Comparing the expressiveness of the π-calculus and CCS

ROB VAN GLABBEEK, Data61, CSIRO, Sydney, Australia,
and School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

This paper shows that the π-calculus with implicit matching is no more expressive than CCS_γ, 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_γ that is valid up to strong barbed bisimilarity.

The full π-calculus can be similarly expressed in CCS_γ enriched with the triggering operation of Meije.

I also show that these results cannot be recreated with CCS in the rôle of CCS_γ, 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.

CCS Concepts: • **Theory of computation → Process calculi**.

Additional Key Words and Phrases: Expressiveness, π-calculus, CCS, mobility, translations, valid encodings, compositionality, strong barbed bisimilarity

ACM Reference Format:
Rob van Glabbeek. 2022. Comparing the expressiveness of the π-calculus and CCS. *ACM Trans. Comput. Logic* 37, 4, Article 111 (March 2022), 58 pages. https://doi.org/10.48550/arXiv.2203.11519

1 INTRODUCTION

The π-calculus [24–26, 35] has been advertised as an "extension to the process algebra CCS" [25] 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 [29]—such as ACP and CSP.

> "the π-calculus has a much greater expressiveness than CCS"
> [Sangiorgi [34]]

> "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 [24]]

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

Following [11, 12] 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.

An extended abstract of this paper appeared in Proc. ESOP'22 [13]. The current version adds explanation here and there, and includes an appendix with the proof of my main theorem.
Author's current affiliation and address: Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh, Informatics Forum, Office 3.49, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom, rvg@cs.stanford.edu.

To appear in ACM Transactions on Computational Logic and at https://arxiv.org/abs/2203.11519.

Robert de Simone [36] showed that a wide class of process calculi, including CCS [22], CSP [6], ACP [4] and SCCS [20], are expressible up to strong bisimilarity in MEIJE [1]. In [9] 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 [36], 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* [28, 35]. 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 [28, 35]. For CCS, the familiar notion of *strong bisimilarity* [21] 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 [35]. 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 [35].

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 [25, 26]. It turns out, however, that the matching operator $[x{=}y]P$ of [25, 26] calls for a target language that is a step further away from CCS. The book [35] 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 [25, 26] to facilitate complete equational axiomatisations of the π-calculus, and [35] 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_γ with the *triggering* operator of Meije, which allows a relabelling of the first action of its argument only.

In stating that each translated π-calculus expression is strongly barbed bisimilar to its original, I need a precise definition of strong barbed bisimilarity on both the π-calculus and $\text{CCS}_\gamma^{\text{trig}}$. For π I take the standard definition [35] in which a process has a barb $\bar{x}$, resp. x, exactly when it has a outgoing transition labelled $\bar{x}y$ or $\bar{x}(y)$, resp. xy or $x(y)$, in its early (or equivalently late) operational semantics. On the side of CCS there is no such established standard definition of a barb. As $\text{CCS}_\gamma^{\text{trig}}$, like CCS, is parametrised by the choice of a set $\mathscr{A}$ of actions, I may choose any such $\mathscr{A}$ that works best. I do this by including in $\mathscr{A}$ all actions $\bar{x}y$ and xy for pairs of π-calculus names x and y, but I also include such actions for fresh names x and y, as well as actions of different forms. For the purpose of defining barbs on this chosen instantiation of $\text{CCS}_\gamma^{\text{trig}}$, I partition the set of names x and y from which I built actions into *public* and *private* names, making sure that all names inherited from the π-calculus are classified as public. Now I define $\bar{x}$, resp. x, to be barb of a $\text{CCS}_\gamma^{\text{trig}}$ process, exactly when it has a outgoing transition whose label has the form $\bar{x}y$, resp. xy, with x a public name. As a result, $\text{CCS}_\gamma^{\text{trig}}$-transition labels that do not match early transition labels from the π-calculus typically generate no barbs. This mode of operation allows me to mimic the essence of the restriction operator from the π-calculus in $\text{CCS}_\gamma^{\text{trig}}$ by renaming a restricted π-calculus name into a private name that generates no barbs.

By a general result of [12], the validity up to strong barbed bisimilarity of my translation from π_{IM} to CCS_γ (and from π to $\text{CCS}_\gamma^{\text{trig}}$) 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 $\text{CCS}_\gamma^{\text{trig}}$ 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 $\text{CCS}_\gamma^{\text{trig}}$ contexts. As a consequence, any statement $P \sim_E Q$ stating that two π-calculus processes P and Q—possibly a specification and an implementation of a distributed system—are strongly early congruent, can be translated in a statement $\mathscr{T}(P) \sim_E \mathscr{T}(Q)$, where $\mathscr{T}(P)$ and $\mathscr{T}(Q)$ are the $\text{CCS}_\gamma^{\text{trig}}$-processes obtained from P and Q through my transition, and $\sim_E$ is a new semantic equivalence on $\text{CCS}_\gamma^{\text{trig}}$ that arises as the congruence closure of strong barbed bisimilarity under translated contexts. This makes it possible to replay π-calculus verifications in $\text{CCS}_\gamma^{\text{trig}}$.

Having established that π_{IM} can be expressed in CCS_γ, 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_γ—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.

Table 1. Structural operational semantics of CCS

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

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

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

2 CCS

CCS [21] is parametrised with 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 $\mathscr{L} := \mathscr{A} \uplus \overline{\mathscr{A}}$ 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, $a, b, c, \ldots$ range over $\mathscr{L}$ and α, β over Act. A *relabelling* is a function $f : \mathscr{L} \to \mathscr{L}$ satisfying $f(\bar{a}) = \overline{f(a)}$; it extends to Act by $f(\tau) := \tau$. The class $\mathrm{T_{CCS}}$ of CCS *terms*, *expressions*, *processes* or *agents* is the smallest class[1] including:

$\alpha.P$	for $\alpha \in Act$ and $P \in \mathrm{T_{CCS}}$	*prefixing*
$\sum_{i\in I} P_i$	for I an index set and $P_i \in \mathrm{T_{CCS}}$	*choice*
$P\|Q$	for $P, Q \in \mathrm{T_{CCS}}$	*parallel composition*
$P\backslash L$	for $L \subseteq \mathscr{L}$ and $P \in \mathrm{T_{CCS}}$	*restriction*
$P[f]$	for f a relabelling and $P \in \mathrm{T_{CCS}}$	*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 $A \stackrel{\text{def}}{=} P$, with $P \in \mathrm{T_{CCS}}$. The semantics of CCS is given by the labelled transition relation $\to \subseteq \mathrm{T_{CCS}} \times Act \times \mathrm{T_{CCS}}$. The transitions $P \xrightarrow{\alpha} Q$ with $P, Q \in \mathrm{T_{CCS}}$ and $\alpha \in Act$ are derived from the rules of Table 1.

Arguably, the most authentic version of CCS [22] 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 [22], 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_γ

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

[1] CCS [21, 22] allows arbitrary index sets I in summations $\sum_{i\in I} P_i$. As a consequence, $\mathrm{T_{CCS}}$ 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 $\mathrm{CCS}^{\mathrm{trig}}_\gamma$, 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.

Table 2. The actions

α	Kind	$O(\alpha)$	$\mathrm{fn}(\alpha)$	$\mathrm{bn}(\alpha)$
$M\tau$	Silent	–	$\mathrm{n}(M)$	$\emptyset$
$M\bar{x}y$	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	$\mathrm{n}(M) \cup \{x\}$	$\{y\}$

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

Moreover, relabelling operators $f : \mathscr{A} \to Act$ 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_γ.

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

I write $P \xrightarrow{\alpha} Q$ for $(P, \alpha, Q) \in \to$, $P \xrightarrow{\alpha}$ for $\exists Q.\ P \xrightarrow{\alpha} Q$, and $P \not\xrightarrow{\alpha}$ 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 4.2. A *strong bisimulation* is a symmetric relation $\mathscr{R}$ on the states of an LTS such that

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

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

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

Definition 4.3. 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.

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_γ need to have

[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.

the forms of Table 2. For the translation into barbed transition systems I take $B := \mathcal{L} \cup \overline{\mathcal{L}}$, where $\overline{\mathcal{L}} := \{\bar{a} \mid a \in \mathcal{L}\}$, and $O(\alpha)$ as indicated in Table 2, provided $M = \varepsilon$ and $O(\alpha) \in B$. Thus transitions labelled $M\bar{x}y$, $M\bar{x}(y)$, Mxy or $Mx(y)$ with $M \neq \epsilon$ or $x \in \mathcal{R}$ generate no barbs.

Definition 4.4. A *strong barbed bisimulation* is a symmetric relation $\mathscr{R}$ on the states of a BTS such that

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

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

Again, $\mathrel{\dot{\sim}}$ 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_γ. It can also be used to compare processes from different LTSs, namely by taking their disjoint union.

5 THE π-CALCULUS

The π-calculus [25, 26] 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})—addressed in blue here—drops the matching operator, instead allowing prefixes of the form $M\tau.P$, $M\bar{x}y.P$ and $Mx(y).P$, with M a matching sequence.

A *substitution* is a partial function $\sigma\colon \mathcal{N} \rightharpoonup \mathcal{N}$ such that $\mathcal{N} \setminus (dom(\sigma) \cup range(\sigma))$ 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 $x \in dom(\sigma)$ 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 $x \in dom(\sigma)$, 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:

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

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

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 [26], 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 $\mathcal{Z} := \mathcal{N}$ 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 [26] the *late* and *early* bisimulation semantics of the π-calculus were proposed.

Definition 6.1. A *late bisimulation* is a symmetric relation $\mathscr{R}$ on π-processes such that, whenever $P \mathscr{R} Q$, 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 $P' \mathscr{R} Q'$,
(2) if $P \xrightarrow{x(z)} P'$ then $\exists Q' \forall y.\ Q \xrightarrow{x(z)} Q' \wedge P'\{y/z\} \mathscr{R} Q'\{y/z\}$,
(3) if $P \xrightarrow{\bar{x}(z)} P'$ then $\exists Q'$ with $Q \xrightarrow{\bar{x}(z)} Q'$ and $P' \mathscr{R} Q'$.

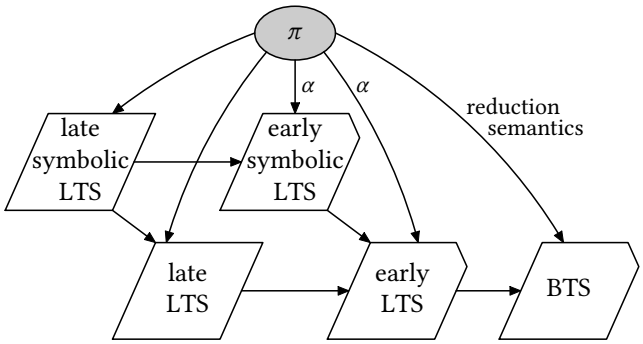


Fig. 1. Semantics of the π-calculus

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

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

MATCH:
$$\frac{P \xrightarrow{\alpha} P'}{[x{=}x]P \xrightarrow{\alpha} P'}$$

IDE:
$$\frac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'} \; (A(\vec{x}) \stackrel{\text{def}}{=} P)$$

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

COM:
$$\frac{P \xrightarrow{\bar{x}y} P',\ Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$$

CLOSE:
$$\frac{P \xrightarrow{\bar{x}(z)} P',\ Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$$

RES:
$$\frac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'} \; (y \notin \mathrm{n}(\alpha))$$

ALPHA-OPEN:
$$\frac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(z)} P'\{z/y\}} \begin{pmatrix} y \neq x \\ z \notin \mathrm{fn}((\nu y)P')\end{pmatrix}$$

The rules **SUM**, **PAR**, **COM** and **CLOSE** additionally have symmetric forms, with the rôles of P and Q exchanged.

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

MATCH:
$$\frac{P \xrightarrow{\alpha} P'}{[x{=}x]P \xrightarrow{\alpha} P'}$$

IDE:
$$\frac{P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'} \; (A(\vec{x}) \stackrel{\text{def}}{=} P)$$

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

EARLY-COM:
$$\frac{P \xrightarrow{\bar{x}y} P',\ Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

EARLY-CLOSE:
$$\frac{P \xrightarrow{\bar{x}(z)} P',\ Q \xrightarrow{xz} Q'}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')} \begin{pmatrix} z \notin \\ \mathrm{fn}(Q)\end{pmatrix}$$

RES:
$$\frac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'} \; (y \notin \mathrm{n}(\alpha))$$

OPEN:
$$\frac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(y)} P'} \; (y \neq x)$$

ALPHA:
$$\frac{P \equiv Q, \quad Q \xrightarrow{\alpha} Q'}{P \xrightarrow{\alpha} Q'}$$

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 $\{\vec{y}/\vec{x}\}$.

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 [26, 35] 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 6.2 ([26]). Let $P \equiv Q$ and $\mathrm{bn}(\alpha) \cap \mathrm{n}(Q) = \emptyset$.
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 [27] 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 [35]. The semantics

in [27, 35] 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 6.2, 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 6.3. 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 $P \equiv \bar{x}y|(\nu w)(x(z)) \xrightarrow{\tau}_E \mathbf{0}|(\nu w)\mathbf{0}$.

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

Lemma 6.4 ([27]). Let $P \in \mathrm{T}_\pi$ 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 6.5 ([27]). An *early bisimulation* is a symmetric relation $\mathscr{R}$ on T_π such that, whenever $P \mathscr{R} Q$ and α is an action with $\mathrm{bn}(\alpha) \cap (\mathrm{n}(P) \cup \mathrm{n}(Q)) = \emptyset$,
- if $P \xrightarrow{\alpha}_E P'$ then $\exists Q'$ with $Q \xrightarrow{\alpha}_E Q'$ and $P' \mathscr{R} Q'$.

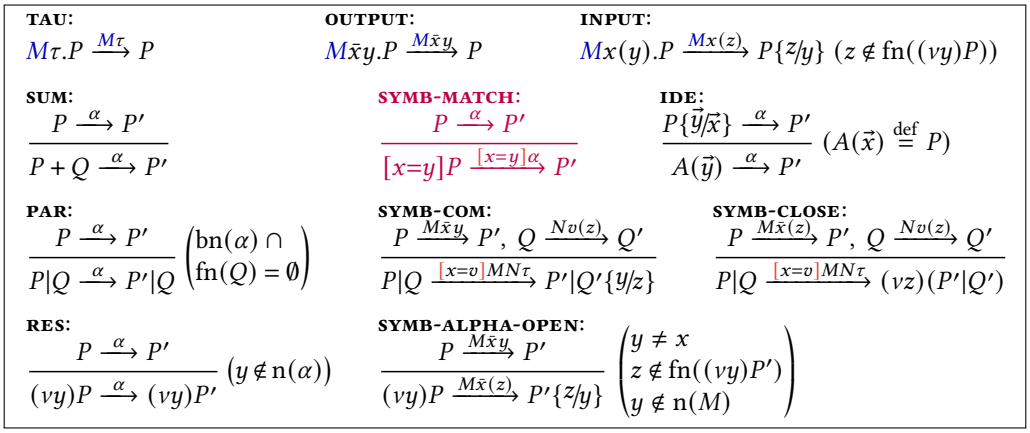
Processes P and Q are early bisimilar iff $P \mathscr{R} Q$ for some early bisimulation $\mathscr{R}$.

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 6.2 and 6.4 the same BTS is obtained either way. This defines strong barbed bisimilarity $\dot{\sim}$ on T_π. The congruence closure of $\dot{\sim}$ is early congruence [35]. In [23] 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 [34] yet another operational semantics of the π-calculus was introduced, in a style called *symbolic* by Hennessy & Lin [18], 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.

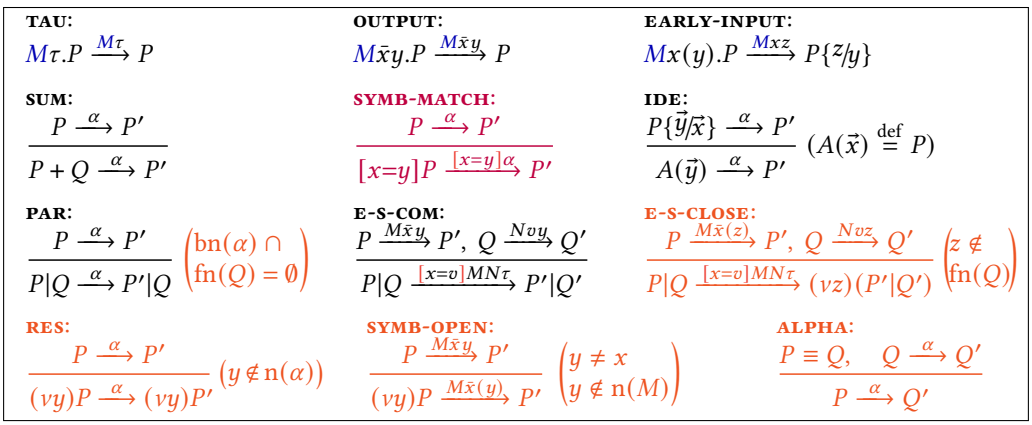
In the operational semantics of CCS, τ-actions can be thought of as reactions that actually take place, whereas a transition labelled a merely represents the potential of a reaction with the environment, one that can take place only if the environment offers a complementary transition $\bar{a}$. In case the environment never does an $\bar{a}$, this potential will not be realised. A reduction semantics (as in [24]) 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.

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



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

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



The orange rules become redundant with dropping the restriction operator—cf. Section 10.

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 with actions having an empty matching sequence are exactly the transitions of the early semantics (actions with a nonempty matching sequence generate no barbs), 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).

The main result of this paper is a compositional translation from π_{IM} to CCS_γ (and from π to $\mathrm{CCS}_\gamma^{\mathrm{trig}}$) such that each π-process is strongly barbed bisimilar to its translation. This is a result about the BTS of the π-calculus. As we have seen that all 5 semantics of the π-calculus illustrated in Figure 1 lead to the same BTS, I may—and will—prove this result in terms of the BTS extracted from the early symbolic LTS of the π-calculus, generated by the rules of Table 6; it will then apply equally well to any of the other four semantic routes of associating this BTS with π.

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 $g \in \Sigma$,

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

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 $\tau.$, $\bar{x}y.$, $x(y).$, (νy) and $[x{=}y]$ for $x, y \in \mathcal{N}$, and the nullary operators (or *constants*) $\mathbf{0}$ and $A(y_1, \ldots, y_n)$ 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, $C[P_1, \ldots, P_n]$ is the result of substituting P_i for X_i, for each $i = 1, \ldots, n$. A ternary context is often introduced as $C[\ ,\ ,\ ]$.

Definition 7.1. 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 $\mathscr{T}(g(P_1, \ldots, P_n)) = C_g[\mathscr{T}(P_1), \ldots, \mathscr{T}(P_n)]$.

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 [11, 12], but could be simplified here since [11, 12] also cover the case that $\mathcal{L}'$ is not simple. Moreover, here I restrict attention to what are called *closed term languages* in [12].

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 $\dot{\sim}$. 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).

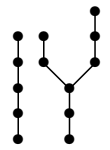
Definition 8.1. Strong reduction bisimilarity, $\leftrightarrows_r$, is defined just as strong barbed equivalence in Definition 4.4, but without the requirement on barbs.

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

Definition 8.2. Let $\leftarrowtail$ be the smallest preorder on CCS contexts such that $\sum_{i\in I} E_i \leftarrowtail E_j$ for all $j \in I$, $E|F \leftarrowtail E$, $E|F \leftarrowtail F$, $E\backslash L \leftarrowtail E$, $E[f] \leftarrowtail E$ and $A \leftarrowtail P$ for all $A \in \mathcal{K}$ with $A \stackrel{\text{def}}{=} P$. A variable X occurs *unguarded* in a context E if $E \leftarrowtail X$.

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 8.3. 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. □

Lemma 8.4. 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 8.5. There is no translation from $\pi^{\P}_A$ to CCS that is valid up to $\leftrightarrows_r$.

Proof. Suppose, towards a contradiction, that $\mathscr{T}$ is a translation from $\pi^{\P}_A$ to CCS that is valid up to $\leftrightarrows_r$. This means that $\mathscr{T}$ is compositional and that $\mathscr{T}(P) \leftrightarrows_r P$ for any $\pi^{\P}_A$-process P.

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

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

Since $\bar{x}v|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0}) \xrightarrow{\tau}$ as well as $\mathscr{T}\big(\bar{x}v|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})\big) \leftrightarrows_r \bar{x}v|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})$, it follows that $\mathscr{T}\big(\bar{x}v|x(y).(\mathbf{0}|\mathbf{0}|\mathbf{0})\big) \xrightarrow{\tau}$, i.e., $D[\mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0}), \mathscr{T}(\mathbf{0})] \xrightarrow{\tau}$. Hence Lemma 8.4 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 $\pi^{\P}_A$-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 $\bar{x}v|x(y).(R|\mathbf{0}|\mathbf{0})$, for $R \in \{\bar{y}z|v(w), 0|v(w), \bar{y}z|0, \tau\}$.

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

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

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 $\leftrightarrows_r$ and there is no $S \in \mathrm{T}_\pi$ such that

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

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

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

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

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

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

Moreover since $\tau \xrightarrow{\tau}$, also $U := \mathscr{T}(\tau) \xrightarrow{\tau}$, but, as 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 8.3, $\mathscr{T}(\bar{y}z|v(w)) = C_|[\mathscr{T}(\bar{y}z), \mathscr{T}(v(w))] \xrightarrow{\tau}$, yet $\bar{y}z|v(w) \not\xrightarrow{\tau}$. This contradicts the validity of $\mathscr{T}$ up to $\leftrightarrows_r$. □

9 A VALID TRANSLATION OF π_{IM} INTO CCS_γ

In this section I will define my translation $\mathscr{T}$ from π_{IM} to CCS_γ and state the theorem that asserts is correctness. The proof of that theorem is 34 pages long and given in the appendix. Since π_{IM} is parameterised with an infinite set $\mathcal{N}$ of names and sets $\mathcal{K}_n$ of n-ary agent identifiers, wheres CCS_γ is parameterised with a set $\mathcal{K}$ of agent identifiers, an alphabet $\mathscr{A}$ of visible actions, a subset $\mathscr{S} \subseteq \mathscr{A}$ of synchronisations, and a partial communication function $\gamma : (\mathscr{A}\backslash\mathscr{S})^2 \rightharpoonup \mathscr{S} \cup \{\tau\}$, first I must choose the parameters $\mathcal{K}$, $\mathscr{A}$, $\mathscr{S}$ and γ of CCS_γ as a function of $\mathcal{N}$ and the $\mathcal{K}_n$.

To start, $\mathcal{K}$ 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.

To obtain $\mathscr{A}$, I first of all introduce an infinite set $\mathcal{S} = \{s_1, s_2, \dots\}$ of *spare names*, disjoint from $\mathcal{N}$, and define $\mathcal{Z} := \mathcal{N} \uplus \mathcal{S}$ to be the set of *public names*. I furthermore will introduce a set $\mathcal{R}$ of *private names*, disjoint from $\mathcal{Z}$, and define $\mathcal{H} := \mathcal{Z} \uplus \mathcal{R}$, the set of all names. I take *Act* to be the set of all expressions $M\tau$, $M\bar{x}y$ and Mxy from Table 2, as defined in Section 4 (in terms of $\mathcal{Z}$ and $\mathcal{R}$), so $\mathscr{A} := Act\backslash\{\tau\}$. Here x and y range over $\mathcal{H}$, and M is a (possibly empty) matching sequence, a sequence of matches $[x{=}y]$ with $x, y \in \mathcal{H}$ and $x \neq y$. Recall that transitions labelled $M\tau$, $M\bar{x}y$ or Mxy with $M \neq \epsilon$ generate no barbs, nor do transitions labelled τ, or transitions labelled $\bar{x}y$ or xy, with $x \in \mathcal{R}$ a private name. My translation will be such that $\mathscr{T}(P) \mathrel{\dot{\leftrightarrow}} P$ for each π_{IM} process P. As P by definition does not have barbs $s \in \mathcal{S}$, neither does the CCS_γ-process $\mathscr{T}(P)$.

The set $\mathscr{S} \subseteq \mathscr{A}$ of synchronisations consists of all actions $M\tau$ with $M \neq \varepsilon$. The communication function $\gamma : (\mathscr{A}\backslash\mathscr{S})^2 \rightharpoonup \mathscr{S} \cup \{\tau\}$ is given by $\gamma(M\bar{x}y, Nvy) = [x{=}v]MN\tau$, and its commutative variant, just as for rule **E-S-COM** in Table 6. Recall from Section 6 that $[x{=}v]M$ denotes the matching sequence M prepended with $[x{=}v]$ if $x \neq v$, whereas $[x{=}x]M$ simply denotes M.

The compositional encoding of Table 7, which will be illustrated with examples in Section 11, defines my translation from π_{IM} to CCS_γ. Here the CCS_γ agent identifier A has the defining equation $A \stackrel{\text{def}}{=} \mathscr{T}(P)$ when $A(\vec{x}) \stackrel{\text{def}}{=} P$ was the defining equation of the agent identifier A from the π-calculus. In Table 7, $[z/y]$, $[p_y]$, $[\ell]$, $[r]$ and $[\vec{y}/\vec{x}]$ are CCS relabelling operators, to be formally defined below.

To explain what this encoding does, inaction, silent prefix, output prefix and choice are translated homomorphically. The input prefix $Mx(y).P$ is translated into an infinite sum over all possible input values z that could be received, of the input action Mxz followed by the continuation

Table 7. Translation from π_{IM} to CCS_γ.

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

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 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. Additionally, the operator $[p_y]$ injectively renames all private names that already occur in the scope of (νy). This ensures that the new private name p is fresh, so that no name clashes occur.

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.

It remains to define the relabelling functions z/y, p_y, ℓ, r and $\vec{y}/\vec{x}$ of type $\mathscr{A} \to \mathscr{A}$ that underlay the CCS relabelling operators $[z/y]$, $[p_y]$ $[\ell]$, $[r]$ and $[\vec{y}/\vec{x}]$ of Table 7. Here p_y is defined for all $y \in \mathcal{Z}$, and $\vec{y}/\vec{x}$ is defined when $\vec{x} = (x_i)_{i=1}^n \in \mathcal{N}^n$ and $\vec{y} = (y_i)_{i=1}^n \in \mathcal{H}^n$ are vectors of names of the same length, with all the x_i distinct. The relabelling z/y is a special case of this with $n = 1$.

Each of these relabelling functions $\sigma : \mathscr{A} \to \mathscr{A}$ is given as substitution of type $\sigma : \mathcal{H} \rightharpoonup \mathcal{H}$ that extends to a total function of type $\sigma : \mathcal{H} \to \mathcal{H}$ by $f(x) = x$ when $x \notin \underline{dom(\sigma)}$. These functions lift homomorphically to $\mathscr{A}$, so that for instance $\sigma([u{=}v]\bar{x}y) := [\sigma(u){=}\sigma(v)]\overline{\sigma(x)}\sigma(y)$.

In order for my encoding to work, these substitutions must have the following properties:

- $\vec{y}/\vec{x}$ sends x_i to y_i for $i = 1, \ldots, n$, and leaves other names in $\mathcal{N}$ unchanged,
- p_y sends y to the canonical private name called p, and leaves other names in $\mathcal{Z}$ unchanged,
- $dom(\vec{y}/\vec{x}) \subseteq \mathcal{Z}$ and $dom(\ell), dom(r) \subseteq \mathcal{R}$,
- p_y, ℓ and r must be injective, even after being extended to total functions of type $\mathcal{H} \to \mathcal{H}$,
- $\ell(q) \neq r(q')$ for any private names q, q' obtained from p by applications of p_y, ℓ and r,
- all these substitutions σ much be surjective, in the sense that $dom(\sigma) \subseteq range(\sigma)$.[4]

The spare names $\mathcal{S}$ have been introduced solely to make the substitution $\vec{y}/\vec{x}$ surjective. 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 $i > n$. Abbreviate $\{\vec{y}/\vec{x}\}^{\mathcal{S}}$ by $\vec{y}/\vec{x}$ and $\{z/y\}^{\mathcal{S}}$ by z/y.

Choose $p \notin \mathcal{N}$, the *canonical private name*. Define the set of *proper* private names by $\mathcal{R}_0 := \{\varsigma p \mid \varsigma \in \{e, \ell, r\}^*\}$. They are obtained from p by attaching a sequence of tags e, ℓ and r as a left-superscript. I understand p to be the member of $\mathcal{R}_0$ with the empty sequence of tags. The function p_y simply adds a tag e to any proper private name; this is done by prepending e to its sequence of tags. As a result, p_y is injective on $\mathcal{R}_0 \cup \{y\}$. Likewise ℓ and r add tags ℓ and r, respectively.

[4] So a partial function $\sigma : \mathcal{H} \rightharpoonup \mathcal{H}$ is surjective iff its extension to a total function $\sigma : \mathcal{H} \to \mathcal{H}$ is surjective.

The only remaining problem is that p_y, ℓ and r are not yet surjective. To this end I define the set $\mathcal{R}$ of *private names* as $\{q^v \mid q \in \mathcal{R}_0 \wedge v \in \{\dagger\}^*\}$, with the understanding that $\mathcal{R}_0 \subseteq \mathcal{R}$.[5] For $\eta \in \{\ell, r, e\}$ and $y \in \mathcal{Z}$, let the surjective substitutions $\eta : \mathcal{R} \rightharpoonup \mathcal{R}$ and $p_y : \{y\} \cup \mathcal{R} \rightharpoonup \{y\} \cup \mathcal{R}$ be given by:

$$\begin{array}{lll} & & p_y(y) := p \\ \eta(\varsigma p) := {}^{\eta\varsigma}p & & p_y(p^\dagger) := y \\ \eta(\varsigma p^{v\dagger}) := {}^{\varsigma}p^v \text{ if } \varsigma \neq \eta\zeta & & p_y(u) := e(u) \text{ if } u \neq y, p^\dagger. \end{array}$$

These $\sigma : \mathcal{H} \rightharpoonup \mathcal{H}$ are injective, i.e., $x[\sigma] \neq y[\sigma]$ when $x \neq y$. My use of spare and private names is reminiscent of the locally nameless representation of variables as occurs for instance in [7].

The main result of this paper states the validity of the translation of Table 7, and thus that CCS_γ is at least as expressive as π_{IM}:

Theorem 9.1. For $P \in \mathrm{T}_{\pi_{\mathrm{IM}}}$ one has $\mathscr{T}(P) \mathrel{\dot{\leftrightarrow}} P$.

See the appendix for a proof.

Theorem 9.1 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 9.1 each π transition $P \xrightarrow{\tau} P'$ can be matched by a CCS_γ transition $\mathscr{T}(P) \xrightarrow{\tau} Q$ with $\mathscr{T}(P') \mathrel{\dot{\leftrightarrow}} Q$. Likewise, each CCS_γ transition $\mathscr{T}(P) \xrightarrow{\tau} Q$ can be matched by a π transition $P \xrightarrow{\tau} P'$ with $\mathscr{T}(P') \mathrel{\dot{\leftrightarrow}} Q$. 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.

Corollary 9.2. There exists an equivalence $\sim_E \subseteq \mathrel{\dot{\leftrightarrow}}$ on $\mathrm{T}_{\pi_{\mathrm{IM}}} \uplus \mathrm{T}_{\mathrm{CCS}_\gamma}$, the disjoint union of the π_{IM}- and CCS_γ-processes, such that $\mathscr{T}$ is valid up to $\sim_E$,

- the restriction of $\sim_E$ to $\mathrm{T}_{\pi_{\mathrm{IM}}}$ is the congruence closure of strong barbed bisimilarity on π_{IM}, i.e., the largest congruence for π_{IM} that is contained in $\mathrel{\dot{\leftrightarrow}}$, which is strong early congruence,
- and the restriction of $\sim_E$ to $\mathrm{T}_{\mathrm{CCS}_\gamma}$ is the congruence closure of $\mathrel{\dot{\leftrightarrow}}$ under translated contexts.

Proof. This is the application of [12, Theorem 4] to Theorem 9.1. □

Because of these properties, also on CCS_γ the equivalence $\sim_E$ could be called strong early congruence; it is however not a natural equivalence on CCS_γ when not considering the translation $\mathscr{T}$. By Corollary 9.2, any statement $P \sim_E Q$ stating that two π-calculus processes P and Q—possibly a specification and an implementation of a distributed system—are strongly early congruent, can be translated in a CCS_γ statement $\mathscr{T}(P) \sim_E \mathscr{T}(Q)$. This makes it possible to replay π-calculus verifications in CCS_γ.

10 THE IDEAS BEHIND THIS ENCODING AND THE PROOF OF ITS VALIDITY

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

[5] In [13] I use a prime instead of a dagger.

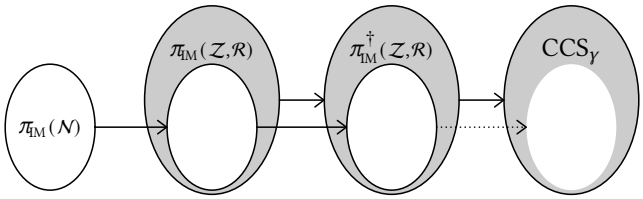


Fig. 2. Translation from the π-calculus with implicit matching to CCS_γ
Definitions of the intermediate languages $\pi_{\mathrm{IM}}(\mathcal{Z},\mathcal{R})$ and $\pi^{\dagger}_{\mathrm{IM}}(\mathcal{Z},\mathcal{R})$ are not provided here.

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:

(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, and its validity has been established in Section 6.
(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.

