Finitely Branching LTS's from Reaction Semantics for Process Calculi

Pietro Di Gianantonio   Furio Honsell   Marina Lenisa

Dipartimento di Matematica e Informatica, Università di Udine
via delle Scienze 206, 33100 Udine, Italy.
{digianantonio,honsell,lenisa}@dimi.uniud.it

Abstract. We experiment Leifer-Milner *RPO approach* to CCS and to π-calculus. The basic category in which we carry out the construction is the category of term contexts. Several issues and problems emerge from this experiment; for them we propose some original solutions.

Introduction

Recently, much attention has been devoted to derive *labelled transition systems* and *bisimilarity congruences* from *reaction systems*, in the context of process languages and graph rewriting, [6,7,4,5,1,2]. In the theory of process algebras, the operational semantics of CCS was originally given via a labelled transition system (LTS), while more recent process calculi have been presented via reaction systems plus structural rules. Reactive systems naturally induce behavioral 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, or else it is an heavy, ad-hoc task to prove that they are congruences.

Leifer and Milner [6] presented a general categorical method, based on the notion of Relative Pushout (RPO), for deriving a transition system from a reaction 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. Some difficulties have arisen in applying the approach directly to such languages, viewed as Lawvere theories, because of structural rules. Thus more complex categorical constructions have been introduced by Sassone and Sobocinski in [7].

In this work, we apply the RPO technique to the prototypical examples of CCS and pi-calculus.

Aims and basic choices are the following.
i) To consider simple and quite fundamental case studies in which to experiment the RPO approach.
ii) To apply the RPO approach in the category of term contexts. In this category, arrows represent syntactic terms or contexts. The use of a category so strictly

related to the original syntax has the advantage that the generated LTS has a quite direct and intuitive interpretation.

In carrying out the simpler case study given by CCS, we have found the following problems. For all of them we propose some original solutions.
– *Structural rules.* In [7], Sassone and Sobocinski proposed the use of G-categories to deal with reduction systems like CCS, where, beside the reduction rules, there is a series of structural rules. However, so far G-categories have been used to treat just tiny fragments of CCS, while in other treatments of CCS [1], structural rules are avoided through a graph encoding; namely there is a single graph representation for each class of structurally equivalent terms. In this work, we show how, using a suitably defined G-category, one can directly apply the RPO approach to the full CCS calculus.
– *Names.* Another issue is given by names, and name biding. In this work we propose *de Brujin indexes* as a suitable instrument to deal with the issues that name manipulation poses. We found out that de Brujin indexes can be suitably used also for π-calculus, where name manipulation is more sophisticated than in CCS.
– *Pruning the LTS.* The simple application of the RPO approach generates LTS's that are quite redundant, in the sense that most of the transitions can be eliminated from the LTS without affecting the induced bisimilarity. From a practical point of view, having such large trees makes the proofs of bisimilarity unnecessarily complex. In this work, we propose a general technique that can be used in order to identify sets of transitions that can be eliminated from the LTS, without modifying the induced bisimilarity. In detail, we introduce a notion of *definability* of a transition in terms of a set of other transitions T. We prove that, given a LTS constructed via the RPO technique, if the class T of transitions is such that any other transition in the original LTS is definable from T, then the restricted LTS, obtained by considering only transitions in T, induces the same bisimilarity of the original LTS.

The result of the above technique is a LTS for CCS that coincides with the original LTS proposed by Milner. The above construction, applied to the more sophisticated case of the π-calculus, gives us a notion of bisimilarity which turns out to coincide with the *syntactical bisimilarity* of [4], and it is strictly included in Sangiorgi's *open bisimilarity.* In the π-calculus case, the LTS that we obtain by our general construction, although quite reduced, is not directly finitely branching. However, it can be turned into a finite one, by working in the setting of *categories of second-order term contexts* of [3], where *parametric rules* can be represented.

Summary. In Section 1, we present CCS syntax and reaction semantics with de Brujin indexes. In Section 2, we summarize the theory of G-categories, and Leifer-Milner theory of reaction systems in a G-category setting. In Section 3, we present a construction allowing to prune the LTS obtained by applying the previous theory of reaction systems. In Sections 4 and 5, we study LTS's and bisimilarities obtained by applying the above general constructions to CCS, and

π-calculus, respectively. Final remarks and directions for future work appear in Section 6.

1 CCS Processes with de Brujin Indexes

In this section, we present a version of Milner's CCS with *de Brujin indexes*, together with the reaction system. Such presentation allows us to deal smoothly with binding operators, and it is needed for extending in a natural way the structural congruence on processes to contexts.

In our presentation, CCS *names* $a_0, a_1, \ldots$ are replaced by de Brujin indexes $r_0, r_1, \ldots$, which are *name references*. The intuition about indexes is that

- 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 ν's;
- otherwise, if $i < n$, then r_i is bound by the $i+1$-th ν on its left;
- binding operators ν do not contain any name.

