Efficient Bisimilarities from Second-order Reaction Semantics for π-calculus*

Pietro Di Gianantonio[1] Svetlana Jakšić[2] Marina Lenisa[1]

[1] Dipartimento di Matematica e Informatica, Università di Udine, Italy.
{digianantonio,lenisa}@dimi.uniud.it
[2] Faculty of Engineering, University of Novi Sad, Serbia. sjaksic@uns.ac.rs

Abstract. We investigate Leifer and Milner *RPO approach* for deriving efficient (finitely branching) LTS's and bisimilarities for π-calculus. To this aim, we work in a category of *second-order term contexts* and we apply a general *pruning technique*, which allows to simplify the set of transitions in the LTS obtained from the original RPO approach. The resulting LTS and bisimilarity provide an alternative presentation of *symbolic LTS* and Sangiorgi's *open bisimilarity*.

Introduction

Recently, much attention has been devoted to deriving *labelled transition systems* and *bisimilarity congruences* from *reactive systems*, in the context of process languages and graph rewriting. Through the notion of *contextual equivalence*, reactive systems naturally induce behavioural equivalences which are *congruences* w.r.t. contexts, while LTS's naturally induce *bisimilarity equivalences* with coinductive characterizations. However, such equivalences are not congruences in general, and it can be a difficult task to derive LTS's inducing bisimilarities which are congruences.

Leifer and Milner [1] presented a general categorical method, based on the notion of Relative Pushout (RPO), for deriving a transition system from a reactive system, in such a way that the induced *bisimilarity* is a *congruence*. The labels in Leifer-Milner's transition system are those *contexts* which are *minimal* for a given reaction to fire. In the literature, some case studies have been carried out in the setting of process calculi, for testing the expressivity of Leifer-Milner's approach [2,3,4,5,6,7,8]. Moreover, to deal with structural rules, an elaboration of the RPO theory in the *G-category setting* (GRPO) has been introduced by Sassone and Sobocinski in [2].

In general, in applying the RPO construction one needs to deal with the following problems:
– To encode all the characteristics of the language, mainly: structural rules, name abstraction, name hiding.

* Work partially supported by PRIN Project SISTER 20088HXMYN and FIRB Project RBIN04M8S8, both funded by MIUR.

– To obtain a label transition system which is usable, where proofs of bisimilarities require to consider only a *finite* set of transitions at each step. Almost always, the RPO approach generates LTS's that are quite large and often redundant, in the sense that most of the transitions can be eliminated from the LTS without affecting the induced bisimilarity.
– When the RPO construction is performed, by embedding the category of terms in a larger category, the resulting LTS can contain states that do not correspond to any term of the language, and whose intuitive meaning is quite difficult to grasp.

In order to solve the above problems, the RPO construction needs to be tuned-up, that is we have to find a convenient category in which to perform the construction, and general methods for pruning the LTS.

In a previous work [7], we solve the above problems for the prototypical example of CCS. In [7], we use a *category of term contexts*, *i.e.* a Lawvere category. We encode names, and name binding using *de Bruijn indexes*; this allows a relatively simple and formally correct treatment of names, which, when represented natively, can be quite subtle to treat. Moreover, in [7] we introduce a general technique, which allows to *prune* an LTS obtained from a RPO-construction, without modifying the induced bisimilarity. This is achieved by eliminating *definable* sets of transitions, *i.e* transitions whose effect can be obtained by other transitions. In [7], by using the above ideas in applying the (G)RPO construction to CCS, we obtain the *standard LTS* from the *standard reaction semantics*. This is an indication that the (G)RPO technique in combination with our general pruning technique can generate useful LTS's.

In the present work, we treat in detail the π-calculus. The techniques developed for CCS turn out to be useful also for the π-calculus, but for the latter, in order to get an efficient LTS, a further ingredient is necessary, *i.e. second-order contexts*. Categories *of second-order term contexts* have been introduced in [9] as generalizations of the Lawvere category of terms, where *parametric rules* can be readily represented. Intuitively, if we apply Leifer-Milner technique to π-calculus by working in a standard Lawvere category of term contexts, in the resulting LTS, for any process P exposing an output prefix, we need to consider transitions $P \xrightarrow{[\ ]|a(x).Q} P'$, for all Q. All these label contexts are "minimal" for the reduction to fire; we cannot avoid Q, since, in the resulting process P', a substitution is applied to Q. This makes the LTS inefficient. To overcome this problem, we use second-order contexts. In this way, all the above transitions can be parametrically captured by a single transition $P \xrightarrow{[\ ]|a(x).X} P''$, where X is a variable representing a generic term, which will be possibly instantiated in the future.

The final result of our construction produces a bisimilarity which is a mild variation of Sangiorgi's *open bisimilarity*. In order to get the final efficient characterization of our bisimilarity, we need a further ad-hoc pruning. However, even if the GRPO construction does not directly give the final result, once applied, it produces an LTS which is a superset of the final usable one. Identifying redun-

dant transitions is then not so difficult; the only difficult part is to prove that these are redundant.

Interestingly enough, our analysis provides new insights on the theory of π-calculus, namely we obtain an alternative presentation of symbolic LTS and open bisimilarity, where *distinctions* do not appear.

Remarkably, the Leifer-Milner technique has lead us to a bisimilarity congruence substantially in a direct way, just using general tools, without the need of new concepts. Whereas, in the standard treatment, in moving from CCS to π-calculus, various new notions are required, such as bound output transitions, distinctions, etc. In conclusion, the results for CCS of [7] and the above results for π-calculus are rather satisfactory, and they are an indication that the general techniques used in this paper could also give new insights on more recent calculi, whose theory is still evolving.

Related Work. The RPO construction has been applied to π-calculus in [3,10]. In [3], History Dependent Automata are used to present a reactive system for the fragment of π-calculus without the ν-operator. The reactive system is obtained by starting from an LTS and then incorporating the labels in the initial state of the transition. The reactive system considered in [10] is based on the theory of bigraphs and models the asynchronous π-calculus.

The present work is also related to [11]. Both works use categories that are suitable generalizations of the Lawvere category of contexts. However, in our work we strictly apply the RPO construction to derive an LTS for the π-calculus, while [11] uses the RPO construction as a sort of inspiration for defining directly an LTS for the π-calculus. The two works use a quite different notion of generalized context, and thus also the obtained LTS's are quite different.

Summary. In Section 1, a presentation of π-calculus syntax with de Bruijn indexes and parametric reaction semantics is given. In Section 2, the GRPO technique is applied to π-calculus, and efficient characterizations of the GIPO bisimilarity are investigated. In Section 3, GIPO bisimilarity is compared with open bisimilarity. Final remarks appear in Section 4. In Appendix A, the theory of RPO's in the G-category setting, and the general pruning technique of [7] are recalled; in Appendix B some proofs are collected.

1 Second-order π-calculus Processes

In this section, we present a version of π-calculus with *de Bruijn indexes* together with reaction semantics. Such presentation allows us to deal smoothly with binding operators, and it is needed for extending to contexts the structural congruence on processes. In our presentation, π-calculus *names* $a_0, a_1, \ldots$ are replaced by de Bruijn indexes $r_0, r_1, \ldots$, which are *name references*.

Intuitively, a name reference can be viewed as a link (or a pointer). So a bound name is replaced by a link to the corresponding binding operator, while a free name is replaced by a link to its occurrence in a list of names. Concretely, links are represented by natural numbers, and:

– binding operators ν and input prefix do not contain any name;

– the index r_i refers to the free name a_j if $j = i - n \geq 0$ and r_i appears under the scope of n binding operators;
– otherwise, if $i < n$, then r_i is bound by the $i + 1$-th binding operator on its left.

E.g. in $\nu r_1().\overline{r_2}r_0.0$, r_0 is bound by the input prefix $r_1()$, while r_1 and r_2 both refer to the free name a_0. In standard syntax, the above process will be written as $(\nu a)a_0(a').\overline{a_0}a'.0$.

Definition 1 (π-calculus Processes). *Let $r_0, r_1, \ldots \in \mathcal{R}$ be* name references*; we will use r, s as metavariables for name references. We define*

$$\begin{array}{rl} (\mathcal{A}ct \ni)\ \alpha ::= & \tau \mid r() \mid \overline{r}s \quad \text{actions} \\ (\mathcal{G} \ni)\ M ::= & 0 \mid M_1 + M_2 \mid \alpha.P \mid Y \quad \text{guarded processes} \\ (\mathcal{P} \ni)\ P ::= & M \mid X \mid \nu P \mid P_1|P_2 \mid rec\ X.P \mid \sigma P \quad \text{processes} \end{array}$$

where

– *$X, X_0, X_1, \ldots \in \mathcal{X}$ are process variables, and $Y, Y_0, Y_1, \ldots \in \mathcal{Y}$ are guarded process variables; we will use Z to range over $\mathcal{X} \cup \mathcal{Y}$;*
– *the process variable X appears guarded in $rec\ X.P$;*
– *σ is a name substitution obtained as a finite composition of the transformations $\{\delta_i\}_{i\geq 0} \cup \{s_i\}_{i\geq 0} \cup \{t_{ij}\}_{i,j\geq 0}$, where δ_i, s_i represent the i-th shifting and the i-th swapping, respectively, and $t_{i,j}$ are the singleton substitutions, defined by:*

$$\delta_i(r_j) = \begin{cases} r_{j+1} & \text{if } j \geq i \\ r_j & \text{if } j < i \end{cases} \qquad s_i(r_j) = \begin{cases} r_j & \text{if } j \neq i, i+1 \\ r_{i+1} & \text{if } j = i \\ r_i & \text{if } j = i+1 \end{cases}$$

$$t_{i,j}(r_k) = \begin{cases} r_k & \text{if } k \neq i \\ r_j & \text{if } k = i \end{cases}$$

A closed process *is a process in which each occurrence of a variable is in the scope of a rec operator.*

In the following definition, we introduce the notion of *second-order context*, consisting of a *variable substitution θ* and a *first-order context*:

Definition 2 (Second-order Contexts). *We define the* second-order 1-hole process contexts (contexts) *by:*

$$\mathbb{C} ::= [\,]_\theta \mid \nu\mathbb{C} \mid P + \mathbb{C} \mid \mathbb{C} + P \mid P|\mathbb{C} \mid \mathbb{C}|P \mid rec\ X.\mathbb{C} \mid \sigma\mathbb{C}$$

where $\theta = \theta^{\mathcal{X}} + \theta^{\mathcal{Y}} : \mathcal{X} + \mathcal{Y} \to \mathcal{P} + \mathcal{G}$ is a substitution of processes for process variables, mapping (guarded) process variables into (guarded) processes.