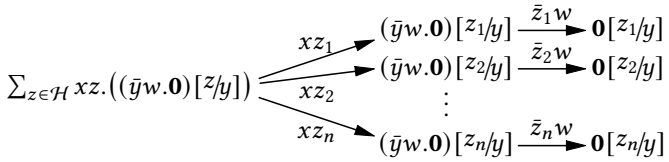
As indicated in Figure 2, my translation maps the π-calculus with implicit matching to a subset of CCS_γ. On that subset, π-calculus behaviour can be replayed faithfully, at least up to strong early congruence, the congruence closure of strong barbed bisimilarity. However, the interaction between a translated 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 π_{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 EXAMPLES

Example 11.1. The outgoing transitions of $x(y).\bar{y}w$ are

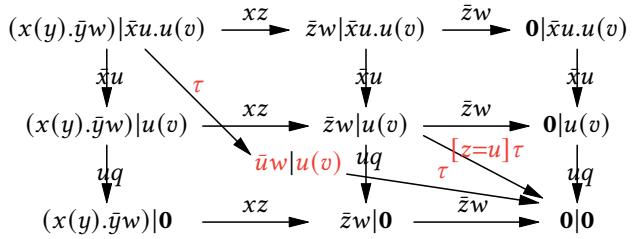
$$x(y).\bar{y}w \xrightarrow{xz_1} \bar{z}_1 w \xrightarrow{\bar{z}_1 w} \mathbf{0}, \quad x(y).\bar{y}w \xrightarrow{xz_2} \bar{z}_2 w \xrightarrow{\bar{z}_2 w} \mathbf{0}, \quad \ldots, \quad x(y).\bar{y}w \xrightarrow{xz_n} \bar{z}_n w \xrightarrow{\bar{z}_n w} \mathbf{0}\,.$$

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



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



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_γ.

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

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

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

Example 11.3. 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_γ is

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

Up to strong bisimilarity, its transition system is the same as that of P or $\mathscr{T}(P)$ from Example 11.2, 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 $\mathscr{T}(Q) \mathrel{\dot{\sim}} Q$, since private names generate no barbs.

Example 11.4. The process $(\nu x)(x(y)) \mid (\nu x)(\bar{x}u)$ has no outgoing transitions. Accordingly, its translation

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

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 11.5. 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 $\mathscr{T}\big((\nu y)\big(\bar{x}y.\bar{y}w\big)\big)\,[\ell] \xrightarrow{\bar{x}{}^{\ell}p} (\bar{y}w.\mathbf{0})[p_y][\ell]$. Since the substitution r used in the relabelling operator $[r]$ is surjective, there is a name s that is mapped to ${}^{\ell}p$, namely ${}^{\ell}p^{\dagger}$. Considering that $\mathscr{T}(x(u).u(v)) \xrightarrow{xs} \mathscr{T}(u(v))[s/u]$,

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

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

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

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 11.7. 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

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

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_{γ} 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.[6]

This example shows that spare names play a crucial role in intermediate states of CCS_γ-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.

12 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, 36]:

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

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_γ, as rooted weak bisimilarity [2], the weak congruence of [21, 22], is a congruence for CCS_γ but not for triggering. However, rooted branching bisimilarity [14] is a congruence for triggering [10].

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

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

Theorem 9.1 applies to this extended translation as well:

Theorem 12.1. For $P \in \mathrm{T}_\pi$ one has $\mathscr{T}(P) \leftrightarrows P$.

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

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

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

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 $A \mathrel{\dot{\sim}} Q$, and hence no translation of CCS in the π-calculus that is valid up to $\mathrel{\dot{\sim}}$.[7]

14 RELATED WORK

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

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

The main result of [3] (Theorem 5.3), stating the correctness of this encoding, says that $P \leftrightarrows_r Q$ iff $\mathcal{E}(P) \leftrightarrows_r \mathcal{E}(Q)$, for all π-processes P and Q. Here $\leftrightarrows_r$ is strong reduction bisimilarity—see Definition 8.1. 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 $P \leftrightarrows_r \mathcal{E}(P)$ for all π-processes P, i.e., that $\mathcal{E}$ is valid up to $\leftrightarrows_r$.

This result contradicts my Theorem 8.5 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 8.5, 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 $P \not\leftrightarrows_r \mathcal{E}(P)$. Using a trivial process Q such that $P \leftrightarrows_r Q \leftrightarrows_r \mathcal{E}(Q)$, 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_γ.

In [33], 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

[7] In [30] it was already mentioned, by reference to Pugliese [personal communication, 1997] that CCS relabelling operators cannot be encoded in the π-calculus.

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 $tau \rightarrow \text{CSP}[P]$, where $\rightarrow$ is CSP action prefixing and $\text{CSP}[P]$ 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 $x?z \rightarrow \text{CSP}[P\{z/y\}]$. 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, [33] works with translations $\text{CSP}[P]_{\kappa,\sigma}$, 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 $\text{CSP}[P|Q]$ the arguments P and Q may appear in the translated CSP process only in the shape $\text{CSP}[P]$ and $\text{CSP}[Q]$, not $\text{CSP}[P]_{\kappa,\sigma'}$ for new values of σ'.

As pointed out in [16, 31], 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_γ 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 [35]). Conversely, I observed that CCS (and thus certainly CCS_γ) cannot be encoded in the π-calculus.

I also showed that the upgrade of CCS to CCS_γ is necessary to capture the expressiveness of the π-calculus up to strong barbed bisimilarity. This does not rule out the possibility of a translation from the π-calculus to CCS that is valid up to some weaker equivalence, such as *weak barbed bisimilarity* [28, 35]. While I conjecture that such a translation does not exist, this paper leaves that question open.

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_γ. 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 $\dot{\leftrightarrow}$ 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 [9], 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 [26] 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^τ_R. This would however require work that goes well beyond what is presented here.

An alternative approach is to use a version of CCS_γ featuring a *choice quantifier* [19] instead of infinitary summation, a construct that looks remarkably like an infinite sum, but is as finite as any quantifier from predicate logic. A traditional 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 f from names to CCS relabelling operators and the variant $\sum_{z\in\mathcal{H}} E[f(z)]$ of choice quantification. Syntactically, this construct is a unary operator with argument E. Semantically it behaves the same as the infinite choice between all processes $E[f(z)]$ for $z \in \mathcal{H}$. My translation needs one instance of this construct, with $f(z)$ given by the relabelling function z/y. 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 [17], 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 the π-calculus, 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 [8] 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 [30] 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 [30] finds uniformity a reasonable criterion for encodings, because it guarantees that the translation maintains the degree of distribution of the system. In [32], 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 [15] 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 [11] and elsewhere. My encoding satisfies all other criteria of [15] (operational correspondence, divergence reflection and success sensitiveness).

REFERENCES

[1] Didier Austry and Gérard Boudol. 1984. Algèbre de processus et synchronisations. Theoretical Computer Science 30, 1 (1984), 91–131. https://doi.org/10.1016/0304-3975(84)90067-7

[2] Jos C.M. Baeten and W. Peter Weijland. 1990. *Process Algebra.* Cambridge University Press. https://doi.org/10.1017/CBO9780511624193

[3] Richard Banach and Franck van Breugel. 1998. Mobility and Modularity: expressing π-calculus in CCS. Preprint. http://theory.stanford.edu/~rvg/Pi.CCS.ext.abs.pdf

[4] Jan A. Bergstra and Jan Willem Klop. 1986. Algebra of communicating processes. In *Mathematics and Computer Science.* North-Holland, 89–138.

[5] Gérard Boudol. 1992. *Asynchrony and the π-calculus (Note).* Tech. Rep. 1702. INRIA.

[6] Stephen D. Brookes, Tony (C.A.R.) Hoare, and Bill (A.W.) Roscoe. 1984. A theory of communicating sequential processes. *J. ACM* 31, 3 (1984), 560–599. https://doi.org/10.1145/828.833

[7] Arthur Charguéraud. 2012. The Locally Nameless Representation. *Journal of Automated Reasoning* 49, 3 (2012), 363–408. https://doi.org/10.1007/s10817-011-9225-2

[8] Gian-Luigi Ferrari, Ugo Montanari, and Paola Quaglia. 1996. A Pi-Calculus with Explicit Substitutions. Theoretical Computer Science 168, 1 (1996), 53–103. https://doi.org/10.1016/S0304-3975(96)00063-1

[9] Rob J. van Glabbeek. 1994. On the expressiveness of ACP (extended abstract). In Proc. ACP'94 *(Workshops in Computing).* Springer, 188–217. https://doi.org/10.1007/978-1-4471-2120-6_8

[10] Rob J. van Glabbeek. 2011. On Cool Congruence Formats for Weak Bisimulations. Theoretical Computer Science 412, 28 (2011), 3283–3302. https://doi.org/10.1016/j.tcs.2011.02.036

[11] Rob J. van Glabbeek. 2012. Musings on Encodings and Expressiveness. In Proc. EXPRESS/SOS'12 *(*EPTCS, Vol. 89). Open Publishing Association, 81–98. https://doi.org/10.4204/EPTCS.89.7

[12] Rob J. van Glabbeek. 2018. A Theory of Encodings and Expressiveness. In Proc. FoSSaCS'18 *(*LNCS, Vol. 10803). Springer, 183–202. https://doi.org/10.1007/978-3-319-89366-2_10

[13] Rob J. van Glabbeek. 2022. Comparing the expressiveness of the π-calculus and CCS. https://arxiv.org/abs/2203.11519v1 Extended abstract in I. Sergey, editor: *Programming Languages and Systems:* Proceedings 31st *European Symposium on Programming,* ESOP'22; held as part of the *European Joint Conferences on Theory and Practice of Software,* ETAPS'22, Munich, Germany, April 2022, LNCS 13240, Springer, 2022, pp. 548–574, doi: 10.1007/978-3-030-99336-8_20.

[14] Rob J. van Glabbeek and W. Peter Weijland. 1996. Branching Time and Abstraction in Bisimulation Semantics. *J. ACM* 43, 3 (1996), 555–600. https://doi.org/10.1145/233551.233556

[15] Daniele Gorla. 2010. Towards a unified approach to encodability and separation results for process calculi. *Information and Computation* 208, 9 (2010), 1031–1053. https://doi.org/10.1016/j.ic.2010.05.002

[16] Daniele Gorla and Uwe Nestmann. 2016. Full abstraction for expressiveness: history, myths and facts. *Mathematical Structures in Computer Science* 26, 4 (2016), 639–654. https://doi.org/10.1017/S0960129514000279

[17] Jan Friso Groote and Mohammad Reza Mousavi. 2014. *Modeling and Analysis of Communicating Systems.* MIT Press.

[18] Matthew Hennessy and Huimin Lin. 1995. Symbolic Bisimulations. Theoretical Computer Science 138, 2 (1995), 353–389. https://doi.org/10.1016/0304-3975(94)00172-F

[19] Bas Luttik. 2003. On the expressiveness of choice quantification. *Annals of Pure and Applied Logic* 121, 1 (2003), 39–87. https://doi.org/10.1016/S0168-0072(02)00082-9

[20] Robin Milner. 1983. Calculi for synchrony and asynchrony. Theoretical Computer Science 25 (1983), 267–310. https://doi.org/10.1016/0304-3975(83)90114-7

[21] Robin Milner. 1989. *Communication and Concurrency.* Prentice Hall, Englewood Cliffs.

[22] Robin Milner. 1990. Operational and algebraic semantics of concurrent processes. In *Handbook of Theoretical Computer Science.* Elsevier Science Publishers B.V. (North-Holland), Chapter 19, 1201–1242.

[23] Robin Milner. 1992. Functions as Processes. *Mathematical Structures in Computer Science* 2, 2 (1992), 119–141. https://doi.org/10.1017/S0960129500001407

[24] Robin Milner. 1999. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press.

[25] Robin Milner, Joachim Parrow, and David Walker. 1992. A Calculus of Mobile Processes, I. *Information and Computation* 100 (1992), 1–40. https://doi.org/10.1016/0890-5401(92)90008-4

[26] Robin Milner, Joachim Parrow, and David Walker. 1992. A Calculus of Mobile Processes, II. *Information and Computation* 100 (1992), 41–77. https://doi.org/10.1016/0890-5401(92)90009-5

[27] Robin Milner, Joachim Parrow, and David Walker. 1993. Modal Logics for Mobile Processes. Theoretical Computer Science 114 (1993), 149–171. https://doi.org/10.1016/0304-3975(93)90156-N

[28] Robin Milner and Davide Sangiorgi. 1992. Barbed Bisimulation. In Proc. ICALP'92 *(*LNCS, Vol. 623). Springer, 685–695. https://doi.org/10.1007/3-540-55719-9_114

[29] Uwe Nestmann. 2006. Welcome to the Jungle: A Subjective Guide to Mobile Process Calculi. In Proc. CONCUR'06 *(*LNCS, Vol. 4137). Springer, 52–63. https://doi.org/10.1007/11817949_4

[30] Catuscia Palamidessi. 2003. Comparing the Expressive Power of the Synchronous and Asynchronous Pi-Calculi. *Mathematical Structures in Computer Science* 13, 5 (2003), 685–719. https://doi.org/10.1017/S0960129503004043

[31] Joachim Parrow. 2016. General conditions for full abstraction. *Mathematical Structures in Computer Science* 26, 4 (2016), 655–657. https://doi.org/10.1017/S0960129514000280

[32] Kirstin Peters, Uwe Nestmann, and Ursula Goltz. 2013. On Distributability in Process Calculi. In *Proc. ESOP'13 (*LNCS, Vol. 7792). Springer, 310–329. https://doi.org/10.1007/978-3-642-37036-6_18

[33] Bill (A.W.) Roscoe. 2010. CSP is Expressive Enough for π. In *Reflections on the Work of C.A.R. Hoare*. Springer, 371–404. https://doi.org/10.1007/978-1-84882-912-1_16

[34] Davide Sangiorgi. 1996. A Theory of Bisimulation for the pi-Calculus. *Acta Informatica* 33, 1 (1996), 69–97. https://doi.org/10.1007/s002360050036

[35] Davide Sangiorgi and David Walker. 2001. *The π-calculus: A Theory of Mobile Processes.* Cambridge University Press.

[36] Robert de Simone. 1985. Higher-level synchronising devices in Meije-SCCS. Theoretical Computer Science 37 (1985), 245–267. https://doi.org/10.1016/0304-3975(85)90093-3

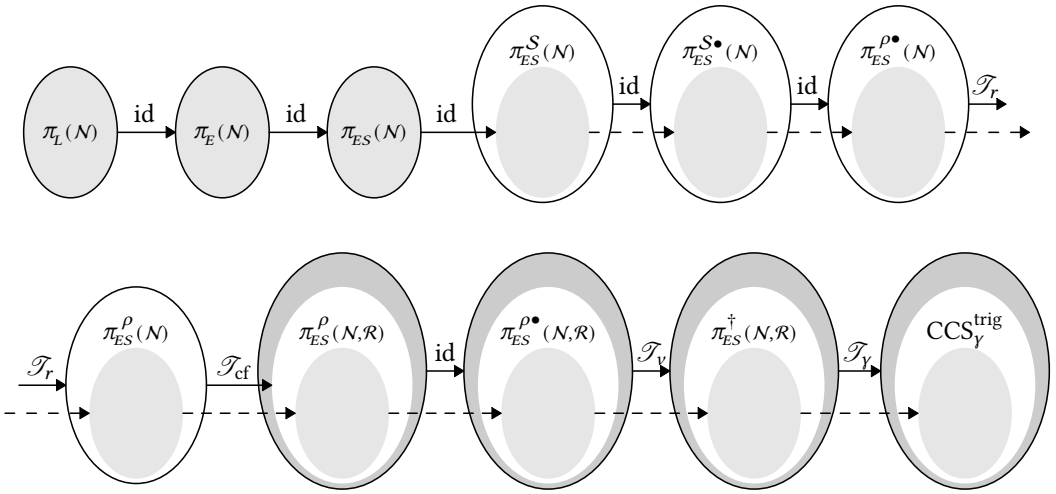


Fig. 3. Translation from the π-calculus to $\mathrm{CCS}_\gamma^{\mathrm{trig}}$

A PROOF OF THEOREM 12.1, INCLUDING THEOREM 9.1

As indicated in Figure 3, my translation from π to $\mathrm{CCS}_\gamma^{\mathrm{trig}}$ proceeds in ten steps. Section 9 presents the translation in one step: the composition of these constituent translations. Its decomposition in Figure 3 describes how this appendix proves its validity.

Each of the eleven languages in Figure 3 comprises syntax, determining what are the valid expressions or processes, a structural operational semantics generating an LTS, and a BTS extracted from the LTS in the way described in Section 4.

My translation starts from the π-calculus $\pi_L(\mathcal{N})$ with the late operational semantics, as defined in [26]. The argument $\mathcal{N}$ denotes the set of names taken as a parameter of the π-calculus. The first step is the identity mapping to $\pi_E(\mathcal{N})$, the calculus with the same syntax but the early operational semantics. The validity of this translation step is the statement that each $\pi_L(\mathcal{N})$-expression P is strongly barbed bisimilar with the same expression P, but now seen as a state in the LTS generated by the early operational semantics. As remarked in Section 6, this is an immediate consequence of Lemmas 6.2 and 6.4.

The second translation step likewise goes to the π-calculus with the early symbolic semantics. Its validity has been concluded at the end of Section 6.

The calculus $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$ is a variant of $\pi_{ES}(\mathcal{N})$ in which the substitutions that occur in the early symbolic operational rules **early-input** and **ide** are changed into surjective substitutions. This is achieved by extending the set of names from $\mathcal{N}$ to $\mathcal{Z} := \mathcal{N} \uplus \mathcal{S}$, for a countable set of *spare names* $\mathcal{S}$. In order to preserve the integrity of the calculus, this change forces new definitions of α-conversion, the free names of a process, and the application of a substitution to a process. These concepts differ from the old ones only when spare names are involved. The $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$-processes that employ names from $\mathcal{N}$ only form a subcalculus of $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$ that behaves just like $\pi_{ES}(\mathcal{N})$. This yields the third translation step.

The calculus $\pi_{ES}^{\rho\bullet}(\mathcal{N})$ is a variant of $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$ enriched with a CCS-style relabelling operator $[\sigma]$ for each surjective substitution σ. Moreover, all substitutions that are used in the operational semantics are replaced by relabelling operators. Section A.4 documents that the identity mapping is a valid translation from $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$ to $\pi_{ES}^{\rho\bullet}(\mathcal{N})$. This works only when using surjective substitutions, and that is the reason $\pi_{ES}^{\mathcal{S}}(\mathcal{N})$ appears in the previous translation step.

In $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ there is no room for rule **alpha**. Hence, before translating $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ into $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ I show that on $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ one can equally well use a version of the early symbolic operational semantics in which all α-conversion has been moved into stronger versions of rules **res** and **symb-open**. This can be seen as the fourth translation step.

The calculus $\pi^{\rho}_{ES}(\mathcal{N})$ is the variant of $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ in which agent identifiers may be called only with their own declared names as parameters, i.e. such that $\vec{y} = \vec{x}$ in rule **ide**. In Section A.6 I establish the validity of a translation from $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ to $\pi^{\rho}_{ES}(\mathcal{N})$ that turns each call $A(\vec{y})$ to an agent identifier with defining equation $A(\vec{x}) \stackrel{\text{def}}{=} P$ into the expression $A(\vec{x})[\vec{y}/\vec{x}]$.

In $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ the set of names employed is further extended with a collection $\mathcal{R}$ of *private names*. I write $\mathcal{N},\mathcal{R}$ instead of $\mathcal{N} \uplus \mathcal{R}$ to indicate, amongst others, that only the names in $\mathcal{Z} := \mathcal{N} \uplus \mathcal{S}$—the *public* ones—generate barbs. The interior white ellipse denotes the class of *clash-free* processes within $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$, defined in Section A.7. That section also presents the translation $\mathscr{T}_{\text{cf}}$ that turns each $\pi^{\rho}_{ES}(\mathcal{N})$ process into a clash-free $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ process.

Step 8, recorded in Section A.8, eliminates α-conversion from the operational semantics of $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$, and step 9, recorded in Section A.9, eliminates the restriction operator from the language by translating (sub)expressions $(\nu z)P$ into P. These steps does not preserve $\dot{\sim}$ for the language as a whole, but, since there are no $\mathcal{R}$-barbs, they do so on the sublanguage that arises as the image of the previous translation steps, namely on the clash-free processes in $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$.

After these steps, the resulting language $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$ is in De Simone format. Moreover, as shown in Section A.10, it is easily translated into $\text{CCS}^{\text{trig}}_{\gamma}$.

A different proof of the same result, Theorem 12.1, saying that $\mathscr{T}$ is a valid translation from $\pi_L(\mathcal{N})$ to $\text{CCS}^{\text{trig}}_{\gamma}$, up to strong barbed bisimilarity, appears in [13, Appendix 1]. That proof avoids step 3 from the current approach, which is the most laborious one, as well as step 4. As a result, that proof is 35% shorter. The remaining steps are delivered in the order 1–2–7–8–9–5–6–10. These steps, in that order, with step 6 deemed trivial, are the seven steps described in Section 10.

However, that proof appears less direct: its third step, making processes clash-free, introduces two auxiliary sets of names $\mathcal{B}$ and $\mathcal{D}$, as well as "ruthless substitutions", with an infinity of replicated agent identifiers. The ruthless substitutions are eliminated again after the sixth step, and the names from $\mathcal{B}$ and $\mathcal{D}$ at the very end. The present proof avoids those detours. Moreover, its intermediate languages make somewhat more sense in their own right, and appear better suited for potential further work on this topic.

A.1 Substitution

This section shows to what extend substitution preserves and reflects the behaviour of π-calculus processes. As behaviour I will use the labelled translation relation generated by the early symbolic operational semantics.

In the next section, I will need that the results of this section remain valid for an adapted version of the π-calculus, in which the meaning of the construct $x(y).P$ will be changed. Technically, this will manifest itself by

- the use of surjective substitutions $\{z/y\}^{\mathcal{S}}$ and $\{\vec{y}/\vec{x}\}^{\mathcal{S}}$ instead of $\{z/y\}$ and $\{\vec{y}/\vec{x}\}$ in rules **early-input** and **ide**,
- the use of a relation $\equiv^{\mathcal{S}}$ instead of $\equiv$ in rule **alpha**,
- the use of a function **fn** instead of fn in rules **par** and **e-s-close**,
- a different definition of $(x(y).P)\sigma$.

The function FN will be defined inductively by

$$\begin{array}{ll}
\textsc{fn}(\mathbf{0}) & := \emptyset \\
\textsc{fn}(M\tau.P) & := \mathrm{n}(M) \cup \textsc{fn}(P) \\
\textsc{fn}(M\bar{x}y.P) & := \mathrm{n}(M) \cup \{x,y\} \cup \textsc{fn}(P) \\
\textsc{fn}(Mx(y).P) & := \mathrm{n}(M) \cup \{x\} \cup \textsc{fn}(P)^{-}_{y} \\
\textsc{fn}((\nu y)P) & := \textsc{fn}(P)\backslash\{y\} \\
\textsc{fn}([x{=}y]P) & := \{x,y\} \cup \textsc{fn}(P) \\
\textsc{fn}(P|Q) & := \textsc{fn}(P) \cup \textsc{fn}(Q) \\
\textsc{fn}(P+Q) & := \textsc{fn}(P) \cup \textsc{fn}(Q) \\
\textsc{fn}(A(\vec{y})) & := \{y_1,\ldots,y_n\}.
\end{array}$$

where $\textsc{fn}(P)^{-}_{y}$ is yet to be defined. So FN differs from fn only when applied to processes $x(y).P$.

I will use the following definition of substitution:

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

where $P\sigma[y \mapsto z]$ is yet to be defined. Here z is chosen outside $\textsc{fn}((\nu y)P) \cup dom(\sigma) \cup range(\sigma)$; when $y \notin dom(\sigma) \cup range(\sigma)$ one always picks $z := y$. The original definition of substitution is retrieved by taking $P\sigma[y \mapsto z] := P\{z/y\}\sigma$.

To facilitate reuse, I record below the properties of substitutions, FN and $\equiv^{\mathcal{S}}$ that will be needed in my proofs.

$$\textsc{fn}(P\{\vec{y}/\vec{x}\}^{\mathcal{S}}) \subseteq \textsc{fn}(A(\vec{y})) \tag{4}$$

$$P \equiv^{\mathcal{S}} Q \Rightarrow \textsc{fn}(P) = \textsc{fn}(Q) \tag{5}$$

$$\textsc{fn}(P\{z/y\}^{\mathcal{S}}) \subseteq \textsc{fn}(x(y).P) \cup \{z\} \tag{6}$$

$$P\sigma[y \mapsto w]\{z[\sigma]/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}}\sigma \tag{7}$$

$$\textsc{fn}(P\sigma) = \{x[\sigma] \mid x \in \textsc{fn}(P)\} \tag{8}$$

$$P\{\vec{y}[\sigma]/\vec{x}\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{\vec{y}/\vec{x}\}^{\mathcal{S}}\sigma \tag{9}$$

$$((\nu y)P) \equiv^{\mathcal{S}} (\nu w)(P\{w/y\}) \tag{10}$$

$$(P\sigma_1)\sigma_2 \equiv^{\mathcal{S}} P(\sigma_2 \circ \sigma_1)^{8} \tag{11}$$

$$(\forall x \in \textsc{fn}(P).\, x[\sigma] = x[\sigma']) \Rightarrow P\sigma \equiv^{\mathcal{S}} P\sigma' \tag{12}$$

$$P \equiv^{\mathcal{S}} Q \Rightarrow P\sigma \equiv^{\mathcal{S}} Q\sigma \tag{13}$$

$$P\{w/y\}^{\mathcal{S}}\{z/w\} \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}} \tag{14}$$

$$P\epsilon \equiv^{\mathcal{S}} P \tag{15}$$

$$U \equiv^{\mathcal{S}} (x(y).P)\sigma \Rightarrow U = x[\sigma](w).V \text{ with } V \equiv^{\mathcal{S}} P\sigma[y \mapsto w] \tag{16}$$

$$U \equiv^{\mathcal{S}} (\nu y)P \Rightarrow U = (\nu w)V \text{ with } V \equiv^{\mathcal{S}} P\{y/w\} \tag{17}$$

[8]For substitutions σ_1 and σ_2, their composition $\sigma_2 \circ \sigma_1$ is defined by $dom(\sigma_2 \circ \sigma_1) = dom(\sigma_1) \cup dom(\sigma_2)$ and $x[\sigma_2 \circ \sigma_1] = x[\sigma_1][\sigma_2]$.

Here $A(\vec{x}) \stackrel{\text{def}}{=} P$ in (4) and (9), $w \in \mathcal{N} \setminus \text{FN}((\nu y)P)$ in (10), $y \in \mathcal{N}$ and $w \in \text{FN}(x(y)P)$ in (14), $w \in \mathcal{N} \setminus (\text{FN}((\nu y)P) \cup dom(\sigma) \cup range(\sigma))$ in (7), and ϵ is the empty substitution. As is easy to check, these properties hold for the original π-calculus, that is, when using $\{z/y\}$ and $\{\vec{y}/\vec{x}\}$ for $\{z/y\}^{\mathcal{S}}$ and $\{\vec{y}/\vec{x}\}^{\mathcal{S}}$, $\equiv$ for $\equiv^{\mathcal{S}}$, fn for FN (i.e., $\text{FN}(P)^{-}_{y} := \text{FN}(P)\setminus\{y\}$), and $P\sigma[y \mapsto z] := P\{z/y\}\sigma$. In the next section I will show that these properties also hold for the adapted version of the π-calculus introduced there. Here I will use them as axioms.

First I state two standard properties of names in transitions.

Define the *input arguments* $\iota(\alpha)$ of an action α by

$$\iota(M\bar{x}(y)) = \iota(M\bar{x}y) = \iota(M\tau) := \emptyset \quad \text{and} \quad \iota(Mxz) = \{z\}.$$

LEMMA A.1. If $P \xrightarrow{\alpha} Q$ then $\text{n}(\alpha)\setminus(\iota(\alpha) \cup \text{bn}(\alpha)) \subseteq \text{FN}(P)$.

PROOF. A trivial induction on the inference of $P \xrightarrow{\alpha} Q$, using (4) and (5) when $P \xrightarrow{\alpha} Q$ is derived by IDE or ALPHA. □

LEMMA A.2. If $P \xrightarrow{\alpha} Q$ then $\text{FN}(Q) \subseteq \text{FN}(P) \cup \iota(\alpha) \cup \text{bn}(\alpha)$.

PROOF. A trivial induction on the inference of $P \xrightarrow{\alpha} Q$, using (6) when $P \xrightarrow{\alpha} Q$ is derived by EARLY INPUT, Lemma A.1 when $P \xrightarrow{\alpha} Q$ is derived by E-S-COM, and (4) and (5) when $P \xrightarrow{\alpha} Q$ is derived by IDE or ALPHA. □

Definition A.3. The *depth* of an interference of a transition (from the rules in the operational semantics) is the length of the longest path in the proof tree of that inference. The *α-depth* is defined likewise, but not counting applications of rule ALPHA.

Following [26], in the following lemmas the phrase

if $P \xrightarrow{\alpha} P'$ then *equally* $Q \xrightarrow{\alpha} Q'$

means that if $P \xrightarrow{\alpha} P'$ may be inferred then so, by an inference of the same α-depth, may be $Q \xrightarrow{\alpha} Q'$.

For $\sigma : \mathcal{N} \rightharpoonup \mathcal{N}$ a substitution and α an action $M\tau$, Mxy, $M\bar{x}y$ or $M\bar{x}(z)$, write $\alpha[\sigma]$ for $M[\sigma]\tau$, $M[\sigma]x[\sigma]y[\sigma]$, $M[\sigma]\bar{x}[\sigma]y[\sigma]$ and $M[\sigma]\bar{x}[\sigma](z[\sigma])$, respectively.

Lemma A.8 below shows how substitutions preserve behaviour. On the way to obtain it, I start with the special case of input and (bound) output transitions.

LEMMA A.4. If $R \xrightarrow{\alpha} R'$, with α not of the form $M\tau$, and σ is a substitution with $\text{bn}(\alpha[\sigma]) \cap \text{FN}(R\sigma) = \emptyset$, then equally $R\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} R'\sigma$.

PROOF. With induction on the inference of $R \xrightarrow{\alpha} R'$.

- Suppose $R \xrightarrow{\alpha} R'$ is derived by rule EARLY-INPUT. Then $R = Mx(y).P$, $\alpha = Mxz$ and $R' = P\{z/y\}^{\mathcal{S}}$. So $R\sigma = M[\sigma]x[\sigma](w).(P\sigma[y \mapsto w]) \wedge \alpha[\sigma] = M[\sigma]x[\sigma]z[\sigma]$ where w is chosen outside $\text{FN}((\nu y)P) \cup dom(\sigma) \cup range(\sigma)$. By rule EARLY-INPUT and (7)
$$R\sigma \xrightarrow{\alpha[\sigma]} P\sigma[y \mapsto w]\{z[\sigma]/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}}\sigma = R'\sigma.$$
- The case that $R \xrightarrow{\alpha} R'$ is derived by OUTPUT is trivial.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by rule IDE. Then $R = A(\vec{y})$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. By rule IDE, $P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha} R'$. Since $\text{FN}(P\{\vec{y}/\vec{x}\}^{\mathcal{S}}\sigma) \subseteq \text{FN}(R\sigma)$ by (4) and (8), $\text{bn}(\alpha[\sigma]) \cap \text{FN}(P\{\vec{y}/\vec{x}\}^{\mathcal{S}}\sigma) = \emptyset$. Hence $P\{\vec{y}/\vec{x}\}^{\mathcal{S}}\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} R'\sigma$, by induction. By Property (9), $P\{\vec{y}[\sigma]/\vec{x}\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{\vec{y}/\vec{x}\}^{\mathcal{S}}\sigma$. Therefore $P\{\vec{y}[\sigma]/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} R'\sigma$, by rule ALPHA. Thus $R\sigma = A(\vec{y}[\sigma]) \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} R'\sigma$ by IDE.
- The cases that $R \xrightarrow{\alpha} R'$ is derived by rule SUM, SYMB-MATCH or ALPHA (using (5)) are trivial.

