A Bisimulation for the Blue Calculus

Silvano Dal Zilio

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Bisimulation for the Blue Calculus

Silvano Dal-Zilio

N° 3664

Avril 1999

THÈME 1

ISSN 0249-6399

A Bisimulation for the Blue Calculus

Silvano Dal-Zilio*

Thème 1 — Réseaux et systèmes
Projet MEIJE

Rapport de recherche n° 3664 — Avril 1999 — 44 pages

Abstract: The Blue calculus is a direct extension of both the lambda and the pi calculi. In this report, we define an equivalence for this calculus based on barbed congruence, and we prove the validity of the replication laws. For example, we prove that a replicated resource, shared by many processes, can be safely copied and distributed.

Key-words: pi-calculus, blue-calculus, verification, bisimulation, barbed congruence

* Email : silvano.dalzilio@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Une bisimulation pour le Calcul Bleu

Résumé : Le calcul bleu est une extension à la fois du pi-calcul et du lambda-calcul. Dans ce rapport, nous définissons une équivalence pour ce calcul, qui se base sur la congruence à barbes, et nous démontrons la validité des lois de réplication. Par exemple, nous prouvons qu'une "ressource répliquée", partagée par plusieurs processus, peut être copiée et distribuée de façon sure.

Mots-clés : pi-calcul, calcul bleu, vérification, bisimulation, congruence à barbes

1 Introduction

The blue calculus [10], denoted $\boldsymbol{\pi}^\star$ hereafter, is a variant of the asynchronous π-calculus (denoted $\boldsymbol{\pi}$) that directly contains functions, in the sense that it contains the lambda calculus. Compared with $\boldsymbol{\pi}$ [22], it has no choice or matching operators. Moreover, we consider in this report a variant already used in [12], such that the usage of names is restricted to forbid declarations on a received channel: that is only the *output capability* of names may be transmitted. We name this condition the *locality property*, since it implies that no receiver on a given name can be dynamically created. Thus $\boldsymbol{\pi}^\star$ is better defined as a variant of the "local π-calculus" [19], a calculus considered as the basis of experimental programming languages like PICT [28][1] or JOIN [16].

Nonetheless $\boldsymbol{\pi}^\star$ is not an alleviated version of the π-calculus. The main novelty, in comparison with the calculi already mentioned, is that abstractions: $(\boldsymbol{\lambda} x)P$, are processes of the calculus, and that any process may be applied to any name. Another feature of $\boldsymbol{\pi}^\star$ is that replication is replaced by recursion. The construct $(\mathbf{def}\ u = P\ \mathbf{in}\ Q)$, called definition, is comparable to the "letrec-construct" of functional languages. Definitions have also an interpretation in terms of $\boldsymbol{\pi}$ processes. They can be viewed as the equivalent of the paradigmatic process $(\boldsymbol{\nu} u)(!u(\tilde{x}).P \mid Q)$, which is at the core of the interpretation of functions in $\boldsymbol{\pi}$. See [21] and [24] for example.

Now that we have introduced a new calculus, the problem arises to find a satisfactory behavioral equivalence between $\boldsymbol{\pi}^\star$-terms and to prove "sensible" laws on the calculus using this equivalence. A problem proved to be hard for the π-calculus. In this paper we choose *barbed congruence* [23] (denoted $\approx_b$) as our candidate equivalence and we prove the so-called *replication theorem*. We briefly define these two terms now. *Barbed congruence* is a bisimulation based behavioral equivalence that preserves a notion of observability, called barbs. More precisely, barbed congruence is the coarser congruence that verifies these properties. The *replication theorem* states that private resources can be safely duplicated. For example we prove the following equation:

$$\mathbf{def}\ u = P\ \mathbf{in}\ (Q \mid R) \approx_b (\mathbf{def}\ u = P\ \mathbf{in}\ Q) \mid (\mathbf{def}\ u = P\ \mathbf{in}\ R)$$

We refer the reader to the Lemma 6.1, where the complete set of replication laws are listed. In the π-calculus, an equivalent of this property can be found in the seminal tutorial by R. Milner [20, Sect. 5.4], where the equivalence used is strong ground congruence [22]. But this equality holds only with the side condition that the link to the resource (the channel u in this example) may not be emitted, i.e. does not appear in object position of an output. Another side condition is that u does not appear in subject position of an input. Only recently, Merro and Sangiorgi [19] prove the same equation, without the first side condition, for barbed congruence in π-calculus with the locality property.

[1] however one should note that in PICT, there are no implicit restrictions on the use of a received channel.

The choice of barbed congruence has two main motivations. First, "*it is a uniform basis to define behavioral equivalences on different process calculi*" [6]. Thus it is amenable to comparison with equivalences on $\boldsymbol{\pi}$ and the join-calculus. Second, the use of an equivalence that is a bisimulation and a congruence, eases proofs of correctness for programs transformations and interpretations (or encodings) of languages. Nonetheless there is a major drawback in the choice of barbed congruence, namely that proofs of bisimilarity results involve a quantification over all contexts. An example of this phenomenon is given in Sect. 3.1. To avoid this pitfall, we define a labeled transition system, and a corresponding labeled bisimulation that is finer than barbed congruence. Our goal: that is proving the replication theorem for the barbed congruence, is achieved by proving that this bisimulation verifies the replication theorem (see Lemma 6.1). This is not the only result of this report. The definition of the labeled transition system is interesting since it yields a better understanding of the blue calculus and its functional and "higher-order" features. In particular, it yields a better understanding on the connection between private resources in $\boldsymbol{\pi}^\star$, explicit substituions and the full λ-calculus.

structure of the paper The rest of the paper is organized as follows. In Sect. 2, we review the blue calculus augmented with extensible records and we define barbed congruence. In Sect. 4, we present the labeled transition system. In Sect. 5, we sketch the proof of validity of the "up-to context" proof technique [26] for the labeled bisimulation. The complete proof is given in Appendix A. We use this result to prove, at the same time, three properties: (i) the labeled bisimulation is a congruence, (ii) it contains the structural equivalence and (iii) it verifies the replication laws (see Sect. 6). In Sect. 7, we prove that the labeled bisimulation is a barbed bisimulation, and thus that the replication laws are valid for $\approx_b$. We conclude in Sect. 8 with some examples of application of the replication laws and some final remarks.

2 The Calculus

The blue calculus is a variant of the mini asynchronous π-calculus [7], in which functions, or abstractions in the λ-calculus terminology, are directly embedded. It has no choice, matching or guarded output operators. While $\boldsymbol{\pi}$ enforces an indirect style of programming, in the sense that one has to explicitly manage "result channels" to implement functions, $\boldsymbol{\pi}^\star$ provides a better "programming notation" for higher-order concurrency. Indeed G. Boudol shows in [10] that $\boldsymbol{\pi}$, compared to $\boldsymbol{\pi}^\star$, is a "continuation passing style calculus".

The terms of $\boldsymbol{\pi}^\star$ (extended with records) are given in Tab. 1. In the definition of the calculus, we use two disjoint categories of names: channel names, denoted $u, v, w \ldots$, and labels, denoted $k, l, m \ldots$. The semantics of the calculus is given in a chemical style [5]. For convenience, we split the description of our calculus along three syntactical categories. For each category, we examine the reduction relation $\rightarrow$, and the structural equivalence $\equiv$. The description uses the standard notion of *evaluation context*. Contexts, denoted $\mathrm{C}, \mathrm{D}, \ldots$

Table 1 Syntax of the Blue Calculus: $\boldsymbol{\pi}^\star$

$$
\begin{array}{rcll}
P, Q, R & ::= & u \mid (\boldsymbol{\lambda} u)P \mid (Pu) \mid \mathbf{def}\, u = P \,\mathbf{in}\, Q \mid & \text{(functional part)} \\
 & & \mathsf{0} \mid (P \mid Q) \mid (\boldsymbol{\nu} u)P \mid \langle u \Leftarrow P \rangle \mid & (\pi\text{-calculus part}) \\
 & & [\,] \mid [R,\, l = P\,] \mid (Pl) & \text{(record part)}
\end{array}
$$

are defined using the syntax of $\boldsymbol{\pi}^\star$-terms, plus a constant $[\_]$. We denote $\mathrm{C}[P]$ the process obtained by filling the hole in C with the process P. Evaluation contexts, $\mathrm{E}, \mathrm{F}, \ldots$, are contexts such that the hole does not occur within an abstraction, a record or a declaration:

$$
\mathrm{E} ::= \quad [\_] \mid (\mathrm{E}u) \mid (\mathrm{E} \mid P) \mid (P \mid \mathrm{E}) \mid (\boldsymbol{\nu} u)\mathrm{E} \mid \mathbf{def}\, u = P \,\mathbf{in}\, \mathrm{E} \mid (\mathrm{E}l)
$$

As usual, $\tilde{u}$ denotes the tuple of names $(u_1, \ldots, u_n)$, and the symbol ϵ is used to denote the empty tuple. We will sometimes use $(\boldsymbol{\lambda}\tilde{u})P$ instead of $(\boldsymbol{\lambda} u_1)\ldots(\boldsymbol{\lambda} u_n)P$, and $(\boldsymbol{\nu}\tilde{u})P$ instead of $(\boldsymbol{\nu} u_1)\ldots(\boldsymbol{\nu} u_n)P$. We also denote $(P\tilde{u})$ the process $(Pu_1 \ldots u_n)$. Two general rules can be given on the reduction relation:

$$
\frac{P \to P' \qquad P \equiv Q}{Q \to P'} \text{ (red equiv)}
$$

$$
\frac{P \to P' \qquad \mathrm{E} \text{ evaluation context}}{\mathrm{E}[P] \to \mathrm{E}[P']} \text{ (red context)}
$$

Concerning the structural equivalence, we assume that α-convertible processes are equal.

2.1 Description of the Functional Fragment of $\boldsymbol{\pi}^\star$

The functional part of our calculus is generated by the grammar

$$
P, Q, R \ldots ::= u \mid (\boldsymbol{\lambda} u)P \mid (Pu) \mid \mathbf{def}\, u = P \,\mathbf{in}\, Q
$$

Thus it is the "small" λ-calculus extended with the (recursive) definition operator. We use the adjective small since a process can only be applied to a name and not to another process: we say that the blue calculus is *name passing*. Nonetheless, the "high-order" λ-calculus application can be recovered using the definition

$$
(P\ Q) =_{\mathrm{def}} \mathbf{def}\, u = Q \,\mathbf{in}\, (P\ u) \qquad (u \notin \mathbf{fn}(P) \cup \mathbf{fn}(Q))
$$

There is a main difference here with respect to the original presentation of $\boldsymbol{\pi}^\star$ [10]. We have replaced "floating definitions" $\langle u = P \rangle$, equivalent to an infinite parallel composition of "one-shot" declarations, by a construction that mixes together restriction, replication and definition: $(\mathbf{def}\, u = P \,\mathbf{in}\, Q) =_{\mathrm{def}} (\boldsymbol{\nu} u)(\langle u = P \rangle \mid Q)$.

To ensure the *locality property*, i.e. that no received names can be used as subject part of a declaration, we split the category of channel names in two. We consider variables: $x, y, p, \ldots$ on one side, and references: $u, v, \ldots$ on the other side. References are bound by restrictions: $(\boldsymbol{\nu}\, u)P$, while variables are bound by λ-abstractions: $(\boldsymbol{\lambda} x)P$, and definitions: $\mathbf{def}\, p = R \,\mathbf{in}\, P$.

There is another convention. Let D be the sequence $(p_1 = P_1, \ldots, p_n = P_n)$, such that the p_i's are pairwise distinct. We call D an *environment*. To simplify notations, we denote by $(\mathbf{def}\, D \,\mathbf{in}\, P)$ the following process.

$$\mathbf{def}\, D \,\mathbf{in}\, P \;=_{\text{def}}\; \mathbf{def}\, p_1 = P_1 \,\mathbf{in}\, (\mathbf{def}\, p_2 = P_2 \,\mathbf{in}\, (\ldots \mathbf{def}\, p_n = P_n \,\mathbf{in}\, P))$$

We also allow empty definitions, such that $n = 0$. In this case, we denote D by $\emptyset$, and we regard $(\mathbf{def}\, \emptyset \,\mathbf{in}\, P)$ as identical to P.

We assume the reader is familiar with the notions of free and bound variables, α-conversion and substitution (denoted $P\{u/x\}$). We denote $\mathbf{fn}(P)$ (resp. $\mathbf{bn}(P)$) the set of free (resp. bound) names and variables in P. These sets are defined as usual considering that binders are abstractions and definitions. We denote $\mathbf{def}(D)$ the set of names defined by D, that is the set $\{p_1, \ldots, p_n\}$ with the notation used above.

For the functional subset of $\boldsymbol{\pi}^\star$, the axioms defining structural equivalence are [9]:

$$\begin{aligned}
(\mathbf{def}\, p = R \,\mathbf{in}\, P) \mid Q &\equiv \mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q) && (p \notin \mathbf{fn}(Q)) \\
\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu}\, u)P) &\equiv (\boldsymbol{\nu}\, u)(\mathbf{def}\, p = R \,\mathbf{in}\, P) && (u \notin \mathbf{fn}(R)) \\
(\mathbf{def}\, p = R \,\mathbf{in}\, P)u &\equiv \mathbf{def}\, p = R \,\mathbf{in}\, (Pu) && (u \neq p) \\
((\boldsymbol{\nu}\, u)P)v &\equiv (\boldsymbol{\nu}\, u)(Pv) && (u \neq v)
\end{aligned}$$

On the functional fragment, the reduction is defined by the "small" β reduction and the reduction rule for definitions:

$$\frac{}{((\boldsymbol{\lambda} x)P)u \;\rightarrow\; P\{u/x\}}\ (\text{red beta})$$

$$\frac{(\{p\} \cup \mathbf{fn}(R)) \cap \mathbf{bn}(\mathrm{E}) = \emptyset}{\mathbf{def}\, p = R \,\mathbf{in}\, \mathrm{E}[p] \;\rightarrow\; \mathbf{def}\, p = R \,\mathbf{in}\, \mathrm{E}[R]}\ (\text{red def})$$

2.2 Description of the π-calculus Fragment of $\boldsymbol{\pi}^\star$

In this section, we consider the operators directly derived from the π-calculus, that is

$$P, Q, R ::= \mathtt{0} \;\big|\; \langle u \Leftarrow P \rangle \;\big|\; (P \mid P) \;\big|\; (\boldsymbol{\nu}\, u)P$$

The new construct introduced here is the declaration: $\langle u \Leftarrow P \rangle$, that can be interpreted as a resource, located at u, accessible only once. Roughly speaking, the declaration $\langle u \Leftarrow (\boldsymbol{\lambda} x)P \rangle$ is the equivalent of the π-calculus input guard: $u(x).P$, while the application $(uv_1 \dots v_n)$ is the equivalent of the π-calculus output: $\bar{u}\langle \tilde{v} \rangle$. The declaration is useful to model processes with a mutable state.

As stated in the previous section, and according to the conclusion of the original presentation of $\boldsymbol{\pi}^\star$ [10], the use of references is restricted in our calculus: although they can appear under a λ-abstraction, they cannot be abstracted upon. For example, $(\boldsymbol{\lambda} u)\langle u \Leftarrow P \rangle$ is not a valid process. This restriction ensures the locality property, but it also ensures that a name cannot be used both in a definition and in a declaration, like in $(\mathbf{def}\, u = R \,\mathbf{in}\, (\langle u \Leftarrow P \rangle \mid Q))$ for example. Structural rules for the π-calculus fragment are the "scope extrusion" rule of $\boldsymbol{\pi}$, the usual rules for the commutative monoid $(P, \mid, \mathtt{0})$ and rules managing application:

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad P \mid Q \equiv Q \mid P$$

$$\mathtt{0} \mid P \equiv P \qquad \mathtt{0}v \equiv \mathtt{0}$$

$$((\boldsymbol{\nu} u)P) \mid Q \equiv (\boldsymbol{\nu} u)(P \mid Q) \quad (u \notin \mathbf{fn}(Q))$$

$$(\boldsymbol{\nu} u)(\boldsymbol{\nu} v)P \equiv (\boldsymbol{\nu} v)(\boldsymbol{\nu} u)P$$

$$(P \mid Q)u \equiv (Pu) \mid (Qu) \qquad \langle u \Leftarrow P \rangle v \equiv \langle u \Leftarrow P \rangle$$

Concerning the reduction relation, there is a communication rule that consumes a declaration and fetches the resource at the "location" of the message:

$$\frac{}{\langle u \Leftarrow P \rangle \mid (uv_1 \dots v_n) \rightarrow (Pv_1 \dots v_n)} \text{ (red decl)}$$

2.3 Description of the Record Fragment of $\boldsymbol{\pi}^\star$

Records are incrementally built from the empty record $[\,]$, using the extend/update operation $[\,Q\,,\; l = P\,]$ that adds/overrides the field l with value P, to the record Q. Intuitively, a record is a function from a finite set of labels to processes, and selection is function application. This intuition is strengthened by the "dot-less" notation for selection: we use the notation (Pl), instead of the more familiar notation $(P \cdot l)$.

It would be convenient to add a system of kinds to tag the use of names in $\boldsymbol{\pi}^\star$ (together with a notion of well-formed terms), in order to formalize the distinction made between labels, variables and references. We believe that the details of this system are clear to the reader, and we omit to define it here. In the remainder of this paper, we assume that names are always used correctly, and we use the letters $a, b, \dots$ to denote any names i.e., a variable, a reference or a label. Therefore, the term (Pa) denotes an application or a record selection.

The structural rules for record selection (Pl) are the same as the ones for application. For example, we have the following equalities.

$$(P \mid Q)l \;\equiv\; (Pl) \mid (Ql) \qquad\qquad \langle u \Leftarrow P\rangle l \;\equiv\; \langle u \Leftarrow P\rangle$$

We use $[\, l_1 = P_1, \ldots, l_n = P_n \,]$, instead of $[\, [\, [\,] \,,\, l_1 = P_1 \,] \,, \ldots l_n = P_n \,]$, whenever the l_i's are distinct. There are two reduction rules for records:

$$\frac{}{([\, Q \,,\, l = P \,]l) \;\to\; P}\ \text{(red sel)} \qquad \frac{k \neq l}{([\, Q \,,\, l = P \,]k) \;\to\; (Qk)}\ \text{(red over)}$$

This formalization of records is closely related to the one proposed by Wand [29], since there is a single operation to either modify or add a field to a record.

3 Operational Equivalence

In the definition of a calculus, the choice of the equivalence relation used to reason about equality between terms is a difficult task. In this paper, contrary to previous presentations of the blue calculus, we choose for candidate a behavioral equivalence that is also a bisimulation. This relation, called barbed congruence ($\approx_b$), is the biggest bisimulation that preserves simple observations called barbs and that is a congruence [18]. For people familiar with bisimulation for $\boldsymbol{\pi}$, it is a variant of the weak barbed congruence [23]. Another related equivalence is the one defined for the join-calculus [6]. In particular, the reader should note that, as in JOIN, we consider the coarsest barbed bisimulation that is a congruence, instead of considering the closure under every context of barbed bisimulation.

Nonetheless, the definition of this equivalence does not directly follow from its $\boldsymbol{\pi}$ counterpart. Indeed, whereas the observable behaviors considered in CCS and $\boldsymbol{\pi}$ are the visible outputs (a choice equivalent to observing free names in head position in $\boldsymbol{\pi}^\star$), we choose instead to observe the presence of values, following the intuition that the basic values are abstractions. Note that this choice of "observables" agrees with the definition of Morris-style equivalences for the λ-calculus [4, ex. 16.5.5]. Since we are in a concurrent calculus, we also consider that a value in parallel with an arbitrary process is a value.