Notation. *We will often denote substitutions by the list of variables which are actually substituted,* i.e. *as* $\{P_1/X_1, \ldots, P_m/X_m, M_1/Y_1, \ldots, M_n/Y_n\}$*, omitting the variables which are left unchanged. Moreover, for denoting second-order contexts, we will also use the notation* $C[\ ]_\theta$*, when we need to make explicit the variable substitution* θ*.*

Notice that in the above definition of contexts we do not distinguish between guarded and general contexts, thus also "ill-formed" contexts, such as $([\ ]_\theta|P)+P'$ are included at this stage. In Section 2, where we will apply the GIPO technique, we will give a precise definition of guarded and general contexts.

In what follows, we will refer to π-calculus processes with de Bruijn indexes and second-order contexts as *terms*, denoted by T. Intuitively, when a second-order context $C[\ ]_\theta$ is applied to a term T, the variable substitution θ is applied to T and the resulting term is put in the hole. In order to formalize this notion of context application, we first need to introduce the notion of applying a substitution to a term:

Definition 3 (Context Application).
(i) Let T *be a term, and let* θ *be a variable substitution. We define the extension* $\widehat{\theta}$ *to terms, by induction on* T *as:*

$$\widehat{\theta}(Z) = \theta(Z) \qquad \widehat{\theta}([\ ]_{\theta'}) = [\ ]_{\widehat{\theta}\circ\theta'}$$

$$\widehat{\theta}(T_1 + T_2) = \widehat{\theta}(T_1) + \widehat{\theta}(T_2) \qquad \widehat{\theta}(T_1 \mid T_2) = \widehat{\theta}(T_1) \mid \widehat{\theta}(T_2)$$

$$\widehat{\theta}(\sigma T) = \sigma\widehat{\theta}(T) \qquad \widehat{\theta}(\nu T) = \nu(\widehat{\theta}(T))$$

$$\widehat{\theta}(rec\ X.T) = rec\ X.\widehat{\theta'}(T)\ , \qquad where\ \theta'(Z) = \begin{cases} \theta(Z) & if\ Z \neq X \\ X & if\ Z = X \end{cases}$$

In what follows, by abuse of notation, we will often denote $\widehat{\theta}(T)$ *simply by* $\theta(T)$*.*
(ii) Let $\mathbb{C}$ *be a context and let* T *be a term, the application of* $\mathbb{C}$ *to* T*, denoted by* $\mathbb{C}\cdot T$*, is defined by induction on* $\mathbb{C}$ *by:*

$$[\ ]_\theta \cdot T = \widehat{\theta}(T) \qquad \nu\mathbb{C}\cdot T = \nu(\mathbb{C}\cdot T)$$

$$(P + \mathbb{C})\cdot T = P + (\mathbb{C}\cdot T) \qquad (\mathbb{C} + P)\cdot T = (\mathbb{C}\cdot T) + P$$

$$(P \mid \mathbb{C})\cdot T = P \mid (\mathbb{C}\cdot T) \qquad (\mathbb{C} \mid P)\cdot T = (\mathbb{C}\cdot T) \mid P$$

$$(rec\ X.\mathbb{C}\cdot T) = rec\ X.(\mathbb{C}\cdot T) \qquad (\sigma\mathbb{C})\cdot T = \sigma(\mathbb{C}\cdot T)$$

In order to apply the GRPO technique to π-calculus, it is convenient to extend the structural congruence, which is usually defined only on processes, to all contexts. Here is where the syntax presentation à la de Bruijn plays an important rôle. Namely the π-calculus rule

$$(\nu aP) \mid Q \equiv \nu a(P \mid Q)\ ,\ \text{if } a \text{ not free in } Q$$

is problematic to extend to contexts with the usual syntax, since, if Q is a context, we have to avoid captures by the ν-operator of the free variables of the processes that will appear in the holes of Q. Using de Bruijn indexes (and index transformations), the above rule can be naturally extended to contexts as:

$$(\nu P) \mid \mathbb{C} \equiv \nu(P \mid \delta_0\mathbb{C})$$

where the shifting operator δ_0 avoids the capture of free name references. In the standard syntax there is no way of defining a general name substitution playing the role of δ_0.

The complete definition of the structural congruence is as follows:

Definition 4 (Structural Congruence). *Let T be a term.* Structural congruence *is the equivalence relation $\equiv$, closed under process constructors, inductively generated by the usual axioms on $|$, $+$, and by:*

$$\begin{array}{ll}
(\mathbf{nu}) & \nu 0 \equiv 0 \quad T|(\nu T') \equiv \nu((\delta_0 T)|T') \quad \nu\nu T \equiv \nu\nu \mathbf{s}_0 T \\
& \tau.\nu P \equiv \nu\tau.P \quad \overline{r}s.\nu P \equiv \nu\delta_0(\overline{r}s).P \quad r().\nu P \equiv \nu\delta_0(r()).s_0 P \\
(\mathbf{sigma}) & \sigma 0 \equiv 0 \qquad \sigma(\overline{r}s.T) \equiv \overline{\sigma(r)}\sigma(s).\sigma(T) \\
& \sigma(\tau.T) \equiv \tau.\sigma(T) \qquad \sigma(r().T) \equiv \sigma(r)().\sigma_{+1}T \\
& \sigma(T|T') \equiv \sigma(T)|\sigma(T') \qquad \sigma(rec\ X.T) \equiv rec\ X.(\sigma T) \\
& \sigma(T+T') \equiv \sigma(T)+\sigma(T') \quad \sigma(\nu T) \equiv \nu(\sigma_{+1}T) \\
& \sigma_1 \ldots \sigma_m T \equiv \sigma'_1 \ldots \sigma'_n T\ , \quad \textit{if}\ \ \sigma_1 \circ \ldots \circ \sigma_m = \sigma'_1 \circ \ldots \circ \sigma'_n \\
(\mathbf{subs}) & [\ ]_\theta \equiv [\ ]_{\theta_1}\ \textit{if}\ \forall X\ \theta(X) \equiv \theta_1(X) \qquad (\mathbf{rec})\ \ rec\ X.P \equiv P[rec\ X.P/X]
\end{array}$$

$$\textit{where } \sigma_{+1}(r_i) = \begin{cases} r_0 & \textit{if } i = 0 \\ (\sigma(r_{i-1}))_{+1} & \textit{otherwise} \end{cases} \qquad \sigma(\alpha) = \begin{cases} \overline{\sigma}(r) & \textit{if } \alpha = \overline{r} \\ \sigma(r) & \textit{if } \alpha = r \\ \tau & \textit{if } \alpha = \tau \end{cases}$$

The last three (**nu**)-rules are not standard in π-calculus presentations, since they are not strictly necessary for proving the basic syntactic properties of the calculus. However, they are safe because they allow to move, inside/outside the ν operator, prefixes which are not captured by ν, see *e.g.* [12]. The assumption of such extra rules is not essential in our presentation, however it simplifies the GIPO construction. As far as the (**sigma**)-rule, notice that there is an effective procedure to determine whether $\sigma_1 \circ \ldots \circ \sigma_m = \sigma'_1 \circ \ldots \circ \sigma'_n$. Namely, the two compositions are equal if and only if they contain the same number of transformations in the forms δ_i and their behaviour coincides on an initial segment of indexes (whose length can be calculated from the δ_i's and the s_i's involved). Finally, the unfolding rule (**rec**) is given only for processes P. It cannot be directly extended to contexts, since their unfolding can produce multiple-hole contexts. However, the above (**rec**)-rule is sufficient for our purposes, since we will only need it in reducing processes.

As in the standard presentation, one can easily show that each π-calculus process P is structurally congruent to a process in *normal form*, *i.e.* a process of the shape $\nu^k(\Sigma_{j=1}^{m_1} S_{1,j} \mid \ldots \mid \Sigma_{j=1}^{m_n} S_{n,j})$, where all unguarded restrictions are at the top level, and name substitutions do not appear at the top level. We use S to range over processes of the shape $\alpha.P$ or σY. If $m_i = 1$ for some $i \in \{1, \ldots n\}$ then S can also be of the form σX.

Definition 5 (Reaction Semantics). *The* reaction relation $\rightarrow$ *is the least relation closed under the following* reaction rules *and* reaction contexts*:*

Reaction rules. $(r().X_1 + Y_1) \mid (\overline{r}r_j.X_2 + Y_2) \to (\nu(t_{0,j+1}X_1)) \mid X_2$
$\tau.X + Y \to X$

Reaction contexts. $\mathbb{D} ::= [\,]_\theta \mid \nu\mathbb{D} \mid P|\mathbb{D} \mid \mathbb{D}|P \mid \sigma\mathbb{D}$

where σ is a permutation of name references (a one to one reference substitution).

Notice that the permutation σ in the definition of reaction contexts is not strictly necessary for defining the reaction semantics. It could be omitted, without changing the reaction semantics, since, using the congruence rules, name substitutions distribute over the actions. However, in view of the GIPO construction of Section 2 it is useful to include it.

A mapping $\mathcal{T}$ from standard π-calculus syntax into our de Bruijn presentation can be defined by structural induction, using an extra set of names with negative indexes $(a_{-1}, a_{-2}, \ldots)$. The most meaningful cases are: $\mathcal{T}(P) = \mathcal{T}_0(P)$, $\mathcal{T}_n(a_i(a_j).P) = r_{i+n}().\mathcal{T}_{n+1}(P_{\{a_{-n-1}/a_j\}})$, $\mathcal{T}_n(\overline{a_i}a_j.P) = \overline{r_{i+n}}r_{j+n}.\mathcal{T}_n(P)$.