E.g. in $\nu\nu\overline{r}_0.r_2.0$, $\overline{r}_0$ is bound by the internal ν, while r_2 refers to the free name a_0. Formally:

Definition 1 (CCS Processes). *Let $r_0, r_1, \ldots \in \mathcal{NR}$ be a set of* name references*, let $\alpha \in \mathcal{A}ct = \{r_i, \overline{r}_i | r_i \in \mathcal{N}\} \cup \{\tau\}$ be a set of* actions*, and let $x, y, z, \ldots \in \mathcal{X}$ be a set of process variables, then we define*

$$\begin{array}{rll} (\mathcal{G} \ni)\ M ::= 0 \ \mid\ \alpha.P \mid M_1 + M_2 \mid \alpha.x & \text{guarded processes} \\ (\mathcal{P} \ni)\ P ::= M \mid \nu P \mid P_1|P_2 \mid rec\ x.P \mid \varphi P & \text{processes} \end{array}$$

where φ is a (injective) index transformation*, obtained as a finite composition of the transformations $\{\delta_i\}_{i\geq 0} \cup \{s_i\}_{i \geq 0}$, where δ_i, s_i represent the i-th* shifting *and the i-th* swapping*, respectively, defined by*

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

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

The index transformations φ in Definition 1 above are needed for dealing with α-rule explicitly.

In order to apply the GRPO technique to CCS, 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 Brujin plays an important rôle. Namely the CCS rule

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

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 Brujin indexes (and index transformations), we above rule can be naturally extended to contexts as follows:

$$C[\,]|(\nu C'[\,]) \equiv \nu((\delta_0 C[\,])|C'[\,])$$

The complete definition of structural congruence is as follows:

Definition 2 (Structural Congruence). *Let $C[\,], C'[\,], C''[\,]$ denote 0-holed or 1-holed contexts. The* structural congruence *is the relation $\equiv$, closed under process constructors, inductively generated by the following set of axioms:*

(**par**)	$C[\,]\|0 \equiv C[\,] \quad C[\,]\|C'[\,] = C'[\,]\|C[\,]$ $C[\,]\|(C'[\,]\|C''[\,]) \equiv (C[\,]\|C'[\,])\|C''[\,]$
(**plus**)	$C[\,] + 0 = C[\,] \quad C[\,] + C'[\,] \equiv C'[\,] + C[\,]$ $C[\,] + (C'[\,] + C''[\,]) = (C[\,] + C'[\,]) + C''[\,]$
(**rec**)	$rec\ x.C[\,] \equiv C[\,][rec\ x.C[\,]/x]$
(**nu**)	$\nu 0 \equiv 0 \quad C[\,]\|(\nu C'[\,]) \equiv \nu((\delta_0 C[\,])\|C'[\,]) \quad \nu\nu C[\,] \equiv \nu\nu \mathbf{s}_0 C[\,]$
(**phi**)	$\varphi 0 \equiv 0 \quad \varphi(\alpha.C[\,]) \equiv \varphi(\alpha).\varphi(C[\,])$ $\varphi(C[\,]\|C'[\,]) \equiv \varphi(C[\,])\|\varphi(C'[\,]) \quad \varphi(rec\ x.C[\,]) \equiv rec\ x.(\varphi C[\,])$ $\varphi(C[\,] + C'[\,]) \equiv \varphi(C[\,]) + \varphi(C'[\,]) \quad \varphi(\nu C[\,]) \equiv \nu(\varphi_{+1} C[\,])$ $\varphi_1 \ldots \varphi_m[\,] \equiv \varphi'_1 \ldots \varphi'_n[\,]\ ,\quad if\ \ \varphi_1 \circ \ldots \circ \varphi_m = \varphi'_1 \circ \ldots \circ \varphi'_n$

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

The last (**phi**)-rule in the above definition is useful for dealing with structural congruence of contexts (but of course is not necessary when dealing only with processes). Notice that there is an effective procedure to determine whether $\varphi_1 \circ \ldots \circ \varphi_m = \varphi'_1 \circ \ldots \circ \varphi'_n$. Namely, the two compositions are equal if and only if they contain the same number of transformations in the forms δ_i and their behavior coincides on an initial segment of indexes (whose length can be calculated from the δ_i's and the s_i's involved.)

As in the standard presentation, one can easily show that each CCS process is structurally congruent to a process in *normal form*, *i.e.* a process of the shape $\nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$, where all unguarded restrictions are at the top level, and index transformations do not appear at the top level. A similar normal form can be defined also for contexts. Reaction semantics, defined up-to structural congruence, is as follows:

Definition 3 (Reaction Semantics). *The* reaction relation $\rightarrow$ *is the least relation (on closed processes) closed under the following* reaction rules *and* reaction contexts*:*

Reaction rules. $\quad r.P + M \mid \overline{r}.Q + N \rightarrow P|Q \qquad \tau.P + M \rightarrow P$

Reaction contexts. $\quad D[\,] ::= [\,] \mid \nu D[\,] \mid P|D[\,] \mid D[\,]|P$

Of course, one can easily define a mapping from standard CCS syntax into our de Brujin presentation, in such a way that reaction semantics is preserved. We omit the details.