Definition 3.1 (Barbs) A barb, we say also a *value*, is a process generated by the following grammar.

$$V ::= (\boldsymbol{\lambda} x)P \;\big|\; [\, Q \,,\, l = P \,] \;\big|\; (V \mid P) \;\big|\; (P \mid V) \;\big|\; (\boldsymbol{\nu} u)V \;\big|\; \mathbf{def}\, p = R \,\mathbf{in}\, V$$

Values are the "observable terms" of $\boldsymbol{\pi}^\star$. We denote $P \downarrow$ the fact that P is a value. The weak version of barbs used in the definition of $\approx_b$ is $P \Downarrow$, such that $P \Downarrow$ if there exists a value V such that $P \to^* V$.

With this definition, a value is a process that can reduce when it is applied to a name, or when it is selected. In particular, the empty record [ ] is not a value. The definition of "being a value" extend to contexts in a straightforward way. We say that $\mathrm{C}\downarrow$ if and only if $\mathrm{C}[\mathtt{0}]\downarrow$. It is easy to see that $\mathrm{C}\downarrow$ implies that $\mathrm{C}[P]\downarrow$ for all processes P. Before giving the definition of barbed congruence, we define the notion of *compositional* relation.

Definition 3.2 (Compositional) The relation $\mathscr{D}$ is compositional, or closed under every context, if for all context C and couple $(P,Q)\in\mathscr{D}$, we have: $(\mathrm{C}[P],\mathrm{C}[Q])\in\mathscr{D}$.

Barbed congruence is the biggest compositional bisimulation that preserves barbs.

Definition 3.3 (Barbed Congruence) A relation $\mathscr{D}$ is a *weak barbed simulation* if for every couple $(P,Q)\in\mathscr{D}$, we have:

(i) $P\rightarrow P'$ implies $Q\rightarrow^{*}Q'$ and $(P',Q')\in S$;

(ii) $P\downarrow$ implies $Q\Downarrow$.

The relation $\mathscr{D}$ is a weak barbed bisimulation, if $\mathscr{D}$ and $\mathscr{D}^{-1}$ are weak barbed simulations. The processes P and Q are observationally equivalent, written $P\approx_b Q$, if and only if $(P,Q)\in\mathscr{D}$ for some weak barbed bisimulation $\mathscr{D}$ that is compositional.

3.1 Properties of Barbed Congruence

In this section, we prove that beta conversion is a "valid law" of barbed congruence: this is formally stated in Lemma 3.3. This example is interesting since it shows how quantification over any context appears in direct proofs of congruence properties. Before proving Lemma 3.3, we first give some intermediary results. In Lemma 3.1, we study the interaction between contexts and observations. In Lemma 3.2, we give a property that allows to simplify the reasoning on the reductions that a redex can perform under a context.

Lemma 3.1 (Barbs and Contexts) *If the process* $\mathrm{C}[P]$ *is a value, the there are two cases. Either* (i) *the context* C *contains a value i.e.,* $\mathrm{C}\downarrow$*; either* (ii) *the context* C *is an evaluation context and* P *is a value.*

Proof By induction on the size of C. □

We define the *depth of a context* C, denoted $h(\mathrm{C})$, as the number of abstractions and declarations that one goes through to reach the hole. We also suppose that the depth of the context $(\mathbf{def}\,p=\mathrm{C}\,\mathbf{in}\,P)$, is $h(\mathrm{C})+1$. Clearly, the depth is preserved by structural manipulation and it equals zero if and only if C is an evaluation context.

Lemma 3.2 (Redex and Contexts) *If* $\mathrm{C}[((\boldsymbol{\lambda}x)P)a] \to Q$, *then there are two possibilities*

(i) ***the reduction comes from the context**: there exists a context* D, *such that* $Q \equiv \mathrm{D}[((\boldsymbol{\lambda}x)P\ a)\{b/y\}]$, *where* $\{b/y\}$ *may be the identity substitution (i.e.* $b = y$*), and where* D *may have two holes, i.e. two occurrences of* $[\_]$. *Moreover, for all processes* R, *we have* $\mathrm{C}[R] \to \mathrm{D}[R\{b/y\}]$, *and* $h(\mathrm{D}) \leq h(\mathrm{C})$;

(ii) ***the reduction comes from the redex**:* $Q \equiv \mathrm{C}[P\{a/x\}]$.

Proof We only give the sketch of this proof here. As stated in [10], every process is structurally equivalent to a normal form:

$$(\boldsymbol{\nu}\,\tilde{u})(\mathbf{def}\ D\ \mathbf{in}\ (V_1 \mid \cdots \mid V_m \mid M_1 \mid \cdots \mid M_s \mid \langle v_1 \Leftarrow R_1\rangle \mid \cdots \mid \langle v_r \Leftarrow R_r\rangle)) \qquad (1)$$

where the V_i's are abstractions or records: $(\boldsymbol{\lambda}x)P$ or $[Q\,,\, l = P]$, and the M_i's are applications or selections: $(P\ a)$, and where P and Q are also in normal form. We can always suppose that structural equivalence is used only at the beginning and at the end of a reduction. That is, for every reduction $P \to P'$, we can always suppose that there exist two processes Q and Q' such that: $P \equiv Q$, and $Q \to Q'$, and $Q' \equiv P'$, and such that the inference of $Q \to Q'$ does not use rule (red equiv). Moreover, we can suppose that Q is in the normal form defined by equation (1).

In our case, we can suppose that the context C is in normal form. The proof is made by induction on the size of C. We only study two cases. The first is when $\mathrm{C} =_{\mathrm{def}} (\mathrm{D}\ b)$. If $\mathrm{C}[((\boldsymbol{\lambda}x)P)a] \to Q$, there are two possibilities:

- **case** $\mathrm{D}[((\boldsymbol{\lambda}x)P)a]$ **reduces:** we use the induction hypothesis;
- **case there is a** β**-reduction with argument** b**:** and the redex is in D. If the hole $[\_]$, is not in the redex, then we are in case (i), with $b = y$. If $[\_]$ is in the redex, since D is in normal form, we have $\mathrm{D}[((\boldsymbol{\lambda}x)P)a] \equiv ((\boldsymbol{\lambda}y)\mathrm{B}[((\boldsymbol{\lambda}x)P)a])$, and therefore $Q \equiv \mathrm{B}[((\boldsymbol{\lambda}x)P)a]\{b/y\}$. This is case ($i$).

The second interesting case is when $\mathrm{C} =_{\mathrm{def}} (\mathbf{def}\ p = \mathrm{D}\ \mathbf{in}\ \mathrm{E}[p])$, and when the reduction is $\mathrm{C}[((\boldsymbol{\lambda}x)P)a] \to (\mathbf{def}\ p = \mathrm{D}[((\boldsymbol{\lambda}x)P)a]\ \mathbf{in}\ \mathrm{E}[\mathrm{D}[((\boldsymbol{\lambda}x)P)a]])$. This is case ($i$), where $b = y$, and the resulting context has two holes. Note that the (red def) rule is the only reduction rule that can duplicate the hole in a context. □

With these two lemmas, we can prove the principal result of this section.

Lemma 3.3 (Beta Conversion) *The rule for beta reduction is an algebraic law of our system:* $((\boldsymbol{\lambda}x)P\ a) \approx_b P\{a/x\}$.

Proof (Lemma 3.3) Let $\mathscr{D}$ be the relation:

$$\mathscr{D} \;=\; \mathcal{I}d \cup \big\{(\mathrm{C}[(\boldsymbol{\lambda}x)P\ a], \mathrm{C}[P\{a/x\}]) \;\big|\; \text{for all}\ \ \mathrm{C}, P \text{ and } a\big\}$$

Note that this relation is clearly compositional. We prove that $\mathscr{D}$ and $\mathscr{D}^{-1}$ are weak barbed simulations. Let P_1 and P_2 be the processes defined by: $P_1 =_{\mathrm{def}} \mathrm{C}[(\boldsymbol{\lambda}x)P\ a]$ and

$P_2 =_{\text{def}} \mathrm{C}[P\{a/x\}]$. Suppose P_1 reduces in P_1', we prove that there exists a process P_2' such that $P_2 \rightarrow P_2'$ and $P_1' \mathscr{D} P_2'$. Using Lemma 3.2, we can exhibit two cases. Either $P_1' = P_2$, and we use the fact that $P_2 \mathscr{D} P_2$, or $P_1' = \mathrm{D}[((\boldsymbol{\lambda} x)P\ a)\{b/y\}]$. In this case, P_2 can reduce to $P_2' = \mathrm{D}[P\{a/x\}\{b/y\}]$ and (P_1', P_2') is in $\mathscr{D}$. The proof is similar if we suppose that P_2 reduces. We prove also that $(P_1, P_2) \in \mathscr{D}$ and $P_1 \downarrow$ (respectively $P_2 \downarrow$) implies $P_2 \Downarrow$ (resp. $P_1 \Downarrow$):

- suppose $P_1 \downarrow$, since $((\boldsymbol{\lambda} x)P\ a)$ is not a value, $\mathrm{C}[R] \downarrow$ for all R and therefore $P_2 \downarrow$;
- suppose $P_2 \downarrow$. There are two cases. (i) for all R, $\mathrm{C}[R] \downarrow$, and, in particular, $P_1 \downarrow$; (ii) C is an evaluation context and $P\{a/x\} \downarrow$. In this case, $P_1 \rightarrow P_2 \downarrow$ and thus $P_1 \Downarrow$.

Therefore, $\mathscr{D}$ and $\mathscr{D}^{-1}$ are weak barbed bisimulations, and thus: $(P, Q) \in \mathscr{D} \Rightarrow P \approx_b Q$. The expected result follows by choosing $\mathrm{C} = [\_]$. □

With a similar proof, it is possible to prove a similar result for definition fetching, see Proposition 3.1 below. The proof is omitted in this report.

Proposition 3.1 (Definition Fetching) $\mathbf{def}\, p = R \,\mathbf{in}\, p \approx_b \mathbf{def}\, p = R \,\mathbf{in}\, R$

4 Labeled Transition System with Definitions

In this section, we define a labeled transition system for the blue calculus. We recall that (Pa) denotes either an application (Pu), or a selection (Pl). We define four different kinds of actions in the labeled transition system:

$$\mu ::= \tau \mid \lambda\, a \mid \mathbf{out}\, u.(\tilde{v}; \tilde{a}) \mid \mathbf{in}\, u.(\tilde{b}; \tilde{a})$$

These four actions are respectively:

silent action (τ): which corresponds to internal reduction. Like in $\boldsymbol{\pi}$, two processes in parallel can synchronize during a communication. This is rule (par com) of the labeled transition system (l.t.s.). We also have to consider two other reduction patterns though. One for application, corresponding to the reduction rule (red beta) and to the rule (app com) of the l.t.s.. The other for definition, that corresponds to the reduction rule (red def) and to the rule (def com) of the l.t.s.;

lambda action ($\lambda\, a$): which is an action performed by a value, see rules (lambda) below, and which corresponds to a λ-abstraction or a record;

in action ($\mathbf{in}\, u.(\tilde{b}; \tilde{a})$): which is an input of the names $\tilde{a}$ on the channel u. The tuple $\tilde{b}$, is used to remember the applications/selections than one goes through to reach the declaration;

out action ($\mathbf{out}\, u.(\tilde{v};\tilde{a})$): which is an output of the names in $\tilde{a}$, on the channel u. In an output action: $P \overset{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D; P')$, the tuple $\tilde{v}$ is the set of "opened" restrictions i.e., the set of references in $\tilde{a}$ that were restricted and opened (scope extrusion). In the same transition, the environment D corresponds to the opened definitions.

To distinguish the different actions, we will often use the "kind of the action μ", denoted $\kappa(\mu)$, such that $\kappa(\mu) \in \{\tau, \lambda, \mathbf{out}, \mathbf{in}\}$. In our transition systems, defined in Sect. 4.1, the transitions can be schematically divided in two categories. The first category correspond to the transitions $P \overset{\mu}{\longmapsto} P'$, such that $\kappa(\mu) \in \{\tau, \lambda, \mathbf{in}\}$. The second category correspond to emissions: $P \overset{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D; P')$. In this transition, D denotes the environment $(p_1 = R_1, \ldots, p_n = R_n)$, and P' is a process.

Remark In the definition of the output action $\mathbf{out}\, u.(\tilde{v};\tilde{a})$, we implicitly consider that the tuple of restricted names $\tilde{v}$ is not ordered: we consider that it is a "set". This is important to prove properties such as: $(\boldsymbol{\nu} u)(\boldsymbol{\nu} v)P \sim_d (\boldsymbol{\nu} v)(\boldsymbol{\nu} u)P$. □

To simplify the presentation of the reduction rules and the proofs, we omit to define the symmetric of the rules concerning parallel composition. We also do not take α-conversion into account. More formally, we consider that bound names are always unique and different from each other: it is sufficient to implicitly take an underlying representation of names based on De Bruijn indices [15]. Nonetheless, we will not explicitly use De Bruijn indices. This solution is already not tractable for hand made proofs in the λ-calculus, and it is really painful in the π-calculus: see the work of D. Hirschkoff [17], on computer-aided bisimulation proofs for $\boldsymbol{\pi}$.

The transition rules presented in this section are non-standard, even if rules concerning lambda actions are put aside. Part of their complexity result from the "polyadic" nature of the calculus: more than one name may be exchanged in a communication. Another source of complexity follows from the higher-order presentation of the output actions. In a transition $P \overset{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D; P')$, for example, the set D is somehow "part" of the label. Therefore we are in a situation comparable to transition systems where processes appear in labels [25]. Our l.t.s. can be distinguished from transition system for the π-calculus for other reasons. Indeed, in our l.t.s., τ-transitions are not equivalent to reductions obtained by the relation $\rightarrow$. For example we have:

$$\begin{array}{rcl} \langle u \Leftarrow Q \rangle \mid \mathbf{def}\, p = R \,\mathbf{in}\, (u\ p \mid P) & \rightarrow & \mathbf{def}\, p = R \,\mathbf{in}\, ((Q\ p) \mid P) \\ & \overset{\tau}{\longmapsto} \equiv & \mathbf{def}\, p = R \,\mathbf{in}\, ((Q\ p) \mid (\mathbf{def}\, p = R \,\mathbf{in}\, P)) \end{array}$$

The source of this difference lies in the transition rules for **out** actions and, more particularly in the rule (def open), that concerns the definition operator. Intuitively, the point is that a private resource may not be communicated outside it scope. Instead, as it is done in rule (def open), we create a copy of the resource using a fresh name.

4.1 Definition of the Labeled Transition System

Axioms

$$(\boldsymbol{\lambda} x)P \overset{\lambda\, u}{\longmapsto} P\{u/x\} \quad \text{(lambda)} \qquad a \overset{\mathbf{out}\, u.(\epsilon;\, \epsilon)}{\longmapsto} (\emptyset; \mathsf{0}) \quad \text{(out)}$$

$$[\, R\,,\, l = N\,] \overset{\lambda\, l}{\longmapsto} N \quad \text{(lambda)} \qquad [\, R\,,\, l = N\,] \overset{\lambda\, k}{\longmapsto} (Rk) \ (\text{if } k \neq l) \quad \text{(lambda)}$$

$$\langle u \Leftarrow P \rangle \overset{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} (P\ \tilde{a}) \quad \text{(decl)}$$

Rules for tau, lambda and in actions

$$\frac{P \overset{\lambda\, a}{\longmapsto} P'}{P \mid Q \overset{\lambda\, a}{\longmapsto} P' \mid (Q\ a)} \ \text{(par lambda)} \qquad \frac{P \overset{\lambda\, a}{\longmapsto} P'}{P\ a \overset{\tau}{\longmapsto} P'} \ \text{(app com)}$$

$$\frac{P \overset{\tau}{\longmapsto} P'}{P \mid Q \overset{\tau}{\longmapsto} P' \mid Q} \ \text{(par tau)} \qquad \frac{P \overset{\tau}{\longmapsto} P'}{P\ a \overset{\tau}{\longmapsto} P'\ a} \ \text{(app tau)}$$

$$\frac{P \overset{\mu}{\longmapsto} P' \qquad (\kappa(\mu) \in \{\tau, \lambda, \mathbf{in}\}\ \&\ u \notin \mathbf{fn}(\mu))}{(\boldsymbol{\nu} u)P \overset{\mu}{\longmapsto} (\boldsymbol{\nu} u)P'} \ \text{(new mu)}$$

$$\frac{P \overset{\mu}{\longmapsto} P' \qquad (\kappa(\mu) \in \{\tau, \lambda, \mathbf{in}\}\ \&\ p \notin \mathbf{fn}(\mu))}{\mathbf{def}\, p = R \,\mathbf{in}\, P \overset{\mu}{\longmapsto} \mathbf{def}\, p = R \,\mathbf{in}\, P'} \ \text{(def mu)}$$

Rules for in actions

$$\frac{P \overset{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} P'}{P \mid Q \overset{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} P' \mid (Q\ \tilde{b})} \ \text{(par in)} \qquad \frac{P \overset{\mathbf{in}\, u.((c,\, \tilde{b});\, \tilde{a})}{\longmapsto} P'}{(P\ c) \overset{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} P'} \ \text{(app in)}$$

Rules for out actions

$$\frac{P \overset{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P') \qquad (\tilde{v}, \mathbf{def}(D)) \cap \mathbf{fn}(Q) = \emptyset}{P \mid Q \overset{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; (P' \mid Q))} \ \text{(par out)}$$

$$\frac{P \overset{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P') \qquad c \notin (\tilde{v}, \mathbf{def}(D))}{(P\ c) \overset{\mathbf{out}\, u.(\tilde{v};\, (\tilde{a},\, c))}{\longmapsto} (D; (P'\ c))} \ \text{(app out)}$$

$$\frac{P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;P') \qquad (v \notin u,\tilde{v},\tilde{a},\mathbf{fn}(D))}{(\boldsymbol{\nu} v)P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;(\boldsymbol{\nu} v)P')} \text{ (new out)}$$

$$\frac{P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;P') \qquad (u \neq v)}{(\boldsymbol{\nu} v)P \overset{\mathbf{out}\ u.((v,\tilde{v});\tilde{a})}{\longmapsto} (D;P')} \text{ (new open)}$$

$$\frac{P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;P') \qquad (p \notin u,\tilde{v},\tilde{a},\mathbf{fn}(D))}{\mathbf{def}\ p = R\ \mathbf{in}\ P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;\mathbf{def}\ p = R\ \mathbf{in}\ P')} \text{ (def out)}$$

$$\frac{P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;P') \qquad (p \notin u,\tilde{v},\mathbf{def}(D))}{\mathbf{def}\ p = R\ \mathbf{in}\ P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} ((p = R,D);\mathbf{def}\ p = R\ \mathbf{in}\ P')} \text{ (def open)}$$

Communication rules

$$\frac{P \overset{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D;P') \qquad Q \overset{\mathbf{in}\ u.(\epsilon;\tilde{a})}{\longmapsto} Q'}{P \mid Q \overset{\tau}{\longmapsto} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (P' \mid Q'))} \text{ (par com)}$$

$$\frac{P \overset{\mathbf{out}\ p.(\tilde{v};\tilde{a})}{\longmapsto} (D;P')}{\mathbf{def}\ p = R\ \mathbf{in}\ P \overset{\tau}{\longmapsto} \mathbf{def}\ p = R\ \mathbf{in}\ (\boldsymbol{\nu}\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (R\ \tilde{a} \mid P'))} \text{ (def com)}$$

4.2 Definition of the Def-bisimulation

Using the labeled transition system for the blue calculus, it is possible to define a labeled bisimulation, denoted $\sim_d$, say def-bisimulation. The operator $|\tilde{a}|$ denotes the size (number of elements) of the tuple $\tilde{a}$.

Definition 4.1 (Similar tuples) For every relation $\mathscr{D}$, we denote $(\mathbf{def}\ D\ \mathbf{in}\ \tilde{a})\ \mathscr{D}\ (\mathbf{def}\ D'\ \mathbf{in}\ \tilde{b})$ the fact that the two following properties hold.

