closure of $\overset{\bullet}{\sim}$ is early congruence [33]. In [21] a *reduction semantics* of the π-calculus is given, that yields a BTS right away. Up to strong barbed bisimilarity, this BTS is the same as the one extracted from the late or early LTS.

In [32] yet another operational semantics of the π-calculus was introduced, in a style called *symbolic* by Hennessy & Lin [16], who had proposed it for a version of value-passing CCS. It is presented in Table 5. The transitions are labelled with actions α of the form $M\beta$, where M is a matching sequence and β an action as in the late operational semantics. When $x \neq y$ the matching sequence M prepended with $[x{=}y]$ is denoted $[x{=}y]M$; however, $[x{=}x]M$ simply denotes M.

In the operational semantics of CCS, τ-actions can be thought of as reactions that actually take place, whereas a transition labelled a merely represents the

Table 5. Late symbolic structural operational semantics of the π-calculus

tau: $M\tau.P \xrightarrow{M\tau} P$

output: $M\bar{x}y.P \xrightarrow{M\bar{x}y} P$

input: $Mx(y).P \xrightarrow{Mx(z)} P\{z/y\}\ \ (z \notin \mathrm{fn}((\nu y)P))$

sum:
$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$$

symb-match:
$$\frac{P \xrightarrow{\alpha} P'}{[x{=}y]P \xrightarrow{[x=y]\alpha} P'}$$

ide:
$$\frac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'}\quad (A(\vec{x}) \overset{\mathrm{def}}{=} P)$$

par:
$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\quad \begin{pmatrix}\mathrm{bn}(\alpha)\cap\\ \mathrm{fn}(Q)=\emptyset\end{pmatrix}$$

symb-com:
$$\frac{P \xrightarrow{M\bar{x}y} P',\ Q \xrightarrow{Nv(z)} Q'}{P|Q \xrightarrow{[x=v]MN\tau} P'|Q'\{y/z\}}$$

symb-close:
$$\frac{P \xrightarrow{M\bar{x}(z)} P',\ Q \xrightarrow{Nv(z)} Q'}{P|Q \xrightarrow{[x=v]MN\tau} (\nu z)(P'|Q')}$$

res:
$$\frac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}\quad \begin{pmatrix}y \notin\\ \mathrm{n}(\alpha)\end{pmatrix}$$

symb-alpha-open:
$$\frac{P \xrightarrow{M\bar{x}y} P'}{(\nu y)P \xrightarrow{M\bar{x}(z)} P'\{z/y\}}\quad \begin{pmatrix}y \neq x\\ z \notin \mathrm{fn}((\nu y)P')\\ y \notin \mathrm{n}(M)\end{pmatrix}$$

For the π-calculus, the blue Ms are omitted; for π_{IM} the purple rules.

potential of a reaction with the environment, one that can take place only if the environment offers a complementary transition $\bar{a}$. In case the environment never does an $\bar{a}$, this potential will not be realised. A reduction semantics (as in [22]) yields a BTS that only represents directly the realised actions—the τ-transitions or *reductions*—and reasons about the potential reactions by defining the semantics of a system in terms of reductions that can happen when placing the system in various contexts. An LTS, on the other hand, directly represents transitions that could happen under some conditions only, annotated with the conditions that enable them. For CCS, this annotation is the label a, saying that the transition is conditional on an $\bar{a}$-signal from the environment. As a result of this, semantic equivalences defined on labelled transitions systems tend to be congruences for most operators right away, and do not need much closure under contexts.

Seen from this perspective, the operational semantics of the π-calculus of Table 3 or 4 is a compromise between a pure reduction semantics and a pure labelled transition system semantics. Input and output actions are explicitly included to signal potential reactions that are realised in the presence of a suitable communication partner, but actions whose occurrence is conditional on two different names x and y denoting the same channel are entirely omitted, even though any π-process can be placed in a context in which x and y will be identified. As a consequence of this, the early and late bisimilarities need to be closed under all possible substitutions or identifications of names before they turn into early and late congruences. The operational semantics of Table 5 adds the conditional transitions that where missing in Table 3, and hence can be seen as a true labelled transition system semantics.

In this paper I need the early symbolic operational semantics of the π-calculus, presented in Table 6. Although new, it is the logical combination of the early and the (late) symbolic semantics. Its transitions that are labelled

Table 6. Early symbolic structural operational semantics of the π-calculus

tau: $M\tau.P \xrightarrow{M\tau} P$	**output**: $M\bar{x}y.P \xrightarrow{M\bar{x}y} P$	**early-input**: $Mx(y).P \xrightarrow{Mxz} P\{^{z}\!/\!_{y}\}$
sum: $\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	**symb-match**: $\dfrac{P \xrightarrow{\alpha} P'}{[x{=}y]P \xrightarrow{[x=y]\alpha} P'}$	**ide**: $\dfrac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'} \quad (A(\vec{x}) \overset{\text{def}}{=} P)$
par: $\dfrac{P \xrightarrow{\alpha} P'}{P\mid Q \xrightarrow{\alpha} P'\mid Q} \begin{pmatrix}\text{bn}(\alpha)\\ \cap\, \text{fn}(Q)\\ =\emptyset\end{pmatrix}$	**e-s-com**: $\dfrac{P \xrightarrow{M\bar{x}y} P',\ Q \xrightarrow{Nvy} Q'}{P\mid Q \xrightarrow{[x=v]MN\tau} P'\mid Q'}$	**e-s-close**: $\dfrac{P \xrightarrow{M\bar{x}(z)} P',\ Q \xrightarrow{Nvz} Q'}{P\mid Q \xrightarrow{[x=v]MN\tau} (\nu z)(P'\mid Q')} \begin{pmatrix} z\notin \\ \text{fn}(Q)\end{pmatrix}$
res: $\dfrac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'} \begin{pmatrix} y\notin \\ \text{n}(\alpha)\end{pmatrix}$	**symb-open**: $\dfrac{P \xrightarrow{M\bar{x}y} P'}{(\nu y)P \xrightarrow{M\bar{x}(y)} P'} \begin{pmatrix} y\neq x\\ y\notin \text{n}(M)\end{pmatrix}$	**alpha**: $\dfrac{P\equiv Q,\quad Q \xrightarrow{\alpha} Q'}{P \xrightarrow{\alpha} Q'}$

with actions having an empty matching sequence are exactly the transitions of the early semantics, so the BTS extracted from this semantics is the same.

For π_{IM}, rule **symb-match** is omitted, but **tau**, **output** and **input** carry the matching sequence M (indicated in blue).

7 Valid translations

A *signature* Σ is a set of *operator symbols* g, each of which is equipped with an *arity* $n \in \mathbb{N}$. The set T_Σ of *closed terms* over Σ is the smallest set such that, for all $g \in \Sigma$,

$$P_1,\dots,P_n \in \mathrm{T}_\Sigma \quad \Rightarrow \quad g(P_1,\dots,P_n) \in \mathrm{T}_\Sigma\ .$$

Call a language *simple* if its expressions are the closed terms T_Σ over some signature Σ. The π-calculus is simple in this sense; its signature consists of the binary operators $+$ and $|$, the unary operators τ, $\bar{x}y.$, $x(y).$, (νy) and $[x{=}y]$ for $x,y \in \mathcal{N}$, and the nullary operators (or *constants*) $\mathbf{0}$ and $A(y_1,\dots,y_n)$ for $A \in \mathcal{K}_n$ and $y_i \in \mathcal{N}$. CCS is not quite simple, since it features the infinite choice operator.

Let $\mathcal{L}$ be a language. An n-ary $\mathcal{L}$-context C is an $\mathcal{L}$-expression that may contain special *variables* $X_1, ..., X_n$—its *holes*. For C an n-ary context, $C[P_1,\dots,P_n]$ is the result of substituting P_i for X_i, for each $i = 1,\dots,n$.

