BRICS RS-97-52 P. Quaglia: On the Finitary Characterization of π-Congruences

BRICS

Basic Research in Computer Science

On the Finitary Characterization of π-Congruences

Paola Quaglia

BRICS Report Series RS-97-52

ISSN 0909-0878 December 1997

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/52/`

On the finitary characterization of π-congruences

Paola Quaglia*

BRICS**, Aarhus University

Abstract

Some alternative characterizations of late full congruences, either strong or weak, are presented. Those congruences are classically defined by requiring the corresponding ground bisimilarity under all name substitutions.

We first improve on those infinitary definitions by showing that congruences can be alternatively characterized in the π-calculus by sticking to a finite number of carefully identified substitutions, and hence, by resorting to only a finite number of ground bisimilarity checks.

Then we investigate the same issue in both the ground and the non-ground $\pi\xi$-calculus, a CCS-like process algebra whose ground version has already been proved to coincide with ground π-calculus. The $\pi\xi$-calculus perspective allows processes to be explicitly interpreted as functions of their free names. As a result, a couple of alternative characterizations of π-congruences are given, each of them in terms of the bisimilarity of one single pair of $\pi\xi$-processes. In one case, we exploit λ-closures of processes, so inducing the effective generation of the substitutions necessary to infer non-ground equivalence. In the other case, a more promising call-by-need discipline for the generation of the wanted substitutions is used. This last strategy is also adopted to show a coincidence result with open semantics.

By minor changes, all of the above characterizations for late semantics can be suited for congruences of the early family.

0 Introduction

The π-calculus [MPW92] is a by now well-known process algebra that allows the description of mobile systems. The calculus, that has been shown to have great flexibility and expressive power, exploits a name-passing interaction paradigm. Names, synonyms of channels, can be sent around and received, possibly changing the local/global acquaintances of the inputting process. For instance, conveying that $\overline{x}z$ and $x(y)$ stay, respectively, for an output and an input action at channel x, the following behaviour hold of process $\overline{x}z.P \mid x(y).Q$.

$$\overline{x}z.P \mid x(y).Q \xrightarrow{\tau} P \mid Q\{z/y\}$$

*Funded by the EU, under the Marie Curie TMR Programme. Address: BRICS, Dep. of Computer Science, Aarhus University, Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark. email: `quaglia@brics.dk`.

**Basic Research in Computer Science, Centre of the Danish National Research Foundation.

Namely, the given process can perform a τ-action and then behave like P in parallel with the actual instance of Q where all the free occurrences of y have been replaced by z.

As the above transition highlights, name substitution has a fundamental role in the operational semantics of the calculus. This same influence is mirrored at the extensional level of bisimulation semantics, essentially due to the fact that substitutions can increase process move capabilities. For instance, unless the notion of name instantiation is moved inside the bisimulation definition itself (like, *e.g.*, in the open semantics [San96]), bisimulation fails to be a congruence w.r.t. input action. The canonical example to see this is in terms of the interleaving interpretation of parallel composition, which equates the two processes $(\overline{x} \mid y)$ and $(\overline{x}.y+y.\overline{x})$ (we are omitting unnecessary syntactic details). Each of the – either strong or τ-forgetting – *late* or *early* bisimulation semantics, hereby generically denoted by $\dot{\asymp}$, is such that

$$\begin{array}{rcl} \overline{x} \mid y & \dot{\asymp} & \overline{x}.y + y.\overline{x} \\ x(y).(\overline{x} \mid y) & \dot{\not\asymp} & x(y).(\overline{x}.y + y.\overline{x}) \end{array}$$

The reason for the above inequivalence is that, whenever the leading input action $x(y)$ causes y to be instantiated by x, the left-hand process $x(y).(\overline{x} \mid y)$ can perform a τ-move that the right-hand agent is not able to match.

The usual way to extract a full congruence $\asymp$ from the *ground* relation $\dot{\asymp}$ passes through a closure over name substitutions, so that classical definitions read as follows:

$$P \asymp Q \text{ iff } P\sigma \mathrel{\dot{\asymp}} Q\sigma \text{ for all substitutions } \sigma$$

Full congruences, also called *non-ground* relations, are pragmatically much more appealing than their ground duals. Though, the universal quantification over substitutions induces a prohibitive requirement to check in practice.

The main concern of this paper is to present alternative finitary characterizations of late, either strong or weak, π-calculus full congruences. A coincidence result with open semantics is also proved. Our investigation is carried on in both the π-calculus and the $\pi\xi$-calculus, a generalization of the calculus appeared in [FMQ96].

As for π-calculus, by adequately classifying name substitutions, we show that $P \asymp Q$ can be expressed in terms of a finite number of bisimilarity checks $P\sigma \mathrel{\dot{\asymp}} Q\sigma$, for carefully chosen substitutions σ.

Then we turn to the $\pi\xi$-calculus, either retaining its ground form or defining a non-ground version of it. This allows us to provide a couple of alternative characterizations of late π-congruences as CCS-like bisimilarities [Mil89, HM85] of one suitable single pair of $\pi\xi$-processes, either ground or non-ground.

The ground $\pi\xi$-calculus was introduced with the intent of explicitly mirroring into a CCS-like setting the π-calculus meta-syntactic operation of substitution. The motivation for this being the possibility to easily export to the π-calculus some well-know results about the automated verification and the mathematical theory (logical, axiomatic, and denotational) of CCS equivalences (see, *e.g.*, [CPS93, HM85, ABV94, Abr91]). The main operational idea underpinning the $\pi\xi$-calculus is to avoid to directly apply name substitutions to π-processes. An explicit component ξ is deemed to serve the same purpose. It acts as some kind of environment and represents associations among names. So, the generic $\pi\xi$-process looks like $\xi :: P$, where P is an agent obtained by the usual π-calculus syntax added with abstraction prefixes '$\lambda y.$'. These last prefixes, in turn, are used to associate a concrete operational counterpart to the name substitution occurring in the definition of the input clause of π-calculus bisimulations, whose strong late version reads like the following:

if $P \xrightarrow{x(y)} P'$ with y fresh, then there is Q' such that
$Q \xrightarrow{x(y)} Q'$ and for all w, $P'\{w/y\}$ is late bisimilar to $Q'\{w/y\}$

Remarkably, the operational interpretation of λ-abstraction prefixes ensures that input parameters can always be actualized by a finite number of names. So, except for the unguarded behaviour of the replication operator, $\pi\xi$-processes can be modelled by finitely branching structures, which is generally not the case for data-dependent agents.

As we recalled, the main concern of the research leading to the $\pi\xi$-calculus has been to figure out an operational setting which could support a standard CCS bisimulation semantics, both strong and observational. Precisely, the $\pi\xi$-calculus operational semantics is defined as a two level transition system. The low level system leaves environments unaffected. It defines the relation $P \xrightarrow{\alpha,C} P'$ where C makes explicit the requirements on names which are to be met in order for the symbolic step to be converted into a concrete move. The top level transition system is then defined by rules of the following shape

$$\frac{P \xrightarrow{\alpha,C} P'}{\xi :: P \overset{\delta(\xi',\alpha,C)}{\longrightarrow\!\!\!\triangleright} \xi' :: P'} \quad \xi' \in \eta(\xi,\alpha,C) \qquad (\triangle)$$

where the function η takes care of extending the environment ξ with the name associations activated by the symbolic transition $P \xrightarrow{\alpha,C} P'$, while the function δ yields a concrete observable action. So, the extensional semantics of the calculus is given as standard strong and weak bisimulations and observational congruence.

In [FMQ95] and [FMQ96] it was shown that suitable distinct definitions of $(\triangle)$ and of the pair (η, δ) lead to the operational and axiomatic characterization of both late and early, either strong or weak, ground π-calculus semantics. The challenging issue of characterizing the corresponding full congruences was left open.

Here we fill that gap. We stress the correspondence between π-calculus name substitutions and $\pi\xi$-calculus environments, and prove the claim that reasoning about non-ground semantics imposes to regard any π-calculus process as an actual function of its free names.

First, relying on the ground $\pi\xi$-calculus, we show that suitable λ-closures of P and Q induce, in an effective way, the generation of those environments corresponding to the finite set of substitutions sufficient to infer $P \asymp Q$ from the ground bisimilarities of $P\sigma$ and $Q\sigma$. By this, we prove a coincide result with late strong and weak congruences, and corresponding axiomatic characterizations for finite processes.

Next, we argue that for strong semantics a similar result can be more efficiently achieved by adequately sophisticating the notion of environment and slightly modifying the top level transition system $(\triangle)$. This leads to the definition of the non-ground version of the $\pi\xi$-calculus. Such a calculus allows us to encode in a natural way a call-by-need strategy for the generation of the substitutions required to decide late full congruence. Finally, an analogous call-by-need instantiation discipline is used to provide an alternative characterization of open semantics.

By minor changes, all of the characterizations stated for late equivalence relations do hold for non-ground semantics of the early family [Qua96]. Also, by virtue of the peculiar $\pi\xi$-calculus interpretation of input actions, those characterizations allow finite π-processes to be modelled by finite transition systems. So, our results could be particularly useful in the perspective of verifying mobile systems.

The rest of the paper is organized as follows. Section 1 and Section 2 contain the necessary background about π-calculus and ground $\pi\xi$-calculus, respectively. In Section 3 we prove that each non-ground late semantics can be expressed as the conjunction of a finite number of corresponding ground bisimilarities. Building on this property, in Section 4 we characterize late π-congruences in terms of standard strong and weak bisimilarities of one single suitable pair of ground $\pi\xi$-processes. In Section 5 non-ground $\pi\xi$-calculus is introduced. Then, from a more promising perspective, yet another characterization of late strong non-groundness is stated, together with a coincidence result for open semantics. Eventually, Section 6 contains some concluding remarks. Also, Appendix A reports (minor modifications and extensions of) the encodings and constructions which appeared in [FMQ96] and are recalled to prove, in Appendix B, the characterization of open congruence.

1 The π-calculus

An overview of the π-calculus follows. Let $\mathcal{N}$ be a denumerably infinite set of names, ranged over by x, y, z, ... The syntax of π-calculus processes (ranged over by P, Q, ...) is defined by the following grammar:

$$P \; ::= \; \mathtt{nil} \;\Big|\; x(y).P \;\Big|\; \overline{x}y.P \;\Big|\; \tau.P \;\Big|\; P+P \;\Big|\; P \mid P \;\Big|\; [x=y]P \;\Big|\; (y)P \;\Big|\; !P$$

Most of the operators resemble the corresponding CCS constructors [Mil80, Mil89]. The main exceptions are as follows. The matching operator $[x=y]P$ is meant to test names for equality. The restriction operator $(y)P$ declares y to be a new name for local use in P. The replication (or bang) operator '!' is used to express infinite behaviours.

Prefixes $x(y)$, $\overline{x}y$, and τ are called, respectively, bound input, free output, and silent (or internal, or unobservable) action. The adjective *bound* recalls that brackets act as formal binder, namely all of the free occurrences of y in P are bound by $x(y)$ in $x(y).P$. Indeed, the restriction operator (y) in $(y)P$ is another kind of formal binder. Prefix $\overline{x}y$ is given the appellative of *free* as opposed to the bound output action $\overline{x}(y)$. This last action is not available at the syntactic level, and denotes the ability to communicate at x the private name y. Either in $x(y)$ or in $\overline{x}y$ or in $\overline{x}(y)$, the name x is said to be the *subject*, while y is called the *object*. Also, for the bound input $x(y)$, the name y is sometimes referred to either as *parameter* or as *placeholder*.

If a name is not bound, it is called free. The set of names occurring free in the action α is written $\mathrm{fn}(\alpha)$. Dually, the set of bound names is written $\mathrm{bn}(\alpha)$. The set of names of the action α, written $\mathrm{n}(\alpha)$, is defined to be the union of free and bound names. The unobservable action τ is such that $\mathrm{n}(\tau)=\emptyset$. Free and bound names of process P, denoted $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ respectively, are defined in the obvious way. Also, $\mathrm{n}(P)=\mathrm{fn}(P)\cup\mathrm{bn}(P)$, and $\mathrm{fn}(P,Q)$ is sometimes used as a shorthand for $\mathrm{fn}(P)\cup\mathrm{fn}(Q)$.

The π-calculus operational semantics is defined inductively, in the style of [Plo81], by the rules shown in Tab. 1 together with additional symmetric rules for binary operators. Some of the rules, and in general most of the π-calculus theory, make use of the meta-syntactic operation of name substitution. Name substitutions (ranged

$$\tau.P \xrightarrow{\tau} P \qquad\qquad x(y).P \xrightarrow{x(w)} P\{w/y\} \quad w \notin \mathrm{fn}((y)P)$$

$$\overline{x}y.P \xrightarrow{\overline{x}y} P \qquad\qquad \frac{P \xrightarrow{\alpha} P'}{[x=x]P \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad\qquad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{y/z\}} \qquad\qquad \frac{P \xrightarrow{\overline{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P \mid Q \xrightarrow{\tau} (w)(P' \mid Q')}$$

$$\frac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'} \quad y \notin \mathrm{n}(\alpha) \qquad\qquad \frac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(w)} P'\{w/y\}} \quad y \neq x,\ w \notin \mathrm{fn}((y)P')$$

$$\frac{P \mid !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$$

Table 1: the π-calculus operational semantics

over by σ, σ', ...) are functions from $\mathcal{N}$ to $\mathcal{N}$ defined almost everywhere as the identity. Sometimes, when the substitution σ differs from the identity for the names in $\{x_1, \ldots, x_n\}$, σ is simply written $\{x_1\sigma/x_1, \ldots, x_n\sigma/x_n\}$. The sets $\{x_1, \ldots, x_n\}$ and $\{x_1\sigma, \ldots, x_n\sigma\}$ are then referred to as domain and codomain of σ, written $\mathrm{Dm}(\sigma)$ and $\mathrm{Im}(\sigma)$, respectively. The term $P\sigma$ denotes the process obtained from P by simultaneously substituting, for each x, any free occurrence of x in P by $x\sigma$, with change of bound names to avoid name clashes. We will use the symbol $\equiv_\alpha$ to denote the relation of α-convertibility. Syntactic identity of P and Q will be written $P \equiv Q$.

Process operational behaviours are quotiented by strong or weak equivalence relations defined as bisimulation games. Distinct semantics have been defined, reflecting different strategies for the actual instantiation of names. For instance, depending on the assumed relative atomicity of the act of committing on the input channel and the act of instantiating the input placeholder, π-calculus bisimulation semantics naturally split into *early* and *late* families [MPW92].

Late semantics give input actions a functional operational intuition. When inputting, a process becomes a function of the actual transmitted name. So, the input clauses of late bisimulation games claim that the derivatives of the inputting processes continue to simulate each other for all the possible instantiations of the formal parameter. In the following, as usual, we let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$, the labelled relation $\xRightarrow{\alpha}$ be $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$, and $\xRightarrow{\hat{\alpha}}$ be $\Longrightarrow$ if $\alpha = \tau$, $\xRightarrow{\alpha}$ otherwise.

Definition 1 Let $\mathcal{S}$ be a binary symmetric relation over π-calculus processes. Then

- $\mathcal{S}$ is a *strong late ground bisimulation* if $P\ \mathcal{S}\ Q$ implies that
  - if $P \xrightarrow{\alpha} P'$ with $\alpha \neq x(y)$ and $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \xrightarrow{\alpha} Q'$ and $P'\ \mathcal{S}\ Q'$
  - if $P \xrightarrow{x(y)} P'$ with $y \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \xrightarrow{x(y)} Q'$ and, for all w, $P'\{w/y\}\ \mathcal{S}\ Q'\{w/y\}$
- $\mathcal{S}$ is a *weak late ground bisimulation* if $P\ \mathcal{S}\ Q$ implies that
  - if $P \xrightarrow{\alpha} P'$ with $\alpha \neq x(y)$ and $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ and $P'\ \mathcal{S}\ Q'$
  - if $P \xrightarrow{x(y)} P'$ with $y \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \Longrightarrow\xrightarrow{x(y)} Q'$ and, for all w, $P'\{w/y\}\ \mathcal{S}\ Q'\{w/y\}$

P is *strong late ground bisimilar* to Q, written $P \dot{\sim}_L Q$, if $P\ \mathcal{S}\ Q$ for some strong late ground bisimulation $\mathcal{S}$.

P is *weak late ground bisimilar* to Q, written $P \dot{\approx}_L Q$, if $P\ \mathcal{S}\ Q$ for some weak late ground bisimulation $\mathcal{S}$.

P is *weak late ground equal* to Q, written $P \dot{\simeq}_L Q$, iff $P \dot{\approx}_L Q$ and whenever $P \xrightarrow{\tau} P'$ then for some Q', $Q \stackrel{\tau}{\Longrightarrow} Q'$ with $P' \dot{\approx}_L Q'$, and symmetrically. $\square$

A couple of observations about the previous definition are worth noticing. First, the input clause of weak late ground bisimilarity breaks down the double arrow of the usual CCS definition of weak bisimilarity. Nevertheless, the claim of instantiating y just after the input move is not surprising if we think again of the late functional intuition about input steps. Interpreting the derivative processes P' and Q' as functions of y, requires the bisimilarity of $P'\{w/y\}$ and $Q'\{w/y\}$ for all w, without giving Q' the opportunity of silently slip to a distinct function. Second, weak late ground bisimilarity is not preserved by non-deterministic context. Analogously to the CCS case, to guarantee substitutivity in those contexts, weak late ground equality requires each initial silent action to be matched by at least one unobservable move.

All the above late behavioural relations are equivalences, but neither of them is preserved by substitution of names, and then by input prefix. For instance, process $P \equiv [x = y]\overline{x}x.\mathtt{nil}$, having no outgoing transition, is either bisimilar or equal, in each of the above senses, to the inactive process $\mathtt{nil}$. This is no longer true after substituting x for y in P, and hence when checking, *e.g.*, the corresponding equivalence of $x(y).P$ and $x(y).\mathtt{nil}$.

Indeed, as it is common in the π-calculus literature, the adjective *ground*, together with the use of a dotted relational symbol, stays just to recall that the actual equivalences are not preserved by substitution of names. Such a propriety is achieved by requiring bisimilarity over all name substitutions, so getting the following relations, sometimes called *non-ground* bisimilarities.

Definition 2 Let P and Q be π-calculus processes. Then P and Q are

- *strong late bisimilar*, written $P \sim_L Q$, if $P\sigma \dot{\sim}_L Q\sigma$ for all substitutions σ
- *weak late bisimilar*, written $P \simeq_L Q$, if $P\sigma \dot{\simeq}_L Q\sigma$ for all substitutions σ

Also, we write $P \approx_L Q$, if $P\sigma \dot{\approx}_L Q\sigma$ for all substitutions σ. $\square$

As far as the definition of $\approx_L$ is concerned, notice that, although preserved by name substitutions, this equivalence, just as its ground version, is not preserved by non-deterministic context.

In the following, unless otherwise stated, we will always refer to bisimilarities of the late family, so the adjective 'late' will be freely omitted. Further omissions will be those of unnecessary syntactic details, like, *e.g.*, trailing `.nil`'s.

2 The ground $\pi\xi$-calculus

This section provides an overview of the ground $\pi\xi$-calculus, a generalization of the calculus appeared in [FMQ96]. Also, some coincidence results with ground π-calculus semantics are collected. In Appendix A, we report the main constructions and encodings needed to prove those results, and we comment on which extent the actual proofs differ from those presented in [FMQ96]. For complete details the reader is referred to [Qua96].

In the $\pi\xi$-calculus, processes (ranged over by $S, S_1, \dots$) are written $\xi :: P$. The right component of the state operator '$\_ :: \_$' is (essentially) a π-calculus process, while ξ keeps track of the associations among names carried out in the past of the ongoing computation. Name substitutions are never applied to the right component of $\xi :: P$. Hence ξ can be viewed as an environment giving the actual associations of names. The syntax of the right component of the process $\xi :: P$ is formally defined as follows.

$$P ::= \mathtt{nil} \;\Big|\; x(y).P \;\Big|\; \lambda y.P \;\Big|\; \overline{x}y.P \;\Big|\; \tau.P \;\Big|\; P + P \;\Big|\; P \mid P \;\Big|\; [x = y]P \;\Big|\; (y)P \;\Big|\; !P$$

Also, we assume that there is no homonymy either among bound names or among free and bound names of P. This assumption, that amounts to reason up to α-conversion, could be easily fulfilled by using an indexing mechanism *a là* De Bruijn. But for this requirement on names, prefixes λy, which act as formal binders, are the only novelty w.r.t. either the π-calculus syntax or the process algebra presented in [FMQ96]. For the sake of convenience, in spite of new prefixes, we will often refer to the right component of any $\pi\xi$-calculus process as to π-calculus process. The operational role of λ-prefixes is to call for an actual instantiation of the formal parameter y. In the calculus appeared in [FMQ96], an analogous observable result was obtained by means of more complex environments ξ. The introduction of λ-prefixes allows to simplify the semantic model in [FMQ96] and most of the constructions and intermediate results needed to prove characterization theorems.

The operational semantics of the $\pi\xi$-calculus follows the SOS style and is based on a two-stage approach. The first stage consists of the definition of a symbolic semantics where transition labels record requirements on names. The evaluation of those requirements is then the main concern of the top level transition system.

The symbolic operational semantics is given by the axioms and rules reported in Tab. 2 together with symmetric rules for choice and asynchronous parallel composition. At this first operational level, requirements on names are not checked and name instantiation is not applied to processes, but rather recorded by transition labels (ranged over by ω, $\omega', \dots$). Labels are pairs of the form $\langle \alpha, C \rangle$: the first component is essentially an action in the same sense of the π-calculus; the second component, called *obligation*, is a logical formula that codes requirements on names.

$$\tau.P \xrightarrow{\langle\tau,\mathtt{true}\rangle} P$$

$$\overline{x}y.P \xrightarrow{\langle\overline{x}y, x\downarrow\rangle} P$$

$$x(y).P \xrightarrow{\langle x(y), x\downarrow\rangle} P$$

$$\lambda y.P \xrightarrow{\langle [y],\mathtt{true}\rangle} P$$

$$\frac{P \xrightarrow{\omega} P'}{[x=y]P \xrightarrow{\mu^x_y(\omega)} P'}$$

$$\frac{P \xrightarrow{\omega} P'}{P+Q \xrightarrow{\omega} P'}$$

$$\frac{P \xrightarrow{\omega} P'}{P \mid Q \xrightarrow{\omega} P' \mid Q}$$

$$\frac{P \xrightarrow{\omega} P' \qquad Q \xrightarrow{\omega'} Q'}{P \mid Q \xrightarrow{\omega \| \omega'} P' \mid Q'}$$

$$\frac{P \xrightarrow{\omega} P'}{(y)P \xrightarrow{o_y\omega} P'}$$

$$\frac{P \xrightarrow{\omega} P'}{(y)P \xrightarrow{\nu_y\omega} (y)P'}$$

$$\frac{(P)^{\mathrm{dec\_0}} \mid !(P)^{\mathrm{dec\_1}} \xrightarrow{\omega} P'}{!P \xrightarrow{\omega} P'}$$

$$\mu^x_y\langle\alpha, C\rangle \quad = \quad \langle\alpha, C \wedge x = y\rangle$$

$$\langle\alpha_1, C_1'\rangle \parallel \langle\alpha_2, C_2'\rangle = \begin{cases} \langle\tau[y/w], C_1 \wedge C_2 \wedge x = z\rangle & \text{if } \alpha_1 \in \{\overline{x}y, \overline{x}(y)\},\quad C_1' = C_1 \wedge x\downarrow \\ & \text{and } \alpha_2 = z(w),\quad C_2' = C_2 \wedge z\downarrow \\ & \text{or symmetrically} \\ \langle\tau, \mathtt{false}\rangle & \text{otherwise} \end{cases}$$

$$\nu_y\langle\alpha, C\rangle = \begin{cases} \langle\alpha, C \wedge y \neq z\rangle & \text{if } \alpha = \overline{x}z \\ \langle\alpha, C\rangle & \text{otherwise} \end{cases}$$

$$o_y\langle\alpha, C\rangle = \begin{cases} \langle\overline{x}(z), C \wedge y = z\rangle & \text{if } \alpha = \overline{x}z \\ \langle\alpha, \mathtt{false}\rangle & \text{otherwise} \end{cases}$$

Table 2: symbolic operational semantics

The execution of the new prefixes λy results in the actions $[y]$ which, although resembling the concretions of [Mil92], have no counterpart in the monadic π-calculus. Also, differently from the π-calculus, communication is characterized by a single inference rule. More precisely, we avoided to use the so-called *Close* rule which describes the communication of a private name and causes a restriction to appear

$$
\begin{array}{rcl}
(P)^{\mathtt{dec\_}a} & = & \mathtt{case}\ P\ \mathtt{in} \\
 & & \begin{array}{rcl}
\mathtt{nil} & : & \mathtt{nil} \\
\tau.P_1 & : & \tau.(P_1)^{\mathtt{dec\_}a} \\
\overline{x}y.P_1 & : & \overline{x}y.(P_1)^{\mathtt{dec\_}a} \\
x(y^{(s)}).P_1 & : & x(y^{(s\cdot a)}).(P_1\{y^{(s\cdot a)}/y^{(s)}\})^{\mathtt{dec\_}a} \\
\lambda y^{(s)}.P_1 & : & \lambda y^{(s\cdot a)}.(P_1\{y^{(s\cdot a)}/y^{(s)}\})^{\mathtt{dec\_}a} \\
[x=y]P_1 & : & [x=y](P_1)^{\mathtt{dec\_}a} \\
P_1 + P_2 & : & (P_1)^{\mathtt{dec\_}a} + (P_2)^{\mathtt{dec\_}a} \\
P_1 \mid P_2 & : & (P_1)^{\mathtt{dec\_}a} \mid (P_2)^{\mathtt{dec\_}a} \\
(y^{(s)})P_1 & : & (y^{(s\cdot a)})(P_1\{y^{(s\cdot a)}/y^{(s)}\})^{\mathtt{dec\_}a} \\
!P_1 & : & !(P_1)^{\mathtt{dec\_}a}
\end{array} \\
 & & \mathtt{end\_case}
\end{array}
$$

Table 3: definition of $(\_)^{\mathtt{dec\_}a}$

on top of synchronizing processes. In the $\pi\xi$-calculus the information about privacy of names is completely captured by environments. Before plunging processes into environments, we only impose a consistency requirement: no process must be allowed to commit on a link which is not known outside. To this end, the input and the output transition labels include the obligation $x\downarrow$. It actually demands for a delayed check against the privacy of channel x.

The non homonymy condition we assume on names has to dynamically hold for every expansion of the replicated process $!P$. So, decorated versions of P are used in the premise of the rule for the bang operator. This is meant to assure that any parallel copy of the agent P borns uniquely its bound names. The idea is to label bound names by superscripts and to expand, *e.g.*, process $!x(y).y$ into $x(y^{(0)}).y^{(0)} \mid !x(y^{(1)}).y^{(1)}$. To this purpose, we use $y^{(s)}$ to indicate that the name y is superscripted by the finite string s of zeros and ones. Then $y^{(s_1)} \neq y^{(s_2)}$ whenever $s_1 \neq s_2$, while y is a short-hand for y superscripted by the empty string[1]. We feel free to omit the indication of superscripts when it is clear from the context the identity of names. Function $(\_)^{\mathtt{dec\_}a}$ is defined in Tab. 3, where $s{\cdot}a$ denotes the concatenation of the string s with $a \in \{0,1\}$.