(*i*) $\tilde{a}$ and $\tilde{b}$ have the same size, say n. That is: $|\tilde{a}| = |\tilde{b}| = n$;

(*ii*) for all $i \in [1..n]$, the names a_i and b_i link to related resources i.e., $(\mathbf{def}\ D\ \mathbf{in}\ a_i)\ \mathscr{D}\ (\mathbf{def}\ D'\ \mathbf{in}\ b_i)$.

We define, in Definition 4.2, the relation of def-bisimulation, that equates processes obtained by « replication of the definition operators », like for example: $(\mathbf{def}\ p = R\ \mathbf{in}\ (P \mid Q))$

and $(\mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P) \mid Q))$. In Sect. 5 and 6, we prove that this equivalence is included in barbed congruence.

Definition 4.2 (Def-bisimulation) The relation $\mathscr{D}$ is a strong d-simulation, if for all $(P,Q) \in \mathscr{D}$ the three following properties hold:

(i) if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \notin \{\mathbf{out}\}$, then there exists a process Q' such that $Q \stackrel{\mu}{\longmapsto} Q'$ and $P' \,\mathscr{D}\, Q'$;

(ii) if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{b})}{\longmapsto} (D \mid P')$, then there exists a process Q' and an environment D' such that $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{c})}{\longmapsto} (D';Q')$, and $P' \,\mathscr{D}\, Q'$, and $(\mathbf{def}\, D \,\mathbf{in}\, \tilde{b}) \,\mathscr{D}\, (\mathbf{def}\, D' \,\mathbf{in}\, \tilde{c})$;

(iii) for all substitutions σ, of names for variables, we have $P\sigma \,\mathscr{D}\, Q\sigma$.

A relation that verifies properties (i) and (ii) is called a *ground simulation*. We say that $\mathscr{D}$ is a d-bisimulation, if both $\mathscr{D}$ and $\mathscr{D}^{-1}$ are d-simulations. The strong def-bisimulation relation $\sim_d$, is defined by: $P \sim_d Q$, if there exist a strong d-bisimulation $\mathscr{D}$, such that $P \,\mathscr{D}\, Q$.

It is easy to prove that the def-bisimulation does not characterize barbed congruence. Indeed, an important algebraic law of the asynchronous π-calculus [18], that is false in the synchronous one, is $u(x).\bar{u}x \approx \mathtt{0}$. In the blue calculus we can prove a similar law, that is $\langle u \Leftarrow u \rangle \approx_b \mathtt{0}$, but obviously $\langle u \Leftarrow u \rangle \not\sim_d \mathtt{0}$[2], since the former can do an input action and the latter is inert. Therefore, a first "improvement" could be to define an "asynchronous bisimulation", as was done for $\boldsymbol{\pi}$ in [3].

With an asynchronous bisimulation, the equality $\langle u \Leftarrow u \rangle \approx \mathtt{0}$, becomes an algebraic law of the system. Nonetheless, we believe that this equality is of little use, especially in the cases where one wants to prove properties of transformations or interpretations, as it is the case in Sect. 6.1. Indeed, the process $\langle u \Leftarrow u \rangle$ creates a "silly link" between a name and itself: this process never appears in practical examples. At the opposite, we can prove that the process $(\mathbf{def}\, p = u \,\mathbf{in}\, P)$: the "equator" of p and u in P, is such that $(\mathbf{def}\, p = u \,\mathbf{in}\, P) \approx_d P\{u/p\}$.

There is another example of processes that are not def-bisimilar, while they are behaviorally equivalent. Let P and Q be the processes

$$\begin{array}{lll} P & =_{\mathrm{def}} & (\boldsymbol{\nu}\, d)(\mathbf{def}\, p = (\boldsymbol{\nu}\, c)\langle c = d \rangle \,\mathbf{in}\, (up \mid \langle d \Leftarrow R_1 \rangle)) \\ Q & =_{\mathrm{def}} & (\boldsymbol{\nu}\, d)(\mathbf{def}\, p = \mathtt{0} \,\mathbf{in}\, (up \mid \langle d \Leftarrow R_2 \rangle)) \end{array}$$

Since "nobody can communicate" with the name c in $(\boldsymbol{\nu}\, c)\langle c = d \rangle$, it is obvious that $(\boldsymbol{\nu}\, c)\langle c = d \rangle \approx_b \mathtt{0}$. Therefore, since the name d remains private, nobody can communicate with the declaration $\langle d \Leftarrow R \rangle$. Nonetheless the process P can make the action $\mathbf{out}\, u.(d;p)$,

[2] and even $\langle u \Leftarrow u \rangle \not\approx_d \mathtt{0}$, where $\approx_d$ is the weak def-bisimulation.

while Q can only make the action $\mathbf{out}\, u.(\epsilon; p)$. That is, only in the first transition, we can observe that a private name is extruded.

This last example is typical of what happens in higher-order process calculi, where processes, and not names, are exchanged during a communication. For example, the process P can be interpreted as $(\boldsymbol{\nu}\, d)(\bar{u}\langle(\boldsymbol{\nu}\, c)(c(x).\bar{d}\langle\rangle)\rangle \mid \ldots)$, and Q as $(\boldsymbol{\nu}\, d)(\bar{u}\langle \mathtt{0}\rangle \mid \ldots)$. Like the problem concerning asynchrony and $\sim_d$, we believe that this category of non-bisimilar, yet congruent processes, is not important.

5 Bisimulation Up-To

In this section, we study the "up-to context" proof technique [26], and we prove that a bisimulation up-to context is a def-bisimulation. The complete proof can be found in Appendix A. One of the interests of this proof technique, is that it allows us to avoid a direct proof that $\sim_d$ is a congruence. Other properties that follow from the proof of Th. 5.1 are listed in Sect. 6.

We start by defining the notion of up-to contexts simulation. For each relation $\mathscr{D}$, we define $\mathscr{D}_{\mathrm{C}}$ as the smallest compositional relation that contains the transitive and reflexive closure of $\mathscr{D}$, that is:

- for all process P, we have $P\, \mathscr{D}_{\mathrm{C}}\, P$;
- if $P\, \mathscr{D}\, Q$, then $P\, \mathscr{D}_{\mathrm{C}}\, Q$;
- if $P\, \mathscr{D}_{\mathrm{C}}\, Q$ and $Q\, \mathscr{D}_{\mathrm{C}}\, R$, then $P\, \mathscr{D}_{\mathrm{C}}\, R$;
- if $P\, \mathscr{D}_{\mathrm{C}}\, Q$, then: $(\boldsymbol{\nu}\, u)P\, \mathscr{D}_{\mathrm{C}}\, (\boldsymbol{\nu}\, u)Q$, and $(Pa)\, \mathscr{D}_{\mathrm{C}}\, (Qa)$, and $(\boldsymbol{\lambda} x)P\, \mathscr{D}_{\mathrm{C}}\, (\boldsymbol{\lambda} x)Q$, and $\langle u \Leftarrow P\rangle\, \mathscr{D}_{\mathrm{C}}\, \langle u \Leftarrow Q\rangle$;
- if $P\, \mathscr{D}_{\mathrm{C}}\, Q$ and $R\, \mathscr{D}_{\mathrm{C}}\, S$, then: $(P \mid R)\, \mathscr{D}_{\mathrm{C}}\, (Q \mid S)$, and $(\mathbf{def}\, p = P\, \mathbf{in}\, R)\, \mathscr{D}_{\mathrm{C}}\, (\mathbf{def}\, p = Q\, \mathbf{in}\, S)$, and $[\, P\, ,\, l = R\,]\, \mathscr{D}_{\mathrm{C}}\, [\, Q\, ,\, l = S\,]$.

Definition 5.1 (Ground Simulation Up-To Contexts) The relation $\mathscr{D}$ is a ground simulation *up-to contexts*, if for all $(P, Q) \in \mathscr{D}$ we have:

(i) if $P \overset{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q' such that $Q \overset{\mu}{\longmapsto} Q'$ and $P'\, \mathscr{D}_{\mathrm{C}}\, Q'$;

(ii) if $P \overset{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, then there exists a process Q' and an environment D' such that $Q \overset{\mathbf{out}\, u.(\tilde{v};\tilde{b})}{\longmapsto} (D' \mid Q')$ with $P'\, \mathscr{D}_{\mathrm{C}}\, Q'$ and $(\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_{\mathrm{C}}\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{b})$.

Likewise, we can define a notion of bisimulation up-to structural equivalence. We denote $\mathscr{D}_{\equiv}$ the relation defined by: $\mathscr{D}_{\equiv} =_{\mathrm{def}} \{(P, Q) \mid \exists (P', Q') \in \mathscr{D}.P \equiv P' \;\&\; Q \equiv Q'\}$.

Definition 5.2 (Ground Simulation Up-To $\equiv$) The relation $\mathscr{D}$ is a ground simulation *up-to* $\equiv$, if for all $(P,Q) \in \mathscr{D}$ we have:

(*i*) if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q' such that $Q \stackrel{\mu}{\longmapsto} Q'$ and $P' \mathscr{D}_{\equiv} Q'$;

(*ii*) if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, then there exists a process Q' and an environment D' such that $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{b})}{\longmapsto} (D' \mid Q')$, with $P' \mathscr{D}_{\equiv} Q'$ and $(\mathbf{def}\, D \,\mathbf{in}\, \tilde{a}) \mathscr{D}_{\equiv} (\mathbf{def}\, D' \,\mathbf{in}\, \tilde{b})$.

We define a third notion of bisimulation up-to, namely the notion of *bisimulation up-to replication*. For each relation $\mathscr{D}$, we define $\mathscr{D}_R$ as the smallest equivalence relation containing $\mathscr{D}$, that is closed under the replication laws (see Lemma 6.1), that is:

- for all process P we have $P \mathscr{D}_R P$. Moreover, for all processes P, Q and R, we have: if $P \mathscr{D}_R Q$, then $Q \mathscr{D}_R P$, and if $P \mathscr{D}_R Q$ and $Q \mathscr{D}_R R$, then $P \mathscr{D}_R R$;
- if $P \mathscr{D} Q$, then $P \mathscr{D}_R Q$;
- if $(\mathbf{def}\, D \,\mathbf{in}\, a) \mathscr{D}_R (\mathbf{def}\, D' \,\mathbf{in}\, b)$, and if: $\mathbf{fn}(P) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset$, then: $(\mathbf{def}\, D \,\mathbf{in}\, (Pa)) \mathscr{D}_R (\mathbf{def}\, D' \,\mathbf{in}\, (Pb))$;
- for all processes P and Q we have:

$$
\begin{array}{rcll}
(\mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q)) & \mathscr{D}_R & (\mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P) \mid Q)) & \\
(\mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q)) & \mathscr{D}_R & (\mathbf{def}\, p = R \,\mathbf{in}\, (P \mid (\mathbf{def}\, p = R \,\mathbf{in}\, Q))) & \\
(\mathbf{def}\, p = R \,\mathbf{in}\, (Pp)) & \mathscr{D}_R & (\mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P)p)) & \\
(\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\lambda} x)P)) & \mathscr{D}_R & ((\boldsymbol{\lambda} x)(\mathbf{def}\, p = R \,\mathbf{in}\, P)) & (x \notin \mathbf{fn}(R)) \\
(\mathbf{def}\, p = R \,\mathbf{in}\, (\langle u \Leftarrow P \rangle)) & \mathscr{D}_R & (\langle u \Leftarrow (\mathbf{def}\, p = R \,\mathbf{in}\, P) \rangle) & \\
(\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P) & \mathscr{D}_R & (\mathbf{def}\, p = R, q = S \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P)) & (q \notin \mathbf{fn}(R)) \\
(\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P) & \mathscr{D}_R & (\mathbf{def}\, p = R, q = (\mathbf{def}\, p = R \,\mathbf{in}\, S) \,\mathbf{in}\, P) & (q \notin \mathbf{fn}(R))
\end{array}
$$

With the relation $\mathscr{D}_R$, we can define the notion of simulation up-to replication.

Definition 5.3 (Ground Simulation Up-To Replication) The relation $\mathscr{D}$ is a ground simulation *up-to replication*, if for all $(P,Q) \in \mathscr{D}$:

(*i*) if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q such that $Q \stackrel{\mu}{\longmapsto} Q'$ and $P' \mathscr{D}_R Q'$;

(*ii*) if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, then there exists a process Q and an environment D' such that $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{b})}{\longmapsto} (D' \mid Q')$, with $P' \mathscr{D}_R Q'$ and $(\mathbf{def}\, D \,\mathbf{in}\, \tilde{a}) \mathscr{D}_R (\mathbf{def}\, D' \,\mathbf{in}\, \tilde{b})$.

In this report, we prove that a bisimulations up-to is a def-bisimulation, that is we have the following theorem.

Theorem 5.1 (The Up-To Proof Technique) *Let $\mathscr{D}$ be a relation closed by substitution of names for variables. (i) : If $\mathscr{D}$ and $\mathscr{D}^{-1}$ are ground simulation up-to contexts, then $\mathscr{D}$ is a def-bisimulation, that is $\mathscr{D} \subseteq \sim_d$. (ii) : If $\mathscr{D}$ and $\mathscr{D}^{-1}$ are ground simulation up-to structural equivalence, then $\mathscr{D} \subseteq \sim_d$. (iii) If $\mathscr{D}$ and $\mathscr{D}^{-1}$ are ground simulation up-to replication, then $\mathscr{D} \subseteq \sim_d$.*

The complete proof of Th. 5.1 is given in Appendix A. This proof is very technical, therefore, in this section, we only give the sketch of the proof. In this proof, we use the function $F(X)$ defined by:

$$
\begin{aligned}
F(X) \;=\;& X \cup \sim_d \cup \left\{(P,Q) \,\middle|\, \exists R.(P,R) \in X \text{ and } (R,Q) \in X\right\} \\
& \cup \left\{((\boldsymbol{\nu} u)P, (\boldsymbol{\nu} u)Q) \,\middle|\, (P,Q) \in X\right\} \\
& \cup \left\{((P\ a), (Q\ a)) \,\middle|\, (P,Q) \in X\right\} \\
& \cup \left\{(\langle u \Leftarrow P\rangle, \langle u \Leftarrow Q\rangle) \,\middle|\, (P,Q) \in X\right\} \\
& \cup \left\{((\boldsymbol{\lambda} x)P, (\boldsymbol{\lambda} x)Q) \,\middle|\, (P,Q) \in X\right\} \\
& \cup \left\{([P_1\ ,\ l = P_2\,], [Q_1\ ,\ l = Q_2\,]) \,\middle|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\right\} \\
& \cup \left\{(P_1 \mid P_2, Q_1 \mid Q_2) \,\middle|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\right\} \\
& \cup \left\{(\mathbf{def}\ p = P_1\ \mathbf{in}\ P_2, \mathbf{def}\ p = Q_1\ \mathbf{in}\ Q_2) \,\middle|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\right\} \\
& \cup D(X) \cup \mathscr{E}
\end{aligned}
$$

In this section, we do not define the relation $\mathscr{E}$ and the function D. The reader can find these definitions in Appendix A.

The function $F(X)$ is a function that, given a relation X, makes the closure under "all the operators of $\boldsymbol{\pi}^\star$". In particular, for every relation $\mathscr{D}$, it is clear that the relation $\bigcup_{n\geq 0} F^n(\mathscr{D})$ (exists and) is compositional and that it verifies the relation:

$$\mathscr{D}_{\mathrm{C}} \subseteq \bigcup_{n\geq 0} F^n(\mathscr{D})$$

One can see F as an example of "*respectful*" function, as defined by D. Sangiorgi in[26]. Just to give an intuition, the relations $D(X)$ and $\mathscr{E}$ are such that they (respectively) makes a closure under "basic replication laws" (see Lemma 6.1 and Definition 5.2) and "structural equivalence laws". In particular, we prove that: $\mathscr{D}_{\equiv} \subseteq \bigcup_{n\geq 0} F^n(\mathscr{D})$, and that: $\equiv \subseteq \bigcup_{n\geq 0} F^n(\mathscr{E})$, and that $\mathscr{D}_R \subseteq \bigcup_{n\geq 0} F^n(\mathscr{D})$.

Unfortunately, we have to prove the three properties of Th. 5.1 together. That is, we cannot split the proof in three different parts: one for the up-to context proof technique, the other for the up-to $\equiv$, and the last for up-to replication. This is why the definition of $F(X)$ is so complex.

We denote $\mathscr{D}_\infty$ the relation $\mathscr{D}_\infty =_{\mathrm{def}} \bigcup_{n\geq 0} F^n(\mathscr{D})$. In particular $(\mathscr{D}_{\mathrm{C}} \cup \mathscr{D}_{\equiv} \cup \mathscr{D}_R) \subseteq \mathscr{D}_\infty$. Rather than proving Th. 5.1 directly, we prove that $\mathscr{D}_\infty$ is a def-simulation. More precisely we prove Lemma 5.1, that is a more general property than Th. 5.1:

Definition 5.4 (Ground Simulation Up-To F) we say that $\mathscr{D}$ is a ground simulation up-to F, if for all $(P,Q) \in \mathscr{D}$ we have:

(i) if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q such that $Q \stackrel{\mu}{\longmapsto} Q'$ and $P' \mathscr{D}_\infty Q'$;

(ii) if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, then there exists a process Q and an environment D' such that $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{b})}{\longmapsto} (D' \mid Q')$, with $P' \mathscr{D}_\infty Q'$ and $(\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{b})$.

It is clear that the three notions of simulation up-to defined at the beginning of this section are instances of simulation up-to F, Then, to prove Th. 5.1, it is sufficient to prove the following property.

Lemma 5.1 (Bisimulation Up-To F) *if $\mathscr{D}$ and $\mathscr{D}^{-1}$ are ground simulations up-to F, and if $\mathscr{D}$ is closed by substitution of references for variables, then $\mathscr{D}_\infty$ is a def-bisimulation, that is $\mathscr{D}_\infty \subseteq \sim_d$.*

Proof It is easy to prove that, if $\mathscr{D}$ is closed by substitution of references for variables, then the same property holds for $F^n(\mathscr{D})$, and thus for $\mathscr{D}_\infty$. Therefore, we only need to prove that $\mathscr{D}_\infty$ is a ground bisimulation. To prove this last property, we prove the property $(\sharp)$: for all integer k, the relation $F^k(\mathscr{D})$ is a simulation up-to F. More precisely, we prove by induction on k that if $(P,Q) \in F^k(\mathscr{D})$, then:

(i) if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q such that $Q \stackrel{\mu}{\longmapsto} Q'$ and $P' \mathscr{D}_\infty Q'$;

(ii) if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, then there exists a process Q and an environment D' such that $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{c})}{\longmapsto} (D' \mid Q')$, with $P' \mathscr{D}_\infty Q'$ and $(\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{c})$.

The proof in the case $k = 0$ is straightforward, since $F^0(\mathscr{D})$ is the identity relation. The general case is proved in appendix A.

Once proved the property $(\sharp)$, it is clear that $\mathscr{D}_\infty$ is a def-bisimulation. It follows that: $\mathscr{D}_\infty \subseteq \sim_d$. Then, to prove Th. 5.1, we use the fact that: $\mathscr{D} \subseteq F(\mathscr{D}) \subseteq \bigcup_{n\geq 0} F^n(\mathscr{D}) = \mathscr{D}_\infty \subseteq \sim_d$. □

Note that, in the definition of $F(X)$, we use the relation $\sim_d$. Indeed we have $F(X) = X \cup \sim_d \cup \ldots$ Therefore a corollary of Lemma 5.1 is that bisimulation "up-to strong bisimulation" are def-bisimulation. Th. 5.1 is also a corollary of this lemma. For example, suppose that $\mathscr{D}$ is closed under substitutions, and that $\mathscr{D}$ and $\mathscr{D}^{-1}$ are bisimulations up-to contexts. Therefore, since obviously $\mathscr{D}_{\mathrm{C}} \subseteq \mathscr{D}_\infty$, we have that $\mathscr{D}$ and $\mathscr{D}^{-1}$ are simulations up-to F and, using Lemma 5.1 we can prove that $\mathscr{D}_\infty$ is a def-bisimulation and thus: $\mathscr{D} \subseteq \mathscr{D}_\infty \subseteq \sim_d$.