Definition 6. Let $\mathcal{L}'$ and $\mathcal{L}$ languages, generating sets of closed terms $\mathrm{T}_{\mathcal{L}'}$ and $\mathrm{T}_{\mathcal{L}}$. Let $\mathcal{L}'$ be simple, with signature Σ. A *translation* from $\mathcal{L}'$ to $\mathcal{L}$ (or an *encoding* from $\mathcal{L}'$ into $\mathcal{L}$) is a function $\mathscr{T} : \mathrm{T}_{\mathcal{L}'} \to \mathrm{T}_{\mathcal{L}}$. It is *compositional* if for each n-ary operator $g \in \Sigma$ there exists an n-ary $\mathcal{L}$-context C_g such that $\mathscr{T}(g(P_1,\dots,P_n)) = C_g[\mathscr{T}(P_1),\dots,\mathscr{T}(P_n)]$.

Let $\sim$ be an equivalence relation on $\mathrm{T}_{\mathcal{L}'} \cup \mathrm{T}_{\mathcal{L}}$. A translation $\mathscr{T}$ from $\mathcal{L}'$ to $\mathcal{L}$ is *valid up to* $\sim$ if it is compositional and $\mathscr{T}(P) \sim P$ for each $P \in \mathrm{T}_{\mathcal{L}'}$.

The above definition stems in essence from [10,11], but could be simplified here since [10,11] also covered the case that $\mathcal{L}'$ is not simple. Moreover, here I restrict attention to what are called *closed term languages* in [11].

8 The unencodability of π into CCS

In this section I show that there exists no translation of the π-calculus to CCS that is valid up to $\stackrel{\bullet}{\sim}$. I even show this for the fragment $\pi_A^\P$ of the (asynchronous) π-calculus without choice, recursion, matching and restriction (thus only featuring inaction, action prefixing and parallel composition).

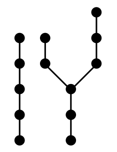
Definition 7. *Strong reduction bisimilarity,* $\leftrightarrow_r$, is defined just as strong barbed equivalence in Definition 4, but without the requirement on barbs.

I show that there is no translation of $\pi_A^\P$ to CCS that is valid up to $\leftrightarrow_r$. As $\leftrightarrow_r$ is coarser than $\stackrel{\bullet}{\sim}$, this implies my claim above. It may be useful to read this section in parallel with the first half of Section 14.

Definition 8. Let $\leftarrow\!\!\prec$ be the smallest preorder on CCS contexts such that $\sum_{i\in I} E_i \leftarrow\!\!\prec E_j$ for all $j \in I$, $E|F \leftarrow\!\!\prec E$, $E|F \leftarrow\!\!\prec F$, $E\backslash L \leftarrow\!\!\prec E$, $E[f] \leftarrow\!\!\prec E$ and $A \leftarrow\!\!\prec P$ for all $A \in \mathcal{K}$ with $A \stackrel{\text{def}}{=} P$. A variable X occurs *unguarded* in a context E if $E \leftarrow\!\!\prec X$.

If the hole X_1 occurs unguarded in the unary context $E[\ ]$ and $U \xrightarrow{\tau}$ (resp. $U \xrightarrow{\tau}\xrightarrow{\tau}$) then $E[U] \xrightarrow{\tau}$ (resp. $E[U] \xrightarrow{\tau}\xrightarrow{\tau}$).

Lemma 3. Let $E[\ ]$ be a unary and $C[\ ,\ ]$ a binary CCS context, and $P, Q, P', Q', U \in \mathrm{T}_{\mathrm{CCS}}$. If $E[C[P,Q]] \xrightarrow{\tau}$ and $U \xrightarrow{\tau}$ but neither $E[C[P',Q]] \xrightarrow{\tau}$ nor $E[C[P,Q']] \xrightarrow{\tau}$ nor $E[U] \xrightarrow{\tau}\xrightarrow{\tau}$, then $C[P,Q] \xrightarrow{\tau}$.


CCS proof trees

Proof. Since the only rule in the operational semantics of CCS with multiple premises has a conclusion labelled τ, it can occur at most once in the derivation of a CCS transition. Thus, such a derivation is a tree with at most two branches, as illustrated at the right. Now consider the derivation of $E[C[P,Q]] \xrightarrow{\tau}$. If none of its branches prods into the subprocess P, the transition would be independent on what is substituted here, thus yielding $E[C[P',Q]] \xrightarrow{\tau}$. Thus, by symmetry, both P and Q are visited by branches of this proof. It suffices to show that these branches come together within the context C, as this implies $C[P,Q] \xrightarrow{\tau}$. So suppose, towards a contradiction, that the two branches come together in E. Then E must have the form $E_1[E_2[\ ]|E_3[\ ]]$, where the hole X_1 occurs unguarded in E_2, E_3 as well as E_1. But in that case $E[U] \xrightarrow{\tau}\xrightarrow{\tau}$, contradicting the assumptions. $\square$

Lemma 4. If $D[\ ,\ ,\ ]$ is a ternary CCS context, $P_1, P_2, P_3 \in \mathrm{T}_{\mathrm{CCS}}$, and $D[P_1,P_2,P_3] \xrightarrow{\tau}$, then there exists an $i \in \{1,2,3\}$ and a CCS context $E[\ ]$ such that $D'[P] \xrightarrow{\tau} E[P]$ for any $P \in \mathrm{T}_{\mathrm{CCS}}$. Here D' is the unary context obtained from $D[\ ,\ ,\ ]$ by substituting P_j for the hole X_j, for all $j \in \{1,2,3\}$, $j \neq i$.

Proof. Since the derivation of $D[P_1, P_2, P_3] \xrightarrow{\tau}$ has at most two branches, one of the P_i is not involved in this proof at all. Thus, the derivation remains valid if any other process P is substituted in the place of that P_i; the target of the transition remains the same, except for P taking the place of P_i in it. □

Theorem 1. There is no translation from $\pi_A^{\P}$ to CCS that is valid up to $\leftrightarroweq_r$.

Proof. Suppose, towards a contradiction, that $\mathscr{T}$ is a translation from $\pi_A^{\P}$ to CCS that is valid up to $\leftrightarroweq_r$. By definition, this means that $\mathscr{T}$ is compositional and that $\mathscr{T}(P) \leftrightarroweq_r P$ for any $\pi_A^{\P}$-process P.

As $\mathscr{T}$ is compositional, there exists a ternary CCS context $D[\ ,\ ,\ ]$ such that, for any $\pi_A^{\P}$-processes R, S, T,

$$\mathscr{T}\big(\bar{x}v \mid x(y).(R|S|T)\big) = D[\mathscr{T}(R), \mathscr{T}(S), \mathscr{T}(T)].$$

Since $\bar{x}v\big|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0}) \xrightarrow{\tau}$ as well as $\mathscr{T}\big(\bar{x}v\big|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})\big) \leftrightarroweq_r \bar{x}v\big|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})$, it follows that $\mathscr{T}\big(\bar{x}v\big|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}$, i.e., $D[\mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0})] \xrightarrow{\tau}$. Hence Lemma 4 can be applied. For simplicity I assume that $i = 1$; the other two cases proceed in the same way. So there is a CCS context $E[\ ]$ such that $D[P, \mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0})] \xrightarrow{\tau} E[P]$ for all CCS terms P. In particular, for all $\pi_A^{\P}$-terms R,
$\mathscr{T}\big((\bar{x}v|x(y).(R|\mathbf{0}|\mathbf{0})\big) = D[\mathscr{T}(R), \mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0})] \xrightarrow{\tau} E[\mathscr{T}(R)].$ (1)

I examine the translations of the π-calculus expressions $\bar{x}v\big|x(y).(R|\mathbf{0}|\mathbf{0})$, for $R \in \{\bar{y}z|v(w), 0|v(w), \bar{y}z|0, \tau\}$.

