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 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 redundant 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 the extended version of the present paper [12], the theory of RPO's in the G-category setting and the general pruning technique of [7] are recalled, and some proofs are presented.

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*

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

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:

$$\widehat{\theta}(Z) = \theta(Z) \qquad \widehat{\theta}([\ ]_{\theta'}) = [\ ]_{\widehat{\theta}\circ\theta'}$$
$$\widehat{\theta}(T_1 + T_2) = \widehat{\theta}(T_1) + \widehat{\theta}(T_2) \qquad \widehat{\theta}(T_1 \mid T_2) = \widehat{\theta}(T_1) \mid \widehat{\theta}(T_2)$$
$$\widehat{\theta}(\sigma T) = \sigma\widehat{\theta}(T) \qquad \widehat{\theta}(\nu T) = \nu(\widehat{\theta}(T))$$
$$\widehat{\theta}(rec\ X.T) = rec\ X.\widehat{\theta'}(T)\ , \qquad \text{where } \theta'(Z) = \begin{cases} \theta(Z) & \text{if } Z \neq X \\ X & \text{if } Z = X \end{cases}$$

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:*

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

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

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.* [13]. 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. $(r().X_1 + Y_1) \mid (\overline{r}r_j.X_2 + Y_2) \rightarrow (\nu(t_{0,j+1}X_1)) \mid X_2$

$\tau.X + Y \rightarrow X$

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 $P \rightarrow Q$ in the ordinary reaction system iff $\mathcal{T}(P) \rightarrow \mathcal{T}(Q)$ 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 [12], 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 $\mathbb{C} \circ t \rightarrow v$, and for no subcontext $\mathbb{C}'$ of $\mathbb{C}$ and no subterm v' of v, there is a reaction $\mathbb{C}' \circ t \rightarrow v'$. 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, based on the notion of finite approximation, show that, in the extended LTS, the bisimilarity is still a congruence. 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 $\mathbb{C}^{\mathcal{B}}_{\mathcal{A}}$ generated by the grammar:*

$$\mathbb{C}^{\mathcal{G}}_{\mathcal{G}} ::= [\,]_\theta \mid \alpha.\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} \mid \mathbb{C}^{\mathcal{G}}_{\mathcal{G}} + M \mid M + \mathbb{C}^{\mathcal{G}}_{\mathcal{G}}$$
$$\mathbb{C}^{\mathcal{G}}_{\mathcal{P}} ::= \alpha.\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \mathbb{C}^{\mathcal{G}}_{\mathcal{P}} + M \mid M + \mathbb{C}^{\mathcal{G}}_{\mathcal{P}}$$
$$\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} ::= \mathbb{C}^{\mathcal{G}}_{\mathcal{G}} \mid \nu\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} \mid \mathbb{C}^{\mathcal{P}}_{\mathcal{G}}|P \mid P|\mathbb{C}^{\mathcal{P}}_{\mathcal{G}} \mid \sigma\mathbb{C}^{\mathcal{P}}_{\mathcal{G}}$$
$$\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} ::= [\,]_\theta \mid \mathbb{C}^{\mathcal{G}}_{\mathcal{P}} \mid \nu\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \mathbb{C}^{\mathcal{P}}_{\mathcal{P}}|P \mid P|\mathbb{C}^{\mathcal{P}}_{\mathcal{P}} \mid \sigma\mathbb{C}^{\mathcal{P}}_{\mathcal{P}}$$

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.*

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: (1) already present in P, (2) by instantiating variables in P, (3) appearing in the context;
- in case of variable instantiation by an output action, the name sent can be either private or public. A detailed description of the GIPO contexts of Table 1 and a proof of the above proposition appear in [12].

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]. 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 [12]. 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. We define an LTS, F, composed by the GIPO contexts marked by $\star$ in Table 1. The proof that the bisimilarity induced by the LTS F coincides with the GIPO bisimilarity is based on the technique of the "bisimulation up-to", and it appears in [12].

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 simpler LTS, namely the LTS of 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. Intuitively, X plays the rôle of the environment, that can send to or receive names from P'; the name substitution σ records the names received from P'.