6 Properties of the Def-Bisimulation

A direct consequence of Th. 5.1 is that the strong bisimulation is a congruence that contains structural equivalence.

Theorem 6.1 *The strong def-bisimulation is a congruence and contains structural equivalence, that is:* $\equiv \subset \sim_d$ *, and, for any context* C*, we have* $P \sim_d Q \Rightarrow \mathrm{C}[P] \sim_d \mathrm{C}[Q]$.

Proof The identity relation $\mathcal{I}d$, is a particular example of bisimulation up-to contexts. Therefore, using the proof of Th. 5.1, we prove that: $\bigcup_{n\geq 0} F^n(\mathcal{I}d) \subseteq \sim_d$, and the second property follows from the fact that $\equiv \subseteq \bigcup_{n\geq 0} F^n(\mathcal{I}d)$.

Likewise, to prove that $\sim_d$ is a congruence, we use the fact that it is a particular example of bisimulation up-to contexts. Indeed, using the notation of the proof of Th. 5.1, we prove that $\bigcup_{n\geq 0} F^n(\sim_d)$ is a strong bisimulation, and thus: $\bigcup_{n\geq 0} F^n(\sim_d) \subseteq \sim_d$. This implies that $F(\sim_d) \subseteq \sim_d$, that is $\sim_d$ is closed under contexts. □

The last result states that definitions can be "distributed under every contexts". Therefore, definitions act like substitutions. This will be exemplified at the end of Sect. 6.1.

Lemma 6.1 (Replication Laws) *The following laws hold:*

(*i*) *If* $p \notin \mathbf{fn}(P)$*, thus* $\mathbf{def}\ p = R\ \mathbf{in}\ P \sim_d P$;

(*ii*) $\mathbf{def}\ p = R\ \mathbf{in}\ (P \mid Q) \sim_d \mathbf{def}\ p = R\ \mathbf{in}\ ((\mathbf{def}\ p = R\ \mathbf{in}\ P) \mid Q)$;

(*iii*) *if* $p \neq q$ *and* $q \notin \mathbf{fn}(R)$*, then*

$$\begin{array}{lcl} \mathbf{def}\ p = R\ \mathbf{in}\ (\mathbf{def}\ q = S\ \mathbf{in}\ P) & \sim_d & \mathbf{def}\ p = R, q = S\ \mathbf{in}\ (\mathbf{def}\ p = R\ \mathbf{in}\ P) \\ \mathbf{def}\ p = R\ \mathbf{in}\ (\mathbf{def}\ q = S\ \mathbf{in}\ P) & \sim_d & \mathbf{def}\ q = (\mathbf{def}\ p = R\ \mathbf{in}\ S)\ \mathbf{in}\ (\mathbf{def}\ p = R\ \mathbf{in}\ P) \end{array}$$

(*iv*) $\mathbf{def}\ p = R\ \mathbf{in}\ (\langle u \Leftarrow P\rangle) \sim_d \langle u \Leftarrow (\mathbf{def}\ p = R\ \mathbf{in}\ P)\rangle$;

(*v*) *if* $x \notin \mathbf{fn}(R)$*, then* $\mathbf{def}\ p = R\ \mathbf{in}\ ((\boldsymbol{\lambda} x)P) \sim_d (\boldsymbol{\lambda} x)(\mathbf{def}\ p = R\ \mathbf{in}\ P)$

(*vi*) *for every context* C *that does not bind the names free in* R*, and for every process* P*, the following law hold:* $\mathbf{def}\ p = R\ \mathbf{in}\ (\mathrm{C}[P]) \sim_d \mathbf{def}\ p = R\ \mathbf{in}\ (\mathrm{C}[(\mathbf{def}\ p = R\ \mathbf{in}\ P)])$.

An immediate corollary of laws (*i*) and (*ii*), is that: $\mathbf{def}\ p = R\ \mathbf{in}\ (P \mid Q) \sim_d (\mathbf{def}\ p = R\ \mathbf{in}\ P) \mid (\mathbf{def}\ p = R\ \mathbf{in}\ Q)$, which is the law announced in introduction of this report.

Proof The properties (*i*) to (*v*) are direct consequences of the fact that $F(\sim_d) \subseteq \sim_d$. To prove Lemma 6.1-(*vi*), we prove that the two processes P_1 and P_2 such that:

$$\mathrm{D} =_{\mathrm{def}} \mathbf{def}\ p = R\ \mathbf{in}\ \mathrm{C} \quad \text{and} \quad P_1 =_{\mathrm{def}} \mathrm{D}[P] \quad \text{and} \quad P_2 =_{\mathrm{def}} \mathrm{D}[\mathbf{def}\ p = R\ \mathbf{in}\ P]$$

are bisimilar. Let $[\![.]\!]$ be the function on processes such that (in each case, we suppose that we have used α-conversion to avoid the capture of free names in R):

$$\begin{array}{rcl}
[\![\mathbf{def}\, p = R \,\mathbf{in}\, u]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, u \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\lambda} x)P)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, (\boldsymbol{\lambda} x)([\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!]) \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, (P\ a)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, ([\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!]\ a) \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} u)P)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, (\boldsymbol{\nu} u)([\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!]) \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, ([\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!] \mid [\![\mathbf{def}\, p = R \,\mathbf{in}\, Q]\!]) \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, (\langle u \Leftarrow P\rangle)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in}\, \langle u \Leftarrow ([\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!])\rangle \\
[\![\mathbf{def}\, p = R \,\mathbf{in}\, (\mathbf{def}\, q = S \,\mathbf{in}\, Q)]\!] & = & \mathbf{def}\, p = [\![R]\!] \,\mathbf{in} \left(\begin{array}{l} \mathbf{def}\, q = [\![\mathbf{def}\, p = R \,\mathbf{in}\, S]\!] \\ \mathbf{in}\, [\![\mathbf{def}\, p = R \,\mathbf{in}\, Q]\!] \end{array} \right)
\end{array}$$

and such that $[\![.]\!]$ is an homomorphism in the other cases. The function $[\![.]\!]$ is used to "maximally" distribute definitions. If we add the constant $[\_]$, such that $[\![\mathbf{def}\, p = R \,\mathbf{in}\, [\_]]\!] = [\_]$, this function extends to a function from contexts to contexts. Using the distribution laws, it is easy to prove that, for all process P, we have: $[\![P]\!] \sim_d P$ and: $[\![\mathrm{D}]\!][P] \sim_d \mathrm{D}[P]$. Therefore it is obvious that:

$$P_1 \sim_d [\![P_1]\!] = [\![\mathrm{D}]\!][[\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!]] \quad \text{and} \quad \mathrm{D}[[\![\mathbf{def}\, p = R \,\mathbf{in}\, P]\!]] \sim_d P_2$$

The expected result i.e., $P_1 \sim_d P_2$, follows from the transitivity of $\sim_d$. □

6.1 Interpretation of the λ-Calculus

Let Λ denotes the untyped λ-calculus.

$$M, N, L ::= x \mid \lambda x.M \mid (MN)$$

We consider the full calculus, that is without any particular reduction strategy, and we denote $\leftrightarrow_\beta$ the β-conversion relation, that is the smallest congruence such that $(\lambda x.M)M \leftrightarrow_\beta M\{N/x\}$. The replication laws can be used to prove the correctness of a (very) simple interpretation from Λ to $\boldsymbol{\pi}^\star$ already sketched in Sect. 2.1. Our interpretation of Λ is

$$\begin{array}{rcl}
(\!|x|\!) & = & x \\
(\!|\lambda x.M|\!) & = & (\boldsymbol{\lambda} x)(\!|M|\!) \\
(\!|(MN)|\!) & = & \mathbf{def}\, p = (\!|N|\!) \,\mathbf{in}\, ((\!|M|\!)\ p) \quad p \notin \mathbf{fn}(M, N)
\end{array}$$

In the following, we will make no distinction between variables of Λ and names of $\boldsymbol{\pi}^\star$. We prove that the interpretation of a Λ-redex, is a redex of $\boldsymbol{\pi}^\star$. Let $P[x{:=}Q]$ be the process defined by

$$P[x{:=}Q] =_{\mathrm{def}} \mathbf{def}\, x = Q \,\mathbf{in}\, P \qquad \text{if } x \notin \mathbf{fn}(Q)$$

Therefore, using Lemma 3.3, we have

$$(\!|(\lambda x.M)N|\!) = ((\boldsymbol{\lambda} x)(\!|M|\!)\ p)[p{:=}(\!|N|\!)] \approx_b ((\!|M|\!)\{p/x\})[p{:=}(\!|N|\!)]$$

To reason about the interpretation of β reduction, we assume that the replication laws are valid for barbed congruence. This result, stated in Th. 7.1, is proved latter. With this hypothesis, we can prove that:

$$(\!(M)\!)\{p/x\})[p{:=}(\!(N)\!)] \approx_b (\!(M)\!)\{(\!(N)\!)/x\} = (\!(M\{N/x\})\!)$$

Therefore we can prove that the interpretation of beta convertible terms are equivalent.

Theorem 6.2 *If $M \leftrightarrow_\beta N$, then $(\!(M)\!) \approx_b (\!(N)\!)$.*

More precisely, we can draw a parallel between the replication laws and the transformations defined in the λ-calculus with *explicit substitutions* [2]. Indeed, if we extend Λ with explicit substitutions: $M[x{:=}N]$, and if we define $(\!(M[x{:=}N])\!)$ to be the process $(\!(M)\!)[x{:=}(\!(N)\!)]$, we can prove that:

let x, y be two variables such that $x \neq y$ and $y \notin \mathbf{fn}(R)$

$$\begin{array}{rcll}
(P[y{:=}Q])[x{:=}R] & \approx_b & (P[x{:=}R])[y{:=}(Q[x{:=}R])] & \\
x[x{:=}R] & \approx_b & R & \text{(Proposition 3.1)} \\
y[x{:=}R] & \approx_b & y & \text{(Lemma 6.1-}(i)\text{)} \\
(\!(\lambda y.M)\!)[x{:=}(\!(N)\!)] & \approx_b & (\boldsymbol{\lambda} y)((\!(M)\!)[x{:=}(\!(N)\!)]) & \text{(Lemma 6.1-}(v)\text{)} \\
(\!(M\ L)\!)[x{:=}(\!(N)\!)] & \approx_b & (\!((M[x{:=}N])\ (L[x{:=}N]))\!) & \text{(Lemma 6.1-}(ii)\text{)}
\end{array}$$

This last result has to be compared with the remark of R. Milner [21, Sect. 4], that the correspondence between the λ-calculus with explicit substitution and the π-calculu may be closer than between the λ-calculus and the π-calculus.

7 Relation Between Transitions and Reduction

In this section, we prove that the labeled transition system is "faithful" to the notion of reduction. That is, a reduction $P \to Q$ in the original calculus can be mapped to a τ-transition of the labeled reduction system (this is Lemma 7.2). Nonetheless, a main difference with labeled semantics defined for $\boldsymbol{\pi}$ is that τ-actions are not equivalent to reduction. The following example was already given in Sect. 4.2

$$\begin{array}{rcl}
\langle u \Leftarrow Q \rangle \mid \mathbf{def}\, p = R \,\mathbf{in}\, (u\ p \mid P) & \to & \mathbf{def}\, p = R \,\mathbf{in}\, ((Q\ p) \mid P) \\
& \overset{\tau}{\longmapsto} \equiv & \mathbf{def}\, p = R \,\mathbf{in}\, ((Q\ p) \mid (\mathbf{def}\, p = R \,\mathbf{in}\, P))
\end{array}$$

The intuition is that, in the extrusion of bound references, definitions are duplicated. To establish this formally, we need a preliminary result proved in Appendix B.

Lemma 7.1 (Relation Between Transitions and Actions)

(*i*) ***tau actions:*** *if $P \overset{\tau}{\longmapsto} P'$ and* E *is an evaluation context, then* $\mathrm{E}[P] \overset{\tau}{\longmapsto} \mathrm{E}[P']$*;*

(*ii*) ***lambda actions:*** *if* $P \stackrel{\lambda a}{\longmapsto} P'$, *then* $(Pa) \rightarrow P'$;

(*iii*) ***in actions:*** *if* $P \stackrel{\mathbf{in}\ u.(\tilde{b};\tilde{a})}{\longmapsto} P'$, *then* $(P\ \tilde{b}) \mid (u\tilde{a}) \rightarrow P'$;

(*iv*) ***out actions:*** *if* $P \stackrel{\mathbf{out}\ u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$, *then there exists a process* R *such that:*

$$P \equiv (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (u\tilde{a} \mid R)) \quad \textit{and} \quad (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ R) \sim_d (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ P')$$

In Lemma 7.1, we have related each kind of actions to a reduction rule. For example, λ-actions are associated with the (red beta) rule. With this property, we can prove that the l.t.s. simulates the reduction, that is:

Lemma 7.2 ($\stackrel{\tau}{\longmapsto}$ simulates $\rightarrow$) *If there is a reduction from a process* P *to a process* P', *then there exists an equivalent (labeled) transition up-to structural equivalence, that is:*

(*i*) *if* $P \rightarrow P'$, *then there exists* R *such that* $P \equiv R$, *and* $R \stackrel{\tau}{\longmapsto} R'$, *and* $R' \equiv P'$.

Conversely, the existence of a tau transition implies the existence of a reduction. In this case, the residuals are bisimilar. That is:

(*ii*) *if* $P \stackrel{\tau}{\longmapsto} P'$, *then there exists* R *such that* $P \rightarrow R$, *and* $R \sim_d P'$.

Proof (Lemma 7.2) The proof of property (*i*) is made by induction on the inference of $P \rightarrow P'$.

- **case (red equiv):** we have $P \equiv Q$ and $Q \rightarrow P'$. Thus, using the induction hypothesis, there exists R and R' such that $Q \equiv R$, $R \stackrel{\tau}{\longmapsto} R'$, $R' \equiv P'$. The result follows from transitivity of the structural equivalence: $P \equiv Q \equiv R \Rightarrow P \equiv R$;
- **case (red context):** we have $P = \mathrm{E}[Q]$, $Q \rightarrow Q'$ and $P' = \mathrm{E}[Q']$. Therefore there exists R such that $Q \equiv R \stackrel{\tau}{\longmapsto} R' \equiv Q'$. Using Lemma 7.1-(*i*) and the fact that $\equiv$ is a congruence, it follows that $P \equiv \mathrm{E}[R] \stackrel{\tau}{\longmapsto} \mathrm{E}[R'] \equiv P'$;
- **case (red beta) and (red sel):** in the first case, we have $P = ((\boldsymbol{\lambda}x)Q)u$ and $P' = Q\{u/x\}$. Therefore rule (lambda) implies that: $(\boldsymbol{\lambda}x)Q \stackrel{\lambda\,u}{\longmapsto} Q\{u/x\}$ and rule (app lambda) implies that $P \stackrel{\tau}{\longmapsto} P'$. The proof is similar in the second case;
- **case (red decl):** Let $\tilde{a}$ be the tuple $(a_1, \ldots, a_n)$. We have $P = (\langle u \Leftarrow Q\rangle \mid u\tilde{a})$ and $P' = (Q\tilde{a})$. Therefore, using rule (decl) we have: $\langle u \Leftarrow Q\rangle \stackrel{\mathbf{in}\ u.(\epsilon;\tilde{a})}{\longmapsto} (Q\tilde{a})$. Moreover, using rule (out) and (app out), we have: $(u\tilde{a}) \stackrel{\mathbf{out}\ u.(\epsilon;\tilde{a})}{\longmapsto} (\emptyset; \mathtt{0}\tilde{a})$. Thus, using rule (par com), it follows that: $P \stackrel{\tau}{\longmapsto} \mathbf{def}\ \mathtt{0}\ \mathbf{in}\ (P' \mid (\mathtt{0}\tilde{a})) \equiv P'$;
- **case (red def):** we have $P = (\mathbf{def}\ p = T\ \mathbf{in}\ \mathrm{E}[p])$ and $P' = (\mathbf{def}\ p = T\ \mathbf{in}\ \mathrm{E}[T])$, with the side condition that p and the free name of T are not bound by E. Using the structural equivalence, we can always suppose that E is in normal form i.e.,

$$\mathrm{E} = (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (\ [\_]\ \tilde{a} \mid Q))$$

Thus $P \equiv R =_{\text{def}} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D, p = T\, \mathbf{in}\, (p\tilde{a} \mid Q))$, and using rule (out), (app out), and (def com) we have:

$$P \equiv R \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D, p = R\, \mathbf{in}\, (T\tilde{a} \mid (\mathtt{0}\ \tilde{a} \mid Q))) \equiv P'$$

The proof of the second property is made by induction on the inference of $P \stackrel{\tau}{\longmapsto} P'$. We make a case analysis on the last rule.

- **case (par tau), (app tau), (new mu) and (def mu):** the result follows from rule (red context) and the fact that $\sim_d$ is a congruence;
- **case (app lambda):** here we have a β reduction, that is: $P = (Qa)$ and $Q \stackrel{\lambda\, a}{\longmapsto} P'$. The result follows from Lemma 7.1-(ii). The case of record selection is similar;
- **case (par com) and (def com):** we have $P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D \mid P')$ and $Q \stackrel{\mathbf{in}\, u.(\epsilon;\tilde{a})}{\longmapsto} Q'$ implies $P \mid Q \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (P' \mid Q'))$. Therefore, using Lemma 7.1-(iii), $Q \mid (u\tilde{a}) \rightarrow Q'$ and, using Lemma 7.1-(iv), there exists R such that $P \equiv (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (u\tilde{a} \mid R))$ and $(\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, R) \sim_d (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, P')$. Combining these two properties we obtain:

$$\begin{array}{rcll} (P \mid Q) & \equiv & (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (u\tilde{a} \mid R)) \mid Q & \\ & \equiv & (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (R \mid (Q \mid u\tilde{a}))) & \\ & \rightarrow & (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (R \mid Q')) & (2) \\ & \sim_d & (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (P' \mid Q')) & (3) \end{array}$$

where relation (2) is proved using Lemma 7.1-(iii) and rule (red context), and relation (3) is proved using Th. 6.1. In the last relation, we also use the fact that $\mathbf{fn}(Q) \cap (\tilde{v}, \mathbf{def}(D)) = \emptyset$ implies $\mathbf{fn}(Q') \cap (\tilde{v}, \mathbf{def}(D)) = \emptyset$. The proof is similar in the case (def com).

□

We prove now that a process is a value, in the sense of Definition 3.1, if and only if we can derive a lambda action from it.

Lemma 7.3 (Characterization of Values) *The process P is a value, denoted $P\downarrow$, if and only if there exists a name a such that $P \stackrel{\lambda\, a}{\longmapsto} P'$.*

Proof The first implication is proved by induction on the size of P, and the reverse property is proved by induction on the inference of $P \stackrel{\lambda\, a}{\longmapsto} P'$. □

Finally we prove our main result, stating that the def-bisimulation is included in the barbed congruence.

Theorem 7.1 *The def-bisimulation is a barbed-bisimulation, that is $P \sim_d Q$ implies $P \approx_b Q$.*

Proof In Sect. 5, we already proved that $\sim_d$ is symmetric and that it is a congruence. Therefore it is enough to prove that (1): the def-bisimulation $\sim_d$ is a bisimulation, and that (2): the def-bisimulation preserves the barbs: $P \sim_d Q$ and $P\downarrow$, implies $Q\downarrow$.