2 The Theory of Reactive Systems in the G-category Setting

In this section, we summarize the categorical notions necessary in the remaining of the article. These are the theories of G-categories, and the reaction systems formulated in a G-category setting [6,8].

The basic idea is to formulate the notion of *reaction system*, in a setting whereby *contexts* are modeled as arrows of a category, *terms* are arrows having as source a special object 0, and reaction rules are pairs of terms.

For our purpose it is necessary to consider a more involved formulation of the theory where G-categories are used. G-categories are a particular form of 2-categories where morphisms between arrows are all isomorphisms. G-categories are useful in dealing with calculi where there are structural equivalence relations on terms, CCS and π-calculus are typical examples. For these calculi, two cells isomorphisms represent equivalence relations on contexts. The extra complexity of using G-categories is motivated by the fact that the simpler approach of using categories with arrows representing equivalence classes of contexts (or terms) induces an incorrect bisimilarity, [8] .

Definition 4. *A 2-category $\mathcal{C}$ consists of*

- *A set of objects: X, Y, Z, ...*
- *For any pair of objects $X, Y \in \mathcal{C}$, a category $C(X,Y)$. Objects in $C(X,Y)$ are called* 1-cells morphisms*, and denoted by $f : X \to Y$. Arrows in $C(X,Y)$ are called* 2-cells isomorphisms *and represented by $\alpha : f \Rightarrow g$ or by* $\begin{tikzcd} X \arrow[r, bend left, "f"{name=U}] \arrow[r, bend right, "g"'{name=D}] & Y \arrow[from=U, to=D, Rightarrow, "\alpha"] \end{tikzcd}$.
  Composition in $\mathcal{C}(X,Y)$ is called vertical composition *and it is denoted by* $\bullet$.
- *For all objects X, Y and Z, there is a functor $\circ : \mathcal{C}(Y,Z) \times \mathcal{C}(X,Y) \to \mathcal{C}(X,Y)$, called* horizontal composition*, which is associative and admits the identity 2-cells of id_X as identities.*

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

We define here the G-category formed by the *finite* (*i.e.* without the *rec* operator) CCS terms and contexts, with terms (and contexts) equipped with a structural equivalence. Since the CCS grammar needs to distinguish between guarded and generic terms, the category needs to contain two distinct objects. Formally:

- Objects are $0, \mathcal{G}, \mathcal{P}$.

– Arrows from 0 to $\mathcal{G}$ ($\mathcal{P}$) are guarded processes (generic processes). Arrows from $\mathcal{G}$ ($\mathcal{P}$) are contexts that take a guarded term (a term). More formally, the arrows $\mathcal{A} \to \mathcal{B}$ are the contexts $C^{\mathcal{B}}_{\mathcal{A}}[\,]$ generated by the grammar:
$C^{\mathcal{G}}_{\mathcal{G}}[\,] ::= [\,] \mid \alpha.C^{\mathcal{P}}_{\mathcal{G}}[\,] \mid C^{\mathcal{G}}_{\mathcal{G}}[\,] + M \mid M + C^{\mathcal{G}}_{\mathcal{G}}[\,]$
$C^{\mathcal{G}}_{\mathcal{P}}[\,] ::= \quad \alpha.C^{\mathcal{P}}_{\mathcal{P}}[\,] \mid C^{\mathcal{G}}_{\mathcal{P}}[\,] + M \mid M + C^{\mathcal{G}}_{\mathcal{P}}[\,]$
$C^{\mathcal{P}}_{\mathcal{G}}[\,] ::= \quad C^{\mathcal{G}}_{\mathcal{G}}[\,] \mid \nu C^{\mathcal{P}}_{\mathcal{G}}[\,] \mid C^{\mathcal{P}}_{\mathcal{G}}[\,]|P \mid P|C^{\mathcal{P}}_{\mathcal{G}}[\,] \mid \delta C^{\mathcal{P}}_{\mathcal{G}}[\,]$
$C^{\mathcal{P}}_{\mathcal{P}}[\,] ::= [\,] \mid C^{\mathcal{G}}_{\mathcal{P}}[\,] \mid \nu C^{\mathcal{P}}_{\mathcal{P}}[\,] \mid C^{\mathcal{P}}_{\mathcal{P}}[\,]|P \mid P|C^{\mathcal{P}}_{\mathcal{P}}[\,] \mid \delta C^{\mathcal{P}}_{\mathcal{P}}[\,]$
For simplicity, in what follows we will omit the tag $\mathcal{P}, \mathcal{G}$ from the contexts.
– 2-cell isomorphisms between $C[\,]$ and $C'[\,]$ are the one-to-one maps between the instances of actions in $C[\,]$ and $C'[\,]$ induced by the proof of structural congruence. By structural induction on the proof of structural congruence, it is possible to show that two structurally congruent finite terms have the same number of instances for each action, and each proof of congruence induces a one to one maps between actions in an obvious way.