Since $\bar{x}v\big|x(y).(\bar{y}z|v(w)|\mathbf{0}|\mathbf{0}) \xrightarrow{\tau}\xrightarrow{\tau}$ and $\mathscr{T}$ respects $\leftrightarroweq_r$,

$$\mathscr{T}\big(\bar{x}v\big|x(y).(\bar{y}z|v(w)|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}\xrightarrow{\tau}.$$

In the same way, neither $\mathscr{T}\big(\bar{x}v\big|x(y).(0|v(w)|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}\xrightarrow{\tau}$
nor $\mathscr{T}\big(\bar{x}v\big|x(y).(\bar{y}z|0|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}\xrightarrow{\tau}$. (2)
Furthermore, since $\mathscr{T}$ respects $\leftrightarroweq_r$ and there is no $S \in \mathrm{T}_\pi$ such that

$$\bar{x}v|x(y).(\bar{y}z|v(w)|\mathbf{0}|\mathbf{0}) \xrightarrow{\tau} S \stackrel{\tau}{\not\longrightarrow},$$

there is no $S \in \mathrm{T}_{\mathrm{CCS}}$ with $\mathscr{T}\big(\bar{x}v|x(y).(\bar{y}z|v(w)|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau} S \stackrel{\tau}{\not\longrightarrow}$. (3)

By (1) and (3), $E[\mathscr{T}(\bar{y}z|v(w))] \xrightarrow{\tau}$.

By (1) and (2), $E[\mathscr{T}(0|v(w))] \stackrel{\tau}{\not\longrightarrow}$ and $E[\mathscr{T}(\bar{y}z|0)] \stackrel{\tau}{\not\longrightarrow}$.

Since $\mathscr{T}$ is compositional, there is a binary CCS context $C_|[\ ,\ ]$ such that $\mathscr{T}(P|Q) = C_|[\mathscr{T}(P), \mathscr{T}(Q)]$ for any $P, Q \in \mathrm{T}_\pi$. It follows that

$$\begin{aligned}&E[C_|[\mathscr{T}(\bar{y}z), \mathscr{T}(v(w))]] \xrightarrow{\tau}\\ &E[C_|[\mathscr{T}(\mathbf{0}), \mathscr{T}(v(w))]] \stackrel{\tau}{\not\longrightarrow}\\ &E[C_|[\mathscr{T}(\bar{y}z), \mathscr{T}(\mathbf{0})]] \stackrel{\tau}{\not\longrightarrow}.\end{aligned}$$

Moreover since $\tau \xrightarrow{\tau}$, also $U := \mathscr{T}(\tau) \xrightarrow{\tau}$, but, since it is not the case that $\bar{x}v\big|x(y).(\tau|\mathbf{0}|\mathbf{0}) \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}$, neither holds $\mathscr{T}\big(\bar{x}v\big|x(y).(\tau|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}$, and neither $E[U] \xrightarrow{\tau}\xrightarrow{\tau}$. So by Lemma 3, $\mathscr{T}(\bar{y}z|v(w)) = C_|[\mathscr{T}(\bar{y}z), \mathscr{T}(v(w))] \xrightarrow{\tau}$, yet $\bar{y}z|v(w) \stackrel{\tau}{\not\longrightarrow}$. This contradicts the validity of $\mathscr{T}$ up to $\leftrightarroweq_r$. □

9 A valid translation of π_{IM} into CCS$_\gamma$

Given a set $\mathcal{N}$ of names, I now define the parameters $\mathcal{K}$, $\mathscr{A}$ and γ of the language CCS$_\gamma$ that will be the target of my encoding. First of all, $\mathcal{K}$ will be the disjoint union of all the sets $\mathcal{K}_n$ for $n \in \mathbb{N}$, of n-ary agent identifiers from the chosen instance of the π-calculus.

Take $p \notin \mathcal{N}$. Let $\mathcal{R}_0 := \{{}^\varsigma p \mid \varsigma \in \{e,\ell,r\}^*\}$. The set $\mathcal{R}$ of *private names* is $\{u^\upsilon \mid u \in \mathcal{R}_0 \wedge \upsilon \in \{'\}^*\}$. Let $\mathcal{S} = \{s_1, s_2, \ldots\}$ be an infinite set of *spare names*, disjoint from $\mathcal{N}$ and $\mathcal{R}$. Let $\mathcal{Z} := \mathcal{N} \uplus \mathcal{S}$ and $\mathcal{H} := \mathcal{Z} \uplus \mathcal{R}$.[4]

I take *Act* to be the set of all expressions α from Table 2, as defined in Section 4 (in terms of $\mathcal{Z}$ and $\mathcal{R}$), so $\mathscr{A} := \mathit{Act}\backslash\{\tau\}$. The communication function γ is given by $\gamma(M\bar{x}y, Nvy) = [x{=}v]MN\tau$, just as for rule **e-s-com** in Table 6.

For $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{N}^n$ and $\vec{y} = (y_1, \ldots, y_n) \in \mathcal{H}^n$, with the x_i distinct, let $\{\vec{y}/\vec{x}\}^\mathcal{S} : \mathcal{S} \cup \{x_1, \ldots, x_n\} \rightharpoonup \mathcal{H}$ be the substitution σ with $\sigma(x_i) = y_i$ and $\sigma(s_i) = x_i$ for $i = 1, ..., n$, and $\sigma(s_i) = s_{i-n}$ for $i > n$. These functions extend homomorphically to $\mathscr{A}$ and thereby constitute CCS$_\gamma$ relabellings. Abbreviate $[\{\vec{y}/\vec{x}\}^\mathcal{S}]$ by $[\vec{y}/\vec{x}]$ and $[\{z/y\}^\mathcal{S}]$ by $[z/y]$.

For $\eta \in \{\ell, r, e\}$ and $y \in \mathcal{Z}$, let the surjective substitutions $\eta : \mathcal{R} \rightharpoonup \mathcal{R}$ and $p_y : \{y\} \cup \mathcal{R} \to \{y\} \cup \mathcal{R}$ be given by:

$$\begin{array}{lcll} \eta({}^\varsigma p) & := & {}^{\eta\varsigma}p & \\ \eta({}^\varsigma p^{\upsilon\prime}) & := & {}^\varsigma p^\upsilon & \text{if } \varsigma \neq \eta\zeta \end{array} \qquad \begin{array}{lcll} p_y(y) & := & p & \\ p_y(p') & := & y & \\ p_y(u) & := & e(u) & \text{if } u \neq y, p'. \end{array}$$

These $\sigma : \mathcal{H} \rightharpoonup \mathcal{H}$ are injective, i.e., $x[\sigma] \neq y[\sigma]$ when $x \neq y$. Also they yield CCS$_\gamma$ relabellings. The following compositional encoding, which will be illustrated with examples in Section 12, defines my translation from π_{IM} to CCS$_\gamma$.

$$\begin{array}{lcll}
\mathscr{T}(\mathbf{0}) & := & \mathbf{0} & \\
\mathscr{T}(M\tau.P) & := & M\tau.\mathscr{T}(P) & \\
\mathscr{T}(M\bar{x}y.P) & := & M\bar{x}y.\mathscr{T}(P) & \\
\mathscr{T}(Mx(y).P) & := & \sum_{z\in\mathcal{H}} Mxz.\big(\mathscr{T}(P)[z/y]\big) & \\
\mathscr{T}((\nu y)P) & := & \mathscr{T}(P)[p_y] & \\
\mathscr{T}(P \mid Q) & := & \mathscr{T}(P)[\ell] \parallel \mathscr{T}(Q)[r] & \\
\mathscr{T}(P + Q) & := & \mathscr{T}(P) + \mathscr{T}(Q) & \\
\mathscr{T}(A(\vec{y})) & := & A[\vec{y}/\vec{x}] & \text{when } A(\vec{x}) \stackrel{\text{def}}{=} P
\end{array}$$

where the CCS$_\gamma$ agent identifier A has the defining equation $A = \mathscr{T}(P)$ when $A(\vec{x}) \stackrel{\text{def}}{=} P$ was the defining equation of the agent identifier A from the π-calculus.