Here we present a detailed description of transitions in Table 2. Row 1 corresponds to a τ-reaction. Row 2 corresponds to the case where the process P exposes two complementary actions. In this rule 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. 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 3 and 4 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 these rules the variable 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. In row 5 the GIPO context instantiates the variable X by the whole communication redex.

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 $P \mid X \sim_C Q \mid X$.*

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.

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, [14]. 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, [14]. An advantage of our presentation lies in the fact that our bisimilarity has a direct definition from 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. 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.

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 this presentation we use a traditional approach.

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.

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	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots \mid P_{i'j'} \mid \ldots)$ $L' \equiv L$
3	$\exists i, j, i', j'.\ (i \neq i' \ \wedge\ S_{i,j} = a(b).P_{i,j} \ \wedge\ S_{i',j'} = \overline{a}'c.P_{i',j'})$ $a, a' \in \mathit{free}(P),\ a < a'$	$\{a'/a\}$	$P' \equiv (\nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots \mid P_{i'j'} \mid \ldots))\{a'/a\}$ $L' \equiv L$
4	$\exists i, j.\ S_{i,j} = a(b).P_{i,j} \ \wedge\ a \in \mathit{free}(P) \cup L$	xy	$P' \equiv \nu \boldsymbol{a}(\ldots \mid P_{ij}\{c/b\} \mid \ldots)$ $(c \notin \mathit{bn}(P) \vee c \in L) \ \wedge\ L \equiv L'$
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 $y \equiv \begin{cases} c & \text{if } c \notin \mathit{bn}(P) \\ \nu c & \text{otherwise} \end{cases}$

The resulting LTS is quite similar to symbolic LTS, the main difference being that, for input transitions, in the symbolic LTS just one name is considered, while in the present LTS also previously communicated private names (recorded in the list) are considered.

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* $x = \nu a \ \wedge\ y = \nu a' \ \wedge\ \forall i.\ (a = L(i) \iff a' = M(i))$.

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

On the π-fragment without the ν-operator, the list L disappears in our LTS, hence the GIPO bisimilarity coincides with the *syntactical bisimilarity*, which is defined as the bisimilarity induced by the symbolic LTS on the π-fragment without ν [3]. Syntactical bisimilarity is a variant of the open bisimilarity. The latter is defined on the symbolic LTS, by relaxing the condition on fusion transitions: a fusion transition $P \xrightarrow{\{a'/a\}} P'\{a'/a\}$ can be simulated either by a transition with the same fusion label or by a τ-transition $Q \xrightarrow{\tau} Q'$ such that $P'\{a'/a\}$ and $Q'\{a'/a\}$ are bisimilar. The standard definition of open bisimilarity on the full calculus uses the extra machinery of *distinctions*. An interesting result that we obtain is that 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 to be simulated either by the same fusion or by a τ-transition (where in the resulting process we apply the fusion). Then we have:

Theorem 2. $\approx = \sim_O$.

The above theorem (whose proof is sketched in [12]) gives us a new efficient characterization of the open bisimilarity. The most evident difference between our presentation and the standard presentation is that in the latter distinctions are needed, while we use lists in the LTS but no distinctions. 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 consider also all the transitions, where a previously communicated bound name (contained in the list L) is received.

Finally, notice that the asymmetric definition of $\approx$ for fusion labels follows the pattern of the *semi-saturated bisimilarity* introduced in [6].

4 Conclusions and Future Work

We have applied the GRPO construction to the full π-calculus, using two extra 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**(2) (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**(3) (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**(1) (2007) 5–11
12. Di Gianantonio, P., Jaksic, S., Lenisa, M.: Efficient bisimilarities from second-order reaction semantics for pi-calculus. Technical report, Università di Udine, available at http://sole.dimi.uniud.it/∼marina.lenisa/Papers/Soft-copy-pdf/tr10.pdf (2010)
13. Parrow, J.: An introduction to the pi-calculus. In Bergstra, Ponse, Smolka, eds.: Handbook of Process Algebra, Elsevier (2001) 479–543
14. Sangiorgi, D.: A theory of bisimulation for the pi-calculus. Acta Inf. **33**(1) (1996) 69–97