For any pair of π-calculus processes P, Q on the standard syntax, it turns out that $P \to Q$ in the ordinary reaction system iff $\mathcal{T}(P) \to \mathcal{T}(Q)$ in our reaction system. We omit the details.

2 Applying the GIPO Technique to Second-order π-calculus

For lack of space, we do not present in detail the (G)RPO construction, we refer to [1] for a general introduction to the RPO technique, to [2] for the presentation of the GRPO technique and to [7], or to Appendix A, for a compact presentation of all the theory on which the results presented here are based.

However, in order to grasp most of the material presented in the paper, the following informal and intuitive explanations of the GRPO construction may suffice. The main idea in the RPO construction is to define an LTS, starting from a reaction system. The states of the derived LTS are terms, while the labels are the *minimal contexts* necessary for a given reaction to fire. In more detail, the LTS contains the transition $t \xrightarrow{\mathbb{C}}_I v$, if the reaction system contains the reaction $\mathbb{C} \circ t \to v$, and for no subcontext $\mathbb{C}'$ of $\mathbb{C}$ and no subterm v' of v, there is a reaction $\mathbb{C}' \circ t \to v'$. This idea is formalized using a category where arrows represent terms or contexts. The notion of minimal context is defined in terms of a (relative) pushout construction. The main theoretical result is that the LTS, obtained by the RPO construction, induces a *bisimilarity* that is a *congruence*. The GRPO technique is a further elaboration of the RPO technique necessary to deal with the structural rules of the syntax; here the main idea is to perform the RPO construction in a *2-category*. A 2-category is a category having an extra notion of morphism between arrows. When such morphisms are isomorphisms, as in the GRPO construction, the 2-category is called *G-category*. In our setting, morphisms between two arrows represent a structural congruence between two terms (the two arrows), together with an induced mapping between

occurrences of name references in the two terms. G-categories always allow to distinguish between two different name references denoting the same name, also when structural rules are used. In some cases, the RPO construction in the standard categories having as arrows equivalence classes of terms fails to produce the correct transitions, an example being $r_0().0 \mid \overline{r_0} r_1.0$, see [2] for more details.

We define here the G-category formed by the *finite* (*i.e.* without the *rec* operator) second-order π-calculus terms equipped with structural congruence. We restrict the G-category to contain only finite processes, because we need the 2-cell morphisms to be isomorphisms. When π-calculus processes contain the *rec* operator, two congruent processes can contain different numbers of actions, so, in general, there does not exist a one-to-one map between occurrences of name references.

It is possible to recover an LTS for the whole set of π-processes by extending the rules obtained for the finite calculus, namely allowing parametric rules to be applied also to terms containing the *rec* operator (and by considering the unfolding rule for *rec*). Quite general arguments show that, in the extended LTS, the bisimilarity is still a congruence. Briefly, the arguments are the following. To any infinite process P one associates the set of its finite approximations $\{P_i^o \mid i > 0\}$ obtained by replacing, in the unfolding of P, the subterms containing *rec* with 0. For the finite approximations, the following properties hold:

- each P_i^o is simulated by P,
- if every finite approximation of P is simulated by Q, then also P is simulated by Q,
- if a finite process P^o is simulated by Q, then there exists a finite approximation of Q simulating P^o.

Since the behaviour of a term is described by the behaviour of its finite approximations, from the fact that simulation is a precongruence on finite approximations it follows that simulation is also a precongruence on infinite processes.

Moreover, once restricted to finite processes, in the definition of π-calculus term category, it is sufficient to consider *linear* terms, that is terms where each variable appears at most once. This restriction is justified by the fact that, in the GIPO transition system, closed terms generate only linear open terms; moreover, it simplifies the GIPO construction below.

Since the π-calculus grammar needs to distinguish between guarded and generic terms, the category needs to contain two corresponding distinct objects. Formally:

Definition 6 (Category of Second-order π-calculus Terms). *Let $\mathcal{C}_\pi$ be the category defined by:*

- *Objects are ϵ, $\mathcal{G}$, $\mathcal{P}$.*
- *Arrows from ϵ to $\mathcal{G}$ ($\mathcal{P}$) are linear (un)guarded processes,* i.e. *processes where each variable appears at most once. Arrows $\mathcal{A} \to \mathcal{B}$ are the contexts $\mathbb{C}_{\mathcal{A}}^{\mathcal{B}}$ generated by the grammar:*
  $\mathbb{C}_{\mathcal{G}}^{\mathcal{G}} ::= [\,]_\theta \mid \alpha.\mathbb{C}_{\mathcal{G}}^{\mathcal{P}} \mid \mathbb{C}_{\mathcal{G}}^{\mathcal{G}} + M \mid M + \mathbb{C}_{\mathcal{G}}^{\mathcal{G}}$

$\mathbb{C}^{\mathcal{G}}_{\mathcal{P}} ::= \alpha.\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \mathbb{C}^{\mathcal{G}}_{\mathcal{P}} + M \mid M + \mathbb{C}^{\mathcal{G}}_{\mathcal{P}}$
$\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} ::= \mathbb{C}^{\mathcal{G}}_{\mathcal{G}} \mid \nu\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} \mid \mathbb{C}^{\mathcal{P}}_{\mathcal{G}}|P \mid P|\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} \mid \sigma\mathbb{C}^{\mathcal{P}}_{\mathcal{G}}$
$\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} ::= [\ ]_\theta \mid \mathbb{C}^{\mathcal{G}}_{\mathcal{P}} \mid \nu\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \mathbb{C}^{\mathcal{P}}_{\mathcal{P}}|P \mid P|\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \sigma\mathbb{C}^{\mathcal{P}}_{\mathcal{P}}$
where any context $\mathbb{C}^{\mathcal{B}}_{\mathcal{A}} = C[\ ]_\theta$ *is linear,* i.e. *any variable appears at most once in* $C[\ ]$ *and in the codomain of* θ.

The identity arrow on $\mathcal{G}$ *and* $\mathcal{P}$ *is* $[\ ]_{id}$. *The only arrow with codomain* ϵ *is the identity. The composition between morphisms* $T : \mathcal{A} \to \mathcal{A}'$, $T' : \mathcal{A}' \to \mathcal{A}''$ *is the context application* $T' \cdot T$.

In what follows, when not necessary, we will omit tags from contexts.

One can easily prove that the above definition is well-posed. In particular, associativity of composition follows from associativity of composition of variable substitutions.

By induction on a proof of structural congruence, it is possible to show that two structurally congruent finite terms have the same number of occurrences for each action, and each proof of congruence induces a one to one map between instances of name references in an obvious way. Thus we can define:

Definition 7 (2-cell isomorphisms). *2-cell isomorphisms between* T *and* T' *in* $\mathcal{C}_\pi$ *are the one-to-one maps between occurrences of name references in* T *and* T', *induced by the proof of structural congruence.*

The above maps induce a structure of G-category on $\mathcal{C}_\pi$. Horizontal composition corresponds to the union of the one-to-one maps , while vertical composition amounts to standard function composition. One can easily check that horizontal and vertical compositions are well-behaved, in particular the "middle-four interchange law" holds. Thus we have:

Proposition 1. *The structural congruence on terms induces a structure of G-category on* $\mathcal{C}_\pi$.

Now we can define the G-reaction system of finite (second order) π-calculus processes:

Definition 8 (G-reaction system). *The G-reaction system* $\mathbf{C}_\pi$ *consists of*

- *the G-category of* π*-calculus terms* $\mathcal{C}_\pi$*;*
- *the distinguished object* ϵ*;*
- *the subset of linear reaction contexts of Definition 5;*
- *the reaction rules of Definition 5.*

One can easily check that the set of reaction contexts as defined above are composition-reflecting and closed under 2-cells. In particular, in proving that contexts are composition-reflecting, it turns out to be essential to have included also reaction contexts of the shape $\sigma\mathbb{D}$, for σ a permutation.

Proposition 2. *The G-reaction system* $\mathbf{C}_\pi$ *has redex GRPOs.*

A proof of the above proposition appears in Appendix B.

Table 1 summarizes the GIPO contexts (i.e. the labels in the derived LTS) for every possible term (up-to structural congruence). For simplicity, we denote a term equivalence class simply by a special representative. For each process P, on the basis of its form (specified in the first column of the table), the corresponding GIPO contexts are listed, i.e. the "minimal" contexts which make possible a certain reaction. Redex squares can be classified according to the following "parameters":

- type of the reaction rule (τ-reaction or communication);
- how elements of the redex are obtained:
  - already present in P;
  - by instantiating variables in P;
  - appearing in the context;
- in case of variable instantiation by an output action, the name sent can be either private or public.

A more detailed description of the GIPO contexts of Table 1 follows.

Rows 1–3 correspond to a τ-reaction, while rows 4–13 correspond to a communication reaction. In particular, row 1 takes into account the case where an internal transition in the process P is present. In such case the GIPO context is the identity, up-to a certain bijective name substitution β and a variable substitution δ. The substitution δ sends all variables into variables with even index (see the note at the bottom of Table 1), and it is used to preserve linearity in the term $\mathbb{C}\cdot P$. Namely, δ ensures that the variables with odd indexes will not appear in the process, and hence they can be used in the context (see *e.g.* row 2). Row 2 corresponds to the case where a variable Z appears at the top of a process P, and the GIPO context instantiates the variable with a τ-transition. Row 3 shows all GIPO contexts where the τ-reaction is "all inside the context" (and the process plays a passive rôle). Row 4 corresponds to the case where the process P exposes two complementary actions. Then the minimal context in which the communication arises is $\beta\iota[\ ]_\delta$, where β is a name substitution and ι is the identity, if the channel references r and r' in the complementary actions already matches, or a singleton substitution fusing the two channel references, otherwise, see the note at the bottom of Table 1. Here we use a function $[\![\ ,\ ]\!]$ to express the fact that the two *occurrences* of name references in the complementary actions refer to the same name. This function, given a process and an *occurrence* of a name reference r_i in it, provides the "absolute" index of the name referred by the the occurrence r_i, if r_i is free in P, that is $[\![P, r_i]\!] = j$ means that r_i refers to the free name a_j; otherwise, if r_i is bound, $[\![P, r_i]\!]$ provides the negative index corresponding to the nesting level of the occurrence r_i inside the binding operators (ν or input prefix) in P (we omit the formal definition). Rows 5 and 6 take into account the case where the process P exposes either an input or an output action and the GIPO context provides the complementary action for the communication. In rows 7–12 we consider all cases where one or two variables appear in P. Then a communication reaction arises when the GIPO context instantiates the two variables by a complementary actions (rows 7,7'), or it instantiates a variable by

an input (output) action and provides the complementary action (rows 8,9,9'), or the GIPO context instantiates the variable in P by the whole communication redex (rows 10,10'), or finally the GIPO context instantiates the variable with an action and the complementary action already appears in the process (rows 11,11',12). Notice that when a variable is instantiated with an output action, we need to consider two possible instantiations, that is the one where the name sent is not locally bounded (7,9,10,11) and the one where the name sent is private, *i.e.* locally bounded by ν (7',9',10',11'). The last row 13 in the table takes into account the cases where a communication redex is all in the context.