- Suppose $R \xrightarrow{\alpha} R'$ is derived by **PAR**. Then $R = P|Q$, $P \xrightarrow{\alpha} P'$ and $R' = P'|Q$. Since $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{FN}(R\sigma) = \emptyset$, $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{FN}(P\sigma) = \emptyset$. So by induction $P\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} P'\sigma$. Moreover $R\sigma = P\sigma|Q\sigma$ and $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{FN}(Q\sigma) = \emptyset$. So $R\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} P'\sigma|Q\sigma = R'\sigma$ by rule **PAR**.
- Let $R \xrightarrow{\alpha} R'$ be derived by rule **RES**. Then $R = (\nu y)P$, $P \xrightarrow{\alpha} P'$, $y \notin \mathrm{n}(\alpha)$ and $R' = (\nu y)P'$. Pick a w outside

$$\mathrm{FN}((\nu y)P') \cup \mathrm{FN}((\nu y)P) \cup dom(\sigma) \cup range(\sigma) \cup \mathrm{n}(\alpha[\sigma]).$$

  Then $R\sigma \equiv^{\mathcal{S}} ((\nu w)(P\{w/y\}))\sigma = (\nu w)(P\{w/y\}\sigma)$ by (10) and (13). Since $w \notin \mathrm{bn}(\alpha[\sigma])$ and $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{FN}(R\sigma) = \emptyset$, also $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{FN}(P\{y/w\}\sigma) = \emptyset$ by (5). So by induction, using (11) and the substitution $\sigma \circ \{w/y\}$,

$$P\{y/w\}\sigma \equiv^{\mathcal{S}} \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} P'\{y/w\}\sigma.$$

  Thus $R\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} (\nu w)(P'\{y/w\}\sigma)$ by **RES** and **ALPHA**. Furthermore, $(\nu w)(P'\{y/w\}\sigma) \equiv^{\mathcal{S}} R'\sigma$ by (10) and (13).
- Let $R \xrightarrow{M\bar{x}(y)} R'$ be derived by rule **SYMB-OPEN**. Then $R = (\nu y)P$, $P \xrightarrow{M\bar{x}y} R'$, $y \neq x$ and $y \notin \mathrm{n}(M)$. Now $R\sigma = (\nu z)(P\{z/y\}\sigma)$ for some $z \notin \mathrm{FN}((\nu y)P) \cup dom(\sigma) \cup range(\sigma)$. Since $w := y[\sigma] \notin \mathrm{FN}(R\sigma)$ (the side-condition of this lemma), $R\sigma \equiv^{\mathcal{S}} (\nu w)(P\{z/y\}\sigma\{w/z\})$ by (10). Note that $u[\{w/z\} \circ \sigma \circ \{z/y\}] = u[\sigma]$ for $u \in \mathrm{FN}(P)$. Hence $P\{z/y\}\sigma\{w/z\} \equiv^{\mathcal{S}} P\sigma$ by (11) and (12). By induction,

$$P\sigma \xrightarrow{M[\sigma]\bar{x}[\sigma]w} \equiv^{\mathcal{S}} R'\sigma.$$

  By Lemma A.1, $\mathrm{n}(M) \cup \{x, y\} \subseteq \mathrm{FN}(P)$. Hence $\mathrm{n}(M) \cup \{x\} \subseteq \mathrm{FN}(R) = \mathrm{FN}(P)\backslash\{y\}$, using that $y \notin \mathrm{n}(M) \cup \{x\}$. Thus $\mathrm{n}(M[\sigma]) \cup \{x[\sigma]\} \subseteq \mathrm{FN}(R\sigma)$ by (8), and therefore $w \notin \mathrm{n}(M[\sigma]) \cup \{x[\sigma]\}$. Thus, by **SYMB-OPEN** and **ALPHA**,

$$R\sigma \equiv^{\mathcal{S}} (\nu w)(P\sigma) \xrightarrow{M[\sigma]\bar{x}[\sigma](w)} \equiv^{\mathcal{S}} R'\sigma. \qquad \square$$

Before generalising the above result to include the case $\alpha = M\tau$, I cover three more helpful standard properties.

LEMMA A.5. If $R \xrightarrow{Mxz} R_z$ and $w \notin \mathrm{FN}(R)$ then equally $R \xrightarrow{Mxw} R_w$ for some process R_w with $R_z \equiv^{\mathcal{S}} R_w\{z/w\}$.

PROOF. With induction on the inference of $P \xrightarrow{Mxz} P_z$.

- Suppose $R \xrightarrow{Mxz} R_z$ is derived by rule **EARLY-INPUT**. Then $R = Mx(y).P$ and $R_z = P\{z/y\}^{\mathcal{S}}$. Moreover, $R \xrightarrow{Mxw} R_w := P\{w/y\}^{\mathcal{S}}$. By (14), $R_z \equiv^{\mathcal{S}} R_w\{z/w\}$.
- Suppose $R \xrightarrow{Mxz} R_z$ is derived by **PAR**. Then $R = P|Q$, $P \xrightarrow{Mxz} P_z$ and $R_z = P_z|Q$. So by induction there is a process P_w such that $P \xrightarrow{Mxw} P_w$ and $P_z \equiv^{\mathcal{S}} P_w\{z/w\}$. Thus $R \xrightarrow{Mxw} P_w|Q$ by **PAR**. Moreover, $(P_w|Q)\{z/w\} = P_w\{z/w\}|Q\{z/w\} \equiv^{\mathcal{S}} P_z|Q = R_z$ by (12) and (15), using that $w \notin \mathrm{FN}(Q)$.
- Let $R \xrightarrow{Mxz} R_z$ be derived by rule **RES**. Then $R = (\nu y)P$, $P \xrightarrow{Mxz} P_z$, $y \notin \mathrm{n}(M) \cup \{x, z\}$ and $R_z = (\nu y)P_z$. Pick a $v \notin \mathrm{FN}(P) \cup \mathrm{n}(M) \cup \{x, y\}$. By induction there is a process P_v such that $P \xrightarrow{Mxv} P_v$ and $P_z \equiv^{\mathcal{S}} P_v\{z/v\}$. So $R \xrightarrow{Mxv} (\nu y)P_v$ by **RES**. By (12), (15) and Lemma A.4,

$$R \equiv^{\mathcal{S}} R\{w/v\} \xrightarrow{Mxw} R_w \equiv^{\mathcal{S}} ((\nu y)P_v)\{w/v\}$$

  for some R_w. Hence $R \xrightarrow{Mxw} R_w$ by rule **ALPHA**.

By Lemma A.2 $\text{FN}(P_v)\setminus\{y,v\}\subseteq\text{FN}(P)\setminus\{y\}=\text{FN}(R)\not\ni w$. Pick $u\notin\text{FN}(P_v)\cup\text{FN}(P_v\{z/v\})\cup\{v,w\}$. Then

$$\begin{array}{c}R_w\{z/w\}\equiv^{\mathcal{S}}((vy)P_v)\{w/v\}\{z/w\}\equiv^{\mathcal{S}}\\((vu)P_v\{u/v\})\{w/v\}\{z/w\}\equiv^{\mathcal{S}}(vu)(P_v\{u/y\}\{w/v\}\{z/w\})\\\equiv^{\mathcal{S}}(vu)(P_v\{z/v\}\{u/y\})\equiv^{\mathcal{S}}(vy)(P_v\{z/v\})\equiv^{\mathcal{S}}R_z\end{array}$$

by (10)–(13), using that $v\neq y\neq z$ and $w\neq u\neq v$.

- The other four cases are trivial, using (4) and (5) when $P\xrightarrow{\alpha}Q$ is derived by **IDE** or **ALPHA**. □

LEMMA A.6. If $R\xrightarrow{Mxw}R_w$ with $w\notin\text{FN}(R)$, and z a name, then equally $R\xrightarrow{Mxz}R_z$ for some R_z with $R_z\equiv^{\mathcal{S}}R_w\{z/w\}$.

PROOF. With induction on the inference of $R\xrightarrow{Mxw}R_w$.

- Let $R\xrightarrow{Mxw}R_w$ be derived by rule **RES**. Then $R=(vy)P$, $P\xrightarrow{Mxw}P_w$, $y\notin\text{n}(M)\cup\{x,w\}$ and $R_w=(vy)P_w$. Pick a $v\notin\text{FN}(P)\cup\text{FN}(P_w)\cup\text{n}(M)\cup\{x,y\}$. By induction there is a process P_v with $P\xrightarrow{Mxv}P_v$ and $P_v\equiv^{\mathcal{S}}P_w\{z/w\}$. So $R\xrightarrow{Mxv}(vy)P_v$ by **RES**. By (12), (15) and Lemma A.4,

$$R\equiv^{\mathcal{S}}R\{z/v\}\xrightarrow{Mxz}R_z\equiv^{\mathcal{S}}((vy)P_v)\{z/v\}$$

for some R_z. Hence $R\xrightarrow{Mxz}R_z$ by rule **ALPHA**. Pick $u\notin\text{FN}(P_w)\cup\text{FN}(P_v)\cup\{z,v,w\}$. Then

$$\begin{array}{c}R_w\{z/w\}\equiv^{\mathcal{S}}((vy)P_w)\{z/w\}\equiv^{\mathcal{S}}\\((vu)P_w\{u/y\})\{z/w\}\equiv^{\mathcal{S}}((vu)P_w\{u/y\}\{z/w\})\equiv^{\mathcal{S}}\\(vu)(P_w\{z/w\}\{u/y\}\{z/v\})\equiv^{\mathcal{S}}(vu)(P_v\{u/y\}\{z/v\})\\\equiv^{\mathcal{S}}(vu)(P_v\{u/y\})\{z/v\}\equiv^{\mathcal{S}}((vy)P_v)\{z/v\}\equiv^{\mathcal{S}}R_z\end{array}$$

by (10)–(13), using that $v\neq z\neq y\neq w\neq u\neq v$.

- All other cases proceed as in the proof of Lemma A.5. □

The above proofs cater both to the full π-calculus and to π_{IM}, namely by considering explicitly the input prefix $Mx(y)$. Henceforth I focus on the full π-calculus, and drop the M from the input prefix. It can always be retrieved via an application of **SYMB-MATCH**.

LEMMA A.7. If $R\xrightarrow{Mx(z)}R'$ and $w\notin\text{FN}(R)$ then equally $R\xrightarrow{Mx(w)}\equiv^{\mathcal{S}}R'\{w/z\}$.

PROOF. With induction on the inference of $R\xrightarrow{Mx(z)}R'$.

- Let $R\xrightarrow{M\bar{x}(z)}R'$ be derived by rule **SYMB-OPEN**. Then $R=(vz)P$, $P\xrightarrow{M\bar{x}z}R'$, $z\neq x$ and $z\notin\text{n}(M)$. By Lemma A.4, $P\{w/z\}\xrightarrow{M\bar{x}w}\equiv^{\mathcal{S}}R'\{w/z\}$. By Lemma A.1, $\text{n}(M)\cup\{x\}\subseteq\text{FN}(R)$. Hence $w\notin\text{n}(M)\cup\{x\}$. By (10), $R\equiv^{\mathcal{S}}(vw)(P\{w/z\})$, so rules **SYMB-OPEN** and **ALPHA** yield $R\xrightarrow{M\bar{x}(w)}\equiv^{\mathcal{S}}R'\{w/z\}$.
- Let $R\xrightarrow{M\bar{x}(z)}R'$ be derived by rule **RES**. Then $R=(vy)P$, $P\xrightarrow{M\bar{x}(z)}P'$, $y\notin\text{n}(M)\cup\{x,z\}$ and $R'=(vy)P'$. Pick a $v\notin\{y\}\cup\text{FN}(P)\cup\text{FN}(P')\cup\text{n}(M)\cup\{x\}$. By induction $P\xrightarrow{M\bar{x}(v)}\equiv^{\mathcal{S}}P'\{v/z\}$. So $R\xrightarrow{M\bar{x}(v)}\equiv^{\mathcal{S}}(vy)(P'\{v/z\})$ by **RES**. By (12), (15) and Lemma A.4,

$$R\equiv^{\mathcal{S}}R\{w/v\}\xrightarrow{M\bar{x}(w)}\equiv^{\mathcal{S}}((vy)(P'\{v/z\}))\{w/v\}.$$

Pick a $u\notin\text{FN}(P')\cup\text{FN}(P'\{v/z\})\cup\{z,v,w\}$. Then

$$\begin{array}{c}((vy)(P'\{v/z\}))\{w/v\}\equiv^{\mathcal{S}}(vu)(P'\{v/z\}\{u/y\}\{w/v\})\equiv^{\mathcal{S}}\\(vu)(P'\{u/y\}\{w/z\})\equiv^{\mathcal{S}}((vy)P')\{w/z\}=R'\{w/z\},\end{array}$$

by (10)–(13), using that $\text{FN}(P')\not\ni v\neq y\neq z\neq u\neq v$. By rule **ALPHA**, $R\xrightarrow{M\bar{x}(w)}\equiv^{\mathcal{S}}R'\{w/z\}$.

- The other five cases are trivial. □

Lemma A.8. If $R \xrightarrow{\alpha} R'$ and σ is a substitution with $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{fn}(R\sigma) = \emptyset$ then equally $R\sigma \xrightarrow{\alpha[\sigma]} \equiv^{\mathcal{S}} R'\sigma$.

Proof. With induction on the inference of $R \xrightarrow{\alpha} R'$.

- The case that $R \xrightarrow{\alpha} R'$ is derived by rule **tau** is trivial.
- The cases that $R \xrightarrow{\alpha} R'$ is derived by rule **output**, **early-input**, **sum**, **symb-match**, **ide**, **par**, **res**, **symb-open** or **alpha** are already covered by Lemma A.4.
- The case that $R \xrightarrow{\alpha} R'$ is derived by **e-s-com** is trivial.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by **e-s-close**. Then $R = P|Q$, $P \xrightarrow{M\bar{x}(z)} P'$, $Q \xrightarrow{N\bar{v}z} Q'$, $z \notin \mathrm{fn}(Q)$, $\alpha = [x{=}v]MN\tau$ and $R' = (\nu z)(P'|Q')$. Pick $w \notin \mathrm{fn}(R) \cup \mathrm{fn}(R') \cup dom(\sigma) \cup range(\sigma)$. Now $P \xrightarrow{M\bar{x}(w)} \equiv^{\mathcal{S}} P'\{w/z\}$ and $Q \xrightarrow{N\bar{v}w} \equiv^{\mathcal{S}} Q'\{w/z\}$ by Lemmas A.7 and A.6. Thus $P\sigma \xrightarrow{M[\sigma]\bar{x}[\sigma](w)} \equiv^{\mathcal{S}} P'\{w/z\}\sigma$, $Q\sigma \xrightarrow{N[\sigma]\bar{v}[\sigma]w} \equiv^{\mathcal{S}} Q'\{w/z\}\sigma$ by Lemma A.4, using that $w[\sigma] = w \notin \mathrm{fn}(P\sigma)$, which follows by (8) from $w \notin \mathrm{fn}(P) \cup range(\sigma)$. Hence

$$R\sigma \xrightarrow{[x[\sigma]=v[\sigma]]M[\sigma]N[\sigma]\tau} (\nu w)((P'|Q')\{w/z\}\sigma)$$

by rule **e-s-close**. As $w \notin \mathrm{fn}((\nu z)(P'|Q'))$, by (10),

$$R'\sigma \equiv^{\mathcal{S}} (\nu w)((P'|Q')\{w/z\}\sigma). \qquad \square$$

Next I would have liked to show that substitutions also reflect behaviour, in the sense that, possibly under the same side condition as in Lemma A.8, $R\sigma \xrightarrow{\beta} U$ implies $R \xrightarrow{\alpha} R'$ for some α and R' with $\alpha[\sigma] = \beta$ and $R'\sigma \equiv^{\mathcal{S}} U$. Unfortunately, this property does not hold.

Example A.9. $(x(y).P)\{w/z\} \xrightarrow{xz} P\{w/z\}\{z/y\}$, yet there is no α with $\alpha[\{w/z\}] = xz$.

In order to rescue the reflection property, I restrict attention to surjective substitutions.

Definition A.10. A substitution $\sigma\colon \mathcal{N} \rightharpoonup \mathcal{N}$ is called *surjective* if $dom(\sigma) \subseteq range(\sigma)$.

Using rule **alpha**, the below reflection lemma can equally well be formulated with $R\sigma \xrightarrow{\beta} U'$ instead of $R\sigma \equiv^{\mathcal{S}} U \xrightarrow{\beta} U'$. However, its inductive proof requires the form stated.

Lemma A.11. If $R\sigma \equiv^{\mathcal{S}} U \xrightarrow{\beta} U'$ and σ is a surjective substitution, then equally $R \xrightarrow{\alpha} R'$ for some α and R' with $\alpha[\sigma] = \beta$ and $R'\sigma \equiv^{\mathcal{S}} U'$.

Proof. With induction on the α-depth of the inference of $U \xrightarrow{\beta} U'$, with a nested induction on the number of applications of rule **alpha** at the end of that inference.

- The cases that $U \xrightarrow{\beta} U'$ is derived by rule **tau** or **output** are trivial.
- Suppose $U \xrightarrow{\beta} U'$ is derived by **early-input**. Then $R = x(y).P$ and $U = x[\sigma](w).V$ with $V \equiv^{\mathcal{S}} (P\sigma[y \mapsto w])$ by (16). So $\beta = x[\sigma]v$ and $U' = V\{v/w\}^{\mathcal{S}}$. Since σ is surjective, there is a z with $z[\sigma] = v$. By **early-input** $R \xrightarrow{\alpha} R'$ with $\alpha = Mxz$ and $R' = P\{z/y\}^{\mathcal{S}}$. Now $\alpha[\sigma] = \beta$ and $R'\sigma = P\{z/y\}^{\mathcal{S}}\sigma \equiv^{\mathcal{S}} P\sigma[y \mapsto w]\{z[\sigma]/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} V\{v/w\}^{\mathcal{S}} = U'$, using (7) and (12).
- Suppose $U \xrightarrow{\beta} U'$ is derived by **ide**. Then $R = A(\vec{y})$ and $U = R\sigma = A(\vec{y}[\sigma])$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $P\{\vec{y}[\sigma]/\vec{x}\} \xrightarrow{\beta} U'$. Since $P\{\vec{y}[\sigma]/\vec{x}\} \equiv^{\mathcal{S}} P\{\vec{y}/\vec{x}\}\sigma$ by (9), $P\{\vec{y}/\vec{x}\}\sigma \xrightarrow{\beta} U'$ by **alpha**. Moreover, the inference of $P\{\vec{y}/\vec{x}\}\sigma \xrightarrow{\beta} U'$ has the same α-depth as that of $P\{\vec{y}[\sigma]/\vec{x}\} \xrightarrow{\beta} U'$, which is smaller than the α-depth of the inference of $U \xrightarrow{\beta} U'$. Hence, by induction, $P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha} R'$ for some α and R' with $\alpha[\sigma] = \beta$ and $R'\sigma \equiv^{\mathcal{S}} U'$. By rule **ide** $R \xrightarrow{\alpha} R'$.
- The cases that $U \xrightarrow{\beta} U'$ is derived by rule **sum** or **symb-match** are trivial.

- Suppose $U \xrightarrow{\beta} U'$ is derived by **par**. Then $R = P|Q$, $R\sigma = P\sigma|Q\sigma$, $U = V|W$, $P\sigma \equiv^{\mathcal{S}} V \xrightarrow{\beta} V'$, $Q\sigma \equiv^{\mathcal{S}} W$, $\mathrm{bn}(\beta) \cap \mathrm{fn}(W) = \emptyset$ and $U' = V'|W$. By induction $P \xrightarrow{\alpha} P'$ for some α and P' with $\alpha[\sigma] = \beta$ and $P'\sigma \equiv^{\mathcal{S}} V'$. By (5) $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{fn}(Q\sigma) = \emptyset$, so by (8) $\mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$. Consequently, $R = P|Q \xrightarrow{\alpha} P'|Q$ by rule **par**, and $(P'|Q)\sigma \equiv^{\mathcal{S}} V'|W = U'$.
- Suppose $U \xrightarrow{\beta} U'$ is derived by **e-s-com**. In that case $R = P|Q$, $R\sigma = P\sigma|Q\sigma$, $U = V|W$, $\beta = [x{=}v]MN\tau$, $P\sigma \equiv^{\mathcal{S}} V \xrightarrow{M\bar{x}y} V'$, $Q\sigma \equiv^{\mathcal{S}} W \xrightarrow{Nvy} W'$ and $U' = V'|W'$. By induction, there are matching sequences K, L with $K[\sigma] = M$ and $L[\sigma] = N$, names q, r, z, u with $q[\sigma] = x$, $r[\sigma] = v$, $z[\sigma] = y$ and $u[\sigma] = y$, and processes P' and Q' with $P'\sigma \equiv^{\mathcal{S}} V'$ and $Q'\sigma \equiv^{\mathcal{S}} W'$, such that $P \xrightarrow{K\bar{q}z} P'$ and $Q \xrightarrow{Lru} Q'$.
  Pick $w \notin \mathrm{fn}(Q)$. By Lemma A.5 there is a process P_w such that $Q \xrightarrow{Lrw} Q_w$ and $Q' \equiv^{\mathcal{S}} Q_w\{u/w\}$. By Lemma A.6 there is a P_z such that $Q \xrightarrow{Lrz} Q_z$ and $Q_z \equiv^{\mathcal{S}} Q_w\{z/w\}$. By rule **e-s-com** $R \xrightarrow{[q=r]KL\tau} R' := P'|Q_z$. By (11)–(13) $Q'\sigma \equiv^{\mathcal{S}} Q_z\sigma$. So $([q{=}r]KL\tau)[\sigma] = [x{=}v]MN\tau$ and $R'\sigma = (P'|Q_z)\sigma \equiv^{\mathcal{S}} V'|W' = U'$.
- Suppose $U \xrightarrow{\beta} U'$ is derived by rule **e-s-close**. Then $R = P|Q$, $R\sigma = P\sigma|Q\sigma$, $U = V|W$, $\beta = [x{=}v]MN\tau$, $P\sigma \equiv^{\mathcal{S}} V \xrightarrow{M\bar{x}(z)} V'$, $Q\sigma \equiv^{\mathcal{S}} W \xrightarrow{Nvz} W'$ and $U' = (\nu z)(V'|W')$ for some $z \notin \mathrm{fn}(W)$. Pick $w \notin \mathrm{fn}(U) \cup dom(\sigma) \cup range(\sigma)$. By (5) $w \notin \mathrm{fn}(W) = \mathrm{fn}(Q\sigma)$ so $w \notin \mathrm{fn}(Q)$ by (8). Now $V \xrightarrow{M\bar{x}(w)} \equiv^{\mathcal{S}} V'\{w/z\}$ and $W \xrightarrow{Nvw} \equiv^{\mathcal{S}} W'\{w/z\}$ by Lemmas A.7 and A.6. Moreover, the inferences of these transitions have a smaller α-depth than that of $U \xrightarrow{\beta} U'$. By induction, using that $w \notin dom(\sigma) \cup range(\sigma)$, there are matching sequences K, L with $K[\sigma] = M$ and $L[\sigma] = N$, names q, r with $q[\sigma] = x$ and $r[\sigma] = v$, and processes P' and Q' with $P'\sigma \equiv^{\mathcal{S}} V'\{w/z\}$ and $Q'\sigma \equiv^{\mathcal{S}} W'\{w/z\}$, such that $P \xrightarrow{K\bar{q}(w)} P'$ and $Q \xrightarrow{Lrw} Q'$. Therefore $R \xrightarrow{[q=r]KL\tau} (\nu w)(P'|Q')$ by **e-s-com**. Moreover, $([q{=}r]KL\tau)[\sigma] = \beta$. By Lemma A.2, $\mathrm{fn}(U') \subseteq \mathrm{fn}(U) \not\ni w$. So by (10) $((\nu w)(P'|Q'))\sigma = (\nu w)(P'\sigma|Q'\sigma) \equiv^{\mathcal{S}} (\nu w)(V'\{w/z\}|W'\{w/z\}) \equiv^{\mathcal{S}} U'$.
- Suppose $U \xrightarrow{\beta} U'$ is derived by **res**. Then $R = (\nu y)P$ and $R\sigma = (\nu z)(P\{z/y\}\sigma)$ for some $z \notin \mathrm{fn}((\nu y)P) \cup dom(\sigma) \cup range(\sigma)$. By (17), $U = (\nu w)V$ with $V \equiv^{\mathcal{S}} P\{z/y\}\sigma\{w/z\}$. So $w \notin \mathrm{n}(\beta)$, $V \xrightarrow{\beta} V'$ and $U' = (\nu w)V'$. Since σ is surjective, there is a u with $z[\sigma] = w$. Now $V \equiv^{\mathcal{S}} P\{z/y\}\sigma\{w/z\} \equiv^{\mathcal{S}} P\{u/y\}\sigma$ by (11) and (12). By rule **alpha** $P\{u/y\}\sigma \xrightarrow{\beta} V'$. Moreover, the inference of $P\{u/y\}\sigma \xrightarrow{\beta} V'$ has the same α-depth as that of $V \xrightarrow{\beta} V'$, which is smaller than the one of $U \xrightarrow{\beta} U'$. So by induction $P\{u/y\} \xrightarrow{\alpha} P'$ for some α and P' with $\alpha[\sigma] = \beta$ and $P'\sigma \equiv^{\mathcal{S}} V'$. Since $w \notin \mathrm{n}(\beta)$ one has $u \notin \mathrm{n}(\alpha)$. Hence $(\nu u)(P\{u/y\}) \xrightarrow{\alpha} (\nu u)P'$ by **res**. Moreover, since $w \notin \mathrm{fn}((\nu w)V) = \mathrm{fn}(U) = \mathrm{fn}(R\sigma)$, using (5), $u \notin \mathrm{fn}(R)$ by (8). Hence $R \equiv^{\mathcal{S}} (\nu u)(P\{u/y\})$ by (10) and $R \xrightarrow{\alpha} (\nu u)P'$ by **alpha**.
  Suppose that there is an $x \in \mathrm{fn}((\nu u)P')$ with $x[\sigma] = w$. Then, by Lemma A.2, $x \in \mathrm{n}(\alpha)$ or $x \in \mathrm{fn}(R)$. In the first case $w \in \mathrm{n}(\alpha[\sigma]) = \mathrm{n}(\beta)$, which has been ruled out. In the second case, by (8), $w \in \mathrm{fn}(R\sigma) = \mathrm{fn}(U)$, contradicting $U = (\nu w)V$. Hence there is no such x. Now pick $v \notin \mathrm{fn}(P') \cup dom(\sigma) \cup range(\sigma) \cup \mathrm{fn}(P'\sigma)$ and obtain

$$\begin{array}{lll} ((\nu u)P')\sigma & \equiv^{\mathcal{S}} ((\nu v)(P'\{v/u\}))\sigma & \text{by (10), (13)} \\ & = (\nu v)(P'\{v/u\}\sigma) & \text{by definition} \\ & \equiv^{\mathcal{S}} (\nu v)(P'\sigma\{v/w\}) & \text{by (11), (12)} \\ & \equiv^{\mathcal{S}} (\nu w)(P'\sigma) & \text{by (10)} \\ & \equiv^{\mathcal{S}} (\nu w)V' = U'. & \end{array}$$

- Suppose $U \xrightarrow{\beta} U'$ is derived by **symb-open**. Then $R = (\nu y)P$ and $R\sigma = (\nu z)(P\{z/y\}\sigma)$ for some $z \notin \mathrm{fn}((\nu y)P) \cup dom(\sigma) \cup range(\sigma)$. By (17), $U = (\nu w)V$ with $V \equiv^{\mathcal{S}} P\{z/y\}\sigma\{w/z\}$. So $\beta = M\bar{x}(w)$, $V \xrightarrow{M\bar{x}w} U'$ and $w \notin \mathrm{n}(M) \cup \{x\}$. Since σ is surjective, there is a u with $u[\sigma] = w$. Now $V \equiv^{\mathcal{S}} P\{z/y\}\sigma\{w/z\} \equiv^{\mathcal{S}} P\{u/y\}\sigma$ by (11) and (12). By rule **alpha** $P\{u/y\}\sigma \xrightarrow{M\bar{x}w} U'$.

Moreover, the inference of $P\{u/y\}\sigma \xrightarrow{M\bar{x}w} U'$ has the same α-depth as that of $V \xrightarrow{M\bar{x}w} U'$, which is smaller than the one of $U \xrightarrow{\beta} U'$. By induction $P\{u/y\} \xrightarrow{K\bar{q}r} P'$ for some K, q, r and P' with $K[\sigma] = M$, $q[\sigma] = x$, $r[\sigma] = w$ and $P'\sigma \equiv^S U'$.
Since $u[\sigma] = w \notin \text{FN}((\nu w)V) = \text{FN}(U) = \text{FN}(R\sigma)$, using (5), $u \notin \text{FN}(R)$ by (8). Hence $R \equiv^S (\nu u)(P\{u/y\})$ by (10). By Lemma A.1 $r \in \text{FN}(P\{u/y\})$. So if $r \neq u$ then $r \in \text{FN}((\nu u)(P\{u/y\}) = \text{FN}(R)$, again using (5), and $w = r[\sigma] \in \text{FN}(R\sigma)$ by (8), yielding a contradiction. Thus $r = u$. Since $w = u[\sigma] \notin \text{n}(M) \cup \{x\}$ one has $y \notin \text{n}(K) \cup \{q\}$. Hence $(\nu u)(P\{u/y\}) \xrightarrow{K\bar{q}(u)} P'$ by **SYMB-OPEN** and $R \xrightarrow{K\bar{q}(u)} P'$ by **ALPHA**.

- Suppose $U \xrightarrow{\beta} U'$ is derived by **ALPHA**. Then there is a $V \equiv^S U$ such that $V \xrightarrow{\alpha} U'$ is derived by a simpler proof. So $R\sigma \equiv^S V$ and by induction $R \xrightarrow{\alpha} R'$ for some α and R' with $\alpha[\sigma] = \beta$ and $R'\sigma \equiv^S U'$. □

A.2 A π-calculus with surjective substitutions

When needed I will address the π-calculus of Section 5 as $\pi(\mathcal{N})$, to make the choice of the set $\mathcal{N}$ of names explicit. Here I introduce a variant $\pi^{\mathcal{S}}(\mathcal{N})$ of the π-calculus that additionally employs a countably infinite collection of *spare names* $\mathcal{S} = \{s_1, s_2, \dots\}$, disjoint with $\mathcal{N}$. Its syntax is the same as the one of $\pi(\mathcal{N} \uplus \mathcal{S})$, except that in expressions $x(y).P$ one must take $y \in \mathcal{N}$. Moreover, in defining equations $A(\vec{x}) \stackrel{\text{def}}{=} P$ I require that $x_1, \dots, x_n \in \mathcal{N}$. The definition of substitution on $\pi^{\mathcal{S}}(\mathcal{N})$ is fine-tuned by requiring that when a bound name $y \in \mathcal{N}$ is changed into a bound name z to avoid name capture, one always chooses $z \in \mathcal{N}$.

As semantics of $\pi^{\mathcal{S}}(\mathcal{N})$ I use the variant $\xrightarrow{\alpha}{}^{\mathcal{S}}_{ES}$ of the early symbolic transition relation $\xrightarrow{\alpha}_{ES}$ where the substitutions $\{z/y\}$ and $\{\vec{y}/\vec{x}\}$ introduced by the operational rules **EARLY-INPUT** and **IDE** are changed into surjective substitutions $\{z/y\}^{\mathcal{S}}$ and $\{\vec{y}/\vec{x}\}^{\mathcal{S}}$. The motivation for this is to make Lemma A.11 applicable to these substitutions. In order to preserve the validity of (4)–(17), this change forces new definitions of $\equiv^{\mathcal{S}}$, FN and $\sigma[y \mapsto w]$.

Definition A.12. Let $\mathcal{H}$ be the set of all names currently in use, for now that is $\mathcal{N} \cup \mathcal{S}$. For $\vec{x} = (x_1, \dots, x_n) \in \mathcal{N}^n$ and $\vec{y} = (y_1, \dots, y_n) \in \mathcal{H}^n$, where the names x_i are all distinct, define the surjective substitution

$$\{\vec{x}/\vec{y}\}^{\mathcal{S}} : \mathcal{S} \cup \{x_1, \dots, x_n\} \to \mathcal{S} \cup \{x_1, \dots, x_n, y_1, \dots, y_n\}$$

by $\{\vec{x}/\vec{y}\}^{\mathcal{S}}(x_i) = y_i$ and $\{\vec{x}/\vec{y}\}^{\mathcal{S}}(s_i) = x_i$ for $i = 1, \dots, n$, and $\{\vec{x}/\vec{y}\}^{\mathcal{S}}(s_i) = s_{i-n}$ for $i > n$.

This is in fact the simplest possible adaptation of the substitution $\{\vec{y}/\vec{x}\}$ that makes it surjective. It is essential for my purposes that the axioms (4)–(17) of Section A.1 be satisfied, so that Lemmas A.8 and A.11 hold for $\pi^{\mathcal{S}}(\mathcal{N})$. In view of Property (11), to obtain (7) I need to ensure that

$$\{z[\sigma]/w\}^{\mathcal{S}} \circ \sigma[y \mapsto w] = \sigma \circ \{z/y\}^{\mathcal{S}}. \tag{18}$$

The definition of $\sigma[y \mapsto w]$ can be found by solving this equation. In Table 8, $\sigma : \mathcal{H} \rightharpoonup \mathcal{H}$ is an arbitrary substitution, $y, w \in \mathcal{N}$, and $w \notin dom(\sigma) \cup range(\sigma)$. Thus, $\sigma[y \mapsto w]$ is defined as the substitution that sends a name from the first column to the corresponding name from the last column. The middle columns show that now (18) is satisfied.