Here we restrict the G-category to contain only finite processes because we need the 2-cell morphisms to be isomorphisms. When CCS processes contain the *rec* operator, two congruent processes can contain two different numbers of actions, so there cannot exists a one-to-one map between the sets of actions. However, it is possible to recover a LTS for the whole CCS processes by defining the LTS associated to an infinite process P (a terms containing *rec*) as the supremum of the LTS associated to the approximants of P. For lack of space we omit the details.

Definition 5 (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 $d' \in \mathcal{D}$.

In our leading example a G-reaction system for CCS is obtained by taking as reaction rules and reaction contexts the ones given in Definition 3. It is immediate to check that this definition is correct.

A G-reaction system induces a *reaction relation* $\to$ on 1-cells, defined by: $t \to u$ if there exist $\langle l, r \rangle \in \mathbf{R}$, $\alpha : dl \Rightarrow t$ and $\beta : u \Rightarrow dr$. Observe that the reaction relation is closed by 2-cell isomorphisms. In the CCS example, the above reaction relation coincides with the canonical one given in Definition 3

The behavior of a reaction system is expressed as an unlabelled transition system. On the other hand, many useful behavioral equivalences are only defined for lts's.

From a reaction systems it is possible to derive a lts by taking as labels the contexts that transform a term into a term for which a reduction rule applies. In [6], the authors formalize these ideas and propose to take as labels the "smallest contexts allowing for a reaction". A categorical criterion for identifying the smallest contexts is given by the *relative pushouts* construction. In [8] this categorical construction is extended to G-categories.

Definition 6 (GRPO/GIPO).

(i) Let $\mathcal{C}$ be a G-category and let us consider the commutative diagram in Fig. 2(i). Any tuple $\langle I_5, e, f, g, \beta, \gamma, \delta \rangle$ which makes diagram in Fig. 2(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. 2(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 2(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.*

$$\begin{tikzcd}
& I_4 & \\
I_2 \arrow[ur, "c"] & \overset{\alpha}{\Longrightarrow} & I_3 \arrow[ul, "d"'] \\
& 0 \arrow[ul, "t"] \arrow[ur, "l"'] &
\end{tikzcd}$$

(i)

$$\begin{tikzcd}
& I_4 & \\
I_2 \arrow[ur, "c"] \arrow[r, "e"] \arrow[ur, phantom, "\gamma", shift left=3] & I_5 \arrow[u, "g"] \arrow[d, phantom, "\overset{\beta}{\Rightarrow}"] & I_3 \arrow[ul, "d"'] \arrow[l, "f"'] \arrow[ul, phantom, "\delta", shift right=3] \\
& 0 \arrow[ul, "t"] \arrow[ur, "l"'] &
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
& & I_4 & & \\
& & I_6 \arrow[u, "g'"] & & \\
I_2 \arrow[uurr, "c"] \arrow[urr, "e'"] \arrow[rr, "e"] \arrow[urr, phantom, "\gamma'", shift left=4] \arrow[urr, phantom, "\varphi", shift right=4] & & I_5 \arrow[u, "h"] & & I_3 \arrow[uull, "d"'] \arrow[ull, "f'"'] \arrow[ll, "f"'] \arrow[ull, phantom, "\delta'", shift right=4] \arrow[ull, phantom, "\psi", shift left=4]
\end{tikzcd}$$

(iii)

Fig. 1. Redex Square and Relative Pushout.

Definition 7 (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 $d \in \mathcal{D}$, $ct = dl$, $\langle l, r \rangle \in$ **R** and the diagram in Fig. 2(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, [8]. It follows that the transition relation is independent from the chosen representative of an equivalence class. Let $\sim_I$ denote the bisimilarity induced by the GIPO lts.

Another important property is the pasting property for GIPO squares.

Lemma 1 (GIPO pasting, [8]). *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.

Definition 8 (Redex GRPO). *Let* **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. 2(i), with* l *the left-hand side of a reaction rule and* d *a reaction context.*
A G-reaction system **C** *is said to* have redex GRPOs *if every redex square has a GRPO.*

$$\begin{tikzcd}
{} \arrow[rr, "f_0"] \arrow[dd, "e_0"'] & & {} \arrow[d, "f_1"] \\
& \overset{g_1\alpha\bullet\sigma f_0}{\Nearrow} & {} \arrow[d, "e_2"] \\
{} \arrow[r, "g_0"'] & {} \arrow[r, "g_1"'] & {}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{} \arrow[r, "f_0"] \arrow[d, "e_0"'] \arrow[dr, phantom, "\overset{\alpha}{\Nearrow}"] & {} \arrow[r, "f_1"] \arrow[d, "e_1"] \arrow[dr, phantom, "\overset{\sigma}{\Nearrow}"] & {} \arrow[d, "e_2"] \\
{} \arrow[r, "g_0"'] & {} \arrow[r, "g_1"'] & {}
\end{tikzcd}$$

(i) (ii)

Fig. 2. IPO pasting.