We now comment on environments, left components of $\pi\xi$-processes. An environment is a set of equations on two distinct entities: names and constants. Constants are taken from a denumerably infinite set $\mathcal{D}$ that is ranged over by $c, c_1, c_2, \ldots$, and disjoint from the set $\mathcal{N}$ of names. As we are dealing with π-calculus processes where there is no homonymy either among bound names or among free and bound names, we suppose that free and bound names of π-processes are taken from two disjoint, infinite, subsets of $\mathcal{N}$, called $\mathcal{N}_I$ and $\mathcal{N}_{RT}$, respectively. Similarly, we assume that the set of constants $\mathcal{D}$ is partitioned into two disjoint sets $\mathcal{D}_I$ and $\mathcal{D}_{RT}$.

[1]We would like to thank Franck van Breugel for pointing out the misbehaviour of a previous decoration operator.

Definition 3 A *ground environment* ξ is an equivalence relation over $\mathcal{N} \cup \mathcal{D}$ which is:

- *consistent*: $c_i \; \xi \; c_j$ implies $c_i = c_j$
- *finitely active*: the set $\{(a, b) \mid a \; \xi \; b \text{ and } a \neq b\}$ is finite

Given an environment ξ, we denote by $[a]_\xi$ the equivalence class of ξ containing a. Such a class is said to be *undefined* if $\mathcal{D} \cap [a]_\xi = \emptyset$, *defined-by-constant* – or shortly *defined* – otherwise.
A constant c is *active* in ξ iff there exists y such that $y \in [c]_\xi$, it is *inactive* otherwise.
The family of all ground environments is denoted by $\mathcal{E}$, and the identity ground environment is denoted by $\mathrm{Id}_\mathcal{E}$.
Let R_1, R_2 be relations over $\mathcal{N} \cup \mathcal{D}$. Then $R_1 + R_2$ is defined as the smallest equivalence relation including $(R_1 \cup R_2)$. □

We will often refer to ground environments simply as to environments. Also, in view of the above mentioned consistency requirement, we will let ξ sometimes assume the reading of a partial function. Whenever $(y \; \xi \; c)$, we will denote the constant c as $\xi(y)$ and will say that the partial function $\xi(\_)$ is defined on y, denoted by $\xi(y)\downarrow$. If $\xi(\_)$ is not defined on y, then we will write $\xi(y)\uparrow$.

The coming definition identifies the environments where π-calculus processes are let start running.

Definition 4 Letting $N \subseteq \mathcal{N}_I$, the *initial ground environment* ξ^N is defined as

$$\xi^N = \mathrm{Id}_\mathcal{E} + \{(x, \imath(x)) \mid x \in N\}$$

where $\imath : \mathcal{N}_I \to \mathcal{D}_I$ is a bijective mapping, and $\mathcal{D}_I \cap \mathcal{D}_{RT} = \emptyset$ and $\mathcal{D}_I \cup \mathcal{D}_{RT} = \mathcal{D}$ and $\mathcal{N}_I \cap \mathcal{N}_{RT} = \emptyset$ and $\mathcal{N}_I \cup \mathcal{N}_{RT} = \mathcal{N}$. □

During the execution of a $\pi\xi$-calculus process, the generation of fresh constants is needed. To this end, we assume the existence of the following suitable functions. Function $\mathrm{all}\mathcal{D} : \mathcal{E} \longrightarrow 2^\mathcal{D}_f$ returns the finite set of all the constants which are active in its argument. Function $\mathrm{new}\mathcal{D} : \mathcal{E} \longrightarrow \mathcal{D}_{RT}$ yields a constant which is inactive in its argument. Here we assume that $\mathrm{new}\mathcal{D}$ only depends on the active run-time constants in the argument, so that whenever $\mathrm{all}\mathcal{D}(\xi_1) \cap \mathcal{D}_{RT} = \mathrm{all}\mathcal{D}(\xi_2) \cap \mathcal{D}_{RT}$ the equality $\mathrm{new}\mathcal{D}(\xi_1) = \mathrm{new}\mathcal{D}(\xi_2)$ does hold.

The late operational semantics of $\pi\xi$-calculus processes is described by the inference rules of Tab. 4. Both rules allow to infer the behaviour of $\xi :: P$ from a symbolic transition of P and the definition of the result and of the update function, δ and η respectively. The function δ yields the observable result out of the transition. The possibly many-valued function η takes care of extending the environment ξ with the name associations activated by the transition. In defining η, the satisfiability of the obligation C is checked by means of a specialized evaluation function. Whenever the requirements expressed by C are not met in the environment ξ, the application η returns the empty set, so that the $\pi\xi$-calculus process at hand is unable to move.

The rule associated with symbolic inputs shows the first-class role of instantiation in the $\pi\xi$-calculus. Correspondingly to the execution of the symbolic action $x(y)$, the process $\xi :: P$ evolves to the process $\xi' :: \lambda y.P'$ whose right component is an explicit function of the placeholder y. The next – compulsory – move of the

$$\frac{P \xrightarrow{\omega} P' \qquad \omega \neq \langle x(y), C\rangle}{\xi :: P \overset{\delta(\xi',\omega)}{\longrightarrow\!\!\triangleright} \xi' :: P'} \quad \xi' \in \eta(\xi,\omega)$$

$$\frac{P \xrightarrow{\langle x(y),C\rangle} P'}{\xi :: P \quad \overset{\delta(\xi',\langle x(y),C\rangle)}{\longrightarrow\!\!\triangleright} \quad \xi' :: \lambda y.P'} \quad \xi' \in \eta(\xi,\langle x(y), C\rangle)$$

Table 4: definition of $\longrightarrow\!\!\triangleright$

process $\xi' :: \lambda y.P'$ is the instantiation of the input parameter. Notice, in fact, that the operational description guarantees instantiations to have priority over any other action even when the inputting agent is, *e.g.*, underneath a parallel composition.

The definition of the update and of the result function (η,δ) is reported in Tab. 5. In the definition of η, elements are coerced to be singleton sets. The first step in computing the update function η consists in checking the satisfiability of the obligation. If the obligation evaluates to false in the environment, then η results in the empty set. Otherwise, depending on the structure of the action α, the update function yields a set of environments obtained by possibly adding a pair to ξ. Any information about the privacy of names is consistently captured by environments at the top level. So, outputting the – syntactically – free name y may be the same as outputting a private name. Precisely, if $\xi(y)\uparrow$ then $\eta(\xi, \overline{x}y, C)$ is exactly the same as $\eta(\xi, \overline{x}(y), C)$. It results into an environment where the name y is associated with a new constant.

One comment is in place for placeholder instantiations. Correspondingly to those actions, the function η yields as many environments as the possible choices of c in all$\mathcal{D}(\xi)$, plus a new constant. This corresponds to instantiate y with (possibly a superset of) all the free names of the process at hand, plus a new fresh one. The intuitive reason for this relies on the following. Given any π-calculus process P', the agents $P'\{z/y\}$ and $P'\{u/y\}$ have analogous move potentials whenever $z, u \notin \mathrm{fn}(P')$. Precisely, either $P'\{z/y\}$ or $P'\{u/y\}$ have the same action capabilities as P' has. As a consequence of the above observation, if P performs the input $x(y)$ transforming into P', then the relevant instantiations of y in $P'\{w/y\}$ are given by those names w such that $w \in \mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \{y\}$. Our definition of $\eta(\xi, [y], C)$ is just meant to mimic those instantiations. The function all$\mathcal{D}(\xi)$ plays the role of $\mathrm{fn}(P)$ while new$\mathcal{D}(\xi)$ stays for the set $\{y\}$. Notice, however, that at any time during execution only finitely many constants are active. This implies that finite $\pi\xi$-processes can be always represented by labelled trees which are finitely branching.

Consider now the result function δ. It yields either τ or the constant(s) associated with the relevant name(s). The parameter of the action $x(y)$ is not relevant. When inputting, the process becomes a function of y and then depends on the actual instantiation of the placeholder. The parameter will become observable at the next step.

What is crucial here is that the result function δ computes concrete – vs. sym-

$$
\begin{array}{rcl}
[\![C]\!]\xi & = & \texttt{case}\ C\ \texttt{in} \\
& & \begin{array}{rcl}
\texttt{true} & : & \texttt{tt} \\
\texttt{false} & : & \texttt{ff} \\
x\!\downarrow & : & \xi(x)\!\downarrow \longrightarrow \texttt{tt}\,,\quad \texttt{ff} \\
x = y & : & x\ \xi\ y \longrightarrow \texttt{tt}\,,\quad \texttt{ff} \\
x \neq y & : & x\ \xi\ y \longrightarrow \texttt{ff}\,,\quad \texttt{tt} \\
C_1 \wedge C_2 & : & [\![C_1]\!]\xi\ \texttt{and}\ [\![C_2]\!]\xi
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\eta\xi\alpha C & = & \neg[\![C]\!]\xi \longrightarrow \emptyset,\quad \texttt{case}\ \alpha\ \texttt{in} \\
& & \qquad \begin{array}{rcl}
\tau & : & \xi \\
\tau[x/y] & : & \xi + (y, x) \\
\overline{x}(y), \overline{x}y & : & \xi(y)\!\downarrow \longrightarrow \xi,\quad \xi + (y,\ \mathrm{new}\mathcal{D}(\xi)) \\
x(y) & : & \xi \\
[y] & : & \bigcup_{c \in (\mathrm{all}\mathcal{D}(\xi)\ \cup\ \mathrm{new}\mathcal{D}(\xi))} \xi + (y, c)
\end{array} \\
& & \qquad \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\delta\xi\alpha C & = & \texttt{case}\ \alpha\ \texttt{in} \\
& & \begin{array}{rcl}
\tau, \tau[x/y] & : & \tau \\
\overline{x}(y), \overline{x}y & : & \overline{\xi(x)}\xi(y) \\
x(y) & : & \xi(x) \\
[y] & : & [\xi(y)]
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

Table 5: definition of (η, δ)

bolic – labels (ranged over by $\rho, \rho_1, \ldots$) which do not include obligations anymore. As a consequence, the operational behaviour of $\pi\xi$-calculus processes can be quotiented by the usual CCS bisimulation equivalences [Par81, Mil83, HM85]. We recall those semantics, assuming, as usual, $\Longrightarrow$ to be the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\rho}{\Longrightarrow}$ to be $\Longrightarrow\xrightarrow{\rho}\Longrightarrow$, and $\overset{\widehat{\rho}}{\Longrightarrow}$ to be $\Longrightarrow$ if $\rho = \tau$, $\overset{\rho}{\Longrightarrow}$ otherwise.

Definition 5 Let $\mathcal{S}$ be a binary symmetric relation over $\pi\xi$-calculus processes. Then

- $\mathcal{S}$ is a *strong bisimulation* if $S_1\ \mathcal{S}\ S_2$ implies that

  if $S_1 \xrightarrow{\rho} S_1'$ then for some S_2', $S_2 \xrightarrow{\rho} S_2'$ and $S_1'\ \mathcal{S}\ S_2'$

- $\mathcal{S}$ is a *weak bisimulation* if $S_1\ \mathcal{S}\ S_2$ implies that if $S_1 \overset{\rho}{\longrightarrow\!\!\!\triangleright} S_1'$ then for some S_2', $S_2 \overset{\widehat{\rho}}{\Longrightarrow\!\!\!\triangleright} S_2'$ and $S_1' \,\mathcal{S}\, S_2'$

S_1 is *strong bisimilar* to S_2, written $S_1 \sim S_2$, if $S_1 \,\mathcal{S}\, S_2$ for some strong bisimulation $\mathcal{S}$.

S_1 is *weak bisimilar* to S_2, written $S_1 \approx S_2$, if $S_1 \,\mathcal{S}\, S_2$ for some weak bisimulation $\mathcal{S}$.

S_1 is *weak congruent* (or *observational congruent*) to S_2, written $S_1 \approx^c S_2$, iff $S_1 \approx S_2$ and whenever $S_1 \overset{\tau}{\longrightarrow\!\!\!\triangleright} S_1'$ then for some S_2', $S_2 \overset{\tau}{\Longrightarrow\!\!\!\triangleright} S_2'$ and $S_1' \approx S_2'$, and symmetrically. □

The finite fragment of the symbolic operational semantics of Tab. 2 (that is, all the rules but the one for the bang operator) fits in a simple generalization of the De Simone format [DS85] where labels of transitions are elements of an algebra with several operations rather than elements of a monoid of actions.

As a result of this, we can state a head-normalizing axiom system for the finite terms of the transition system in Tab. 2 by simply exploiting the procedure presented in [ABV94]. The first step of that procedure consists in breaking down, by means of auxiliary operators, each (so-called smooth and not distinctive) process constructor f whose operational behaviour is described by more than one inference rule. This equation expresses the behaviour of the constructor f as a sum of auxiliary operators, one for each inference rule characterizing f (*e.g.*, $(y)P = (\nu_y)P + (o_y)P$). The procedure goes on imposing distributive, action and inaction equations on smooth and distinctive operators. Distributive laws describe the interplay between the non-deterministic choice operator and the other operators (*e.g.*, $(\nu_y)(P + Q) = (\nu_y)P + (\nu_y)Q$). Action laws describe the interactions of all the operators with prefixing. Here, we need to extend action laws to the case of the generalized prefixing $\omega.P$. Action laws cause auxiliary operators to be pushed as deep as possible inside terms (*e.g.*, $(\nu_y)(\omega.P) = \nu_y(\omega).(y)P$). Finally, the so called inaction equations identify as the inactive process any expression having no outgoing transition (*e.g.*, $(\nu_y)\,\mathtt{nil} = \mathtt{nil}$). The axiom system $\mathcal{A}_a$ in Tab. 6 is the actual result of the application of the above procedure. The system has the main propriety of being head-normalizing, namely $\mathcal{A}_a$ allows any term P to be rewritten as a sum of prefixes.

Once its π-component has been reduced in head normal form, any finite $\pi\xi$-agent can be transformed into a finite labelled tree by means of the laws of the system $\mathcal{A}_L$ of Tab. 6. Those axioms allow $\pi\xi$-calculus processes to be rewritten into terms given by the following grammar

$$\begin{array}{rcl} S & ::= & \xi :: P \;\big|\; \rho; S \;\big|\; S \oplus S \\ \rho & ::= & \tau \;\big|\; c \;\big|\; [c] \;\big|\; c_1 c_2 \;\big|\; \overline{c_1} c_2 \end{array}$$

where P makes use of generalized action prefixes ω as explained above, and ';' is a prefixing operator, and '$\oplus$' is a non-deterministic choice constructor. We assume that the empty summation $\bigoplus_\emptyset$ stays for the $\pi\xi$-process $\xi :: \mathtt{nil}$ where ξ is intended to be some fixed environment, say the identity relation. Axioms $C1' - C4'$ of system $\mathcal{A}_L$ are the usual monoidal laws for summation where, for arbitrary instantiations of the environment ξ, process $\xi :: \mathtt{nil}$ is a neutral element of the top level summation operator. Axioms U and V are the corresponding of the two operational rules which define the transition relation $\longrightarrow\!\!\!\triangleright$. Eventually, the D law describes the distributivity

$$
\begin{array}{lll}
\mathcal{A}_a: & (C1) & P_1 + P_2 = P_2 + P_1 \\
& (C2) & (P_1 + P_2) + P_3 = P_1 + (P_2 + P_3) \\
& (C3) & P + P = P \\
& (C4) & P + \mathtt{nil} = P \\
& (EX) & P_1 \mid P_2 = P_1 \rfloor\!\rfloor P_2 + P_2 \rfloor\!\rfloor P_1 + P_1 \parallel P_2 \\
& (R) & (x)P = (\nu_x)P + (o_x)P \\
& (A1) & [x = y]\, \omega.P = \mu_y^x(\omega).P \\
& (A2) & \omega.P_1 \rfloor\!\rfloor P_2 = \omega.(P_1 \mid P_2) \\
& (A3) & \omega_1.P_1 \parallel \omega_2.P_2 = (\omega_1 \parallel \omega_2).(P_1 \mid P_2) \\
& (A4) & (\nu_x)(\omega.P) = \nu_x(\omega).(x)P \\
& (A5) & (o_x)(\omega.P) = o_x(\omega).P \\
& (D1) & [x = y](P_1 + P_2) = [x = y]P_1 + [x = y]P_2 \\
& (D2) & (P_1 + P_2) \rfloor\!\rfloor P = P_1 \rfloor\!\rfloor P + P_2 \rfloor\!\rfloor P \\
& (D3) & (P_1 + P_2) \parallel P = P_1 \parallel P + P_2 \parallel P \\
& (D4) & P \parallel (P_1 + P_2) = P \parallel P_1 + P \parallel P_2 \\
& (D5) & (\nu_x)(P_1 + P_2) = (\nu_x)P_1 + (\nu_x)P_2 \\
& (D6) & (o_x)(P_1 + P_2) = (o_x)P_1 + (o_x)P_2 \\
& (IN) & [x = y]\,\mathtt{nil} = (\nu_x)\,\mathtt{nil} = (o_x)\,\mathtt{nil} = \mathtt{nil} \rfloor\!\rfloor P = P \parallel \mathtt{nil} = \mathtt{nil} \parallel P = \mathtt{nil}
\end{array}
$$

$$
\begin{array}{lll}
\mathcal{A}_L: & (C1') & S_1 \oplus S_2 = S_2 \oplus S_1 \\
& (C2') & (S_1 \oplus S_2) \oplus S_3 = S_1 \oplus (S_2 \oplus S_3) \\
& (C3') & S \oplus S = S \\
& (C4') & S \oplus \xi :: \mathtt{nil} = S \\
& (U) & \xi :: \omega.P = \bigoplus_{\xi' \in \eta_L(\xi, \omega)} \delta(\xi', \omega); (\xi' :: P) \quad \text{if } \omega \neq \langle x(y), C \rangle \\
& (V) & \xi :: \langle x(y), C \rangle.P = \bigoplus_{\xi' \in \eta_L(\xi, \langle x(y), C \rangle)} \delta(\xi', \langle x(y), C \rangle); (\xi' :: \langle [y], \mathtt{true} \rangle.P) \\
& (D) & \xi :: (P_1 + P_2) = \xi :: P_1 \oplus \xi :: P_2
\end{array}
$$

$$
\begin{array}{lll}
\mathcal{A}_w: & (T1) & \rho; \tau; S = \rho; S \\
& (T2) & S \oplus \tau; S = \tau; S \\
& (T3) & \rho; (S_1 \oplus \tau; S_2) \oplus \rho; S_2 = \rho; (S_1 \oplus \tau; S_2)
\end{array}
$$

Table 6: axiom systems $\mathcal{A}_a$, $\mathcal{A}_L$, $\mathcal{A}_w$

of the low level choice operator over the state construct '$\_ :: \_$'. Notice that, while U and V are axiom schemata with an infinite number of possible instantiations, in every instantiation the summation $\oplus_{\xi'}$ is extended only to a finite number of summands.

One comment relative to weak semantics is due. Referring to [Hen88], Luca Aceto discussed in [Ace94] the problems given raise by the interplay between the auxiliary constructs for expanding parallel compositions and the second Milner's τ-law. Here the issue is completely avoided, as the ACP [BK84] merge operators are only used for the axiomatization $\mathcal{A}_a$ of the low level symbolic transition system, while Milner's τ-laws, reported in Tab. 6 as $\mathcal{A}_w$, are only adopted for the top level transition system.

The coming theorems deal with the characterization of ground π-calculus bisimilarities in terms of the CCS-like $\pi\xi$-calculus semantics.

Theorem 6 *Let $\xi^N :: P$ and $\xi^N :: Q$ be $\pi\xi$-calculus processes with P and Q finite and such that $N = \mathrm{fn}(P, Q)$. Then*

1. $\xi^N :: P \sim \xi^N :: Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L \vdash \xi^N :: P = \xi^N :: Q$
2. $\xi^N :: P \approx^c \xi^N :: Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L, \mathcal{A}_w \vdash \xi^N :: P = \xi^N :: Q$ $\square$

Theorem 7 *(coincidence with late ground semantics)*
Let P, Q be π-calculus processes, and let $N = \mathrm{fn}(P, Q)$. Then

1. $P \dot\sim_L Q$ *iff* $(\xi^N :: P) \sim (\xi^N :: Q)$.
   Also, if P and Q are finite, then $P \dot\sim_L Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L \vdash \xi^N :: P = \xi^N :: Q$
2. $P \dot\approx_L Q$ *iff* $(\xi^N :: P) \approx (\xi^N :: Q)$
3. $P \dot\simeq_L Q$ *iff* $(\xi^N :: P) \approx^c (\xi^N :: Q)$.
   Also, if P and Q are finite, then $P \dot\simeq_L Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L, \mathcal{A}_w \vdash \xi^N :: P = \xi^N :: Q$

$\square$

The above results can be proven by minor changes to similar characterizations already appeared in [FMQ96]. The $\pi\xi$-calculus, as presented here, differs from the semantic model introduced in [FMQ96] just for the explicit use of abstraction prefixes λy. Also, [FMQ96] only dealt with strong semantics. However the proofs of the weak clauses of both Th. 6 and Th. 7 do not present relevant difficulties w.r.t. the corresponding strong items. For instance, once the completeness of the axiom system $(\mathcal{A}_a, \mathcal{A}_L)$ has been shown for strong semantics, the proof of Th. 6 requires to pass through a saturation lemma, just as it is the case in CCS [Mil89]. An analogous reasoning holds of Th. 7. There is a one-to-one correspondence between the τ-moves of π-processes and $\pi\xi$-processes, therefore the characterization of weak semantics is not much more complex than that of strong.

The main concern of the coincidence proof is to define suitable encodings of π-calculus processes into $\pi\xi$-calculus processes and vice-versa. We can comment on this point illustrating the idea that underlies the proof of

$$(\xi^N :: P) \sim (\xi^N :: Q) \quad \text{implies} \quad P \dot\sim_L Q.$$

Given a strong bisimulation $\mathcal{S}$ containing the pair $(\xi^N :: P, \xi^N :: Q)$, it is a matter of encoding it into a late ground bisimulation, say $Tr(\mathcal{S})$, containing (P, Q). The construction is based on a translation of pairs of $\pi\xi$-calculus processes into pairs of π-agents. Starting from environments, the translation essentially generates the name substitutions to be applied to the π-component of $\pi\xi$-processes. Given any pair in $\mathcal{S}$, the key point is that the same constant in the environments of the paired processes has to be substituted by the same name. The observability of input parameter actualizations guarantees that $\mathcal{S}$ contains pairs of $\pi\xi$-calculus processes where, once executed an input action, the respective placeholders have been instantiated in the same way. This exactly captures the flavour of the late bisimulation input clause. For $(P', Q') \in Tr(\mathcal{S})$ and $P' \xrightarrow{x(y)} P''$, a π-calculus process Q'' is shown to exist such that $Q' \xrightarrow{x(y)} Q''$. Moreover, the relation $Tr(\mathcal{S})$ is proved to contain not only all the pairs $(P''\{w/y\}, Q''\{w/y\})$ but also (if any) all the pairs of the derivatives of theirs.

The encoding from $\pi\xi$-calculus to π-calculus is somehow made complex by the structural differences between the two algebras. Translating constants into names and reconstructing the name substitutions coded by environments is not enough. One has as well to remove possible occurrences of leading λ-prefixes, and to recover the possible nesting of those restrictions whose occurrence has been syntactically 'lost' because of the use of one single CCS-like communication rule vs. the π-calculus dichotomy between the Close and the usual synchronization rule. Obviously, all these issue are reversed when encoding the π-calculus into the $\pi\xi$-calculus.

Eventually, another subtle point is the characterization of the operational correspondence between the behaviours of processes which are the one encoding of the other. As it is natural, any proof of this kind can only be based on the syntactic structure of processes, and then, as far as the $\pi\xi$-calculus is concerned, on the symbolic transition system at the first level. Nevertheless, the behaviour of $\pi\xi$-calculus processes is ruled by a two level transition system where the top-level evaluation of the obligation C plays a significative role: a symbolic derivation for P may result in no move at all for the global process $\xi : P$. Hence, an actual operational correspondence between the two calculi can be faithfully captured only at the second level of the $\pi\xi$-calculus. Also, contrary to $\pi\xi$-calculus processes, in the π-calculus all of the bound names are always syntactically bound. So the π-inference of $Q \xrightarrow{\alpha} Q'$ could pass through the application of a so-called *Open* rule which has no dual in any symbolic derivation of the $\pi\xi$-calculus process corresponding to Q. Given any symbolic free output action of the $\pi\xi$-calculus, there is no way to argue whether or not it is actually free till the environment is investigated in the top level transition system.

The above issues result in quite non standard theorems about the operational correspondence of processes of the two calculi, where, from the $\pi\xi$-perspective, we are obliged to reason at the level of the symbolic $\pi\xi$-calculus transition system and to simultaneously carry around inductive information about obligations and names to be used as look-aheads for the top level.

We end up this section with a last comment on Theorem 7. Essentially, an initial ground environment ξ^N codes the information that the names in N are all distinct the one from the other. So, comparing the behaviours of $\xi^N :: P$ and $\xi^N :: Q$, with $N = \text{fn}(P, Q)$, is the same as assuming that the names occurring free in both P and Q are, without exception, all distinct. This exactly captures the ground view which does not take into account that putting processes into *arbitrary* contexts might result in causing *arbitrary* equalities of names. Hence, not surprisingly, the bisimilarity of $\xi^N :: P$ and $\xi^N :: Q$ can only correspond to ground – vs. non-ground – π-calculus bisimilarity of P and Q. We will see in the following that, to capture the flavour of π-calculus non-groundness, either more generous $\pi\xi$-processes, or a more abstract notion of environment is needed.

3 Limiting π-groundness checks

Let any of the relations in $\{\sim_L, \approx_L, \simeq_L\}$ be denoted by the symbol $\asymp$ and assume that $\dot{\asymp}$ stays for the corresponding ground relation. The equivalence $\asymp$ is classically defined by closing $\dot{\asymp}$ over any name substitution, and hence by requiring an infinite number of ground bisimilarity checks. In the following we will show that $P \asymp Q$ can be alternatively expressed in terms of a finite number of checks $P\sigma \dot{\asymp} Q\sigma$ for carefully chosen substitutions σ. In the next section, relying on particular features of the $\pi\xi$-calculus, we will be able to effectively generate the set of name substitutions to be taken into account. The finitary characterization presented below is referred to semantics of the late family, though exactly the same result does hold for early relations [Qua96].

The universal quantification over substitutions in the definition of non-ground bisimilarities gives raise to a heavy requirement to check in practice. Nevertheless, not all of the infinite name substitutions are either always or equally relevant to infer process equivalence. Resorting to infinite checks would be undoubtly useless to infer, *e.g.*, $z(y) \asymp z(u)$. More generally, given an arbitrary process P and any name $x \notin \mathrm{fn}(P)$, for all w it holds that $P\{w/x\} \equiv P$. Hence, a first simple improvement in checking $P \asymp Q$ is to consider, rather than all the possible name substitutions, only those whose domains are given by the union of the free names of P and Q.