Definition A.13. For $x \in \mathcal{H}$ and $y \in \mathcal{N}$, let

$$x^-_y := \begin{cases} y & \text{if } x = s_1 \\ s_k & \text{if } x = s_{k+1} \\ x & \text{otherwise} \end{cases} \qquad \text{and} \qquad x^+_y := \begin{cases} s_1 & \text{if } x = y \\ s_{k+1} & \text{if } x = s_k \\ x & \text{otherwise.} \end{cases}$$

Now $\sigma[y \mapsto w]$, with $y, w \in \mathcal{N}$, is the substitution with $dom(\sigma[y \mapsto w]) = dom(\sigma) \cup \{y, w\} \cup \mathcal{S}$

Name n	$n[\{z/y\}^{\mathcal{S}}]$	$n[\{z/y\}^{\mathcal{S}}][\sigma] = n[\sigma[y \mapsto w]][\{z[\sigma]/w\}^{\mathcal{S}}]$	$n[\sigma[y \mapsto w]]$
y	z	$z[\sigma]$	w
w (if $\neq y$)	w	w	s_1
s_1 if $y=w$	w	w	s_1
s_1 if $y\neq w$	y	$y[\sigma] \neq w$	if $y[\sigma] \in \mathcal{N}$ $y[\sigma]$ if $y[\sigma] = s_k$ s_{k+1}
s_{i+1}	s_i	$s_i[\sigma] \neq w$	if $s_i[\sigma] \in \mathcal{N}$ $s_i[\sigma]$ if $s_i[\sigma] = s_k$ s_{k+1}
$x \in \mathcal{N}$ $x\neq y,w$	x	$x[\sigma] \neq w$	if $x[\sigma] \in \mathcal{N}$ $x[\sigma]$ if $x[\sigma] = s_k$ s_{k+1}

Table 8. Solving (18)

given by

$$x[\sigma[y \mapsto w]] := \begin{cases} w & \text{if } x = y \\ (x_y^-[\sigma])_w^+ & \text{otherwise.} \end{cases}$$

As far as the computational interpretation of the π-calculus concerns, the replacement of $P\{\vec{y}/\vec{x}\}$ by $P\{\vec{y}/\vec{x}\}^{\mathcal{S}}$ in rule **IDE** is of no consequence, as the names from $\mathcal{S}$ do not occur free in P anyway. However, the replacement of $P\{z/y\}$ by $P\{z/y\}^{\mathcal{S}}$ in rule **EARLY-INPUT** changes the meaning of the construct $x(y).P$. In the original π-calculus there are no free occurrences of y in $x(y).P$, and thus, after receiving a name $z \neq y$, the resulting process $P\{z/y\}$ cannot do an action $\xrightarrow{\bar{y}w}$. Here, however, upon receiving a name $z \neq y$, in the resulting process $P\{z/y\}^{\mathcal{S}}$ the spare name s_1 is elevated to y, so that $P\{z/y\}^{\mathcal{S}} \xrightarrow{\bar{y}w}$ is possible. For this reason, y can be considered a free name of $x(y).P$ when s_1 is a free name of P. More in general, a spare name s_k can be seen as a potential name $y \in \mathcal{N}$. This potential is realised when s_k occurs in the scope of k input prefixes, and the choice of y is made by the outermost of these. Hence the definition of FN, partly given in Section A.1, is completed by defining $\text{FN}(P)_y^- := \{x_y^- \mid x \in \text{FN}(P)\backslash\{y\}\}$.

Definition A.13 matches this intuition. To apply $\sigma[y \mapsto w]$ to a name $x \neq y$, first elevate x one level, then apply σ, and finally undo the elevation.

Note that $\text{T}_{\pi(\mathcal{N})} \subseteq \text{T}_{\pi^{\mathcal{S}}(\mathcal{N})}$, that is, the ordinary π-calculus processes form a subset of the processes in the π-calculus with surjective substitutions. A process $P \in \text{T}_{\pi(\mathcal{N})}$ by definition satisfies $\text{fn}(P) \subseteq \mathcal{N}$. The following lemma implies that on $\text{T}_{\pi(\mathcal{N})}$ there is no difference between FN and fn.

LEMMA A.14. If $\text{fn}(P) \subseteq \mathcal{N}$ then $\text{FN}(P) = \text{fn}(P)$.

PROOF. A trivial structural induction on P. □

I now proceed to show that the axioms (4), (6) and (8) from Section A.1—the ones not mentioning $\equiv^{\mathcal{S}}$—are satisfied.

LEMMA A.15. $\text{FN}(P\sigma) = \{x[\sigma] \mid x \in \text{FN}(P)\}$.

PROOF. With induction on the size of the parse tree of P. Note that this size is preserved under substitution. All cases are trivial, except for the two detailed below.

Let $P = x(y).Q$. Then $P\sigma = x[\sigma](z).(Q\sigma[y \mapsto z])$, so $\text{FN}(P\sigma) = \{x[\sigma]\} \cup \text{FN}(Q\sigma[y \mapsto z])_z^-$. By induction,

$$\text{FN}(Q\sigma[y \mapsto z]) = \{v[\sigma[y \mapsto z]] \mid v \in \text{FN}(Q)\} =$$
$$\{z \mid y \in \text{FN}(Q)\} \cup \{(v_y^-[\sigma])_z^+ \mid v \in \text{FN}(Q)\backslash\{y\}\},$$

so $\text{FN}(Q\sigma[y \mapsto z])^-_z = \{x^-_z \mid x \in \text{FN}(Q\sigma[y \mapsto z])\backslash\{z\}\} = \{v^-_y[\sigma] \mid v \in \text{FN}(Q)\backslash\{y\}\}$. Here I use that $u^+_z \neq z$ and $(u^+_z)^-_z = u$ for all names u.

Since $\text{FN}(P) = \{x\} \cup \{v^-_y \mid v \in \text{FN}(Q)\backslash\{y\}\}$, it follows that $\text{FN}(P\sigma) = \{x[\sigma] \mid x \in \text{FN}(P)\}$.

Let $P = (\nu y)Q$. Then $P\sigma = (\nu z)(Q\{z/y\}\sigma)$, so $\text{FN}(P\sigma) = \text{FN}(Q\{z/y\}\sigma)\backslash\{z\}$. By induction (applied twice) $\text{FN}(Q\{z/y\}\sigma) = \{v[\{z/y\}][\sigma] \mid v \in \text{FN}(Q)\}$. Considering that $z \notin \text{FN}(Q) \cup \mathit{dom}(\sigma) \cup \mathit{range}(\sigma)$,

$$\text{FN}(Q\{z/y\}\sigma)\backslash\{z\} = \{v[\sigma] \mid v \in \text{FN}(Q)\backslash\{y\}\}.$$

Since $\text{FN}(P) = \text{FN}(Q)\backslash\{y\}$, again the claim follows. □

For recursion equations $A(\vec{x}) \stackrel{\text{def}}{=} P$ the π-calculus requires $\text{fn}(P) \subseteq \{x_1, \ldots, x_n\}$, and I didn't bother to change that for $\pi^{\mathcal{S}}$. Since I require that $x_1, \ldots, x_n \in \mathcal{N}$, Lemma A.14 implies that also $\text{FN}(P) \subseteq \{x_1, \ldots, x_n\}$. Using this, (4) immediately follows from (8), which is Lemma A.15.

Lemma A.16. $\text{FN}(P\{z/y\}^{\mathcal{S}}) \subseteq \text{FN}(x(y).P) \cup \{z\}$.

Proof. If $v \in \text{FN}(P\{z/y\}^{\mathcal{S}})$ then $v = x[\{z/y\}^{\mathcal{S}}]$ for some $x \in \text{FN}(P)$, using Lemma A.15. If moreover $v \neq z$ then $v = x^-_y$. Hence $v \in \text{FN}(P)^-_y \subseteq \text{FN}(x(y).P)$. □

Now I define $\equiv^{\mathcal{S}}$ in such a way that (5) holds.

Definition A.17. For $y, z \in \mathcal{N}$ with $y \neq z$, let $\{z/y\}^3$ be the substitution σ with $\mathit{dom}(\sigma) = \{y, z, s_1\}$, $\sigma(y) = z$, $\sigma(z) = s_1$ and $\sigma(s_1) = y$. Moreover, let $\{y/y\}^3 := \epsilon$, the substitution with $\mathit{dom}(\epsilon) = \emptyset$. Let $\equiv^{\mathcal{S}}$ be the smallest congruence satisfying

(i) $(\nu y)P \equiv^{\mathcal{S}} (\nu z)(P\{z/y\})$ for any $z \notin \text{FN}((\nu y)P)$,
(ii) and $x(y).P \equiv^{\mathcal{S}} x(z).(P\{z/y\}^3)$ for any $z \in \mathcal{N}$.

Note that in $x(y).P$ the spare name s_1 occurring in P will be elevated to y, but in $x(z).Q$ it will be elevated to z. Therefore, when renaming the bound name y into z, the spare name s_1 should be renamed into y right away. Moreover, there is no reason to require that z not occur free in P; its free occurrences can simply be renamed into s_1.[9]

Lemma A.18. $P \equiv^{\mathcal{S}} Q \Rightarrow \text{FN}(P) = \text{FN}(Q)$.

Proof. Using transitivity of $\equiv^{\mathcal{S}}$, it suffices to prove this for the special case that P and Q differ by only one application of the generating equations from Definition A.17. Below I deal with the special case that this single application does not occur within a proper subterm of P; the general case then follows by a straightforward structural induction on P. There are two possibilities to consider.

Let $P = x(y).R$ and $Q = x(z).(R\{z/y\}^3)$. In this case $\text{FN}(P) = \{x\} \cup \text{FN}(R)^-_y$ and $\text{FN}(Q) = \{x\} \cup \text{FN}(R\{z/y\}^3)^-_z$. Moreover, $\text{FN}(R)^-_y = \{v^-_y \mid v \in \text{FN}(R)\backslash\{y\}\}$ and, by Lemma A.15,

$$\text{FN}(R\{z/y\}^3)^-_z = \{v[\{z/y\}^3]^-_z \mid v \in \text{FN}(R) \wedge v[\{z/y\}^3] \neq z\}.$$

Note that $v = y$ iff $v[\{z/y\}^3] = z$. So take $v \neq y$. A simple case distinction shows that $v[\{z/y\}^3]^-_z = v^-_y$. Consequently, $\text{FN}(R)^-_y = \text{FN}(R\{z/y\}^3)^-_z$ and thus $\text{FN}(P) = \text{FN}(Q)$.

Let $P = (\nu y).R$ and $Q = (\nu z).(R\{z/y\})$ with $z \notin \text{FN}((\nu y)P)$. Then $\text{FN}(P) = \text{FN}(R)\backslash\{y\}$ and $\text{FN}(Q) = \text{FN}(R\{z/y\})\backslash\{z\}$. So $\text{FN}(R\{z/y\}) = \{v[\{z/y\}] \mid v \in \text{FN}(R)\}$ by Lemma A.15. When $v \neq y, z$ one has $v = v[\{z/y\}] \neq z$, and when $v[\{z/y\}] \neq z$ then $v \neq y, z$. Consequently, $\text{FN}(P) = \text{FN}(Q)$. □

[9] An alternative form of Definition A.17 requires (ii) only for $z \notin \text{FN}((\nu y)P)$. This yields the same equivalence $\equiv^{\mathcal{S}}$ as Definition A.17, since even when $z \in \text{FN}((\nu y)P)$ one derives $x(y).P \equiv^{\mathcal{S}} x(w).(P\{w/y\}^3) \equiv^{\mathcal{S}} x(z).(P\{z/y\}^3)$ for some $w \notin \text{FN}((\nu y)P)$, using (19), that $P\{z/y\}^3\{w/z\}^3 \equiv^{\mathcal{S}} P\{w/y\}^3$.

The next two lemmas pave the way for Lemma A.21, which establishes the validity of Properties (11) and (12).

Lemma A.19. Let $y, w, z \in \mathcal{N}$, Then, for all $x \in \mathcal{H}$,

$$x[\sigma_1[y \mapsto w]][\sigma_2[w \mapsto z]] = x[(\sigma_2 \circ \sigma_1)[y \mapsto z]].$$

Proof. By Definition A.13, $x[\sigma_1[y \mapsto w]] = w$ only if $x = y$. So

$$x[\sigma_1[y \mapsto w]][\sigma_2[w \mapsto z]] = \begin{cases} z & \text{if } x = y \\ ((x_y^-[\sigma_1])_w^+)_w^-[\sigma_2]_z^+ & \text{otherwise.} \end{cases}$$

Moreover,

$$x[(\sigma_2 \circ \sigma_1)[y \mapsto z]] = \begin{cases} z & \text{if } x = y \\ (x_y^-[\sigma_1][\sigma_2])_z^+ & \text{otherwise.} \end{cases}$$

Since $(v_w^+)_w^- = v$ for all $v \in \mathcal{H}$, the lemma follows. □

Lemma A.20. Let $y, w, z \in \mathcal{N}$. Then, for all $x \in \mathcal{H}$,

$$x[\sigma[y \mapsto w]][\{z/w\}^3] = x[\sigma[y \mapsto z]]$$

$$x[\{w/y\}^3][\sigma[w \mapsto z]] = x[\sigma[y \mapsto z]].$$

Proof. If $x = y$ then $x[\sigma[y \mapsto w]][\{z/w\}^3] = z = x[\sigma[y \mapsto z]]$. Otherwise, considering that $v_w^+[\{z/w\}^3] = v_z^+$ for any $v \in \mathcal{H}$, $x[\sigma[y \mapsto w]][\{z/w\}^3] = (x_y^-[\sigma])_w^+[\{z/w\}^3] = (x_y^-[\sigma])_z^+ = x[\sigma[y \mapsto z]]$. The second statement follows likewise, using that $x[\{w/y\}^3]_w^- = x_y^-$ when $x \neq y$, □

Lemma A.21.

(1) $(P\sigma_1)\sigma_2 \equiv^{\mathcal{S}} P(\sigma_2 \circ \sigma_1)$.
(2) $(\forall x \in \text{fn}(P).\, x[\sigma] = x[\sigma']) \Rightarrow P\sigma \equiv^{\mathcal{S}} P\sigma'$.

Proof. I prove both statements by simultaneous structural induction on P.

(1) All cases are trivial, except for those where P has the form $x(y).Q$ or $(\nu y)Q$.
Let $P = x(y).Q$. By the definition of substitution, there is a u such that the last step in the below derivation holds. Likewise, there are w and z such that the first two steps hold.

$$\begin{aligned}
(P\sigma_1)\sigma_2 &= \big(x[\sigma_1](w).(Q\sigma_1[y \mapsto w])\big)\sigma_2 \\
&= x[\sigma_1][\sigma_2](z).\big((Q\sigma_1[y \mapsto w])\sigma_2[w \mapsto z]\big) \\
&\equiv^{\mathcal{S}} x[\sigma_2 \circ \sigma_1](z).\big(Q(\sigma_2[y \mapsto w] \circ \sigma_1[w \mapsto z])\big) \\
&\equiv^{\mathcal{S}} x[\sigma_2 \circ \sigma_1](z).\big(Q((\sigma_2 \circ \sigma_1)[y \mapsto z])\big) \\
&\equiv^{\mathcal{S}} x[\sigma_2 \circ \sigma_1](u).\big(Q((\sigma_2 \circ \sigma_1)[y \mapsto z]\{u/z\}^3)\big) \\
&\equiv^{\mathcal{S}} x[\sigma_2 \circ \sigma_1](u).\big(Q((\sigma_2 \circ \sigma_1)[y \mapsto u])\big) \\
&= P(\sigma_2 \circ \sigma_1).
\end{aligned}$$

Here the third step is an application of the induction hypotheses regarding statement 1) and the fourth step applies the induction hypotheses regarding statement 2), also using Lemma A.19. The fifth step applies Definition A.17 of $\equiv^{\mathcal{S}}$. The sixth step applies the induction hypotheses regarding both statements, also using Lemma A.20.
Let $P = (\nu y)Q$. By the definition of substitution, there is an u such that the last step in the below derivation holds. Likewise, there are w and z such that the first two steps hold. Now

choose a name v outside $\text{FN}\big(Q\{w/y\}\sigma_1\{z/w\}\sigma_2\big) \cup \text{FN}\big(Q\{u/y\}(\sigma_2 \circ \sigma_1)\big) \cup \{z, u\}$.

$$\begin{aligned}(P\sigma_1)\sigma_2 = &\ ((\nu w)(Q\{w/y\}\sigma_1))\sigma_2\\ = &\ (\nu z)(Q\{w/y\}\sigma_1\{z/w\}\sigma_2)\\ \equiv^S &\ (\nu v)(Q\{w/y\}\sigma_1\{z/w\}\sigma_2\{v/z\})\\ \equiv^S &\ (\nu v)(Q\{u/y\}(\sigma_2 \circ \sigma_1)\{v/u\})\\ \equiv^S &\ (\nu u)(Q\{u/y\}(\sigma_2 \circ \sigma_1))\\ = &\ P(\sigma_2 \circ \sigma_1).\end{aligned}$$

The third and fifth step apply Definition A.17 of $\equiv^S$. The fourth applies the induction hypotheses regarding both statements, also using that, for all $x \in \text{FN}(Q)$,

$$x[\{w/y\}][\sigma_1][\{z/w\}][\sigma_2][\{v/z\}] = x[\{u/y\}][\sigma_2 \circ \sigma_1][\{v/u\}].$$

The latter follows by a straightforward case distinction, considering that $w \notin \text{FN}(Q)\backslash\{y\} \cup dom(\sigma_1) \cup range(\sigma_1)$, that $z \notin \text{FN}(Q\{w/y\}\sigma_1)\backslash\{w\} \cup dom(\sigma_2) \cup range(\sigma_2)$ and that $u \notin \text{FN}((\nu y)Q) \cup dom(\sigma_2 \circ \sigma_1) \cup range(\sigma_2 \circ \sigma_1)$.

(2) Again all cases are trivial, except for the ones that P has the form $x(y).Q$ or $(\nu y)Q$.
Let $P = x(y).Q$ and $u[\sigma] = u[\sigma']$ for all $u \in \text{FN}(P)$. Then, for certain $z, w \in \mathcal{N}$, $P\sigma = x[\sigma](z).(Q\sigma[y \mapsto z])$ and $P\sigma' = x[\sigma'](w).(Q\sigma'[y \mapsto w])$.
By Definition A.17, $P\sigma' \equiv^S x[\sigma'](z).(Q\sigma'[y \mapsto w]\{z/w\}^3)$. By the induction hypotheses and Lemma A.20, $P\sigma' \equiv^S x[\sigma'](z).(Q\sigma'[y \mapsto z])$.
By Lemma A.1, $x \in \text{FN}(P)$, and thus $x[\sigma] = x[\sigma']$. To round of the proof by another appeal to the induction hypotheses regarding statement 2), it suffices to show that $u[\sigma[y \mapsto z]] = u[\sigma'[y \mapsto z]]$ for all $u \in \text{FN}(Q)$. This amounts to showing that $(u^-_y[\sigma])^+_z = (u^-_y[\sigma'])^+_z$ for all $u \in \text{FN}(Q)\backslash\{y\}$. Now for each $u \in \text{FN}(Q)\backslash\{y\}$ one has $u^-_y \in \text{FN}(P)$, by the definition of $\text{FN}(Q)^-_y$. Hence $u^-_y[\sigma] = u^-_y[\sigma']$, and the proof of this case is done.
Let $P = (\nu y)Q$ and let $u[\sigma] = u[\sigma']$ for all $u \in \text{FN}(P)$. Then $P\sigma = (\nu z)(Q\{z/y\}\sigma)$ and $P\sigma' = (\nu w)(Q\{w/y\}\sigma')$ for certain names z, w. Pick $v \notin \text{FN}(P\sigma) \cup \{z, w\} \cup dom(\sigma) \cup dom(\sigma')$. Then, by Definition A.17, $P\sigma \equiv^S (\nu v)(Q\{z/y\}\sigma\{v/z\})$ and $P\sigma' \equiv^S (\nu v)(Q\{w/y\}\sigma'\{v/w\})$. Now $u[\{z/y\}][\sigma][\{v/z\}] = u[\{v/y\}][\sigma]$ for all $u \in \text{FN}(Q)$, using that $z \notin \text{FN}(P) \cup dom(\sigma) \cup range(\sigma)$ and $v \notin dom(\sigma)$. Hence, by induction, $Q\{z/y\}\sigma\{v/z\} \equiv^S Q\{v/y\}\sigma$. Likewise, $Q\{w/y\}\sigma'\{v/w\} \equiv^S Q\{v/y\}\sigma'$. By Lemma A.15 $\text{FN}(Q\{v/y\})\backslash\{v\} \subseteq \text{FN}(P)$, so, again by induction, $Q\{v/y\}\sigma \equiv^S Q\{v/y\}\sigma'$, and hence $P\sigma \equiv^S P\sigma'$. □

Property (10) follows immediately from Definition A.17, and (7) follows from (11) (i.e. Lemma A.21.1) and (18). Also (9) follows from Lemma A.21, since $\text{FN}(P) \subseteq \{x_1, \ldots, x_n\}$. Hence it remains to verify Properties (13)–(17).

Lemma A.22. $P \equiv^S Q \Rightarrow P\sigma \equiv^S Q\sigma$.

Proof. Using transitivity of $\equiv^S$, it suffices to prove this for the special case that P and Q differ by only one application of the generating equations from Definition A.17. Below I deal with the special case that this single application does not occur within a proper subterm of P; the general case then follows by a structural induction on the size of the parse tree of P, recalling that this size is preserved under substitution.

There are two possibilities to consider.

Let $P = x(y).R$ and $Q = x(z).(R\{z/y\}^3)$. Then $P\sigma = x[\sigma](w).(R\sigma[y \mapsto w])$ and $Q\sigma = x[\sigma](u).((R\{z/y\}^3)\sigma[z \mapsto u])$. Thus $Q\sigma \equiv^S x[\sigma](w).(R\{z/y\}^3\sigma[z \mapsto u]\{w/u\}^3)$ by Definition A.17, so it suffices to show that

$$R\sigma[y \mapsto w] \equiv^S R\{z/y\}^3\sigma[z \mapsto u]\{w/u\}^3,$$

which follows from Lemmas A.20 and A.21.

Let $P = (\nu y)R$ and $Q = (\nu z)(R\{z/y\})$ with $z \notin \text{FN}(R)$. In that case $P\sigma = (\nu w)(R\{w/y\}\sigma)$, $Q\sigma = (\nu u)(R\{z/y\}\{u/z\}\sigma)$ for names w, u satisfying $w, u \notin \text{FN}(P) \cup \mathit{dom}(\sigma) \cup \mathit{range}(\sigma)$ (using that $\text{FN}(P) = \text{FN}(Q)$, by Lemma A.18). Choose a fresh name $v \notin \text{FN}(R\{w/y\}\sigma) \cup \text{FN}(R\{z/y\}\{u/z\}\sigma) \cup \{w, u\}$. Then one obtains $P\sigma \equiv^{\mathcal{S}} (\nu v)(R\{w/y\}\sigma\{v/w\})$ and $Q\sigma \equiv^{\mathcal{S}} (\nu v)(R\{z/y\}\{u/z\}\sigma\{v/u\})$. It suffices to show that $R\{w/y\}\sigma\{v/w\} \equiv^{\mathcal{S}} R\{z/y\}\{u/z\}\sigma\{v/u\}$. This follows from Lemma A.21, provided that, for all $x \in \text{FN}(R)$,

$$x[\{w/y\}][\sigma][\{v/w\}] = x[\{z/y\}][\{u/z\}][\sigma][\{v/u\}].$$

The latter is established by a simple case distinction. □

Lemma A.23. Let $y \in \mathcal{N}$ and $w \notin \text{FN}(x(y).P)$.
Then $P\{w/y\}^{\mathcal{S}}\{z/w\} \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}}$.

Proof. Since $w \notin \text{FN}(P)^-_y$, one has $w^+_y \notin \text{FN}(P)$. Now the statement follows from Lemma A.21, considering that, for all $x \in \text{FN}(P)$, $x[\{w/y\}^{\mathcal{S}}][\{z/w\}] = x[\{z/y\}^{\mathcal{S}}]$. □

Property (15) follows by a straightforward structural induction on P, using Lemma A.21 for the cases that $P = x(y).Q$ or $P = (\nu y)Q$.

Using Lemma A.21 it is trivial to check, for $y, w, z \in \mathcal{N}$, that

$$P\{w/y\}^3\{z/w\}^3 \equiv^{\mathcal{S}} P\{z/y\}^3 \tag{19}$$

$$P\{w/y\}^3\{u/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{u/y\}^{\mathcal{S}}. \tag{20}$$

Lemma A.24. If $U \equiv^{\mathcal{S}} x(y).P$ then $U = x(w).V$ for some $V \equiv^{\mathcal{S}} P\{w/y\}^3$. Likewise, if $U \equiv^{\mathcal{S}} (\nu y)P$ then $U = (\nu w)V$ for some $w \notin \text{FN}(U)$ and $V \equiv^{\mathcal{S}} P\{w/y\}$.

Proof. Since $U \equiv^{\mathcal{S}} x(y).P$, one has $U = U_0 \equiv^{\mathcal{S}} U_1 \equiv^{\mathcal{S}} U_2 \equiv^{\mathcal{S}} \cdots \equiv^{\mathcal{S}} U_n = x(y).P$, and each step $U_i \equiv^{\mathcal{S}} U_{i+1}$ involves exactly one application of the generating equations from Definition A.17. I prove the statement by induction on n.

The case that $n = 0$ is trivial; take $w := y$.

Let $n > 0$. By induction $U_1 = x(w).V$ with $V \equiv^{\mathcal{S}} P\{w/y\}^3$.

If for $U = U_0 \equiv^{\mathcal{S}} U_1 = x(w).V$ the application of the generating equation from Definition A.17 occurs entirely within V, one has $U = x(w).W$ with $W \equiv^{\mathcal{S}} V \equiv^{\mathcal{S}} P\{w/y\}^3$.

If the application is at top level, then $U = x(z).W$ and either $W = V\{z/w\}^3$ or $V = W\{w/z\}^3$. By (19), (15), in either case $W \equiv^{\mathcal{S}} V\{z/w\}^3$. Thus $W \equiv^{\mathcal{S}} P\{z/y\}^3$ by (19).

The second statement is obtained in the same way. □

The last property to be checked, (16), follows from Lemma A.24.

Corollary A.25. If $U \equiv^{\mathcal{S}} (x(y).P)\sigma$ then $U = x[\sigma](w).V$ with $V \equiv^{\mathcal{S}} P\sigma[y \mapsto w]$.

Proof. By definition $(x(y).P)\sigma = x[\sigma](z).(P\sigma[y \mapsto z])$ for some $z \in \mathcal{N}$. By Lemma A.24 $U = x(w).V$ for some $V \equiv^{\mathcal{S}} P\sigma[y \mapsto z]\{w/z\}^3$. By Lemmas A.20 and A.21 $V \equiv^{\mathcal{S}} P\sigma[y \mapsto w]$. □

Since the axioms (4)–(17) of Section A.1 hold for $\pi^{\mathcal{S}}(\mathcal{N})$, so do Lemmas A.1–A.11 from Section A.1.

As mentioned earlier, the classical π-calculus $\pi(\mathcal{N})$ can be seen as a subcalculus of the π-calculus with surjective substitutions $\pi^{\mathcal{S}}(\mathcal{N})$; its processes P satisfy $\text{fn}(P) \subseteq \mathcal{N}$. Lemma A.14 shows that on this subcalculus one has $\text{fn}(P) = \text{FN}(P)$, so that FN can be seen as an extension of fn from $\pi(\mathcal{N})$ to all of $\pi^{\mathcal{S}}(\mathcal{N})$.

Likewise, the application $P\sigma$ of a substitution σ to a process P is unchanged up to $\equiv$ when $\text{fn}(P) \subseteq \mathcal{N}$ and $\sigma : \mathcal{H} \rightharpoonup \mathcal{N}$. Namely, if $\text{fn}(Q) \subseteq \mathcal{N}$, $w \notin \text{FN}((\nu y)Q) \cup \mathit{dom}(\sigma) \cup \mathit{range}(\sigma)$ and $\sigma : \mathcal{H} \rightharpoonup \mathcal{N}$, then $Q\sigma[y \mapsto w] \equiv Q\{w/y\}\sigma$.

The next lemma shows that also $\equiv^{\mathcal{S}}$ and $\equiv$ coincide on the subcalculus $\pi(\mathcal{N})$ of $\pi^{\mathcal{S}}(\mathcal{N})$.

Lemma A.26. Let $\mathrm{fn}(P|Q) \subseteq \mathcal{N}$. Then $P \equiv^{\mathcal{S}} Q \Leftrightarrow P \equiv Q$.

Proof. "$\Rightarrow$": It suffices to consider the case that P and Q differ by only one application of the generating equations from Definition A.17. Below I deal with the special case that this single application does not occur within a proper subterm of P; the general case then follows by structural induction on P.

The case that $P = (\nu y)R$ and $Q = (\nu z)(R\{z/y\})$ with $z \notin \mathrm{fn}((\nu y)R)$ is trivial.

So let $P = x(y).R$ and $Q = x(z).(R\{z/y\}^3)$. As $\mathrm{fn}(P) \subseteq \mathcal{N}$ and $\mathrm{fn}(Q) \subseteq \mathcal{N}$, also $\mathrm{fn}(R) \subseteq \mathcal{N}$ and $\mathrm{fn}(R\{z/y\}^3) \subseteq \mathcal{N}$, so $z \notin \mathrm{fn}((\nu y)R)$. Considering that $v[\{z/y\}^3] = v[\{z/y\}]$ for all $v \in \mathrm{fn}(R)$, and that the classical notion of substitution may be applied here, $R\{z/y\}^3 \equiv R\{z/y\}$. Hence $P \equiv Q$.

"$\Leftarrow$": It suffices to consider the case that P and Q differ by only one application of the generating equations from Section 5. Below I deal with the special case that this single application does not occur within a proper subterm of P; the general case then follows by structural induction on P.

The case that $P = (\nu y)R$ and $Q = (\nu z)(R\{z/y\})$ with $z \notin \mathrm{fn}((\nu y)R)$ is trivial.

So let $P = x(y).R$ and $Q = x(z).(R\{z/y\})$ with $z \notin \mathrm{fn}((\nu y)R)$. In that case $\mathrm{fn}(R) = \mathrm{fn}(R) \subseteq \mathcal{N}$. Considering that $v[\{z/y\}] = v[\{z/y\}^3]$ for all $v \in \mathrm{fn}(R)$, Lemma A.21.2 yields $R\{z/y\} \equiv^{\mathcal{S}} R\{z/y\}^3$. Hence $P \equiv^{\mathcal{S}} Q$. □

Now I will show that the identity $\mathrm{id} : \mathrm{T}_{\pi_{ES}(\mathcal{N})} \to \mathrm{T}_{\pi^{\mathcal{S}}_{ES}(\mathcal{N})}$ is a valid translation from $\pi_{ES}(\mathcal{N})$ to $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$. My definition of $R \xrightarrow{\alpha}_{ES} R'$ implies that $R, R' \in \mathrm{T}_{\pi_{ES}(\mathcal{N})}$ and $\mathrm{n}(\alpha) \subseteq \mathcal{N}$.

Lemma A.27. If $R \xrightarrow{\alpha}_{ES} R'$ then equally $R \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ for some $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ process U' with $U' \equiv^{\mathcal{S}} R'$.

Proof. With induction on the inference of $R \xrightarrow{\alpha}_{ES} R'$.

- Suppose $R \xrightarrow{\alpha}_{ES} R'$ is derived by rule **early-input**. Then $R = x(y).P$, $\alpha = xz$ and $R' = P\{z/y\}$. Using **early-input** from $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$, $R \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U' := P\{z/y\}^{\mathcal{S}}$. By Lemma A.14 $\mathrm{fn}(P) \subseteq \mathcal{N}$. So by (12) $U' \equiv^{\mathcal{S}} R'$.
- Suppose $R \xrightarrow{\alpha}_{ES} R'$ is derived by **ide**. Then $R = A(\vec{y})$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha}_{ES} R'$. So by induction $P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ for some $U' \equiv^{\mathcal{S}} R'$. As $\mathrm{fn}(P) \subseteq \mathcal{N}$, $P\{\vec{y}/\vec{x}\} \equiv^{\mathcal{S}} P\{\vec{y}/\vec{x}\}^{\mathcal{S}}$ by (12) and Lemma A.14. Hence $P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ by **alpha**. Thus $R \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ by rule **ide** from $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$.
- All other cases are trivial. □

Lemma A.28. If $R \in \mathrm{T}_{\pi_{ES}(\mathcal{N})}$ and $R \equiv^{\mathcal{S}} U \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ with $\iota(\alpha) \cup \mathrm{bn}(\alpha) \subseteq \mathcal{N}$, then equally $R \xrightarrow{\alpha}_{ES} R'$ for some $R' \equiv^{\mathcal{S}} U'$.

Proof. With induction on the α-depth of the inference of $U \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$, with a nested induction on the number of applications of rule **alpha** at the end of that inference.

- Suppose $U \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ is derived by rule **early-input**. Then $R = x(y).P$ and $U = x(w).V$ with $V \equiv^{\mathcal{S}} P\{w/y\}^3$ by Lemma A.24. Moreover, $\alpha = xz$ and $y, w, z \in \mathcal{N}$. So $U' = V\{z/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{w/y\}^3\{z/w\}^{\mathcal{S}} \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}}$ by (20).
  By **early-input** from $\pi_{ES}(\mathcal{N})$, $R \xrightarrow{\alpha}_{ES} R' := P\{z/y\}$. By Lemma A.14 $\mathrm{fn}(P) \subseteq \mathcal{N}$. By (12) $R' \equiv^{\mathcal{S}} P\{z/y\}^{\mathcal{S}} \equiv^{\mathcal{S}} U'$.
- Suppose $U \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$ is derived by rule **ide**. Then $R = U = A(\vec{y})$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha}^{\mathcal{S}}_{ES} U'$. As $P \in T_{\pi_{ES}(\mathcal{N})}$, and $\{y_1, \ldots, y_n\} = \mathrm{fn}(R) \subseteq \mathcal{N}$, also $P\{\vec{y}/\vec{x}\} \in T_{\pi_{ES}(\mathcal{N})}$. Using (12), $P\{\vec{y}/\vec{x}\} \equiv^{\mathcal{S}} P\{\vec{y}/\vec{x}\}^{\mathcal{S}}$. So by induction $P\{\vec{y}/\vec{x}\} \xrightarrow{\alpha}_{ES} R'$ for some $R' \equiv^{\mathcal{S}} U'$. Consequently, $R \xrightarrow{\alpha}_{ES} R'$ by rule **ide** from $\pi_{ES}(\mathcal{N})$.