The GIPO LTS described in Table 1 is quite redundant. Namely, there are many GIPO contexts which are intuitively redundant; *e.g.* all contexts in rows 3 and 13, which are "not engaged". Moreover, in various other cases the effect of some GIPO contexts can be simulated by a restricted set of simpler contexts. Many redundant contexts can be eliminated by applying the general pruning technique presented in [7] and recalled in Appendix A. The result is the LTS of *reduced GIPO contexts*, R, formed by the contexts marked by $*$ in the column R of Table 1, in which the name substitution β is restricted to be the identity. Namely, the GIPO LTS of Table 1 is *definable* from the set R of reduced GIPO contexts. A proof of this can be found in Appendix B. As a consequence, our general pruning technique ensures that the bisimilarity $\sim_R$ induced by the LTS defined in column R coincides with the original GIPO bisimilarity $\sim_G$, and hence it is a congruence.

A further simplified LTS can be obtained by an ad-hoc analysis. In fact, one can prove that the GIPO context in row 2 of Table 1 can be eliminated, since intuitively it just allows us to observe that a variable appears in the term, but there are other contexts that allow us to observe this. Also the GIPO contexts in rows 7',9',11' are redundant; intuitively, the behavior of a process which receives a new bound name is subsumed by the behavior of the same process receiving a new free name. Moreover, the σ's of the GIPO contexts in rows 5,6,8,9 can also be avoided.

Formally, we define an LTS, F, composed by the GIPO contexts marked by $\star$ in column F of Table 1. The proof that the bisimilarity induced by the LTS F coincides with the original GIPO bisimilarity is based on the technique of the "bisimulation up-to", and follows from Lemma 2 given in Appendix B.

Proposition 3. *The bisimilarity $\sim_F$ induced by the LTS F coincides with the original GIPO bisimilarity $\sim_G$, and hence it is a congruence.*

Apparently, the LTS F obtained is still infinitely branching. This is due to the fact that we consider transitions where the context contains an output action $\overline{r}s.X$, and s can be any reference. But, when comparing two processes P, Q in the bisimilarity relation, it is sufficient to consider s to be a reference to a name in P or Q, or a reference to just a new name not appearing in P or Q. In this way, we get a finitely branching LTS.

Now, if our aim is to define a bisimilarity relation on π-calculus processes which do not contain process variables, then it is possible to consider a much

Table 1. π-calculus GIPO contexts.

	Process $P \equiv \nu^k(\Sigma_{j=1}^{m_1} S_{1,j} \mid \ldots \mid \Sigma_{j=1}^{m_n} S_{n,j})$	**GIPO context** $\mathbb{C}$	**R**	**F**
1	$\exists i,j.\ S_{i,j} = \tau.P_{i,j}$	$\beta[\ ]_\delta$	$*$	$\star$
2	$\exists i,j.\ S_{i,j} = \sigma Z$	$\beta[\ ]_{\{(\tau.X_1+Y_1)/\delta Z\}\circ\delta}$	$*$	
3		$C'[\ ]_\theta\ +\ \tau.X_1$ $C'[\ ]_\theta \mid (\tau.X_1 + Y_1)$ $\tau.C'[\ ]_\theta + Y_1$		
4	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = r().P_{i,j}\ \wedge\ S_{i',j'} = \overline{r}'s.P_{i',j'}$	$\beta\iota[\ ]_\delta$	$*$	$\star$
5	$\exists i,j.\ S_{i,j} = r().P_{i,j}$	$(\overline{r}'s.X_1 + Y_1) \mid (\sigma[\ ]_\delta\ +\ Y_3)$	$*$	$\star$
6	$\exists i,j.\ S_{i,j} = \overline{r}s.P_{i,j}$	$(r'().X_1 + Y_1) \mid (\sigma[\ ]_\delta\ +\ Y_3)$	$*$	$\star$
7	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = \sigma_1 Z\ \wedge\ S_{i',j'} = \sigma_2 Z'$	$\beta\iota[\ ]_{\{(r().X_1+Y_1)/\delta Z,(\overline{r}'s.X_3+Y_3)/\delta Z'\}\circ\delta}$	$*$	$\star$
7'	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = \sigma_1 Z\ \wedge\ S_{i',j'} = \sigma_2 Z'$	$\beta\iota[\ ]_{\{(r().X_1+Y_1)/\delta Z,\nu(\overline{r}'r_0.X_3+Y_3)/\delta Z'\}\circ\delta}$	$*$	
8	$\exists i,j.\ S_{i,j} = \sigma' Z$	$(\overline{r}'s.X_1 + Y_1) \mid \sigma[\ ]_{\{(r().X_3+Y_3)/\delta Z\}\circ\delta}$	$*$	$\star$
9	$\exists i,j.\ S_{i,j} = \sigma' Z$	$(r'().X_1 + Y_1) \mid \sigma[\ ]_{\{(\overline{r}s.X_3+Y_3)/\delta Z\}\circ\delta}$	$*$	$\star$
9'	$\exists i,j.\ S_{i,j} = \sigma' Z$	$(r'().X_1 + Y_1) \mid \sigma[\ ]_{\{\nu(\overline{r}r_0.X_3+Y_3)/\delta Z\}\circ\delta}$	$*$	
10	$\exists i\ m_i = 1\ \wedge\ S_{i,1} = \sigma X$	$\beta[\ ]_{\{((r().X_1+Y_1)\ \mid\ (\overline{r}'s.X_3+Y_3))/\delta X\}\circ\delta}$	$*$ $r \neq r'$	$\star$ $r \neq r'$
10'	$\exists i\ m_i = 1\ \wedge\ S_{i,1} = \sigma X$	$\beta[\ ]_{\{((r().X_1+Y_1)\ \mid\ \nu(\overline{r}'r_0.X_3+Y_3))/\delta X\}\circ\delta}$		
11	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = \sigma Z\ \wedge\ S_{i',j'} = r().P_{i',j'}$	$\beta\iota[\ ]_{\{(\overline{r}'s.X_1+Y_1)/\delta Z\}\circ\delta}$	$*$	$\star$
11'	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = \sigma Z\ \wedge\ S_{i',j'} = r().P_{i',j'}$	$\beta\iota[\ ]_{\{\nu(\overline{r}'r_0.X_1+Y_1)/\delta Z\}\circ\delta}$	$*$	
12	$\exists i,j,i',j'.\ i \neq i' \wedge$ $S_{i,j} = \sigma Z\ \wedge\ S_{i',j'} = \overline{r}s.P_{i',j'}$	$\beta\iota[\ ]_{\{(r'().X_1+Y_1)/\delta Z\}\circ\delta}$	$*$	$\star$
13		$C'[\ ]_\theta \mid (r().X_1 + Y_1) \mid (\overline{r}s.X_3 + Y_3)$ $(\overline{r}s.X_1 + Y_1) \mid (C'[\ ]_\theta\ +\ r().X_3)$ $(\overline{r}s.X_1 + Y_1) \mid (r().C'[\ ]_\theta + Y_3)$		

where:

- the substitution $\delta = [X_{2h}/X_h,\ Y_{2h}/Y_h]_{h\geq 0}$ sends all variables into variables with even index;
- $C'[\ ]_\theta$ in rows 3 and 13 is any second-order context s.t. the variables in the GIPO context are not in the codomain of θ;
- r, r' are such that $[\![\mathbb{C}\cdot P, r]\!] = [\![\mathbb{C}\cdot P, r']\!]$;
- if $\mathbb{C}$ is of the form $\beta\iota\mathbb{C}'$, then ι is the identity if $[\![\mathbb{C}'\cdot P, r]\!] = [\![\mathbb{C}'\cdot P, r']\!]$, and a singleton substitution otherwise.

$*$ where β, if it appears, is the identity.

$\star$ where β and σ, if they appear, are the identity.

Table 2. π-calculus final GIPO contexts for closed processes.