With some ingenuity, the number of necessary substitutions may be further on diminished. The relevant issue to be investigated here is how the application of name substitutions relates to possible changes in process move potentials. First, at least in the absence of a mismatching operator, substitutions do not decrease those potentials. Indeed, if a process R may execute an action α transforming into R', then, up to α-equivalence, $R\sigma$ can perform $\alpha\sigma$ and become $R'\sigma$ [MPW92]. However, as the coming example shows, substitutions may increase performance capabilities.

Example 8 Assuming $x \neq y$, let $R_1 \equiv (\overline{x} \mid y)$ and $\sigma_x = \{x/x, x/y\}$. The parallel components of $R_1\sigma_x$ may be involved in a communication which is forbidden to the subprocesses of R_1. A τ-move is equally possible for $R_1\sigma_y$ for $\sigma_y = \{y/x, y/y\}$. More generally, given any substitution $\sigma_z = \{z/x, z/y\}$, a communication occurs between the subagents $\overline{x}\sigma_z$ and $y\sigma_z$ of $R_1\sigma_z$. □

Suppose to want to check the congruence of the two processes R_1 and R_2, with R_1 the same as in Ex. 8 and R_2 such that $\mathrm{fn}(R_1, R_2) = \{x, y\}$. We already know that comparable action capabilities are expected for the two processes $R_1\sigma_x$ and $R_1\sigma_y$. This suggests that the bisimilarity of $R_1\sigma_x$ and $R_2\sigma_x$ might be related to that of $R_1\sigma_y$ and $R_2\sigma_y$ in a very precise sense. If so, then checking both of $R_1\sigma_x \dot{\sim}_L R_2\sigma_x$ and $R_1\sigma_y \dot{\sim}_L R_2\sigma_y$ would be superfluous. In the following we will show the claim that $R_1 \sim_L R_2$ can be inferred by choosing any pair of distinct names (w, u) and resorting to only two checks:

$$R_1\{u/x, u/y\} \dot{\sim}_L R_2\{u/x, u/y\} \qquad R_1\{u/x, w/y\} \dot{\sim}_L R_2\{u/x, w/y\}$$

The comparability of the move potentials of the two processes $R_1\sigma_x$ and $R_1\sigma_y$ of Ex. 8 depends on the fact that either σ_x or σ_y map both x and y into the same target name. From this we gain the intuition that substitutions can be quotiented according to the way their domains are partitioned into subsets of names sharing the same image, no matter what such an image is.

Definition 9 The name substitution $\sigma : \mathcal{N} \to \mathcal{N}$ is said to *represent the partition of* N into the k disjoint and non-empty sets $N_1, \ldots, N_k$ iff $\forall j, h \in \{1, \ldots, k\}$: $j \neq h, \forall x, y \in N_j, \forall z \in N_h$ it holds that $x\sigma = y\sigma$ and $x\sigma \neq z\sigma$. $\square$

Given any ground behavioural equivalence $\dot{\asymp}$, our next goal is to prove that checking $P\sigma \dot{\asymp} Q\sigma$ is just the same as checking $P\sigma' \dot{\asymp} Q\sigma'$ whenever σ and σ' represent the same partition of $\mathrm{fn}(P, Q)$. As auxiliary intermediate results, we prove statements on the relationship between the strong and the weak behaviour of $P\sigma$ and $P\sigma'$ with σ and σ' representing the same partition of $\mathrm{fn}(P)$. Since the substitutions $\sigma_x = \{x/x, x/y\}$ and $\sigma_y = \{y/x, y/y\}$ considered in Ex. 8 represent the same partition of $\{x, y\} = \mathrm{fn}(R_1)$, the coming lemmata actually formalize the intuitive comparability of the action potentials of $R_1\sigma_x$ and $R_1\sigma_y$.

Lemma 10 *Let* $\sigma, \sigma' : \mathcal{N} \to \mathcal{N}$ *represent the same partition of* $\mathrm{fn}(P)$. *If* $P\sigma \xrightarrow{\alpha} P_1$ *with* $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P\sigma, P\sigma')$ *and such that* $(\mathrm{bn}(\alpha))\sigma = (\mathrm{bn}(\alpha))\sigma' = \mathrm{bn}(\alpha)$, *then by an inference of equal depth* $P\sigma' \xrightarrow{\beta} P_2$ *where, for some action* γ *and some process* P' *with* $\mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \mathrm{bn}(\alpha)$, *it holds that* $\alpha \equiv \gamma\sigma$, $\beta \equiv \gamma\sigma'$ *and* $P_1 \equiv_\alpha P'\sigma$, $P_2 \equiv_\alpha P'\sigma'$.

Proof: By induction on depth of inference. Each rule of the π-calculus transition system is considered in turn as the last rule applied. We show in the following only the more interesting cases, the others are either simpler or analogous. In particular, we suppose that the last rule applied in the inference of $P\sigma \xrightarrow{\alpha} P_1$ is either the input axiom or the communication rule.

(Inp)
$P \equiv x(y).R$. Then, for $z \notin \mathrm{fn}((y)R, R\sigma)$ and such that $z\sigma = z$, it holds that $P\sigma \equiv_\alpha x\sigma(z).R\{z/y\}\sigma$. Analogously, for $u \notin \mathrm{fn}((y)R, R\sigma')$ and such that $u\sigma' = u$, it holds that $P\sigma' \equiv_\alpha x\sigma'(u).R\{u/y\}\sigma'$. Suppose that $\alpha = x\sigma(w)$ with $w = w\sigma = w\sigma'$. Then

$$P_1 \equiv_\alpha R\{z/y\}\sigma\{w/z\} \equiv R\{z/y\}\{w/z\}\sigma \equiv R\{w/y\}\sigma$$

by $z = z\sigma$ and $w = w\sigma$. Also, by the hypothesis that $w \notin \mathrm{fn}(P\sigma')$ and by $u = u\sigma'$ and $w = w\sigma'$, it follows that

$$P\sigma' \xrightarrow{x\sigma'(w)} P_2 \equiv_\alpha R\{u/y\}\sigma'\{w/u\} \equiv R\{w/y\}\sigma'$$

Then the thesis, taking $\gamma = x(w)$ and $P' \equiv R\{w/y\}$.

(Com)
$P \equiv R_1 \mid R_2$ and $\alpha = \tau$. Then $R_1\sigma \xrightarrow{\overline{x\sigma}z\sigma} R_1'$ and $R_2\sigma \xrightarrow{y\sigma(w)} R_2'$ (or symmetrically) where we can assume w fresh w.r.t. $\mathrm{fn}(R_2\sigma, R_2\sigma')$ and such that $w = w\sigma = w\sigma'$ and where $x\sigma = y\sigma$ and $P_1 \equiv R_1' \mid R_2'\{z\sigma/w\}$

$\Longrightarrow$ By ind. hyp. $R_1\sigma' \xrightarrow{\overline{x\sigma'}z\sigma'} R_1''$ and $R_2\sigma' \xrightarrow{y\sigma'(w)} R_2''$ with

$$\begin{array}{lll} R_1' & \equiv_\alpha & T_1\sigma \\ R_1'' & \equiv_\alpha & T_1\sigma' \end{array} \qquad \begin{array}{lll} R_2' & \equiv_\alpha & T_2\sigma \\ R_2'' & \equiv_\alpha & T_2\sigma' \end{array}$$

for some T_1 and T_2 such that $\mathrm{fn}(T_1) \subseteq \mathrm{fn}(R_1)$ and $\mathrm{fn}(T_2) \subseteq \mathrm{fn}(R_2) \cup \{w\}$

$\Longrightarrow$ As σ and σ' represent the same partition of $\mathrm{fn}(P)$, they also represent the same partition of $\mathrm{fn}(R_1, R_2) \subseteq \mathrm{fn}(P)$. So, by $x\sigma = y\sigma$, it follows $x\sigma' = y\sigma'$. Then $P\sigma' \xrightarrow{\tau} P_2$ with

$$P_2 \equiv R_1'' \mid R_2''\{z\sigma'/w\} \equiv_\alpha T_1\sigma' \mid T_2\sigma'\{z\sigma'/w\} \equiv T_1\sigma' \mid T_2\{z/w\}\sigma'$$

by $w = w\sigma'$

$\Longrightarrow$ The thesis, taking $P' \equiv T_1 \mid T_2\{z/w\}$. In fact

$$P_1 \equiv R_1' \mid R_2'\{z\sigma/w\} \equiv_\alpha T_1\sigma \mid T_2\sigma\{z\sigma/w\} \equiv T_1\sigma \mid T_2\{z/w\}\sigma$$

by $w = w\sigma$. Also, $\mathrm{fn}(P') = \mathrm{fn}(T_1, T_2) \setminus \{w\} \subseteq \mathrm{fn}(R_1, R_2)$. $\square$

Lemma 11 *Let $\sigma, \sigma' : \mathcal{N} \to \mathcal{N}$ represent the same partition of $\mathrm{fn}(P)$. If $P\sigma \Longrightarrow P_1$ then by a derivation of equal length $P\sigma' \Longrightarrow P_2$ where, for some process P' with $\mathrm{fn}(P') \subseteq \mathrm{fn}(P)$, it holds that $P_1 \equiv_\alpha P'\sigma$ and $P_2 \equiv_\alpha P'\sigma'$.*

Proof: By induction on the length n ($n \geq 0$) of the derivation of P_1 from $P\sigma$.

(base) For the base case $P_1 \equiv P\sigma$, then simply take $P' \equiv P$.

(step) Assume now that $P\sigma \Longrightarrow P_1 \xrightarrow{\tau} P_1'$ with the derivation $P\sigma \Longrightarrow P_1$ of length $n > 0$.

$\Longrightarrow$ By ind. hyp. $P\sigma' \Longrightarrow P_2$ with a derivation of n steps, and for some P' it holds that $P_1 \equiv_\alpha P'\sigma$ and $P_2 \equiv_\alpha P'\sigma'$

$\Longrightarrow$ By $P_1 \xrightarrow{\tau} P_1'$ and by $P_1 \equiv_\alpha P'\sigma$ and by $\equiv_\alpha \subset \dot{\sim}_L$ it follows that $P'\sigma \xrightarrow{\tau} P_1'' \equiv_\alpha P_1'$

$\Longrightarrow$ As $\mathrm{fn}(P'\sigma) = \mathrm{fn}(P_1) \subseteq \mathrm{fn}(P\sigma)$ and $\mathrm{fn}(P'\sigma') = \mathrm{fn}(P_2) \subseteq \mathrm{fn}(P\sigma')$, the hypotheses of Lemma 10 are satisfied. By such a lemma $P'\sigma' \xrightarrow{\tau} P_2''$ where, for some P'', it holds that $P_1'' \equiv_\alpha P''\sigma$ and $P_2'' \equiv_\alpha P''\sigma'$

$\Longrightarrow$ By $P_2 \equiv_\alpha P'\sigma'$ it follows that $P_2 \xrightarrow{\tau} P_2' \equiv_\alpha P_2''$. Hence the thesis, since $P_1' \equiv_\alpha P_1'' \equiv_\alpha P''\sigma$ and $P_2' \equiv_\alpha P_2'' \equiv_\alpha P''\sigma'$. $\square$

Given two substitutions σ, σ' representing the same partition of $\mathrm{fn}(P, Q)$, the next theorem relates the ground (in)equivalence of $P\sigma$ and $Q\sigma$ to the (in)equivalence of $P\sigma'$ and $Q\sigma'$.

Theorem 12 *Let $\sigma, \sigma' : \mathcal{N} \to \mathcal{N}$ represent the same partition of $\mathrm{fn}(P, Q)$ and let $\dot{\asymp} \in \{\dot{\sim}_L, \dot{\approx}_L, \dot{\simeq}_L\}$. Then $P\sigma \dot{\asymp} Q\sigma$ iff $P\sigma' \dot{\asymp} Q\sigma'$.*

Proof: We first prove the result relative to strong semantics. To this end, let $\mathcal{S} = \bigcup_n \mathcal{S}_n$ where

$$\begin{array}{rcl} \mathcal{S}_0 & = & \dot{\sim}_L \\ \mathcal{S}_{n+1} & = & \{\, (P\sigma, Q\sigma) \;\mid\; P\sigma' \; \mathcal{S}_n \; Q\sigma' \text{ and } \\ & & \qquad \sigma, \sigma' \text{ represent the same partition of } \mathrm{fn}(P, Q) \,\} \end{array}$$

We show that $\mathcal{S}$ is a strong late ground bisimulation by proving, by induction on n, that $P \; \mathcal{S}_n \; Q$ implies that

- if $P \xrightarrow{\alpha} P'$ with $\alpha \neq x(y)$ and $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P,Q)$, then for some Q', $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{S} Q'$
- if $P \xrightarrow{x(y)} P'$ with $y \notin \mathrm{fn}(P,Q)$, then for some Q', $Q \xrightarrow{x(y)} Q'$ and, for all w, $P'\{w/y\} \mathcal{S} Q'\{w/y\}$

(BASE) By definition of $\mathcal{S}_0$.

(STEP) Assume $n > 0$ and suppose $(P\sigma, Q\sigma) \in \mathcal{S}_n$ by $(P\sigma', Q\sigma') \in \mathcal{S}_{n-1}$ and σ, σ' representing the same partition of $\mathrm{fn}(P,Q)$. We only consider bound actions, the other cases are easier.

(BOUND OUTPUT)
Suppose that $P\sigma \xrightarrow{\overline{x\sigma}(y)} P_1$ with $y \notin \mathrm{fn}(P\sigma, Q\sigma)$

$\Longrightarrow$ For some $y' \notin \mathrm{fn}(P\sigma, P\sigma', Q\sigma, Q\sigma')$ and such that $y' = y'\sigma = y'\sigma'$, it holds that $P\sigma \xrightarrow{\overline{x\sigma}(y')} P_1\{y'/y\}$

$\Longrightarrow$ By Lemma 10, $P\sigma' \xrightarrow{\overline{x\sigma'}(y')} P_2$ with $P_1\{y'/y\} \equiv_\alpha P'\sigma$ and $P_2 \equiv_\alpha P'\sigma'$ for some process P' such that $\mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \{y'\}$

$\Longrightarrow$ By ind. hyp. it follows from $(P\sigma', Q\sigma') \in \mathcal{S}_{n-1}$ that $Q\sigma' \xrightarrow{\overline{x\sigma'}(y')} Q_2$ with $(P_2, Q_2) \in \mathcal{S}$

$\Longrightarrow$ By Lemma 10, $Q\sigma \xrightarrow{\overline{x\sigma}(y')} Q_1$ with $Q_2 \equiv_\alpha Q'\sigma'$ and $Q_1 \equiv_\alpha Q'\sigma$ for some Q' such that $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q) \cup \{y'\}$.

$\Longrightarrow$ By $y \notin \mathrm{fn}(Q\sigma)$ it holds that $Q\sigma \xrightarrow{\overline{x\sigma}(y)} Q_1\{y/y'\}$. Hence the thesis $(P_1, Q_1\{y/y'\}) \in \mathcal{S}$ by

$$\begin{array}{rcl} P_1 & \equiv & P_1\{y'/y\}\{y/y'\} \\ & \equiv_\alpha & P'\sigma\{y/y'\} \\ & \mathcal{S} & Q'\sigma\{y/y'\} \\ & \equiv_\alpha & Q_1\{y/y'\} \end{array}$$

where $(P'\sigma\{y/y'\}, Q'\sigma\{y/y'\}) \in \mathcal{S}$ is justified by $(P_2, Q_2) \in \mathcal{S}$, *i.e.* $(P'\sigma', Q'\sigma') \in \mathcal{S}$. In fact, from $y \notin \mathrm{fn}(P\sigma, Q\sigma)$ and $y' \notin \mathrm{fn}(P\sigma', Q\sigma')$ it follows the freshness of y and y' w.r.t. the relevant codomains of $\sigma\{y/y'\}$ and $\sigma' = \sigma'\{y'/y'\}$. Then $\sigma\{y/y'\}$ and σ' represent the same partition of $\mathrm{fn}(P', Q') \subseteq \mathrm{fn}(P,Q) \cup \{y'\}$.

(INPUT)
Suppose that $P\sigma \xrightarrow{x\sigma(y)} P_1$ with $y \notin \mathrm{fn}(P\sigma, Q\sigma)$

$\Longrightarrow$ For some $y' \notin \mathrm{fn}(P\sigma, P\sigma', Q\sigma, Q\sigma')$ and such that $y' = y'\sigma = y'\sigma'$, it holds that $P\sigma \xrightarrow{x\sigma(y')} P_1\{y'/y\}$

$\Longrightarrow$ By Lemma 10, $P\sigma' \xrightarrow{x\sigma'(y')} P_2$ with $P_1\{y'/y\} \equiv_\alpha P'\sigma$ and $P_2 \equiv_\alpha P'\sigma'$ for some process P' with $\mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \{y'\}$

$\Longrightarrow$ By ind. hyp., it follows from $(P\sigma', Q\sigma') \in \mathcal{S}_{n-1}$ that $Q\sigma' \xrightarrow{x\sigma'(y')} Q_2$ with $(P_2\{u/y'\}, Q_2\{u/y'\}) \in \mathcal{S}$ for all u

$\Longrightarrow$ By Lemma 10, $Q\sigma \xrightarrow{x\sigma(y')} Q_1$ with $Q_2 \equiv_\alpha Q'\sigma'$ and $Q_1 \equiv_\alpha Q'\sigma$ for some Q' such that $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q) \cup \{y'\}$. Then $(P'\sigma'\{u/y'\}, Q'\sigma'\{u/y'\}) \in \mathcal{S}$ for all u

$\Longrightarrow$ By $y \notin \mathrm{fn}(Q\sigma)$ it holds that $Q\sigma \xrightarrow{x\sigma(y)} Q_1\{y/y'\}$. Then the thesis, since $(P_1\{w/y\}, Q_1\{y/y'\}\{w/y\}) \in \mathcal{S}$ for all w. In fact

$$\begin{array}{rcl} P_1\{w/y\} & \equiv & P_1\{y'/y\}\{y/y'\}\{w/y\} \\ & \equiv_\alpha & P'\sigma\{w/y'\} \\ & \mathcal{S} & Q'\sigma\{w/y'\} \\ & \equiv_\alpha & Q_1\{w/y'\} \\ & \equiv & Q_1\{y/y'\}\{w/y\} \end{array}$$

where $(P'\sigma\{w/y'\}, Q'\sigma\{w/y'\}) \in \mathcal{S}$ is justified by $(P'\sigma'\{u/y'\}, Q'\sigma'\{u/y'\}) \in \mathcal{S}$ with $\sigma\{w/y'\}$ and $\sigma'\{u/y'\}$ representing the same partition of $\mathrm{fn}(P', Q') \subseteq \mathrm{fn}(P, Q) \cup \{y'\}$.

An analogous proof schema is used to handle τ-forgetting relations. Indeed, suppose that either $P\sigma' \dot{\approx}_L Q\sigma'$ or $P\sigma' \dot{\simeq}_L Q\sigma'$. By hypothesis, a weak late ground bisimulation $\mathcal{U}$ there exists such that $(P\sigma', Q\sigma') \in \mathcal{U}$ and, when ground equality is assumed, if $P\sigma' \xrightarrow{\tau} P'$ then for some Q', $Q\sigma' \stackrel{\tau}{\Longrightarrow} Q'$ with $(P', Q') \in \mathcal{U}$, and symmetrically. Then $\mathcal{T} = \bigcup_n \mathcal{T}_n$ where

$$\begin{array}{rcl} \mathcal{T}_0 & = & \mathcal{U} \\ \mathcal{T}_{n+1} & = & \{\ (P\sigma, Q\sigma) \mid P\sigma'\ \mathcal{T}_n\ Q\sigma' \text{ and} \\ & & \qquad \sigma, \sigma' \text{ represent the same partition of } \mathrm{fn}(P, Q)\ \} \end{array}$$

is shown to be a weak late ground bisimulation by proving, by induction on n, that $P\ \mathcal{T}_n\ Q$ implies

- if $P \xrightarrow{\alpha} P'$ with $\alpha \neq x(y)$ and $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ and $P'\ \mathcal{T}\ Q'$
- if $P \xrightarrow{x(y)} P'$ with $y \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \Longrightarrow\xrightarrow{x(y)} Q'$ and, for all w, $P'\{w/y\}\ \mathcal{T}\ Q'\{w/y\}$

Also, under the assumption $P\sigma' \dot{\simeq}_L Q\sigma'$, the relation $\mathcal{T}$ is proved to be such that whenever $P\sigma \xrightarrow{\tau} P''$ then for some Q'', $Q\sigma \stackrel{\tau}{\Longrightarrow} Q''$ with $(P'', Q'') \in \mathcal{T}$, and symmetrically.

W.r.t. the proof for strong bisimulation, the only remarkable issue in the proofs for τ-forgetting semantics is the possibly interleaved application of Lemma 10 and Lemma 11 for dealing with weak behaviours. $\square$

The above theorem guarantees that lots of checks may be saved when trying to infer the non-ground bisimilarity of two processes P and Q. In fact, once a certain $P\sigma \dot{\asymp} Q\sigma$ has been proved, any other test on the ground equivalence of $P\sigma'$ and $Q\sigma'$ is useless whenever σ and σ' happen to represent the same partition of $\mathrm{fn}(P, Q)$.

Considering again the introductory Ex. 8, assume to want to test whether or not $R_1 \equiv (\overline{x} \mid y) \sim_L R_2$ with $\mathrm{fn}(R_2) = \{x, y\}$. In such a case checking just the ground late bisimilarity of $R_1\{x/x, x/y\}$ and $R_2\{x/x, x/y\}$ gives full information about the late ground equivalence of any pair of processes $R_1\sigma$ and $R_2\sigma$ with $\sigma = \{z/x, z/y, w_1/u_1, \ldots, w_n/u_n\}$.

We will show in the following that late congruences of the two processes P and Q may be expressed in terms of a finite number of checks on the corresponding ground bisimilarity of $P\sigma$ and $Q\sigma$. To do that, we first introduce a definition which allows substitutions to be grouped into families representing a given set of names.

Definition 13 Let $N \subseteq \mathcal{N}$ be a set of names and $\{\sigma_i\}_{i \in I}$ be a family of name substitutions $\sigma_i : \mathcal{N} \to \mathcal{N}$. Then $\{\sigma_i\}_{i \in I}$ is a *partition family of* N iff the following holds:

- if $N = \emptyset$ then $\{\sigma_i\}_{i \in I}$ contains only the identity substitution
- for each partition of $N \neq \emptyset$ into k disjoint and non-empty sets $N_1, \ldots, N_k$, there is exactly one substitution in $\{\sigma_i\}_{i \in I}$ that represents $N_1, \ldots, N_k$ $\Box$

Notice that an infinite number of distinct partition families of $N \neq \emptyset$ there exists. However, any partition family of N is somehow unredundant: it contains one and only one representative of each of the possible partition of N.

Relying on the notion of partition family, we can eventually prove the general result which justifies our initial claim that, whenever $\mathrm{fn}(R_1, R_2) = \{x, y\}$ and $u \neq w$, the strong late congruence of R_1 and R_2 can be characterized as conjunction of the following ground bisimilarities:

$$R_1\{u/x, u/y\} \dot{\sim}_L R_2\{u/x, u/y\} \qquad R_1\{u/x, w/y\} \dot{\sim}_L R_2\{u/x, w/y\}$$

Theorem 14 *Let P, Q be π-calculus processes and let $\dot{\asymp} \in \{\dot{\sim}_L, \dot{\approx}_L, \dot{\simeq}_L\}$. Also, assume $\asymp$ to be the non-ground relation corresponding to the chosen $\dot{\asymp}$. Then $P \asymp Q$ iff $P\sigma \dot{\asymp} Q\sigma$ for all $\sigma \in \{\sigma_i\}_{i \in I}$ with $\{\sigma_i\}_{i \in I}$ partition family of* $\mathrm{fn}(P, Q)$.

PROOF:

$\Longrightarrow$) By definition of $\asymp$.

$\Longleftarrow$) Assume that a partition family $\{\sigma_i\}_{i \in I}$ of $\mathrm{fn}(P, Q)$ there exists such that $P\sigma \dot{\asymp} Q\sigma$ for all $\sigma \in \{\sigma_i\}_{i \in I}$.

Any name substitution σ' represents one out of the possible partitions of $\mathrm{fn}(P, Q)$. (For instance, if $\mathrm{Dm}(\sigma') \cap \mathrm{fn}(P, Q) = \emptyset$, then σ' behaves over $\mathrm{fn}(P, Q)$ just like the identity substitution and hence σ' represents the partition of $\mathrm{fn}(P, Q)$ into singleton sets.)

By definition of partition family, the substitution σ' represents the same partition of $\mathrm{fn}(P, Q)$ as some $\sigma \in \{\sigma_i\}_{i \in I}$. Hence the thesis, by Th. 12. $\Box$

As finite set, $\mathrm{fn}(P, Q)$ only has a finite number of distinct partitions. Then any partition family of $\mathrm{fn}(P, Q)$ is finite. So, an immediate corollary of Theorem 14 is that two π-calculus processes can be shown to be congruent by relying on a (big but) finite number of ground bisimilarity checks.

4 Processes as functions

In the following we will focus on the opportunity of relating the results of Section 3 to the peculiar features of the ground $\pi\xi$-calculus. Letting $\asymp \in \{\sim_L, \approx_L, \simeq_L\}$, this will show up in the characterization of $P \asymp Q$ as corresponding CCS-bisimilarity of two single $\pi\xi$-processes.

Theorem 7 in Section 2 shows that name substitutions are naturally encoded by environments. Precisely, it proves that a suitable management of environments can take the place of the π-calculus meta-syntactic operation of substitution. So, letting $N = \mathrm{fn}(P, Q)$, we expect for instance the double implication

$$(\xi^{N\sigma} :: P\sigma) \sim (\xi^{N\sigma} :: Q\sigma) \;\text{ iff }\; (\xi^{N\sigma} :: P) \sim (\xi^{N\sigma} :: Q)$$

to hold. Actually, a more abstract property can be proved. Applying the substitution σ to P and to Q is definitely unnecessary in the $\pi\xi$-calculus. Besides that, the mere application of σ to N can be rendered by making explicit the way how σ quotients its domain into subsets sharing the same image.

Lemma 15 *Let P_1, P_2 be π-calculus processes with $N = \mathrm{fn}(P_1, P_2)$, and, given a name substitution $\sigma : \mathcal{N}_I \to \mathcal{N}_I$, assume $\xi^{N/\sigma} = \mathrm{Id}_{\mathcal{E}} + \{(x, \imath(x\sigma)) \mid x \in N\}$. Also, let $\doteq \in \{\sim, \approx, \approx^c\}$. Then $(\xi^{N\sigma} :: P_1\sigma) \doteq (\xi^{N\sigma} :: P_2\sigma)$ iff $(\xi^{N/\sigma} :: P_1) \doteq (\xi^{N/\sigma} :: P_2)$.*

PROOF: We start by commenting on the shape of the environments ξ (ξ', respectively) which are reachable along the derivations from a given $\xi^{N\sigma} :: P\sigma$ ($\xi^{N/\sigma} :: P$, respectively).