Table 9. Early symbolic structural operational semantics of the π-calculus without rule **ALPHA**

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

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

EARLY-INPUT: $Mx(y).P \xrightarrow{Mxz} P\{z/y\}^{\mathcal{S}}$

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: $$\frac{P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha} P'}{A(\vec{y}) \xrightarrow{\alpha} P'} \; (A(\vec{x}) \stackrel{\text{def}}{=} P)$$

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

E-S-COM: $$\frac{P \xrightarrow{M\bar{x}y} P',\ Q \xrightarrow{Nvy} Q'}{P|Q \xrightarrow{[x=v]MN\tau} P'|Q'}$$

E-S-CLOSE: $$\frac{P \xrightarrow{M\bar{x}(z)} P',\ Q \xrightarrow{Nvz} Q'}{P|Q \xrightarrow{[x=v]MN\tau} (\nu z)(P'|Q')} \begin{pmatrix} z \notin \\ \mathrm{fn}(Q) \end{pmatrix}$$

RES-ALPHA: $$\frac{P\{z/\!\!/y\} \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu z)P'} \begin{pmatrix} z \notin \mathrm{FN}((\nu y)P) \\ z \notin \mathrm{n}(\alpha) \end{pmatrix}$$

SYMB-OPEN-ALPHA: $$\frac{P \xrightarrow{M\bar{x}y} P'}{(\nu y)P \xrightarrow{M\bar{x}(z)} P'\{z/\!\!/y\}} \begin{pmatrix} y \neq x \\ z \notin \mathrm{FN}((\nu y)P) \\ y \notin \mathrm{n}(M) \end{pmatrix}$$

- Suppose $U \xrightarrow{\alpha}{}^{\mathcal{S}}_{ES} U'$ is derived by **E-S-CLOSE**. Then $R = P|Q$, $U = V|W$, $P \equiv^{\mathcal{S}} V \xrightarrow{M\bar{x}(z)}{}^{\mathcal{S}}_{ES} V'$, $Q \equiv^{\mathcal{S}} W \xrightarrow{Nvz}{}^{\mathcal{S}}_{ES} W'$, $\alpha = [x{=}v]MN\tau$ and $U' = (\nu z)(V'|W')$ for some $z \notin \mathrm{FN}(Q)$. Pick $w \in \mathcal{N}\backslash\mathrm{FN}(R)$. Now $V \xrightarrow{M\bar{x}(w)}{}^{\mathcal{S}}_{ES} \equiv^{\mathcal{S}} V'\{w/z\}$ and $W \xrightarrow{Nvw}{}^{\mathcal{S}}_{ES} \equiv^{\mathcal{S}} W'\{w/z\}$ by Lemmas A.7 and A.6. Moreover, the inferences of these transitions have a smaller α-depth than that of the transition $R \xrightarrow{\alpha}{}^{\mathcal{S}}_{ES} U'$. So $P \xrightarrow{M\bar{x}(w)}_{ES} P' \equiv^{\mathcal{S}} V'\{w/z\}$ and $Q \xrightarrow{Nvw}_{ES} Q' \equiv^{\mathcal{S}} W'\{w/z\}$ by induction. Applying rule **E-S-CLOSE**, $R \xrightarrow{\alpha}_{ES} R' := (\nu w)(P'|Q')$. Lemma A.2 yields $\mathrm{FN}(R') \subseteq \mathrm{FN}(R) \not\ni w$. Using this, one obtains $R' \equiv^{\mathcal{S}} (\nu w)(V'\{w/z\}|W'\{w/z\}) \equiv^{\mathcal{S}} (\nu z)(V'|W') = U'$.
- All other cases are trivial. In the case of **E-S-COM** apply Lemma A.1 to obtain $y \in \mathcal{N}$. □

THEOREM A.29. $P \rightleftharpoons \mathrm{id}(P)$ for any $P \in \mathrm{T}_{\pi_{ES}(\mathcal{N})}$.

PROOF. It suffices to show that the symmetric closure of

$$\mathscr{R} := \{(R,U) \mid R \in \mathrm{T}_{\pi_{ES}(\mathcal{N})}, U \in \mathrm{T}_{\pi^{\mathcal{S}}_{ES}(\mathcal{N})} \wedge R \equiv^{\mathcal{S}} U\}$$

is a strong barbed bisimulation. Let $(R,U) \in \mathscr{R}$.

Let $R \xrightarrow{\tau}_{ES} R'$. By Lemma A.27, $R \xrightarrow{\tau}{}^{\mathcal{S}}_{ES} U'$ for some $U' \equiv^{\mathcal{S}} R'$. So $U \xrightarrow{\tau}{}^{\mathcal{S}}_{ES} U'$ by rule **ALPHA** of $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$.

Let $U \xrightarrow{\tau}{}^{\mathcal{S}}_{ES} U'$. By Lemma A.28, $\exists R'.\ R \xrightarrow{\tau}_{ES} R' \equiv^{\mathcal{S}} U'$.

Let $U{\downarrow_b}$ with $b \in \mathcal{Z} \cup \overline{\mathcal{Z}}$. Then $U \xrightarrow{by}{}^{\mathcal{S}}_{ES} U'$ or $U \xrightarrow{b(y)}{}^{\mathcal{S}}_{ES} U'$ for some y and U', using the definition of O in Section 4. By Lemmas A.5 and A.7 I may assume, without loss of generality, that $y \in \mathcal{N}$ in case of input or bound output actions. So $R \xrightarrow{by}_{ES} R'$ or $R \xrightarrow{b(y)}_{ES} R'$ by Lemma A.28. Thus $R{\downarrow_b}$.

The implication $R{\downarrow_b} \Rightarrow U{\downarrow_b}$ proceeds likewise. □

A.3 The elimination of ALPHA

Let $\xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet}$ be the transition relation on T_π generated by the rules of Table 9. Here $\{z/\!\!/y\}$ denotes the substitution σ with $dom(\sigma) = \{y,z\}$, $\sigma(y) = z$ and $\sigma(z) = y$.[10] Compared to the operational semantics of $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$, rule **ALPHA** is omitted, but applications of this rule are incorporated in rules

[10] In rule **RES-ALPHA**, thanks to the side condition $z \notin \mathrm{FN}((\nu y)P)$, using (12), $P\{z/\!\!/y\} \equiv^{\mathcal{S}} P\{z/y\}$. For this reason, the versions with $\{z/\!\!/y\}$ and $\{z/y\}$ are equivalent, up to strong bisimilarity. The use of $\{z/\!\!/y\}$ instead of $\{z/y\}$ makes the substitution surjective, which will be needed in Section A.4.

The same can be said about **SYMB-OPEN-ALPHA**, since $\mathrm{FN}((\nu y)P') \subseteq \mathrm{FN}((\nu y)P)$ by Lemma A.2.

RES-ALPHA and **SYMB-OPEN-ALPHA**.[11] Lemmas A.30 and A.31 below say that up to strong bisimilarity the transitions relations $\xrightarrow{\alpha}{}^{S}_{ES}$ and $\xrightarrow{\alpha}{}^{S}_{\bullet}$ of $\pi^{S}_{ES}(\mathcal{N})$ are equivalent.

LEMMA A.30. If $R \xrightarrow{\alpha}{}^{S}_{\bullet} R'$ then equally $R \xrightarrow{\alpha}{}^{S}_{ES} \equiv^{S} R'$.

PROOF. With induction on the inference of $R \xrightarrow{\alpha}{}^{S}_{\bullet} R'$. The only nontrivial cases are when $R \xrightarrow{\alpha}{}^{S}_{\bullet} R'$ is derived by rule **RES-ALPHA** or **SYMB-OPEN-ALPHA**.

By (12), $P\{z/\!\!/y\} \equiv^{S} P\{z/y\}$ when $z \notin \text{FN}((\nu y)P)$. Thus one obtains $(\nu y)P \equiv^{S} (\nu z)(P\{z/y\}) \equiv^{S} (\nu z)(P\{z/\!\!/y\})$. Using this, each application of rule **RES-ALPHA** can be mimicked by an application of **RES** followed by one of **ALPHA**.

Now suppose $R \xrightarrow{\alpha}{}^{S}_{\bullet} R'$ is derived by **SYMB-OPEN-ALPHA**. Then $R = (\nu y)P$, $\alpha = M\bar{x}(z)$, $y \neq x$, $z \notin \text{FN}(R)$, $y \notin \text{n}(M)$, $P \xrightarrow{M\bar{x}y}{}^{S}_{\bullet} P'$ and $R' = P'\{z/\!\!/y\}$. By induction $P \xrightarrow{M\bar{x}y}{}^{S}_{ES} \equiv^{S} P'$. By Lemma A.1 $\text{n}(M) \cup \{x\} \subseteq \text{FN}(P)\backslash\{y\} = \text{FN}(R) \not\ni z$. So by Lemma A.4 $P\{z/\!\!/y\} \xrightarrow{M\bar{x}z}{}^{S}_{ES} \equiv^{S} P'\{z/\!\!/y\}$. By **SYMB-OPEN** $(\nu z)(P\{z/\!\!/y\}) \xrightarrow{M\bar{x}(z)}{}^{S}_{ES} \equiv^{S} P'\{z/\!\!/y\} = R'$. So by rule **ALPHA** $R \xrightarrow{\alpha}{}^{S}_{ES} \equiv^{S} R'$. □

LEMMA A.31. If $R \equiv^{S} U$ and $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ then equally $U \xrightarrow{\alpha}{}^{S}_{\bullet} U'$ for some U' with $R' \equiv^{S} U'$.

PROOF. With induction on the α-depth of the inference of $R \xrightarrow{\alpha}{}^{S}_{ES} R'$, with a nested induction on the number of applications of rule **ALPHA** at the end of the inference.

- The cases that $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ is derived by rule **TAU** or **OUTPUT** are trivial.
- Suppose $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ is derived by **EARLY-INPUT**. Then $R = x(y).P$, $\alpha = xz$ and $R' = P\{z/y\}^{\mathcal{S}}$. By Lemma A.24 $U = x(w).V$ for some $V \equiv^{S} P\{w/y\}^{3}$. Thus $U \xrightarrow{\alpha}{}^{S}_{\bullet} V\{z/w\}^{\mathcal{S}}$. So $V\{z/w\}^{\mathcal{S}} \equiv^{S} P\{w/y\}^{3}\{z/w\}^{\mathcal{S}} \equiv^{S} P\{z/y\}^{\mathcal{S}} = R'$ by (13) and (20).
- Suppose $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ is derived by **IDE**. Then $R = A(\vec{y})$, so $U = R$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha}{}^{S}_{ES} R'$. By induction $P\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\alpha}{}^{S}_{\bullet} U'$ for some U' with $R' \equiv^{S} U'$. By **IDE** $U \xrightarrow{\alpha}{}^{S}_{\bullet} U'$.
- Suppose $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ is derived by **PAR**. Then $R = P|Q$, $P \xrightarrow{\alpha}{}^{S}_{ES} P'$, $R' = P'|Q$ and $\text{bn}(\alpha) \cap \text{FN}(Q) = \emptyset$. Thus $U = V|W$ with $P \equiv^{S} V$, $Q \equiv^{S} W$ and $\text{bn}(\alpha) \cap \text{FN}(W) = \emptyset$. By induction $V \xrightarrow{\alpha}{}^{S}_{\bullet} V'$ for some V' with $P' \equiv^{S} V'$. So $U = V|W \xrightarrow{\alpha}{}^{S}_{\bullet} V'|W$ by rule **PAR**, and $R' \equiv^{S} V'|W$.
- The cases that $R \xrightarrow{\alpha}{}^{S}_{\bullet} R'$ is derived by rule **E-S-COM**, **E-S-CLOSE**, **SUM** or **SYMB-MATCH** are (also) trivial.
- Suppose $R \xrightarrow{\alpha}{}^{S}_{ES} R'$ is derived by rule **RES**. Then $R = (\nu z)P$, $P \xrightarrow{\alpha}{}^{S}_{ES} P'$, $z \notin \text{n}(\alpha)$ and $R' = (\nu z)P'$. So $U = (\nu y)V$ for some $y \notin \text{FN}((\nu z)P)$ and $V \equiv^{S} P\{y/z\}$, using Lemma A.24. Employing (13), (15) and Lemma A.21, $V\{z/\!\!/y\} \equiv^{S} P\{y/z\}\{z/\!\!/y\} \equiv^{S} P\{z/\!\!/z\} \equiv^{S} P$. So by induction $V\{z/\!\!/y\} \xrightarrow{\alpha}{}^{S}_{\bullet} V'$ for some $V' \equiv^{S} P'$. By (5) $z \notin \text{FN}((\nu z)P) = \text{FN}((\nu y)(P\{y/z\})) = \text{FN}((\nu y)V)$. Hence $U = (\nu y)V \xrightarrow{\alpha}{}^{S}_{\bullet} (\nu z)V'$ by rule **RES-ALPHA**. Moreover, $R' = (\nu z)P' \equiv^{S} (\nu z)V'$.
- Let $R \xrightarrow{M\bar{x}(z)}{}^{S}_{ES} R'$ be derived by **SYMB-OPEN**. Then $R = (\nu z)P$, $P \xrightarrow{M\bar{x}z}{}^{S}_{ES} R'$, $z \neq x$ and $z \notin \text{n}(M)$. Thus $U = (\nu y)V$ for $y \notin \text{FN}((\nu z)P)$ and $V \equiv^{S} P\{y/z\}$, using Lemma A.24. Now $P\{y/z\} \xrightarrow{M\bar{x}y}{}^{S}_{ES} R'\{y/z\}$ by Lemma A.4. Moreover, the α-depth of the inference of $P\{y/z\} \xrightarrow{M\bar{x}y}{}^{S}_{ES} R'\{y/z\}$ equals that of $P \xrightarrow{M\bar{x}z}{}^{S}_{ES} R'$, which is smaller than that of $R \xrightarrow{M\bar{x}(z)}{}^{S}_{ES} R'$. By induction $V \xrightarrow{M\bar{x}y}{}^{S}_{\bullet} V'$ for some $V' \equiv^{S} R'\{y/z\}$. By (5), $z \notin \text{FN}(R) = \text{FN}(U)$. Employing Lemma A.1, $\text{n}(M) \cup \{x\} \subseteq \text{FN}(P)\backslash\{z\} = \text{FN}(R) \not\ni y$. Consequently, $U = (\nu y)V \xrightarrow{M\bar{x}(z)}{}^{S}_{\bullet} V'\{z/\!\!/y\}$ by **SYMB-OPEN-ALPHA**.

[11] Besides the insignificant difference of $\{z/\!\!/y\}$ versus $\{z/y\}$, rule **SYMB-OPEN-ALPHA** differs from **SYMB-ALPHA-OPEN** in the more restrictive side condition $z \notin \text{FN}((\nu y)P)$ versus $z \notin \text{FN}((\nu y)P')$. Whereas the two rules are interchangeable up to strong barbed bisimilarity, they are not up to strong bisimilarity, for the transition $(\nu y)(\bar{x}y.\mathbf{0} + \bar{z}x.\mathbf{0}) \xrightarrow{\bar{x}(z)} \mathbf{0}$ is not derivable from Table 9. I need **SYMB-OPEN-ALPHA** to obtain a transition relation that is strongly bisimilar with the one of $\pi^{S}_{ES}(\mathcal{N})$.

Moreover, $V'\{z/\!\!/y\} \equiv^{\mathcal{S}} R'\{y/z\}\{z/\!\!/y\} \equiv^{\mathcal{S}} R'$ by (11)–(13) and (15), using that $y \notin \text{FN}(R) \supseteq \text{FN}(R')\backslash\{z\}$.

- Suppose $R \xrightarrow{\alpha}{}^{\mathcal{S}}_{ES} R'$ is derived by **ALPHA**. Then there is a $Q \equiv^{\mathcal{S}} R$ such that $Q \xrightarrow{\alpha}{}^{\mathcal{S}}_{ES} R'$ is derived by a simpler proof. So $U \equiv^{\mathcal{S}} Q$ and by induction $U \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} R'$. □

Combining Lemmas A.30 and A.31 yields the following result, stating that rule **ALPHA** is not needed on top of the rules of Table 9.

COROLLARY A.32. If $R \equiv^{\mathcal{S}} U$ and $R \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} R'$ then equally $U \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} U'$ for some U' with $R' \equiv^{\mathcal{S}} U'$. □

Together, Lemmas A.30 and A.31 imply that the relation $\equiv^{\mathcal{S}}$ is a strong (barbed) bisimulation between $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ expressions equipped with the semantics of Table 6, amended as described in Sections A.1 and A.2, and $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ expressions equipped with the semantics of Table 9. Hence, up to $\dot{\sim}$ it doesn't matter whether which of these semantics one employs.

Although I will not need this in this paper, a trivial adaptation of the above proofs shows that also the early and early symbolic semantics of the π-calculus, displayed in Tables 4 and 6 can be equivalently adapted by dropping rule **ALPHA** at the cost of strengthening **RES** and **OPEN** (or **SYMB-OPEN**) into **RES-ALPHA** and **OPEN-ALPHA** (or **SYMB-OPEN-ALPHA**).

A.4 Replacing substitution by relabelling

Let $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ be the variant of the π-calculus with surjective substitutions $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$, enriched with a postfix-written relabelling operator $[\sigma]$ for each surjective substitution σ. Its operational semantics is given by the rules of Table 9, together with the rule **RELABELLING**

$$\frac{P \xrightarrow{\alpha} P'}{P[\sigma] \xrightarrow{\alpha[\sigma]} P'[\sigma]} \ (\text{bn}(\alpha[\sigma]) \cap \text{FN}(P[\sigma]) = \emptyset)$$

for the relabelling operators, except that the substitutions $\{z/y\}^{\mathcal{S}}$ and $\{\vec{y}/\vec{x}\}^{\mathcal{S}}$ that appear in rules **EARLY-INPUT** and **IDE** are replaced by applications of the relabelling operators $[\{z/y\}^{\mathcal{S}}]$ and $[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$, respectively, and the substitution $\{z/\!\!/y\}$ that appears in rules **RES-ALPHA** and **SYMB-OPEN-ALPHA** is replaced by the relabelling operator $[\{z/\!\!/y\}]$. Moreover, rules **PAR** and **E-S-CLOSE** gain side conditions $\text{bn}(\alpha) \cap \text{FN}(P) = \emptyset$ and $z \notin \text{FN}(P)$, respectively.[12] Except for **IDE**, $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ has the semantics of Table 10. The function FN, used in several rules, is extended to $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ by $\text{FN}(P[\sigma]) := \{x[\sigma] \mid x \in \text{FN}(P)\}$. Recall that names are chosen from $\mathcal{N} \uplus \mathcal{S}$, except that in $x(y).P$ one always has $y \in \mathcal{N}$. Moreover, in defining equations $A(\vec{x}) \stackrel{\text{def}}{=} P$ I require that $x_1, \ldots, x_n \in \mathcal{N}$, and $\text{fn}(P) \subseteq \{x_1, \ldots, x_n\}$.[13]

I will show that the identity $\text{id} : \text{T}_{\pi^{\mathcal{S}}_{ES}(\mathcal{N})} \to \text{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})}$ is a valid translation from $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ to $\pi^{\rho\bullet}_{ES}(\mathcal{N})$. As justified in Section A.3, at the side of $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ I will use the transition relation $\longrightarrow^{\mathcal{S}}_{\bullet}$ generated by Table 9. Lemmas A.30 and A.31 imply that Lemmas A.6–A.11 also hold for $\longrightarrow^{\mathcal{S}}_{\bullet}$. I will denote the transition relation of $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ simply by $\xrightarrow{\alpha}$.

For P a $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ process, let $\widehat{P}$ be the $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ process obtained from P by recursively replacing each subterm $Q[\sigma]$ by $Q\sigma$, and each agent identifier A by $A\hat{}$. Here $A\hat{}$ is a fresh agent identifier with defining equation $A\hat{}(\vec{x}) \stackrel{\text{def}}{=} \widehat{P}$ when $A(\vec{x}) \stackrel{\text{def}}{=} P$ was the defining equation of A. Note that $\text{FN}(\widehat{P}) = \text{FN}(P)$ for each $P \in \text{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})}$.

[12] These side conditions are not essential—they could be left out here, or added to the original π-calculus, without ill effects—but are included to obtain a stronger and simpler version of Lemmas A.44 and A.51 in Sections A.6 and A.7.

[13] Optionally, one could furthermore require that no relabelling operators may occur in defining equations; this would simplify the definition of $\widehat{P}$.

Lemma A.33. If $R \xrightarrow{\beta} R'$ then equally $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$.

Proof. By induction of the inference of $R \xrightarrow{\beta} R'$.

- Suppose $R \xrightarrow{\beta} R'$ is derived by **tau**. Then $R = \tau.P$, $\beta = \tau$ and $R' = P$. Moreover, $\widehat{R} = \tau.\widehat{P}$ and $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \widehat{R'}$.
- The case that $R \xrightarrow{\beta} R'$ stems from **output** goes likewise.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **early input**. Then $R = x(y).P$, $\beta = xz$ and $R' = P[\{z/y\}^{\mathcal{S}}]$. Moreover, $\widehat{R} = x(y).\widehat{P}$ and $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \widehat{P}\{z/y\}^{\mathcal{S}} = \widehat{P[\{z/y\}^{\mathcal{S}}]} = \widehat{R'}$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **sum**. Then $R = P + Q$ and $P \xrightarrow{\beta} R'$. Now $\widehat{R} = \widehat{P} + \widehat{Q}$. So $\widehat{P} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$ by induction. Hence, by **sum**, $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **symb-match**. Then $R = [x{=}y]P$, $P \xrightarrow{\alpha} R'$ and $\beta = [x{=}y]\alpha$. Now $\widehat{R} = [x{=}y]\widehat{P}$. By induction $\widehat{P} \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$. By **symb-match**, $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **ide**. Then $R = A(\vec{y})$ with $A(\vec{x}) \stackrel{\text{def}}{=} P$ and $P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] \xrightarrow{\beta} R'$. Now $\widehat{R} = \widehat{A}(\vec{y})$ with $\widehat{A}(\vec{x}) \stackrel{\text{def}}{=} \widehat{P}$. Moreover, $\widehat{P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]} = \widehat{P}\{\vec{y}/\vec{x}\}^{\mathcal{S}}$. By induction $\widehat{P}\{\vec{y}/\vec{x}\}^{\mathcal{S}} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$. By **ide**, $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **par**. Then $R = P|Q$, $P \xrightarrow{\beta} P'$, $\mathrm{bn}(\beta) \cap \mathrm{fn}(Q) = \emptyset$ and $R' = P'|Q$. Now $\widehat{R} = \widehat{P}|\widehat{Q}$, $\mathrm{bn}(\beta) \cap \mathrm{fn}(\widehat{Q}) = \emptyset$ and $\widehat{R'} = \widehat{P'}|\widehat{Q}$. By induction $\widehat{P} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$. Thus $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$, by **par**.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **e-s-com**. Then $R = P|Q$, $P \xrightarrow{M\bar{x}y} P'$, $Q \xrightarrow{Nvy} Q'$, $R' = P'|Q'$, and $\beta = [x{=}v]MN\tau$. Now $\widehat{R} = \widehat{P}|\widehat{Q}$ and $\widehat{R'} = \widehat{P'}|\widehat{Q'}$. By induction $\widehat{P} \xrightarrow{M\bar{x}y}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$ and $\widehat{Q} \xrightarrow{Nvy}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{Q'}$. Thus $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$, by **e-s-com**.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **e-s-close**. Then $R = P|Q$, $P \xrightarrow{M\bar{x}(z)} P'$, $Q \xrightarrow{Nvz} Q'$, $R' = (\nu z)(P'|Q')$, $z \notin \mathrm{fn}(Q)$ and $\beta = [x{=}v]MN\tau$. Now $\widehat{R} = \widehat{P}|\widehat{Q}$, $z \notin \mathrm{fn}(\widehat{Q})$ and $\widehat{R'} = (\nu z)(\widehat{P'}|\widehat{Q'})$. By induction $\widehat{P} \xrightarrow{M\bar{x}(z)}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$ and $\widehat{Q} \xrightarrow{Nvz}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{Q'}$. Thus $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$, by **e-s-close**.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **res-alpha**. Then $R = (\nu y)P$, $P[\{z/\!\!/y\}] \xrightarrow{\beta} P'$, $R' = (\nu z)P'$, $z \notin \mathrm{fn}(R)$ and $z \notin \mathrm{n}(\beta)$. Now $\widehat{R} = (\nu y)\widehat{P}$, $z \notin \mathrm{fn}(\widehat{R})$ and $\widehat{R'} = (\nu z)\widehat{P'}$. By induction $\widehat{P}\{z/\!\!/y\} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$. Hence, $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **symb-open-alpha**. Then $R = (\nu y)P$, $\beta = M\bar{x}(z)$, $P \xrightarrow{M\bar{x}y} P'$, $R' = P'[\{z/\!\!/y\}]$, $y \notin x$, $z \notin \mathrm{fn}(R)$ and $y \notin \mathrm{n}(M)$. Now $\widehat{R} = (\nu y)\widehat{P}$, $z \notin \mathrm{fn}(\widehat{R})$ and $\widehat{R'} = \widehat{P'}\{z/\!\!/y\}$. By induction $\widehat{P} \xrightarrow{M\bar{x}y}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$. Hence, $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$ by **symb-open-alpha**.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **relabelling**. Then $R = P[\sigma]$, $P \xrightarrow{\alpha} P'$, $R' = P'[\sigma]$, $\beta = \alpha[\sigma]$ and $\mathrm{bn}(\beta) \cap \mathrm{fn}(R) = \emptyset$. By induction $\widehat{P} \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}$. As $\mathrm{fn}(R) = \mathrm{fn}(\widehat{R}) = \mathrm{fn}(\widehat{P}\sigma)$, by Lemma A.8 $\widehat{R} = \widehat{P}\sigma \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{P'}\sigma = \widehat{R'}$. □

Lemma A.34. If $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$ with $\mathrm{bn}(\beta) \cap \mathrm{fn}(R) = \emptyset$ then equally $R \xrightarrow{\beta} R'$ for some R' with $\widehat{R'} \equiv^{\mathcal{S}} U$.

Proof. By induction on the depth of the inference of $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$, with a nested induction on the number of topmost relabelling operators in R.

- First suppose $R = P[\sigma]$. Then $\widehat{R} = \widehat{P}\sigma$. By Lemma A.11 $\widehat{P} \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} V$ for some α and V with $\alpha[\sigma] = \beta$ and $V\sigma \equiv^{\mathcal{S}} U$. Moreover, the depth of the inference of $\widehat{P} \xrightarrow{\alpha}{}^{\mathcal{S}}_{\bullet} V$ is the same as that of $\widehat{P}\sigma \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$. As $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{fn}(P[\sigma]) = \emptyset$, also $\mathrm{bn}(\alpha) \cap \mathrm{fn}(P) = \emptyset$. So by induction $P \xrightarrow{\alpha} P'$ for some P' with $\widehat{P'} \equiv^{\mathcal{S}} V$. By **relabelling** $R \xrightarrow{\beta} P'[\sigma]$. Furthermore one has $\widehat{P'[\sigma]} = \widehat{P'}\sigma \equiv^{\mathcal{S}} V\sigma \equiv^{\mathcal{S}} U$.
  Henceforth I suppose that R is not of the form $P[\sigma]$.

- Suppose $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$ is derived by rule **E-S-COM**. Then $R = P|Q$, $\widehat{R} = \widehat{P}|\widehat{Q}$, $\beta = [x{=}v]MN\tau$, $\widehat{P} \xrightarrow{M\bar{x}y}{}^{\mathcal{S}}_{\bullet} V$, $\widehat{Q} \xrightarrow{Nvy}{}^{\mathcal{S}}_{\bullet} W$ and $U = V|W$. By induction $P \xrightarrow{M\bar{x}y} P'$ and $Q \xrightarrow{Nvy} Q'$ for some P' and Q' with $\widehat{P'} \equiv^{\mathcal{S}} V$ and $\widehat{Q'} \equiv^{\mathcal{S}} W$. By **E-S-COM** $R \xrightarrow{\beta} P'|Q' \equiv^{\mathcal{S}} V|W = U$.
- Suppose $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$ is derived by rule **RES-ALPHA**. Then $R = (vy)P$, $\widehat{R} = (vy)\widehat{P}$, $\widehat{P}\{z/\!\!/y\} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} V$, $z \notin \text{FN}(\widehat{R})$, $z \notin \text{n}(\beta)$ and $U = (vz)V$. As $\text{bn}(\beta) \cap \text{FN}((vy)P) = \emptyset$ and $z \notin \text{n}(\beta) \cup \text{FN}((vy)P)$, also $\text{bn}(\beta) \cap \text{FN}(P[\{z/\!\!/y\}]) = \emptyset$. So by induction $P[\{z/\!\!/y\}] \xrightarrow{\beta} P'$ for some P' such that $\widehat{P'} \equiv^{\mathcal{S}} V$. Now $R \xrightarrow{\beta} (vz)P'$ by **RES-ALPHA**. Moreover, $\widehat{(vz)P'} = (vz)\widehat{P'} \equiv^{\mathcal{S}} (vz)V = U$.
- The cases that $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$ is derived by **TAU**, **OUTPUT**, **EARLY-INPUT**, **SUM**, **SYMB-MATCH**, **IDE**, **PAR** or **SYMB-OPEN-ALPHA** are also trivial.
- Suppose $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$ is derived by rule **E-S-CLOSE**. Then $R = P|Q$, $\widehat{R} = \widehat{P}|\widehat{Q}$, $\beta = [x{=}v]MN\tau$, $\widehat{P} \xrightarrow{M\bar{x}(z)}{}^{\mathcal{S}}_{\bullet} V$, $\widehat{Q} \xrightarrow{Nvz}{}^{\mathcal{S}}_{\bullet} W$, $z \notin \text{FN}(\widehat{Q})$ and $U = (vz)(V|W)$. Pick $w \notin \text{FN}(\widehat{R}) \cup \text{FN}(U)$. Then $\widehat{P} \xrightarrow{M\bar{x}(w)}{}^{\mathcal{S}}_{\bullet} V' \equiv^{\mathcal{S}} V\{w/z\}$ and $\widehat{Q} \xrightarrow{Nvw}{}^{\mathcal{S}}_{\bullet} W' \equiv^{\mathcal{S}} W\{w/z\}$ by Lemmas A.7 and A.6. Moreover, the inferences of these transitions have a smaller depth than that of $\widehat{R} \xrightarrow{\beta}{}^{\mathcal{S}}_{\bullet} U$. So by induction $P \xrightarrow{M\bar{x}(w)} P'$ and $Q \xrightarrow{Nvw} Q'$ for some P' and Q' with $\widehat{P'} \equiv^{\mathcal{S}} V'$ and $\widehat{Q'} \equiv^{\mathcal{S}} W'$. Hence $R \xrightarrow{\beta} (vw)(P'|Q')$ by **E-S-CLOSE**. Moreover,

$$\widehat{(vw)(P'|Q')} = (vw)(\widehat{P'}|\widehat{Q'}) \equiv^{\mathcal{S}} (vw)(V'|W') \equiv^{\mathcal{S}}$$
$$(vw)((V|W)\{w/z\}) \equiv^{\mathcal{S}} (vz)(V|W) = U. \qquad \square$$

Let $\mathscr{T}_\rho$ be the identity translation from $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$ to $\pi^{\rho\bullet}_{ES}(\mathcal{N})$.

THEOREM A.35. $\mathscr{T}_\rho(P) \mathrel{\dot{\sim}} P$ for any $P \in \text{T}_{\pi^{\mathcal{S}}_{ES}(\mathcal{N})}$.

PROOF. It suffices to show that the symmetric closure of

$$\mathscr{R} := \{(R,U) \mid R \in \text{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})}, U \in \text{T}_{\pi^{\mathcal{S}}_{ES}(\mathcal{N})} \wedge \widehat{R} \equiv^{\mathcal{S}} U\}$$

is a strong barbed bisimulation, as it contains $(\mathscr{T}_\rho(P), P)$.

Let $(R,U) \in \mathscr{R}$. Let $R \xrightarrow{\tau} R'$. Then $\widehat{R} \xrightarrow{\tau}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$ by Lemma A.33 and $U \xrightarrow{\tau}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} \widehat{R'}$ by Corollary A.32. So $U \xrightarrow{\tau}{}^{\mathcal{S}}_{\bullet} U'$ for some U' with $(R',U') \in \mathscr{R}$.

Let $U \xrightarrow{\tau}{}^{\mathcal{S}}_{\bullet} U'$. Then $\widehat{R} \xrightarrow{\tau}{}^{\mathcal{S}}_{\bullet} \equiv^{\mathcal{S}} U'$ by Corollary A.32. By Lemma A.34 $R \xrightarrow{\tau} R'$ for some R' with $\widehat{R'} \equiv^{\mathcal{S}} U'$.

Let $U{\downarrow_b}$ with $b \in \mathcal{Z} \cup \overline{\mathcal{Z}}$. Then $U \xrightarrow{by}{}^{\mathcal{S}}_{\bullet} U'$ or $U \xrightarrow{b(y)}{}^{\mathcal{S}}_{\bullet} U'$ for some y and U', using the definition of O in Section 4. Hence $\widehat{R} \xrightarrow{by}{}^{\mathcal{S}}_{\bullet}$ or $\widehat{R} \xrightarrow{b(y)}{}^{\mathcal{S}}_{\bullet}$ by Corollary A.32. In the second case, by Lemma A.7 I may assume, without loss of generality, that $y \notin \text{FN}(\widehat{R})$. So $R \xrightarrow{by}{}^{\mathcal{S}}_{ES}$ or $R \xrightarrow{b(y)}{}^{\mathcal{S}}_{ES}$ by Lemma A.34. Thus $R{\downarrow_b}$.

The implication $R{\downarrow_b} \Rightarrow U{\downarrow_b}$ proceeds likewise. $\square$

The following example shows why the side condition $\text{bn}(\alpha[\sigma]) \cap \text{FN}(P[\sigma]) = \emptyset$ in rule **RELABELLING** is necessary.