To explain what this encoding does, inaction, silent prefix, output prefix and choice are translated homomorphically. The input prefix is translated into an infinite sum over all possible input values z that could be received, of the received message Mxz followed by the continuation process $\mathscr{T}(P)[z/y]$. Here $[z/y]$ is a CCS relabelling operator that simulates substitution of z for y in $\mathscr{T}(P)$. This

[4] The names in $\mathcal{S}$ and in $\mathcal{R}\backslash\mathcal{R}_0$ exist solely to make the substitutions $\{\vec{y}/\vec{x}\}^\mathcal{S}$, η and p_y surjective. Here σ is surjective iff $\mathit{dom}(\sigma) \subseteq \mathit{range}(\sigma)$.

implements the rule **early-input** from Table 6. Agent identifiers are also translated homomorphically, except that their arguments $\vec{y}$ are replaced by relabelling operators.

Restriction is translated by simply dropping the restriction operator, but renaming the restricted name y into a private name p that generates no barbs. The operator $[p_y]$ injectively renames all private names ${}^{\varsigma}p$ that occur in the scope of (νy) by tagging all of them with a tag e. This ensures that the new private name p is fresh, so that no name clashes can occur that in π_{IM} would have been prevented by the restriction operator.

Parallel composition is almost translated homomorphically. However, each private name on the right is tagged with an r, and on the left with an ℓ. This guarantees that private names introduced at different sides of a parallel composition cannot interact. Interaction is only possible when the name is passed on in the appropriate way.

The main result of this paper states the validity of the above translation, and thus that CCS_γ is at least as expressive as π_{IM}:

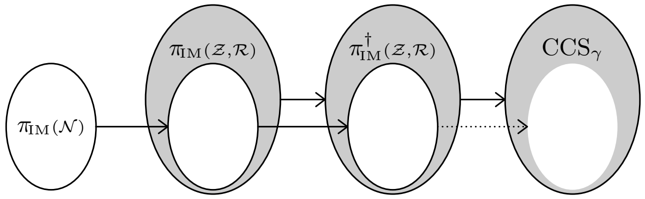
Theorem 2. *For $P \in \mathrm{T}_\pi$ one has $\mathscr{T}(P) \overset{\bullet}{\sim} P$.*

See http://theory.stanford.edu/~rvg/abstracts.html#153 for a proof.

Theorem 2 says that each π-calculus process is strongly barbed bisimilar to its translation as a CCS_γ process. The labelled transition systems of the π-calculus and CCS_γ are both of the type presented in Section 4, i.e. with transition labels taken from Table 2. There also the associated barbs are defined. By Theorem 2 each π transition $P \xrightarrow{\tau} P'$ can be matched by a CCS_γ transition $\mathscr{T}(P) \xrightarrow{\tau} Q$ with $\mathscr{T}(P') \overset{\bullet}{\sim} Q$. Likewise, each CCS_γ transition $\mathscr{T}(P) \xrightarrow{\tau} Q$ can be matched by a π transition $P \xrightarrow{\tau} P'$ with $\mathscr{T}(P') \overset{\bullet}{\sim} Q$. Moreover, if P has a barb x (or $\bar{x}$) then so does $\mathscr{T}(P)$, and vice versa. Here a π or CCS_γ process P has a barb $a \in \mathcal{Z} \cup \overline{\mathcal{Z}}$ iff $P \xrightarrow{ay} P'$ or $P \xrightarrow{a(y)} P'$ for some name $y \in \mathcal{H}$ and process P'. Transitions $P \xrightarrow{M\bar{x}y} P'$, $P \xrightarrow{M\bar{x}(y)} P'$, $P \xrightarrow{Mxy} P'$ or $P \xrightarrow{Mx(y)} P'$ with $M \neq \varepsilon$ or $x \in \mathcal{R}$ generate no barbs.

10 The ideas behind this encoding

The above encoding combines seven ideas, each of which appears to be necessary to achieve the desired result. Accordingly, the translation could be described as the composition of seven encodings, leading from π_{IM} to CCS_γ via six intermediate languages. Here a language comprises syntax as well as semantics. Each of the intermediate languages has a labelled transition system semantics where the labels are as described in Section 4. Accordingly, at each step it is well-defined whether strong barbed bisimilarity is preserved, and one can show it is. These proofs go by induction on the derivation of transitions, where the transitions with visible labels are necessary steps even when one would only be interested in the transitions with τ-labels. There are various orders in which the seven steps can be taken. The seven steps are:



$\pi_{\rm IM}(\mathcal{N})$
$\pi_{\rm IM}(\mathcal{Z},\mathcal{R})$
$\pi^\dagger_{\rm IM}(\mathcal{Z},\mathcal{R})$
CCS_γ

Fig. 2. Translation from the π-calculus with implicit matching to CCS_γ
Definitions of the intermediate languages $\pi_{\rm IM}(\mathcal{Z},\mathcal{R})$ and $\pi^\dagger_{\rm IM}(\mathcal{Z},\mathcal{R})$ are not provided here.

1. Moving from the late operational semantics (Table 3) to the early one (Table 4). This translation is syntactically the identity function, but still its validity requires proof, as the generated LTS changes. The proof amounts to showing that the same barbed transition system is obtained before and after the translation—see Section 6.
2. Moving from a regular operational semantics (Table 4) to a symbolic one (Table 6). This step commutes with the previous one.
3. Renaming the bound names of a process in such a way that the result is clash-free [3], meaning that all bound names are different and no name occurs both free and bound. The trick is to do this in a compositional way. The relabelling operators $[\ell]$, $[r]$ and $[p_y]$ in the final encoding stem from this step.
4. Eliminating the need for rule **alpha** in the operational semantics. This works only for clash-free processes, as generated by the previous step.
5. Dropping the restriction operators, while preserving strong barbed bisimilarity. This eliminates the orange parts of Table 6. For this purpose clash-freedom and the elimination of **alpha** are necessary.
6. Changing all occurrences of substitutions into applications of CCS relabelling operators.
7. The previous six steps generate a language with a semantics in the De Simone format. So from here on a translation to MEIJE or aprACP_R is known to be possible. The last step, to CCS_γ, involves changing the remaining form of name-binding into an infinite sum.

As indicated in Figure 2, my translation maps the π-calculus with implicit matching to a subset of CCS_γ. On that subset, π-calculus behaviour can be replayed faithfully, at least up to strong early congruence, the congruence closure of strong barbed bisimilarity (cf. [11]). However, the interaction between a translated π-calculus process and a CCS_γ process outside the image of the translation may be disturbing, and devoid of good properties. Also, in case intermediate languages are encountered on the way from $\pi_{\rm IM}$ to CCS_γ, which is just one of the ways to prove my result, no guarantees are given on the sanity of those languages outside the image of the source language, i.e. on their behaviour outside the realm of clash-free processes after Step 3 has been made.

11 Triggering

To include the general matching operator in the source language I need to extend the target language with the *triggering* operator $s{\Rightarrow}P$ of Meije [1,34]:

$$\frac{P \xrightarrow{\alpha} P'}{s{\Rightarrow}P \xrightarrow{s\alpha} P'}$$

Meije features *signals* and *actions*; each signal s can be "applied" to an action α, and doing so yields an action $s\alpha$. In this paper the actions are as in Table 2, and a signal is an expression $[x{=}y]$ with $x, y \in \mathcal{N}$; application of a signal to an action was defined in Section 6.

Triggering cannot be expressed in CCS$_\gamma$, as rooted weak bisimilarity [2], the weak congruence of [19,20], is a congruence for CCS$_\gamma$ but not for triggering. However, rooted branching bisimilarity [12] is a congruence for triggering [9].

My translation from $\pi_{\rm IM}$ to CCS$_\gamma$ can be extended into one from the full π-calculus to CCS$_\gamma^{\rm trig}$ by adding the clause

$$\mathscr{T}([x{=}y]P) \;:=\; [x{=}y]{\Rightarrow}\mathscr{T}(P).$$

Theorem 2 applies to this extended translation as well.