By the definitions of $\xi^{N\sigma}$ and $\xi^{N/\sigma}$ and η, the only differences between ξ and ξ' are relative to the equivalence classes containing elements of $B = \{\imath(x) \mid x \in N\sigma\}$. Precisely, for all $c \in B$ it holds that $[c]_\xi \cap \mathcal{N}_I = \{\imath^{-1}(c)\}$ while $[c]_{\xi'}$ may be either such that $[c]_{\xi'} \cap \mathcal{N}_I \neq \{\imath^{-1}(c)\}$ or such that $\mathrm{card}([c]_{\xi'} \cap \mathcal{N}_I) > 1$ (for example think of the name substitutions $\{y/z\}$ and $\{y/z, y/w\}$ which give raise to $[\imath(y)]_{\xi'} \cap \mathcal{N}_I = \{z\}$ and to $[\imath(y)]_{\xi'} \cap \mathcal{N}_I = \{z, w\}$, resp.). The above observation is all we need to construct a relation containing $(\xi^{N\sigma} :: P_1\sigma, \xi^{N\sigma} :: P_2\sigma)$ starting from a relation containing $(\xi^{N/\sigma} :: P_1, \xi^{N/\sigma} :: P_2)$, and vice-versa.

Assume that $(\xi^{N/\sigma} :: P_1) \doteq (\xi^{N/\sigma} :: P_2)$. Then there exists a relation $\mathcal{S}$, subset of either $\sim$ or $\approx$, which contains the pair $(\xi^{N/\sigma} :: P_1, \xi^{N/\sigma} :: P_2)$ and possibly some other relevant pairs when the two processes are supposed to be observational congruent. We define in the following a relation $\mathcal{S}'$ that proves $(\xi^{N\sigma} :: P_1\sigma) \doteq (\xi^{N\sigma} :: P_2\sigma)$.

$$\mathcal{S}' = \bigcup_{(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}} (\xi'_P :: P\sigma, \xi'_Q :: Q\sigma)$$

where ξ'_P is defined as follows, and ξ'_Q is constructed in an analogous way.

$$\xi'_P = \widetilde{\xi_P} + \bigcup_{c \in B} \{(c, \imath^{-1}(c))\} \cup \{(c, x) \mid x \in [c]_{\xi_P} \setminus N\}$$

with $\widetilde{\xi_P}$ such that the following holds

- if $a\, \xi_P\, b$ and, for all $c \in B$, $a \notin [c]_{\xi_P}$, then $a\, \widetilde{\xi_P}\, b$
- if $a\, \xi_P\, b$ and there exists $c \in B$ such that $a \in [c]_{\xi_P}$, then $a\, \widetilde{\xi_P}\, a$ and $b\, \widetilde{\xi_P}\, b$

Intuitively, $\widetilde{\xi_P}$ is the same as ξ_P but for 'removing' the equivalence classes $[c]$ such that $c \in B$, namely, except for replacing any of such $[c]$ by the union of the equivalence classes $[a] = \{a\}$ with $a \in [c]$.

Now assume $(\xi^{N\sigma} :: P_1\sigma) \doteq (\xi^{N\sigma} :: P_2\sigma)$ and let $\mathcal{T}$, contained in either $\sim$ or $\approx$, be the relation which witnesses the hypothesis. In this case the thesis can be shown by transforming $\mathcal{T}$ into $\mathcal{T}'$ as follows.

$$\mathcal{T}' = \bigcup_{(\xi_P :: P, \xi_Q :: Q) \in \mathcal{T}} \bigcup_{P', Q'} (\xi''_P :: P', \xi''_Q :: Q')$$

where

$$\xi''_P = \widetilde{\xi_P} + \bigcup_{c \in B} \{(c, x) \mid \imath(x\sigma) = c\} \cup \{(c, x) \mid x \in [c]_{\xi_P} \setminus N\sigma\}$$

with $\widetilde{\xi_P}$ defined as above, and ξ''_Q defined analogously to ξ''_P. As far as P' and Q' are concerned, if σ is injective, then P' and Q' are simply $P\sigma^{-1}$ and $Q\sigma^{-1}$, but this is not the most general case. So, P' (resp. Q') is taken from the family of processes which are obtained by non-deterministically substituting in P (resp. in Q) any occurrence of the name y by any name x such that $x\sigma = y$. *E.g.*, letting $P = \overline{y}y$ and $\sigma = \{y/x, y/z\}$, such a family should be $\{\overline{x}x, \overline{x}y, \overline{x}z, \overline{y}x, \overline{y}y, \overline{y}z, \overline{z}x, \overline{z}y, \overline{z}z\}$. $\square$

The above result strongly depends on the fact that environments naturally represent, via equivalence classes, partitions of sets of names.

Definition 16 The environment ξ *represents the partition* of $N \subseteq \mathcal{N}_I$ into the disjoint and non-empty sets $N_1, \ldots, N_k$ iff the following holds:

- $N = \bigcup_{c \in \text{all}\mathcal{D}(\xi)} ([c]_\xi \cap \mathcal{N})$
- for all $j \in \{1, \ldots, k\}$ there exists $c \in \text{all}\mathcal{D}(\xi)$ such that $N_j = \big([c]_\xi \cap \mathcal{N}\big)$ $\square$

Notice that the partition of N represented by a given environment ξ is identified resorting to the constants in $\text{all}\mathcal{D}(\xi)$. Nevertheless, the actual distinguishing feature of any environment is just the represented partition, rather than the identity of its active constants. This is claimed by the next proposition.

Proposition 17 *Let P_1 and P_2 be π-calculus processes, and let the environments ξ and ξ' represent the same partition of $N \subseteq \mathcal{N}_I$. Also, let $\doteq \in \{\sim, \approx, \approx^c\}$. Then $\xi :: P_1 \doteq \xi :: P_2$ iff $\xi' :: P_1 \doteq \xi' :: P_2$.*

Proof: Let $\mathcal{S}$ be a relation over $\pi\xi$-calculus processes which proves that $\xi :: P_1 \doteq \xi :: P_2$ (resp. $\xi' :: P_1 \doteq \xi' :: P_2$). Then a relation $\mathcal{S}'$ which shows that $\xi' :: P_1 \doteq \xi' :: P_2$ (resp. $\xi :: P_1 \doteq \xi :: P_2$) may be defined by adequately translating the active constants of the environments of any pair $(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}$. $\square$

It was shown above that $\pi\xi$-calculus environments represent partitions of sets of names in a genuine way. A more effective feature can now be focussed on. In the $\pi\xi$-calculus, partitions of sets can be effectively generated by sequential compositions of λ-prefixes.

Example 18 Consider for instance the tree depicted in Fig. 1. It represents the transition system associated with the $\pi\xi$-process

$$S_0 \equiv (\text{Id}_\mathcal{E} :: \lambda x.\lambda y.\lambda z.\,\texttt{nil})$$

But for the root node, the environment of any other agent in Fig. 1 is given a compact visualization that only reports on the relevant equivalence classes. We now comment on the computations starting with S_0. No constant is active in $\text{Id}_\mathcal{E}$. Then, when the leading prefix λx fires, the name x is deterministically associated with $c_1 = \text{new}\mathcal{D}(\text{Id}_\mathcal{E})$. Correspondingly, S_0 becomes

$$S_1 \equiv (\text{Id}_\mathcal{E} + (x, c_1) :: \lambda y.\lambda z.\,\texttt{nil})$$

with $(\text{Id}_\mathcal{E} + (x, c_1))$ representing the single partition of $\{x\}$ into $\{x\}$. Differently from S_0, the environment of process S_1 contains one active constant. So, the symbolic step labelled by $\langle [y], \texttt{true} \rangle$ induces two distinct transitions of S_1. The one is labelled by $[c_1]$, the other is labelled by the concretion of a new constant c_2. The two

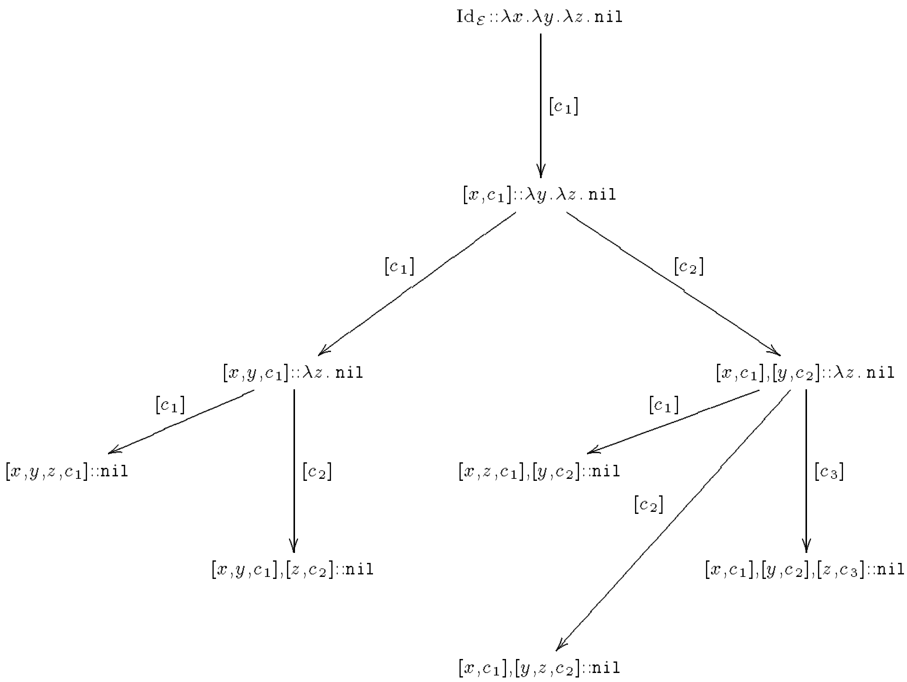

Figure 1: transition system of $\mathrm{Id}_\mathcal{E} :: \lambda x.\lambda y.\lambda z.\,\mathtt{nil}$

derivatives of S_1 have a common right component ($\lambda z.\,\mathtt{nil}$), but their environments are distinct because of the association of the name y with the constant c_1 and with the constant c_2, respectively. Such environments represent the partitioning of the set $\{x, y\}$ into $\{x, y\}$ and into $\{x\} \cup \{y\}$, respectively. Computations go on by the firing of the λz prefix and actually the environments of the leaf-processes represent all the possible partitions of $\{x, y, z\}$ into disjoint and non-empty sets. □

The above comment on Fig. 1 can be formalized as follows.

Proposition 19 *Suppose that $\xi :: \lambda y.P \xrightarrow{\rho} \xi' :: P$, then the following holds.*

1. *If $\xi = \mathrm{Id}_\mathcal{E}$ then $\rho = [\mathrm{new}\mathcal{D}(\mathrm{Id}_\mathcal{E})]$ and ξ' represents the only possible partition of $\{y\}$*
2. *If ξ represents the partition of N into the disjoint and non-empty sets $N_1, \ldots, N_k$, with $y \notin N$, then*
   - *if $\rho = [c]$ with $([c]_\xi \cap \mathcal{N}) = N_j$, then ξ' represents the partition of $N \cup \{y\}$ into the k disjoint sets $N_1, \ldots, N_j \cup \{y\}, \ldots, N_k$*

- *if* $\rho = [\mathrm{new}\mathcal{D}(\xi)]$ *then* ξ' *represents the partition of* $N \cup \{y\}$ *into the* $k+1$ *disjoint sets* $N_1, \ldots, N_k, \{y\}$

PROOF: By definition of η, it holds that $\xi' = \xi + (y, c_j)$ with $\rho = [c_j]$. Then the thesis, by Def. 16. $\square$

In the following, the partition generation property of λ-prefixes will be used to describe partition families. This passes through the definition of λ-closures.

Definition 20 Let P be a π-calculus process and $\mathcal{L}$ be a list of names. Then the λ-*closure of* P *w.r.t.* $\mathcal{L}$ is defined as $\lambda\_\mathrm{clos}(P, \mathcal{L})$ where

$$\lambda\_\mathrm{clos}(P, [\,]) = P$$
$$\lambda\_\mathrm{clos}(P, [y, \mathcal{L}]) = \lambda\_\mathrm{clos}(\lambda y.P, \mathcal{L})$$

with $[\,]$ denoting the empty list, and $[\_, [\_]]$ denoting the 'cons' list operator. $\square$

The bisimilarity of two agents obtained by a suitable λ-closure may now be used to express the bisimilarity of a whole substitution-indexed family of pairs of $\pi\xi$-processes.

Lemma 21 *Let* P_1, P_2 *be* π*-calculus processes with* $N = \mathrm{fn}(P_1, P_2)$ *and let* $\mathcal{L}$ *be a list containing all and only the elements of* N. *Also, assume* $\doteq \in \{\sim, \approx, \approx^c\}$ *and, given a name substitution* σ, *let* $\xi^{N/\sigma} = \mathrm{Id}_{\mathcal{E}} + \{(x, \imath(x\sigma)) \mid x \in N\}$. *Then* $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}) \doteq \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L})$ *iff* $(\xi^{N/\sigma} :: P_1) \doteq (\xi^{N/\sigma} :: P_2)$ *for all* $\sigma \in \{\sigma_i\}_{i\in I}$ *with* $\{\sigma_i\}_{i\in I}$ *partition family of* N.

PROOF: If $N = \emptyset$ then the thesis is immediate by $\lambda\_\mathrm{clos}(P_i, \mathcal{L}) \equiv P_i$ for $i \in \{1, 2\}$ and by $\xi^{N/\sigma} = \mathrm{Id}_{\mathcal{E}}$. Hence suppose $N \neq \emptyset$. Using Prop. 19, it can be proved by induction that

$$\bigcup\{\xi_p \mid \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_i, \mathcal{L}) \longrightarrow\!\!\triangleright^* \xi_p :: P_i\}$$

represents a partition family of N, namely each environment ξ_p represents a distinct partition of N into the disjoint and non-empty sets $N_1, \ldots, N_k$. Also notice that no internal non-determinism raises at any step during the computation $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_i, \mathcal{L}) \longrightarrow\!\!\triangleright^* \xi_p :: P_i$.
Hence $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}) \doteq \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L})$ iff $\xi_p :: P_1 \doteq \xi_p :: P_2$ for all ξ_p representing the partition p of N. Actually, any of the above environments ξ_p has the following shape

$$\xi_p = \mathrm{Id}_{\mathcal{E}} + \{(x, \xi_p(x)) \mid x \in N\}$$

where $\mathrm{all}\mathcal{D}(\xi_p)$ is given by the first k run-time generated constants rather than by constants in $\mathcal{D}_I$. Then each ξ_p differs from $\xi^{N/\sigma}$ only for the identities of the active constants. Hence the thesis by Prop. 17. $\square$

Any non-ground bisimilarity of the π-calculus processes P and Q , say $P \asymp Q$, was already proved (*cf.* Th. 14) to be expressible in terms of a finite number of suitable checks on $P\sigma \doteqdot Q\sigma$. We now show that $P \asymp Q$ can equally be characterized as the corresponding CCS-like bisimilarity of two single $\pi\xi$-processes whose right components are the λ-closures of P and Q.

Theorem 22 *(coincidence with late non-ground semantics)*
Let P, Q be π-calculus processes and let $\mathcal{L}$ be a list containing all and only the elements of $\mathrm{fn}(P,Q)$*. Then*

1. $P \sim_L Q$ *iff* $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) \sim \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$
2. $P \approx_L Q$ *iff* $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) \approx \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$
3. $P \simeq_L Q$ *iff* $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) \approx^c \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$

Proof: Assume $\dot{\asymp}$ to be the ground dual of $\asymp \in \{\sim_L, \approx_L, \simeq_L\}$ and the π-calculus dual of $\doteq \in \{\sim, \approx, \approx^c\}$. Then, letting $N = \mathrm{fn}(P,Q)$, the following holds:
$P \asymp Q$
iff (Th. 14) $P\sigma \,\dot{\asymp}\, Q\sigma$ for all $\sigma \in \{\sigma_i\}_i$ with $\{\sigma_i\}_i$ partition family of N
iff (Th. 7) $(\xi^{N\sigma} :: P\sigma) \doteq (\xi^{N\sigma} :: Q\sigma)$ for all $\sigma \in \{\sigma_i\}_i$ with $\{\sigma_i\}_i$ partition family of N
iff (Lemma 15) $(\xi^{N/\sigma} :: P) \doteq (\xi^{N/\sigma} :: Q)$ for all $\sigma \in \{\sigma_i\}_i$ with $\{\sigma_i\}_i$ partition family of N and $\xi^{N/\sigma} = \mathrm{Id}_{\mathcal{E}} + \{(x, \imath(x\sigma)) \mid x \in N\}$
iff (Lemma 21) $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) \doteq \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$. □

Theorem 23 *(equational characterizations of late non-ground semantics)*
Let P, Q be finite π-calculus processes and let $\mathcal{L}$ be a list containing all and only the elements of $\mathrm{fn}(P,Q)$*. Then*

1. $P \sim_L Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L \vdash \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) = \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$
2. $P \simeq_L Q$ *iff* $\mathcal{A}_a, \mathcal{A}_L, \mathcal{A}_w \vdash \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}) = \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})$

Proof: By Th. 22 and Th. 6. Only recall that equational characterizations of late $\pi\xi$-calculus semantics does hold for processes of the $\pi\xi$-calculus, hence for processes whose right component may contain arbitrary occurrences of λ-prefixes. □

Suppose that $\mathrm{card}(\mathrm{fn}(P,Q)) = n$. Then there are $n!$ distinct permutations of the elements in $\mathrm{fn}(P,Q)$ and consequently $n!$ distinct lists built up by $\mathrm{fn}(P,Q)$. Theorem 22 asserts that the bisimilarity of P and Q can be checked relying just on one of those lists, no matter which one. The role of $\mathcal{L}$ in $\lambda\_\mathrm{clos}(P,\mathcal{L})$ is only that of generating environments that represent a partition family of the due names. So, the actual sequencing of the list can only influence the generation order, but the desired final effect is independent on it. Roughly, the choice of the list $\mathcal{L}$ stays to $\lambda\_\mathrm{clos}(P,\mathcal{L})$ the same way as the choice of a correct sequential algorithm for computing the partitions of a finite set stays to the computed function.

When processes P and Q are to be compared, what does really matter is to use the same list $\mathcal{L}$ in defining either the λ-closure of P or the λ-closure of Q. This guarantees that the bisimilarity of the pair of processes

$$\big(\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P,\mathcal{L}), \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(Q,\mathcal{L})\big)$$

is factorized into the bisimilarity of all the pairs in the set given by

$$\textstyle\bigcup_p (\xi_p :: P, \xi_p :: Q)$$

where p ranges over the possible partitions of $\mathrm{fn}(P,Q)$ and ξ_p represents just the partition p.

The results stated in Theorem 22 are far from unexpected. Putting a given process in any context may result in binding some or all of its free names. Therefore, reasoning about congruence compells to think of any process as of a function of its free names. Not randomly, any non-ground relation $\asymp$ coincides with its ground counterpart $\dot{\asymp}$ over terms without any free name. Also, the following might be proved. If $\mathrm{fn}(P,Q)=\{y_1,\dots,y_n\}$ then checking $P \asymp Q$ is the same as picking up a fresh name, say x, and checking

$$x(y_1).x(y_2).\dots.x(y_n).P \quad \dot{\asymp} \quad x(y_1).x(y_2).\dots.x(y_n).Q$$

Given the process $x(y_1).x(y_2).\dots.x(y_n).P$ which represents a 'prefix-closure' of P, the late paradigm directly suggests that $x(y_2).\dots.x(y_n).P$ is a function of y_1, and that the agent $x(y_3).\dots.x(y_n).P$ is a function of both y_1 and y_2, ..., and P is a function of all of $\{y_1,\dots,y_n\}$. The λ-closures used in the $\pi\xi$-calculus and the above prefix-closures express exactly the same intrinsic functional dependence of processes on their free names.

A final comment can be related to the observation above. The result in Theorem 22 could have been proved by using an alternative strategy whose pattern can be sketched out as follows.

$$\begin{array}{ll} & P \asymp Q \\ \text{iff} & x(y_1).x(y_2).\dots.x(y_n).P \;\dot{\asymp}\; x(y_1).x(y_2).\dots.x(y_n).Q \\ \text{iff} & \xi^{\{x\}} :: x(y_1).x(y_2).\dots.x(y_n).P \;\doteq\; \xi^{\{x\}} :: x(y_1).x(y_2).\dots.x(y_n).Q \\ \text{iff} & \mathrm{Id}_{\mathcal{E}} :: \lambda y_1.\lambda y_2.\dots.\lambda y_n.P \;\doteq\; \mathrm{Id}_{\mathcal{E}} :: \lambda y_1.\lambda y_2.\dots.\lambda y_n.Q \end{array}$$

The approach we used exploits more transparently the correspondence between π-calculus substitutions and $\pi\xi$-environments.

5 Call-by-need generation strategies

In this section the non-ground version of the $\pi\xi$-calculus is defined. Relying on that, a more promising characterization of strong late π-calculus congruence is presented. Also, a coincidence result for open semantics is stated.

We commented that reasoning about non-groundness imposes to regard any π-calculus process as a function depending on its free names. So, as $\pi\xi$-calculus is concerned, more sophisticated environments are adopted where free names are associated with constants not by default, but rather following a call-by-need discipline. This corresponds to adopting a call-by-need strategy for the generation of the substitutions needed to reason about non-groundness.

It was shown that late non-ground bisimilarities of P and Q can be expressed, for σ ranging over a partition family of $\mathrm{fn}(P,Q)$, as conjunctions of the corresponding ground equivalences of $P\sigma$ and $Q\sigma$. We also pointed out that any partition family of $\mathrm{fn}(P,Q)$ has finite cardinality. So, the alternative characterizations stated in Section 3 are effective improvements on the usual definitions of non-ground bisimilarities. Nevertheless, the finite number of checks they require is quite big: as many checks are needed as the cardinality of a partition family of $\mathrm{fn}(P,Q)$, which grows more than 2^n with $n=\mathrm{card}(\mathrm{fn}(P,Q))$. Depending on n, the pre-processing phase consisting either in generating a partition family of $\mathrm{fn}(P,Q)$, or in taking the ground $\pi\xi$-calculus view and hence firing the λ-abstractions prefixing P and Q, may be really prohibitive. Worst than that, such a pre-processing phase could also be useless. Think, for instance, to have to check the congruence of the two processes

$$\begin{array}{rcl} P & \equiv & x(y).\mathtt{nil} \\ Q & \equiv & x(y).(z)\overline{z}x.Q_1 \end{array}$$

Since $(z)\overline{z}x.Q_1$ is deadlocked, there is no sensible reason for generating lots of substitutions whose number essentially depends on the cardinality of $\mathrm{fn}(Q_1)$. Free names could more properly and more efficiently be dealt with as variables and could be instantiated following a by-need discipline. Consider for instance the following example.

Example 24 Think of the π-calculus process $(\overline{x} \mid y)$ as of the agent

$$P_v \quad \equiv \quad \overline{v}_x : \{x\} \mid v_y : \{x, y\}$$

The syntax used for describing P_v is meant to suggest that x and y are dealt with as variables whose types are non-disjoint sets of names. Roughly, P_v gives the intuition that, whenever $(\overline{x} \mid y)$ is put in an arbitrary context, x and y may be substituted either by distinct names or by the same name, which we conveyed to be x (x results as intersection of the types of v_x and v_y). We can now discuss the action potentials of P_v. Process P_v can surely interleave the execution of the actions $\overline{v}_x$ and v_y. The more, with the proviso that v_x and v_y assume the same value, P_v can also perform a τ-step. $\square$

The intuition which underlies Ex. 24 will be rendered formal in the $\pi\xi$-calculus by adequately sophisticating the definitions of environment and of evaluation function. In particular, we will retain either the symbolic transition system of Section 2, or the idea of expressing extensional semantics in terms of ordinary strong bisimilarity. Though, the top-level transition relation and the update and result functions will be refined following a call-by-need name instantiation strategy.

In the present perspective, environments become sets of equations over three distinct entities: names, constants, and variables. Names and constants are as described in Section 2. Variables (ranged over by $v, v_1, \ldots$) are typed, each type being a finite subset of $\mathcal{D}$. More specifically, variables are taken from a domain $\mathcal{V}$ which is supposed to contain one variable per type and to be disjoint by both $\mathcal{N}$ and $\mathcal{D}$. Notationally $v : D$ indicates that the variable v takes values in the finite set $D \subset \mathcal{D}$, and an association of the form $(x\ \xi\ (v : D))$ means that x may take as value any of the constants in D.

Definition 25 A *non-ground environment* ξ is an equivalence relation over $\mathcal{N} \cup \mathcal{D} \cup \mathcal{V}$ which is:

- *consistent*: $c_i\ \xi\ c_j$ implies $c_i = c_j$ and $c\ \xi\ (v : D)$ implies $c \in D$ and $(v_1 : D_1)\ \xi\ (v_2 : D_2)$ implies $D_1 \cap D_2 \neq \emptyset$
- *finitely active*: the set $\{(a, b) \mid a\ \xi\ b \text{ and } a \neq b\}$ is finite

A variable v is *active* in ξ iff there exists $a \neq v$ with $(v\ \xi\ a)$. A constant c is *active* in ξ iff there exists either $a \neq c$ with $(c\ \xi\ a)$ or a variable $v : D$ which is active in ξ and $c \in D$. Variables and constants which are not active, are called *inactive*.

Still, the equivalence class of ξ containing a is indicated as $[a]_\xi$, and the family of all non-ground environments is denoted $\mathcal{E}$, and we write $\mathrm{Id}_{\mathcal{E}}$ for the identity environment. □

Because of the grown complexity of the environments we now deal with, we refine the notions of definedness and undefinedness of the partial function $\xi(\_\_)$. We say that ξ is defined on y (still denoted by $\xi(y)\downarrow$) iff for some $a \in \mathcal{D} \cup \mathcal{V}$ it holds that $(y\ \xi\ a)$. If $\xi(\_\_)$ is not defined on y, then we write $\xi(y)\uparrow$. Also, the class $[y]_\xi$ is said to be *undefined* if $\xi(y)\uparrow$, *defined* otherwise. When $[y]_\xi$ is defined, we call it *defined-by-variable* if $\mathcal{D} \cap [y]_\xi = \emptyset$, *defined-by-constant* otherwise.

Describing initial non-ground environments needs to generate fresh variables. So, we assume the existence of the following new functions:

$$\mathrm{new}\mathcal{V} : \mathcal{E} \longrightarrow 2^{\mathcal{D}}_f \longrightarrow \mathcal{V} \qquad \mathrm{all}\mathcal{V} : \mathcal{E} \longrightarrow 2^{\mathcal{V}}_f \qquad \mathrm{typ}\mathcal{V} : \mathcal{E} \longrightarrow 2^{\mathcal{D}}_f$$

Functions $\mathrm{new}\mathcal{D}$ and $\mathrm{all}\mathcal{D}$ are defined the same as they were for ground environments, however recall that in the meanwhile the notion of active constant has been slightly modified. The application $\mathrm{new}\mathcal{V}(\xi) : D$ returns a variable of type D which is inactive in ξ. The function $\mathrm{all}\mathcal{V}$ returns the finite set of all the variables which are active in the argument, and eventually $\mathrm{typ}\mathcal{V}(\xi)$ yields the union of the types of the variables in $\mathrm{all}\mathcal{V}(\xi)$.

We define below the non-ground dual of the initial environment ξ^N. It actually represents in an abstract way all the possible partitions of N.