(1) Suppose $P \sim_d Q$ and $P \to P'$. We prove that there exists a process Q' such that $Q \to Q'$ and $P' \sim_d Q'$. From Lemma 7.2-(i), it follows that there exists a process R such that $(\sharp) : P \equiv R$, $R \stackrel{\tau}{\longmapsto} R'$ and $(\natural) : R' \equiv P'$. From Th. 6.1 and property $(\sharp)$, it follows that $Q \sim_d R$. Therefore, since $P \sim_d Q$, there exists a process S such that: $Q \stackrel{\tau}{\longmapsto} S$ with $R' \sim_d S$, and also (using Th. 6.1 and property $(\natural)$): $P' \sim_d S$. Finally, using Lemma 7.2-(ii), there exists a process Q' such that: $Q \to Q'$ and $Q' \sim_d S$, and thus such that: $Q' \sim_d P'$;

(2) Suppose $P\downarrow$. From Lemma 7.3, it follows that there exists a name a such that $P \stackrel{\lambda\, a}{\longmapsto} P'$. Since $P \sim_d Q$, there exists a process Q' such that $Q \stackrel{\lambda\, a}{\longmapsto} Q'$, and thus $Q\downarrow$.

□

This last result implies that the replication laws are also valid for barbed congruence, that is we have the following properties.

Theorem 7.2 (Replication Laws)

$$
\begin{array}{rcll}
\mathbf{def}\ p = R\ \mathbf{in}\ (P \mid Q) & \approx_b & (\mathbf{def}\ p = R\ \mathbf{in}\ P) \mid (\mathbf{def}\ p = R\ \mathbf{in}\ Q) & \\
\mathbf{def}\ p = R\ \mathbf{in}\ (Pa) & \approx_b & \mathbf{def}\ p = R\ \mathbf{in}\ ((\mathbf{def}\ p = R\ \mathbf{in}\ P)a) & \\
\mathbf{def}\ p = R\ \mathbf{in}\ (\mathbf{def}\ q = S\ \mathbf{in}\ P) & \approx_b & \mathbf{def}\ q = (\mathbf{def}\ p = R\ \mathbf{in}\ S)\ \mathbf{in}\ (\mathbf{def}\ p = R\ \mathbf{in}\ P) & \\
\mathbf{def}\ p = R\ \mathbf{in}\ (\langle u \Leftarrow P\rangle) & \approx_b & \langle u \Leftarrow (\mathbf{def}\ p = R\ \mathbf{in}\ P)\rangle & \\
(\mathbf{def}\ p = R\ \mathbf{in}\ ((\boldsymbol{\lambda} x)P)) & \approx_b & (\boldsymbol{\lambda} x)(\mathbf{def}\ p = R\ \mathbf{in}\ P) & (x \notin \mathbf{fn}(Q))
\end{array}
$$

8 Conclusion

To my knowledge, there is no direct proof of any non-trivial laws for barbed congruence, that is a proof exhibiting a barbed bisimulation that contains the desired equalities. In his article on the interpretation of Λ in $\boldsymbol{\pi}$ [21], R. Milner gives an insightful intuitive reason for this fact: the reduction relation informs us solely on the internal behavior of a process P; it describes how P's parts may interact with each other, but not how P may interact with the environment.

A first solution to reduce the proofs to a tractable size, is to prove a *context lemma*, like G. Boudol did in [10] for example, and to use "up-to" proof techniques. But, with these techniques, one can only prune the set of contexts that have to be considered. Therefore, the most useful solution is certainly to define a labeled bisimulation that is included in the congruence and to prove the interesting properties with this equivalence.

In this report, we have used this last solution to prove that an extended formulation of the replication theorem is valid for barbed congruence. Only recently, a similar property was proved for the local π-calculus [19]. But this calculus is not polyadic and the labeled bisimulation used is not higher-order.

The method used in the proof of this key result is also interesting. The major novelty is that τ-transition does not match reduction, i.e. $P \stackrel{\tau}{\longmapsto} P'$ does not imply $P \to P'$ (see Lemma 7.2). This "trick" has greatly simplified our proofs. Another key step is the use of the up-to context proof method, instead of a direct proof that $\sim_d$ is a congruence: in $\boldsymbol{\pi}$, one generally proves that a bisimulation is a congruence, by exhibiting one bisimulation for each constructor, a method that is not possible for $\boldsymbol{\pi}^\star$.

To conclude, I presented some applications that stress the importance of the replication theorem. In Sect. 6.1, we have used the replication laws to prove the correctness of an encoding of $(\Lambda, \leftrightarrow_\beta)$. A similar result is given in [11], where the source language considered is a calculus of primitive objects [1]. In this article, we define a set of derived constructs for imperative "named objects", where imperative means that an object can be cloned. In this setting, an object is a $\boldsymbol{\pi}^\star$ process, denoted $(\mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ P)$, such that selection of the field l_j on the reference o, triggers the execution of the method P_i with x_i bound to o

$$\mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ (o \Leftarrow l_j) \to^* \mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ (P_j\{o/x_j\})$$

The replication theorem is used to prove algebraic laws such as, under certain hypotheses on the occurrences of o in P and Q, namely that the object o is used only for cloning:

$$\begin{cases} \mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ (\mathbf{clone}(o)) & \approx_b \quad \mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ o \\ \mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ (P \mid Q) & \approx_b \quad (\mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ P) \mid \\ & \qquad\quad (\mathbf{obj}\, o\, \mathbf{is}\, (l_i = \varsigma(x_i)P_i^{\,i\in I})\, \mathbf{in}\ Q) \end{cases}$$

Another application of the replication theorem is found in the distributed calculus of [13], built on top of $\boldsymbol{\pi}^\star$. In this calculus, we consider the rule

$$(\star) \qquad \mathbf{def}\, p = R\, \mathbf{in}\, (P \mid Q) \equiv (\mathbf{def}\, p = R\, \mathbf{in}\, P) \mid (\mathbf{def}\, p = R\, \mathbf{in}\, Q)$$

as primitive in the definition of the structural equivalence. Note that the "soundness" of this system is implied by Lemma 6.1-(ii). The reason that motivates this choice, is that it simplifies the definition of the reduction relation. Let us define part of the syntax of the distributed $\boldsymbol{\pi}^\star$, denoted $\mathbf{D}\boldsymbol{\pi}^\star$ here, to argument this claim. This calculus extend $\boldsymbol{\pi}^\star$ with locations: $[s :: P]$, that stands for the process P located at the site s, and with an RPC like construct: $\mathbf{go}\ s'.Q$, used to spawn the process Q at the site s'. In $\mathbf{D}\boldsymbol{\pi}^\star$, the following reduction is admissible

$$\left(\begin{array}{l} [s :: \mathbf{def}\, p = R\, \mathbf{in}\, (\mathbf{go}\ s'.p \mid P)] \mid \\ [s' :: Q] \end{array}\right) \to \left(\begin{array}{l} [s :: \mathbf{def}\, p = R\, \mathbf{in}\, P] \mid \\ [s' :: \mathbf{def}\, p = R\, \mathbf{in}\, p \mid Q] \end{array}\right)$$

Thus, rule ($\star$) allows one to consider that private and replicated resources are basic bricks that can be safely copied in a communication between distant locations. Another interpretation is that, if efficiency is not taken into account, an RPC communication amounts to send the code of the function being called at the location of the client. This intuition is at the heart of the paradigm of remote evaluation [27].

A Proof of the Up-To Technique Theorem

In this section we prove Lemma 5.1. Note that we proved in section 5 that this lemma implies Th. 5.1. Let $F(X)$ be the function from relations to relations sketched in Sect. 5. That is $F(X) =_{\text{def}} G(X) \cup D(X) \cup \mathscr{E}$, where $G(X)$ is a function that makes a closure over $\boldsymbol{\pi}^\star$'s operators, and $D(X)$ is a function that makes a closure over "basic distribution laws", and $\mathscr{E}$ is a relation used to recover the "structural equivalence laws".

$$
\begin{aligned}
G(X) \;=\;& X \cup \sim_d \cup \big\{(P,Q) \,\big|\, \exists R.(P,R) \in X \text{ and } (R,Q) \in X\big\} \\
& \cup \big\{((\boldsymbol{\nu} u)P, (\boldsymbol{\nu} u)Q) \,\big|\, (P,Q) \in X\big\} \\
& \cup \big\{((P\ a),(Q\ a)) \,\big|\, (P,Q) \in X\big\} \\
& \cup \big\{(\langle u \Leftarrow P\rangle, \langle u \Leftarrow Q\rangle) \,\big|\, (P,Q) \in X\big\} \\
& \cup \big\{((\boldsymbol{\lambda} x)P, (\boldsymbol{\lambda} x)Q) \,\big|\, (P,Q) \in X\big\} \\
& \cup \big\{([P_1\,,\, l = P_2\,],[Q_1\,,\, l = Q_2\,]) \,\big|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\big\} \\
& \cup \big\{(P_1 \mid P_2, Q_1 \mid Q_2) \,\big|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\big\} \\
& \cup \big\{(\mathbf{def}\, p = P_1 \,\mathbf{in}\, P_2, \mathbf{def}\, p = Q_1 \,\mathbf{in}\, Q_2) \,\big|\, (P_1,Q_1) \text{ and } (P_2,Q_2) \in X\big\}
\end{aligned}
$$

$$
\begin{aligned}
D(X) \;=\;& \{(\mathbf{def}\, D \,\mathbf{in}\, (P\ a), \mathbf{def}\, D' \,\mathbf{in}\, (P\ b)) \mid\ (\mathbf{def}\, D \,\mathbf{in}\, u, \mathbf{def}\, D' \,\mathbf{in}\, v) \in X \text{ and} \\
& \qquad\qquad \mathbf{fn}(P) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset\ \} \\
& \cup \{(\mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q), \mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P) \mid Q))\} \\
& \cup \{(\mathbf{def}\, p = R \,\mathbf{in}\, (P\ p), \mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P)\ p))\} \\
& \cup \big\{(\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\lambda} x)P), (\boldsymbol{\lambda} x)(\mathbf{def}\, p = R \,\mathbf{in}\, P)) \,\big|\, x \notin \mathbf{fn}(R)\big\} \\
& \cup \{(\mathbf{def}\, p = R \,\mathbf{in}\, (\langle u \Leftarrow P\rangle), \langle u \Leftarrow (\mathbf{def}\, p = R \,\mathbf{in}\, P)\rangle)\} \\
& \cup \big\{(\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P, \mathbf{def}\, p = R, q = S \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P)) \,\big|\, q \notin \mathbf{fn}(R)\big\} \\
& \cup \big\{(\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P, \mathbf{def}\, p = R, q = (\mathbf{def}\, p = R \,\mathbf{in}\, S) \,\mathbf{in}\, P) \,\big|\, q \notin \mathbf{fn}(R)\big\}
\end{aligned}
$$

$$
\begin{aligned}
\mathscr{E} \;=\;& \{(P \mid \mathtt{0}, P)\} \cup \{(P \mid Q, Q \mid P)\} \cup \{(P \mid (Q \mid R), (P \mid Q) \mid R)\} \\
& \cup \big\{(\mathbf{def}\, p = R \,\mathbf{in}\, P, P) \,\big|\, p \notin \mathbf{fn}(P)\big\} \ \cup\ \big\{((\boldsymbol{\nu} u)P, P) \,\big|\, u \notin \mathbf{fn}(P)\big\} \\
& \cup \{(\mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} u)P), (\boldsymbol{\nu} u)(\mathbf{def}\, p = R \,\mathbf{in}\, P))\} \\
& \cup \big\{(\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P, \mathbf{def}\, q = S, p = R \,\mathbf{in}\, P) \,\big|\, q \neq p, p \notin \mathbf{fn}(S), q \notin \mathbf{fn}(R)\big\} \\
& \cup \{((\boldsymbol{\nu} u)(\boldsymbol{\nu} v)P, (\boldsymbol{\nu} v)(\boldsymbol{\nu} u)P)\} \\
& \cup \big\{((\boldsymbol{\nu} u)P \mid Q, (\boldsymbol{\nu} u)(P \mid Q)) \,\big|\, u \notin \mathbf{fn}(Q)\big\} \\
& \cup \big\{((\mathbf{def}\, p = R \,\mathbf{in}\, P) \mid Q, \mathbf{def}\, p = R \,\mathbf{in}\, (P \mid Q)) \,\big|\, p \notin \mathbf{fn}(Q)\big\} \\
& \cup \big\{(((\boldsymbol{\nu} u)P)\ a, (\boldsymbol{\nu} u)(P\ a)) \,\big|\, a \neq u\big\} \\
& \cup \big\{((\mathbf{def}\, p = R \,\mathbf{in}\, P)\ a, \mathbf{def}\, p = R \,\mathbf{in}\, (P\ a)) \,\big|\, a \neq p\big\} \\
& \cup \{((P \mid Q)\ a, (P\ a) \mid (Q\ a))\} \\
& \cup \{(\langle u \Leftarrow P\rangle\ a, \langle u \Leftarrow P\rangle)\} \cup \{(\mathtt{0}\ a, \mathtt{0})\}
\end{aligned}
$$

The function F is monotonic and for all $n \geq 1$, we have $X \subseteq F(X) \subseteq F^n(X)$. Thus, for every relation $\mathscr{D}$, we can define the relation $\mathscr{D}_\infty =_{\text{def}} \bigcup_{n \geq 0} F^n(\mathscr{D})$. It is straightforward to prove that if $\mathscr{D}$ is closed by substitution of names for variables, then the same holds for $F^n(\mathscr{D})$, and thus for $\mathscr{D}_\infty$. Moreover, is easy to prove that, for every relation $\mathscr{D}$, the relation

$\bigcup_{n\geq 0} G^n(\mathscr{D})$ is a congruence, and that the relation $\bigcup_{n\geq 0} G^n(\mathscr{E})$ contains $\equiv$. Concerning the function $D(X)$, it is easy to prove that $\mathscr{D}_R \subseteq \mathscr{D}_\infty$. Moreover, it is possible to prove that $(\mathbf{def}\, D\, \mathbf{in}\, u)\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, v)$, implies $(\mathbf{def}\, D\, \mathbf{in}\, P\{u/x\})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P\{v/x\})$, whenever $(\mathbf{def}(D) \cup \mathbf{def}(D')) \cap \mathbf{fn}(P) = \emptyset$.

In this section, we prove that if $\mathscr{D}$ is a bisimulation up-to F (see Lemma 5.1), then $\mathscr{D}_\infty$ is a def-simulation. We have already noted that it is closed by substitutions, therefore it is sufficient to prove that $\mathscr{D}_\infty$ is a ground simulation. Before to establish this result, we prove two intermediary lemmas.

Lemma A.1 *Let $\mathscr{D}$ be a bisimulation up-to F, and let $\mathscr{D}_\infty$ be the relation $\bigcup_{n\geq 0} F^n(\mathscr{D})$, defined above. Let $\tilde{a}, \tilde{c}$ be two tuples, and D, D' be two definitions lists such that:*
$(\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{c})$. *The following properties hold:*

(*i*) *if $P\, \mathscr{D}_\infty\, Q$ and $(\mathbf{fn}(P) \cap \mathbf{def}(D)) = (\mathbf{fn}(Q) \cap \mathbf{def}(D')) = \emptyset$, then*
$(\mathbf{def}\, D\, \mathbf{in}\, (P\ \tilde{a}))\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, (Q\ \tilde{c}))$

(*ii*) *if $P \stackrel{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} P'$ and $\mathbf{fn}(P) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset$, then there exists a process P'' such that $P \stackrel{\mathbf{in}\, u.(\tilde{b};\, \tilde{c})}{\longmapsto} P''$ and $(\mathbf{def}\, D\, \mathbf{in}\, P')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P'')$. And if $P \stackrel{\mathbf{in}\, u.(\tilde{a};\, \tilde{b})}{\longmapsto}$ P' and $\mathbf{fn}(P) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset$, then there exists a process P'' such that $P \stackrel{\mathbf{in}\, u.(\tilde{c};\, \tilde{b})}{\longmapsto} P''$ and $(\mathbf{def}\, D\, \mathbf{in}\, P')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P'')$;*

Let a, c be two names such that $(\mathbf{def}\, D\, \mathbf{in}\, a)\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, c)$.

(*iii*) *if $P \stackrel{\lambda\, a}{\longmapsto} P'$ and $\mathbf{fn}(P) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset$, then there exists a process P'' such that $P \stackrel{\lambda\, c}{\longmapsto} P''$ with $(\mathbf{def}\, D\, \mathbf{in}\, P')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P'')$.*

Proof Let $\tilde{a}, \tilde{c}$ be two tuples of the same size, and P, Q be two processes such that $P\, \mathscr{D}_\infty\, Q$. For the first property, we make an induction on the size of $\tilde{a}$.

- **case** $|\tilde{a}| = 0$**:** since $\left\{(\mathbf{def}\, p = R\, \mathbf{in}\, P, P) \;\middle|\; p \notin \mathbf{fn}(P)\right\}$ is a subset of $\mathscr{D}_\infty$, we have

$$(\mathbf{def}\, D\, \mathbf{in}\, P)\, \mathscr{D}_\infty\, P\, \mathscr{D}_\infty\, Q\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, Q)$$

  in the cases such that $(\mathbf{fn}(P) \cap \mathbf{def}(D)) = (\mathbf{fn}(Q) \cap \mathbf{def}(D')) = \emptyset$;

- **case** $|\tilde{a}| = n + 1$**:** suppose that $(\mathbf{def}\, D\, \mathbf{in}\, (P\ \tilde{a}))\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, (Q\ \tilde{c}))$. The pair $((\mathbf{def}\, D\, \mathbf{in}\, (Pa)), (\mathbf{def}\, D'\, \mathbf{in}\, (Pc)))$ is in $\mathscr{D}_\infty$, therefore, using the transitivity of $\mathscr{D}_\infty$, and the fact that it is a congruence, we have:

$$\begin{array}{rcl}
\mathbf{def}\, D\, \mathbf{in}\, (P\ \tilde{a}\ a_{n+1}) & = & \mathbf{def}\, D\, \mathbf{in}\, ((\mathbf{def}\, D\, \mathbf{in}\, P\ \tilde{a})\ a_{n+1}) \\
 & \mathscr{D}_\infty & \mathbf{def}\, D'\, \mathbf{in}\, ((\mathbf{def}\, D\, \mathbf{in}\, P\ \tilde{a}) c_{n+1}) \\
 & \mathscr{D}_\infty & \mathbf{def}\, D'\, \mathbf{in}\, ((\mathbf{def}\, D'\, \mathbf{in}\, Q\ \tilde{c})\ c_{n+1}) \\
 & \mathscr{D}_\infty & \mathbf{def}\, D'\, \mathbf{in}\, (Q\ \tilde{c}\ c_{n+1})
\end{array}$$

We prove the second property by induction on the inference of $P \stackrel{\mathbf{in}\,u.(\tilde{b};\tilde{a})}{\longmapsto} P'$. Let $\mu_a = \mathbf{in}\,u.(\tilde{b};\tilde{a})$ and $\mu_c = \mathbf{in}\,u.(\tilde{b};\tilde{c})$:

- **case (decl):** we have $P = \langle u \Leftarrow P_1 \rangle \stackrel{\mu_a}{\longmapsto} P' = (P_1\ \tilde{a})$. Therefore: $P \stackrel{\mu_c}{\longmapsto} (P_1\ \tilde{c})$ and, since $P_1 \mathscr{D}_\infty P_1$, the expected result: $(\mathbf{def}\ D\ \mathbf{in}\ (P_1\ \tilde{a}))\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ (P_1\ \tilde{c}))$, follows from Lemma A.1-(i);

- **case (new mu):** we have $P = (\boldsymbol{\nu}\, u)P_1$ and $P_1 \stackrel{\mu_a}{\longmapsto} P_1'$ implies $P \stackrel{\mu_a}{\longmapsto} (\boldsymbol{\nu}\, u)P_1'$. Therefore, using the induction hypothesis, it follows that there exists a process P_1'' such that $P_1 \stackrel{\mu_c}{\longmapsto} P_1''$, with $(\mathbf{def}\ D\ \mathbf{in}\ P_1')\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ P_1'')$. The result follows from the fact that $\mathscr{D}_\infty$ is a congruence. The proof is similar in the case (def mu) and (par in);

- **case (app in):** we have: $P = (P_1\ b)$ and $P_1 \stackrel{\mathbf{in}\,u.((b,\tilde{b});\tilde{a})}{\longmapsto} P_1'$, implies $(P\ b) \stackrel{\mu_a}{\longmapsto} P_1'$. Using the induction hypothesis, it follows that there exists a process P_1'' such that $P_1 \stackrel{\mathbf{in}\,u.((b,\tilde{b});\tilde{c})}{\longmapsto} P_1''$ and $(\mathbf{def}\ D\ \mathbf{in}\ P_1')\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ P_1'')$. The result follows from the fact that $(P\ b) \stackrel{\mu_c}{\longmapsto} P_1''$.