12 Examples

Example 2. The outgoing transitions of $x(y).\bar{y}w$ are

$$\begin{tikzcd}
 & \bar{z}_1 w \arrow[r, "\bar{z}_1 w"] & \mathbf{0} \\
x(y).\bar{y}w \arrow[ur, "xz_1"] \arrow[r, "xz_2"'] \arrow[ddr, "xz_n"'] & \bar{z}_2 w \arrow[r, "\bar{z}_2 w"] & \mathbf{0} \\
 & \vdots & \\
 & \bar{z}_n w \arrow[r, "\bar{z}_n w"] & \mathbf{0}\,.
\end{tikzcd}$$

The same applies to its translation $\sum_{z\in\mathcal{H}} xz.\big((\bar{y}w.\mathbf{0})[^z\!/\!_y]\big)$.

$$\begin{tikzcd}
 & (\bar{y}w.\mathbf{0})[^{z_1}\!/\!_y] \arrow[r, "\bar{z}_1 w"] & \mathbf{0}[^{z_1}\!/\!_y] \\
\sum_{z\in\mathcal{H}} xz.\big((\bar{y}w.\mathbf{0})[^z\!/\!_y]\big) \arrow[ur, "xz_1"] \arrow[r, "xz_2"'] \arrow[ddr, "xz_n"'] & (\bar{y}w.\mathbf{0})[^{z_2}\!/\!_y] \arrow[r, "\bar{z}_2 w"] & \mathbf{0}[^{z_2}\!/\!_y] \\
 & \vdots & \\
 & (\bar{y}w.\mathbf{0})[^{z_n}\!/\!_y] \arrow[r, "\bar{z}_n w"] & \mathbf{0}[^{z_n}\!/\!_y]
\end{tikzcd}$$

Here the z_i range over all names in $\mathcal{N}$. Below I flatten such a picture by drawing the arrows only for one name z, which however still ranges over $\mathcal{N}$.

Example 3. The transitions of $P = x(y).\bar{y}w \mid \bar{x}u.u(v)$ are

$$\begin{tikzcd}
(x(y).\bar{y}w)|\bar{x}u.u(v) \arrow[rr, "xz"] \arrow[d, "\bar{x}u"] \arrow[ddr, "\tau"] & & \bar{z}w|\bar{x}u.u(v) \arrow[rr, "\bar{z}w"] \arrow[d, "\bar{x}u"] & & \mathbf{0}|\bar{x}u.u(v) \arrow[d, "\bar{x}u"] \\
(x(y).\bar{y}w)|u(v) \arrow[rr, "xz"] \arrow[dd, "uq"] & & \bar{z}w|u(v) \arrow[rr, "\bar{z}w"] \arrow[dd, "uq"] \arrow[ddrr, "{[z{=}u]\tau}"] & & \mathbf{0}|u(v) \arrow[dd, "uq"] \\
 & \bar{u}w|u(v) \arrow[drrr, "\tau"] & & & \\
(x(y).\bar{y}w)|\mathbf{0} \arrow[rr, "xz"] & & \bar{z}w|\mathbf{0} \arrow[rr, "\bar{z}w"] & & \mathbf{0}|\mathbf{0}
\end{tikzcd}$$

Here $\bar{u}w|u(v)$ is the special case of $\bar{z}w|u(v)$ obtained by taking $z := u$. It thus also has outgoing transitions labelled $\bar{u}w$ and uq, for $q \in \mathcal{N}$.

Up to strong bisimilarity, the same transition system is obtained by the translation $\mathscr{T}(P)$ of P in CCS$_\gamma$.

$$\mathscr{T}(P) = \left(\sum_{z\in\mathcal{H}} xz.((\bar{y}w.\mathbf{0})[{}^{z}\!/\!{}_{y}])\right)[\ell] \;\Bigg\|\; \left(\bar{x}u.\sum_{z\in\mathcal{H}} uz(\mathbf{0}[{}^{z}\!/\!{}_{v}])\right)[r]$$

Since there are no restriction operators in this example, the relabelling operators $[\ell]$ and $[r]$ are of no consequence. Here

$$\mathscr{T}(P) \xrightarrow{\tau} (\bar{y}w.\mathbf{0})[{}^{u}\!/\!{}_{y}][\ell] \;\Big\|\; \sum_{z\in\mathcal{H}} uz(\mathbf{0}[{}^{z}\!/\!{}_{v}])[r] \xrightarrow{\tau} \mathbf{0}[{}^{u}\!/\!{}_{y}][\ell] \,\|\, \mathbf{0}[{}^{w}\!/\!{}_{v}][r].$$

Example 4. Let $Q = (\nu x)\big(x(y).\bar{y}w \mid (\nu u)\big(\bar{x}u.u(v)\big)\big)$. It has no other transitions than

$$Q \xrightarrow{\tau} (\nu x)(\nu u)\big(\bar{u}w|u(v)\big) \xrightarrow{\tau} (\nu x)(\nu u)(\mathbf{0}|\mathbf{0}).$$

Its translation $\mathscr{T}(Q)$ into CCS$_\gamma$ is

$$\left(\left(\sum_{z\in\mathcal{H}} xz.((\bar{y}w.\mathbf{0})[{}^{z}\!/\!{}_{y}])\right)[\ell] \;\Bigg\|\; \left(\bar{x}u.\sum_{z\in\mathcal{H}} uz(\mathbf{0}[{}^{z}\!/\!{}_{v}])\right)[p_u][r]\right)[p_x]$$

Up to strong bisimilarity, its transition system is the same as that of P or $\mathscr{T}(P)$ from Example 3, except that in transition labels the name u is renamed into the private name ${}^{er}p$, and x is renamed into the private name p. One has $\mathscr{T}(Q) \stackrel{\bullet}{\sim} Q$, since private names generate no barbs.

Example 5. The process $(\nu x)(x(y)) \mid (\nu x)(\bar{x}u)$ has no outgoing transitions. Accordingly, its translation

$$\left(\sum_{z\in\mathcal{H}} xz.(\mathbf{0}\{{}^{z}\!/\!{}_{y}\})\right)[p_x][\ell] \;\Bigg\|\; (\bar{x}u)[p_x][r]$$

only has outgoing transitions labelled ${}^{\ell}pz$ for $z\in\mathcal{H}$ and $\overline{{}^{r}p}u$. Since the names ${}^{\ell}p$ and ${}^{r}p$ are private, these transitions generate no barbs. In this example, the relabelling operators $[\ell]$ and $[r]$ are essential. Without them, the mentioned transitions would have complementary names, and communicate into a τ-transition.

Example 6. Let $P = (\nu y)\big(\bar{x}y.\bar{y}w\big) \mid x(u).u(v)$. Then

$$P \xrightarrow{\tau} (\nu y)\big(\bar{y}w \mid y(v)\big) \xrightarrow{\tau} (\nu y)(\mathbf{0}|\mathbf{0}).$$

Now $\mathscr{T}\,\big((\nu y)\big(\bar{x}y.\bar{y}w\big)\big) = (\bar{x}y.\bar{y}w.\mathbf{0})[p_y]$ and

$$\mathscr{T}(x(u).u(v)) = \sum_{z\in\mathcal{H}} xz.\left(\left(\sum_{z\in\mathcal{H}} uz.(\mathbf{0}[{}^{z}\!/\!{}_{v}])\right)[{}^{z}\!/\!{}_{u}]\right).$$

Hence $\mathscr{T}\left((\nu y)(\bar{x}y.\bar{y}w)\right)[\ell] \xrightarrow{\bar{x}{}^{\ell}p} (\bar{y}w.\mathbf{0})[p_y][\ell]$. Since the substitution r used in the relabelling operator $[r]$ is surjective, there is a name s that is mapped to ${}^{\ell}p$, namely ${}^{\ell}p'$. Considering that $\mathscr{T}(x(u).u(v)) \xrightarrow{xs} \mathscr{T}(u(v))[s/u]$,

$$\mathscr{T}(P) \xrightarrow{\tau} (\bar{y}w.\mathbf{0})[p_y][\ell] \Big\| \left(\sum_{z\in\mathcal{H}} (uz.\mathbf{0})[z/v]\right)[s/u][r].$$

These parallel components can perform actions $\overline{{}^{\ell}p}w$ and ${}^{\ell}pw$, synchronising into a τ-transition, and thereby mimicking the behaviour of P.