Definition 26 Letting $\mathcal{L}$ be a list of distinct elements of $\mathcal{N}_I$, the *initial non-ground environment* $\xi^{\mathcal{L}}$ is defined as

$$\xi^{\mathcal{L}} = \mathrm{add\_var}(\mathrm{Id}_{\mathcal{E}}, \mathcal{L})$$

$$\text{where} \quad \begin{array}{l} \mathrm{add\_var}(\xi, [\,]) = \xi \\ \mathrm{add\_var}(\xi, [x, \mathcal{L}]) = \mathrm{add\_var}(\xi + (x, \mathrm{new}\mathcal{V}(\xi) : (\mathrm{all}\mathcal{D}(\xi) \cup \mathrm{new}\mathcal{D}(\xi)), \mathcal{L}) \end{array}$$

□

In the non-ground $\pi\xi$-calculus the evaluation of symbolic obligations becomes a crucial issue: some equalities of names must be checked, some others must be imposed. The equalities to be tested are those relative to names whose instantiation has been caused by a communication or a placeholder concretion. The equalities to be forced, instead, are relative to those names the process at hand is supposed to be a permanent function of. This exactly captures the essence of non-groundness. Since we are reasoning here modulo name substitutions, imposing the equality $x = y$ when both x and y have initially been free names (*cf.* Ex. 24) is equivalent to assuming that, no matter what $x\sigma$ and $y\sigma$ actually are, if they are the same then the process can move and from now onwards x and y must be taken to be the same.

As a consequence of the above discussion, the obligations of the symbolic semantics must sometimes be considered as actual constraints. This, in turn, implies that the evaluation of obligations cannot be any longer a boolean predicate. It rather yields a relation to be added with the environment. To explain this we can focus our attention on the evaluation of the obligation $x = y$ in ξ. First suppose that the equality of x and y must only be checked. Then, depending on whether $x \in [y]_\xi$ or not, the evaluation function returns the empty relation (denoted ϕ) or the inconsistent relation (written ε). Assume now that x and y are recognized as names which could be arbitrarily substituted by putting the process into a given context. In this case the equality $x = y$ is forced into the environment by letting the evaluation function return the constraint-relation (x, y).

Having to cope with inconsistent relations, a more general notion of sum operation on environments is needed, too.

Definition 27 Let R_1, R_2 be relations over $\mathcal{N} \cup \mathcal{D} \cup \mathcal{V}$ or the distinguished element $\varepsilon \notin \mathcal{N} \cup \mathcal{D} \cup \mathcal{V}$ denoting the inconsistent relation. The *sum* $R_1 \uplus R_2$ is defined as:

$$R_1 \uplus R_2 = \begin{cases} \varepsilon & \text{if } R_1 = \varepsilon \text{ or } R_2 = \varepsilon \\ \varepsilon & \text{if } R_1 + R_2 \text{ is not consistent} \\ R_1 + R_2 & \text{otherwise} \end{cases}$$

□

A couple of comments are due. First, observe that the active variables of the initial non-ground environment $\xi^{\mathcal{L}}$ have incremental types. So, given any pair of variables $(v_1 : D_1), (v_2 : D_2) \in \mathrm{all}\mathcal{V}(\xi^{\mathcal{L}})$ it holds that $D_1 \cap D_2 \neq \emptyset$. This guarantees that the names occurring in $\mathcal{L}$ can be arbitrarily joined in the same equivalence class without giving raise to any inconsistency. Second, recall that observable results will be actual functions. Then, correspondingly to the possible inconsistency induced by adding environments with constraints, the result function may yield on some arguments a distinguished error element denoted by $\perp$.

It will be made clear soon that requiring the external observer to interact with functions is too strong a requirement when aiming at characterizing extensional semantics via ordinary bisimilarity. We use functions to give a compact representation of behaviours modulo substitutions. Suppose P to be the process at hand, with $\mathrm{fn}(P) = \{x_1, \ldots, x_n\}$. Intuitively, observing the n-ary function f gives information that for all the tuples $(y_1, \ldots, y_n)$ such that $f(y_1, \ldots, y_n) \neq \perp$, process $P\{y_1/x_1, \ldots, y_n/x_n\}$ can move performing $f(y_1 \ldots y_n)$. By this, a process P_1 able to execute $g_1 = \lambda x_1 x_2 . x_2$ has the same action capabilities of P_2 which can perform both g_1 and $g_2 = \lambda x_1 x_2 .\ x_1 = x_2 \rightarrow x_2, \perp$. Nevertheless, if the bisimulation game required to match just the above behaviours, then the two processes would not be equated. In fact P_1 could not properly react to the g_2-move played by P_2. This leads us to reason about functional observations via the standard partial ordering relation over functions.

Definition 28 Let f and h be n-ary functions. Then $f \sqsubseteq h$ iff $f\tilde{a} \neq \perp$ implies $f\tilde{a} = h\tilde{a}$. The constant function always yielding $\perp$ is denoted by $\underline{\perp}$. □

5.1 Strong late non-groundness, again

In the following we will introduce the late non-ground $\pi\xi$-calculus, namely we will define a new top-level transition relation that allows us to characterize late π-calculus non-groundness without resorting to λ-closures.

Introducing the issue of non-groundness we commented on the opportunity of considering any process P as a function of its free names $\tilde{x}$. We also noticed that correspondingly to the firing of input actions any process behaves like a function of the input parameter. These two kinds of functionality are conceptually distinct and must be guaranteed do not interfere. The dependence on the tuple $\tilde{x}$ continues to hold throughout the computation. By contrast, the dependence on an input placeholder is not permanent, it gets immediately lost when the parameter is actualized.

$$\frac{P \xrightarrow{\omega} P' \qquad \omega \neq \langle x(y), C\rangle \qquad \xi' \in \eta_N(\xi, \omega) \qquad \rho \sqsubseteq \delta_N(\xi', \omega) \qquad \rho \neq \bot}{\xi :: P \xrightarrow{\rho}\!\!\rhd_{\eta_N} \xi' :: P'}$$

$$\frac{P \xrightarrow{\langle x(y), C\rangle} P' \qquad \xi' \in \eta_N(\xi, x(y), C) \qquad \rho \sqsubseteq \delta_N(\xi', x(y), C) \qquad \rho \neq \bot}{\xi :: P \xrightarrow{\rho}\!\!\rhd_{\eta_N} \xi' :: \lambda y.P'}$$

Table 7: definition of $\longrightarrow\!\!\rhd_{\eta_N}$

So, although any derivative of P may have some free names more than P, it must still be dealt with as a function of $\tilde{x}$ only. The information on the identity of the names in the tuple $\tilde{x}$ cannot be recovered syntactically, then it becomes a parameter of the update function, and hence of the top-level transition relation.

The non-ground late behaviour of $\pi\xi$-processes is described by the relation $\longrightarrow\!\!\rhd_{\eta_N}$ defined in Table 7. The parameter N represents the set of names that the given running process is supposed to be a permanent function of. The extensional $\pi\xi$-semantics is then given by the strong bisimulation induced by $\longrightarrow\!\!\rhd_{\eta_N}$, hereby denoted $\sim_{\eta_N}$.

New pairs of update and result functions are involved in the definition of the non-ground transition relation. Besides this, the main difference between $\longrightarrow\!\!\rhd_{\eta_N}$ and its ground dual $\longrightarrow\!\!\rhd$ is that labels are not directly given by the result function $\delta_N(\xi', \omega)$, but rather by functions less than or equal to $\delta_N(\xi', \omega)$ and still distinct from $\bot$.

The actual definition of the non-ground update function η_N requires a specialized evaluation function whose definition is reported in Tab. 8 together with the definition of (η_N, δ_N). The revised evaluation function takes as additional argument the set of names whose instantiation is to be forced by-need. As expected, $\{\!|C|\!\}N\xi$ yields either a finite relation over $\mathcal{N} \cup \mathcal{D} \cup \mathcal{V}$ or the inconsistent relation ε.

The first step in computing $\eta_N(\xi, \alpha, C)$ is adding ξ with the relation yielded by the evaluation function $\{\!|C|\!\}N\xi$. If the sum operation causes inconsistency, then the update function returns the empty set and, by definition of $\longrightarrow\!\!\rhd_{\eta_N}$, the $\pi\xi$-calculus process taken into account remains blocked. When $\xi \uplus \{\!|C|\!\}N\xi$ results into an environment $\xi_1 \neq \varepsilon$, the definition of $\eta_N(\xi, \alpha, C)$ is essentially the same as that of $\eta(\xi_1, \alpha, C)$ and indeed in this event we still use the ground sum operation '$+$'.

One single difference between η_N and η is worth noticing. It is relative to the instantiation of input placeholders, and then to the definition of $\eta_N(\xi, [y], C)$. The ground update function would associate y with all the constants active in ξ_1 plus a new one. Here y is non-deterministically associated with all the active variables (*i.e.*, with each of the names initially free) and with all the constants activated during the ongoing computation plus a new one.

The observable result $\delta_N(\xi', \omega)$ yields a function of all the variables which are active in ξ'. Maybe associating variables with some tuple of constants in the variables types gives raise to inconsistency. For instance, this is the case for the association of

$$
\begin{array}{rcl}
\{\!|C|\!\}N\xi & = & \texttt{case}\ C\ \texttt{in} \\
 & & \quad \begin{array}{rcl}
 \texttt{true} & : & \phi \\
 \texttt{false} & : & \varepsilon \\
 x{\downarrow} & : & \xi(x){\downarrow} \longrightarrow \phi, \quad \varepsilon \\
 x = y & : & x \in N \ \texttt{and}\ y \in N \longrightarrow (x,y), \quad x\ \xi\ y \longrightarrow \phi, \quad \varepsilon \\
 x \neq y & : & x\ \xi\ y \longrightarrow \varepsilon, \quad \phi \\
 C_1 \wedge C_2 & : & \{\!|C_1|\!\}N\xi \uplus \{\!|C_2|\!\}N\xi
 \end{array} \\
 & & \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\eta_N \xi \alpha C & = & \texttt{case}\ \xi \uplus \{\!|C|\!\}\xi\ \texttt{in} \\
 & & \quad \begin{array}{rcl}
 \varepsilon & : & \emptyset \\
 \xi_1 & : & \texttt{case}\ \alpha\ \texttt{in} \\
 & & \quad \begin{array}{rcl}
 \tau & : & \xi_1 \\
 \tau[x/y] & : & \xi_1 + (y,x) \\
 \overline{x}(y), \overline{x}y & : & \xi_1(y){\downarrow} \longrightarrow \xi_1, \quad \xi_1 + (y,\ \mathrm{new}\mathcal{D}(\xi_1)) \\
 x(y) & : & \xi_1 \\
 {[y]} & : & \bigcup_{(v:D)\in \mathrm{all}\,\mathcal{V}(\xi_1)} \xi_1 + (y, v:D) \quad \cup \\
 & & \bigcup_{c \in (\mathrm{all}\,\mathcal{D}(\xi_1)\ \setminus\ \mathrm{typ}\,\mathcal{V}(\xi_1))\ \cup\ \mathrm{new}\mathcal{D}(\xi_1))} \xi_1 + (y,c)
 \end{array} \\
 & & \texttt{end\_case}
 \end{array} \\
 & & \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\delta_N \xi \alpha C & = & \lambda d_1 : D_1 \ldots \lambda d_m : D_m \quad . \quad \texttt{case}\ \xi \uplus \sum_{i=1}^{m}(v_i, d_i)\ \texttt{in} \\
 & & \qquad \begin{array}{rcl}
 \varepsilon & : & \perp \\
 \xi_1 & : & \texttt{case}\ \alpha\ \texttt{in} \\
 & & \quad \begin{array}{rcl}
 \tau, \tau[x/y] & : & \tau \\
 \overline{x}(y), \overline{x}y & : & \overline{\xi_1(x)}\xi_1(y) \\
 x(y) & : & \xi_1(x) \\
 {[y]} & : & [\xi_1(y)]
 \end{array} \\
 & & \texttt{end\_case}
 \end{array} \\
 & & \qquad \texttt{end\_case}
\end{array}
$$

where $\{v_1 : D_1, \ldots, v_m : D_m\} = \mathrm{all}\mathcal{V}(\xi), \quad D_1 \subset D_2 \subset \ldots \subset D_m$

Table 8: definition of (η_N, δ_N)

$(v_1 : \{c_1\}, v_2 : \{c_1, c_2\})$ with the tuple (c_1, c_2) when $v_2 \in [v_1]_{\xi'}$. Then the function δ_N may return on some tuple of arguments the distinguished element $\perp$ denoting undefinedness. However for $\xi' \in \eta_N(\xi, \omega)$ it holds that $\delta_N(\xi', \omega) \neq \underline{\perp}$. Also, the result yielded by the actualization $\delta_N(\xi', \omega)(\tilde{c}) \neq \perp$ is analogous to the observation returned by the ground function δ.

In order to give a deeper intuition about the definition of η_N and δ_N we discuss in the following a case study.

Example 29 Let $N = \{x, z\}$ and let $\xi^{\mathcal{L}}$ be as defined below.

$$\xi^{\mathcal{L}} = \mathrm{Id}_{\mathcal{E}} + (x, v_1 : \{c_1\}) + (z, v_2 : \{c_1, c_2\})$$

We compute η_N and δ_N correspondingly to few symbolic actions and environments.

1. First let $\omega = \langle [y], \mathtt{true} \rangle$.
   Function $\eta_N(\xi^{\mathcal{L}}, \omega)$ returns a set composed by three updated environments, say $\{\xi_1, \xi_2, \xi_3\}$. Assuming $\mathrm{new}\mathcal{D}(\xi^{\mathcal{L}}) = c_3$, the following equalities hold.

$$\begin{array}{ll} \xi_1 = \xi^{\mathcal{L}} + (y, v_1) = \xi^{\mathcal{L}} + (y, x) & \delta_N(\xi_1, \omega) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\, [d_1] \\ \xi_2 = \xi^{\mathcal{L}} + (y, v_2) = \xi^{\mathcal{L}} + (y, z) & \delta_N(\xi_2, \omega) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\, [d_2] \\ \xi_3 = \xi^{\mathcal{L}} + (y, c_3) & \delta_N(\xi_3, \omega) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\, [c_3] \end{array}$$

2. Assume now $\omega' = \langle \tau, y = z \rangle$.
   Here $\eta_N(\xi_1, \omega') = \eta_N(\xi_3, \omega') = \emptyset$ in fact the input placeholder y was instantiated by z neither in ξ_1 nor in ξ_3. Hence $\{\!|y = z|\!\}N\xi_1 = \{\!|y = z|\!\}N\xi_3 = \varepsilon$. By contrast, as $\{\!|y = z|\!\}N\xi_2 = \phi$, the following holds

$$\begin{aligned} &\eta_N(\xi_2, \omega') = \{\xi_2\} \\ &\delta_N(\xi_2, \omega') = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\, \tau \end{aligned}$$

   The idea is that, no matter what any $z\sigma$ could be, the action τ may take place only if y was instantiated by $z\sigma$. In such a case the action surely fires, with 'surely' being represented by the fact that $\delta_N(\xi_2, \omega')$ is a constant function.

3. Now consider $\omega'' = \langle \tau, x = z \rangle$.
   Here, letting $j = 1, 2, 3$, the following holds.

$$\begin{aligned} &\{\!|x = z|\!\}N\xi_j = (x, z) \\ &\eta_N(\xi_j, \omega'') = \{\xi_j + (x, z)\} \\ &\delta_N(\xi_j + (x, z), \omega'') = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\, d_1 = d_2 \rightarrow \tau, \perp \end{aligned}$$

   The above indicates that, independently on the actual instantiation of y, the action τ may be performed only with the proviso that $x\sigma$ and $z\sigma$ are the same. $\square$

Example 29 makes clear that the information furnished by the set indexing η_N cannot be recovered by simply checking the association of names with variables. As it is the case for $(y\ \xi_1\ v_1)$ and for $(y\ \xi_2\ v_2)$, those associations might have been inherited, for instance, because of the instantiation of some placeholder.

We can now provide a better understanding of the reason why we defined the relation $\longrightarrow\!\!\!\triangleright_{\eta_N}$ resorting to the partial ordering on functions, namely using labels $\rho \sqsubseteq \delta_N(\xi', \omega)$. First observe that

- the $\pi\xi$-calculus non-ground process $\xi :: P$ stays for a substitution-indexed family of π-calculus processes, say $\{P_\pi\sigma\}_\sigma$

- any substitution σ is actually witnessed by a tuple $\tilde{c}$ such that $\delta_N(\xi', \omega)(\tilde{c}) \neq \perp$

In this view, function $\delta_N(\xi', \omega)$ is a monolithic – functional vs. point-wise – representation of the family of labels $\{\alpha_\sigma\}_\sigma$ such that $P_\pi \sigma \xrightarrow{\alpha_\sigma} P'_\pi$. Taking $\rho \sqsubseteq \delta_N(\xi', \omega)$ is meant to factorize the observable effect $\delta_N(\xi', \omega)$ while retaining a functional view about processes. Indeed, if the relation $\longrightarrow\!\!\triangleright_{\eta_N}$ were defined assuming $\rho = \delta_N(\xi', \omega)$ then the induced strong bisimulation semantics over $\pi\xi$-calculus processes would be finer than late congruence. We comment further on this issue by means of the example below.

Example 30 Consider the following late congruent processes.

$$\begin{array}{lcl} P & \equiv & x(y).(\tau + [x = z]\tau) \\ Q & \equiv & x(y).\tau \end{array}$$

Let $\xi^{\mathcal{L}} = \mathrm{Id}_{\mathcal{E}} + (x, v_1 : \{c_1\}) + (z, v_2 : \{c_1, c_2\})$ with $N = \{x, z\}$ and also assume $\xi_1 = \xi^{\mathcal{L}} + (y, v_1)$. Then the following holds.

$$\xi^{\mathcal{L}} :: P \xrightarrow{\rho}\!\!\triangleright_{\eta_N} \xi^{\mathcal{L}} :: \lambda y.(\tau + [x = z]\tau) \xrightarrow{\rho'}\!\!\triangleright_{\eta_N} \xi_1 :: \tau + [x = z]\tau$$

$$\xi^{\mathcal{L}} :: Q \xrightarrow{\rho}\!\!\triangleright_{\eta_N} \xi^{\mathcal{L}} :: \lambda y.\tau \xrightarrow{\rho'}\!\!\triangleright_{\eta_N} \xi_1 :: \tau$$

A close look at Ex. 29 shows that

$$\delta_N(\eta_N(\xi_1, \tau, [x = z]), \tau, [x = z]) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\ d_1 = d_2 \to \tau, \perp$$

while

$$\delta_N(\eta_N(\xi_1, \tau, \mathtt{true}), \tau, \mathtt{true}) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\ \tau$$

If $\longrightarrow\!\!\triangleright_{\eta_N}$ were directly labelled by $\delta_N(\xi', \omega)$ in Tab. 7, then $\xi^{\mathcal{L}} :: P$ and $\xi^{\mathcal{L}} :: Q$ would be deemed to be not strong bisimilar. In fact, the derivative $\xi_1 :: \tau$ of $\xi^{\mathcal{L}} :: Q$ could not match the move played by $\xi_1 :: (\tau + [x = z]\tau)$ and labelled $\lambda d_1 d_2.\ d_1 = d_2 \to \tau, \perp$. $\square$

In the following we address the issue of translating non-ground environments into ground environments. Once this has been done, we can proceed and prove the characterization theorem.

The following definition is about the consistent transformation of variables into constants in their types.

Definition 31 Let ξ be a non-ground environment. Then a *variable substitution for* ξ is a substitution from variables to constants defined as it follows.

- if $\mathrm{all}\mathcal{V}(\xi) = \emptyset$ then the only variable substitution for ξ is the empty substitution
- if $\mathrm{all}\mathcal{V}(\xi) = \{v_1 : D_1, \ldots, v_m : D_m\}$ with $D_1 \subset \ldots \subset D_m$, then a variable substitution σ for ξ is defined as $\sigma = \sigma^{v_1}(\xi) \ldots \sigma^{v_m}(\xi)$ where
  - $\sigma^{v_j}(\xi)$ is the substitution $\{a_j / v_j\}$ with

$$a_j = \begin{cases} \xi(v_j) & \text{if } [v_j]_\xi \text{ is defined-by-constant} \\ c_j \in D_j & \text{otherwise} \end{cases}$$

- $\forall i, j = 1, \ldots, m$ if $v_i \in [v_j]_\xi$ then $\mathrm{Im}(\sigma^{v_i}(\xi)) = \mathrm{Im}(\sigma^{v_j}(\xi))$ $\square$

Variable substitutions are applied to non-ground environments in order to encode them into sets of ground environments. We convey to use the notation below.

<u>Notation</u> Let ξ be a non-ground environment, and let σ be a substitution from variables to constants. Then $\xi\sigma$ is defined to be

$$\xi\sigma = \begin{cases} \xi & \text{if } \sigma \text{ is empty} \\ (\mathrm{erase}\mathcal{V}(\xi + (v, c))v)\sigma' & \text{if } \sigma = \{c/v\}\sigma' \end{cases}$$

where the application of $\mathrm{erase}\mathcal{V} : \mathcal{E} \to \mathcal{V} \to \mathcal{E}$ to the arguments ξ and v is assumed to return an environment like ξ but where v is made inactive, namely all and only the associations (a, v) with $a \neq v$ have been erased. $\square$

The following proposition asserts that there exists a precise relationship between the two evaluation functions considered so far.

Proposition 32 *Let σ be a variable substitution for ξ, and C be an obligation. Then $\xi \uplus \{\!|C|\!\}N\xi \neq \varepsilon$ iff $\xi \uplus \{\!|C|\!\}N\xi \uplus \sum_{\mathrm{Dm}(\sigma)}(v_j, v_j\sigma) \neq \varepsilon$ iff $[\![C]\!]\xi\sigma$ iff $\xi\sigma = (\xi \uplus \{\!|C|\!\}N\xi)\sigma$.*

Proof: Notice, in the definition of the two evaluation functions, the correspondence between `tt` and the empty relation ϕ and the one between `ff` and the inconsistent relation ε. Then the thesis is an immediate consequence of the definition of variable substitution. $\square$

Making use of variable substitutions, the functional behaviours of non-ground late $\pi\xi$-calculus processes can be related to the behaviours of ground agents. It can be shown that a precise relationship there exists. We informally explain it in the following.

- Suppose that a ground environment is defined to be equal to $\xi\sigma$ with ξ non-ground and σ variable substitution for ξ. Then the fact that $\xi\sigma :: P$ can move implies that $\xi :: P$ can move as well, since its functional behaviour is distinct from $\perp$ in at least one point.

- For the other way round, if the non-ground environment ξ' is reached by executing the observable function f then there exists a corresponding ground step for each point of definedness of f.

Proposition 33 *Let $\xi :: P$ be a non-ground $\pi\xi$-calculus process. Then the following holds.*

1. *Assume $\xi\sigma :: P \xrightarrow{\rho}\!\!\triangleright\ \xi'' :: P'$ with σ variable substitution for ξ.*

   Then $\xi :: P \xrightarrow{f}\!\!\triangleright_{\eta_N} \xi' :: P'$ with $f(\mathrm{Im}(\sigma)) = \rho$. Also, σ is a variable substitution for ξ' and $\xi'' = \xi'\sigma$.

2. *Assume $\xi :: P \xrightarrow{f}\!\!\triangleright_{\eta_N} \xi' :: P'$.*
   Then $f(\tilde{c}) \neq \perp$ iff $\tilde{c} = \mathrm{Im}(\sigma')$ with σ' variable substitution for ξ'. Moreover any σ' such that $f(\mathrm{Im}(\sigma')) \neq \perp$ is also a variable substitution for ξ and $\xi\sigma' :: P \xrightarrow{\rho}\!\!\triangleright\ \xi'\sigma' :: P'$ with $\rho = f(\mathrm{Im}(\sigma'))$.

PROOF:

1. Let $\xi\sigma :: P \stackrel{\rho}{\longrightarrow\!\!\triangleright} \xi'' :: P'$

   $\Longrightarrow$ $P \stackrel{\langle\alpha,C\rangle}{\longrightarrow} P''$ with $[\![C]\!]\xi\sigma$ and P'' such that either $P' \equiv P''$ or $P' \equiv \lambda y.P''$ depending on α

   $\Longrightarrow$ By Prop. 32 and the hypothesis that σ is a variable substitution for ξ it follows that $\xi \uplus \{\!|C|\!\}N\xi \neq \varepsilon$ and that $(\xi \uplus \{\!|C|\!\}N\xi)\sigma = \xi\sigma$

   $\Longrightarrow$ The thesis comes by the definitions of η, η_N, δ, δ_N and by the definition of variable substitution.

2. Assume $\xi :: P \stackrel{f}{\longrightarrow\!\!\triangleright}_{\eta_N} \xi' :: P'$

   $\Longrightarrow$ $P \stackrel{\langle\alpha,C\rangle}{\longrightarrow} P''$ with $\xi \uplus \{\!|C|\!\}N\xi \neq \varepsilon$ and P'' such that either $P' \equiv P''$ or $P' \equiv \lambda y.P''$.

   Moreover, the thesis that $f(\tilde{c}) \neq \perp$ iff $\tilde{c} = \mathrm{Im}(\sigma')$ with σ' variable substitution for ξ' directly comes by the definition of variable substitution. In fact variable substitutions are the only possible associations of variables with constants which do not give raise to inconsistency. Hence any $\tilde{c}$ distinct from the codomain of a variable substitution for ξ' is such that $\xi' \uplus \sum_{i=1}^{m}(v_i, c_i) = \varepsilon$ which implies $f(\tilde{c}) = \perp$.

   Also, as in general ξ' encodes more constraints than ξ does, any variable substitution for ξ' is a variable substitution for ξ as well

   $\Longrightarrow$ By Prop. 32 and the hypothesis that σ' is a variable substitution for ξ it follows that $[\![C]\!]\xi\sigma'$ holds. Hence the thesis that $\xi\sigma' :: P \stackrel{\rho}{\longrightarrow\!\!\triangleright} \xi'\sigma' :: P'$ is a consequence of the definitions of of η, η_N, δ, δ_N. $\square$

We can now state a more efficient characterization of strong late non-ground π-calculus semantics. The coincidence result exploits the relationship between non-ground $\pi\xi$-processes and the $\pi\xi$-agents obtained by λ-closure. On the one hand, non-ground environments are let to burst out into sets of ground environments. On the other way round, sets of ground environments are given a compact representation via one single non-ground environment.

Theorem 34 *(coincidence with strong late non-ground semantics)*
Let P_1, P_2 be π-calculus processes and let $\mathcal{L}$ be a list containing all and only the elements of the set $N = \mathrm{fn}(P_1, P_2)$. Then $P_1 \sim_L P_2$ iff $\xi^{\mathcal{L}} :: P_1 \sim_{\eta_N} \xi^{\mathcal{L}} :: P_2$.

PROOF: We prove in the following that

$$\xi^{\mathcal{L}} :: P_1 \sim_{\eta_N} \xi^{\mathcal{L}} :: P_2 \ \text{ iff } \ \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}) \sim \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L})$$

then the thesis comes by Th. 22.