We prove the third property by induction on the inference of $P \stackrel{\lambda\, u}{\longmapsto} P'$. The property is trivial if a is a label, or if $a = b$. Therefore, in the following proof, we consider that a is bound by D:

- **case (lambda):** we have $P = (\boldsymbol{\lambda} x)P_1$. The result follows from the fact that

  $$(\mathbf{def}\ D\ \mathbf{in}\ P\{a/x\})\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ P\{c/x\})$$

  We obtain this result by using the "laws in $D(X)$", to distribute the definitions over any construct of P. Then we use the fact that, for all Q such that $(\mathbf{fn}(Q) \cap (\mathbf{def}(D) \cup \mathbf{def}(D'))) = \emptyset$, the relation $(\mathbf{def}\ D\ \mathbf{in}\ (Q\ a))\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ (P\ c))$ is valid.

- **case (par lambda):** The result follows from property (i) and the rule of distribution of definitions over parallel composition;

- **case (new mu) and (def mu):** the result follows from $\mathscr{D}_\infty$ being a congruence.

□

Lemma A.2 (In Actions and Application) *If $P \stackrel{\mathbf{in}\,u.(\tilde{b};\tilde{a})}{\longmapsto} P'$, then there exists an* **in***-transition such that $P \stackrel{\mathbf{in}\,u.((\tilde{b},a);(\tilde{a},a))}{\longmapsto} P''$ with $P'' \equiv (P'\ a)$.*

Using rule (app in), we obtain, as a corollary, that if $P \stackrel{\mathbf{in}\,u.(\epsilon;\tilde{a})}{\longmapsto} P'$, then $(P\ a) \stackrel{\mathbf{in}\,u.(\epsilon;(\tilde{a},a))}{\longmapsto} P''$ with $P'' \equiv (P'\ a)$.

Proof The proof is by induction on the inference of $P \stackrel{\mathbf{in}\,u.(\tilde{b};\tilde{a})}{\longmapsto} P'$. For convenience we will use μ to denote the action $\mathbf{in}\,u.(\tilde{b};\tilde{a})$, and μ_a to denote the action $\mathbf{in}\,u.((\tilde{b},a);(\tilde{a},a))$. We make a case analysis on the last rule.

- **case (decl):** we have $\langle u \Leftarrow P\rangle \stackrel{\mu}{\longmapsto} (P\ \tilde{a})$ for all $\tilde{b}$ and $\tilde{a}$. Therefore $\langle u \Leftarrow P\rangle \stackrel{\mu\, a}{\longmapsto} (P\ \tilde{a}\ a)$;
- **case (new mu):** we have $P = (\boldsymbol{\nu} u)P_1$ and $P_1 \stackrel{\mu}{\longmapsto} P_1'$ implies $P \stackrel{\mu}{\longmapsto} P' = (\boldsymbol{\nu} u)P_1'$. Therefore, since $a \neq u$, we can use the induction hypothesis and rule (new mu) to prove that $(\boldsymbol{\nu} u)P \stackrel{\mu\, a}{\longmapsto} (\boldsymbol{\nu} u)P_1''$ with $P_1'' \equiv (P_1'\ a)$. The result follows from $(\boldsymbol{\nu} u)P_1'' \equiv (\boldsymbol{\nu} u)(P_1'\ a) \equiv (\boldsymbol{\nu} u)P_1'\ a$. We have a similar proof in the case (def mu);
- **case (par in):** we have $P = (P_1 \mid Q_1)$ and $P_1 \stackrel{\mu}{\longmapsto} P_1'$ implies $P \stackrel{\mu}{\longmapsto} P' = (P_1' \mid (Q_1\ \tilde{b}))$. Using the induction hypothesis and rule (par in), it follows that $P \stackrel{\mu\, a}{\longmapsto} (P_1'' \mid (Q_1\ \tilde{b}\ a))$ with $P_1'' \equiv (P_1'\ a)$. The result follows from $(P_1'' \mid (Q_1\ \tilde{b}\ a)) \equiv ((P_1'\ a) \mid (Q_1\ \tilde{b}\ a)) \equiv (P_1' \mid (Q_1\ \tilde{b}))\ a$;
- **case (app in):** let ξ be the action $\mathbf{in}\, u.((c, \tilde{b}); \tilde{a})$. We have $P = (P_1\ c)$ and $P_1 \stackrel{\xi}{\longmapsto} P_1'$ implies $P \stackrel{\xi}{\longmapsto} P' = P_1'$. Using the induction hypothesis and rule (app in), it follows that $P \stackrel{\mu\, a}{\longmapsto} P_1''$ with $P_1'' \equiv (P_1'\ a)$, which is the expected result.

□

We now prove the main result of this appendix, that is: if $\mathscr{D}$ and $\mathscr{D}^{-1}$ are bisimulations up-to F, then $\mathscr{D}_\infty$ is a ground simulation. More precisely, we simultaneously prove two properties: if $(P, Q) \in F^k(\mathscr{D})$, then

- if $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$, then there exists a process Q' such that: $Q \stackrel{\mu}{\longmapsto} Q'$, and $P'\, \mathscr{D}_\infty\, Q'$;
- if $P \stackrel{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P')$, then there exists a process Q' and an environment D' such that: $Q \stackrel{\mathbf{out}\, u.(\tilde{v};\, \tilde{c})}{\longmapsto} (D'; Q')$, with $P'\, \mathscr{D}_\infty\, Q'$, and $|\tilde{a}| = |\tilde{c}|$ (say n), and for all $i \in [1..n]$ we have $(\mathbf{def}\ D\ \mathbf{in}\ a_i)\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ c_i)$.

This proof is made by induction on k and on the definition of $F(X)$.

Since $F^0(X) = \mathcal{I}d$, the case for $k = 0$ is straightforward: the identity relation is clearly a bisimulation. For the case $k = (n + 1)$, the proof is divided along the three main subsets of $F(X)$. Note that, using the property of $\mathscr{E}$ and $G(X)$ given at the beginning of this appendix, it is easy to prove that $P \equiv Q$ implies $P\, \mathscr{D}_\infty\, Q$. Likewise, using the property of $G(X)$, it is easy to prove that $\mathscr{D}_\infty$ is compositional.

A.1 Rules for Congruence

case $P\, \mathscr{D}\, Q$ or $P \sim_d Q$: by definition, the relations $\mathscr{D}$ and $\sim_d$ are included in $\mathscr{D}_\infty$. Therefore the results follows from the definition of the bisimulation up-to F.

case $(P, Q) \in F^n(\mathscr{D})$ and $(Q, R) \in F^n(\mathscr{D})$: suppose that $P \stackrel{\mu}{\longmapsto} P'$ and $\kappa(\mu) \neq \mathbf{out}$. We use the induction hypothesis to prove that $R \stackrel{\mu}{\longmapsto} R'$ and $P'\, \mathscr{D}_\infty\, R'$, and the induction hypothesis again to prove that $Q \stackrel{\mu}{\longmapsto} Q'$ and $R'\, \mathscr{D}_\infty\, Q'$. The result follows from the transitivity of $\mathscr{D}_\infty$. The proof is similar in the case of output actions.

case $P = (\boldsymbol{\nu}\, u)P_1$ **and** $Q = (\boldsymbol{\nu}\, u)Q_1$**:**

- **case** $\kappa(\mu) \neq$ **out:** the last rule for inferring $P \overset{\mu}{\longmapsto} P'$ is (new mu), and $P' = (\boldsymbol{\nu}\, u)P_1'$ with $P_1 \overset{\mu}{\longmapsto} P_1'$. Using the induction hypothesis, it follows that $Q_1 \overset{\mu}{\longmapsto} Q_1'$ with $(P_1', Q_1') \in \mathscr{D}_\infty$ and, using rule (new mu), there exists a transition $Q \overset{\mu}{\longmapsto} (\boldsymbol{\nu}\, u)Q_1'$. The result follows from $(\boldsymbol{\nu}\, u)P_1' \; \mathscr{D}_\infty \; (\boldsymbol{\nu}\, u)Q_1'$;

- **case** $\kappa(\mu) =$ **out:** if the last rule is (new out), then there exists a process P_1' and an environment D such that $P' = (D; (\boldsymbol{\nu}\, u)P_1')$ and $P_1 \overset{\mu}{\longmapsto} (D; P_1')$. Using the induction hypothesis, it follows that $Q_1 \overset{\mu}{\longmapsto} (D'; Q_1')$ with $(P_1', Q_1') \in \mathscr{D}_\infty$. Like in the previous case, the result follows using rule (new out) and the fact that $(P_1', Q_1') \in \mathscr{D}_\infty$ implies $((\boldsymbol{\nu}\, u)P_1', (\boldsymbol{\nu}\, u)Q_1') \in \mathscr{D}_\infty$;

  Suppose the last rule is (new open). Thus $P_1 \overset{\mathbf{out}\; v.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1')$ and $u \neq v$. Using the induction hypothesis, it follows that $Q_1 \overset{\mathbf{out}\; v.(\tilde{v};\, \tilde{c})}{\longmapsto} (D'; Q_1')$ with: $(P_1' \; \mathscr{D}_\infty \; Q_1')$ and $(\mathbf{def}\; D \;\mathbf{in}\; \tilde{a}) \; \mathscr{D}_\infty \; (\mathbf{def}\; D' \;\mathbf{in}\; \tilde{c})$, that is the expected result.

case $P = (P_1\ a)$ **and** $Q = (Q_1\ a)$**:** we make a case analysis on the last rule of the transition $P \overset{\mu}{\longmapsto} P'$. Suppose that $P_1 \overset{\mu_1}{\longmapsto} P_1'$ (with $\kappa(\mu_1) \neq$ **out**):

- **case (app tau):** we have $\mu = \mu_1 = \tau$. We use the induction hypothesis to prove that $Q_1 \overset{\tau}{\longmapsto} Q_1'$ with $P_1' \; \mathscr{D}_\infty \; Q_1'$. The result follows applying rule (app tau);

- **case (app com):** we have $\mu = \tau$ and $\mu_1 = \lambda\, a$. Using the induction hypothesis, we prove that there exists a transition $Q_1 \overset{\mu_1}{\longmapsto} Q_1'$ with $P_1' \; \mathscr{D}_\infty \; Q_1'$. The result follows from rule (app com);

- **case (app in):** we have $\mu_1 = \mathbf{in}\, u.((a, \tilde{b}); \tilde{a})$ and $\mu = \mathbf{in}\, u.(\tilde{b}; \tilde{a})$. Using the induction hypothesis, there exists a transition $Q_1 \overset{\mu_1}{\longmapsto} Q_1'$ with $P_1' \; \mathscr{D}_\infty \; Q_1'$. The result follows from rule (app in);

- **case (app out):** we have $P_1 \overset{\mathbf{out}\; u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1')$ implies $P \overset{\mathbf{out}\; u.(\tilde{v};\, (\tilde{a},\, c))}{\longmapsto} (D; (P_1'\ c))$. Using the induction hypothesis, it follows that $Q_1 \overset{\mathbf{out}\; u.(\tilde{v};\, \tilde{c})}{\longmapsto} (D'; Q_1')$ with $P_1' \; \mathscr{D}_\infty \; Q_1'$ and $(\mathbf{def}\; D \;\mathbf{in}\; \tilde{a}) \; \mathscr{D}_\infty \; (\mathbf{def}\; D' \;\mathbf{in}\; \tilde{c})$. The result follows from rule (app out) and the fact that $a \notin (\mathbf{def}(D) \cup \mathbf{def}(D'))$ implies $(\mathbf{def}\; D \;\mathbf{in}\; a) \; \mathscr{D}_\infty \; a \; \mathscr{D}_\infty \; (\mathbf{def}\; D' \;\mathbf{in}\; a)$.

case $P = (\boldsymbol{\lambda} x)P_1$ **and** $Q = (\boldsymbol{\lambda} x)Q_1$**:** the last rule is necessarily (lambda). The result follows from the fact that $\mathscr{D}_\infty$ is closed by instantiation. The proof is similar in the case $P = [\, P_1\ ,\ l = P_2\,]$ and $Q = [\, Q_1\ ,\ l = Q_2\,]$ and in the case $P = \langle u \Leftarrow P_1 \rangle$ and $Q = \langle u \Leftarrow Q_1 \rangle$.

case $P = (P_1 \mid P_2)$ **and** $Q = (Q_1 \mid Q_2)$**:** If the last rule of the judgment $P \overset{\mu}{\longmapsto} P'$ is (par tau), (par lambda), (par in) or (par out), the result follows from the induction hypothesis and the fact that $(P, Q) \in \mathscr{D}_\infty$ implies $(P\ \tilde{b}, Q\ \tilde{b}) \in \mathscr{D}_\infty$ for any tuple of names $\tilde{b}$. The only interesting case is such that the last rule used is (par com). Suppose that $P_1 \overset{\mathbf{out}\; u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1')$ and $P_2 \overset{\mathbf{in}\; u.(\epsilon;\, \tilde{a})}{\longmapsto} P_2'$ and $P' = (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\; D \;\mathbf{in}\; (P_1' \mid P_2'))$. Using

the induction hypothesis, it follows that $Q_1 \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{c})}{\longmapsto} (D'; Q_1')$ with $(\dagger) : P_1' \mathscr{D}_\infty Q_1'$ and $(\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{c})$. Using Lemma A.1, this last property implies that there exists P_2'' such that:

$$P_2 \stackrel{\mathbf{in}\, u.(\epsilon;\tilde{c})}{\longmapsto} P_2'' \quad \text{with} \quad (\mathbf{def}\, D\, \mathbf{in}\, P_2')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P_2'')$$

Finally, we use the induction hypothesis on the pair (P_2, Q_2) to prove that: $Q_2 \stackrel{\mathbf{in}\, u.(\epsilon;\tilde{c})}{\longmapsto} Q_2'$ with $Q_2' \mathscr{D}_\infty P_2''$, which implies that:

$$(\ddagger) : (\mathbf{def}\, D'\, \mathbf{in}\, Q_2')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, P_2'')\, \mathscr{D}_\infty\, (\mathbf{def}\, D\, \mathbf{in}\, P_2')$$

Therefore there is a τ-transition from Q such that: $Q \stackrel{\tau}{\longmapsto} Q' = (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\, D'\, \mathbf{in}\, (Q_1' \mid Q_2'))$. Since the variables in $\mathbf{def}(D)$ (resp. $\mathbf{def}(D')$) are not free in P_1' (resp. Q_1'), we have that

$$\begin{array}{lll} P' = (\mathbf{def}\, D\, \mathbf{in}\, (P_1' \mid P_2')) & \mathscr{D}_\infty & (P_1' \mid \mathbf{def}\, D\, \mathbf{in}\, P_2') \\ & \mathscr{D}_\infty & (Q_1' \mid \mathbf{def}\, D'\, \mathbf{in}\, Q_2')\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, (Q_1' \mid Q_2')) = Q' \end{array}$$

In this deduction, we also use the properties $(\dagger)$ and $(\ddagger)$, and the fact that $\mathscr{D}_\infty$ is a congruence.

case $P = (\mathbf{def}\, p = P_1\, \mathbf{in}\, P_2)$ **and** $Q = (\mathbf{def}\, p = Q_1\, \mathbf{in}\, Q_2)$**:** The interesting cases are for the rule (def open) and (def com).

- **case (def open):** suppose that $P_2 \stackrel{\mathbf{out}\, p.(\tilde{v};\tilde{a})}{\longmapsto} (D; P_2')$ and that the open transition is:

  $$P \stackrel{\mathbf{out}\, p.(\tilde{v};\tilde{a})}{\longmapsto} ((p = P_1, D) \mid P_2')$$

  Using the induction hypothesis, there exists a transition $Q_2 \stackrel{\mathbf{out}\, p.(\tilde{v};\tilde{c})}{\longmapsto} (D'; Q_2')$ with

  $$(i) : (\mathbf{def}\, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, D'\, \mathbf{in}\, \tilde{c}) \qquad (ii) : P_2' \mathscr{D}_\infty Q_2' \qquad (iii) : P_1 \mathscr{D}_\infty Q_1$$

  The result follows from the fact that (i) and (iii) implies:

  $$(\mathbf{def}\, p = P_1, D\, \mathbf{in}\, \tilde{a})\, \mathscr{D}_\infty\, (\mathbf{def}\, p = Q_1, D'\, \mathbf{in}\, \tilde{c})$$

  and that (ii) and (iii) implies: $(\mathbf{def}\, p = P_1\, \mathbf{in}\, P_2')\, \mathscr{D}_\infty\, (\mathbf{def}\, p = Q_1\, \mathbf{in}\, Q_2')$;

- **case (def com):** suppose that $P_2 \stackrel{\mathbf{out}\, p.(\tilde{v};\tilde{a})}{\longmapsto} (D; P_2')$, and that the τ-transition is:

  $$P \stackrel{\tau}{\longmapsto} \mathbf{def}\, p = P_1\, \mathbf{in}\, (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (P_1\ \tilde{a} \mid P_2'))$$

  Using the induction hypothesis, there exists a transition from Q_2 such that $Q_2 \stackrel{\mathbf{out}\, p.(\tilde{v};\tilde{c})}{\longmapsto} (D'; Q_2')$ with the properties (i), (ii) and (iii) given in the previous case. Therefore we can exhibit a τ transition from Q such that:

  $$Q \stackrel{\tau}{\longmapsto} \mathbf{def}\, p = Q_1\, \mathbf{in}\, (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\, D'\, \mathbf{in}\, (Q_1\ \tilde{c} \mid Q_2'))$$

Since the variables in $\mathbf{def}(D)$ (resp. $\mathbf{def}(D')$) are not free in P_1 and P_2' (resp. Q_1 and Q_2'), we have that:

$$\begin{array}{lcl} (\mathbf{def}\ D\ \mathbf{in}\ (P_1\ \tilde{a}\ |\ P_2')) & \mathscr{D}_\infty & ((\mathbf{def}\ D\ \mathbf{in}\ (P_1\ \tilde{a}))\ |\ P_2') \\ (\mathbf{def}\ D'\ \mathbf{in}\ (Q_1\ \tilde{c}\ |\ Q_2')) & \mathscr{D}_\infty & ((\mathbf{def}\ D'\ \mathbf{in}\ (Q_1\ \tilde{a}))\ |\ Q_2') \end{array}$$

Using Lemma A.1-(i), we conclude that $(\mathbf{def}\ D'\ \mathbf{in}\ (Q_1\ \tilde{c}))\ \mathscr{D}_\infty\ (\mathbf{def}\ D\ \mathbf{in}\ (P_1\ \tilde{a}))$. The result follows from the fact that $\mathscr{D}_\infty$ is a congruence.