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

Theorem 1. *Let* **C** *be a G-reaction system having redex GRPOs. Then the GIPO bisimilarity* $\sim_I$ *is a congruence w.r.t. all contexts, i.e. if* $a{\sim_I}b$ *then for all* c *of the appropriate type,* $ca{\sim_I}cb$*.*

3 Pruning the GIPO LTS

In this section we present a construction 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.* We will prove that 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. Intuitively, a transition in a LTS is definable if can be replicated using other transitions (and contexts).

Definition 9. *Given a G-reaction system $\mathbb{C}$, having redex GRPOs, let T be a subset of the whole set of GIPO transitions,*

(i) we say that T is closed under bisimulation if for any $[t_1], [t'_1], [t_2], [t'_2]$, $[f]$, such that $[t_1]{\sim_I}[t'_1]$, $[t_2]{\sim_I}[t'_2]$, $[t_1]\xrightarrow{[f]}_I[t_2]$, $[t'_1]\xrightarrow{[f]}_I[t'_2]$, we have that: $[t_1]\xrightarrow{[f]}_I[t_2] \in T$ iff $[t'_1]\xrightarrow{[f]}_I[t'_2] \in T$

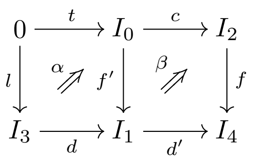
(ii) we say that the whole GIPO LTS is definable *from T if there exists a set of tuples $\{\ \langle [f_k], [f'_k], P_k, e_k\rangle | k \in K\}$ of the following form:*
- *$[f_k]$ GIPO label, $[f'_k]$ GIPO label or $f'_k = \epsilon$ with $f_k : I_k \to I'_k$, $f'_k : I_k \to J_k$ (where we set $\epsilon : I_k \to I_k$)*
- *P_k is a Hennessy-Milner proposition with modal operators labeled by GIPO labels*
- *$e_k : J_k \to I_k$ (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 i and $t'' : 0 \to J_k$ satisfying the next four conditions:
- *$[f] = [f_k]$,*
- *$([t]\xrightarrow{[f'_k]}_I[t''] \in T)$ or $(t'' = t \wedge f'_k = \epsilon)$*
- *In the T LTS, the state $[t'']$ satisfies the proposition P_k*
- *$([t'] = [e_k(t'')] \wedge J_k \neq 0)$ or $([t'] = [e_k] \wedge J_k = 0)$*

Proposition 1. *Given a reactive system $\mathbb{C}$, and a subset T of transition that is closed under GIPO bisimulation and such that the whole GIPO LTS is definable from T, then $\sim_I = \sim_T$, i.e. the two GIPO LTS induce the same bisimilarity.*

Proof. Consider the relation $S = \{\langle [ct], [cu]\rangle \mid [t] \sim_T [u],\ c \text{ context}\}$. It is easy to prove that $\sim_I \subseteq \sim_T \subseteq S$. If we prove that S is an GIPO bisimilarity (ie $S \subseteq \sim_I$), then the three relations are equal. So we prove that, for any $\langle [ct], [cu]\rangle \in S$, if $[ct]\xrightarrow{[f]}_I[t']$, then there exists u' s.t. $[cu]\xrightarrow{[f]}_I[u']$ with $[t']S[u']$.

Consider the following diagram:


$$\begin{tikzcd}
0 \arrow[r, "t"] \arrow[d, "l"'] & I_0 \arrow[r, "c"] \arrow[d, "f'"'] & I_2 \arrow[d, "f"] \\
I_3 \arrow[r, "d"'] \arrow[ur, phantom, "\overset{\alpha}{\nearrow}"] & I_1 \arrow[r, "d'"'] \arrow[ur, phantom, "\overset{\beta}{\nearrow}"] & I_4
\end{tikzcd}$$

where the outer rectangle is the GIPO inducing the transition $[ct]\xrightarrow{[f]}_I[t']$, namely $[t'] = [d'dr]$ with $\langle l, r\rangle$ reaction rule, and the left square is obtained from an GIPO construction starting from l and t. There are two cases to consider:

- If the transition labeled by $[f']$ is in T, then, since $[t]\xrightarrow{[f']}_I[dr]$, there exists u'', $[u]\xrightarrow{[f']}_I[u'']$, $[u''] \sim_T [dr]$. By composition of RPO squares, $[cu]\xrightarrow{[f]}_I[d'u'']$, from which the thesis.