(IF)
Assume that a relation $\mathcal{S} \subseteq \sim_{\eta_N}$ there exists with $(\xi^{\mathcal{L}} :: P_1, \xi^{\mathcal{L}} :: P_2) \in \mathcal{S}$. Then transform the relation $\mathcal{S}$ into the relation $\mathcal{S}_\lambda \cup \mathcal{S}_1$ with the two components defined as follows.

$$\mathcal{S}_\lambda \;\; = \;\; \bigcup_{\mathcal{L}_2, \xi_{\mathcal{L}_1}} \{(\xi_{\mathcal{L}_1} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}_2), \xi_{\mathcal{L}_1} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L}_2))\}$$

where $\mathcal{L}_1, \mathcal{L}_2$ are lists such that $\mathcal{L} = [\mathcal{L}_1, \mathcal{L}_2]$ and, for $j = 1, 2$, the environment $\xi_{\mathcal{L}_1}$ ranges over the environments which are reachable from $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_j, [\mathcal{L}_1, \mathcal{L}_2])$ by firing the λ-abstractions over the names contained in $\mathcal{L}_1$.

$$\mathcal{S}_1 \quad = \quad \bigcup_{(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}} \{(\mathrm{adj\_con}(\xi_P \sigma) :: P, \mathrm{adj\_con}(\xi_Q \sigma) :: Q)\}$$

where σ is a variable substitution for both ξ_P and ξ_Q and $\mathrm{adj\_con}(\_\_)$ is a function which possibly renames the constants of the defined equivalence classes of its argument. The use of the function $\mathrm{adj\_con}(\_\_)$ is due to the fact that, when applying a variable substitution to a non-ground environment, some care is needed in order to be respectful of the constant generation mechanism. For instance, assume that such a mechanism returns in the order the constants $(c_1, c_2, c_3, \dots)$ and that $\xi = \mathrm{Id}_{\mathcal{E}} + (x, v_1 : \{c_1\}) + (z, v_2 : \{c_1, c_2\}) + (y, c_3)$ and eventually that $\sigma = \{c_1/v_1, c_1/v_2\}$. In such a case $\xi\sigma = \mathrm{Id}_{\mathcal{E}} + (x, c_1) + (z, c_1) + (y, c_3)$. Instead of this we want to get the environment $\mathrm{Id}_{\mathcal{E}} + (x, c_1) + (z, c_1) + (y, c_2)$. That is why we use the function $\mathrm{adj\_con}(\xi\sigma)$. It is supposed to suitably act on $[c]_{\xi\sigma}$ in order to let $\mathrm{all}\mathcal{D}(\xi\sigma)$ be given by the first k generated constants whenever $\mathrm{card}(\mathrm{all}\mathcal{D}(\xi\sigma)) = k$.

By construction the relation $\mathcal{S}_\lambda \cup \mathcal{S}_1$ contains the pair $(\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}), \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L}))$. We now want to prove that $(\mathcal{S}_\lambda \cup \mathcal{S}_1) \subseteq \sim$.

Given any pair $(\xi_{\mathcal{L}_1} :: P, \xi_{\mathcal{L}_1} :: Q) \in \mathcal{S}_\lambda$ it can be easily proved that whenever $\xi_{\mathcal{L}_1} :: P$ moves then the process $\xi_{\mathcal{L}_1} :: Q$ can match the move and the derivative agents are either in $\mathcal{S}_\lambda$ or in $\mathcal{S}_1$. Letting y_n be the last name in the list $\mathcal{L}$, the only care is relative to processes of the shape $P \equiv \lambda y_n.P_i$ and $Q \equiv \lambda y_n.P_j$ with $i, j = 1, 2$ and $i \neq j$. In such cases the thesis that the derivative agents belong to the relation $\mathcal{S}_1$ just depends on the definitions of $\lambda\_\mathrm{clos}(\_\_, \mathcal{L})$ and of $\xi^{\mathcal{L}}$. In fact, suppose that y_k is the k-th element of the list $\mathcal{L}$. Then there is a neat relationship between the *set* of constants associated with y_k in the environments which are derivable from $\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(\_\_, \mathcal{L})$ and the *type* of the variable associated with y_k in the environment $\xi^{\mathcal{L}}$. Both the two sets are given by the first k generated constants. Then the thesis directly comes from the definition of variable substitution.

Now consider the relation $\mathcal{S}_2$ defined as it follows.

$$\mathcal{S}_2 \quad = \quad \bigcup_{(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}} \{(\xi_P \sigma :: P, \xi_Q \sigma :: Q)\}$$

where σ is a variable substitution for both ξ_P and ξ_Q. As far as the proof that $\mathcal{S}_1 \subseteq \sim$ is concerned, we get rid of the function $\mathrm{adj\_con}(\_\_)$ and show instead that the relation $\mathcal{S}_2$ is a late ground strong bisimulation. Then the thesis is a consequence of Prop. 17.

Assume that $(\xi_P \sigma :: P, \xi_Q \sigma :: Q) \in \mathcal{S}_2$ because of $(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}$ with σ variable substitution for both ξ_P and ξ_Q. Also assume that $\xi_P \sigma :: P \xrightarrow{\rho} \xi'' :: P'$.

$\Longrightarrow$ By Prop. 33 it follows that $\xi_P :: P \xrightarrow{f} \xi_{P'} :: P'$ with $\rho = f(\mathrm{Im}(\sigma))$ and σ variable substitution also for $\xi_{P'}$ and $\xi'' = \xi_{P'} \sigma$

$\Longrightarrow$ By $(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}$ it follows that some $\xi_{Q'} :: Q'$ there exists such that $\xi_Q :: Q \xrightarrow{f} \xi_{Q'} :: Q'$ with $(\xi_{P'} :: P', \xi_{Q'} :: Q') \in \mathcal{S}$

$\Longrightarrow$ By Prop. 33 and by $f(\mathrm{Im}(\sigma)) = \rho$ it comes that $\xi_Q \sigma :: Q \xrightarrow{\rho} \xi_{Q'} \sigma :: Q'$ with σ variable substitution for $\xi_{Q'}$

$\Longrightarrow$ By $(\xi_{P'} :: P', \xi_{Q'} :: Q') \in \mathcal{S}$ and by definition of $\mathcal{S}_2$ it follows that $(\xi_{P'} \sigma :: P', \xi_{Q'} \sigma :: Q') \in \mathcal{S}_2$

(ONLY IF)
Assume that the relation $\mathcal{T} \subseteq \sim$ contains the pair $(\mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_1, \mathcal{L}), \mathrm{Id}_{\mathcal{E}} :: \lambda\_\mathrm{clos}(P_2, \mathcal{L}))$. Then define the relation $\mathcal{T}'$ acting as follows. For each subset of $\mathcal{T}$ of the shape

$$\bigcup_\sigma \{(\mathrm{adj\_con}((\xi^{\mathcal{L}} + \xi_P)\sigma)) :: P, \mathrm{adj\_con}((\xi^{\mathcal{L}} + \xi_Q)\sigma) :: Q)\}$$

where $P, Q \not\equiv \lambda z.R$ for $z \in N$ and σ is a variable substitution for both $(\xi^{\mathcal{L}} + \xi_P)$ and $(\xi^{\mathcal{L}} + \xi_Q)$, the relation $\mathcal{T}'$ is let to contain the pair $(\xi^{\mathcal{L}} + \xi_P :: P, \xi^{\mathcal{L}} + \xi_Q :: Q)$.
By the definitions of $\lambda\_\mathrm{clos}(\_\_, \_\_)$ and of variable substitution, the pair $(\xi^{\mathcal{L}} :: P_1, \xi^{\mathcal{L}} :: P_2)$ is contained in $\mathcal{T}'$. We now prove that $\mathcal{T}' \subseteq \sim_{\eta_N}$.
Assume that the pair $(\xi^v_P :: P, \xi^v_Q :: Q)$ is put in $\mathcal{T}'$ because of

$$\bigcup_\sigma \{(\mathrm{adj\_con}(\xi^v_P\sigma) :: P, \mathrm{adj\_con}(\xi^v_Q\sigma) :: Q)\} \subseteq \mathcal{T}$$

Also, suppose that $\xi^v_P :: P \xrightarrow{f}_{\eta_N} \xi^v_{P'} :: P'$
$\Longrightarrow$ By Prop. 33 for all $\tilde{c}$ such that $f(\tilde{c}) \neq \perp$ it holds that $\tilde{c} = \mathrm{Im}(\sigma')$ with σ' variable substitution for both ξ^v_P and $\xi^v_{P'}$. Moreover it also holds that $\xi^v_P\sigma' :: P \xrightarrow{\rho} \xi^v_{P'}\sigma' :: P'$ with $\rho = f(\tilde{c})$
$\Longrightarrow$ By definition of the function $\mathrm{adj\_con}(\_\_)$ and by

$$\bigcup_\sigma \{(\mathrm{adj\_con}(\xi^v_P\sigma) :: P, \mathrm{adj\_con}(\xi^v_Q\sigma) :: Q)\} \subseteq \mathcal{T} \subseteq \sim$$

it follows that $\xi^v_Q\sigma' :: Q \xrightarrow{\rho} \xi^v_{Q'}\sigma' :: Q'$ with

$$\{(\mathrm{adj\_con}(\xi^v_{P'}\sigma') :: P', \mathrm{adj\_con}(\xi^v_{Q'}\sigma') :: Q')\} \subseteq \mathcal{T}$$

$\Longrightarrow$ $\xi^v_Q :: Q \xrightarrow{h}_{\eta_N} \xi^v_{Q'} :: Q'$ with $(\xi^v_{P'} :: P', \xi^v_{Q'} :: Q') \in \mathcal{T}'$ by

$$\{(\mathrm{adj\_con}(\xi^v_{P'}\sigma') :: P', \mathrm{adj\_con}(\xi^v_{Q'}\sigma') :: Q')\} \subseteq \mathcal{T}$$

for all σ' which is a variable substitution for both $\xi^v_{P'}$ and $\xi^v_{Q'}$ and where $h = f$. In fact, by the assumed generality of σ', the two functions are point-wise equal on each $\tilde{c} = \mathrm{Im}(\sigma')$ and by Prop. 33 these are the only points where the two functions are distinct from $\perp$. $\square$

5.2 Digression on open semantics

In order to show the flexibility of the non-ground $\pi\xi$-calculus, we now go further in adopting a functional interpretation of processes and show an alternative characterization of π-calculus open semantics, whose definition follows [San96].

Definition 35 A binary symmetric relation $\mathcal{S}$ is an *open bisimulation* if $P \;\mathcal{S}\; Q$ implies that for all name substitutions σ

if $P\sigma \xrightarrow{\alpha} P'$ with $\mathrm{bn}(\alpha) \notin \mathrm{fn}(P\sigma, Q\sigma)$, then for some Q', $Q\sigma \xrightarrow{\alpha} Q'$ and $P' \;\mathcal{S}\; Q'$

P is *open bisimilar* to Q, written $P \sim_O Q$, if $P \;\mathcal{S}\; Q$ for some open bisimulation $\mathcal{S}$. $\square$

$$\frac{P \xrightarrow{\omega} P' \qquad \xi' \in \eta_O(\xi, \omega) \qquad \rho \sqsubseteq \delta_O(\xi', \omega) \qquad \rho \neq \bot}{\xi :: P \overset{\rho}{\longrightarrow\!\!\triangleright}_{\eta_O} \xi' :: P'}$$

Table 9: definition of $\longrightarrow\!\!\triangleright_{\eta_O}$

Open semantics moves name instantiation inside the definition of bisimulation, immediately qualifying itself as a congruence. Indeed, as well as late non-ground bisimilarity, it involves a universal quantification over substitutions, so requiring at each step an infinite number of checks. Nevertheless, a more efficient characterization of open bisimilarity was proposed. It is based on a specialized transition system that allows name substitutions to be delayed as much as possible.

The $\pi\xi$-calculus open semantics is in that same spirit. Indeed, free names and instantiation are interpreted much as variables and unification are dealt with in logic programming. When an input action fires, a variable is associated with the formal parameter of the performed action. This shows up in the definitions of the open $\pi\xi$-calculus transition system, denoted by $\longrightarrow\!\!\triangleright_{\eta_O}$, where the actualization of input placeholders is delayed as much as possible. The transition relation $\longrightarrow\!\!\triangleright_{\eta_O}$ is reported on Tab. 9, and again the actual $\pi\xi$-semantics is given by the induced strong bisimulation, written $\sim_{\eta_O}$. There is no need here to let the transition relation be parametric over a set of names. In fact, the present instantiation strategy genuinely consists in delaying actualization as much as possible. At any point during computation, all the variables can still be considered as formal parameters. This implies that the evaluation function can always add constraints on variables. More specifically, suppose that one name out of x and y, or also both, are associated with a variable. If so, evaluating whether or not $x = y$ in ξ results in the constraint (x, y). Then the addition of ξ with the relation (x, y)

- either respects the required equality
  ($e.g.$ $[x]_\xi = [y]_\xi = \{x, y, c\}$)
- or induces the required equality
  ($e.g.$ $[x]_\xi = \{x, c\}$ and $[y]_\xi = \{y, v : \{c\} \cup D\}$ or, also,
  $[x]_\xi = \{x, v_1 : D\}$ and $[y]_\xi = \{y, v_2 : D \cup D'\}$)
- or causes inconsistency
  ($e.g.$ $[x]_\xi = \{x, c_1\}$ and $[y]_\xi = \{y, c_2\}$)

The refined evaluation function $\langle\!\langle C \rangle\!\rangle\xi$ is reported in Tab. 10 together with specialized update and result functions for open semantics. As far as $\langle\!\langle x = y \rangle\!\rangle\xi$ is concerned, notice that a constraint is added to the environment only when ($\xi(x)\!\downarrow$ and $\xi(y)\!\downarrow$). If this is not the case, then we have just to test whether or not $(x\ \xi\ y)$. In fact, if ($\xi(x)\!\uparrow$ or $\xi(y)\!\uparrow$) then at least one name out of x and y is private. Think, for instance, of the obligation in the symbolic action $o_x\langle \overline{z}y, z\!\downarrow\rangle = \langle \overline{z}(y), x = y \wedge z\!\downarrow\rangle$. The issue of variable instantiation plays no role when x and y are as above: no association can make them to be the same.

$$
\begin{array}{rcl}
\langle\!\![C]\!\!\rangle \xi & = & \texttt{case}\ C\ \texttt{in} \\
& & \quad \begin{array}{rcl}
\texttt{true} & : & \phi \\
\texttt{false} & : & \varepsilon \\
x\!\downarrow & : & \xi(x)\!\downarrow\ \longrightarrow \phi, \quad \varepsilon \\
x = y & : & (\xi(x)\!\downarrow \text{ and } \xi(y)\!\downarrow) \longrightarrow (x,y), \quad x\ \xi\ y \longrightarrow \phi, \quad \varepsilon \\
x \neq y & : & x\ \xi\ y \longrightarrow \varepsilon, \quad \phi \\
C_1 \wedge C_2 & : & \langle\!\![C_1]\!\!\rangle \xi \uplus \langle\!\![C_2]\!\!\rangle \xi
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\eta_O \xi \alpha C & = & \texttt{case}\ \xi \uplus \langle\!\![C]\!\!\rangle \xi\ \texttt{in} \\
& & \quad \begin{array}{rcl}
\varepsilon & : & \emptyset \\
\xi_1 & : & \texttt{case}\ \alpha\ \texttt{in} \\
& & \quad \begin{array}{rcl}
\tau & : & \xi_1 \\
\tau[x/y] & : & \xi_1 + (y,x) \\
\overline{x}(y), \overline{x}y & : & \xi_1(y)\!\downarrow\ \longrightarrow \xi_1, \quad \xi_1 + (y,\ \text{new}\mathcal{D}(\xi_1)) \\
x(y) & : & \xi_1 + (y, \text{new}\mathcal{V}(\xi_1) : (\text{all}\mathcal{D}(\xi_1) \cup \text{new}\mathcal{D}(\xi_1))) \\
[y] & : & \xi_1
\end{array} \\
& & \texttt{end\_case}
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

$$
\begin{array}{rcl}
\delta_O \xi \alpha C & = & \lambda d_1 : D_1 \ldots \lambda d_m : D_m \quad . \quad \texttt{case}\ \xi \uplus \sum_{i=1}^{m} (v_i, d_i)\ \texttt{in} \\
& & \qquad \begin{array}{rcl}
\varepsilon & : & \perp \\
\xi_1 & : & \texttt{case}\ \alpha\ \texttt{in} \\
& & \quad \begin{array}{rcl}
\tau, \tau[x/y] & : & \tau \\
\overline{x}(y), \overline{x}y & : & \overline{\xi_1(x)}\xi_1(y) \\
x(y) & : & \xi_1(x) \\
[y] & : & [\xi_1(y)]
\end{array} \\
& & \texttt{end\_case}
\end{array} \\
& & \qquad \texttt{end\_case}
\end{array}
$$

where $\{v_1 : D_1, \ldots, v_m : D_m\} = \text{all}\mathcal{V}(\xi), \quad D_1 \subset D_2 \subset \ldots \subset D_m$

Table 10: definition of (η_O, δ_O)

Apart from the use of distinct criteria for evaluating obligations, the function η_O differs from η_N only as far as input actions are concerned. The improvement in efficiency over the transition systems described so far is made evident by the fact that $\eta_O(\xi, x(y), C)$ causes – at most – the association of the name y with one single fresh variable. The information that such a variable may assume any of the constants already active in ξ or a new one is coded in the variable type. This treatment of input actions delays as much as possible the actual instantiation of the parameter. More properly, the instantiation is fully delegated to the constraint system implemented by the evaluation function $\langle\!\langle \_ \rangle\!\rangle_{\_\_}$ and hence it takes place *only if, and when,* it is strictly necessary for the computation to go on.

The definition of the result function δ_O is analogous to the one of δ_N. Anyway, contrary to δ_N, the number of arguments of the function returned by δ_O is not *a priori* fixed by the definition of the non-ground environment $\xi^{\mathcal{L}}$ chosen to start running. Here, the result yielded by δ_O is a function of all the names in the list $\mathcal{L}$ and of all the placeholders of the input actions already fired. The act of inputting potentially makes free one more name. Hence, assuming the list $\mathcal{L}$ in $\xi^{\mathcal{L}}$ to contain all the free names of the initial π-calculus process, the present transition system interprets any agent as a function of all its free names.

The open instantiation strategy can be illustrated by means of the following example.

Example 36 Let $\xi^{\mathcal{L}}$ be defined as it follows.

$$\xi^{\mathcal{L}} = \mathrm{Id}_{\mathcal{E}} + (x, v_1 : \{c_1\}) + (z, v_2 : \{c_1, c_2\})$$

We investigate on the values returned by η_O and by δ_O when they are applied to distinct environments and symbolic actions.

1. Assume $\omega = \langle x(y), x\!\downarrow\rangle$ and suppose $\mathrm{new}\mathcal{D}(\xi^{\mathcal{L}}) = c_3$. Then

$$\eta_O(\xi^{\mathcal{L}}, \omega) = \{\xi^{\mathcal{L}} + (y, v_3 : \{c_1, c_2, c_3\})\}$$
$$\delta_O(\xi^{\mathcal{L}} + (y, v_3), \omega) = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\lambda d_3 : \{c_1, c_2, c_3\}.\ d_1$$

2. For $\omega' = \langle \tau, true\rangle$ the following holds.

$$\eta_O(\xi^{\mathcal{L}} + (y, v_3), \omega') = \{\xi^{\mathcal{L}} + (y, v_3)\}$$
$$\delta_O(\xi^{\mathcal{L}} + (y, v_3), \omega') = \lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\lambda d_3 : \{c_1, c_2, c_3\}.\ \tau$$

3. Suppose now $\omega'' = \langle \tau, y = z\rangle$. In such a case

$$\eta_O(\xi^{\mathcal{L}} + (y, v_3), \omega'') = \{\xi^{\mathcal{L}} + (y, v_3) + (y, z)\}$$
$$\delta_O(\xi^{\mathcal{L}} + (y, v_3) + (y, z), \omega'') =$$
$$\lambda d_1 : \{c_1\}.\lambda d_2 : \{c_1, c_2\}.\lambda d_3 : \{c_1, c_2, c_3\}.\ d_3 = d_2 \to \tau, \perp$$

   Given any name substitution σ, the result yielded by $\delta_O(\xi^{\mathcal{L}} + (y, v_3) + (y, z), \omega'')$ means that the action τ may be performed only with the proviso that y is instantiated by $z\sigma$. In the meanwhile, the name $z\sigma$ could be either the same as $x\sigma$ ($d_1 = d_2 = d_3 = c_1$) or distinct from it ($d_1 = c_1$ and $d_2 = d_3 = c_2$). Obviously, when y is instantiated by a name which is assumed to be fresh w.r.t. all the other names around ($d_3 = c_3$) the requirement $y = z\sigma$ cannot be met. □

Open π-calculus semantics distinguishes the process $x(y).(\tau.\tau + \tau)$ from $x(y).(\tau.\tau + \tau + \tau.[y = z]\tau)$. Correspondingly in the $\pi\xi$-calculus, we can see that

$$\xi^{\mathcal{L}} :: x(y).(\tau.\tau + \tau) \not\sim_O \xi^{\mathcal{L}} :: x(y).(\tau.\tau + \tau + \tau.[y = z]\tau)$$

for $\xi^{\mathcal{L}} = \mathrm{Id}_{\mathcal{E}} + (x, v_1 : \{c_1\}) + (z, v_2 : \{c_1, c_2\})$. The reason for this is that process $\xi^{\mathcal{L}} :: x(y).(\tau.\tau + \tau)$ cannot win any bisimulation game started by the partner and terminating in the state $\xi^{\mathcal{L}} + (y, v_3) :: [y = z]\tau$. More precisely, a bisimulation relation should contain

- either the pair $(\xi^{\mathcal{L}} + (y, v_3) :: [y = z]\tau, \xi^{\mathcal{L}} + (y, v_3) :: \tau)$
- or the pair $(\xi^{\mathcal{L}} + (y, v_3) :: [y = z]\tau, \xi^{\mathcal{L}} + (y, v_3) :: \mathtt{nil})$

This is impossible. In the first case the process $\xi^{\mathcal{L}} + (y, v_3) :: [y = z]\tau$ could not match the move $(\lambda d_1 d_2 d_3.\tau)$ performed by $\xi^{\mathcal{L}} + (y, v_3) :: \tau$. In the second case, the deadlocked $\xi^{\mathcal{L}} + (y, v_3) :: \mathtt{nil}$ could not react to any $\rho \sqsubseteq \lambda d_1 d_2 d_3.\ d_3 = d_2 \to \tau, \perp$.

The following proposition, analogous to Prop. 32, provides the basis for the encoding of open non-groundness into the ground $\pi\xi$-calculus.

Proposition 37 *Let σ be a variable substitution for ξ, and C be an obligation. Then $\xi \uplus \langle\!\langle C \rangle\!\rangle \xi \neq \varepsilon$ iff $\xi \uplus \langle\!\langle C \rangle\!\rangle \xi \uplus \sum_{\mathrm{Dm}(\sigma)}(v_j, v_j\sigma) \neq \varepsilon$ iff $[\![C]\!]\xi\sigma$.*

PROOF: The statement is a direct consequence of the definitions of the involved evaluation functions and of variable substitution. The only interesting difference between $[\![\_]\!]\_$ and $\langle\!\langle \_ \rangle\!\rangle\_$ raises from the evaluation of obligations of the shape $x = y$ when $(\xi(x){\downarrow}$ and $\xi(y){\downarrow})$. So, assume $\xi \uplus \langle\!\langle x = y \rangle\!\rangle \xi \neq \varepsilon$. This does hold iff $[x]_\xi$ and $[y]_\xi$ are

- either both defined-by-constant and such that $[x]_\xi \cap \mathcal{D} = [y]_\xi \cap \mathcal{D}$
- or both defined-by-variable
- or one is defined-by-variable via $v : D$ and the other is defined-by-constant via $c \in D$

Then the theses by definition of variable substitution and of $\xi\sigma$. □

The next statement is the actual corresponding of Prop. 33. It addresses the relationship between variable substitutions and functional observations.

Proposition 38 *Assume $\xi :: P \xrightarrow{f}_{\eta_O} \xi' :: P'$. Then the following holds.*

1. *$f(\tilde{c}) \neq \perp$ iff $\tilde{c} = \mathrm{Im}(\sigma)$ with σ variable substitution for ξ'*
2. *if σ is a variable substitution for ξ' then $\mathrm{frag}(\sigma, \xi)$ is a variable substitution for ξ, where $\mathrm{frag}(\sigma, \xi) = \{v_1\sigma/v_1, \ldots, v_m\sigma/v_m\}$ for all$\mathcal{V}(\xi) = \{v_1, \ldots, v_m\}$*

PROOF:

1. By definition, the set of the variable substitutions for a given environment represents all the possible ways to transform variables into constants without giving raise to inconsistency. Hence, in the definition of δ_O the sum $\xi' \uplus \sum_{i=1}^{m}(v_i, c_i)$ is distinct from the distinguished element ε iff the tuple $(c_1, \ldots, c_m)$ is the codomain of some variable substitution.

2. The updated environment ξ' potentially encodes some constraints more than ξ. Then, although the other way round is not true, whenever $\mathrm{all}\mathcal{V}(\xi') = \mathrm{all}\mathcal{V}(\xi)$ any variable substitution for ξ' is a variable substitution for ξ as well. By definition of η_O, correspondingly to the symbolic execution of the input action $x(y)$ it holds that $\mathrm{all}\mathcal{V}(\xi') = \mathrm{all}\mathcal{V}(\xi) \cup \{v_m : D_m\}$. In such a case a variable substitution for ξ may be obtained by the variable substitution σ for ξ' by simply getting rid of the component relative to the variable v_m. In fact, correspondingly to the execution of $x(y)$ it holds that $[y]_\xi = \{y\}$ and $[y]_{\xi'} = \{y, v_m\}$. Hence any variable substitution for ξ' has the shape $\sigma^v\{c_m/v_m\}$ with $c_m \in D_m$ and $\mathrm{Dm}(\sigma^v) = \mathrm{all}\mathcal{V}(\xi)$. □

The following characterization theorem concludes our investigation into call-by-need instantiation strategies in the $\pi\xi$-calculus. The proof of the result involves either the previous encoding of non-groundness into groundness or the encoding of $\pi\xi$-calculus into π-calculus (and back) reported in Appendix A.

Theorem 39 *(coincidence with open semantics)*
Let P_1, P_2 be π-calculus processes and let $\mathcal{L}$ be a list containing all and only the elements of $\mathrm{fn}(P_1, P_2)$. *Then $P_1 \sim_O P_2$ iff $\xi^{\mathcal{L}} :: P_1 \sim_{\eta_O} \xi^{\mathcal{L}} :: P_2$*

Proof: See Appendix B. □

6 Concluding remarks

We provided operational and axiomatic characterizations of late, either strong or weak, π-calculus congruences. A coincidence result with open semantics was also shown. Target languages of our characterizations were either the π-calculus or the $\pi\xi$-calculus, a process algebra whose ground version was already proved to coincide with ground π-calculus.

The main contributions of the paper are the finitary nature of the proposed characterizations and the fact that they allow π-calculus semantics to be rephrased in terms of CCS equivalences which come equipped with well-known mathematical properties and verification tools. By this, our results could be particularly promising in the perspective of veryfing mobile systems.