A.2 Distribution Laws

case $P = \mathbf{def}\ D\ \mathbf{in}\ (P_1\ a)$ and $Q = \mathbf{def}\ D'\ \mathbf{in}\ (P_1\ b)$: with the assumption that $(\mathbf{def}\ D\ \mathbf{in}\ u)\ F^n(\mathscr{D})\ (\mathbf{def}\ D'\ \mathbf{in}\ v)$, and that $\mathbf{fn}(P_1) \cap (\mathbf{def}(D) \cup \mathbf{def}(D')) = \emptyset$. It is easy to prove that, with this property, $P\ \mathscr{D}_\infty\ Q$ implies $(\mathbf{def}\ D\ \mathbf{in}\ (P\ a))\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ (Q\ b))$. Suppose that $P_1 \stackrel{\mu_1}{\longmapsto} P_1'$. We make a case analysis on the transition rule applied to reduce $(P\ a)$:

- **case (app tau):** the result follows from $(\mathbf{def}\ D\ \mathbf{in}\ (P_1'\ a), \mathbf{def}\ D'\ \mathbf{in}\ (P_1'\ b)) \in \mathscr{D}_\infty$ and the fact that $(\dagger) : \mathbf{fn}(\mathbf{def}\ D\ \mathbf{in}\ (P_1'\ a)) \cap \mathbf{def}(D) = \mathbf{fn}(\mathbf{def}\ D'\ \mathbf{in}\ (P_1'\ b)) \cap \mathbf{def}(D') = \emptyset$;
- **case (app com):** we have $\mu_1 = \lambda a$ and $(P_1\ a) \stackrel{\tau}{\longmapsto} P_1'$. Therefore, using Lemma A.1-(iii), we conclude that there exists P_1'' such that $(\ddagger) : P_1 \stackrel{\lambda b}{\longmapsto} P_1''$ and $((\mathbf{def}\ D\ \mathbf{in}\ P_1')\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ P_1''))$. The result follows from the fact that $(\ddagger)$ implies $(P_1\ b) \stackrel{\tau}{\longmapsto} P_1''$, and from property $(\dagger)$;
- **case (app in) and (app out):** in the first case we have $\mu_1 = \mathbf{in}\, c.(a, \tilde{b}; \tilde{a})$ and $(P_1\ a) \stackrel{\mathbf{in}\, c.(\tilde{b};\, \tilde{a})}{\longmapsto} P_1'$. Therefore, using Lemma A.1-(ii), there exists a process P_1'' such that $P_1 \stackrel{\mathbf{in}\, c.(b,\, \tilde{b};\, \tilde{a})}{\longmapsto} P_1''$ such that $(\mathbf{def}\ D\ \mathbf{in}\ P_1')\ \mathscr{D}_\infty\ (\mathbf{def}\ D'\ \mathbf{in}\ P_1'')$. The result follows from property $(\dagger)$. Proof is similar in the case of rule (app out): in this case we also use the fact that $\mathbf{fn}(P_1) \cap \mathbf{def}(D) = \emptyset$ and $P_1 \stackrel{\mathbf{out}\, c.(\tilde{v};\, \tilde{a})}{\longmapsto} P_1'$ implies $\tilde{a} \cap \mathbf{def}(D) = \emptyset$.

case $P = \mathbf{def}\ p = R\ \mathbf{in}\ (P_1 \mid Q_1)$ and $Q = \mathbf{def}\ p = R\ \mathbf{in}\ ((\mathbf{def}\ p = R\ \mathbf{in}\ P_1) \mid Q_1)$: we suppose that $P_1 \stackrel{\mu_1}{\longmapsto} P_1'$ and we make a case analysis on the last two rules of the derivation $P \stackrel{\mu}{\longmapsto} P'$. Since the role of Q_1 is symmetric, we will not study the cases where the transition comes from Q_1.

- **case (par tau) followed by (def mu):** we have $\mu_1 = \tau$ and $P' = \mathbf{def}\ p = R\ \mathbf{in}\ (P_1' \mid Q_1)$. Therefore, using rule (def mu), (par tau) and (def mu), we can exhibit a transition

$$Q \stackrel{\tau}{\longmapsto} \mathbf{def}\ p = R\ \mathbf{in}\ (\mathbf{def}\ p = R\ \mathbf{in}\ P_1' \mid Q_1)$$

  The proof is similar in the case (par lambda) followed by (def mu);

- **case (par in) followed by (def mu):** we have $\mu_1 = \mathbf{in}\, u.(\tilde{b};\tilde{a})$ and $P' = \mathbf{def}\, p = R\, \mathbf{in}\, (P_1' \mid Q_1\ \tilde{b})$. Since the last rule is (def mu), the names $\tilde{a}$ must be different from p (this is also true for $\tilde{b}$). Therefore, using rule (def mu), (par in) and (def mu), we can exhibit a transition

$$Q \stackrel{\mu}{\longmapsto} \mathbf{def}\, p = R\, \mathbf{in}\, (\mathbf{def}\, p = R\, \mathbf{in}\, P_1' \mid Q_1\ \tilde{b})$$

  The proof is similar in the case (par out) followed by (def mu);

- **case (par com) followed by (def mu):** the most interesting case is such that $P_1 \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D; P_1')$ with $p \in \tilde{a}$. In this case, we exhibit a transition from Q using rule (def open), (par com) and (def mu). In particular we have

$$\begin{cases} P' = \mathbf{def}\, p = R\, \mathbf{in}\, ((\nu\, \tilde{v})(\mathbf{def}\, D\, \mathbf{in}\, (P_1 \mid Q_1))) \\ Q' = \mathbf{def}\, p = R\, \mathbf{in}\, ((\nu\, \tilde{v})(\mathbf{def}\, p = R, D\, \mathbf{in}\, ((\mathbf{def}\, p = R\, \mathbf{in}\, P_1) \mid Q_1))) \end{cases}$$

  The result follows from $P'\, \mathscr{D}_\infty\, Q'$. In the case $p \notin \tilde{a}$, we exhibit a transition from Q using rule (def out), (par com) and (def mu);

- **case (par out) followed by (def open):** we have $(P_1 \mid Q_1) \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} (D; (P_1' \mid Q_1))$ and

$$P \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} P' = ((p = R, D) \mid \mathbf{def}\, p = R\, \mathbf{in}\, (P_1' \mid Q_1))$$

  Thus, using rule (def open), (par out) and (def mu), we can exhibit a transition from Q such that

$$Q \stackrel{\mathbf{out}\, u.(\tilde{v};\tilde{a})}{\longmapsto} Q' = ((p = R, D) \mid \mathbf{def}\, p = R\, \mathbf{in}\, ((\mathbf{def}\, p = R\, \mathbf{in}\, P_1') \mid Q_1))$$

  The result follows from $P'\, \mathscr{D}_\infty\, Q'$ and the fact that, for all $P\, \mathscr{D}_\infty\, Q$:

$$(\mathbf{def}\, p = R\, \mathbf{in}\, P)\, \mathscr{D}_\infty\, (\mathbf{def}\, p = R\, \mathbf{in}\, Q)$$

- **case (par out) followed by (def com):** we have $\mu = \tau$ and $\mu_1 = \mathbf{out}\, p.(\tilde{v};\tilde{a})$ and $P' = \mathbf{def}\, p = R\, \mathbf{in}\, (R\ \tilde{a} \mid (P_1' \mid Q))$. Thus, using rule (def com), (par tau) and (def mu) we can exhibit a transition

$$\begin{array}{lll} Q \stackrel{\tau}{\longmapsto} Q' & = & \mathbf{def}\, p = R\, \mathbf{in}\, (\mathbf{def}\, p = R\, \mathbf{in}\, (R\ \tilde{a} \mid P_1') \mid Q) \\ & \mathscr{D}_\infty & \mathbf{def}\, p = R\, \mathbf{in}\, ((\mathbf{def}\, p = R\, \mathbf{in}\, (R\ \tilde{a}) \mid \mathbf{def}\, p = R\, \mathbf{in}\, (P_1')) \mid Q) \\ & \mathscr{D}_\infty & P' \end{array}$$

The cases such that the "reduction comes form Q_1", that is such that $Q_1 \stackrel{\mu_1}{\longmapsto} Q_1'$, are similar and use the fact that $((\mathbf{def}\, p = R\, \mathbf{in}\, P)\ a)\, \mathscr{D}_\infty\, (\mathbf{def}\, p = R\, \mathbf{in}\, (P\ a))$.

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1\ p)$ **and** $Q = \mathbf{def}\, p = R \,\mathbf{in}\, ((\mathbf{def}\, p = R \,\mathbf{in}\, P_1)\ p)$**:** we suppose that $P_1 \stackrel{\mu_1}{\longmapsto} P_1'$ and we make a case analysis on the last two rules of the derivation $P \stackrel{\mu}{\longmapsto} P'$:

- **case (app tau) followed by (def mu):** we have $\mu = \mu_1 = \tau$ and $P' = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1'\ p)$. Therefore, using rule (def mu), (par tau) and (def mu), we can exhibit a transition

$$Q \stackrel{\tau}{\longmapsto} \mathbf{def}\, p = R \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P_1'\ p)$$

- **case (app com) followed by (def mu):** we have $\mu = \tau$, $\mu_1 = \lambda p$ and $P' = \mathbf{def}\, p = R \,\mathbf{in}\, P_1'$. We forgot here the proof that there exist a process P_1'' such that for all name b: $P_1 \stackrel{\lambda v}{\longmapsto} P_1''\{v/x\}$. In particular, this implies $P_1' = P_1''\{p/x\}$.

  To avoid name clashes, we use a fresh name q and we sometimes convert the process $(\mathbf{def}\, p = R \,\mathbf{in}\, P)$ to $(\mathbf{def}\, q = R\{q/p\} \,\mathbf{in}\, P\{q/p\})$. Remember that, following the remark in Sect. 4, we consider that α-equivalence is implicit.

  Using rule (def mu), (app tau) and (def mu) we can exhibit the transitions

$$P \stackrel{\tau}{\longmapsto} P' = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1''\{p/x\})$$
$$Q \stackrel{\tau}{\longmapsto} Q' = \mathbf{def}\, p = R \,\mathbf{in}\, (\mathbf{def}\, q = R\{q/p\} \,\mathbf{in}\, (P_1''\{q/p\}\{p/x\}))$$

  The result follows from the fact that $\mathscr{D}_\infty$ contains the set of all the pairs:

$$((\mathbf{def}\, D \,\mathbf{in}\, P\{u/x\}), (\mathbf{def}\, D' \,\mathbf{in}\, P\{v/x\}))$$

  with the side conditions that $(\mathbf{def}\, D \,\mathbf{in}\, u)\ \mathscr{D}_\infty\ (\mathbf{def}\, D' \,\mathbf{in}\, v)$;

- **case (app in) followed by (def mu):** we have $\mu_1 = \mathbf{in}\, u.(p, \tilde{b}; \tilde{a})$ and $P' = \mathbf{def}\, p = R \,\mathbf{in}\, P_1'$. Since the last rule is (def mu), the names $\tilde{a}$ must be different from p (this is also true for $\tilde{b}$). Therefore, using rule (def mu), (app in) and (def mu), we can exhibit a transition

$$Q \stackrel{\mu}{\longmapsto} \mathbf{def}\, p = R \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P_1')\ \mathscr{D}_\infty\ (\mathbf{def}\, p = R \,\mathbf{in}\, P_1') = P'$$

- **case (app out) followed by (def open):** we have $P_1 \stackrel{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1')$ and

$$P \stackrel{\mathbf{out}\, u.(\tilde{v};\, (\tilde{a},\, p))}{\longmapsto} P' = ((p = R, D); \mathbf{def}\, p = R \,\mathbf{in}\, (P_1'\ p))$$

  The interesting case is such that p was already in $\tilde{a}$. Thus, using rule (def open), (app out) and (def open), we can exhibit a transition from Q such that (we choose a fresh name q to replace the bound name p inside P_1 and we denote R_q the process $R\{q/p\}$)

$$Q \stackrel{\mathbf{out}\, u.(\tilde{v};\, \tilde{a},\, q)}{\longmapsto} Q' = ((q = R_q, p = R, D) \mid \mathbf{def}\, q = R_q \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P_1'))$$

  The result follows from $(\mathbf{def}\, q = R_q \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P_1'))\ \mathscr{D}_\infty\ (\mathbf{def}\, p = R \,\mathbf{in}\, P_1')$ and the fact that:

$$(\mathbf{def}\, p = R, D \,\mathbf{in}\, (\tilde{a}, p))\ \mathscr{D}_\infty\ (\mathbf{def}\, q = R_q, p = R, D \,\mathbf{in}\, (\tilde{a}, q))$$

The proof is similar in the case $P = \mathbf{def}\, p = R, q = S \,\mathbf{in}\, P$ and $Q = \mathbf{def}\, p = R, q = S \,\mathbf{in}\, (\mathbf{def}\, p = R \,\mathbf{in}\, P)$, and in the case $P = \mathbf{def}\, p = R, q = S \,\mathbf{in}\, P$ and $Q = \mathbf{def}\, p = R, q = (\mathbf{def}\, p = R \,\mathbf{in}\, S) \,\mathbf{in}\, P$.

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, ((\lambda x)P_1)$ **and** $Q = (\lambda x)(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)$**:** the side condition is $x \notin \mathbf{fn}(R)$. The last two rules of the derivation $P \stackrel{\mu}{\longmapsto} P'$ have to be (lambda) followed by (def mu). Therefore $\mu = \lambda\, u$ and $P' = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1\{u/x\})$. The result follows from rule (lambda) and the fact that $x \notin \mathbf{fn}(R)$ implies $(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)\{u/x\} = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1\{u/x\})$. In the symmetric case, we use the fact that the derivation $Q \stackrel{\mu}{\longmapsto} Q'$ use the rule (lambda).

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, (\langle u \Leftarrow P_1\rangle)$ **and** $Q = \langle u \Leftarrow (\mathbf{def}\, p = R \,\mathbf{in}\, P_1)\rangle$**:** we use the fact that the last two rules of $P \stackrel{\mu}{\longmapsto} P'$ have to be (decl) followed by (def mu). Therefore $\langle u \Leftarrow P_1\rangle \stackrel{\mathbf{in}\, u.(\tilde{b};\,\tilde{a})}{\longmapsto} (P\ \tilde{a})$ and, since the last rule is (def mu), $p \notin \tilde{b} \cup \tilde{a}$. The result follows from the fact that $p \notin \tilde{a}$ implies $(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)\ \tilde{a} \equiv \mathbf{def}\, p = R \,\mathbf{in}\, (P_1\ \tilde{a})$.

A.3 Rules for Structural Equivalence

case $P = P_1 \mid \mathtt{0}$ **and** $Q = P_1$**:** if $\kappa(\mu) \neq \mathbf{out}$, the last rule of $P \stackrel{\mu}{\longmapsto} P'$ is (par tau), (par lambda) or (par in), it is (par out) otherwise. The result follows from $(\mathtt{0}\ \tilde{a})\ \mathscr{D}_\infty\ \mathtt{0}$ and $(P \mid \mathtt{0})\ \mathscr{D}_\infty\ P$. The proof of the symmetric case is similar.

case $P = P_1 \mid Q_1$ **and** $Q = Q_1 \mid P_1$**:** the only interesting case is rule (par com). Suppose that $P_1 \stackrel{\mathbf{out}\, u.(\tilde{v};\,\tilde{a})}{\longmapsto} (D; P_1')$, $Q_1 \stackrel{\mathbf{in}\, u.(\epsilon;\,\tilde{a})}{\longmapsto} Q_1'$ and $P \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (P_1' \mid Q_1'))$. Therefore we can exhibit a transition for Q such that $Q \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (Q_1' \mid P_1'))$. The result follows from the fact that $P\ \mathscr{D}_\infty\ Q$ implies $(\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, P)\ \mathscr{D}_\infty\ (\boldsymbol{\nu}\,\tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, Q)$. The proof is similar in the case $P = P_1 \mid (Q_1 \mid R_1)$ and $Q = (P_1 \mid Q_1) \mid R_1$.

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, Q$ **and** $p \notin \mathbf{fn}(Q)$**:**

- **case** $\kappa(\mu) \neq \mathbf{out}$**:** thus the last rule of $P \stackrel{\mu}{\longmapsto} P'$ is (def mu) and $Q \stackrel{\mu}{\longmapsto} Q'$ implies $P \stackrel{\mu}{\longmapsto} \mathbf{def}\, p = R \,\mathbf{in}\, Q'$. Since $p \notin \mathbf{fn}(Q)$ implies that $p \notin \mathbf{fn}(Q')$, the result follows from $(\mathbf{def}\, p = R \,\mathbf{in}\, Q')\ \mathscr{D}_\infty\ Q'$;
- **case** $\kappa(\mu) = \mathbf{out}$**:** let μ be the action $\mathbf{out}\, u.(\tilde{v};\tilde{a}))$. Since $p \notin \mathbf{fn}(Q)$, the last rule cannot be (def com). Suppose the last rule is (def out), we have: $Q \stackrel{\mu}{\longmapsto} (D'; Q')$ implies $P \stackrel{\mu}{\longmapsto} (D'; \mathbf{def}\, p = R \,\mathbf{in}\, Q')$. The result follows from the fact that $(\mathbf{def}\, p = R \,\mathbf{in}\, Q')\ \mathscr{D}_\infty\ Q'$ and that $(\mathbf{def}\, D' \,\mathbf{in}\, \tilde{a})\ \mathscr{D}_\infty\ (\mathbf{def}\, D' \,\mathbf{in}\, \tilde{a})$. Suppose the last rule is (open def). We have $P \stackrel{\mu}{\longmapsto} ((p = R, D') \mid Q')$. Since $p \notin \mathbf{fn}(Q)$, we conclude that $p \notin \tilde{a}$ and $p \notin \mathbf{fn}(D')$. Therefore $(\mathbf{def}\, p = R, D' \,\mathbf{in}\, \tilde{a})\ \mathscr{D}_\infty\ (\mathbf{def}\, D' \,\mathbf{in}\, \tilde{a})$ (which is the expected result).

The proof is similar in the case $P = (\boldsymbol{\nu}\, u)Q$ and $u \notin \mathbf{fn}(Q)$.