	Process $P \equiv \nu^k(\Sigma_{j=1}^{m_1} S_{1,j} \mid \ldots \mid \Sigma_{j=1}^{m_n} S_{n,j} \mid \sigma X)$	**GIPO Context** $\mathbb{C}$
1	$\exists i,j.\ S_{i,j} = \tau.P_{i,j}$	$[\,]_{id}$
2	$\exists i,j,i',j'.\ i \neq i' \ \wedge\ S_{i,j} = r().P_{i,j} \ \wedge\ S_{i',j'} = \overline{r}'s.P_{i',j'}$	$\iota[\,]_{id}$
3	$\exists i,j.\ S_{i,j} = r().P_{i,j}$	$[\,]_{\{\overline{r}'s.X_1+Y_1/\delta X\}\circ\delta}$
4	$\exists i,j.\ S_{i,j} = \overline{r}s.P_{i,j}$	$[\,]_{\{r'().X_1+Y_1/\delta X\}\circ\delta}$
5		$[\,]_{\{(r().X_1+Y_1 \mid \overline{r'}s.X_3+Y_3)/\delta X\}\circ\delta}$ $r \neq r'$

where:
- r, r' are such that $[\![\mathbb{C}\cdot P, r]\!] = [\![\mathbb{C}\cdot P, r']\!]$;
- if $\mathbb{C}$ is of the form $\iota[\,]_{id}$, then ι is the identity if $[\![P, r]\!] = [\![P, r']\!]$, and a singleton substitution otherwise.

simpler LTS, namely the LTS presented in Table 2. This LTS is intended for processes in the form $\nu^k(P \mid \sigma X)$, with P a closed process. The above set of processes is closed by all transitions, but 5, which is then meant to be applied just once. In order to compare two closed processes P, Q, we proceed by comparing the processes $P|X$ and $Q|X$, using the LTS of Table 2. Namely, if $\sim_C$ denotes the induced bisimilarity, we have:

Proposition 4. *For any pair of closed processes P, Q, we have that $P \sim_F Q$ iff $P \mid X \sim_C Q \mid X$.*

3 GIPO Bisimilarity on Standard Syntax vs Open Bisimilarity

In this section, first we provide a presentation of GIPO LTS and bisimilarity for closed processes in the standard π-calculus syntax. Then, we compare this bisimilarity with Sangiorgi's open bisimilarity, [13]. GIPO bisimilarity turns out to be finer than open bisimilarity; however a small variant of it gives exactly the open bisimilarity. Thus, interestingly enough, we obtain an efficient characterization of open bisimilarity, alternative to Sangiorgi's characterization on the symbolic LTS, [13]. An advantage of our presentation lies in the fact that our bisimilarity has a direct definition of the LTS, without requiring the extra machinery of *distinctions*.

3.1 A Presentation of GIPO Bisimilarity on Standard Syntax

In order to compare our GIPO LTS and bisimilarity with standard LTS's and bisimilarities of π-calculus, it is useful to provide a presentation of GIPO LTS and bisimilarity for closed processes in the standard π-calculus syntax.

The intuitive idea is the following. The LTS in Table 2 uses terms having form $\nu^k(P \mid \sigma X)$. In the standard syntax, there is an immediate correspondent for the part $\nu^k(P)$, that is the corresponding nameful π-calculus term. Less obvious is how to define a correspondent for the σX part. The permutation σ essentially depends on output actions that have been performed in the previous transitions (history), and there are three important aspects: (i) the permutation σ is determined by the list of names that have been communicated by the process P to X (the observer); (ii) σ determines which private names in $\nu^k(P)$ can be used for future communications; (iii) through transitions of kind 5 in Table 2, we can check which public name has been communicated to X, and whether the same private name has been used in two different communications.

The following example illustrates the above remarks. Consider the nameful π-calculus process $(\nu a_2)\overline{a_0}a_2.\overline{a_2}a_1.0$, its correspondent in de Bruijn notation is $\nu\overline{r_1}r_0.\overline{r_0}r_2.0$; put in parallel with a process variable X, the process becomes $\nu(\overline{r_1}r_0.\overline{r_0}r_2.0 \mid \delta_0 X)$. According to Table 2, the only possible transition for this process is through the GIPO context $[\,]_{\{r_0().X_1+Y_1/\delta X\}\circ\delta}$. The application of the context to the term gives $\nu(\overline{r_1}r_0.\overline{r_0}r_2.0 \mid \delta_0 r_0().X_1 + Y_1) \equiv \nu(\overline{r_1}r_0.\overline{r_0}r_2.0 \mid (r_1().\delta_1 X_1 + \delta_1 Y_1)$, so the result of the GIPO transition is $\nu(\overline{r_0}r_2.0 \mid \nu(t_{0,1} \circ \delta_1)X_1) \equiv \nu(\overline{r_0}r_2.0 \mid \nu\delta_0 X_1) \equiv \nu^2(\overline{r_1}r_3.0 \mid \delta_0 X_1)$. In this later process, the bound reference r_1 is visible to the process variable X_1, so it is possible to proceed with the GIPO transition $[\,]_{\{r_0().X_1+Y_1/\delta X_1\}\circ\delta}$, leading to the process $\nu^3((t_{0,4}\circ\delta_1)X_1)$. The name substitution $(t_{0,4}\circ\delta_1)$ transforms both name references r_0 and r_3 in the name reference r_4, so there is the GIPO transition $[\,]_{\{r_0().X_1+Y_1 \mid \overline{r_3}sX_3+Y_3/\delta X_1\}\circ\delta}$, that allows to observe that in the previous transitions the name reference r_3, corresponding to the name a_1, has been passed to X_1.

Given the above observations, we represent the information captured by σX via the list L of private names communicated to X by the process. We omit public names, since they can be represented directly on the labels of the LTS, and their presence in the list is not strictly necessary. Thus in the LTS we consider pairs $\langle \nu \boldsymbol{a} Q, L\rangle$ such that the elements of L are names in $\boldsymbol{a}$. Possible applications of the α-rule to the process apply also to the list of names L.

Labels α in the LTS range over $\alpha ::= \tau \mid \{a'/a\} \mid xy \mid \overline{x}y$, where we assume the set of names ordered, and we denote by $\{a'/a\}$ a singleton substitution, with $a < a'$ in such ordering.

Transitions $\langle P, L\rangle \xrightarrow{\alpha} \langle P', L'\rangle$ are described in Table 3.

Remark. Traditional LTS's use as labels part of the term, dually (G)RPO LTS's use as labels contexts that can interact with the term, and in particular with the part of the term that is "put in evidence" by the traditional LTS; in the presentation above we use a traditional approach.

In order to define the bisimilarity induced by the above LTS, we first need to define a relation on possibly bound names w.r.t. lists of names:

Definition 9. *Let L, M be name lists. We define*
$x =_{LM} y$ *iff* $x = a = y$ *or* $x = \nu a \ \wedge\ y = \nu a' \ \wedge\ \forall i.\ (a = L(i) \iff a' = M(i))$.

Table 3. Transitions in the standard LTS.

	Process $P \equiv \nu \boldsymbol{a}(\Sigma_{j=1}^{m_1} S_{1,j} \mid \ldots \mid \Sigma_{j=1}^{m_n} S_{n,j})$ **List** L	**Label** α	**Process** P' **List** L'
1	$\exists i,j.\ S_{i,j} = \tau.P_{i,j}$	τ	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij} \mid \ldots)$ $L' \equiv L$
2	$\exists i,j,i',j'.\ (i \neq i' \ \wedge\ S_{i,j} = a(b).P_{i,j} \ \wedge$ $S_{i',j'} = \overline{a}c.P_{i',j'})$	τ	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots$ $\ldots \mid P_{i'j'} \mid \ldots)$ $L' \equiv L$
3	$\exists i,j,i',j'.\ (i \neq i' \ \wedge\ S_{i,j} = a(b).P_{i,j} \ \wedge$ $S_{i',j'} = \overline{a}'c.P_{i',j'})$ $a, a' \in \mathit{free}(P),\ a < a'$	$\{a'/a\}$	$P' \equiv (\nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots$ $\ldots \mid P_{i'j'} \mid \ldots))\{a'/a\}$ $L' \equiv L$
4	$\exists i,j.\ S_{i,j} = a(b).P_{i,j} \ \wedge\ a \in \mathit{free}(P) \cup L$	xy	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots)$ $(c \notin bn(P) \vee c \in L) \ \wedge\ L \equiv L'$
5	$\exists i,j.\ S_{i,j} = \overline{a}c.P_{i,j} \ \wedge\ a \in \mathit{free}(P) \cup L$	$\overline{x}y$	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij} \mid \ldots)$ $L' \equiv \begin{cases} L & \text{if } c \in \mathit{free}(P) \\ L : c & \text{otherwise} \end{cases}$

where substitution is capture-avoiding, *i.e.* α-conversion is possibly applied before applying substitution; $x \equiv \begin{cases} a & \text{if } a \in \mathit{free}(P) \\ \nu a & \text{otherwise} \end{cases}$ and $y \equiv \begin{cases} c & \text{if } c \notin bn(P) \\ \nu c & \text{otherwise} \end{cases}$

The above relation on names can be naturally extended to labels. Then, the GIPO bisimilarity can be recovered on standard π-calculus as the canonical bisimilarity induced by the LTS above, up-to the use of the relation $=_{LM}$ on labels instead of equality. That is, for P, Q processes on the standard syntax, $\emptyset$ the empty list, and $\mathcal{T}(P), \mathcal{T}(Q)$ the translations of P,Q in the syntax with de Brujin indexes, we have:

Theorem 1. $(P, \emptyset) \sim (Q, \emptyset)$ *iff* $\mathcal{T}(P) \sim_C \mathcal{T}(Q)$.

3.2 GIPO Bisimilarity vs Syntactical and Open Bisimilarity

One can check that the GIPO bisimilarity coincides with the syntactical bisimilarity introduced in [3] for the π-calculus fragment without the ν-operator. Syntactical bisimilarity is a variant of the open bisimilarity, obtained by requiring that a transition with a fusion label is simulated by a transition with the same fusion (and not by a possibly τ-transition). A stronger result holds, that is a small variation of our bisimilarity $\sim$ coincides with the open bisimilarity $\sim_O$ on the full calculus. Namely, let $\approx$ denote the bisimilarity obtained from $\sim$ by allowing a fusion transition with label $\{a'/a\}$ to be simulated either by the same fusion or by a τ-transition. The asymmetric definition of $\approx$ is reminiscent of the *semi-saturated bisimilarity* introduced in [6]. We have:

Theorem 2. $\approx = \sim_O$.

The above theorem (whose proof is sketched in Appendix B) gives us a new efficient characterization of the open bisimilarity. The most evident difference

between our presentation and the standard symbolic presentation is that in the latter distinctions are needed, while we do not use them. An explanation for this is that, when comparing two terms that can perform an input transition, the open bisimilarity considers just one transition on a free name, while we need to consider also all the transitions, where a previously communicated bound name (contained in the list L) is received.