– If the transition labeled by $[f']$ is not in T, then it is definable by T, and since $[t]\xrightarrow{[f']}_I[dr]$, there exists a tuple $\langle[f'],[f_k],P_k,e_k\rangle$ and a term t'' such that $[t]\xrightarrow{[f_k]}_I[t'']$, $P_k([t''])$, and $[dr]=[e_kt'']$ (or $[dr]=[e_k]$). From the last equality it follows $[t']=[d'dr]=[d'e_kt'']$ $(=[d'e_k])$. Since $[t]\sim_T[u]$, there exists u'', $[u]\xrightarrow{[f_k]}_I[u'']$, $[t'']\sim_T[u'']$ and so $P_k([u''])$ (Hennessy-Milner propositions cannot separate bisimilar elements). From this, $[u]\xrightarrow{[f']}_I[e_ku'']$ $(\xrightarrow{[f']}_I=[e_k])$. By composition of GRPO squares, $[cu]\xrightarrow{[f]}_I[d'e_ku'']$ $(\xrightarrow{[f]}_I[d'e_k])$, from which the thesis.

□

In using the above proposition for the CCS and π-calculus cases, we are not going to use the extra expressivity given by the Hennessy-Milner propositions P_k; in all the tuples $\langle[f_k],[f'_k],P_k,e_k\rangle$ defined in the following, the propositions P_k will be set equal to *true*. Nevertheless, we prefer here to present this general version of the proposition.

4 Applying the GRPO Technique to CCS

In this section, we study LTS's obtained by applying the GRPO technique to CCS. First, we consider the LTS obtained by applying Leifer-Milner theorem in the GRPO setting (Theorem 1 of Section 2). This turns out to be still infinitely branching. However, by applying our general pruning technique of Section 3, we are able to get a *finitely branching LTS* and *GIPO bisimilarity*, which coincide with the original Milner's LTS and strong bisimilarity, respectively.

The basic property that allows to apply the GRPO construction is the following.

Proposition 2. *The G-reaction system of finite CCS processes has redex GRPO.*

Proof. There are several cases to consider depending also on the reaction rule involved, here we consider only the reaction rule $r.P+M\mid\overline{r}.Q+N\rightarrow P|Q$. Given the commuting redex square $\alpha:C[\,]\circ P\Rightarrow D[\,]\circ L$, by structural rules, the reaction contexts $D[\,]$ can be written in the form $\nu^n([]|P_1|\dots P_h)$, while the context $C[\,]$ can be written as $\nu^mC'[\varphi[\,]]$, with $C'[\,]$ not containing any name transformation φ', or hiding operator ν.

If the redex L is all contained (or better mapped by α^{-1}) in the process P, the GRPO has form $\alpha':\varphi[\,]\circ P\Rightarrow(\nu^m[\,]|P_{i_1}|\dots|P_{i_k})\circ L$. Notice that the transformation $\varphi[\,]$, cannot be factorized by the GRPO construction.

If the process P contains only one side of the redex L, the GRPO has form $\alpha':(\varphi[\,]|P')\circ P\Rightarrow(\nu^m[\,]|P_{i_1}|\dots|P_{i_k})\circ L$, with the process P' giving the other side of the redex.

If the redex L is all contained in the context $C[\,]$, the GRPO has form $\alpha':C''[\varphi[\,]]\circ P\Rightarrow(\nu^m[\,]|P')\circ L$, with φP contained in P'.

In the remaining cases, where the main connectives of the redex L are contained in $C'[\,]$, the GRPO has the form $\alpha':C''[\varphi[\,]]\circ P\Rightarrow(\nu^m[\,])\circ L$ □

Table 1 summarizes the set of GIPO contexts (up-to structural congruence) obtained by applying Theorem 1. For simplicity, we denote an equivalence class $[C[\,]]$ by a special representative.

Process $P \equiv \nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$	**GIPO contexts**
$\exists i,j.\ \alpha_{ij}=\tau \ \vee\ (\exists i,j,i',j'.\ \alpha_{ij}=r_l \ \wedge\ \alpha_{i'j'}=\overline{r}_l)$	$\varphi[\,]$
$\exists i,j.\ \alpha_{ij} \neq \tau$	$\varphi[\,]\ +\ M \mid \alpha.Q+N$ with $\delta_0^k(\alpha)=\overline{\varphi_{+k}(\alpha_{ij})}$
	$C[\,] \mid \alpha.Q+M \mid \overline{\alpha}.R+N$ $C[\,]\ +\ \alpha.Q \mid \overline{\alpha}.R+N$ $C[\,]\ +\ \tau.Q$ $C[\,] \mid \tau.Q+N$ $\alpha.C[\,]+M \mid \overline{\alpha}.Q+N$ $\tau.C[\,]+N$

Table 1. CCS GIPO contexts.

The first item in Table 1 corresponds to the case where an internal transition of the process P is considered. In such case the GIPO context is empty. If the process P exposes a non-τ action, then a communication with a context exposing a complementary action can arise (second item). Finally, the last raw shows all GIPO contexts where the reduction is "all inside the context" (and the process plays a passive rôle).

Notice that if we do not use G-categories and work on the category where arrows are equivalence classes of CCS processes, the process $\alpha.0 \mid \overline{\alpha}.0$ will have as only GIPO contexts the contexts $\varphi[\,]$ and the GIPO contexts $\varphi[\,]\ +\ M \mid \alpha.Q+N$ will be missing.