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu}\, u)P_1)$ **and** $Q = (\boldsymbol{\nu}\, u)(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)$**:** the side condition is that $u \notin \mathbf{fn}(R)$. Suppose that $P_1 \stackrel{\mu_1}{\longmapsto} P_1'$ (with $\kappa(m_1) \neq \mathbf{out}$). Thus the last two rules of the transition $P \stackrel{\mu}{\longmapsto} P'$ are (new mu) followed by (def mu). Thus we can exhibit an equivalent transition from Q using rule (def mu) and thus rule (new mu). The proof is similar if $P_1 \stackrel{\mathbf{out}\, u.(\tilde{v};\,\tilde{a})}{\longmapsto} (D; P_1')$. The proof is similar in the symmetric case, that is if $P =$

$(\boldsymbol{\nu} u)(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)$ and $Q = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} u)P_1)$. It is also similar in the case: $P = (\mathbf{def}\, p = R, q = S \,\mathbf{in}\, P)$ and $Q = (\mathbf{def}\, q = S, p = R \,\mathbf{in}\, P)$, and in the case: $P = (\boldsymbol{\nu} u)((\boldsymbol{\nu} v)P)$ and $Q = (\boldsymbol{\nu} v)((\boldsymbol{\nu} u)P)$.

case $P = ((\boldsymbol{\nu} u)P_1 \mid Q_1)$ **and** $Q = (\boldsymbol{\nu} u)(P_1 \mid Q_1)$**:** the most interesting case is such that the last rule of $P \overset{\mu}{\longmapsto} P'$ is (par com). We consider the case such that P_1 makes an **out** transition (with the last rule being (new out)) and Q_1 makes an **in** transition. The proof is similar if the transition from P_1 ends with rule (open new). All the other cases are similar to the case $P = ((\boldsymbol{\nu} u)P_1\ a)$. Suppose that:

$$(\boldsymbol{\nu} u)P_1 \overset{\mathbf{out}\, p\, .\, (\tilde{v};\, \tilde{a})}{\longmapsto} (D; (\boldsymbol{\nu} u)(P_1')) \qquad P' = (\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, ((\boldsymbol{\nu} u)(P_1') \mid Q_1'))$$

Therefore, we can exhibit a "matching transition" from Q such that:

$$Q \overset{\tau}{\longmapsto} Q' = (\boldsymbol{\nu} u)((\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (P_1' \mid Q_1')))$$

The result follows from the fact that the processes $P' \mathscr{D}_\infty Q'$ (since they are structurally equivalent). The proof of the symmetric case is similar.

case $P = \mathbf{def}\, p = R \,\mathbf{in}\, P_1 \mid Q_1$ **and** $Q = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1 \mid Q_1)$**:** the only interesting case in such that the last two rules of $P \overset{\mu}{\longmapsto} P'$ are (def com) followed by (par tau). In every other cases we are in a situation similar to the previous proof. Thus we have: $\mu = \tau$ and:

$$P_1 \overset{\mathbf{out}\, p\, .\, (\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1') \qquad P' = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (R\ \tilde{a} \mid P_1'))) \mid Q_1$$

Therefore we can exhibit a transition

$$Q \overset{\tau}{\longmapsto} Q' = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (R\ \tilde{a} \mid P_1' \mid Q_1)))$$

The result follows from the fact that $P' \mathscr{D}_\infty Q'$ (since they are structurally equivalent). The proof is similar in the symmetric case. The proof is also similar in the case $P = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} u)P_1)$ and $Q = (\boldsymbol{\nu} u)(\mathbf{def}\, p = R \,\mathbf{in}\, P_1)$.

case $P = (\mathbf{def}\, p = R \,\mathbf{in}\, P_1)\ a$ **and** $Q = \mathbf{def}\, p = R \,\mathbf{in}\, (P_1\ a)$**:** the side condition is that $a \neq p$. The only interesting case in such that the last two rules of $P \overset{\mu}{\longmapsto} P'$ are (tau def) followed by (app tau). In every other cases we are in a situation similar to the previous demonstration. Thus we have: $\mu = \tau$ and:

$$P_1 \overset{\mathbf{out}\, p\, .\, (\tilde{v};\, \tilde{a})}{\longmapsto} (D; P_1') \qquad P' = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (R\ \tilde{a} \mid P_1')))\ a$$

Therefore we can exhibit a transition $Q \overset{\tau}{\longmapsto} Q'' = \mathbf{def}\, p = R \,\mathbf{in}\, ((\boldsymbol{\nu} \tilde{v})(\mathbf{def}\, D \,\mathbf{in}\, (R\ \tilde{a}\ a \mid P_1'\ a)))$. The result follows from the fact that $P' \mathscr{D}_\infty Q'$ (since they are structurally equivalent). The proof is similar in the symmetric case.

case $P = (\boldsymbol{\nu} u)P_1\ a$ **and** $Q = (\boldsymbol{\nu} u)(P_1\ a)$**:** we consider the last two rules of $P \stackrel{\mu}{\longmapsto} P'$. If $\kappa(\mu) \neq \mathbf{out}$, the last two rules are (new mu) followed by (app lambda), (app tau) or (app in). In the other case, it is (new out) or (new open) followed by (app out).

- **case (new mu) followed by (app com):** in this case: $\mu = \tau$, $P_1 \stackrel{\lambda\, a}{\longmapsto} P_1'$, $P' = (\boldsymbol{\nu} u)P_1'$ and we can exhibit a transition $Q \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu} u)P_1'$. The result follows from the fact that $(\boldsymbol{\nu} u)P_1' \mathscr{D}_\infty (\boldsymbol{\nu} u)P_1'$. The proof is similar in the case (new mu) followed by (app in);
- **case (new mu) followed by (app tau):** in this case: $\mu = \tau$, $P_1 \stackrel{\tau}{\longmapsto} P_1'$, $P' = (\boldsymbol{\nu} u)P_1'\ a$ and we can exhibit a transition $Q \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu} u)(P_1'\ a)$. The result follows from the fact that $((\boldsymbol{\nu} u)P_1'\ a)\ \mathscr{D}_\infty\ (\boldsymbol{\nu} u)(P_1'\ a)$;
- **case (new out) followed by (app out):** in this case: $\kappa(\mu) = \mathbf{out}$, $P_1 \stackrel{\mu}{\longmapsto} (D; P_1')$, $P' = (D; ((\boldsymbol{\nu} u)P_1'\ a))$ and we can exhibit a transition $Q \stackrel{\mu}{\longmapsto} (D; (\boldsymbol{\nu} u)(P_1'\ a))$. The result follows from $((\boldsymbol{\nu} u)P_1'\ a)\ \mathscr{D}_\infty\ (\boldsymbol{\nu} u)(P_1'\ a)$. The proof is similar in the case (new open) followed by (app out).

The proof of the symmetric case is similar.

case $P = (P_1 \mid Q_1)\ a$ **and** $Q = (P_1\ a) \mid (Q_1\ a)$**:** we study the last two rules of the derivation $P \stackrel{\mu}{\longmapsto} P'$. The first case is such that the transition comes "only from" P_1 (respectively from Q_1), that is it is such that $P_1 \stackrel{\mu'}{\longmapsto} P_1'$ implies that $(P_1 \mid Q_1) \stackrel{\mu'}{\longmapsto} (P_1' \mid Q_1\ \tilde{a})$.

- **case (par tau) followed by (app tau):** we have $\mu = \tau$, $P_1 \stackrel{\mu}{\longmapsto} P_1'$, $(P_1 \mid Q_1) \stackrel{\mu}{\longmapsto} (P_1' \mid Q_1)$ and $P \stackrel{\tau}{\longmapsto} (P_1' \mid Q_1)$. Therefore (using rule (app tau) and then (par tau)) we can exhibit a transition $Q \stackrel{\tau}{\longmapsto} Q' = (P_1' \mid Q_1)$;
- **case (par lambda) followed by (app tau):** we have: $\mu = \tau$, $P_1 \stackrel{\lambda\, u}{\longmapsto} P_1'$, $(P_1 \mid Q_1) \stackrel{\lambda\, u}{\longmapsto} (P_1' \mid Q_1\ a)$ and $P \stackrel{\tau}{\longmapsto} (P_1' \mid Q_1\ a)$. Therefore (using rule (app tau) and then (par lambda)) we can exhibit a transition $Q \stackrel{\tau}{\longmapsto} Q' = (P_1' \mid Q_1\ a)$;
- **case (par in) followed by (app in):** in this case $\mu = \mathbf{in}\ v.(\tilde{b}; \tilde{a})$. Let μ' be the action $\mathbf{in}\ v.((a, \tilde{b}); \tilde{a})$, we have: $P_1 \stackrel{\mu'}{\longmapsto} P_1'$, $(P_1 \mid Q_1) \stackrel{\mu'}{\longmapsto} (P_1' \mid Q_1\ a\ \tilde{b})$ and $P \stackrel{\mu}{\longmapsto} (P_1' \mid Q_1\ a\ \tilde{b}))$. Therefore (using rule (app in) and then (par in)) we can exhibit a transition $Q \stackrel{\mu}{\longmapsto} Q' = (P_1' \mid (Q_1\ a)\ \tilde{b})$;
- **case (par out) followed by (app out):** in this case $\mu = \mathbf{out}\, v.(\tilde{v}; (\tilde{a}, a))$. Let μ' be the action $\mathbf{out}\, v.(\tilde{v}; \tilde{a})$, we have: $P_1 \stackrel{\mu'}{\longmapsto} (D; P_1')$, $(P_1 \mid Q_1) \stackrel{\mu'}{\longmapsto} (D; (P_1' \mid Q_1))$ and $P \stackrel{\mu}{\longmapsto} (D; ((P_1' \mid Q_1)\ a))$. Therefore we can exhibit a transition $Q \stackrel{\mu}{\longmapsto} Q' = (D; ((P_1'\ a) \mid (Q_1\ a)))$.

The last case is such that the last two rules are (par com) followed by (app tau). For example we have $P_1 \stackrel{\mathbf{out}\, u.(\tilde{v}; \tilde{a})}{\longmapsto} (D; P_1')$, $Q_1 \stackrel{\mathbf{in}\, v.(\epsilon; \tilde{a})}{\longmapsto} Q_1'$, $(P_1 \mid Q_1) \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu} \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (P_1' \mid Q_1'))$ and $P \stackrel{\tau}{\longmapsto} (\boldsymbol{\nu} \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (P_1' \mid Q_1'))\ a$. Therefore, using rule (app out) and Lemma A.2, we can exhibit the transitions $(P_1\ a) \stackrel{\mathbf{out}\, u.(\tilde{v}; (\tilde{a}, a))}{\longmapsto} (D; P_1'\ a)$ and $(Q_1\ a) \stackrel{\mathbf{in}\, v.(\epsilon; (\tilde{a}, a))}{\longmapsto} Q_1''$ with $Q_1'' \equiv (Q_1'\ a)$. Using rule (par com) it follows that $Q \stackrel{\tau}{\longmapsto} Q' = (\boldsymbol{\nu} \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ ((P_1'\ a) \mid Q_1'')) \equiv P'$.

case $P = \langle u \Leftarrow P_1 \rangle\ a$ **and** $Q = \langle u \Leftarrow P_1 \rangle$**:** the only possibility is for $\langle u \Leftarrow P_1 \rangle$ to "make an **in**-action". In this case, we have $Q = \langle u \Leftarrow P_1 \rangle \stackrel{\mathbf{in}\, v.(\tilde{b};\, \tilde{a})}{\longmapsto} P_1\ \tilde{a}$. Since we can also derive the transition $\langle u \Leftarrow P_1 \rangle \stackrel{\mathbf{in}\, v.((a,\, \tilde{b});\, \tilde{a})}{\longmapsto} P_1\ \tilde{a}$, it follows, using rule (app in), that $P \stackrel{\mathbf{in}\, v.(\tilde{b};\, \tilde{a})}{\longmapsto} P_1\ \tilde{a}$. The proof is similar in the case $P = (\mathtt{0}\ a)$.

□

B Proofs of Section 7

Lemma B.1 (Preservation of τ Reductions) *If $P \stackrel{\tau}{\longmapsto} P'$ and* E *is an evaluation context, then* $\mathrm{E}[P] \stackrel{\tau}{\longmapsto} \mathrm{E}[P']$.

Proof The proof is made by induction on the size of E. Suppose that $P \stackrel{\tau}{\longmapsto} P'$, the case $\mathrm{E} = [\_]$ is straightforward;

- **case** $(\mathrm{E}\ a)$ **and** $(\mathrm{E}l)$**:** the result follows from rule (app tau);
- **case** $(\mathrm{E} \mid P)$ **and** $(P \mid \mathrm{E})$**:** the result follows from rule (par tau);
- **case** $(\boldsymbol{\nu} u)\mathrm{E}$**:** the result follows from rule (new mu);
- **case** $(\mathbf{def}\, D\, \mathbf{in}\, \mathrm{E})$**:** the result follows from rule (def mu).

□

Lemma B.2 (Lambda Actions) *If $P \stackrel{\lambda\, a}{\longmapsto} P'$, then $(P\ a) \to P'$.*

Proof In this proof, we suppose that have an application and not a record selection (the proof is the symmetric case is similar). The proof is made by a case analysis on the last rule of $P \stackrel{\lambda\, a}{\longmapsto} P'$.

- **case (lambda):** we have $P = (\boldsymbol{\lambda} x)Q$ and $P' = Q\{a/x\}$, thus the result follows from rule (red beta);
- **case (par lambda):** we have $P \stackrel{\lambda\, a}{\longmapsto} P'$ implies $(P \mid Q) \stackrel{\lambda\, a}{\longmapsto} (P' \mid Q)$. Therefore, using the induction hypothesis: $(P\ a) \to P'$. The result follows from rules (red equiv) and (red context): $(P \mid Q)\ a \equiv (P\ a) \mid (Q\ a) \to P' \mid (Q\ a)$;
- **case (new mu) and (def mu):** in the first case, we have $P = (\boldsymbol{\nu} v)Q$, $P' = (\boldsymbol{\nu} v)Q'$ and $Q \stackrel{\lambda\, a}{\longmapsto} Q'$. Therefore, using the induction hypothesis: $(Q\ a) \to Q'$. Or the side conditions of rule (new mu) implies that $a \neq v$, thus: $(P\ a) \equiv (\boldsymbol{\nu} v)(Q\ a) \to (\boldsymbol{\nu} v)Q'$. The proof is similar for the case (def mu).

□

Lemma B.3 (In Actions) *If $P \stackrel{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})}{\longmapsto} P'$, then $(P\ \tilde{b}) \mid (u\ \tilde{a}) \to P'$.*

A corollary of this property is that: if $P \xmapsto{\mathbf{in}\, u.(\epsilon;\, \tilde{a})} P'$, then $P \mid (u\ \tilde{a}) \rightarrow P'$.

Proof Let $\tilde{a}$ (resp. $\tilde{b}$) be the tuple $(a_1, \ldots, a_n)$ (resp. $(b_1, \ldots, b_m)$). We make a case analysis on the last rule of $P \xmapsto{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})} P'$.

- **case (decl):** we have $P = \langle u \Leftarrow Q \rangle \xmapsto{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})} (Q\ a_1 \ldots a_n) = P'$ and $(P\ \tilde{b}) \equiv P$. Therefore, using structural equivalence and rule (red decl), we have: $(P\ \tilde{b}) \mid (u\ \tilde{a}) \rightarrow P'$;
- **case (new mu) and (def mu):** in the first case we have $P = (\boldsymbol{\nu}\, u)Q$, $P' = (\boldsymbol{\nu}\, u)Q'$ and $Q \xmapsto{\mu} Q'$. The side condition implies that $u \notin a, \tilde{a}, \tilde{b}$, thus:

  $$((\boldsymbol{\nu}\, u)Q)\ \tilde{b}) \mid (u\ \tilde{a}) \equiv (\boldsymbol{\nu}\, u)(Q\ \tilde{b} \mid (u\ \tilde{a}))$$

  The result follows from the induction hypothesis and rule (red context). The proof is similar in the case (def mu);
- **case (par in):** we have $P \xmapsto{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})} P'$ implies $P \mid Q \xmapsto{\mathbf{in}\, u.(\tilde{b};\, \tilde{a})} P' \mid (Q\ \tilde{b})$ and $(P \mid Q)\ \tilde{b} \equiv (P\ \tilde{b}) \mid (Q\ \tilde{b})$. The result follows from the induction hypothesis and rule (red context);
- **case (app in):** in this case, we use the fact that with our notations: $((P\ c)\ \tilde{b})$ is $P\ (c, \tilde{b})$.

□

Lemma B.4 (Out Actions) *If* $P \xmapsto{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})} (D; P')$, *then there exists* R *such that:*

$$P \equiv (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (u\ \tilde{a} \mid R)) \quad \textit{and} \quad (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ R) \sim_d (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ P')$$

Proof The intuition behind Lemma B.4 is that, under an evaluation context, we can always "lift" definitions up to the top-level. We make a case analysis on the last rule of $P \xmapsto{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})} (D; P')$.

- **case (out):** we have $P = a \xmapsto{\mathbf{out}\, u.(\epsilon;\, \epsilon)} (\emptyset; \mathtt{0})$. Or $a \equiv (\boldsymbol{\nu}\epsilon)(\mathbf{def}\ \mathbf{in}\ (a \mid \mathtt{0}))$ and $\mathtt{0} \sim_d (\boldsymbol{\nu}\epsilon)(\mathbf{def}\ \mathbf{in}\ \mathtt{0})$. The result follows by choosing $R =_{\mathrm{def}} \mathtt{0}$;
- **case (par out), (app out), (new out) and (def out):** in case (par out), for example, we have $P \xmapsto{\mathbf{out}\, u.(\tilde{v};\, \tilde{a})} (D; P')$. Thus there exists R such that $P \equiv (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (u\ \tilde{a} \mid R))$ and the side condition $\mathbf{fn}(Q) \cap (\mathbf{def}(D) \cup \tilde{v}) = \emptyset$ implies:

  $$(P \mid Q) \equiv (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ D\ \mathbf{in}\ (u\ \tilde{a} \mid (R \mid Q)))$$

  The result follows from $\equiv \subset \sim_d$ and $\sim_d$ is a congruence (see Th. 6.1);
- **case (new open) and (def open):** in the case (new open) we use the fact that $P \sim_d Q$ implies $(\boldsymbol{\nu}\, u)P \sim_d (\boldsymbol{\nu}\, u)Q$. This is a consequence of the fact that $\sim_d$ is a congruence. In the case (def open), we also have to use Lemma 6.1 to prove that:

  $$(\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ p = S, D\ \mathbf{in}\ R) \sim_d (\boldsymbol{\nu}\, \tilde{v})(\mathbf{def}\ p = S, D\ \mathbf{in}\ (\mathbf{def}\ p = S\ \mathbf{in}\ R))$$

□

References

[1] Martín Abadi and Luca Cardelli. *A Theory of Objects*. Springer-Verlag, 1996.

[2] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *Proc. of POPL'90–17th Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, January 1990.

[3] Roberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous pi-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.

[4] Hendrik Pieter Barendregt. *The Lambda Calculus, Its syntax and Semantics*. North Holland, 1981.

[5] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[6] Michele Boreale, Cédric Fournet, and Cosimo Laneve. Bisimulations in the join-calculus. In D. Gries and W.P. de Roever, editors, *Proc. of PROCOMET'98–Programming Concepts and Methods*, pages 68–86. Chapman & Hall, June 1998.

[7] Gérard Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA, 1992.

[8] Gérard Boudol. The π-calculus in direct style. In *Proc. of POPL'97–24th Annual ACM Symposium on Principles of Programming Languages*, pages 228–241, January 1997.

[9] Gérard Boudol. Typing the use of resources in a concurrent calculus. In *Proc. of ASIAN'97–Asian Computing Science Conference*, Lecture Notes in Computer Science, Kathmandu, December 1997. Springer-Verlag.

[10] Gérard Boudol. The π-calculus in direct style. *Higher-Order and Symbolic Computation*, 11:177–208, 1998. extended version of [8], also available as `http://www-sop.inria.fr/meije/personnel/Gerard.Boudol.html`.

[11] Silvano Dal-Zilio. Concurrent objects in the blue calculus. english version of [14], also available as `http://www.inria.fr/meije/personnel/Silvano.Dal_Zilio/`.

[12] Silvano Dal-Zilio. Implicit polymorphic type system for the blue calculus. Technical Report 3244, INRIA, September 1997.

[13] Silvano Dal-Zilio. Quiet and bouncing objects: Two migration abstractions in a simple distributed blue calculus. In *1st International Workshop on Semantics of Objects as Processes*. BRICS Notes Series, July 1998.

[14] Silvano Dal-Zilio. Objets concurrents dans un π-calcul applicatif. In *Proc. of JFLA'99–10ème Journées Francophones des Langages Applicatifs*, February 1999.

[15] Nicolaas Govert de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.

[16] Cédric Fournet and Georges Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proc. of POPL'96–23rd Annual ACM Symposium on Principles of Programming Languages*, pages 372–385, January 1996.

[17] Daniel Hirschkoff. *Mise en oeuvre de preuves de bisimulation.* Thèse d'état, École Nationale des Ponts et Chaussées, 1999.

[18] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[19] Massimo Merro and Davide Sangiorgi. On asynchrony in name-passing calculi. In *Proc. of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[20] Robin Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, Laboratory for Foundations of Computer Science, University of Edinburgh, 1991. Reprinted in Logic and Algebra of Specification, F. Bauer, W.Brauer and H. Schwichtenberg, Eds, Springer Verlag, 1993, 204–246.

[21] Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[22] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, parts i and ii. *Journal of Information and Computation*, 100:1–77, 1992.

[23] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer-Verlag, 1992.

[24] Davide Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Information and Computation*, 111(1), 1994.

[25] Davide Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation*, 131(2):141–178, 1996.

[26] Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, October 1998.

[27] James W. Stamos and David K. Gifford. Remote evaluation. *ACM Transactions on Programming Languages and Systems*, 12(4):537–565, October 1990.

[28] David N. Turner. *The Polymorphic Pi-Calculus: Theory and Implementation.* PhD thesis, University of Edinburgh, 1995.

[29] Mitchell Wand. Complete type inference for simple objects. In *Proc. of LICS'87–2nd Symposium on Logic in Computer Science*, pages 37–44, 1987. A corrigendum to the article appeared in LICS'88.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399