4 Conclusions and Future Work

We have applied the GRPO construction to the full π-calculus, using two extra important ingredients. Firstly, we have worked in a category of second-order contexts, based on a presentation of π-calculus with de Bruijn indexes. Secondly, a general pruning technique has been applied, in order to simplify the LTS obtained by the standard (G)RPO construction. Finally, the application of a more ad-hoc simplification technique has allowed us to get an efficient LTS and bisimilarity, and a new characterization of Sangiorgi's open bisimilarity. As it often happens, also in the present case Leifer-Milner technique by itself does not directly give an efficient LTS and bisimilarity. However, this technique, applied in the setting of second-order contexts and in combination with our general pruning technique, gives us substantially less redundant LTS's and bisimilarities, and leads us to the final efficient presentation. Moreover, new insights on the calculus are obtained by applying this machinery. The construction presented in this paper is solid under variations of π-calculus syntax, *e.g.* including replication or match/mismatch operators. In conclusion, the results obtained for π-calculus in this paper and for CCS in [7] are quite promising; in particular, they show that the Leifer-Milner technique is valuable in suggesting interesting notions of LTS's and bisimilarities. Therefore, it would be worth to experiment the above machinery on more recent calculi, for which the notions of LTS and bisimilarity are still evolving.

References

1. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: CONCUR. Volume 1877 of LNCS., Springer (2000) 243–258
2. Sassone, V., Sobocinski, P.: Deriving bisimulation congruences using 2-categories. Nord. J. Comput. **10** (2003) 163–190
3. Ferrari, G.L., Montanari, U., Tuosto, E.: Model checking for nominal calculi. In Sassone, V., ed.: FoSSaCS. Volume 3441 of LNCS., Springer (2005) 1–24
4. Gadducci, F., Montanari, U.: Observing reductions in nominal calculi via a graphical encoding of processes. In: Processes, Terms and Cycles. Volume 3838 of LNCS., Springer (2005) 106–126
5. Bonchi, F., Gadducci, F., König, B.: Process bisimulation via a graphical encoding. In: ICGT. Volume 4178 of LNCS., Springer (2006) 168–183
6. Bonchi, F., König, B., Montanari, U.: Saturated semantics for reactive systems. In: LICS, IEEE Computer Society (2006) 69–80

7. Di Gianantonio, P., Honsell, F., Lenisa, M.: Finitely branching labelled transition systems from reaction semantics for process calculi. In: WADT. Volume 5486 of LNCS., Springer (2009) 119–134
8. Bonchi, F., Gadducci, F., Monreale, G.V.: Reactive systems, barbed semantics, and the mobile ambients. In de Alfaro, L., ed.: FOSSACS. Volume 5504 of Lecture Notes in Computer Science., Springer (2009) 272–287
9. Di Gianantonio, P., Honsell, F., Lenisa, M.: RPO, second-order contexts, and lambda-calculus. Logical Methods in Computer Science **5** (2009)
10. Jensen, O.H., Milner, R.: Bigraphs and transitions. In: POPL. (2003) 38–49
11. Sobocinski, P.: A well-behaved lts for the pi-calculus: (abstract). Electr. Notes Theor. Comput. Sci. **192** (2007) 5–11
12. Parrow, J.: An introduction to the pi-calculus. In Bergstra, Ponse, Smolka, eds.: Handbook of Process Algebra, Elsevier (2001) 479–543
13. Sangiorgi, D.: A theory of bisimulation for the pi-calculus. Acta Inf. **33** (1996) 69–97
14. Sobocinski, P.: Deriving process congruences from reduction rules. PhD thesis, University of Aarhus (2004)
15. Jensen, O.H., Milner, R.: Bigraphs and transitions. In: POPL, ACM (2003) 38–49

A Appendix: Reaction Systems in the G-category Setting

Definition 10 (G-Category). *A 2-category $\mathcal{C}$ consists of*

- *A set of objects: A, B, C, ...*
- *For any pair of objects $A, B \in \mathcal{C}$, a category $\mathcal{C}(A, B)$. Objects in $\mathcal{C}(A, B)$ are called* 1-cells morphisms*, and denoted by $f : A \to B$. Arrows in $\mathcal{C}(A, B)$ are called* 2-cells isomorphisms *and represented by $\alpha : f \Rightarrow g$ or by* $A \overset{f}{\underset{g}{\overset{\Downarrow\alpha}{\rightrightarrows}}} B$ *. Composition in $\mathcal{C}(A, B)$, called* vertical composition*, is denoted by $\bullet$.*
- *For all objects A, B and C, there is a functor $\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$, called* horizontal composition*, which is associative and admits the identity 2-cells of id_A as identities.*

A G-category is a 2-category whose 2-cells morphisms are all isomorphisms.

Definition 11 (G-Reaction System). *A* G-reaction system $\mathbf{C}$ *consists of:*

- *a G-category $\mathcal{C}$;*
- *a distinguished object $0 \in |\mathcal{C}|$;*
- *a collection $\mathcal{D}$ of 1-cells morphisms, in $\mathcal{C}$. $\mathcal{D}$ is referred as the set of* reaction contexts*, it is required to be closed under 2-cells, and to reflect composition.*
- *a set of pairs $\mathbf{R} \subseteq \bigcup_{I \in |\mathcal{C}|} \mathcal{C}[0, I] \times \mathcal{C}[0, I]$ of* reaction rules*.*

The reaction contexts are those in which a reaction can occur. By composition-reflecting we mean that $d \circ d' \in \mathcal{D}$ implies $d, d' \in \mathcal{D}$, while by closure under 2-cells we mean that if $d \in \mathcal{D}$, $\alpha : d \Rightarrow d'$ then $d' \in \mathcal{D}$.

Definition 12 (GRPO/GIPO).

(i) *Let $\mathcal{C}$ be a G-category and let us consider the commutative diagram in Fig. 1(i). Any tuple $\langle I_5, e, f, g, \beta, \gamma, \delta\rangle$ which makes diagram in Fig. 1(ii) commute and such that $\delta l \bullet g\beta \bullet \gamma t = \alpha$ is called a* candidate *for (i).*

(ii) *A* G relative pushout (RPO) *is the smallest such candidate, i.e. it satisfies the universal property that given any other candidate $\langle I_6, e', f', g', \beta', \gamma', \delta'\rangle$, there exists a mediating morphism given by a tuple $\langle h, \varphi, \psi, \tau\rangle$, with $\tau : g'h \Rightarrow g$, such that diagrams in Fig. 1(iii) commute. Moreover, the following identities on two cells need to be satisfied: $\gamma = \tau e \bullet g'\varphi \bullet \gamma'$, $\delta = \delta' \bullet g'\psi \bullet \tau^{-1} f$, $\beta' = \psi l \bullet h\beta \bullet \varphi t$. Such a mediating morphism must be unique, up to 2-cell isomorphisms.*

(iii) *A commuting square such as diagram in Fig 1(i) is a* G-idem pushout (GIPO) *if $\langle I_4, c, d, id_{I_4}, \alpha, 1_c, 1_d\rangle$ is its GRPO.*

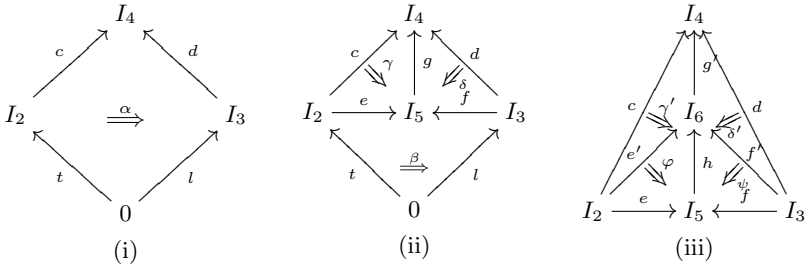

Fig. 1. Redex Square and Relative Pushout.

Definition 13 (GIPO Transition System).

- *States: equivalence classes of arrows $[t] : 0 \to I$ in $\mathcal{C}$, for any I; two arrows are in the same equivalence class if there exists a 2-cell isomorphism between them;*
- *Transitions: $[t] \xrightarrow{[c]}_I [dr]$ iff $d \in \mathcal{D}$, $\langle l, r\rangle \in \mathbf{R}$ and the diagram in Fig. 1(i) is a GIPO.*

An important property of GIPO squares is that they are preserved by the substitution of one edge with a two 2-cell isomorphic one, [14]. It follows that the transition relation is independent from the chosen representative of an equivalence class. Let $\sim_G$ denote the bisimilarity induced by the GIPO LTS.

Another important property is the pasting property for GIPO squares.

Lemma 1 (GIPO pasting, [14]). *Suppose that the square in Fig. 2(i) has an GRPO and that both squares in Fig. 2(ii) commute.*

(i) *If the two squares of Fig. 2(ii) are GIPOs so is the outer rectangle.*

(ii) *It the outer rectangle and the left square of Fig. 2(ii) are GIPOs so is the right square.*

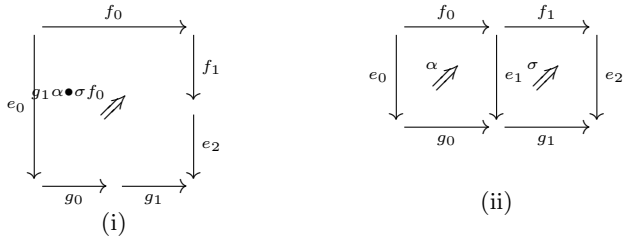

Fig. 2. GIPO pasting.

Definition 14 (Redex GRPO). *Let* $\mathbf{C}$ *be a G-reaction system and* $t : 0 \to I_2$ *an arrow in* $\mathcal{C}$*. A* redex square *is a diagram in the form of Fig. 1(i), with* l *the left-hand side of a reaction rule and* d *a reaction context. A G-reaction system* $\mathbf{C}$ *is said to* have redex GRPOs *if every redex square has a GRPO.*

The following fundamental theorem is provable using the GIPO pasting lemma:

Theorem 3. *Let* $\mathbf{C}$ *be a G-reaction system having redex GRPOs. Then the GIPO bisimilarity* $\sim_G$ *is a congruence w.r.t. all contexts, i.e. if* $a \sim_G b$ *then for all* c *of the appropriate type,* $ca \sim_G cb$*.*

A.1 Pruning the GIPO LTS

In this section we present a construction (which has been firstly introduced in [9]), allowing to prune the LTS obtained by the GIPO construction. In this way it is possible to derive simpler and more usable LTS's. The key notion is that of *definability*. In a GIPO LTS, the GIPO transitions that are "definable" in some suitable sense can be removed without affecting the bisimilarity induced by the LTS.