Example 7. Let $P = (\nu y)\left(\bar{x}y.(\nu y)(\bar{y}w)\right) \mid x(u).u(v)$.
Then $P \xrightarrow{\tau} (\nu y)\left((\nu y)(\bar{y}w) \mid y(v)\right) \not\xrightarrow{\tau}$. One obtains

$$\mathscr{T}(P) \xrightarrow{\tau} (\bar{y}w.\mathbf{0})[p_y][p_y][\ell] \Big\| \left(\sum_{z\in\mathcal{H}} uz.(\mathbf{0}[z/v])\right)[s/u][r]$$

for a name s that under $[r]$ maps to ${}^{\ell}p$. Now the left component can do an action $\overline{{}^{\ell e}p}w$, whereas the left component can merely match with $\overline{{}^{\ell}p}w$. No synchronisation is possible. This shows why it is necessary that the relabelling $[p_y]$ not only renames y into p, but also p into ${}^{e}p$.

Example 8. Let $P = x(y).x(w).\bar{w}u$. Then

$$P|\bar{x}v.\bar{x}y.y(v) \xrightarrow{\tau} x(w).\bar{w}u|\bar{x}y.y(v) \xrightarrow{\tau} \bar{y}u|y(v) \xrightarrow{\tau} \mathbf{0}|\mathbf{0}.$$

Therefore, $\mathscr{T}(P|\bar{x}v.\bar{x}y.y(v))$ must also be able to start with three consecutive τ-transitions. Note that

$$\mathscr{T}(P|\bar{x}v.\bar{x}y.y(v)) = \mathscr{T}(P)[\ell] \Big\| \left(\bar{x}v.\bar{x}y.\sum_{z\in\mathcal{H}} yz(\mathbf{0}[z/y])\right)[r]$$

with

$$\mathscr{T}(P) = \sum_{z\in\mathcal{H}} xz.\left(\left(\sum_{z\in\mathcal{H}} xz.((\bar{w}u.\mathbf{0})[z/w])\right)[z/y]\right).$$

The only way to obtain $\mathscr{T}(P|\bar{x}v.\bar{x}y.y(v))\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}$ is when $\mathscr{T}(P) \xrightarrow{xv} Q \xrightarrow{xy}\xrightarrow{\bar{y}u}$. The CCS$_\gamma$ process Q must be

$$\left(\sum_{z\in\mathcal{H}} xz.((\bar{w}u.\mathbf{0})[z/w])\right)[v/y].$$

Given the semantics of CCS relabelling, one must have $\sum_{z\in\mathcal{H}} xz.((\bar{w}u.\mathbf{0})[z/w]) \xrightarrow{\alpha}$, such that applying the relabelling $[v/y]$ to α yields xy. When simply taking $[\{v/y\}]$ for $[v/y]$, that is, the relabelling that changes all occurrences of the name y in a transition label into v, this is not possible. This shows that a simplification of my translation without use of the spare names $\mathcal{S}$ would not be valid.

Crucial for this example is that I only use surjective substitutions. $[^v\!/\!_y]$ is an abbreviation of $[\{^v\!/\!_y\}^{\mathcal{S}}]$. Here $\{^v\!/\!_y\}^{\mathcal{S}}$ is a surjective substitution that not only renames y into v, but also sends a spare name s to y. This allows me to take $\alpha := xs$. Consequently, in deriving the transition $\sum_{z\in\mathcal{H}} xz.((\bar{w}u.\mathbf{0})[^z\!/\!_w]) \xrightarrow{\alpha}$, I choose z to be s, so that

$$\sum_{z\in\mathcal{H}} xz.((\bar{w}u.\mathbf{0})[^z\!/\!_w]) \xrightarrow{xs} (\bar{w}u.\mathbf{0})[^s\!/\!_w] \xrightarrow{\bar{s}u} \mathbf{0}[^s\!/\!_w].$$

Putting this in the scope of the relabelling $[^v\!/\!_y]$ yields

$$Q \xrightarrow{xy} (\bar{w}u.\mathbf{0})[^s\!/\!_w][^v\!/\!_y] \xrightarrow{\bar{y}u} \mathbf{0}[^s\!/\!_w][^v\!/\!_y]$$

as desired, and the example works out.[5]

This example shows that spare names play a crucial role in intermediate states of CCS$_\gamma$-translations. In general this leads to stacked relabellings from true names into spare ones and back. Making sure that in the end one always ends up with the right names calls for particularly careful proofs that do not cut corners in the bookkeeping of names.

A last example showing a crucial feature of my translation is discussed in Section 14.

13 The unencodability of CCS into π

Let $f : \mathscr{A} \to \mathscr{A}$ be a CCS relabelling function satisfying $f(x_i y) = x_{i+1}y$. Here $(x_i)_{i=0}^{\infty}$ is an infinite sequence of names, and $\mathscr{A}$ is as in Section 4. The CCS process A defined by

$$A := x_0 y.\mathbf{0} + \tau.(A[f])$$

satisfies $\exists P.\ A \xrightarrow{\tau}{}^{*} P \wedge P{\downarrow_{x_i}}$ for all $i \geq 0$, i.e., it has infinitely many *weak barbs*. It is easy to check that all weak barbs of a π-calculus process Q must be free names of Q, of which there are only finitely many. Consequently, there is no π-calculus process Q with $A \stackrel{\bullet}{\approx} Q$, and hence no translation of CCS in the π-calculus that is valid up to $\stackrel{\bullet}{\approx}$.[6]

14 Related work

My translation from π_{IM} to CCS$_\gamma$ is inspired by an earlier translation $\mathcal{E}$ from a version of the π-calculus to CCS, proposed by Banach & van Breugel [3]. The

[5] This use of spare names solves the problem raised in [3, Footnote 5].

[6] In [28] it was already mentioned, by reference to Pugliese [personal communication, 1997] that CCS relabelling operators cannot be encoded in the π-calculus.

paper [3] takes $\mathscr{A} := \{\langle x,y\rangle \mid x,y \in \mathcal{N}\}$ for the visible CCS actions; action $\langle x,y\rangle$ corresponds with my xy, and its complement $\overline{\langle x,y\rangle}$ with my $\bar{x}y$. On the fragment of π featuring inaction, prefixing, choice and parallel composition, the encoding of [3] is given by

$$\begin{array}{lcl}
\mathcal{E}(\mathbf{0}) & := & \mathbf{0} \\
\mathcal{E}(\tau.P) & := & \tau.\mathcal{E}(P) \\
\mathcal{E}(\bar{x}y.P) & := & \overline{\langle x,y\rangle}.\mathcal{E}(P) \\
\mathcal{E}(x(y).P) & := & \sum_{z\in\mathcal{N}}\langle x,z\rangle.\big(\mathcal{E}(P)[^z\!/\!_y]\big) \\
\mathcal{E}(P \mid Q) & := & \mathcal{E}(P) \mid \mathcal{E}(Q) \\
\mathcal{E}(P+Q) & := & \mathcal{E}(P) + \mathcal{E}(Q).
\end{array}$$

The main result of [3] (Theorem 5.3), stating the correctness of this encoding, says that $P \leftrightarroweq_r Q$ iff $\mathcal{E}(P) \leftrightarroweq_r \mathcal{E}(Q)$, for all π-processes P and Q. Here $\leftrightarroweq_r$ is strong reduction bisimilarity—see Definition 7. In fact, replacing the call to Lemma 3.5 in the proof of this theorem by a call to Lemma 3.4, they could equally well have claimed the stronger result that $P \leftrightarroweq_r \mathcal{E}(P)$ for all π-processes P, i.e., that $\mathcal{E}$ is valid up to $\leftrightarroweq_r$.