Example A.36. Let $P = \bar{x}z.x(w).\bar{w}q \mid x(y).(vv)\bar{x}v.y(r)$. In the π-calculus, or in $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$, this process can do two successive τ-steps in a row, but not three:

$$P \xrightarrow{\tau} x(w).\bar{w}q \mid (vv)\bar{x}v.z(r) \xrightarrow{\tau} (vu)(\bar{u}q \mid z(r)) \not\xrightarrow{\tau}.$$

In the above expression the bound name u may be chosen freely from $\mathcal{N}\backslash\{x,z,q\}$. The side condition of rule **SYMB-OPEN-ALPHA** prevents the choices $u = x, z$, since x and z are free in $(vv)\bar{x}(v).z(r)$. The side condition of (the symmetric counterpart of) rule **E-S-CLOSE** prevents the choices $u = x, q$.

In a version of $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ in which rule **relabelling** does not have the side condition $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{fn}(P[\sigma]) = \emptyset$, and **e-s-close** lacks the side condition $z \in \mathrm{fn}(P)$, one obtains $(\nu v)\bar{x}v.y(r) \xrightarrow{\bar{x}(z)} y(r)[z/\!\!/v]$ and thus

$$\big((\nu v)\bar{x}v.y(r)\big)[z/y] \xrightarrow{\bar{x}(z)} y(r)[z/\!\!/v][z/y] \xrightarrow{zq} \mathbf{0}[z/\!\!/v][z/y][q/r]$$

which yields

$$\begin{array}{rcl} P & \xrightarrow{\tau} & x(w).\bar{w}q \mid \big((\nu v)\bar{x}v.y(r)\big)[z/y] \\ & \xrightarrow{\tau} & (\nu z)(\bar{w}q[z/w] \mid y(r)[z/\!\!/v][z/y]) \\ & \xrightarrow{\tau} & (\nu z)(\mathbf{0}[z/w] \mid \mathbf{0}[z/\!\!/v][z/y][q/r]), \end{array}$$

in violation of Theorem A.35.

Although $(\nu v)\bar{x}v.y(r) \xrightarrow{\bar{x}(z)}_{\bullet}^{\mathcal{S}} y(r)\{z/\!\!/v\}$, one has $\big((\nu v)\bar{x}v.y(r)\big)\{z/y\} \xrightarrow{\bar{x}(z)}_{\bullet}^{\mathcal{S}} y(r)\{z/\!\!/v\}\{z/y\}$. This shows that the necessity of the side condition $\mathrm{bn}(\alpha[\sigma]) \cap \mathrm{fn}(R\sigma) = \emptyset$ in Lemma A.4/A.8. Hence $\big((\nu v)\bar{x}v.y(r)\big)[z/y] \xrightarrow{\bar{x}(z)} y(r)[z/\!\!/v][z/y]$ violates Lemma A.33. Since rule **symb-open-alpha** is invoked *before* the relabelling $[z/y]$ is applied—in contrast with $\pi^{\mathcal{S}}_{ES}(\mathcal{N})$, where it is invoked after applying the substitution $\{z/y\}$—the choice $u = z$ cannot be avoided. Here the side condition of **relabelling** comes to the rescue; it rules out the offending transition because $z \in \mathrm{bn}(\bar{x}(z)) \cap \mathrm{fn}(\big((\nu v)\bar{x}v.y(r)\big)[z/y])$.

A.5 α-conversion for $\pi^{\rho\bullet}_{ES}(\mathcal{N})$

For $\pi^{\rho\bullet}_{ES}(\mathcal{N})$-processes P and Q write $P \equiv Q$ iff $\widehat{P} \equiv^{\mathcal{S}} \widehat{Q}$. If $P \equiv Q$ then $\mathrm{fn}(P) = \mathrm{fn}(\widehat{P}) = \mathrm{fn}(\widehat{Q}) = \mathrm{fn}(Q)$.

Lemma A.37. If $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ with $\mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$ then equally $Q \xrightarrow{\alpha} Q'$ for some Q' with $P' \equiv Q'$.

Proof. Suppose $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ with $\mathrm{bn}(\alpha) \cap \mathrm{fn}(P) = \emptyset$. Then $\widehat{P} \equiv^{\mathcal{S}} \widehat{Q}$, and $\widehat{P} \xrightarrow{\alpha}_{\bullet}^{\mathcal{S}} \equiv^{\mathcal{S}} \widehat{P'}$ by Lemma A.33. Therefore, $\widehat{Q} \xrightarrow{\alpha}_{\bullet}^{\mathcal{S}} U$ for some $U \equiv^{\mathcal{S}} \widehat{P'}$ by Corollary A.32. By Lemma A.34 $Q \xrightarrow{\alpha} Q'$ for some Q' with $\widehat{Q'} \equiv^{\mathcal{S}} U$. Now $P' \equiv Q'$. □

Lemma A.38.

(a) $(\forall x \in \mathrm{fn}(P).\, x[\sigma] = x[\sigma']) \Rightarrow P[\sigma] \equiv P[\sigma']$,
(b) $P[\sigma_1][\sigma_2] \equiv P[\sigma_2 \circ \sigma_1]$,
(c) $P[\epsilon] \equiv P$, where ϵ is the empty substitution,
(d) $(P|Q)[\sigma] \equiv P[\sigma] \mid Q[\sigma]$,
(e) $((\nu y)P)[\sigma] \equiv (\nu y)(P[\sigma])$, when $y \notin dom(\sigma) \cup range(\sigma)$,
(f) $(\nu y)P \equiv (\nu z)(P[\{z/\!\!/y\}])$, when $z \notin \mathrm{fn}((\nu y)P)$,
(g) if $P_1 \equiv Q_1$ and $P_2 \equiv Q_2$ then $P_1|P_2 \equiv Q_1|Q_2$, and
(h) if $P \equiv Q$ then $(\nu y)P \equiv (\nu y)Q$ and $P[\sigma] \equiv Q[\sigma]$.

Proof. (a), (b) and (c) follow immediately from (12), (11) and (15), (d) and (e) from the definition of substitution, (g) and (h) from the congruence property of $\equiv^{\mathcal{S}}$ and (13), and (f) follows from (10), using (a) and (h) to replace $\{z/y\}$ by $\{z/\!\!/y\}$. □

The following five lemmas are counterparts for $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ of Lemmas A.1, A.2, A.7, A.6 and A.5 for $\pi_{ES}(\mathcal{N})$, adapted to avoid non-surjective substitutions $\{w/z\}$.

Lemma A.39. If $P \xrightarrow{\alpha} Q$ then $\mathrm{n}(\alpha)\backslash(\iota(\alpha) \cup \mathrm{bn}(\alpha)) \subseteq \mathrm{fn}(P)$.

Proof. Immediately from Lemmas A.33, A.30 and A.1. □

Table 10. Structural operational semantics of the π-calculus with relabelling

TAU: $M\tau.P \xrightarrow{M\tau} P$	**OUTPUT:** $M\bar{x}y.P \xrightarrow{M\bar{x}y} P$	**EARLY-INPUT:** $Mx(y).P \xrightarrow{Mxz} P[\{z/y\}^{\mathcal{S}}]$
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:** $\dfrac{P \xrightarrow{\alpha} P'}{A(\vec{x}) \xrightarrow{\alpha} P'}\ (A(\vec{x}) \stackrel{\text{def}}{=} P)$
PAR: $\dfrac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}\ \begin{pmatrix}\text{bn}(\alpha)\cap \text{FN}(P)=\emptyset\\ \text{bn}(\alpha)\cap \text{FN}(Q)=\emptyset\end{pmatrix}$	**E-S-COM:** $\dfrac{P \xrightarrow{M\bar{x}y} P',\ Q \xrightarrow{Nvy} Q'}{P\|Q \xrightarrow{[x=v]MN\tau} P'\|Q'}$	**E-S-CLOSE:** $\dfrac{P \xrightarrow{M\bar{x}(z)} P',\ Q \xrightarrow{Nvz} Q'}{P\|Q \xrightarrow{[x=v]MN\tau} (\nu z)(P'\|Q')}\ \begin{pmatrix}z\notin \text{FN}(P)\\ z\notin \text{FN}(Q)\end{pmatrix}$
RES-ALPHA: $\dfrac{P[\{z/\!\!/y\}] \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu z)P'}\ \begin{pmatrix}z\notin \text{FN}((\nu y)P)\\ z\notin \text{n}(\alpha)\end{pmatrix}$	**SYMB-OPEN-ALPHA:** $\dfrac{P \xrightarrow{M\bar{x}y} P'}{(\nu y)P \xrightarrow{M\bar{x}(z)} P'[\{z/\!\!/y\}]}\ \begin{pmatrix}y\neq x\\ z\notin \text{FN}((\nu y)P)\\ y\notin \text{n}(M)\end{pmatrix}$	**RELABELLING:** $\dfrac{P \xrightarrow{\alpha} P'}{P[\sigma] \xrightarrow{\alpha[\sigma]} P'[\sigma]}\ \begin{pmatrix}\text{bn}(\alpha[\sigma])\cap\\ \text{FN}(P[\sigma])=\emptyset\end{pmatrix}$

Lemma A.40. If $P \xrightarrow{\alpha} Q$ then $\text{FN}(Q) \subseteq \text{FN}(P) \cup \iota(\alpha) \cup \text{bn}(\alpha)$.

Proof. Immediately from Lemmas A.33, A.30 and A.2. □

Lemma A.41. If $R \xrightarrow{M\bar{x}(z)} R_z$ and $w \notin \text{FN}(R)$ then equally $R \xrightarrow{M\bar{x}(w)} \equiv R_z[\{w/\!\!/z\}]$.

Proof. Let $R \xrightarrow{M\bar{x}(z)} R_z$. Then $\widehat{R} \xrightarrow{M\bar{x}(z)}_{\bullet}^{\mathcal{S}} \equiv^{\mathcal{S}} \widehat{R}_z$ by Lemma A.33. So $\widehat{R} \xrightarrow{M\bar{x}(z)}_{ES}^{\mathcal{S}} U \equiv^{\mathcal{S}} \widehat{R}_z$ by Lemma A.30. Using that $w \notin \text{FN}(R) = \text{FN}(\widehat{R})$, by Lemma A.7 $\widehat{R} \xrightarrow{M\bar{x}(w)}_{ES}^{\mathcal{S}} V$ for a $V \equiv^{\mathcal{S}} U\{w/z\}$. By Lemma A.2, $\text{FN}(U)\backslash\{z\} \subseteq \text{FN}(\widehat{R}) \not\ni w$, and therefore $U\{w/\!\!/z\} \equiv^{\mathcal{S}} U\{w/z\}$ by (12). By Lemma A.31 $\widehat{R} \xrightarrow{M\bar{x}(w)}_{\bullet}^{\mathcal{S}} W$ for a $W \equiv^{\mathcal{S}} V$. By Lemma A.34, using that $w \notin \text{FN}(R)$, $R \xrightarrow{M\bar{x}(w)} R'$ for an R' with $\widehat{R'} \equiv^{\mathcal{S}} W$. Using (13) $\widehat{R'} \equiv^{\mathcal{S}} \widehat{R}_z\{w/\!\!/z\} = \widehat{R_z[\{w/\!\!/z\}]}$, and thus $R' \equiv R_z[\{w/\!\!/z\}]$. □

Lemma A.42. If $R \xrightarrow{Mxz} R_z$ and $z, w \notin \text{FN}(R)$, then equally $R \xrightarrow{Mxw} R_w$ for some R_w with $R_w \equiv R_z[\{w/\!\!/z\}]$.

Proof. The proof is the same as the one of Lemma A.41, but using Lemma A.6 instead of Lemma A.7, thereby also using the precondition $z \notin \text{FN}(R)$. □

Lemma A.43. If $R \xrightarrow{Mxz} R_z$ and w is a name, then equally $R \xrightarrow{Mxw} R_w$ for some R_w such that $R_w[\sigma] \equiv R_z[\sigma]$ for each surjective substitution σ with $z[\sigma] = w[\sigma]$.

Proof. Let $R \xrightarrow{Mxz} R_z$. Then $\widehat{R} \xrightarrow{Mxz}_{\bullet}^{\mathcal{S}} \equiv^{\mathcal{S}} \widehat{R}_z$ by Lemma A.33. So $\widehat{R} \xrightarrow{Mxz}_{ES}^{\mathcal{S}} U \equiv^{\mathcal{S}} \widehat{R}_z$ by Lemma A.30. Pick $v \notin \text{FN}(\widehat{R})$. By Lemma A.5 $\widehat{R} \xrightarrow{Mxv}_{ES}^{\mathcal{S}} V$ for some V with $U \equiv^{\mathcal{S}} V\{z/v\}$, and by Lemma A.6 $\widehat{R} \xrightarrow{Mxw}_{ES}^{\mathcal{S}} W$ for some W with $W \equiv^{\mathcal{S}} V\{w/v\}$. By Lemma A.31 $\widehat{R} \xrightarrow{Mxw}_{\bullet}^{\mathcal{S}} W^{\dagger}$ for some $W^{\dagger} \equiv^{\mathcal{S}} W$. By Lemma A.34 $R \xrightarrow{Mxw} R_w$ for some R_w with $\widehat{R}_w \equiv^{\mathcal{S}} W^{\dagger}$.

Now let σ be a surjective substitution with $z[\sigma] = w[\sigma]$. Then $\widehat{R_w[\sigma]} = \widehat{R}_w\sigma \equiv^{\mathcal{S}} V\{w/v\}\sigma = V\{z/v\}\sigma \equiv^{\mathcal{S}} \widehat{R}_z\sigma \equiv^{\mathcal{S}} \widehat{R_z[\sigma]}$ by(11), (12) and (13), so $R_w[\sigma] \equiv R_z[\sigma]$. □

A.6 Agent identifiers without parameters

Let $\pi^{\rho}_{ES}(\mathcal{N})$ be the variant of $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ in which agent identifiers may be called only with their own declared names as parameters, i.e. such that $\vec{y} = \vec{x}$ in rule IDE. As a result, the relabelling operator $[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$ in this rule can be dropped. The operational semantics of $\pi^{\rho}_{ES}(\mathcal{N})$ is displayed in Table 10.

Let $\mathscr{T}_r\colon \mathrm{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})} \to \mathrm{T}_{\pi^{\rho}_{ES}(\mathcal{N})}$ be the compositional translation satisfying $\mathscr{T}_r(A(\vec{y})) := A_r(\vec{x})[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$, and acting homomorphically on all other operators. Here A_r is a fresh agent identifier with defining equation $A_r(\vec{x}) \stackrel{\text{def}}{=} \mathscr{T}_r(P)$ when $A(\vec{x}) \stackrel{\text{def}}{=} P$ was the defining equation of A. Clearly $\textsc{fn}(\mathscr{T}_r(P)) = \textsc{fn}(P)$. I will show that $\mathscr{T}_r$ is a valid translation from $\pi^{\rho\bullet}_{ES}(\mathcal{N})$ to $\pi^{\rho}_{ES}(\mathcal{N})$, up to strong barbed bisimilarity.

Lemma A.44. If $R \xrightarrow{\beta} R'$ and $\mathrm{bn}(\beta) \cap \textsc{fn}(R) = \emptyset$ then $\mathscr{T}_r(R) \xrightarrow{\beta} \mathscr{T}_r(R')$.

Proof. By induction of the inference of $R \xrightarrow{\beta} R'$.

- Suppose $R \xrightarrow{\beta} R'$ is derived by **early input**. Then $R = x(y).P$, $\beta = xz$ and $R' = P[\{z/y\}^{\mathcal{S}}]$. Now $\mathscr{T}_r(R) = x(y).\mathscr{T}_r(P) \xrightarrow{\beta} \mathscr{T}_r(P)[\{z/y\}^{\mathcal{S}}] = \mathscr{T}_r(R')$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **ide**. Then $R = A(\vec{y})$, $A(\vec{x}) \stackrel{\text{def}}{=} P$ and $P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] \xrightarrow{\beta} R'$. As $\mathrm{bn}(\beta) \cap \textsc{fn}(R) = \emptyset$ and $\textsc{fn}(P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) \subseteq \textsc{fn}(R)$, $\mathrm{bn}(\beta) \cap \textsc{fn}(P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) = \emptyset$. By induction
$$\mathscr{T}_r(P)[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = \mathscr{T}_r(P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) \xrightarrow{\beta} \mathscr{T}_r(R').$$
so by **relabelling** $\mathscr{T}_r(P) \xrightarrow{\alpha} P'$ for some α and P' with $\alpha[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = \beta$ and $P'[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = \mathscr{T}_r(R')$. By rule **ide** from $\pi^{\rho}_{ES}(\mathcal{N})$, $A_r(\vec{x}) \xrightarrow{\alpha} P'$. So by **relabelling**
$$\mathscr{T}_r(R) = A_r(\vec{x})[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] \xrightarrow{\beta} P'[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = \mathscr{T}_r(R'),$$
here using the side condition that $\mathrm{bn}(\beta) \cap \textsc{fn}(R) = \emptyset$.
- Using that $\textsc{fn}(\mathscr{T}_r(Q)) = \textsc{fn}(Q)$ for all Q, all remaining cases are trivial. □

Lemma A.45. If $\mathscr{T}_r(R) \xrightarrow{\beta} U$ then $R \xrightarrow{\beta} R'$ for some R' with $\mathscr{T}_r(R') = U$.

Proof. By induction of the inference of $\mathscr{T}_r(R) \xrightarrow{\beta} U$.

- Suppose $R = A(\vec{y})$. Then $\mathscr{T}_r(R) = A_r(\vec{x})[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$. By **relabelling** $\mathrm{bn}(\beta) \cap \textsc{fn}(\mathscr{T}_r(R)) = \emptyset$ and $A_r(\vec{x}) \xrightarrow{\alpha} V$ for some α and V with $\alpha[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = \beta$ and $V[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = U$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$, so that $A_r(\vec{x}) \stackrel{\text{def}}{=} \mathscr{T}_r(P)$. Via rule **ide** $\mathscr{T}_r(P) \xrightarrow{\alpha} V$. By induction $P \xrightarrow{\alpha} P'$ for some P' with $\mathscr{T}_r(P') = V$. As $\textsc{fn}(P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) \subseteq \textsc{fn}(R) = \textsc{fn}(\mathscr{T}_r(R))$, $\mathrm{bn}(\beta) \cap \textsc{fn}(P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) = \emptyset$. So by **relabelling** $P[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] \xrightarrow{\beta} P'[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$. By rule **ide** from $\pi^{\rho\bullet}_{ES}(\mathcal{N})$, $R \xrightarrow{\beta} P'[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$. Moreover,
$$\mathscr{T}_r(P'[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]) = \mathscr{T}_r(P')[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = V[\{\vec{y}/\vec{x}\}^{\mathcal{S}}] = U.$$
- Using that $\textsc{fn}(\mathscr{T}_r(Q)) = \textsc{fn}(Q)$ for all Q, all other cases are trivial. □

Theorem A.46. $\mathscr{T}_r(P) \mathrel{\dot{\sim}} P$ for any $P \in \mathrm{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})}$.

Proof. It suffices to show that the symmetric closure of
$$\mathscr{R} := \{(R, \mathscr{T}_r(R)) \mid R \in \mathrm{T}_{\pi^{\rho\bullet}_{ES}(\mathcal{N})}\}$$
is a strong barbed bisimulation. This follows from Lemmas A.44 and A.45, also using Lemma A.41 to handle the barbs from R. □

A.7 Clash-free processes

For $\mathcal{R}$ a collection of *private names*, disjoint with $\mathcal{N} \uplus \mathcal{S} =: \mathcal{Z}$, I define $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$ as the variant of $\pi^{\rho}_{ES}(\mathcal{N} \uplus \mathcal{R})$ where in all expressions $x(y).Q$ the name y must be chosen from $\mathcal{N}$, and in defining equations $A(\vec{x}) \stackrel{\text{def}}{=} P$ one has $x_1, \ldots, x_n \in \mathcal{N}$. The semantics of $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$ equals that of $\pi^{\rho}_{ES}(\mathcal{N} \uplus \mathcal{R})$.

Clearly, the relation $\equiv$ and all results from Section A.5 are inherited by $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$.

Definition A.47. The set $\text{RN}(P)$ of *restriction-bound* names of a $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ process P is inductively defined by

- if $R = (\nu y)P$ then $\text{RN}(R) \supseteq \text{RN}(P) \cup \{y\}$,
- if $R = \tau.P$ or $\bar{x}y.P$ or $x(y).P$ then $\text{RN}(R) \supseteq \text{RN}(P)$,
- if $R = [x{=}y]P$ then $\text{RN}(R) \supseteq \text{RN}(P)$,
- if $R = P[\sigma]$ then $\text{RN}(R) \supseteq \{x[\sigma] \mid x \in \text{RN}(P)\}$,
- if $R = P + Q$ or $P|Q$ then $\text{RN}(R) \supseteq \text{RN}(P) \cup \text{RN}(Q)$,
- if $R = A(\vec{x})$ and $A(\vec{x}) \stackrel{\text{def}}{=} P$ then $\text{RN}(R) \supseteq \text{RN}(P)$.

Each statement $y \in \text{RN}(R)$ can be proven using $y \in \text{RN}((\nu y)P)$ as an axiom, and each of the six clauses above as proof rules. It follows that in fact one has = wherever $\supseteq$ is written in Definition A.47.

Definition A.48. The collection $\mathscr{C}$ of *clashing* $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ processes is the smallest such that

(i) if $\text{FN}(R) \cap \text{RN}(R) \neq \emptyset$ then $R \in \mathscr{C}$,
(ii) if $R = P|Q$ and $\text{RN}(P) \cap \text{RN}(Q) \neq \emptyset$ then $R \in \mathscr{C}$,
(iii) if $R = (\nu y)P$ and $y \in \text{RN}(P)$ then $R \in \mathscr{C}$,
(iv) if $R = P[\sigma]$ and $y[\sigma] = z[\sigma]$ for different $y, z \in \text{RN}(P)$ then $R \in \mathscr{C}$,
(v) if $P \in \mathscr{C}$ then also $\tau.P \in \mathscr{C}$, $\bar{x}y.P \in \mathscr{C}$, $x(y).P \in \mathscr{C}$, $[x{=}y]P \in \mathscr{C}$, $(\nu y)P \in \mathscr{C}$ and $P[\sigma] \in \mathscr{C}$,
(vi) if $P \in \mathscr{C}$ or $Q \in \mathscr{C}$ then also $P + Q \in \mathscr{C}$ and $P|Q \in \mathscr{C}$,
(vii) if $A(\vec{x}) \stackrel{\text{def}}{=} P$ and $P \in \mathscr{C}$ then also $A(\vec{x}) \in \mathscr{C}$,
(viii) and if $\text{RN}(R) \cap \mathcal{Z} \neq \emptyset$ then $R \in \mathscr{C}$.

A process P is *clash-free* if $P \notin \mathscr{C}$.

Now take $\mathcal{R}$ as given in Section 9, where also the relabelling operators $[\ell]$, $[r]$, $[e]$ and $[p_y]$ are defined, with $p_y = \{p/\!\!/y\} \circ e$. Let the translation $\mathscr{T}_{\text{cf}} : \text{T}_{\pi^{\rho}_{ES}(\mathcal{N})} \to \text{T}_{\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})}$ satisfy

$$\begin{aligned}\mathscr{T}_{\text{cf}}((\nu y)P) &:= (\nu p)(\mathscr{T}_{\text{cf}}(P)[p_y])\\ \mathscr{T}_{\text{cf}}(P \mid Q) &:= \mathscr{T}_{\text{cf}}(P)[\ell] \parallel \mathscr{T}_{\text{cf}}(Q)[r]\\ \mathscr{T}_{\text{cf}}(A(\vec{x})) &:= A_{\text{cf}}(\vec{x})\end{aligned}$$

and act homomorphically on all other operators. Here A_{cf} is a fresh agent identifier with defining equation $A_{\text{cf}}(\vec{x}) \stackrel{\text{def}}{=} \mathscr{T}_{\text{cf}}(P)$ when $A(\vec{x}) \stackrel{\text{def}}{=} P$ was the defining equation of A.

Lemma A.49. Let $P \in \text{T}_{\pi^{\rho}_{ES}(\mathcal{N})}$ be a $\pi^{\rho}_{ES}(\mathcal{N})$ process. Then $\text{FN}(\mathscr{T}_{\text{cf}}(P)) = \text{FN}(P) \subseteq \mathcal{Z}$ and $\mathscr{T}_{\text{cf}}(P)$ is clash-free.

Proof. The first statement follows by a straightforward structural induction on P.

Let $\mathcal{R}_0 := \{\varsigma p \mid \varsigma \in \{e, \ell, r\}^*\}$; these are the names $\varsigma p^{\upsilon} \in \mathcal{R}$ with $\upsilon = \varepsilon$. A straightforward induction on the derivation of $y \in \text{RN}(\mathscr{T}_{\text{cf}}(P))$ from the rules of Definition A.47 shows that $\text{RN}(\mathscr{T}_{\text{cf}}(P)) \subseteq \mathcal{R}_0$.

Using these facts to handle Requirements (i) and (viii), a straightforward induction on the derivation of $\mathscr{T}_{\text{cf}}(P) \in \mathscr{C}$ from the rules of Definition A.48 leads to a contradiction. For (ii) use that $\forall y, z \in \mathcal{R}_0.\ y[\ell] \neq z[r]$. For (iii) use that $z[p_y] \neq p$ for all $z \in \mathcal{R}_0$. For (iv) use that p_y, ℓ and r are injective substitutions, whereas any relabelling operator $[\sigma]$ inherited from $\pi^{\rho}_{ES}(\mathcal{N})$ satisfies $dom(\sigma) \cap \mathcal{R} = \emptyset$. □

I will show that $\mathscr{T}_{\text{cf}}$ is a valid translation, up to strong barbed bisimilarity.

Lemma A.50. $\mathscr{T}_{\text{cf}}((\nu y)P) \equiv (\nu y)\mathscr{T}_{\text{cf}}(P)$ for $P \in \text{T}_{\pi^{\rho}_{ES}(\mathcal{N})}$.

Proof. By definition $\mathscr{T}_{\text{cf}}((\nu y)P) = (\nu p)(\mathscr{T}_{\text{cf}}(P)[p_y])$. Since $\text{FN}(\mathscr{T}_{\text{cf}}(P)) = \text{FN}(P) \subseteq \mathcal{Z}$ using Lemma A.49, $y \notin \text{FN}(\mathscr{T}_{\text{cf}}(P)[p_y])$. Thus, by Lemma A.38(f,b,a,c,h),

$$(\nu p)(\mathscr{T}_{\text{cf}}(P)[p_y]) \equiv (\nu y)(\mathscr{T}_{\text{cf}}(P)[p_y][\{y/\!\!/p\}]) \equiv (\nu y)\mathscr{T}_{\text{cf}}(P).$$ □

Lemma A.51. If $R \xrightarrow{\beta} R'$ then $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{\beta} \equiv \mathscr{T}_{\mathrm{cf}}(R')$.

Proof. By induction of the inference of $R \xrightarrow{\beta} R'$. Since R is a $\pi^{\rho}_{ES}(\mathcal{N})$ process, $\mathrm{n}(\beta) \cup \mathrm{n}(R) \subseteq \mathcal{Z}$.

- Suppose $R \xrightarrow{\beta} R'$ is derived by **early input**. Then $R = x(y).P$, $\beta = xz$ and $R' = P[\{z/y\}^{\mathcal{S}}]$. Now $\mathscr{T}_{\mathrm{cf}}(R) = x(y).\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\beta} \mathscr{T}_{\mathrm{cf}}(P)[\{z/y\}^{\mathcal{S}}] = \mathscr{T}_{\mathrm{cf}}(R')$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **ide**. Then $R = A(\vec{x})$. Let $A(\vec{x}) \stackrel{\mathrm{def}}{=} P$, so $A_{\mathrm{cf}}(\vec{x}) \stackrel{\mathrm{def}}{=} \mathscr{T}_{\mathrm{cf}}(P)$. Then $P \xrightarrow{\beta} R'$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\beta} \equiv \mathscr{T}_{\mathrm{cf}}(R')$. Applying rule **ide**, $\mathscr{T}_{\mathrm{cf}}(R) = A_{\mathrm{cf}}(\vec{x}) \xrightarrow{\beta} \equiv \mathscr{T}_{\mathrm{cf}}(R')$.
- The cases that $R \xrightarrow{\beta} R'$ is derived by **tau**, **output**, **sum** or **symb-match** are (also) trivial.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **par**. Then $R = P|Q$, $P \xrightarrow{\beta} P'$, $\mathrm{bn}(\beta) \cap \mathrm{fn}(R) = \emptyset$ and $R' = P'|Q$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\beta} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$. As $\mathrm{n}(\beta) \subseteq \mathcal{Z}$, $\beta[\ell] = \beta$. Moreover, $\mathcal{Z} \supseteq \mathrm{fn}(P) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(P)) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(P)[\ell])$ by Lemma A.49, so $\mathrm{bn}(\beta) \cap \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(P)[\ell]) = \emptyset$. Thus, $\mathscr{T}_{\mathrm{cf}}(P)[\ell] \xrightarrow{\beta} V[\ell] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\ell]$ by rule **relabelling** and Lemma A.38(h). Since $\mathcal{Z} \supseteq \mathrm{fn}(Q) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(Q)) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(Q)[r])$ by Lemma A.49, $\mathrm{bn}(\beta) \cap \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(Q)[r]) = \emptyset$. Hence, by rule **par**,

  $$\mathscr{T}_{\mathrm{cf}}(R) = \mathscr{T}_{\mathrm{cf}}(P)[\ell]|\mathscr{T}_{\mathrm{cf}}(Q)[r] \xrightarrow{\beta} V[\ell]|\mathscr{T}_{\mathrm{cf}}(Q)[r] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\ell]|\mathscr{T}_{\mathrm{cf}}(Q)[r] = \mathscr{T}_{\mathrm{cf}}(R'),$$

  in the $\equiv$-step using Lemma A.38(g).
- Suppose $R \xrightarrow{\beta} R'$ is derived by **e-s-com**. Then $R = P|Q$, $P \xrightarrow{M\bar{x}y} P'$, $Q \xrightarrow{Nvy} Q'$, $R' = P'|Q'$, and $\beta = [x{=}v]MN\tau$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{M\bar{x}y} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$ and $\mathscr{T}_{\mathrm{cf}}(Q) \xrightarrow{Nvy} W \equiv \mathscr{T}_{\mathrm{cf}}(Q')$. By rule **relabelling** and Lemma A.38(h) $\mathscr{T}_{\mathrm{cf}}(P)[\ell] \xrightarrow{M\bar{x}y} V[\ell] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\ell]$ and $\mathscr{T}_{\mathrm{cf}}(Q)[r] \xrightarrow{Nvy} W[r] \equiv \mathscr{T}_{\mathrm{cf}}(Q')[r]$. Thus, by **e-s-com** and Lemma A.38(g) $\mathscr{T}_{\mathrm{cf}}(R) = \mathscr{T}_{\mathrm{cf}}(P)[\ell]|\mathscr{T}_{\mathrm{cf}}(Q)[r] \xrightarrow{\beta} V[\ell]|W[r] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\ell]|\mathscr{T}_{\mathrm{cf}}(Q')[r] = \mathscr{T}_{\mathrm{cf}}(R')$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **e-s-close**. Then $R = P|Q$, $P \xrightarrow{M\bar{x}(z)} P'$, $Q \xrightarrow{Nvz} Q'$, $z \notin \mathrm{fn}(R)$, $\beta = [x{=}v]MN\tau$ and $R' = (vz)(P'|Q')$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{M\bar{x}(z)} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$ and $\mathscr{T}_{\mathrm{cf}}(Q) \xrightarrow{Nvz} W \equiv \mathscr{T}_{\mathrm{cf}}(Q')$. Since $z \notin \mathrm{fn}(P) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(P)) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(P)[\ell])$, by rule **relabelling** and Lemma A.38(h) one obtains $\mathscr{T}_{\mathrm{cf}}(P)[\ell] \xrightarrow{M\bar{x}(z)} V[\ell] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\ell]$ and likewise $\mathscr{T}_{\mathrm{cf}}(Q)[r] \xrightarrow{Nvz} W[r] \equiv \mathscr{T}_{\mathrm{cf}}(Q')[r]$. Thus, by **e-s-close** and Lemma A.38(g,h) $\mathscr{T}_{\mathrm{cf}}(R) = \mathscr{T}_{\mathrm{cf}}(P)[\ell]|\mathscr{T}_{\mathrm{cf}}(Q)[r] \xrightarrow{\beta} (vz)(V[\ell]|W[r]) \equiv (vz)(\mathscr{T}_{\mathrm{cf}}(P')[\ell]|\mathscr{T}_{\mathrm{cf}}(Q')[r]) \equiv \mathscr{T}_{\mathrm{cf}}(R')$. The last step follows by Lemma A.50.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **res-alpha**. Then $R = (vy)P$, $P[\{z/\!\!/y\}] \xrightarrow{\beta} P'$, $R' = (vz)P'$, $z \notin \mathrm{fn}(R)$ and $z \notin \mathrm{n}(\beta)$. By induction $\mathscr{T}_{\mathrm{cf}}(P[\{z/\!\!/y\}]) \xrightarrow{\beta} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$. By Lemma A.49 $z \notin \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(R))$. Using this, Lemma A.50, **res-alpha**, Lemma A.38(h), and the observation that $\mathscr{T}_{\mathrm{cf}}(P[\{z/\!\!/y\}]) = \mathscr{T}_{\mathrm{cf}}(P)[\{z/\!\!/y\}]$, one obtains $\mathscr{T}_{\mathrm{cf}}(R) \equiv (vy)\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\beta} (vz)V \equiv (vz)\mathscr{T}_{\mathrm{cf}}(P') \equiv \mathscr{T}_{\mathrm{cf}}(R')$. As $P[\{z/\!\!/y\}] \xrightarrow{\beta} P'$ must be derived by rule **relabelling**, $\mathrm{bn}(\beta) \cap \mathrm{fn}(P[\{z/\!\!/y\}]) = \emptyset$. Since $z \notin \mathrm{bn}(\beta)$ this implies $\mathrm{bn}(\beta) \cap \mathrm{fn}(R) = \emptyset$. Using this, Lemma A.37 yields $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{\beta} \equiv \mathscr{T}_{\mathrm{cf}}(R')$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **symb-open-alpha**. Then $R = (vy)P$, $\beta = M\bar{x}(z)$, $P \xrightarrow{M\bar{x}y} P'$, $y \neq x$, $z \notin \mathrm{fn}(R)$, $y \notin \mathrm{n}(M)$ and $R' = P'[\{z/\!\!/y\}]$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{M\bar{x}y} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$. By Lemma A.50 $\mathscr{T}_{\mathrm{cf}}(R) \equiv (vy)\mathscr{T}_{\mathrm{cf}}(P)$. By Lemma A.49 $z \notin \mathrm{fn}(R) = \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(R)) = \mathrm{fn}((vy)\mathscr{T}_{\mathrm{cf}}(P))$. So by rule **symb-open-alpha** $(vy)\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{M\bar{x}(z)} V[\{z/\!\!/y\}]$. By Lemma A.37 $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{M\bar{x}(z)} \equiv V[\{z/\!\!/y\}]$. Moreover, by Lemma A.38(h), $V[\{z/\!\!/y\}] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\{z/\!\!/y\}] = \mathscr{T}_{\mathrm{cf}}(R')$.
- Suppose $R \xrightarrow{\beta} R'$ is derived by **relabelling**. Then $R = P[\sigma]$, $P \xrightarrow{\alpha} P'$, $\alpha[\sigma] = \beta$, $\mathrm{bn}(\beta) \cap \mathrm{fn}(R) = \emptyset$ and $R' = P'[\sigma]$. By induction $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\alpha} V \equiv \mathscr{T}_{\mathrm{cf}}(P')$. As $\mathrm{bn}(\beta) \cap \mathrm{fn}(\mathscr{T}_{\mathrm{cf}}(R)) = \emptyset$, by **relabelling** and Lemma A.38(h) $\mathscr{T}_{\mathrm{cf}}(R) = \mathscr{T}_{\mathrm{cf}}(P)[\sigma] \xrightarrow{\beta} V[\sigma] \equiv \mathscr{T}_{\mathrm{cf}}(P')[\sigma] = \mathscr{T}_{\mathrm{cf}}(R')$. □

Lemma A.52. If $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ with $\iota(\beta)\cup{\rm bn}(\beta)\subseteq\mathcal{Z}$ then $R \xrightarrow{\beta} R'$ for some R' with $\mathscr{T}_{\rm cf}(R') \equiv U$.