Definition 15. *Given a G-reaction system* $\mathbf{C}$*, having redex GRPOs, let* $\mathcal{T}$ *be a subset of the whole set of GIPO transitions,*

(i) we say that $\mathcal{T}$ *is closed under bisimilarity if for any* $[t_1], [t_1'], [t_2], [t_2'], [f]$*, such that* $[t_1] \sim_G [t_1']$*,* $[t_2] \sim_G [t_2']$*,* $[t_1] \xrightarrow{[f]}_I [t_2]$*,* $[t_1'] \xrightarrow{[f]}_I [t_2']$*, we have that:* $[t_1] \xrightarrow{[f]}_I [t_2] \in \mathcal{T}$ *iff* $[t_1'] \xrightarrow{[f]}_I [t_2'] \in \mathcal{T}$

(ii) we say that the whole GIPO LTS is definable *from* $\mathcal{T}$ *if there exists a set of triples* $\{ \langle [f_k], [f_k'], e_k \rangle | k \in K \}$ *of the following form:*
- $[f_k]$ *GIPO label,* $[f_k']$ *GIPO label or* $f_k' = *$ *with* $f_k : I_k \to I_k'$*,* $f_k' : I_k \to J_k$ *(where we set* $* : I_k \to I_k$*)*
- $e_k : J_k \to I_k$ *(with* J_k *possibly* 0*)*

and such that, in the whole GIPO LTS, there is a transition $[t] \xrightarrow{[f]}_I [t']$ *if and only if there exist* $k \in K$*,* $t'' : 0 \to J_k$ *satisfying:*
- $[f] = [f_k]$*,*
- $([t] \xrightarrow{[f_k']}_I [t''] \in \mathcal{T})$ *or* $(t'' = t \wedge f_k' = *)$
- $([t'] = [e_k(t'')] \wedge J_k \neq 0)$ *or* $([t'] = [e_k] \wedge J_k = 0)$

Intuitively a tuple $\langle [f_k], [f'_k], e_k \rangle$ says that some of the transitions with label $[f_k]$ can be simulated by transitions with label $[f'_k]$ and contexts e_k. We allow the extra case of $f'_k = *$ to deal with those transitions that can be simulated by just inserting the original term in a contexts e_k, following [15] we can call *not engaged* these kind of transitions.

Definition 15 above is a special case of Definition 9 of [7], where the notion of definability involves also a sort of Hennessy-Milner propositions.

Proposition 5 ([7]). *Given a reaction system* $\mathbf{C}$*, and a subset* $\mathcal{T}$ *of transitions that is closed under GIPO bisimilarity and such that the whole GIPO LTS is definable from* $\mathcal{T}$*, then* $\sim_G = \sim_{\mathcal{T}}$*, i.e. the two GIPO LTS induce the same bisimilarity.*

B Appendix

Proof of Proposition 2.
In order to prove that the G-reaction system $\mathbf{C}_\pi$ has redex GRPOs, it is necessary to construct, for any possible redex square, the corresponding GRPO.

A remark about variable substitutions in GRPO constructions. Given a commuting square in the form $\alpha : C[\ ]_{\theta_1} \circ P \Rightarrow D[\ ]_{\theta_2} \circ L$, the square remains commuting also modifying the behavior of θ_1 and θ_2 on the variables not appearing in P and L, respectively. Since in the GRPOs need to consider the most general substitution, it follows that, in any GRPO square having form $\alpha : C[\ ]_{\theta_1} \circ P \Rightarrow D[\ ]_{\theta_2} \circ L$, for any Z_1 not appearing in P, $\theta_1(Z_1)$ must be a variable not appearing in the codomain of θ_2. A symmetric condition holds for any variable Z_2 not appearing in L. Since P and L contain just a finite number of variables, it follows that the substitutions θ_1 and θ_2, for all but a finite number of variables, map variables into variables and have disjoint codomains. In particular, in the redex GRPO that we define, the substitution θ_1, in all but a finite number of cases, maps a variable with index i into a variable with index $2i$, while the substitution θ_2, in all but a finite number of cases, maps a variable into a variable with odd index. For the variables appearing in P and L, the substitutions θ_1 and θ_2 are obtained applying a sort of unification algorithm, as described in [9]. A final condition that the substitutions θ_1 and θ_2 need to satisfy is that any variable must appear at most once in each one of their codomains (linearity condition) and at least once in the union of the codomains.

In more detail, considering a commuting redex square $\alpha : \mathbb{C} \circ P \Rightarrow \mathbb{D} \circ L$, with $L \equiv r.P + M \mid \overline{r}.Q + N \rightarrow P|Q$ the left hand side of the communication rule, we have that the context $\mathbb{C}$ can be written as $\nu^m(\mathbb{C}' \circ \sigma_1[\ ]_{\theta_1})$, with $\mathbb{C}'$ not containing the ν operator and having name substitutions only applied to variables, while the reaction context $\mathbb{D}$ can be written as $\nu^n(\sigma[\ ]_{\theta_2}|P_2)$.

If the redex L is contained (or better mapped by α^{-1}) in the process P, the GRPO has form $\alpha' : \sigma_1[\ ] \circ P \Rightarrow (\nu^m \sigma_2[\ ]_{\theta_2}|P_2) \circ L$. Notice that the name substitution σ_1, if different from a bijection, cannot be factorized by the GRPO construction.

If the process P contains only one side of the redex L, the other side of the redex can be obtained by variable substitution and in this case the GRPO has form $\alpha' : (\sigma_1[\,]_{\{rX+Y/Z\}\circ\delta}) \circ P \Rightarrow (\nu^m \sigma_2[\,]_{\theta_2} | P_2) \circ L$, or the other side of the redex can be generated by the context with a GRPO having form $\alpha' : (\sigma_1[\,]|P') \circ P \Rightarrow (\nu^m \sigma_2[\,]_{\theta_2} | P_{i_1} | \ldots | P_{i_k}) \circ L$.

If the redex L is contained in the context $\mathbb{C}$, the GRPO has form $\alpha' : \mathbb{C}'' \circ \sigma_1[\,] \circ P \Rightarrow (\nu^m[\,]_{\theta_2} | P_2) \circ L$, with φP contained in P'.

If the process P is contained in the redex L (instantiated with θ_2) the GRPO has the form $\alpha' : \mathbb{C}'' \circ \sigma_1[\,] \circ P \Rightarrow ([\,]_{\theta_2}) \circ L$.

Table 1 of Section 2 summarizes the GIPO contexts for every possible process. An analysis of the various cases appear in Section 2. □

Proposition 6.
(i) The GIPO LTS of Table 1 is definable from the set R of reduced GIPO contexts.
(ii) The bisimilarity $\sim_R$ induced by the LTS defined by the reduced GIPO contexts coincides with the original GIPO bisimilarity $\sim_G$, and it is a congruence.

Proof. (i) All transitions corresponding to GIPO contexts in raws 3 and 13 of Table 1 can be easily shown to be definable by suitable triples with $*$ as second element. Transitions corresponding to GIPO contexts in raw 1 of Table 1 are definable by triples of the shape $\langle \beta[\,]_\delta, [\,]_\delta, \beta[\,]_{id} \rangle$. Transitions corresponding to GIPO contexts in raw 2 of Table 1 are definable by triples $\langle \beta[\,]_{\{(\tau.X_1+Y_1)/\delta Z\}\circ\delta}$, $[\,]_{\{(\tau.X_1+Y_1)/\delta Z\}\circ\delta}$, $\beta[\,] \rangle$. Transitions corresponding to GIPO contexts in raws 4–9' of Table 1 can be easily shown to be definable by the corresponding reduced GIPO contexts, obtained by taking β to be the identity. Transitions corresponding to GIPO contexts in raws 10,10' of Table 1 are definable by τ-transitions induced by the reduced GIPO contexts in raw 2, when $r = r'$; the defining tuple, *e.g.* for raw 10, being $\langle \beta[\,]_{\{((r().X_1+Y_1) \,|\, (\overline{r}r_l.X_3+Y_3))/\delta X\}\circ\delta}$, $[\,]_{\{(\tau.X_1+Y_1)/\delta X\}\circ\delta}$, $\beta[\,]_{\{((\nu t_{0,l+1}X_1)|X_3)/X_1\}} \rangle$. Transitions corresponding to GIPO contexts in raw 10', when $r \neq r'$, are definable by transitions corresponding to raw 10, via the tuple $\langle \beta[\,]_{\{((r().X_1+Y_1) \,|\, \nu(\overline{r}'r_0.X_3+Y_3))/\delta X\}\circ\delta}$, $\beta[\,]_{\{((r().X_1+Y_1) \,|\, (\overline{r}'r_l.X_3+Y_3))/\delta X\}\circ\delta}$, $\beta[\,]_{\{0/X_1,\, \nu(\nu\{r_1/r_0\}(\delta_0)_{+1}X_1|X_3)/X_3\}} \rangle$. Finally, transitions induced by GIPO contexts in raws 11-12 can be easily shown to be definable by the corresponding reduced GIPO contexts.
(ii) The proof follows from Proposition 5, by showing that transitions induced by the reduced GIPO contexts are closed under bisimilarity, according to Definition 15. □

Lemma 2. *Let $\sim_F$ denote the bisimilarity induced by the final GIPO contexts (column F of Table 1) and let $\mathcal{R}$ be the relation formed by the pairs of processes in the form $\langle \nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n),\ \nu^k(\sigma Q \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n) \rangle$, and such that there exist a integer l and two name substitutions σ', σ_0 satisfying the following list of conditions.*