This result contradicts my Theorem 1 and thus must be flawed. Where it fails can be detected by pushing the counterexample process $P := \bar{x}v \mid x(y).R$ with $R := \bar{y}u|v(w)$, used in the proof of Theorem 1, through the encoding of [3]. I claim that while $P \xrightarrow{\tau} \bar{v}u|v(w) \xrightarrow{\tau}$, its translation $\mathcal{E}(P)$ cannot do two τ-steps. Hence $P \not\leftrightarroweq_r \mathcal{E}(P)$. Using a trivial process Q such that $P \leftrightarroweq_r Q \leftrightarroweq_r \mathcal{E}(Q)$, this also constitutes a counterexample to [3, Theorem 5.3].

Note that $\mathcal{E}(R) = \overline{\langle y,u\rangle}.\mathbf{0} \mid \sum_{z\in N}\langle v,z\rangle.(\mathbf{0}[^z\!/\!_w])$. This process can perform the actions $\overline{\langle y,u\rangle}$ as well as $\langle v,u\rangle$, but no action τ, since $y \neq v$. Now

$$\mathcal{E}(P) = \overline{\langle x,v\rangle}.\mathbf{0} \mid \sum_{z\in N}\langle x,z\rangle.(\mathcal{E}(R)[^z\!/\!_y]).$$

Its only τ-transition goes to $\mathbf{0} \mid \mathcal{E}(R)[^v\!/\!_y]$. This process can perform the actions $\overline{\langle v,u\rangle}$ as well as $\langle v,u\rangle$, but still no action τ, since $[^v\!/\!_y]$ is a CCS relabelling operator rather than a substitution, and it is applied only after any synchronisations between $\overline{\langle y,u\rangle}.\mathbf{0}$ and $\sum_{z\in N}\langle v,z\rangle.(\mathbf{0}[^z\!/\!_w])$ are derived.

My own encoding $\mathscr{T}$ translates the processes P and R essentially in the same way, but now there is a transition $\mathscr{T}(R) \xrightarrow{[y=v]\tau} (\mathbf{0}\|\mathbf{0}[^u\!/\!_w])$. The renaming $[^v\!/\!_y]$ turns this synchronisation into a τ:

$$\mathscr{T}(P) \xrightarrow{\tau} \mathscr{T}(R)[^v\!/\!_y] \xrightarrow{\tau} (\mathbf{0}\|\mathbf{0}[^u\!/\!_w])[^v\!/\!_y].$$

The crucial innovation of my approach over [3] in this regard is the switch from the early to the early symbolic semantics of the π-calculus, combined with a switch from CCS as target language to CCS$_\gamma$.

In [31], Roscoe argues that CSP is at least as expressive as the π-calculus. As evidence he present a translation from the latter to the former. Roscoe does not provide a criterion for the validity of such a translation, nor a result implying that a suitable criterion has been met. The following observations show that his transition is not compositional, and that it is debatable whether it preserves a reasonable semantic equivalence.

(1) Roscoe translates $\tau.P$ as $tau \rightarrow \textsc{csp}[P]$, where $\rightarrow$ is CSP action prefixing and $\textsc{csp}[P]$ is the translation of the π-expression P. Here tau is a visible CSP action, that is renamed into τ only later in the translation, when combining prefixes into summations. Thus, on the level of prefixes, the translation does not preserve (strong) barbed bisimilarity or any other suitable semantic equivalence. This problem disappears when we stop seeing prefixing and choice as separate operators in the π-calculus, instead using a guarded choice $\sum_{i\in I} \alpha_i.P_i$.
(2) Roscoe translates $x(y).P$ into $x?z \rightarrow \textsc{csp}[P\{z/y\}]$. This is not compositional, since the translation of $x(y).P$ does not merely call the translation of P as a building block, but the result of applying a substitution to P. Substitution is not a CSP operator; it is applied to the π-expression P before translating it. While this mode of translation has some elegance, it is not compositional, and it remains questionable whether a suitable weaker correctness criterion can be formulated that takes the place of compositionality here.
(3) To deal with restriction, [31] works with translations $\textsc{csp}[P]_{\kappa,\sigma}$, where two parameters κ and σ are passed along that keep track of sets of fresh names to translate restricted names into. The set of fresh names σ is partitioned in the translation of $P|Q$ (page 388), such that both sides get disjoint sets of fresh names to work with. Although the idea is rather similar to the one used here, the passing of the parameters makes the translation non-compositional. In a compositional translation $\textsc{csp}[P|Q]$ the arguments P and Q may appear in the translated CSP process only in the shape $\textsc{csp}[P]$ and $\textsc{csp}[Q]$, not $\textsc{csp}[P]_{\kappa,\sigma'}$ for new values of σ'.

As pointed out in [14,29], even the most bizarre translations can be found valid if one only imposes requirements based on semantic equivalence, and not compositionality. Roscoe's translation is actually rather elegant. However, we do not have a decent criterion to say to what extent it is a valid translation. The expressiveness community strongly values compositionality as a criterion, and this attribute is the novelty brought in by my translation.

15 Conclusion

This paper exhibited a compositional translation from the π-calculus to CCS$_\gamma$ extended with triggering that is valid up to strong barbed bisimilarity, thereby showing that the latter language is at least as expressive as the former. Triggering is not needed when restricting to the π-calculus with implicit matching (as used for instance in [33]). Conversely, I observed that CCS (and thus certainly CCS$_\gamma$) cannot be encoded in the π-calculus. I also showed that the upgrade of CCS to CCS$_\gamma$ is necessary to capture the expressiveness of the π-calculus.

A consequence of this work is that any system specification or verification that is carried out in the setting of the π-calculus can be replayed in CCS$_\gamma$. The main idea here is to replace the names that are kept private in the π-calculus by means of the restriction operator, by names that are kept private by means of a careful bookkeeping ensuring that the same private name is never used twice. Of course this in no way suggests that it would be preferable to replay π-calculus specifications or verifications in CCS_γ.

My translation encodes the restriction operator (νy) from the π-calculus by renaming y into a "private name". Crucial for this approach is that private names generate no barbs, in contrast with standard approaches where all names generate barbs. This use of private names is part of the definition of strong barbed bisimilarity $\overset{\bullet}{\sim}$ on my chosen instance of CCS_γ, and justified since that definition is custom made in the present paper. The use of private names can be avoided by placing an outermost CCS restriction operator around any translated π-process. This, however, would violate the compositionality of my translation.

The use of infinite summation in my encoding might be considered a serious drawback. However, when sticking to a countable set of π-calculus names, only countable summation is needed, which, as shown in [8], can be eliminated in favour of unguarded recursion with infinitely many recursion equations. As the original presentation of the π-calculus already allows unguarded recursion with infinitely many recursion equations [24] the latter can not reasonably be forbidden in the target language of the translation. Still, it is an interesting question whether infinite sums or infinite sets of recursion equations can be avoided in the target language if we rule them out in the source language. My conjecture is that this is possible, but at the expense of further upgrading CCS_γ, say to aprACP_R^τ. This would however require work that goes well beyond what is presented here.

An alternative approach is to use a version of CCS featuring a *choice quantifier* [17] instead of infinitary summation, a construct that looks remarkably like an infinite sum, but is as finite as any quantifier from predicate logic. A choice quantifier binds a data variable z (here ranging over names) to a single process expression featuring z. The present application would need a function from names to CCS relabelling operators. When using this approach, the size of translated expressions becomes linear in the size of the originals.

It could be argued that choice quantification is a step towards mobility. On the other hand, if mobility is associated more with scope extrusion than with name binding itself, one could classify CCS_γ with choice quantification as an immobile process algebra. A form of choice quantification is standard in mCRL2 [15], which is often regarded "immobile".

My translation from π to CCS_γ has a lot in common with the attempted translation of π to CCS in [3]. That one is based on the early operational semantics of CCS, rather than the early symbolic one used here. As a consequence, substitutions there cannot be eliminated in favour of relabelling operators.