Relating to π-calculus, it was shown that each non-ground equivalence relation can be expressed by closing the corresponding ground bisimilarity under the substitutions of a suitable finite family $\{\sigma_i\}_i$. This result can be directly instantiated to early semantics, either strong or τ-forgetting [Qua96]. So, the cardinality of $\{\sigma_i\}_i$ gives full insights about the actual complexity of π-congruence checking, both late and early.

Then, the $\pi\xi$-calculus view allowed processes to be explicitely interpreted as functions of their free names. So we resorted, in turn, to ground environments paired with λ-closures of π-processes, and to functional environments paired with standard π-processes.

The main advantage of the first of these approaches is that the mere evolution of a λ-closed term induces the generation of (the environments corresponding to) the substitutions $\{\sigma_i\}_i$ needed to check non-groundness. This, on its side, adds effectiveness to the very first alternative characterization we presented. Also, interpreting late congruence as bisimilarity of ground $\pi\xi$-processes permits equational characterizations for non-groundness to be directly stated on the $\pi\xi$-calculus axiom system for groundness [FMQ95].

Eventually, by introducing the non-ground dual of ground $\pi\xi$-calculus, we exploited, for the characterization of both late and open semantics, a more promising call-by-need discipline for the instantiation of the arguments which the running process is considered to depend on. Analogous instantition strategies are a common feature of *symbolic* semantics for data-dependent calculi [HL95]. Indeed, a call-by-need discipline underlies the alternative characterizations of late and open π-calculus semantics presented in [Lin95, Lin94] and [San96], respectively. The relationship between their approach and the one proposed here deserves further investigation. What is immediately evident is the difference at the extensional level. Symbolic semantics is defined as closure over a family of bisimulation relations indexed by equalities and inequalities on names. By contrast, in the non-ground $\pi\xi$-calculus, the call-by-need strategy is adopted just to recognize the minimal symbolic requirements for the computation to go on. Then those requirements are actually interpreted and made concrete by updating environments. So, $\pi\xi$-processes are equated by standard bisimulation semantics.

Another topic for future work, suggested by the proposed functional interpretation of processes, is the study of possible relationships between our results and the π-calculus presentation in [HLMP97].

A final comment is about non-ground π-calculus semantics of the early family. They can be characterized using an approach analogous to that we adopted for the late case. Precisely, the early view can be retrieved as a special case of late $\pi\xi$-semantics by minor changes to the ground and non-ground transition systems [Qua96]. In the same spirit as the free input actions of [MPW93], those changes essentially amount to make atomic any input step and the subsequent instantiation move.

Acknowledgments

We would like to thank Gian-Luigi Ferrari and Ugo Montanari for the joint work that provided the basis of the $\pi\xi$-calculus, and for helpful comments relating to this.

References

[Abr91] S. Abramsky. A Domain Equation for Bisimulation. *Information and Computation*, 92(2):161–218, 1991.

[ABV94] L. Aceto, B. Bloom, and F. Vaandrager. Turning SOS Rules into Equations. *Information and Computation*, 111(1):1–52, 1994.

[Ace94] L. Aceto. On "axiomatising finite concurrent processes". *SIAM Journal of Computing*, 23(4):852–863, 1994.

[BK84] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information and Control*, 60:109–137, 1984.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems. *ACM Trans. on Programming Languages and Systems*, 15(1):36–72, 1993.

[DS85] R. De Simone. Higher level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science*, 37(3):245–267, 1985.

[FMQ95] G.-L. Ferrari, U. Montanari, and P. Quaglia. The Weak Late π-calculus Semantics as Observation Equivalence. In I. Lee and S.A. Smolka, editors, *Proc. 6th International Conference on Concurrency Theory, CONCUR '95*, volume 962 of *LNCS*, pages 57–71. Springer-Verlag, 1995.

[FMQ96] G.-L. Ferrari, U. Montanari, and P. Quaglia. A π-calculus with Explicit Substitutions. *Theoretical Computer Science*, 168(1):53–103, 1996.

[Hen88] M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal of Computing*, 17(5):997–1017, 1988.

[HL95] M. Hennessy and H. Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138:353–389, 1995.

[HLMP97] F. Honsell, M. Lenisa, U. Montanari, and M. Pistore. Final Semantics for the π-calculus. Submitted for publication, 1997.

[HM85] M. Hennessy and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Lin94] H. Lin. Symbolic Bisimulation and Proof Systems for the π-Calculus. Technical Report 7/94, School of Cognitive and Computing Sciences, University of Sussex, 1994.

[Lin95] H. Lin. Complete Inference Systems for Weak Bisimulation Equivalences in the π-Calculus. In P.D. Mosses, M. Nielsen, and M.I. Schwartzbach, editors, *Proc. 6th International Joint Conference CAAP/FASE, TAPSOFT '95*, volume 915 of *LNCS*. Springer-Verlag, 1995.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[Mil92] R. Milner. The Polyadic π-Calculus: a Tutorial. In F.L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1992.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Part I and II. *Information and Computation*, 100(1):1–77, 1992.

[MPW93] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI-Conference*, volume 104 of *LNCS*. Springer-Verlag, 1981.

[Plo81] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI-FN-19, Computer Science Department, Aarhus University, 1981.

[Qua96] P. Quaglia. *The π-calculus with explicit substitutions.* PhD thesis, Università degli Studi di Pisa, Dipartimento di Informatica, 1996. Report TD-09/96.

[San96] D. Sangiorgi. A Theory of Bisimulation for the π-calculus. *Acta Informatica*, 33(1):69–97, 1996.

Appendix A

We recall in the following the main constructions and encodings needed to show Th. 7. They result from mild simplifications of analogous statements appeared in [FMQ96]. These simplifications, on their hand, are essentially due to the introduction of explicit λ-prefixes which allows us to avoid the use of more sophisticated environments, similar to those presented in Sec. 5 of this paper.

As commented in Sec. 2, the proof of the characterization result is quite complex. The definitions and the few auxiliary statements that follow are not meant to cover each single detail. They have rather been selected to bear out a sketch of the proof of Th. 7, and to provide the formal constructions used in Appendix B to show Th. 39. Complete proofs and detailed comments on encodings can be found in [Qua96], Chapter 3.

The first definition presents the core issue of the transformation of environments into actual name substitutions. Roughly, each relevant name x in ξ is classified according to the following criterion:

- it was involved in the communication of a bound name and there is still a name $z \in [x]_\xi$ which occurs bound in P;
- it was involved in a communication which should be indeed a closing communication;
- it is really (*i.e.* semantically) a free name of $\xi :: P$, namely $\xi(x)\downarrow$.

Definition 40 (restriction, constant, admissible substitutions) Let $\xi :: P$ be a $\pi\xi$-calculus process. Also, given the set $A \subseteq \mathcal{N}$, assume to be able to choose a *canonical representative* of A, say $\mathrm{crep}(A) \in A$. (*E.g.* assume the existence of a standard ordering on the names in $\mathcal{N}$, then $\mathrm{crep}(A)$ might be the least element of A w.r.t. that order). Then we define the following substitutions.

- letting $\mathrm{rep}(x, \xi :: P) = z$ whenever $([x]_\xi \cap \mathrm{bn}(P)) = \{z\}$

  $$s(\xi :: P) = \{\mathrm{rep}(x_0, \xi :: P)/x_0, \dots, \mathrm{rep}(x_m, \xi :: P)/x_m\}$$

  where $\mathrm{Dm}(s(\xi :: P)) = \{x \mid \mathrm{card}([x]_\xi) \geq 2 \text{ and } \xi(x)\uparrow \text{ and } ([x]_\xi \cap \mathrm{bn}(P)) \neq \emptyset\}$
- the *restriction substitution* σ^r of $\xi :: P$ is

  $$\sigma^r(\xi :: P) = \{\mathrm{crep}([x_0]_\xi)/x_0, \dots, \mathrm{crep}([x_m]_\xi)/x_m\}$$

  where $\mathrm{Dm}(\sigma^r(\xi :: P)) = \{x \mid \mathrm{card}([x]_\xi) \geq 2 \text{ and } \xi(x)\uparrow \text{ and } ([x]_\xi \cap \mathrm{bn}(P)) = \emptyset\}$
- the *constant substitution* σ^c of ξ is

  $$\sigma^c(\xi) = \{\xi(x_0)/x_0, \dots, \xi(x_m)/x_m\}$$

  where $\mathrm{Dm}(\sigma^c(\xi)) = \{x \mid [x]_\xi \text{ is defined-by-constant}\}$
- an *admissible name substitution* for $\{c_0, \dots, c_j, c_{j+1}, \dots, c_{j+k}\}$ is an injective substitution

$$
\begin{array}{rcl}
\mathrm{rsp}(R,P) & = & \texttt{case}\ R\ \texttt{in} \\
& & \begin{array}{rcl}
\emptyset & : & \{P\} \\
\{x\} \cup R' & : & \bigcup_{y \in R} \bigcup_{P' \in \mathrm{rnp}(y,P)} \mathrm{rsp}(R \setminus \{y\}, P')
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

where

$$
\begin{array}{rcl}
\mathrm{rnp}(y,P) & = & \texttt{case}\ P\ \texttt{in} \\
& & \begin{array}{rcl}
\texttt{nil} & : & \{\texttt{nil}\} \\
\alpha.P_1 & : & \{\alpha.P_1\} \\
[x=z]P_1 & : & \{[x=z]P_1\} \\
P_1 + P_2 & : & \{P_1 + P_2\} \\
P_1 \mid P_2 & : & y \in (\mathrm{fn}(P_1) \cap \mathrm{fn}(P_2)) \longrightarrow \{(y)(P_1 \mid P_2)\}, \\
& & \{(y)(P_1 \mid P_2)\} \cup \\
& & \{(P_1 \mid P_2')\ \text{s.t.}\ P_2' \in \mathrm{rnp}(y, P_2)\ \text{and}\ y \in \mathrm{bn}(P_1 \mid P_2')\} \\
& & \{(P_1' \mid P_2)\ \text{s.t.}\ P_1' \in \mathrm{rnp}(y, P_1)\ \text{and}\ y \in \mathrm{bn}(P_1' \mid P_2)\} \\
(z)P_1 & : & \{(z)P'\ \text{s.t.}\ P' \in \mathrm{rnp}(y, P_1)\} \\
!P_1 & : & \{!P_1\}
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

Table 11: definition of rsp(__, __)

$$\sigma^n = \{\imath^{-1}(c_0)/c_0, \ldots, \imath^{-1}(c_j)/c_j, x_1/c_{j+1}, \ldots, x_k/c_{j+k}\}$$

where $\{c_0, \ldots, c_j\} \subseteq \mathcal{D}_I$ and $\{c_{j+1}, \ldots, c_{j+k}\} \subseteq \mathcal{D}_{RT}$ and $\imath : \mathcal{N}_I \to \mathcal{D}_I$ is the bijective mapping of Def. 4 and $\{x_1, \ldots, x_k\} \subseteq \mathcal{N}$ □

The coming definition supplies us with the formal machinery needed to recover the lack in $\pi\xi$-calculus of a proper analogous of the Close rule. Given a set of names R and a process P, the function $\mathrm{rsp}(R, P)$ returns a set of processes where the names in R are non-deterministically put on top of the parallel compositions in P.

We then proceed with the formalization of the actual encoding of $\pi\xi$-calculus into π-calculus.

Definition 41 ($\mathrm{rsp}(R, P)$) Let R be a finite set of names, and let P be a π-calculus process. The set $\mathrm{rsp}(R, P)$ is defined in Table 11. □

Notation Given a π-calculus process P and a substitution s, we denote by $[P]^s$ the process obtained from P by substituting any free occurrence of x by xs, without caring of name clashes. Then, given $[P]^s$ and a substitution σ with $\mathrm{Dm}(\sigma) = \mathrm{bn}([P]^s)$, we denote by $[P]^s_\sigma$ the α-conversion of $[P]^s$ which respects σ, namely the π-calculus process obtained by substituting any occurrence of $y \in \mathrm{bn}([P]^s)$ by $y\sigma$. □

Translation 1 (from ground $\pi\xi$-calculus to π-calculus)
Let $\mathcal{S}$ be a bisimulation for η-reachable $\pi\xi$-calculus processes. Then

$$Tr_\xi(\mathcal{S}) = \bigcup_{(\xi_P::P',\xi_Q::Q')\in\mathcal{S}} Tr_\xi\big((\xi_P :: P, \xi_Q :: Q)\big)$$

where

- $P \equiv P''$ if $P' \equiv \lambda y.P''$ and $P \equiv P'$ otherwise
- $Q \equiv Q''$ if $Q' \equiv \lambda z.Q''$ and $Q \equiv Q'$ otherwise
- $Tr_\xi((\xi_P :: P, \xi_Q :: Q))$ is the set of pairs (P_π, Q_π) such that
  - $P_\pi \equiv_\alpha \widehat{P}(\sigma^c(\xi_P)\sigma^n)$
  - $Q_\pi \equiv_\alpha \widehat{Q}(\sigma^c(\xi_Q)\sigma^n)$
  - $\widehat{P} \in \mathrm{rsp}(\mathrm{Im}(\sigma^r(\xi_P :: P)), [P]^{s(\xi_P::P)}\sigma^r(\xi_P :: P))$
  - $\widehat{Q} \in \mathrm{rsp}(\mathrm{Im}(\sigma^r(\xi_Q :: Q)), [Q]^{s(\xi_Q::Q)}\sigma^r(\xi_Q :: Q))$
  - σ^n admissible name substitution for $\mathrm{Im}(\sigma^c(\xi_P)) \cup \mathrm{Im}(\sigma^c(\xi_Q))$. $\square$

Remark 42 Let P_π and Q_π be as in Trans. 1. By definition, for some P_r without homonymy either among bound names or among free and bound names $P_\pi \equiv_\alpha P_r$ where

- $P_r \in \mathrm{rsp}(\mathrm{Dm}(\sigma^r(\xi_P :: P))\sigma_P, [P]^{s(\xi_P::P)}_{\sigma'_P}\sigma_P)$
- $\sigma_P = \sigma^r(\xi_P :: P)\sigma''_P(\sigma^c(\xi_P)\sigma^n)$
- σ'_P, σ''_P are, respectively, the refreshment of the bound names of P and the refreshment of the names in $\mathrm{Im}(\sigma^r(\xi_P :: P))$

Similarly, for a suitable Q_r without homonymy and for suitable substitutions σ'_Q, σ''_Q, it holds that

$$Q_\pi \equiv_\alpha Q_r \in \mathrm{rsp}(\mathrm{Dm}(\sigma^r(\xi_Q :: Q))\sigma_Q, [Q]^{s(\xi_Q::Q)}_{\sigma'_Q}\sigma_Q)$$

where $\sigma_Q = \sigma^r(\xi_Q :: Q)\sigma''_Q(\sigma^c(\xi_Q)\sigma^n)$. $\square$

We now consider the opposite encoding: from π-calculus to $\pi\xi$-calculus. Given a π-process, supposingly obtained by some derivation, we need

- either to know which would be its syntactic structure if no name instantiation (but decoration of bound names of replicated subprocesses) had ever been applied during the derivation;
- or to build a suitable corresponding environment, and possibly prefix the process by a λ-abstraction.

The following definitions, leading to the wanted encoding, are meant to meet the demands above.

Definition 43 (sub(P)) Let P be a π-calculus process. The set sub(P) is defined in Table 12, where n stays for a positive integer. $\square$

$$
\begin{array}{rcl}
\mathrm{sub}(P) & = & \texttt{case } P \texttt{ in} \\
& & \begin{array}{rcl}
\texttt{nil} & : & \{\texttt{nil}\} \\
\alpha.P_1 & : & \{\alpha.P_1\} \cup \mathrm{sub}(P_1) \\
[x=z]P_1 & : & \{[x=z]P_1\} \cup \mathrm{sub}(P_1) \\
P_1 + P_2 & : & \{P_1 + P_2\} \cup \mathrm{sub}(P_1) \cup \mathrm{sub}(P_2) \\
P_1 \mid P_2 & : & \{P_1 \mid P_2\} \cup \{(P_1' \mid P_2') \text{ s.t.} \\
& & \qquad (P_1' \in \mathrm{sub}(P_1) \text{ and } P_2' \equiv P_2) \text{ or} \\
& & \qquad (P_1' \equiv P_1 \text{ and } P_2' \in \mathrm{sub}(P_2)) \text{ or} \\
& & \qquad (P_1' \in \mathrm{sub}(P_1) \text{ and } P_2' \in \mathrm{sub}(P_2))\} \\
(y)P_1 & : & \{(y)P_1\} \cup \{(y)P_1' \text{ s.t. } P_1' \in \mathrm{sub}(P_1)\} \cup \mathrm{sub}(P_1) \\
!P_1 & : & \{!P_1\} \cup \{(P_1' \mid P_1'') \text{ s.t.} \\
& & \qquad P_1' \in \mathrm{sub}((P_1)^{\mathrm{dec\_0}}) \text{ and} \\
& & \qquad P_1'' \in \mathrm{sub}(\mathrm{repl}((P_1)^{\mathrm{dec\_1}}, n-1))\}
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

where

$$
\begin{array}{rcl}
\mathrm{repl}(P, j) & = & \texttt{case } j \texttt{ in} \\
& & \begin{array}{rcl}
j = 0 & : & !P \\
j > 0 & : & (P)^{\mathrm{dec\_0}} \mid \mathrm{repl}((P)^{\mathrm{dec\_1}}, j-1)
\end{array} \\
& & \texttt{end\_case}
\end{array}
$$

Table 12: definition of sub(__)

NOTATION We use the operator $(R)^{\mathrm{dec}^n}$ meaning that the names in the set R have been arbitrarily decorated, up to n times, by either $(\_\_)^{\mathrm{dec\_0}}$ or $(\_\_)^{\mathrm{dec\_1}}$. Namely

$$(R)^{\mathrm{dec}^n} = \bigcup_{x \in R, |s| \leq n} \{x^{(s)}\}$$

where s is a string of zeros and ones. □

Definition 44 ((P_j, P_r, R, s, σ)-dual of P_π) Let P_j, P_π be π-calculus processes. Process $P \in \mathrm{sub}(P_j)$ is a (P_j, P_r, R, s, σ)-*dual* of P_π if a set of names R and name substitutions s, σ exist such that $P_\pi \equiv_\alpha P_r \in \mathrm{rsp}(R, [P]^s\sigma)$ where, letting Res be the set of the restricted names of P_j, the following holds:

- $R \subseteq \bigcup_{k \geq 0} (Res \setminus \mathrm{bn}(P))^{\mathrm{dec}^k}$
- $\mathrm{Dm}(s), \mathrm{Dm}(\sigma) \subseteq \bigcup_{k \geq 0} (\mathrm{bn}(P_j))^{\mathrm{dec}^k}$
- $\mathrm{Im}(s) \subseteq \mathrm{bn}(P)$
- $\mathrm{Im}(\sigma) \subseteq R \cup (\mathcal{N} \setminus \bigcup_{k \geq 0} (\mathrm{n}(P_j))^{\mathrm{dec}^k})$ □

Definition 45 ((N, R_P, s_P, σ_P)-environment for P_r) Let $N \subseteq \mathcal{N}_I$, and $R_P \subseteq \mathcal{N}_{RT}$, and s_P, σ_P be name substitutions, and P_r be a π-calculus process. Also, take $F = \{x \mid x \in \mathrm{Dm}(\sigma_P) \text{ and } x\sigma_P \notin R_P\}$. Then, a (N, R_P, s_P, σ_P)-*environment for* P_r is defined as follows.

- If $F = \emptyset$, then the only (N, R_P, s_P, σ_P)-environment for P_r is given by $\xi = \xi^N + \{(x, y) \mid y = xs_P \text{ or } (y = x\sigma_P \text{ and } x\sigma_P \in R_P) \text{ or } (y\sigma_P = x\sigma_P \text{ and } x\sigma_P \notin R_P)\}$.
- If $F \neq \emptyset$ then choose any ordering $(x_1, \ldots, x_m)$ of its elements.
  An (N, R_P, s_P, σ_P)-environment for P_r is given by the following recurrence:

$$\begin{array}{rcl} \xi_0 & = & \xi \\ \xi_{i+1} & = & \xi_i + (x_{i+1}, \mathrm{new}\mathcal{D}(\xi_i)) \end{array}$$

Let ξ_P be an (N, R_P, s_P, σ_P)-environment for P_r and let ξ_Q be an (N, R_Q, s_Q, σ_Q)-environment for Q_r where $R_Q \subseteq \mathcal{N}_{RT}$ and s_Q, σ_Q are name substitutions and Q_r is a π-calculus process. Then, ξ_P and ξ_Q are *compatible* iff $\mathrm{all}\mathcal{D}(\xi_P) = \mathrm{all}\mathcal{D}(\xi_Q)$ and $\forall x \in \mathrm{Dm}(\sigma_P), \forall y \in \mathrm{Dm}(\sigma_Q)$, ($x\sigma_P \notin \mathrm{bn}(P_r)$ and $y\sigma_Q \notin \mathrm{bn}(Q_r)$ and $x\sigma_P = y\sigma_Q$) implies $\xi_P(x) = \xi_Q(y)$. $\square$

Translation 2 (from π-calculus to ground $\pi\xi$-calculus)
Let P_1, P_2 be π-calculus processes with $N = \mathrm{fn}(P_1, P_2)$ and let $\mathcal{S}$ be a late ground bisimulation containing (P_1, P_2). Then define the relation $Tr_E((P_1, P_2), \mathcal{S})$ as it follows.

$$Tr_E((P_1, P_2), \mathcal{S}) = \bigcup_{(P_\pi, Q_\pi) \in \mathcal{S}} Tr_E((P_1, P_2), (P_\pi, Q_\pi))$$

where $Tr_E((P_1, P_2), (P_\pi, Q_\pi))$ is the set of pairs $(\xi_P :: P, \xi_Q :: Q)$ such that, letting $i, j = 1, 2$ and $i \neq j$, the following holds:

- P is a $(P_i, P_r, R_P, s_P, \sigma_P)$-dual of P_π and Q is a $(P_j, Q_r, R_Q, s_Q, \sigma_Q)$-dual of Q_π
- ξ_P is an (N, R_P, s_P, σ_P)-environment for P_r and
  ξ_Q is an (N, R_Q, s_Q, σ_Q)-environment for Q_r
- ξ_P and ξ_Q are compatible

The translation $Tr_L((P_1, P_2), \mathcal{S})$ is given by $Tr_E((P_1, P_2), \mathcal{S})$ plus the pairs defined as follows. For each subset of $Tr_E((P_1, P_2), \mathcal{S})$ of the shape

$$\bigcup_{c \in \mathrm{all}\mathcal{D}(\xi_P) \cup \mathrm{new}\mathcal{D}(\xi_P)} (\xi_P + (y, c) :: P, \xi_Q + (z, c) :: Q)$$

add $Tr_E((P_1, P_2), \mathcal{S})$ with the pair $(\xi_P :: \lambda y.P, \xi_Q :: \lambda z.Q)$. $\square$

Remark 46 By construction, if $(\xi_P :: P, \xi_Q :: Q) \in Tr_E((P_1, P_2), (P_\pi, Q_\pi))$ then

- $P_\pi \equiv_\alpha P_r \in \mathrm{rsp}((\mathrm{Dm}(\sigma^r(\xi_P :: P)))\sigma^r(\xi_P :: P), [P]^{s(\xi_P :: P)}\sigma^r(\xi_P :: P)(\sigma^c(\xi_P)\sigma^n))$
- $Q_\pi \equiv_\alpha Q_r \in \mathrm{rsp}((\mathrm{Dm}(\sigma^r(\xi_Q :: Q)))\sigma^r(\xi_Q :: Q), [Q]^{s(\xi_Q :: Q)}\sigma^r(\xi_Q :: Q)(\sigma^c(\xi_Q)\sigma^n))$

where P_r and Q_r are without homonymy of names and σ^n is an admissible name substitution for $\mathrm{Im}(\sigma^c(\xi_P)) \cup \mathrm{Im}(\sigma^c(\xi_Q))$ and the restriction substitutions $\sigma^r(\xi_P :: P)$ and $\sigma^r(\xi_Q :: Q)$ were computed fixing a particular choice for the canonical representative crep of Def. 40. $\square$

The next step is to establish an operational correspondence in between π-calculus and $\pi\xi$-calculus. What we state is actually the relationship between behaviours of processes which are the one encoding of the other. Since this is proved in an inductive way, we cannot work directly on translated processes, but rather have to consider the generic process which could occur in the derivation of a transition of the encoded agent. Such a generic process is much similar to the process P_π of Remark 42 and Remark 46, but can have restrictions on a subset of $\mathrm{Im}(\sigma^r(\xi_P :: P))$ instead of on the whole of it.