(i) $\nu^l(\sigma' P \mid \sigma_0 X_0) \sim_F \nu^l(\sigma' Q \mid \sigma_0 X_0)$
(ii) $\forall r, i \,.\, \sigma(r) \in cod(\sigma_i) \Rightarrow \sigma'(r) \in cod(\sigma_0)$;

(iii) $\forall r \,.\, \sigma(r) \geq k \Rightarrow \sigma'(r) \in cod(\sigma_0)$;
(iv) $\forall r \,.\, \sigma'(r) < l \Rightarrow \sigma(r) < k$;
(v) $\forall r, s \,.\, (\sigma'(r) < k \ \wedge \ \sigma'(s) < k) \Rightarrow (\sigma'(r) = \sigma'(s) \Leftrightarrow \sigma(r) = \sigma(s))$
(vi) *For any pair of instances of name references* r, s *appearing in* P *(in* Q*)* $[\![\sigma' P, r]\!] = [\![\sigma' P, s]\!] \ \Rightarrow \ [\![\sigma P, r]\!] = [\![\sigma P, s]\!]$ *(*$[\![\sigma' Q, r]\!] = [\![\sigma' Q, s]\!] \ \Rightarrow \ [\![\sigma Q, r]\!] = [\![\sigma Q, s]\!]$*).*

The relation $R \cup \sim_F$ *is a bisimulation w.r.t. the reduced GIPO LTS.*

Proof. We show that the relation $R \cup \sim_F$ is a bisimulation w.r.t. the reduced GIPO LTS, using the following schema: given a generic pair of terms $\langle \nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n), \ \nu^k(\sigma Q \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n) \rangle \in R$, and given an integer l and substitutions σ', σ_0 for which conditions (i)...(vi) are satisfied, for any transition $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n) \xrightarrow{\mathbf{1}}_I P^*$ in the reduced LTS, it is possible to show that there exists a second transition $\nu^l(\sigma' P \mid \sigma_0 X_0) \xrightarrow{\mathbf{1}'}_I P'^*$, in the final LTS, mimicking the first one. By similarity, there exists Q'^* such that $\nu^l(\sigma' Q \mid \sigma_0 X_0) \xrightarrow{\mathbf{1}'}_I Q'^*$ and, using conditions (ii), ..., (vi), it is possible to show that there exists Q^* such that $\nu^k(\sigma Q \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n) \xrightarrow{\mathbf{1}}_I Q^*$ and $(P^*, Q^*) \in R$. Moreover, to complete the proof, we show if $P \sim_F Q$, for any transition $P \xrightarrow{\mathbf{1}}_I P^*$ in the reduced LTS, then there exists a second transition $P \xrightarrow{\mathbf{1}'}_I P'^*$ in the final LTS, mimicking the first one. By similarity, there exists Q'^* such that $Q \xrightarrow{\mathbf{1}'}_I Q'^*$ and from this fact it follows that there exists Q^* such that $Q \xrightarrow{\mathbf{1}}_I Q^*$ and $(P^*, Q^*) \in R$

Since there are many different kinds of transitions to consider, the proof is lengthy. For shortness, here we present just two meaningful cases, the others follows a similar schema.

Before giving in the details of the proof, we present the following consequences of conditions (ii), (iv) and (v), which motivate their definition.

(ii) for any instance of name reference r appearing in P and for any substitution of the variable X_i with $s().X_i$, if r and s denote the same name in $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_i(s().X_i) \mid \ldots \mid \sigma_n X_n)$, then there exists an instantiation $s'().X_0$ for the variable X_0 such that r and s' denotes the same name in $\nu^l(\sigma' P \mid \sigma_0(s'().X_0))$;
(iv) for any instance of name reference r appearing in P, if r denotes a private name in $\nu^l(\sigma' P \mid \sigma_0 X_0)$, then it denotes a private name in $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n)$;
(v) for any pair of instances of name reference r, s appearing in P, if r, s denote private names in $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n)$, then they denote the same name in $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n)$ if and only if they denote the same name in $\nu^l(\sigma' P \mid \sigma_0 X_0)$.

First case (transition 11):

Suppose that $P \equiv r() P_1 + P_2 \mid P_3$ and suppose that $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n) \equiv \ \nu^k(\sigma(r)().\sigma_{+1} P_1 + \sigma P_2 \mid \sigma P_3 \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X_n)$ makes a transition

with label $\iota[\ ]_{\{\overline{r'}s.X_j/X_j\}}$ becoming $\nu^k(\nu\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}P_1 \mid \iota_{+k}\sigma P_3 \mid \iota_{+k}\sigma_1 X_1 \mid \ldots \mid \iota_{(k}\sigma_m X_m) \equiv \nu^{k+1}(\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}P_1 \mid \delta_0\iota_{+k}\sigma P_3 \mid \delta_0\iota_{+k}\sigma_1 X_1 \mid \ldots \mid \delta_0\iota_{(k}\sigma_m X_m) \equiv \nu^{k+1}(\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}P_1 \mid \{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\delta_0\iota_{+k}\sigma P_3 \mid \delta_0\iota_{+k}\sigma_1 X_1 \mid \ldots \mid \delta_0\iota_{+k}\sigma_m X_m) \equiv \nu^{k+1}(\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}(P_1 \mid \delta_0 P_3) \mid \delta_0\iota_{+k}\sigma_1 X_1 \mid \ldots \mid \delta_0\iota_{+k}\sigma_m X_m)$. This transition is mimicked by $\nu^l(\sigma' P \mid \sigma_0 X_0)$ with a transition having label $[\ ]_{\{\overline{r''}s'.X_0/X_0\}}$, and transforming it in $\nu^{l+1}(\{\delta_0\sigma_0(s')/\mathbf{r_0}\}\sigma'_{+1}(P_1 \mid \delta_0 P_3) \mid \delta_0\sigma_0 X_0)$. The name references r'', s' appearing in the label, are defined by cases:

- The name reference r'' is chosen in such a way that $\sigma'(r) = \sigma_0(r'')$. The argument proving the existence of r'' is the following, if r denotes a private name in $\nu^l(\sigma' P \mid \sigma_0 X_0)$ then, by condition (iv), it denotes a private name also in $\nu^k(\sigma P \mid \sigma_1 X_1 \mid \ldots \mid \sigma_n X^n)$, it follows that ι has to be the identity and by condition (ii) r'' exists. On the other hand, it r denotes a free name in $\nu^l(\sigma' P \mid \sigma_0 X_0)$, then r'' exists by condition (iii)
- If there exists s'' such that $\sigma_j(s) = \sigma(s'')$, s' is chosen in such a way that $\sigma_0(s') = \sigma'(s'')$, by condition (ii), such s exists;
- otherwise s' is chosen to denote a free name, not denoted by any other name reference present in P and Q, or by the r'' previously defined.

By condition (i), $\nu^l(\sigma' Q \mid \sigma_0 X_0)$ can make a transition with label $[\ ]_{\{\overline{r''}s'.X_0/X_0\}}$, by simple arguments, it follows that Q can be written in the form $(r()Q_1 + Q_2) \mid Q_3$, and the result of the transition can be written in the form $\nu^{l+1}(\{\delta_0\sigma_0(s')/\mathbf{r_0}\}\sigma'_{+1}(Q_1 \mid \delta_0 Q_3) \mid \delta_0\sigma_0 X_0)$. It is not difficult to show that $(\nu^{k+1}(\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}(P_1 \mid \delta_0 P_3) \mid \delta_0\iota_{+k}\sigma_1 X_1 \mid \ldots \mid \delta_0\iota_{+k}\sigma_m X_m)$, $\nu^{k+1}(\{\delta_0\iota_{+k}\sigma_j(s)/\mathbf{r_0}\}\iota_{+(k+1)}\sigma_{+1}(Q_1 \mid \delta_0 Q_3) \mid \delta_0\iota_{+k}\sigma_1 X_1 \mid \ldots \mid \delta_0\iota_{+k}\sigma_m X_m) \in R$, from which the thesis. □

Proof of Proposition 4.
Given two closed processes P and Q, since $\sim_F$ is a congruence, it follows that $P \sim_F Q \Leftrightarrow P \mid X \sim_F Q \mid X$.

Let consider the relation $\mathcal{R}'$ whose definition coincides with the relation $\mathcal{R}$, presented in the proof of the previous lemma, except for point (i) that becomes

(i') P, Q are closed processes,
(i") $\nu^l(\sigma' P \mid \sigma_0 X_0) \sim_C \nu^l(\sigma' Q \mid \sigma_0 X_0)$

By a subset of the arguments used in the previous lemma, it is possible to prove that relation $\mathcal{R}' \cup \sim_F$ is a F-bisimulation, and therefore it is contained in $\sim_F$. Immediately, $P \mid X \sim_C Q \mid X \Rightarrow \langle P \mid X,\ Q \mid X\rangle \in \mathcal{R}' \Rightarrow P \mid X \sim_F Q \mid X$.

The inverse implication follows from the fact that the LTS for closed processes is a subset of final LTS. □

Proof of Theorem 2.(Sketch)
The proof follows from the fact that, for any process $\nu\boldsymbol{a}P$, with P ν-free, there is a correspondence between derivations in our LTS and derivations in the symbolic LTS, when in this latter we keep track of distinctions (and we do not permit

fusions of names in the distinctions). That is, for L_0 the empty list and D_0 the empty distinction, we have: $(\nu \boldsymbol{a} P, L_0) \xrightarrow{\alpha'_1} (\nu \boldsymbol{a} P'_1, L_1) \ldots \xrightarrow{\alpha'_n} (\nu \boldsymbol{a} P'_n, L_n)$ iff $(\nu \boldsymbol{a} P, D_0) \xrightarrow{\alpha_1} (\nu \boldsymbol{a}'_1 P_1, D_1) \ldots \xrightarrow{\alpha_n} (\nu \boldsymbol{a}'_n P_n, D_n)$, where for all i, $\boldsymbol{a}'_i$ is a subset of $\boldsymbol{a}$, at each step the correspondent transitions in the two terms are performed, and $P'_i = P_i \sigma_i$, for σ_i the composition of the singleton substitutions arising from input actions and fusions. □