Proof. By induction of the depth of the inference of the transition $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$. Since R is a $\pi^{\rho}_{ES}(\mathcal{N})$ process, using Lemma A.49, ${\rm fn}(\mathscr{T}_{\rm cf}(R)) = {\rm fn}(R) \subseteq \mathcal{Z}$.

- The case that $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **early-input** proceeds as in the proof of Lemma A.51, but using $\iota(\beta) \subseteq \mathcal{N}$.
- Suppose $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **ide**. Then $R = A(\vec{x})$. Let $A(\vec{x}) \stackrel{\rm def}{=} P$, so $A_{\rm cf}(\vec{x}) \stackrel{\rm def}{=} \mathscr{T}_{\rm cf}(P)$. Then $\mathscr{T}_{\rm cf}(P) \xrightarrow{\beta} U$.
  By induction $P \xrightarrow{\beta} R'$ for some R' with $\mathscr{T}_{\rm cf}(R') \equiv U$. Applying rule **ide**, $R \xrightarrow{\beta} R'$.
- The cases that $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **tau**, **output**, **sum** or **symb-match** are (also) trivial.
- Suppose $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by rule **par**. Then $R = P|Q$, $\mathscr{T}_{\rm cf}(P) \xrightarrow{\alpha} V$, $\mathscr{T}_{\rm cf}(R) = \mathscr{T}_{\rm cf}(P)[\ell]|\mathscr{T}_{\rm cf}(Q)[r]$, $\alpha[\ell]=\beta$, ${\rm bn}(\beta)\cap{\rm fn}(\mathscr{T}_{\rm cf}(R))=\emptyset$ and $U = V[\ell]|\mathscr{T}_{\rm cf}(Q)[r]$. Since $\iota(\beta)\cup{\rm bn}(\beta)\subseteq\mathcal{Z}$, also $\iota(\alpha)\cup{\rm bn}(\alpha)\subseteq\mathcal{Z}$. Therefore, by induction, $P \xrightarrow{\alpha} P'$ for some P' with $\mathscr{T}_{\rm cf}(P') \equiv V$. So ${\rm n}(\alpha)\subseteq\mathcal{Z}$ and $\alpha = \alpha[\ell] = \beta$. By rule **par**, using that ${\rm bn}(\beta)\cap{\rm fn}(R)=\emptyset$ via Lemma A.49, $R = P|Q \xrightarrow{\beta} P'|Q$. By Lemma A.38(g,h), $\mathscr{T}_{\rm cf}(P'|Q) = \mathscr{T}_{\rm cf}(P')[\ell]|\mathscr{T}_{\rm cf}(Q)[r] \equiv V[\ell]|\mathscr{T}_{\rm cf}(Q)[r] = U$.
- Suppose $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **e-s-com**. Then $R = P|Q$, $\mathscr{T}_{\rm cf}(R) = \mathscr{T}_{\rm cf}(P)[\ell]|\mathscr{T}_{\rm cf}(Q)[r]$, $\beta = [x{=}v]MN\tau$, $\mathscr{T}_{\rm cf}(P) \xrightarrow{M\bar{x}y} V$, $\mathscr{T}_{\rm cf}(Q) \xrightarrow{Nvz} W$, $y[\ell] = z[r]$ and $U = V[\ell]|W[r]$. Here I use Lemma A.39 to infer that ${\rm n}(M)\cup{\rm n}(N)\cup\{x,y,v\}\subseteq\mathcal{N}$, so that these names are not affected by $[\ell]$ or $[r]$. It follows that $z = y$. By induction $P \xrightarrow{M\bar{x}y} P$ and $Q \xrightarrow{Nvy} Q'$ for processes P', Q' with $\mathscr{T}_{\rm cf}(P') \equiv V$ and $\mathscr{T}_{\rm cf}(Q') \equiv W$. Applying **e-s-com**, $R \xrightarrow{\beta} P'|Q'$. Moreover, employing Lemma A.38(g,h), $\mathscr{T}_{\rm cf}(P'|Q') = \mathscr{T}_{\rm cf}(P')[\ell] \mid \mathscr{T}_{\rm cf}(Q')[r] \equiv V[\ell]|W[r] = U$.
- Suppose $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **e-s-close**. Then $R{=}P|Q$, $\mathscr{T}_{\rm cf}(R) = \mathscr{T}_{\rm cf}(P)[\ell]|\mathscr{T}_{\rm cf}(Q)[r]$, $\mathscr{T}_{\rm cf}(P) \xrightarrow{\alpha} V$, $\mathscr{T}_{\rm cf}(Q) \xrightarrow{\gamma} W$, $\beta = [x{=}v]MN\tau$, $\alpha[\ell] = M\bar{x}(z)$, $\gamma[r] = Nvz$, $z \notin {\rm fn}(\mathscr{T}_{\rm cf}(R))$ and $U = (\nu z)((V[\ell]|W[r])$. By Lemma A.39 $\{x,v\}\cup{\rm n}(M)\cup{\rm n}(N) \subseteq {\rm fn}(\mathscr{T}_{\rm cf}(R)) \subseteq \mathcal{Z}$. So $\alpha = M\bar{x}(w)$ and $\gamma = Nvu$ with $w[\ell] = u[r] = z$. Note that $w, u \notin {\rm fn}(\mathscr{T}_{\rm cf}(R)) \subseteq \mathcal{Z}$. Pick $q \in \mathcal{Z}\backslash{\rm fn}(R)$. By taking $q \neq w, u$ one also has $q \notin {\rm fn}(V|W|V[\ell]|W[r])$.
  By Lemmas A.41 and A.42 $\mathscr{T}_{\rm cf}(P) \xrightarrow{M\bar{x}(q)} V' \equiv V\{q/\!\!/w\}$ and $\mathscr{T}_{\rm cf}(Q) \xrightarrow{Nxq} W' \equiv W\{q/\!\!/w\}$, and the inferences of these transitions have a smaller depth than the one of $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$. Therefore, by induction, $P \xrightarrow{M\bar{x}(q)} P'$ and $Q \xrightarrow{Nxq} Q'$ for some P' and Q' with $\mathscr{T}_{\rm cf}(P') \equiv V'$ and $\mathscr{T}_{\rm cf}(Q') \equiv W'$. By rule **e-s-close** $R \xrightarrow{\beta} (\nu q)(P'|Q')$.
  By Lemma A.50 and Lemma A.38(g,h,b,a,d,f)

$$\begin{aligned}
\mathscr{T}_{\rm cf}((\nu q)(P'|Q')) &\equiv (\nu q)\big((\mathscr{T}_{\rm cf}(P')[\ell] \mid \mathscr{T}_{\rm cf}(Q')[r])\big)\\
&\equiv (\nu q)\big((V'[\ell] \mid W'[r])\big)\\
&\equiv (\nu q)\big((V[\{q/\!\!/w\}][\ell] \mid W[\{q/\!\!/u\}][r])\big)\\
&\equiv (\nu q)\big((V[\ell][\{q/\!\!/z\}] \mid W[r][\{q/\!\!/z\}])\big)\\
&\equiv (\nu q)\big((V[\ell] \mid W[r])[\{q/\!\!/z\}]\big)\\
&\equiv (\nu z)(V[\ell] \mid W[r]) \equiv U.
\end{aligned}$$

- Suppose R has the form $(\nu y)P$, i.e., $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$ is derived by **res-alpha** or **symb-open-alpha**. By Lemma A.50 $\mathscr{T}_{\rm cf}(R) \equiv (\nu y)\mathscr{T}_{\rm cf}(P)$, so by Lemma A.37 $(\nu y)\mathscr{T}_{\rm cf}(P) \xrightarrow{\beta} V \equiv U$, and the inference of this transition has the same depth as the one of $\mathscr{T}_{\rm cf}(R) \xrightarrow{\beta} U$.

  First suppose $(\nu y)\mathscr{T}_{\rm cf}(P) \xrightarrow{\beta} V$ is derived by **res-alpha**. In that case $\mathscr{T}_{\rm cf}(P[\{z/\!\!/y\}]) = \mathscr{T}_{\rm cf}(P)[\{z/\!\!/y\}] \xrightarrow{\beta} W$ and $V = (\nu z)W$ for some W and $z \notin {\rm fn}(R)\cup{\rm n}(\beta)$. So by induction $P[\{z/\!\!/y\}] \xrightarrow{\beta} P'$ for some P' with $\mathscr{T}_{\rm cf}(P') \equiv W$. By **res-alpha** $R = (\nu y)P \xrightarrow{\beta} (\nu z)P'$.

Moreover, by Lemma A.50 and Lemma A.38(h),

$$\mathscr{T}_{\mathrm{cf}}((\nu z)P') \equiv (\nu z)\mathscr{T}_{\mathrm{cf}}(P') \equiv (\nu z)W = V \equiv U.$$

Next suppose $(\nu y)\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\beta} V$ is derived by **SYMB-OPEN-ALPHA**. Then $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{M\bar{x}y} W$, $\beta = M\bar{x}(z)$, $y \neq x$, $z \notin \mathrm{FN}(R)$, $y \notin \mathrm{n}(M)$ and $V = W[\{z/\!\!/y\}]$. By induction $P \xrightarrow{M\bar{x}y} P'$ for some P' with $\mathscr{T}_{\mathrm{cf}}(P') \equiv W$. By **RES-ALPHA** $R \xrightarrow{\beta} P'[\{z/\!\!/y\}]$. By Lemma A.38(h),

$$\mathscr{T}_{\mathrm{cf}}(P'[\{z/\!\!/y\}]) = \mathscr{T}_{\mathrm{cf}}(P')[\{z/\!\!/y\}] \equiv W[\{z/\!\!/y\}] = V \equiv U.$$

- Suppose $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{\beta} U$ is derived by **RELABELLING**. Then $R = P[\sigma]$, $\mathscr{T}_{\mathrm{cf}}(R) = \mathscr{T}_{\mathrm{cf}}(P)[\sigma]$, $\mathscr{T}_{\mathrm{cf}}(P) \xrightarrow{\alpha} V$, $\alpha[\sigma] = \beta$, $\mathrm{bn}(\beta) \cap \mathrm{FN}(R) = \emptyset$ and $U = V[\sigma]$. Since the relabelling operator $[\sigma]$ stems from $\pi^{\rho}_{ES}(\mathcal{N})$, $dom(\sigma) \subseteq \mathcal{Z}$. Hence $\iota(\beta) \cup \mathrm{bn}(\beta) \subseteq \mathcal{Z}$ implies $\iota(\alpha) \cup \mathrm{bn}(\alpha) \subseteq \mathcal{Z}$. By induction $P \xrightarrow{\alpha} P'$ for some P' with $\mathscr{T}_{\mathrm{cf}}(P') \equiv V$. By **RELABELLING** $R = P[\sigma] \xrightarrow{\beta} P'[\sigma]$. Moreover, by Lemma A.38(h), $\mathscr{T}_{\mathrm{cf}}(P'[\sigma]) = \mathscr{T}_{\mathrm{cf}}(P')[\sigma] \equiv V[\sigma] = U$. □

THEOREM A.53. $\mathscr{T}_{\mathrm{cf}}(P) \mathrel{\dot{\sim}} P$ for any $P \in \mathrm{T}_{\pi^{\rho}_{ES}(\mathcal{N})}$.

PROOF. It suffices to show that the symmetric closure of

$$\mathscr{R} := \{(R, U) \mid R \in \mathrm{T}_{\pi^{\rho}_{ES}(\mathcal{N})} \wedge U \equiv \mathscr{T}_{\mathrm{cf}}(R)\}$$

is a strong barbed bisimulation, as it contains $(P, \mathscr{T}_{\mathrm{cf}}(P))$.

Let $(R, U) \in \mathscr{R}$. Let $R \xrightarrow{\tau} R'$. Then $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{\tau} \equiv \mathscr{T}_{\mathrm{cf}}(R')$ by Lemma A.51 and $U \xrightarrow{\tau} \equiv \mathscr{T}_{\mathrm{cf}}(R')$ by Lemma A.37. Thus $U \xrightarrow{\tau} U'$ for some U' with $(R', U') \in \mathscr{R}$.

Let $U \xrightarrow{\tau} U'$. Then $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{\tau} \equiv U'$ by Lemma A.37. By Lemma A.52 $R \xrightarrow{\tau} R'$ for some R' with $\mathscr{T}_{\mathrm{cf}}(R') \equiv U'$.

Let $U{\downarrow_b}$ with $b \in \mathcal{Z} \cup \overline{\mathcal{Z}}$. Then $U \xrightarrow{by} U'$ or $U \xrightarrow{b(y)} U'$ for some y and U', using the definition of O in Section 4. By Lemmas A.43 and A.41 I may assume, without loss of generality, that $y \in \mathcal{Z} \setminus \mathrm{FN}(U)$ in case of input or bound output actions. Hence $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{by}$ or $\mathscr{T}_{\mathrm{cf}}(R) \xrightarrow{b(y)}$ by Lemma A.37. So $R \xrightarrow{by}$ or $R \xrightarrow{b(y)}$ by Lemma A.52. Thus $R{\downarrow_b}$.

The implication $R{\downarrow_b} \Rightarrow U{\downarrow_b}$ proceeds likewise. □

A.8 Eliminating α-conversion for clash-free processes

Let $\xrightarrow{\alpha}_{\bullet}$ be the transition relation on $\mathrm{T}_{\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})}$ obtained by from the transition relation $\xrightarrow{\alpha}$ by replacing the rules **RES-ALPHA** and **SYMB-OPEN-ALPHA** from Table 10 by the rules rules **RES** and **SYMB-OPEN** from Table 6. Thus, the α-conversion implicit in these rules has been removed.

As witnessed by Example 6.3, on $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$ the transition relation $\xrightarrow{\alpha}_{\bullet}$ differs essentially from $\xrightarrow{\alpha}$, and fails to derive some crucial transitions. However, I will show that restricted to the clash-free processes within $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$ this transition relation is just as suitable as $\xrightarrow{\alpha}$; up to strong barbed bisimilarity there is no difference. Since the only the clash-free processes in $\pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R})$ occur as the target of the previous translation step, the relation $\xrightarrow{\alpha}_{\bullet}$ may be used in the source of the next.

LEMMA A.54. If $P \xrightarrow{M\bar{x}(y)}_{\bullet} Q$ then $y \in \mathrm{RN}(P)$.

PROOF. A trivial induction on the inference of $P \xrightarrow{M\bar{x}(y)}_{\bullet} Q$. □

LEMMA A.55. If R is clash-free and $R \xrightarrow{\alpha}_{\bullet} R'$ then $R \xrightarrow{\alpha} \equiv R'$.

PROOF. By induction on the derivation of $R \xrightarrow{\alpha}_{\bullet} R'$.

- Suppose $R \xrightarrow{M\bar{x}(y)}_\bullet R'$ is derived by **SYMB-OPEN**. Then $R = (\nu y)P$, $P \xrightarrow{M\bar{x}y}_\bullet R'$, $y \neq x$ and $y \notin \mathrm{n}(M)$. By induction $P \xrightarrow{M\bar{x}y} U$ for a process $U \equiv R'$. So by **SYMB-OPEN-ALPHA** $R \xrightarrow{M\bar{x}(y)} U[\{y/\!\!/y\}]$. Moreover, $U[\{y/\!\!/y\}] \equiv U \equiv R'$ by Lemma A.38(c,a).
- Suppose $R \xrightarrow{\alpha}_\bullet R'$ is derived by **RES**. Then $R = (\nu y)P$, $P \xrightarrow{\alpha}_\bullet P'$, $R' = (\nu y)P'$ and $y \notin \mathrm{n}(\alpha)$. For $u \in \mathrm{bn}(\alpha)$, by Lemma A.54 $u \in \mathrm{Rn}(P)$, so by the clash-freedom of P, $u \notin \mathrm{fn}(P)$. By induction $P \xrightarrow{\alpha} V$ for some $V \equiv P'$. So by **RELABELLING** $P[\{y/\!\!/y\}] \xrightarrow{\alpha} V[\{y/\!\!/y\}]$ and by rule **RES-ALPHA** $R \xrightarrow{\alpha} (\nu y)(V[\{y/\!\!/y\}])$. By Lemma A.38(c,a,h) $(\nu y)(V[\{y/\!\!/y\}]) \equiv (\nu y)V \equiv (\nu y)P' = R'$
- All other cases are trivial with Lemma A.38(g,h). □

LEMMA A.56. Let R be clash-free. If $R \xrightarrow{\alpha} R'$, $\mathrm{bn}(\alpha) = \emptyset$ and $\iota(\alpha) \cap \mathrm{Rn}(R) = \emptyset$, then $R \xrightarrow{\alpha}_\bullet U$ for some U with $R' \equiv U$.
If $R \xrightarrow{M\bar{x}(z)} R'$ then $R \xrightarrow{M\bar{x}(y)}_\bullet U$ for some y and U with $R' \equiv U[\{z/\!\!/y\}]$ and $z \notin \mathrm{fn}(U)\backslash\{y\}$.

PROOF. By induction on the depth of the derivation of $R \xrightarrow{\alpha} R'$.

- The cases that $R \xrightarrow{\alpha}_\bullet R'$ is derived by **TAU**, **OUTPUT**, **EARLY-INPUT**, **SUM**, **SYMB-MATCH** or **IDE** are trivial.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by **PAR**. Then $R = P|Q$, $P \xrightarrow{\alpha} P'$, $\mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$ and $R' = P'|Q$.
  First assume $\mathrm{bn}(\alpha) = \emptyset$. Since $\iota(\alpha) \cap \mathrm{Rn}(R) = \emptyset$, also $\iota(\alpha) \cap \mathrm{Rn}(P) = \emptyset$. By induction $P \xrightarrow{\alpha}_\bullet V$ for some V with $P' \equiv V$. By rule **PAR** $R \xrightarrow{\alpha}_\bullet V|Q$. Moreover, $R' = P'|Q \equiv V|Q$ by Lemma A.38(g).
  Next assume $\alpha = M\bar{x}(z)$. So $z \notin \mathrm{fn}(R)$. By induction $P \xrightarrow{M\bar{x}(y)}_\bullet V$ for some y, V with $P' \equiv V[\{z/\!\!/y\}]$. By Lemma A.54, $y \in \mathrm{Rn}(P)$, so $y \notin \mathrm{fn}(R)$ by the clash-freedom of R. Hence $R \xrightarrow{M\bar{x}(y)}_\bullet V|Q$ by rule **PAR**. Moreover, $R' = P'|Q \equiv V[\{z/\!\!/y\}]|Q[\{z/\!\!/y\}] \equiv (V|Q)[\{z/\!\!/y\}]$ by Lemma A.38(c,a,g,d), using that $y, z \notin \mathrm{fn}(Q)$. Finally, by Lemma A.40, $\mathrm{fn}(V|Q)\backslash\{y\} \subseteq \mathrm{fn}(R) \not\ni z$.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by rule **E-S-COM**. Then $R = P|Q$, $\alpha = [x{=}v]MN\tau$, $P \xrightarrow{M\bar{x}y} P'$, $Q \xrightarrow{Mvy} Q'$ and $R' = P'|Q'$. By Lemma A.39 $y \in \mathrm{fn}(P)$. So by the clash-freedom of R, $y \notin \mathrm{Rn}(Q)$. By induction $P \xrightarrow{M\bar{x}y}_\bullet V$ and $Q \xrightarrow{Mvy}_\bullet W$ with $P \equiv V$ and $Q' \equiv W$. Hence by **E-S-COM** $R \xrightarrow{\alpha}_\bullet V|W$, and $R' = P'|Q' \equiv V|W$.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by rule **E-S-CLOSE**. Then $R = P|Q$, $\alpha = [x{=}v]MN\tau$, $P \xrightarrow{M\bar{x}(z)} P'$, $Q \xrightarrow{Mvz} Q'$, $z \notin \mathrm{fn}(R)$ and $R' = (\nu z)(P'|Q')$. By induction, one has $P \xrightarrow{M\bar{x}(y)}_\bullet V$ for some y and V with $P' \equiv V[\{z/\!\!/y\}]$ and $z \notin \mathrm{fn}(V)\backslash\{y\}$. By Lemma A.54 $y \in \mathrm{Rn}(P)$, so by the clash-freedom of R, $y \notin \mathrm{Rn}(Q)$ and $y \notin \mathrm{fn}(R)$. By Lemma A.42 $Q \xrightarrow{Mvy} W_y$ for some $W_y \equiv Q'[\{y/\!\!/z\}]$. So by induction $Q \xrightarrow{Mvy}_\bullet W$ for some $W \equiv W_y$. By **E-S-CLOSE** $R \xrightarrow{\alpha}_\bullet (\nu y)(V|W)$. Moreover, by Lemma A.38(h,a,b,c,h,d,f),

$$\begin{aligned} R' = (\nu z)(P'|Q') &\equiv (\nu z)(V[\{z/\!\!/y\}] \mid Q'[\{z/\!\!/y\}][\{z/\!\!/y\}]) \\ &\equiv (\nu z)(V[\{z/\!\!/y\}] \mid W_y[\{z/\!\!/y\}]) \\ &\equiv (\nu z)((V|W)[\{z/\!\!/y\}]) \\ &\equiv (\nu y)(V|W) \end{aligned}$$

  in the last step using that $z \notin \mathrm{fn}(R) \supseteq \mathrm{fn}((\nu y)(V|W))$ by Lemma A.40.
- Suppose $R \xrightarrow{\alpha} R'$ is derived by **RELABELLING**. Then $R = P[\sigma]$, $P \xrightarrow{\beta} P'$, $\beta[\sigma] = \alpha$ and $R' = P'[\sigma]$.

First assume $\mathrm{bn}(\alpha)=\emptyset$. Since $\iota(\alpha)\cap \mathrm{Rn}(R)=\emptyset$, also $\iota(\beta)\cap\mathrm{Rn}(P)=\emptyset$. By induction $P \xrightarrow{\beta}_\bullet V$ for some V with $P' \equiv V$. Now $R \xrightarrow{\alpha}_\bullet V[\sigma]$ by **relabelling**. Moreover, $R'=P'[\sigma]\equiv V[\sigma]$ by Lemma A.38(h).
Next assume $\alpha = M\bar{x}(z)$ and $\beta = K\bar{q}(w)$. So $w[\sigma]=z$. Then $z\notin \mathrm{fn}(R)$ by the side-condition of **relabelling**. By induction $P \xrightarrow{K\bar{q}(v)}_\bullet V$ for some v and V with $P'\equiv V[\{w/\!/v\}]$. Let $y := v[\sigma]$. By Lemma A.54, $v\in\mathrm{Rn}(P)$, so $y\in\mathrm{fn}(R)$. Hence $y\notin\mathrm{fn}(R)$ by the clash-freedom of R. So there is no $n\in\mathrm{fn}(P)$ with $n[\sigma]=y$ or $n[\sigma]=z$. Since $\mathrm{fn}(V)\subseteq\mathrm{fn}(P)\cup\{v\}$ by Lemma A.40, the name v is the only possible $n\in\mathrm{fn}(V)$ with $n[\sigma]=y$ or $n[\sigma]=z$. By rule **relabelling** $R\xrightarrow{M\bar{x}(y)}_\bullet V[\sigma]$. Lemma A.38(h,b,a) yields $R'=P'[\sigma]\equiv V[\{w/\!/v\}][\sigma]\equiv V[\sigma][\{z/\!/y\}]$. Finally, by Lemma A.40, $\mathrm{fn}(V[\sigma])\backslash\{y\}\subseteq\mathrm{fn}(R)\not\ni z$.

- Suppose $R\xrightarrow{M\bar{x}(z)}R'$ is derived by **symb-open-alpha**. Then $R=(\nu y)P$, $P\xrightarrow{M\bar{x}y}P'$, $R'=P'[\{z/\!/y\}]$, $y\neq x$, $z\notin\mathrm{fn}(R)$ and $y\notin\mathrm{n}(M)$. By induction $P\xrightarrow{M\bar{x}y}_\bullet U$ for a process U with $P'\equiv U$. So by **symb-open** $R\xrightarrow{M\bar{x}(y)}_\bullet U$. Moreover, $R'=P'[\{z/\!/y\}]\equiv U[\{z/\!/y\}]$ by Lemma A.38(h). Finally, by Lemma A.40, $\mathrm{fn}(U)\backslash\{y\}\subseteq\mathrm{fn}(R)\not\ni z$.
- Suppose $R\xrightarrow{\alpha}R'$ is derived by **res-alpha**. Then $R=(\nu y)P$, $P[\{z/\!/y\}]\xrightarrow{\alpha}P'$, $R'=(\nu z)P'$ and $z\notin\mathrm{fn}(R)\cup\mathrm{n}(\alpha)$.
  First assume $\mathrm{bn}(\alpha)=\emptyset$. As $\iota(\alpha)\cap\mathrm{Rn}(R)=\emptyset$, $y\notin\iota(\alpha)$ and $\iota(\alpha)\cap\mathrm{Rn}(P)=\emptyset$. Hence $y\notin\mathrm{n}(\alpha)$ by Lemma A.39. By **relabelling** $P\xrightarrow{\alpha}V$ for a V with $V[\{z/\!/y\}]=P'$. So by induction $P\xrightarrow{\alpha}_\bullet W$ for some $W\equiv V$. So by **res** $R\xrightarrow{\alpha}_\bullet(\nu y)W$. By Lemma A.40 $\mathrm{fn}(V)\backslash\{y\}\subseteq\mathrm{n}(\alpha)\cup\mathrm{fn}(P)\backslash\{y\}=\mathrm{n}(\alpha)\cup\mathrm{fn}(R)\not\ni z$. Hence $R'=(\nu z)P'=(\nu z)(V[\{z/\!/y\}])\equiv(\nu y)V$ by Lemma A.38(f).
  Next assume $\alpha=M\bar{x}(w)$. By Lemma A.39 $\mathrm{n}(M)\cup\{x\}\subseteq\mathrm{fn}(R)\not\ni y$. So by **relabelling** $P\xrightarrow{M\bar{x}(u)}W$ for a W with $W[\{z/\!/y\}]=P'$. Here $u:=w[\{z/\!/y\}]$. By induction $P\xrightarrow{M\bar{x}(v)}_\bullet V$ for some v and V with $W\equiv V[\{u/\!/v\}]$ and $w\notin\mathrm{fn}(V)\backslash\{v\}$. By Lemma A.54 $v\in\mathrm{Rn}(P)$. So $v\neq y$ by the clash-freedom of R. Hence $R\xrightarrow{M\bar{x}(v)}_\bullet(\nu y)V$ by **res**. Pick a name $q\notin\mathrm{fn}(R')\cup\mathrm{fn}((\nu y)V)\cup\{w,v\}$. Now

$$\begin{aligned}R'=(\nu z)P' &= (\nu z)(W[\{z/\!/y\}])\\ &\equiv(\nu q)(W[\{z/\!/y\}][\{q/\!/z\}])\\ &\equiv(\nu q)(V[\{u/\!/v\}][\{z/\!/y\}][\{q/\!/z\}])\\ &\equiv(\nu q)(V[\{q/\!/y\}][\{w/\!/v\}])\\ &\equiv((\nu q)(V[\{q/\!/y\}]))[\{w/\!/v\}]\\ &\equiv((\nu y)V)[\{w/\!/v\}]\end{aligned}$$

by Lemma A.38(f,h,b,a,e,f,h), provided that

$$n[\{u/\!/v\}][\{z/\!/y\}][\{q/\!/z\}]=n[\{q/\!/y\}][\{w/\!/v\}] \tag{21}$$

for all $n\in\mathrm{fn}(V)$. This has to be checked only for $n\in\{y,v\}$, because $q\notin\mathrm{fn}(V)\backslash\{y\}$, $w\notin\mathrm{fn}(V)\backslash\{v\}$, u is either w or z, and $z\notin\mathrm{fn}(R)\supseteq\mathrm{fn}(V)\backslash\{y,v\}$ by Lemma A.40. To check (21) I recall that $y\neq v\neq q\neq w\neq z$ and consider two cases.
– Let $w\neq y$. Then $u=w$, and (21) holds for $n=y,v$.
– Let $w=y$. Then $u=z\neq y\neq q$ and again (21) holds for $n=y,v$.
Finally, $w\notin\mathrm{fn}((\nu y)V)\backslash\{v\}$. □

Below a substitution σ is called *clash-free* on a $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ process P iff $\mathrm{Rn}(P)\cap(dom(\sigma)\cup range(\sigma))=\emptyset$.

Observation A.57. If P is clash-free and σ is clash-free on P, then $P[\sigma]$ is clash-free and $\mathrm{Rn}(P[\sigma])=\mathrm{Rn}(P)$.

Lemma A.58. If $x(y).P$ is clash-free and $z \notin \text{Rn}(x(y).P)$ then substitution $\{z/y\}^{\mathcal{S}}$ from **early-input** is clash-free on P.

Proof. By definition $dom(\{z/y\}^{\mathcal{S}}) \subseteq \mathcal{Z}$ and $range(\{z/y\}^{\mathcal{S}}) \subseteq \mathcal{Z} \cup \{z\}$. Since $x(y).P$ is clash-free, so is process P by Definition A.48(v), and $\text{Rn}(P) = \text{Rn}(x(y).P)$ by Definition A.47. Hence $\text{Rn}(P) \subseteq \mathcal{R}$ and $z \notin \text{Rn}(x(y).P) = \text{Rn}(P)$. □

Lemma A.59. If $R \xrightarrow{\alpha}_{\bullet} R'$ with $\iota(\alpha) \cap \text{Rn}(R) = \emptyset$ and R is clash-free, then so is R'. Moreover, $\text{Rn}(R') \subseteq \text{Rn}(R)$ and $\text{bn}(\alpha) \cap \text{Rn}(R') = \emptyset$.

Proof. By induction on the derivation of $R \xrightarrow{\alpha}_{\bullet} R'$. As R is clash-free, $\text{Rn}(R) \subseteq \mathcal{R}$.