Definition 47 We call P_π^- any α-converse of $P_r^- \in \mathrm{rsp}(R\sigma, [P]_{\sigma'}^{s(\xi_P::P)}\sigma)$ where

- $R\sigma \subseteq \mathrm{Dm}(\sigma^r(\xi_P :: P))\sigma$
- $\mathrm{Dm}(\sigma') \subseteq \mathrm{bn}([P]^{s(\xi_P::P)})$
- $\sigma = \sigma^r(\xi_P :: P)\sigma''(\sigma^c(\xi_P)\sigma^n)$

with $\mathrm{Dm}(\sigma'') = \mathrm{Im}(\sigma^r(\xi_P :: P))$ and σ^n admissible name substitution for $D \supseteq \mathrm{Dm}(\sigma^c(\xi_P))$ and the refreshments σ', σ'' such that P_r^- is without homonymy either among its bound names or among its free and bound names. $\square$

Theorem 48 *(relating P_π^- with P) Let P_π^-, P_r^- be as in Def. 47 and let $s = s(\xi_P :: P)\sigma'\sigma$. Then the following holds.*

1. *If $P_\pi^- \xrightarrow{\overline{u_1}u_2} P'_\pi$ then*
   $P \xrightarrow{\langle \overline{x}y, C\wedge x\downarrow\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and* $xs = u_1$ *and* $ys = u_2$ *and*
   $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']_{\sigma'}^{s(\xi_P::P)}\sigma)$

2. *If $P_\pi^- \xrightarrow{\overline{u_1}(w)} P'_\pi$ with w fresh then*
   - *either* $P \xrightarrow{\langle \overline{x}(y), C\wedge x\downarrow\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and* $xs = u_1$ *and*
     $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']_{\sigma'}^{s(\xi_P::P)}\sigma\{w/ys\})$ *and* $ys \notin R\sigma$
   - *or* $P \xrightarrow{\langle \overline{x}y, C\wedge x\downarrow\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and* $xs = u_1$ *and*
     $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma \setminus \{ys\}, [P']_{\sigma'}^{s(\xi_P::P)}\sigma\{w/ys\})$ *and* $ys \in R\sigma$

3. *If $P_\pi^- \xrightarrow{u_1(w)} P'_\pi$ with w fresh then*
   $P \xrightarrow{\langle x(y), C\wedge x\downarrow\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and* $xs = u_1$ *and*
   $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']_{\sigma'}^{s(\xi_P::P)}\sigma\{w/ys\})$

4. *If $P_\pi^- \xrightarrow{\tau} P'_\pi$ then*
   - *either* $P \xrightarrow{\langle \tau, C\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and* $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']_{\sigma'}^{s(\xi_P::P)}\sigma)$
   - *or* $P \xrightarrow{\langle \tau[x/y], C\rangle} P'$ *and* $[\![C]\!]\xi_P$ *and*
     - *if* $[x]_{\xi_P} \cap \mathrm{bn}(P) \neq \emptyset$ *and* $[x]_{\xi_P} \cap \mathrm{bn}(P') = \emptyset$
       then $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma \cup \{w\}, [P']_{\sigma'}^{s(\xi_P::P)}\sigma\{w/xs, w/ys\})$ *with* w *fresh*
     - *else either* $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']_{\sigma'}^{s(\xi_P::P)\{xs/y\}}\sigma)$
       or $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma\{w/xs\}, [P']_{\sigma'}^{s(\xi_P::P)}\sigma\{w/xs, w/ys\})$ *with* w *fresh* $\square$

Theorem 49 *(relating P with P^-_π) Let P^-_π, P^-_r be as in Def. 47 and let $s = s(\xi_P :: P)\sigma'\sigma$. Then the following holds.*

1. *If $P \overset{\langle \overline{x}y, C\wedge x\downarrow\rangle}{\longrightarrow} P'$ and $[\![C]\!]\xi_P$ and $[x]_{\xi_P} \cap \mathrm{bn}(P) = \emptyset$ and $xs \notin R\sigma$ then*
   - *if $ys \notin R\sigma$ then $P^-_\pi \overset{\overline{xs}ys}{\longrightarrow} P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']^{s(\xi_P::P)}_{\sigma'}\sigma)$*
   - *else $P^-_\pi \overset{\overline{xs}(w)}{\longrightarrow} P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma \setminus \{ys\}, [P']^{s(\xi_P::P)}_{\sigma'}\sigma\{w/ys\})$ with w fresh*
2. *If $P \overset{\langle \overline{x}(y), C\wedge x\downarrow\rangle}{\longrightarrow} P'$ and $[\![C]\!]\xi_P$ and $[x]_{\xi_P} \cap \mathrm{bn}(P) = \emptyset$ and $xs \notin R\sigma$ then $P^-_\pi \overset{\overline{xs}(w)}{\longrightarrow} P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']^{s(\xi_P::P)}_{\sigma'}\sigma\{w/ys\})$ with w fresh*
3. *If $P \overset{\langle x(y), C\wedge x\downarrow\rangle}{\longrightarrow} P'$ and $[\![C]\!]\xi_P$ and $[x]_{\xi_P} \cap \mathrm{bn}(P) = \emptyset$ and $xs \notin R\sigma$ then $P^-_\pi \overset{xs(w)}{\longrightarrow} P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']^{s(\xi_P::P)}_{\sigma'}\sigma\{w/ys\})$ with w fresh*
4. *If $P \overset{\langle \tau, C\rangle}{\longrightarrow} P'$ and $[\![C]\!]\xi_P$ then $P^-_\pi \overset{\tau}{\longrightarrow} P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']^{s(\xi_P::P)}_{\sigma'}\sigma)$*
5. *If $P \overset{\langle \tau[x/y], C\rangle}{\longrightarrow} P'$ and $[\![C]\!]\xi_P$ then $P^-_\pi \overset{\tau}{\longrightarrow} P'_\pi$ and*
   - *if $[x]_{\xi_P} \cap \mathrm{bn}(P) \neq \emptyset$ and $[x]_{\xi_P} \cap \mathrm{bn}(P') = \emptyset$ then $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma \cup \{w\}, [P']^{s(\xi_P::P)}_{\sigma'}\sigma\{w/xs, w/ys\})$ with w fresh*
   - *else either $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma, [P']^{s(\xi_P::P)\{xs/y\}}_{\sigma'}\sigma)$ or $P'_\pi \equiv_\alpha P'_r \in \mathrm{rsp}(R\sigma\{w/xs\}, [P']^{s(\xi_P::P)}_{\sigma'}\sigma\{w/xs, w/ys\})$ with w fresh*

□

Eventually, the following proposition provides, in terms of statements on relevant name substitutions and obligations, what is needed to definitely relate (both ways) the symbolic transitions of the processes in Th. 48 and Th. 49 with the concrete moves of the (encoding/encoded) top-level $\pi\xi$-processes.

Proposition 50 *Let P_π, P_r and Q_π, Q_r be as in Remark 42. Then the following holds.*

1. *$xs(\xi_P :: P)\sigma'_P\sigma_P \in \mathrm{fn}(P_\pi)$ implies $[\![x\downarrow]\!]\xi_P$*
2. *$\xi_P(x) = \xi_Q(y) \in \mathcal{D}$ implies $xs(\xi_P :: P)\sigma'_P\sigma_P = ys(\xi_Q :: Q)\sigma'_Q\sigma_Q$*
3. *$[\![x\downarrow]\!]\xi_Q$ implies $[x]_{\xi_Q} \cap \mathrm{bn}(Q) = \emptyset$ and $xs(\xi_Q :: Q)\sigma'_Q\sigma_Q \notin \mathrm{Dm}(\sigma^r(\xi_Q :: Q))\sigma_Q$*

□

Proof of Theorem 7, sketch

Once proved the results on semantic correspondences, those about equational characterizations are easy corollaries of Th. 6. We then concentrate on the first kind of statements, dealing with all of the three items simultaneously.

(ONLY IF)

Assume that either $(\xi^N :: P_1) \doteq (\xi^N :: P_2)$ with $\doteq \in \{\sim, \approx\}$ or $(\xi^N :: P_1) \approx^c (\xi^N :: P_2)$. In both cases, by hypothesis there exists a bisimulation $\mathcal{S} \subseteq \doteq$ containing the pair $(\xi^N :: P_1, \xi^N :: P_2)$, and also some extra hypotheses do hold when

$(\xi^N :: P_1) \approx^c (\xi^N :: P_2)$. The relation $Tr_\xi(\mathcal{S})$ contains by construction the pair (P_1, P_2) and it is proven to be the wanted π-calculus equivalence relation.
Letting $(P_\pi, Q_\pi) \in Tr_\xi((\xi_P :: P, \xi_Q :: Q)) \subseteq Tr_\xi(\mathcal{S})$, the proof is by case analysis on $P_\pi \xrightarrow{\alpha} P'_\pi$. Each case goes as it follows. By Th. 48 the corresponding symbolic transition from P is recovered. Then Prop. 50 is invoked to guarantee that the symbolic step from P is actually converted into a concrete move from $\xi_P :: P$. By the hypothesis $(\xi_P :: P, \xi_Q :: Q) \in \mathcal{S}$, we then have complete information about the corresponding move of $\xi_Q :: Q$ and proceed to find out the wanted transition $Q_\pi \xrightarrow{\alpha} Q'_\pi$ with $(P'_\pi, Q'_\pi) \in Tr_\xi(\mathcal{S})$. This requires a kind of inversion of the above reasoning that appeals, this time, to Prop. 50 and Th. 49. Also, in the case when $\mathcal{S}$ is a weak relation (*i.e.* it proves that $(\xi^N :: P_1)$ and $(\xi^N :: P_2)$ are either weak bisimilar or weak congruent) we resort to an intermediate result about the relationship of the weak behaviours of $\xi_Q :: Q$ and Q_π. Eventually, when the action α is an input action, the fact that $(P'_\pi\{z/w\}, Q'_\pi\{z/w\}) \in Tr_\xi(\mathcal{S})$ for all z follows quite easily by the definition of the encoding.

(IF)
Assume that either $P_1 \dot{\asymp}_L P_2$ with $\dot{\asymp}_L \in \{\dot{\sim}_L, \dot{\approx}_L\}$ or $P_1 \dot{\simeq}_L P_2$. Also suppose that the relation $\mathcal{S} \subseteq \dot{\asymp}_L$ contains the pair (P_1, P_2) and witnesses their bisimilarity. By definition of Tr_L, the relation $Tr_L((P_1, P_2), \mathcal{S})$ contains the pair $(\xi^N :: P_1, \xi^N :: P_2)$. It remains to show that $Tr_L((P_1, P_2), \mathcal{S})$ is the required $\pi\xi$-calculus bisimulation, and, in case $P_1 \dot{\simeq}_L P_2$, also that $Tr_L((P_1, P_2), \mathcal{S})$ guarantees $(\xi^N :: P_1) \approx^c (\xi^N :: P_2)$. By construction, given any pair $(\xi_P :: P, \xi_Q :: Q) \in Tr_L((P_1, P_2), (P_\pi, Q_\pi))$ either both or none of P and Q are led by a λ-abstraction. In both cases the proof of $Tr_L((P_1, P_2), \mathcal{S}) \subseteq \sim$ is based on a case analysis of the actions performed by $\xi_P :: P$. Then assume that $\xi_P :: P \xrightarrow{\rho} \xi_{P'} :: P''$.
If P has the shape $\lambda y.P'$, namely if $\rho = [c]$ for some constant c, then the thesis is an easy consequence of the fact that, by definition of the encoding, some $\xi_Q :: \lambda z.Q'$ there exists such that $(\xi_P :: \lambda y.P', \xi_Q :: \lambda z.Q') \in Tr_L((P_1, P_2), \mathcal{S})$.
Suppose now that P is not led by a λ-abstraction. In this case we first appeal to the symbolic transition of P which induced the concrete move labelled by ρ. Prop. 50 ensures that the hypotheses of Th. 49 are met, hence we retrieve a corresponding transition from P_π. By that, and by $(P_\pi, Q_\pi) \in \mathcal{S}$, we switch to investigate the symbolic behaviour of Q, by invoking Th. 48, and possibly its weak version (in case we are dealing with a τ-forgetting relation). At this point of the proof, Prop. 50 can be applied to guarantee that the symbolic step actually induces the concrete move $\xi_Q :: Q \xrightarrow{\rho} \xi_{Q'} :: Q''$ with $(\xi_{P'} :: P'', \xi_{Q'} :: Q'') \in Tr_L((P_1, P_2), \mathcal{S})$, whether or not P'' is led by a λ-abstraction. $\square$

Appendix B

In the following we will refer to some of the main constructions and statements presented in Appendix A.

Translation 3 (from open $\pi\xi$-calculus to π-calculus)
Let $\mathcal{S}$ be a strong bisimulation relation contained in $\sim_{\eta_O}$. Then

$$Tr^v_\xi(\mathcal{S}) = \bigcup_{(\xi^v_P :: P, \xi^v_Q :: Q) \in \mathcal{S}} Tr_\xi\big((\xi^v_P \sigma^v :: P, \xi^v_Q \sigma^v :: Q)\big)$$

with

- σ^v variable substitution for both ξ^v_P and ξ^v_Q
- $Tr_\xi((\xi^v_P\sigma^v :: P, \xi^v_Q\sigma^v :: Q))$ defined analogously as in Trans. 1, that is as the set of pairs (P_π, Q_π) such that
  - $P_\pi \equiv_\alpha \widehat{P}(\sigma^c(\xi^v_P\sigma^v)\sigma^n)$
  - $Q_\pi \equiv_\alpha \widehat{Q}(\sigma^c(\xi^v_Q\sigma^v)\sigma^n)$
  - $\widehat{P} \in \mathrm{rsp}(\mathrm{Im}(\sigma^r(\xi^v_P\sigma^v :: P)), [P]^{s(\xi^v_P\sigma^v :: P)}\sigma^r(\xi^v_P\sigma^v :: P))$
  - $\widehat{Q} \in \mathrm{rsp}(\mathrm{Im}(\sigma^r(\xi^v_Q\sigma^v :: Q)), [Q]^{s(\xi^v_Q\sigma^v :: Q)}\sigma^r(\xi^v_Q\sigma^v :: Q))$
  - σ^n admissible name substitution for $\mathrm{Im}(\sigma^c(\xi^v_P\sigma^v)) \cup \mathrm{Im}(\sigma^c(\xi^v_Q\sigma^v))$. $\square$

Translation 4 (from π-calculus to open $\pi\xi$-calculus)
Let P_1, P_2 be π-calculus processes and let $\mathcal{S}$ be an open bisimulation containing (P_1, P_2). Then define the relation $Tr_O((P_1, P_2), \mathcal{S})$ as it follows.

$$Tr_O((P_1, P_2), \mathcal{S}) = \bigcup_{\mathcal{S}_{(P_\pi, Q_\pi)}} Tr_O((P_1, P_2), \mathcal{S}_{(P_\pi, Q_\pi)})$$

where

- $\mathcal{S}_{(P_\pi, Q_\pi)}$ is the subset of $\mathcal{S}$ of the shape $\bigcup_{\sigma_\pi} \{(P_\pi\sigma_\pi, Q_\pi\sigma_\pi)\}$
- $Tr_O((P_1, P_2), \mathcal{S}_{(P_\pi, Q_\pi)})$ is given by the set of pairs $(\xi^v_P :: P, \xi^v_Q :: Q)$ such that, letting $i, j = 1, 2$ and $i \neq j$
  - P is a $(P_i, P_r, R_P, s_P, \sigma_P\sigma_\pi)$-dual of $P_\pi\sigma_\pi$
  - Q is a $(P_j, Q_r, R_Q, s_Q, \sigma_Q\sigma_\pi)$-dual of $Q_\pi\sigma_\pi$
  - ξ^v_P and ξ^v_Q such that for all variable substitutions σ^v for both ξ^v_P and ξ^v_Q it holds that $\xi^v_P\sigma^v = \xi_P$ and $\xi^v_Q\sigma^v = \xi_Q$ where ξ_P and ξ_Q are defined as it follows
    - ξ_P is an $(\emptyset, R_P, s_P, \sigma_P\sigma_\pi)$-environment for P_r
    - ξ_Q is an $(\emptyset, R_Q, s_Q, \sigma_Q\sigma_\pi)$-environment for Q_r
    - ξ_P and ξ_Q are compatible $\square$

The observations collected below partially correspond to both Remark 42 and Remark 46. Those comments will allow us to take advantage of the main theorems of Appendix A while reasoning modulo arbitrary substitutions.

Remark 51 The following observations about Translations 3 and 4 do hold.

1. (on Trans. 3)
   Let P_π and Q_π be as in Trans. 3 and let σ_π be any substitution from names to names. By construction, for some P_r without homonymy either among bound names or among free and bound names $P_\pi\sigma_\pi \equiv_\alpha P_r$ where
   - $P_r \in \mathrm{rsp}(\mathrm{Dm}(\sigma^r(\xi^v_P\sigma^v :: P))\sigma_P, [P]^{s(\xi^v_P\sigma^v :: P)}_{\sigma'_P}\sigma_P)$
   - $\sigma_P = \sigma^r(\xi^v_P\sigma^v :: P)\sigma''_P(\sigma^c(\xi^v_P\sigma^v)\sigma^n)\sigma_\pi$
   - σ'_P, σ''_P are, respectively, the refreshment of the bound names of P and the refreshment of the names in $\mathrm{Im}(\sigma^r(\xi^v_P\sigma^v :: P))$

Similarly, for a suitable Q_r without homonymy of names and suitable substitutions σ'_Q, σ''_Q, it holds that $Q_\pi\sigma_\pi \equiv_\alpha Q_r$ where

- $Q_r \in \mathrm{rsp}(\mathrm{Dm}(\sigma^r(\xi^v_Q\sigma^v :: Q))\sigma_Q, [Q]^{s(\xi^v_Q\sigma^v::Q)}_{\sigma'_Q}\sigma_Q)$
- $\sigma_Q = \sigma^r(\xi^v_Q\sigma^v :: Q)\sigma''_Q(\sigma^c(\xi^v_Q\sigma^v)\sigma^n)\sigma_\pi$

2. (on Trans. 4)
Let $(\xi^v_P :: P, \xi^v_Q :: Q) \in Tr_O((P_1, P_2), \mathcal{S}_{(P_\pi,Q_\pi)})$ then

 - $P_\pi\sigma_\pi \equiv_\alpha P_r \in \mathrm{rsp}((\mathrm{Dm}(s_P))s_P, [P]^{s(\xi^v_P\sigma^v::P)}s_P(\sigma^c(\xi^v_P\sigma^v)\sigma^n)\sigma_\pi)$
 - $Q_\pi\sigma_\pi \equiv_\alpha Q_r \in \mathrm{rsp}((\mathrm{Dm}(s_Q))s_Q, [Q]^{s(\xi^v_Q\sigma^v::Q)}s_Q(\sigma^c(\xi^v_Q\sigma^v)\sigma^n)\sigma_\pi)$

 where

 - P_r and Q_r are without homonymy of names
 - s_P stays for the restriction substitution $\sigma^r(\xi^v_P\sigma^v :: P)$
 - s_Q stays for the restriction substitution $\sigma^r(\xi^v_Q\sigma^v :: Q)$
 - σ^v is a variable substitution for both ξ^v_P and ξ^v_Q
 - σ^n is an admissible name substitution for $\mathrm{Im}(\sigma^c(\xi^v_P\sigma^v)) \cup \mathrm{Im}(\sigma^c(\xi^v_Q\sigma^v))$

3. The above observations guarantees that the main statements about operational correspondence recalled in Appendix A (Th. 48, Th. 49, and Prop. 50) can be revisited by letting

 - P_π be substituted by $P_\pi\sigma_\pi$
 - Q_π be substituted by $Q_\pi\sigma_\pi$
 - ξ_P be substituted by $\xi^v_P\sigma^v$
 - ξ_Q be substituted by $\xi^v_Q\sigma^v$
 - P^-_π be substituted by $(P_\pi\sigma_\pi)^-$
 - etcetera. □

We can now prove the coincidence result for open semantics.

Proof of Theorem 39
For the proof of both the 'if' and the 'only if' directions, we suppose that a given relation $\mathcal{S}$ witnesses the hypothesis. Then the actual statement is shown to hold by transforming the relation $\mathcal{S}$ along the guide-lines described in Trans. 3 and Trans. 4.

(ONLY IF)
Assume that a relation $\mathcal{S} \subseteq \sim_{\eta_O}$ there exists with $(\xi^{\mathcal{L}} :: P_1, \xi^{\mathcal{L}} :: P_2) \in \mathcal{S}$. By construction $(P_1, P_2) \in Tr^v_\xi(\mathcal{S})$. We now show that $Tr^v_\xi(\mathcal{S}) \subseteq \sim_O$.
Letting $(P_\pi, Q_\pi) \in Tr_\xi((\xi^v_P\sigma^v :: P, \xi^v_Q\sigma^v :: Q))$, it is a matter of proving that for all substitutions σ_π

whenever $P_\pi\sigma_\pi \xrightarrow{\alpha_\pi} P'_\pi$ with $\mathrm{bn}(\alpha_\pi)$ fresh

then $Q_\pi\sigma_\pi \xrightarrow{\alpha_\pi} Q'_\pi$ and $(P'_\pi, Q'_\pi) \in Tr^v_\xi(\mathcal{S})$

The proof is by case analysis, and each case has the following structure.
Assume $P_\pi\sigma_\pi \xrightarrow{\alpha_\pi} P'_\pi$

$\Longrightarrow$ By Th. 48 and Remark 51 it follows that $P \overset{\langle\alpha_P, C_P\rangle}{\longrightarrow} P'$ with $[\![C_P]\!]\xi^v_P\sigma^v$ and a given relationship between α_P and α_π and between P'_π and the pair $(\xi^v_P\sigma^v, P')$. Also, if the action α_P uses a name as subject – *i.e.* if $\alpha_P \neq \tau, \tau[x/y]$ – then by Prop. 50 and Remark 51 it follows that the equivalence class of such a channel name is defined in the environment $\xi^v_P\sigma^v$. This fact partially satisfies the hypothesis which are necessary for the forthcoming application of the revisited version of Th. 49

$\Longrightarrow$ By $[\![C_P]\!]\xi^v_P\sigma^v$ and by Prop. 37 it follows that $\xi^v_P \uplus (\![C_P]\!)\xi^v_P \neq \varepsilon$

$\Longrightarrow$ $\xi^v_P :: P \overset{f}{\longrightarrow\!\!\triangleright}_{\eta_O} \xi^v_{P'} :: P'$ with some precise relationship between α_P and $f(\mathrm{Im}(\sigma^v))$ and hence between α_π and $f(\mathrm{Im}(\sigma^v))$

$\Longrightarrow$ By $(\xi^v_P :: P, \xi^v_Q :: Q) \in \mathcal{S}$ some $\xi^v_{Q'} :: Q'$ there exists such that $\xi^v_Q :: Q \overset{f}{\longrightarrow\!\!\triangleright}_{\eta_O} \xi^v_{Q'} :: Q'$ and $(\xi^v_{P'} :: P', \xi^v_{Q'} :: Q') \in \mathcal{S}$

$\Longrightarrow$ $Q \overset{\langle\alpha_Q, C_Q\rangle}{\longrightarrow} Q'$ with $\xi^v_Q \uplus (\![C_Q]\!)\xi^v_Q \neq \varepsilon$ and a sharp relationship between α_Q and $f(\mathrm{Im}(\sigma^v))$ and then, by transitivity, between α_Q and α_π

$\Longrightarrow$ By the hypothesis that σ^v is a variable substitution for ξ^v_Q and by Prop. 37 it follows that $[\![C_Q]\!]\xi^v_Q\sigma^v$

$\Longrightarrow$ By Remark 51 and Th. 49 it comes that $Q_\pi\sigma_\pi \xrightarrow{\alpha_\pi} Q'_\pi$ with a relationship between P'_π and Q'_π which guarantees that $(P'_\pi, Q'_\pi) \in Tr^v_\xi(\mathcal{S})$. In fact it holds that $(\xi^v_{P'} :: P', \xi^v_{Q'} :: Q') \in \mathcal{S}$. Moreover, if α_π is not an input action then just $(P'_\pi, Q'_\pi) \in Tr_\xi((\xi^v_{P'}\sigma^v :: P', \xi^v_{Q'}\sigma^v :: Q'))$. Otherwise $(P'_\pi, Q'_\pi) \in Tr_\xi((\xi^v_{P'}\sigma :: P', \xi^v_{Q'}\sigma :: Q'))$ for σ which coincides with σ^v on the common domain and extends it with a new component $\{c/v_{m+1}\}$. Such a component makes ground the last generated variable.

(IF)
Assume now that the relation $\mathcal{S} \subseteq \sim_O$ is given, and that $(P_1, P_2) \in \mathcal{S}$. By construction $(\xi^{\mathcal{L}} :: P_1, \xi^{\mathcal{L}} :: P_2) \in Tr_O((P_1, P_2), \mathcal{S})$. We now show that $Tr_O((P_1, P_2), \mathcal{S})$ is a strong bisimulation contained in $\sim_{\eta_O}$.

Let $(\xi^v_P :: P, \xi^v_Q :: Q) \in Tr_O((P_1, P_2), \mathcal{S}_{(P_\pi, Q_\pi)})$. Also suppose that $\xi^v_P :: P \overset{f}{\longrightarrow\!\!\triangleright}_{\eta_O} \xi^v_{P'} :: P'$ and let σ be any variable substitution for $\xi^v_{P'}$

$\Longrightarrow$ $P \overset{\langle\alpha_P, C_P\rangle}{\longrightarrow} P'$ with $\xi^v_P \uplus (\![C_P]\!)\xi^v_P \neq \varepsilon$ and a fixed relationship between α_P and $f(\mathrm{Im}(\sigma))$

$\Longrightarrow$ By Prop. 38 and Prop. 37 it holds that $[\![C_P]\!]\xi^v_P\sigma^v$ where $\sigma^v = \mathrm{frag}(\sigma, \xi^v_P)$

$\Longrightarrow$ By Remark 51 and Th. 49 it follows that $P_\pi\sigma_\pi \xrightarrow{\alpha_\pi} P'_\pi$ with a precise relationship between P'_π and $\xi^v_{P'} :: P'$ and between α_π and α_P, that is between α_π and $f(\mathrm{Im}(\sigma))$. The ground observation $f(\mathrm{Im}(\sigma))$ is in turn related to $f(\mathrm{Im}(\sigma^v))$ in the obvious way

$\Longrightarrow$ By $(P_\pi, Q_\pi) \in \mathcal{S}$ it follows that $Q_\pi\sigma_\pi \xrightarrow{\alpha_\pi} Q'_\pi$ with $(P'_\pi, Q'_\pi) \in \mathcal{S}$

$\Longrightarrow$ By Remark 51 and Th. 48 it comes that $Q \overset{\langle\alpha_Q, C_Q\rangle}{\longrightarrow} Q'$ with $[\![C_Q]\!]\xi^v_Q\sigma^v$ and a fixed relationship between Q'_π and $\xi^v_Q\sigma^v :: Q'$ and between α_Q and α_π, that is between α_Q and $f(\mathrm{Im}(\sigma^v))$

$\Longrightarrow$ By Prop. 37 it comes that $\xi^v_Q \uplus \langle\!\langle C_Q \rangle\!\rangle \xi^v_Q \neq \varepsilon$. Hence the thesis follows. In fact $\xi^v_Q :: Q \xrightarrow{h}_{\eta_O} \xi^v_{Q'} :: Q'$ and by the assumed generality of σ and by Prop. 38 it derives that the two functions f and h are point-wise equal. Moreover, by $(P'_\pi, Q'_\pi) \in \mathcal{S}$ and the assumption on σ it also holds that $(\xi^v_{P'} :: P', \xi^v_{Q'} :: Q') \in Tr_O((P_1, P_2), \mathcal{S}_{(P'_\pi, Q'_\pi)})$.
$\square$

Recent BRICS Report Series Publications

RS-97-52 **Paola Quaglia.** ***On the Finitary Characterization of π-Congruences*. December 1997. 59 pp.**

RS-97-51 **James McKinna and Robert Pollack.** ***Some Lambda Calculus and Type Theory Formalized*. December 1997. 43 pp.**

RS-97-50 **Ivan B. Damgård and Birgit Pfitzmann.** ***Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP*. December 1997. 19 pp.**

RS-97-49 **Peter D. Mosses.** ***CASL for ASF+SDF Users*. December 1997. 22 pp. Appears in *ASF+SDF'97, Proceedings of the 2nd International Workshop on the Theory and Practice of Algebraic Specifications, Electronic Workshops in Computing*, http://www.springer.co.uk/ewic/workshops/ASFSDF97. Springer-Verlag, 1997.**

RS-97-48 **Peter D. Mosses.** ***CoFI: The Common Framework Initiative for Algebraic Specification and Development*. December 1997. 24 pp. Appears in Bidoit and Dauchet, editors, *Theory and Practice of Software Development. 7th International Joint Conference CAAP/FASE*, TAPSOFT '97 Proceedings, LNCS 1214, 1997, pages 115–137.**

RS-97-47 **Anders B. Sandholm and Michael I. Schwartzbach.** ***Distributed Safety Controllers for Web Services*. December 1997. 20 pp. To appear in *European Theory and Practice of Software. 1st Joint Conference FoSSaCS/FASE/ESOP/CC/TACAS*, ETAPS '97 Proceedings, LNCS, 1998.**

RS-97-46 **Olivier Danvy and Kristoffer H. Rose.** ***Higher-Order Rewriting and Partial Evaluation*. December 1997. 20 pp. Extended version of paper to appear in *Rewriting Techniques and Applications: 9th International Conference*, RTA '98 Proceedings, LNCS, 1998.**

RS-97-45 **Uwe Nestmann.** ***What Is a 'Good' Encoding of Guarded Choice?* December 1997. 28 pp. Revised and slightly extended version of a paper published in *5th International Workshop on Expressiveness in Concurrency*, EXPRESS '97 Proceedings, volume 7 of Electronic Notes in Theoretical Computer Science, Elsevier Science Publishers.**