A crucial step in my translation yields an intermediate language with an operational semantics in De Simone format. In [7] another representation of the π-calculus is given through an operational semantics in the De Simone format. It uses a different way of dealing with substitutions. This type of semantics could be an alternative stepping stone in an encoding from the π-calculus into CCS_γ.

In [28] Palamidessi showed that there exists no uniform encoding of the π-calculus into a variant of CCS. Here *uniform* means that $\mathscr{T}(P|Q)=\mathscr{T}(P)|\mathscr{T}(Q)$. This does not contradict my result in any way, as my encoding is not uniform. Palamidessi [28] finds uniformity a reasonable criterion for encodings, because it guarantees that the translation maintains the degree of distribution of the system. In [30], however, it is argued that it is possible to maintain the degree of distribution of a system upon translation without requiring uniformity. In fact, the translation offered here is a good example of one that is not uniform, yet maintains the degree of distribution.

Gorla [13] proposes five criteria for valid encodings, and shows that there exists no valid encoding of the π-calculus (even its asynchronous fragment) into CCS. Gorla's proof heavily relies on the criterion of *name invariance* imposed on valid encodings. It requires for $P \in \mathrm{T}_\pi$ and an injective substitution σ that $\mathscr{T}(P\sigma) = \mathscr{T}(P)\sigma'$ for some substitution σ' that is obtained from σ through a *renaming policy*. Furthermore, the renaming policy is such that if $dom(\sigma)$ is finite, then also $dom(\sigma')$ is finite. This latter requirement is not met by the encoding presented here, for a single name $x \in \mathcal{N}$ corresponds with an infinite set of actions xy, the "names" of CCS, and a substitution that merely renames x into z must rename each action xy into zy at the CCS end, thus violating the finiteness of $dom(\sigma')$.

My encoding also violates Gorla's compositionality requirement, on grounds that $\mathscr{T}(P)$ appears multiple times (actually, infinitely many) in the translation of $Mx(y).P$. It is however compositional by the definition in [10] and elsewhere. My encoding satisfies all other criteria of [13] (operational correspondence, divergence reflection and success sensitiveness).

References

1. Austry, D., Boudol, G.: Algèbre de processus et synchronisations. TCS **30**(1), 91–131 (1984). https://doi.org/10.1016/0304-3975(84)90067-7
2. Baeten, J.C.M., Weijland, W.P.: Process Algebra. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press (1990). https://doi.org/10.1017/CBO9780511624193
3. Banach, R., van Breugel, F.: Mobility and modularity: expressing π-calculus in CCS. Preprint (1998), http://www.cs.man.ac.uk/~banach/some.pubs/Pi.CCS.ext.abs.pdf
4. Bergstra, J.A., Klop, J.W.: Algebra of communicating processes. In: Mathematics and Computer Science, pp. 89–138. CWI Monograph 1, North-Holland (1986)
5. Boudol, G.: Asynchrony and the π-calculus (note). Tech. Rep. 1702, INRIA (1992)
6. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. ACM **31**(3), 560–599 (1984). https://doi.org/10.1145/828.833
7. Ferrari, G.L., Montanari, U., Quaglia, P.: A pi-calculus with explicit substitutions. Theoretical Computer Science **168**(1), 53–103 (1996). https://doi.org/10.1016/S0304-3975(96)00063-1
8. Glabbeek, R.J. van: On the expressiveness of ACP (extended abstract). In: Proc. ACP'94. pp. 188–217. Workshops in Computing, Springer (1994). https://doi.org/10.1007/978-1-4471-2120-6_8

9. Glabbeek, R.J. van: On cool congruence formats for weak bisimulations. Theoretical Computer Science **412**(28), 3283–3302 (2011). https://doi.org/10.1016/j.tcs.2011.02.036
10. Glabbeek, R.J. van: Musings on encodings and expressiveness. In: Proc. EXPRESS/SOS'12. EPTCS, vol. 89, pp. 81–98. Open Publishing Association (2012). https://doi.org/10.4204/EPTCS.89.7
11. Glabbeek, R.J. van: A theory of encodings and expressiveness. In: Proc. FoSSaCS'18. LNCS, vol. 10803, pp. 183–202. Springer (2018). https://doi.org/10.1007/978-3-319-89366-2_10
12. Glabbeek, R.J. van, Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43**(3), 555–600 (1996). https://doi.org/10.1145/233551.233556
13. Gorla, D.: Towards a unified approach to encodability and separation results for process calculi. Information and Computation **208**(9), 1031–1053 (2010). https://doi.org/10.1016/j.ic.2010.05.002
14. Gorla, D., Nestmann, U.: Full abstraction for expressiveness: history, myths and facts. Mathematical Structures in Computer Science **26**(4), 639–654 (2016). https://doi.org/10.1017/S0960129514000279
15. Groote, J.F., Mousavi, M.R.: Modeling and Analysis of Communicating Systems. MIT Press (2014)
16. Hennessy, M., Lin, H.: Symbolic bisimulations. Theoretical Comp. Sc. **138**(2), 353–389 (1995). https://doi.org/10.1016/0304-3975(94)00172-F
17. Luttik, B.: On the expressiveness of choice quantification. Ann. Pure Appl. Logic **121**, 39–87 (2003). https://doi.org/10.1016/S0168-0072(02)00082-9
18. Milner, R.: Calculi for synchrony and asynchrony. Theoretical Comp. Sc. **25**, 267–310 (1983). https://doi.org/10.1016/0304-3975(83)90114-7
19. Milner, R.: Communication and Concurrency. Prentice Hall, Englewood Cliffs (1989)
20. Milner, R.: Operational and algebraic semantics of concurrent processes. In: Handbook of Theoretical Computer Science, chap. 19, pp. 1201–1242. Elsevier Science Publishers B.V. (North-Holland) (1990)
21. Milner, R.: Functions as processes. Mathematical Structures in Computer Science **2**(2), 119–141 (1992). https://doi.org/10.1017/S0960129500001407
22. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press (1999)
23. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I. I&C **100**, 1–40 (1992). https://doi.org/10.1016/0890-5401(92)90008-4
24. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, II. I&C **100**, 41–77 (1992). https://doi.org/10.1016/0890-5401(92)90009-5
25. Milner, R., Parrow, J., Walker, D.: Modal logics for mobile processes. TCS **114**, 149–171 (1993). https://doi.org/10.1016/0304-3975(93)90156-N
26. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Proc. ICALP'92. LNCS, vol. 623, pp. 685–695. Springer (1992). https://doi.org/10.1007/3-540-55719-9_114
27. Nestmann, U.: Welcome to the jungle: A subjective guide to mobile process calculi. In: Proc. CONCUR'06. LNCS, vol. 4137, pp. 52–63. Springer (2006). https://doi.org/10.1007/11817949_4
28. Palamidessi, C.: Comparing the expressive power of the synchronous and asynchronous pi-calculi. Mathematical Structures in Comp. Science **13**(5), 685–719 (2003). https://doi.org/10.1017/S0960129503004043

29. Parrow, J.: General conditions for full abstraction. Math. Struct. in Comp. Sc. **26**(4), 655–657 (2016). https://doi.org/10.1017/S0960129514000280
30. Peters, K., Nestmann, U., Goltz, U.: On distributability in process calculi. In: Proc. ESOP'13. LNCS, vol. 7792, pp. 310–329. Springer (2013). https://doi.org/10.1007/978-3-642-37036-6_18
31. Roscoe, A.W.: CSP is expressive enough for π. In: Reflections on the Work of C.A.R. Hoare, pp. 371–404. Springer (2010). https://doi.org/10.1007/978-1-84882-912-1_16
32. Sangiorgi, D.: A theory of bisimulation for the pi-calculus. Acta Inf. **33**(1), 69–97 (1996). https://doi.org/10.1007/s002360050036
33. Sangiorgi, D., Walker, D.: The π-calculus: A Theory of Mobile Processes. Cambridge University Press (2001)
34. Simone, R. de: Higher-level synchronising devices in Meije-SCCS. TCS **37**, 245–267 (1985). https://doi.org/10.1016/0304-3975(85)90093-3