- By Definition A.48(iv-v) a subexpression P of a clash-free process R is also clash-free, and by Definition A.47 $\text{Rn}(P) \subseteq \text{Rn}(R)$. Using this, the cases that $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by **tau**, **output**, **sum** or **symb-match** are trivial.
- Suppose $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by **early-input**. Then $R = x(y).P$, $\alpha = xz$ with $z \notin \text{Rn}(R)$, and $R' = P[\{z/y\}^{\mathcal{S}}]$. By Definition A.48(v) P is clash-free and by Definition A.47 $\text{Rn}(P) = \text{Rn}(R)$. By Lemma A.58 and Observation A.57 R' is clash-free and $\text{Rn}(R') = \text{Rn}(R)$. Moreover, $\text{bn}(\alpha) = \emptyset$.
- Suppose $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by **ide**. Then $R = A(\vec{x})$. Let $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $P \xrightarrow{\alpha}_{\bullet} R'$. Process P is clash-free by Definition A.48(vii), and $\text{Rn}(P) = \text{Rn}(R)$ by Definition A.47. By induction $\text{Rn}(R') \subseteq \text{Rn}(P) = \text{Rn}(R)$, process R' is clash-free, and $\text{bn}(\alpha) \cap \text{Rn}(R') = \emptyset$.
- Suppose $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by **par**. Then $R = P|Q$, $P \xrightarrow{\alpha}_{\bullet} P'$ and $R' = P'|Q$. Since $\iota(\alpha) \cap \text{Rn}(R) = \emptyset$ and $\text{Rn}(P) \subseteq \text{Rn}(R)$, also $\iota(\alpha) \cap \text{Rn}(P) = \emptyset$. Using that P is clash-free, by induction P' is clash-free and $\text{Rn}(P') \subseteq \text{Rn}(P)$. Moreover, $\text{bn}(\alpha) \cap \text{Rn}(P') = \emptyset$. Now $\text{Rn}(R') = \text{Rn}(P') \cup \text{Rn}(Q) \subseteq \text{Rn}(P) \cup \text{Rn}(Q) = \text{Rn}(R)$.

  Let $z \in \text{bn}(\alpha)$. Then $z \in \text{Rn}(P)$ by Lemma A.54, so by the clash-freedom of R one has $z \notin \text{Rn}(Q)$. Moreover, $z \notin \text{Rn}(P')$ by the above, so $z \notin \text{Rn}(R')$.

  To show that R' is clash-free, using that P' and Q are clash-free, it suffices to show (i) $\text{fn}(R') \cap \text{Rn}(R') = \emptyset$ and (ii) $\text{Rn}(P') \cap \text{Rn}(Q) = \emptyset$. The latter follows since $\text{Rn}(P) \cap \text{Rn}(Q) = \emptyset$ by the clash-freedom of R. For the former, Lemma A.40 yields $\text{fn}(R') \subseteq \text{fn}(R) \cup \iota(\alpha) \cup \text{bn}(\alpha)$. Moreover $\text{fn}(R) \cap \text{Rn}(R) = \emptyset$ by the clash-freedom of R, $\iota(\alpha) \cap \text{Rn}(R) = \emptyset$ by assumption, and $\text{bn}(\alpha) \cap \text{Rn}(R') = \emptyset$ as derived above. This entails (i).
- Suppose $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by rule **e-s-com**. Then $R = P|Q$, $\alpha = [x{=}v]MN\tau$, $P \xrightarrow{M\bar{x}y}_{\bullet} P'$, $Q \xrightarrow{Mvy}_{\bullet} Q'$ and $R' = P'|Q'$. By Lemma A.39 $y \in \text{fn}(P)$, so $y \notin \text{Rn}(Q)$ by the clash-freedom of R. Using that P and Q are clash-free, by induction P' and Q' are clash-free, $\text{Rn}(P') \subseteq \text{Rn}(P)$ and $\text{Rn}(Q') \subseteq \text{Rn}(Q)$. It follows that $\text{Rn}(R') = \text{Rn}(P') \cup \text{Rn}(Q') \subseteq \text{Rn}(P) \cup \text{Rn}(Q) = \text{Rn}(R)$.

  To show that R' is clash-free, using that P' and Q' are clash-free, it suffices to show (i) $\text{fn}(R') \cap \text{Rn}(R') = \emptyset$ and (ii) $\text{Rn}(P') \cap \text{Rn}(Q') = \emptyset$. The latter follows since $\text{Rn}(P) \cap \text{Rn}(Q) = \emptyset$ by the clash-freedom of R. For the former, Lemma A.40 yields $\text{fn}(R') \subseteq \text{fn}(R)$. As $\text{fn}(R) \cap \text{Rn}(R) = \emptyset$ by the clash-freedom of R, this entails (i).
- Suppose $R \xrightarrow{\alpha}_{\bullet} R'$ is derived by rule **e-s-close**. Then $R = P|Q$, $\alpha = [x{=}v]MN\tau$, $P \xrightarrow{M\bar{x}(z)}_{\bullet} P'$, $Q \xrightarrow{Mvz}_{\bullet} Q'$ and $R' = (\nu z)(P'|Q')$. By Lemma A.54 $z \in \text{Rn}(P)$, so $y \notin \text{Rn}(Q)$ by the clash-freedom of R. Using that P and Q are clash-free, by induction P' and Q' are clash-free, $\text{Rn}(P') \subseteq \text{Rn}(P)$, $\text{Rn}(Q') \subseteq \text{Rn}(Q)$ and $z \notin \text{Rn}(P')$. It follows that $\text{Rn}(R') = \text{Rn}(P') \cup \text{Rn}(Q') \cup \{z\} \subseteq \text{Rn}(P) \cup \text{Rn}(Q) = \text{Rn}(R)$.

  To show that R' is clash-free, using that P' and Q' are clash-free, it suffices to show (i) $\text{fn}(R') \cap \text{Rn}(R') = \emptyset$, (ii) $\text{Rn}(P') \cap \text{Rn}(Q') = \emptyset$ and (iii) $z \notin \text{Rn}(P'|Q')$. That (i) and (ii) hold

follows exactly as in the case of **E-S-COM** above. For (iii), using that $z \in \text{Rn}(P)$ and $z \notin \text{Rn}(P')$, in case $z \in \text{Rn}(P'|Q')$ then $z \in \text{Rn}(Q') \subseteq \text{Rn}(Q)$, contradicting the clash-freedom of R.

- Suppose $R \xrightarrow{\alpha}_\bullet R'$ is derived by **RES**. Then $R = (\nu y)P$, $P \xrightarrow{\alpha}_\bullet P'$, $y \notin \text{n}(\alpha)$ and $R' = (\nu y)P'$. Since $\iota(\alpha) \cap \text{Rn}(R) = \emptyset$ and $\text{Rn}(P) \subseteq \text{Rn}(R)$, also $\iota(\alpha) \cap \text{Rn}(P) = \emptyset$. Using that P is clash-free, by induction P' is clash-free and $\text{Rn}(P') \subseteq \text{Rn}(P)$. Moreover, $\text{bn}(\alpha) \cap \text{Rn}(P') = \emptyset$. Now $\text{Rn}(R') = \text{Rn}(P') \cup \{y\} \subseteq \text{Rn}(P) \cup \{y\} = \text{Rn}(R)$.

  As $\text{bn}(\alpha) \cap \text{Rn}(P') = \emptyset$ and $y \notin \text{n}(\alpha)$, $\text{bn}(\alpha) \cap \text{Rn}(R') = \emptyset$.

  To show that R' is clash-free, using that P' is clash-free, it suffices to show (i) $\text{fn}(R') \cap \text{Rn}(R') = \emptyset$ and (iii) $y \notin \text{Rn}(P')$. The latter follows since $y \notin \text{Rn}(P)$ by the clash-freedom of R. The former follows exactly as in the case of **PAR**.
- Suppose $R \xrightarrow{\alpha}_\bullet R'$ is derived by rule **SYMB-OPEN**. Then $R = (\nu y)P$, $\alpha = M\bar{x}(y)$, $P \xrightarrow{M\bar{x}y}_\bullet R'$, $y \neq x$ and $y \notin \text{n}(M)$. Using that P is clash-free, by induction R' is clash-free and $\text{Rn}(R') \subseteq \text{Rn}(P) \subseteq \text{Rn}(R)$. Since R is clash-free, $y \notin \text{Rn}(P) \supseteq \text{Rn}(R')$, so $\text{bn}(\alpha) \cap \text{Rn}(R') = \emptyset$.
- Suppose $R \xrightarrow{\alpha}_\bullet R'$ is derived by rule **RELABELLING**. Then $R = P[\sigma]$, $P \xrightarrow{\beta}_\bullet P'$, $\beta[\sigma] = \alpha$, $\text{bn}(\alpha) \cap \text{fn}(R) = \emptyset$ and $R' = P'[\sigma]$. Since $\iota(\alpha) \cap \text{Rn}(R) = \emptyset$, also $\iota(\alpha) \cap \text{Rn}(P) = \emptyset$. Using that P is clash-free, by induction P' is clash-free, $\text{Rn}(P') \subseteq \text{Rn}(P)$ and $\text{bn}(\alpha) \cap \text{Rn}(P') = \emptyset$. It follows that $\text{Rn}(R') \subseteq \text{Rn}(R)$.

  Suppose $z \in \text{bn}(\alpha) \cap \text{Rn}(R')$. Then there are $v \in \text{bn}(\beta)$ and $w \in \text{Rn}(P')$ with $v[\sigma] = w[\sigma]$. As $\text{bn}(\alpha) \cap \text{Rn}(P') = \emptyset$ one has $v \neq w$,. By Lemma A.54 $v \in \text{Rn}(P)$. Moreover, $w \in \text{Rn}(P)$. This contradicts Condition (iv) of the clash-freedom of R.

  To show that R' is clash-free, using that P' is clash-free, it suffices to show (i) $\text{fn}(R') \cap \text{Rn}(R') = \emptyset$ and (iv) $y[\sigma] \neq z[\sigma]$ for different $y, z \in \text{Rn}(P')$. The latter follows from the same condition for $\text{Rn}(P)$. The former follows exactly as in the case of **PAR**. □

Let $\mathscr{T}_\bullet$ be the identity translation from the clash-free processes in $\pi^\rho_{ES}(\mathcal{N}, \mathcal{R})$ equipped with the standard transition relation $\xrightarrow{\alpha}$ to clash-free processes in $\pi^\rho_{ES}(\mathcal{N}, \mathcal{R})$ equipped with the alternative transition relation $\xrightarrow{\alpha}_\bullet$. The results above imply that $\mathscr{T}_\bullet$ is valid up to strong barbed bisimilarity.

THEOREM A.60. $\mathscr{T}_\bullet(P) \mathrel{\dot{\sim}} P$ for any clash-free $P \in \text{T}_{\pi^\rho_{ES}(\mathcal{N},\mathcal{R})}$.

PROOF. It suffices to show that the symmetric closure of

$$\mathscr{R} := \{(U, R) \mid U, R \in \text{T}_{\pi^\rho_{ES}(\mathcal{N},\mathcal{R})} \wedge U \equiv R \wedge R \text{ clash-free}\}$$

is a strong barbed bisimulation. Take $(R, U) \in \mathscr{R}$.

Let $U \xrightarrow{\tau} U'$. By Lemma A.37 $R \xrightarrow{\tau} R^\dagger$ for some $R^\dagger \equiv U'$. So $R \xrightarrow{\tau}_\bullet R'$ for some $R' \equiv R^\dagger$ by Lemma A.56. By Lemma A.59 R' is clash-free. Hence $(U', R') \in \mathscr{R}$.

Let $R \xrightarrow{\tau}_\bullet R'$. Then $R \xrightarrow{\tau} R^\dagger$ for some $R^\dagger \equiv R'$ by Lemma A.55. By Lemma A.37 $U \xrightarrow{\tau} U'$ for some $U' \equiv R^\dagger$. By Lemma A.59 R' is clash-free. Hence $(U', R') \in \mathscr{R}$.

Let $U{\downarrow_b}$ with $b \in \mathcal{Z} \cup \overline{\mathcal{Z}}$. Then $U \xrightarrow{by} U'$ or $U \xrightarrow{b(y)} U'$ for some y and U', using the definition of O in Section 4. By Lemmas A.43 and A.41 I may assume, without loss of generality, that $y \notin \text{Rn}(R)$ in case of an input action by, and $y \notin \text{fn}(R)$ in case of a bound output action $b(y)$. Hence $R \xrightarrow{by}$ or $R \xrightarrow{b(y)}$ by Lemma A.37 and $R \xrightarrow{by}_\bullet$ or $R \xrightarrow{b(y)}_\bullet$ by Lemma A.56. Thus $R{\downarrow_b}$.

The implication $R{\downarrow_b} \Rightarrow U{\downarrow_b}$ proceeds likewise. □

A.9 Eliminating restriction

Let $\pi^\dagger_{ES}(\mathcal{N}, \mathcal{R})$ be the variant of $\pi^\rho_{ES}(\mathcal{N}, \mathcal{R})$ without restriction operators. Hence there is no need for rules **RES** and **SYMB-OPEN**. Since the resulting semantics cannot generate transitions labelled

$M\bar{x}(z)$, rule **e-s-close** can be dropped as well. Moreover, $\mathrm{bn}(\alpha)=\emptyset$ for all transition labels α. So the side condition of rules **par** and **relabelling** can be dropped too. Hence one is left with Table 10 without the orange part.

In $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$ I also drop the restriction that $\mathrm{fn}(P)\subseteq\{x_1,\ldots,x_n\}$ in defining equations $A(\vec{x})\stackrel{\text{def}}{=}P$. Thus $A(\vec{x})$ can just as well be denoted A.[14] On $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$ I don't use **fn**.

The ninth step $\mathscr{T}_\nu$ of my translation goes from $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ to $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$. It simply drops all restriction operators. It is defined compositionally by $\mathscr{T}_\nu((\nu y)P)=\mathscr{T}_\nu(P)$ and $\mathscr{T}_\nu(A(\vec{x}))=A_\nu(\vec{x})$, where A_ν is a fresh agent identifier with defining equation $A_\nu(\vec{x})\stackrel{\text{def}}{=}\mathscr{T}_\nu(P)$ when $A(\vec{x})\stackrel{\text{def}}{=}P$ was the defining equation of A; the translation $\mathscr{T}_\nu$ acts homomorphically on all other constructs.

Although this translation in not valid in general, I proceed to prove its validity for clash-free processes. By Theorem A.60 I may use the transition relation $\xrightarrow{\alpha}_\bullet$ on $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$.

For α an action in $\pi^{\rho}_{ES}(\mathcal{N},\mathcal{R})$ one defines the *debinding* of α by $\langle\alpha\rangle:=\alpha$ if α has the form $M\tau$, Mxz or $M\bar{x}y$, and $\langle M\bar{x}(y)\rangle:=M\bar{x}y$.

Lemma A.61. If $R\xrightarrow{\alpha}_\bullet R'$, then $\mathscr{T}_\nu(R)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(R')$.

Proof. By induction on the derivation of $R\xrightarrow{\alpha}_\bullet R'$.

- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by rule **early-input**. Then $R=x(y).P$, $\alpha=xz$, $R'=P[\{z/y\}^{\mathcal{S}}]$ and $\mathscr{T}_\nu(R)=x(y).\mathscr{T}_\nu(P)$. Moreover, $\mathscr{T}_\nu(R)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(P)[\{z/y\}^{\mathcal{S}}]=\mathscr{T}_\nu(P[\{z/y\}^{\mathcal{S}}])$.
- The cases that $R\xrightarrow{\alpha}_\bullet R'$ is derived by **tau** or **output** are even more trivial.
- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by **ide**. Then $R=A(\vec{x})$, say with $A(\vec{x})\stackrel{\text{def}}{=}P$, and therefore $P\xrightarrow{\alpha}_\bullet R'$. Moreover $\mathscr{T}_\nu(R)=A_\nu(\vec{x})$, with A_ν defined by $A_\nu(\vec{x})\stackrel{\text{def}}{=}\mathscr{T}_\nu(P)$. Now $\mathscr{T}_\nu(P)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(R')$ by induction. With rule **ide** one infers $\mathscr{T}_\nu(R)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(R')$.
- The cases that $R\xrightarrow{\alpha}_\bullet R'$ is derived by **sum**, **symb-match**, **par** or **e-s-com** are trivial.
- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by **e-s-close**. Then $R=P|Q$, $\alpha=[x{=}v]MN\tau$, $P\xrightarrow{M\bar{x}(z)}_\bullet P'$, $Q\xrightarrow{N\nu z}_\bullet Q'$, $R'=(\nu z)(P'|Q')$ and $\mathscr{T}_\nu(R)=\mathscr{T}_\nu(P)|\mathscr{T}_\nu(Q)$. Now $\mathscr{T}_\nu(P)\xrightarrow{M\bar{x}z}\mathscr{T}_\nu(P')$ and $\mathscr{T}_\nu(Q)\xrightarrow{N\nu z}\mathscr{T}_\nu(Q')$ by induction. Thus $\mathscr{T}_\nu(R)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(P')|\mathscr{T}_\nu(Q')=\mathscr{T}_\nu(R')$ by application of rule **e-s-com**.
- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by **res**. Then $R=(\nu y)P$, $R'=(\nu y)P'$, $P\xrightarrow{\alpha}_\bullet P'$. Now $\mathscr{T}_\nu(P)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(P')$ by induction. So $\mathscr{T}_\nu(R)=\mathscr{T}_\nu(P)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(P')=\mathscr{T}_\nu(R')$.
- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by **symb-open**. Then $R=(\nu y)P$, $\alpha=M\bar{x}(y)$ and $P\xrightarrow{M\bar{x}y}_\bullet R'$. By induction $\mathscr{T}_\nu(P)\xrightarrow{M\bar{x}y}\mathscr{T}_\nu(R')$. So $\mathscr{T}_\nu(R)=\mathscr{T}_\nu(P)\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(R')$.
- Suppose $R\xrightarrow{\alpha}_\bullet R'$ is derived by **relabelling**. Then $R=P[\sigma]$, $P\xrightarrow{\beta}_\bullet P'$, $\beta[\sigma]=\alpha$ and $R'=P'[\sigma]$. By induction $\mathscr{T}_\nu(P)\xrightarrow{\langle\beta\rangle}\mathscr{T}_\nu(P')$. Hence $\mathscr{T}_\nu(R)=\mathscr{T}_\nu(P)[\sigma]\xrightarrow{\langle\alpha\rangle}\mathscr{T}_\nu(P')[\sigma]=\mathscr{T}_\nu(R')$. □

The next lemma makes use of the set $\mathrm{no}(\beta)$ of *non-output* names of an action β in $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$. Here $\mathrm{no}(\beta):=\mathrm{n}(\beta)$ if β has the form $M\tau$ or Mxz, whereas $\mathrm{no}(M\bar{x}y):=\mathrm{n}(M)\cup\{x\}$.

Lemma A.62. If R is clash-free and $\mathscr{T}_\nu(R)\xrightarrow{\beta}U$, where $\mathrm{no}(\beta)\cap\mathrm{Rn}(R)=\emptyset$, then $R\xrightarrow{\alpha}_\bullet R'$ for some α and R' with $\langle\alpha\rangle=\beta$ and $\mathscr{T}_\nu(R')=U$.

Proof. By induction on the derivation of $\mathscr{T}_\nu(R)\xrightarrow{\beta}U$, and a nested structural induction on R.

- The cases $R=\tau.P$ and $R=\bar{x}y.P$ are trivial.

[14] Here I assume that all sets $\mathcal{K}_n$ are disjoint, i.e., the same π-calculus agent identifier A does occur with multiple arities. When this assumption is not met, an arity-index at the $\pi^{\dagger}_{ES}(\mathcal{N},\mathcal{R})$ identifier A is needed.

- Let $R = x(y).P$. Then $\mathscr{T}_\nu(R) = x(y).\mathscr{T}_\nu(P)$. So $\beta = xz$ and $U = \mathscr{T}_\nu(P)[\{z/y\}^{\mathcal{S}}]$. Furthermore, $R \xrightarrow{xz}_\bullet P[\{z/y\}^{\mathcal{S}}]$ and $\mathscr{T}_\nu(P[\{z/y\}]^{\mathcal{S}}) = \mathscr{T}_\nu(P)[\{z/y\}^{\mathcal{S}}] = U$.
- Let $R = A(\vec{x})$ with $A(\vec{x}) \stackrel{\text{def}}{=} P$. Then $\mathscr{T}_\nu(R) = A_\nu(\vec{x})$ with $A_\nu(\vec{x}) \stackrel{\text{def}}{=} \mathscr{T}_\nu(P)$. So $\mathscr{T}_\nu(P) \xrightarrow{\beta} U$. Process P is clash-free by Definition A.48(vii), and $\text{Rn}(P) = \text{Rn}(R)$ by Definition A.47. So $\text{no}(\beta) \cap \text{Rn}(P[\{\vec{y}/\vec{z}\}^{\mathcal{S}}]) = \emptyset$. Thus, by induction, $P \xrightarrow{\alpha}_\bullet R'$ for some α and R' with $\langle\alpha\rangle = \beta$ and $\mathscr{T}_\nu(R') = U$. Now $R \xrightarrow{\alpha}_\bullet R'$ by **ide**.
- The cases that R is $\mathbf{0}$, $P + Q$ or $[x{=}y]P$ are trivial.
- Let $R = P[\sigma]$. Then $\mathscr{T}_\nu(R) = \mathscr{T}(P)[\sigma]$ and $\mathscr{T}(P) \xrightarrow{\delta} V$ for some δ with $\delta[\sigma] = \beta$, and some V with $V[\sigma] = U$. By definition P is clash-free. Considering that $\text{Rn}(R) \supseteq \{y[\sigma] \mid y \in \text{Rn}(P)\}$, one has $\text{no}(\delta) \cap \text{Rn}(P) = \emptyset$. So by induction $P \xrightarrow{\gamma}_\bullet P'$ for some γ and P' with $\langle\gamma\rangle = \delta$ and $\mathscr{T}_\nu(P') = V$. Let $\alpha := \gamma[\sigma]$. Then $\langle\alpha\rangle = \langle\gamma[\sigma]\rangle = \langle\gamma\rangle[\sigma] = \delta[\sigma] = \beta$. In case $z \in \text{bn}(\alpha)$ then $z = w[\sigma]$ with $w \in \text{fn}(\gamma)$, so $w \in \text{Rn}(P)$ by Lemma A.54 and hence $z \in \text{Rn}(R)$, so the clash-freedom of R implies $z \notin \text{fn}(R)$. Therefore, by **relabelling**, $R = P[\sigma] \xrightarrow{\alpha}_\bullet P'[\sigma]$, and $\mathscr{T}_\nu(P'[\sigma]) = \mathscr{T}_\nu(P')[\sigma] = V[\sigma] = U$.
- Let $R = P|Q$. Then $\mathscr{T}_\nu(R) = \mathscr{T}_\nu(P)|\mathscr{T}_\nu(Q)$. Suppose $\mathscr{T}_\nu(R) \xrightarrow{\beta} U$ is derived by **par**. Then $\mathscr{T}_\nu(P) \xrightarrow{\beta} V$ and $U = V|\mathscr{T}_\nu(Q)$. By definition P is clash-free. Since $\text{Rn}(P) \subseteq \text{Rn}(P|Q)$, $\text{no}(\beta) \cap \text{Rn}(P) = \emptyset$. So by induction $P \xrightarrow{\alpha}_\bullet P'$ for some α and P' with $\langle\alpha\rangle = \beta$ and $\mathscr{T}_\nu(P') = V$. By Lemma A.54 $\text{bn}(\alpha) \subseteq \text{Rn}(P) \subseteq \text{Rn}(R)$, so $\text{bn}(\alpha) \cap \text{fn}(R) = \emptyset$ by the clash-freedom of R. Thus $R \xrightarrow{\alpha}_\bullet P'|Q$ by **par**, and $\mathscr{T}_\nu(P'|Q) = V|\mathscr{T}_\nu(Q) = U$.

  Now suppose $\mathscr{T}_\nu(R) \xrightarrow{\beta} U$ is derived by **e-s-com**. Then $\beta = [x{=}v]MN\tau$, $\mathscr{T}_\nu(P) \xrightarrow{M\bar{x}y} V$, $\mathscr{T}_\nu(Q) \xrightarrow{Nvy} W$ and $U = V|W$. By definition P and Q are clash-free. Since $\text{Rn}(P) \subseteq \text{Rn}(P|Q)$, $\text{no}(M\bar{x}y) \cap \text{Rn}(P) = \emptyset$. So by induction either $P \xrightarrow{M\bar{x}y}_\bullet P'$ or $P \xrightarrow{M\bar{x}(y)}_\bullet P'$ for some P' with $\mathscr{T}_\nu(P') = V$.
  In the first case $y \in \text{fn}(P) \subseteq \text{fn}(R)$ by Lemma A.39, so $y \notin \text{Rn}(R) \supseteq \text{Rn}(Q)$ by the clash-freedom of R. Hence also $\text{no}(Nvy) \cap \text{Rn}(Q) = \emptyset$. By induction $Q \xrightarrow{Nvy}_\bullet Q'$ for some Q' with $\mathscr{T}_\nu(Q') = W$. So $R \xrightarrow{\alpha}_\bullet P'|Q'$ by **e-s-com**, and $\mathscr{T}_\nu(P'|Q') = V|W = U$.
  In the second case $y \in \text{Rn}(P) \subseteq \text{Rn}(R)$ by Lemma A.54, so $y \notin \text{fn}(R) \cup \text{Rn}(Q)$ by the clash-freedom of R. Hence $\text{no}(Nvy) \cap \text{Rn}(Q) = \emptyset$. By induction $Q \xrightarrow{Nvy}_\bullet Q'$ for some Q' with $\mathscr{T}_\nu(Q') = W$. So $R \xrightarrow{\alpha}_\bullet (vy)(P'|Q')$ by **e-s-close**, and $\mathscr{T}_\nu((vy)(P'|Q')) = \mathscr{T}_\nu(P'|Q') = V|W = U$.
- Finally, let $R = (vy)P$. Then $\mathscr{T}_\nu(R) = \mathscr{T}_\nu(P)$. Moreover, $\text{no}(\beta) \cap \text{Rn}(P) = \emptyset$. By induction, $P \xrightarrow{\alpha}_\bullet P'$ for some α and P' with $\langle\alpha\rangle = \beta$ and $\mathscr{T}_\nu(P') = U$.
  In case $y \notin \text{n}(\alpha)$, $R \xrightarrow{\alpha}_\bullet (vy)P'$ by **res**. Moreover, $\mathscr{T}_\nu((vy)P') = \mathscr{T}_\nu(P') = U$.
  In case $y \in \text{n}(\alpha) = \text{n}(\beta)$, using that $\text{Rn}(R) \ni y \notin \text{no}(\beta)$, β must have the form $M\bar{x}y$ with $y \neq x$ and $y \notin \text{n}(M)$. So α is either $M\bar{x}(y)$ or $M\bar{x}y$. If $\alpha = M\bar{x}(y)$ then $y \in \text{Rn}(P)$ by Lemma A.54, contradicting the clash-freedom of R. So $\alpha = M\bar{x}y$. Now $R \xrightarrow{M\bar{x}(y)}_\bullet P'$ by **symb-open**. □

Theorem A.63. If P is clash-free then $\mathscr{T}_\nu(P) \leftrightarroweq P$.

Proof. It suffices to show that the symmetric closure of

$$\mathscr{R} := \left\{(R, \mathscr{T}_\nu(R)) \mid R \text{ in } \pi^{\rho}_{ES}(\mathcal{N}, \mathcal{R}) \text{ is clash-free}\right\}.$$

is a strong barbed bisimulation. So suppose R is clash-free.

Let $R \xrightarrow{\tau}_\bullet R'$. Then $\mathscr{T}_\nu(R) \xrightarrow{\tau} \mathscr{T}_\nu(R')$ by Lemma A.61. Moreover, R' is clash-free by Lemma A.59, so $R' \mathscr{R} \mathscr{T}_\nu(R')$.

Let $\mathscr{T}_\nu(R) \xrightarrow{\tau} U'$. Then, by Lemma A.62, $R \xrightarrow{\tau}_\bullet R'$ for some R' with $\mathscr{T}_\nu(R') = U'$. Moreover, R' is clash-free by Lemma A.59, so $R' \mathscr{R} U'$.

Now let $R{\downarrow_b}$ with $b \in \mathcal{Z} \cup \overline{\mathcal{Z}}$. Then $R \xrightarrow{by}_\bullet$ or $R \xrightarrow{b(y)}_\bullet$ for some y, using the definition of O in Section 4. So $\mathscr{T}_\nu(R) \xrightarrow{by}$ by Lemma A.61. Thus $\mathscr{T}_\nu(R){\downarrow_b}$.

Finally, let $\mathscr{T}_\nu(R){\downarrow_b}$ with $b = x \in \mathcal{Z}$ or $b = \bar{x}$ with $x \in \overline{\mathcal{Z}}$. Then $\mathscr{T}_\nu(R) \xrightarrow{by} U'$ for some y and U'. Since R is clash-free, $\textsc{Rn}(R) \subseteq \mathcal{R}$, so $x \notin \textsc{Rn}(P)$. By Lemma A.43 I may assume that if $b = x$ then $y \notin \textsc{Rn}(R)$. Hence $\textsc{no}(by) \cap \textsc{Rn}(R) = \emptyset$. Thus, by Lemma A.62, $R \xrightarrow{by}_\bullet R'$ or $R \xrightarrow{b(y)}_\bullet R'$ for some R'. Hence $R{\downarrow_b}$. □

A.10 The last step

The language $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ can almost be recognised as an instance of $\text{CCS}^{\text{trig}}_\gamma$. Let Act be the set of all actions $M\tau$, $M\bar{x}y$ and Mxy with names from $\mathcal{H}$. As parameters of $\text{CCS}^{\text{trig}}_\gamma$ I take $\mathcal{K}$ to 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, and $\mathscr{A} := Act\backslash\{\tau\}$. The set $\mathscr{S} \subseteq \mathscr{A}$ of synchronisations consists of all actions $M\tau$ with $M \neq \varepsilon$. The communication function $\gamma : (\mathscr{A}\backslash\mathscr{S})^2 \rightharpoonup \mathscr{S}\cup\{\tau\}$ is given by $\gamma(M\bar{x}y, Nvy) = [x{=}v]MN\tau$, and its commutative variant. Now the parallel composition of $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ turns out to be the same as for this instance of $\text{CCS}^{\text{trig}}_\gamma$. Likewise, the silent and output prefixes are instances of $\text{CCS}^{\text{trig}}_\gamma$ prefixing, and the agent identifiers of $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ are no different from $\text{CCS}^{\text{trig}}_\gamma$ agent identifiers. However, the input prefix of $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ does not occur in $\text{CCS}^{\text{trig}}_\gamma$. Yet, one can identify $Mx(y).P$ with $\sum_{z\in\mathcal{H}} Mxz.(P[\{z/y\}^{\mathcal{S}}])$, for both processes have the very same outgoing transitions. The $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ matching operator is no different from the triggering operator of Meije or $\text{CCS}^{\text{trig}}_\gamma$ (see Section 12): both rename only the first actions their argument process can perform, namely by adding a single match $[x{=}y]$ in front of each of them—this match is suppressed when $x{=}y$.

This yields to the following translation from $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ to $\text{CCS}^{\text{trig}}_\gamma$:

$$\begin{array}{lll@{\qquad}lll}
\mathscr{T}_\gamma(\mathbf{0}) & := \mathbf{0} & & \mathscr{T}_\gamma([x{=}y]P) & := [x{=}y]{\Rightarrow}\mathscr{T}_\gamma(P)\\
\mathscr{T}_\gamma(M\tau.P) & := M\tau.\mathscr{T}_\gamma(P) & & \mathscr{T}_\gamma(P \mid Q) & := \mathscr{T}_\gamma(P)\|\mathscr{T}_\gamma(Q)\\
\mathscr{T}_\gamma(M\bar{x}y.P) & := M\bar{x}y.\mathscr{T}_\gamma(P) & & \mathscr{T}_\gamma(P+Q) & := \mathscr{T}_\gamma(P)+\mathscr{T}_\gamma(Q)\\
\mathscr{T}_\gamma(Mx(y).P) & := \textstyle\sum_{z\in\mathcal{H}} Mxz.\big(\mathscr{T}_\gamma(P)[\{z/y\}]\big) & & \mathscr{T}_\gamma(A) & := A
\end{array}$$

where the $\text{CCS}^{\text{trig}}_\gamma$ defining equations $A = \mathscr{T}_\gamma(P)$ of agent identifiers A are inherited verbatim from $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$.

Here the use of the triggering operator can be avoided by restricting attention to the π-calculus with implicit matching. For that language the clause for $\mathscr{T}_\gamma([x{=}y]P)$ can be dropped, at the expense of the addition of the blue Ms above, which are absent when dealing with the full π-calculus.

Theorem A.64. $\mathscr{T}_\gamma(P) \leftrightarrows P$ for each $\pi^\dagger_{ES}(\mathcal{N},\mathcal{R})$ process P.

Proof. Trivial. □

Putting all steps of my translation from $\pi_L(\mathcal{N})$ to $\text{CCS}^{\text{trig}}_\gamma$ together, I obtain

$$\begin{array}{lll@{\qquad}lll}
\mathscr{T}(\mathbf{0}) & := \mathbf{0} & & \mathscr{T}([x{=}y]P) & := [x{=}y]{\Rightarrow}\mathscr{T}(P)\\
\mathscr{T}(M\tau.P) & := M\tau.\mathscr{T}(P) & & \mathscr{T}(P \mid Q) & := \mathscr{T}(P)[\ell] \parallel \mathscr{T}(Q)[r]\\
\mathscr{T}(M\bar{x}y.P) & := M\bar{x}y.\mathscr{T}(P) & & \mathscr{T}(P+Q) & := \mathscr{T}(P)+\mathscr{T}(Q)\\
\mathscr{T}(Mx(y).P) & := \textstyle\sum_{z\in\mathcal{H}} Mxz.\big(\mathscr{T}(P)[\{z/y\}^{\mathcal{S}}]\big) & & \mathscr{T}(A(\vec{y})) & := A[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]\\
\mathscr{T}((\nu y)P) & := \mathscr{T}(P)[p_y] & & &
\end{array}$$

where the $\text{CCS}^{\text{trig}}_\gamma$ 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 $\pi_L(\mathcal{N})$ agent identifier A. Abbreviating $[\{z/y\}^{\mathcal{S}}]$ by $[z/y]$ and $[\{\vec{y}/\vec{x}\}^{\mathcal{S}}]$ by $[\vec{y}/\vec{x}]$, this is the translation presented in Section 9.