The GIPO LTS described above is still infinitely branching. However, there are many GIPO contexts which are intuitively redundant, *e.g.* all the contexts in the last raw. These are "not engaged", *i.e.* the reduction is all inside the context. Also the class of contexts in the third raw is redundant; namely, the contexts of the shape $[\,]|\alpha.Q$ are sufficient to define the whole class, in the sense of Definition 9 of Section 3. More precisely, we have:

Proposition 3.
i) GIPO LTS is definable from set of GIPO transitions labeled by $\{[\,]\}\cup\{[\,]|\alpha.0 \mid \alpha \in \mathcal{A}\}$.
ii) The bisimilarity induced by the LTS defined by such GIPO contexts (see Table 2) is a congruence.

Proof. i) Transitions corresponding to GIPO contexts of the shape $\varphi[\,]$ (second raw in Table 1) are definable via the tuple $\langle \varphi[\,], [\,], \mathit{true}, \varphi[\,] \rangle$. Transitions corresponding to GIPO contexts of the shape $\varphi[\,]+M \mid \alpha.Q\ +\ N$ (third raw in Table 1) are definable via the tuple $\langle \varphi[\,]+M \mid \alpha.Q\ +\ N, [\,]|\alpha'.0, \mathit{true}, \varphi[\,]|Q \rangle$, where $\delta_0^k(\alpha') = \overline{\alpha}_{ij}$. Transitions corresponding to the contexts $C[\,]$ in the last raw of Table 1 are definable via tuples of the shape $\langle C[\,], \epsilon, true, E[\,]\rangle$, where $E[\,]$ is a 0 or 1-holed context defined according to the following table:

GIPO context	**E**$[\,]$
$C[\,] \mid \alpha.Q + M \mid \overline{\alpha}.R + N$	$C[\,] \mid Q \mid R$
$C[\,] \;+\; \alpha.Q \mid \overline{\alpha}.R + N$	$Q \mid R$
$C[\,] \;+\; \tau.Q$	Q
$C[\,] \mid \tau.Q + N$	$C[\,] \mid Q$
$\alpha.C[\,] + M \mid \overline{\alpha}.Q + N$	$C[\,] \mid Q$
$\tau.C[\,] + N$	$C[\,]$

ii) The proof follows from Proposition 1. □

Process $P \equiv \nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$	**GIPO contexts**
$\exists i, j.\ \alpha_{ij} = \tau \ \vee\ (\exists i, j, i', j'.\ \alpha_{ij} = r_l \ \wedge\ \alpha_{i'j'} = \overline{r}_l)$	$[\,]$
$\exists i, j.\ \alpha_{ij} \neq \tau$	$[\,] \mid \overline{\alpha}_{ij}.0$

Table 2. CCS reduced GIPO contexts.

Now, it is immediate to see that the above GIPO LTS coincides with the standard LTS; namely the GIPO context $[\,]$ corresponds to the τ-transition, while the GIPO context $[\,] \mid \alpha.0$ corresponds to a $\overline{\alpha}$-transition.

Summarizing, we have:

Proposition 4. *The reduced GIPO LTS coincides with the original LTS for CCS, and the GIPO bisimilarity coincides with CCS strong bisimilarity.*

5 The π-calculus Case

In this section, we apply the above machinery to π-calculus. The latter is significantly more difficult to deal with than CCS, because of name substitutions, which arise in the reaction semantics. We will show that the reduced GIPO LTS for π-calculus induces the *syntactical bisimilarity* of [4], which is finer than Sangiorgi's *open bisimilarity.* Our pruning technique does not give us directly a finitely branching LTS, however we will briefly discuss how a finitary GIPO LTS can be obtained by working in the setting of *categories of second-order term contexts* of [3].

We start by introducing the π-calculus syntax with de Brujin indexes.

Definition 10 (π-calculus Processes). *Let $r_0, r_1, \ldots, s_0, s_1, \ldots \in \mathcal{NR}$ be a set of* name references*, and let $x, y, z, \ldots \in \mathcal{X}$ be a set of process variables. We define*

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

where σ is a substitution obtained as a finite composition of shifting operators δ_i's, swapping operators s_i's, and singleton substitutions t_{ij}, defined by:

$$t_{ij}(r_k) = \begin{cases} r_k & \textit{if } k \neq i \\ r_j & \textit{if } k = i \end{cases}$$

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

We denote by $dom(\sigma)$ the set of name references on which σ is not the identity, *i.e.* $\{r_i \mid \sigma(r_i) \neq r_i\}$. π-calculus contexts are defined similarly to CCS contexts. The structural congruence extended to contexts is defined as follows:

Definition 11 (Structural Congruence). *Let $C[\,], C'[\,], C''[\,]$ denote 0-holed or 1-holed contexts. The* structural congruence *is the relation $\equiv$, closed under process constructors, inductively generated by the following set of axioms:*

$$\begin{array}{ll}
(\textbf{par}) & C[\,]|0 \equiv C[\,] \quad C[\,]|C'[\,] = C'[\,]|C[\,] \\
& C[\,]|(C'[\,]|C''[\,]) \equiv (C[\,]|C'[\,])|C''[\,] \\
(\textbf{plus}) & C[\,] + 0 = C[\,] \quad C[\,] + C'[\,] \equiv C'[\,] + C[\,] \\
& C[\,] + (C'[\,] + C''[\,]) = (C[\,] + C'[\,]) + C''[\,] \\
(\textbf{rec}) & rec\ x.C[\,] \equiv C[\,][rec\ x.C[\,]/x] \\
(\textbf{nu}) & \nu 0 \equiv 0 \quad C[\,]|(\nu C'[\,]) \equiv \nu((\delta_0 C[\,])|C'[\,]) \quad \nu\nu C[\,] \equiv \nu\nu \mathbf{s}_0 C[\,] \\
(\textbf{sigma}) & \sigma 0 \equiv 0 \quad \sigma(\overline{r}s.C[\,]) \equiv \overline{\sigma(r)}\sigma(s).\sigma(C[\,]) \\
& \sigma(\tau.C[\,]) \equiv \tau.\sigma(C[\,]) \quad \sigma(r().C[\,]) \equiv \sigma(r)().\sigma_{+1}C[\,] \\
& \sigma(C[\,]|C'[\,]) \equiv \sigma(C[\,])|\sigma(C'[\,]) \quad \sigma(rec\ x.C[\,]) \equiv rec\ x.(\sigma C[\,]) \\
& \sigma(C[\,] + C'[\,]) \equiv \sigma(C[\,]) + \sigma(C'[\,]) \quad \sigma(\nu C[\,]) \equiv \nu(\sigma_{+1}C[\,]) \\
& \sigma_1 \ldots \sigma_m[\,] \equiv \sigma'_1 \ldots \sigma'_n[\,]\ , \quad \textit{if}\ \ \sigma_1 \circ \ldots \circ \sigma_m = \sigma'_1 \circ \ldots \circ \sigma'_n
\end{array}$$

Notice that, similarly to the CCS case, the last (**sigma**)-rule is effective, by definition of the substitutions σ_i, σ'_i.

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 $\nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$, where all unguarded restrictions are at the top level, and substitutions do not appear at the top level.

Definition 12 (Reaction Semantics). *The* reaction relation $\rightarrow$ *is the least relation closed under the following* reaction rules *and* reaction contexts*:*

Reaction rules. $\quad r().P + M \mid \overline{r}r_j.Q + N \rightarrow (\nu(t_{0j+1}P))|Q \qquad \tau.P + M \rightarrow P$

Reaction contexts. $\quad D[\,] ::= [\,] \mid \nu D[\,] \mid P|D[\,] \mid D[\,]|P$

The above rule for communication may seem strange, but one can easily check that it is equivalent to the original one. It is motivated by the fact that, by using a ν operator in the resulting process, we avoid the introduction of operators for index decrementing, which would be problematic for the GRPO construction.

Table 3 summarizes the set of GIPO contexts (up-to structural congruence) obtained by applying Theorem 1. Table 4 summarizes the set of reduced GIPO contexts, which define all the GIPO contexts, according to Definition 9 of Section 3.

Notice that, when the process exposes an output action and the context an input one, *i.e.* $C[\,] = \sigma[\,] + M \mid r'().Q{+}N$ (fifth raw in Table 3), we cannot get rid of Q in the reduced context (last raw of Table 4). This is because the transition provides a substitution for Q, depending on the process P (and hence the context $e()$ required in Definition 9 would not be uniform on all processes). Moreover, if σ acts also on $fr(\nu Q)$, then we cannot get rid of it, since otherwise it would appear in the context $e()$ and it would act also on names in Q, which we do not want. Therefore, the reduced GIPO LTS that we obtain, although significantly simpler than the original one, is still infinitely branching, since a process P, which exposes an output action, makes infinitely many transitions $P \xrightarrow{\sigma[\ ]|r'().Q}$, for any Q. In Section 5.1, we will sketch how to overcome this problem getting a finitely branching characterization of the GIPO LTS and bisimilarity.

GIPOprocess $P \equiv \nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$	**GIPO contexts**
$\exists i,j.\ \alpha_{ij} = \tau\ \vee$ $\exists i,j,i',j'.\ \alpha_{ij} = r_l()\ \wedge\ \alpha_{i'j'} = \overline{r}_l s$	$\sigma[\ ]$
$\exists i,j,i',j'.\ \alpha_{ij} = r_h()\ \wedge\ \alpha_{i'j'} = \overline{r}_l s\ \wedge\ h \neq l$	$\sigma[\ ]$ with $\sigma_{+k}(r_h) = \sigma_{+k}(r_l)$
$\exists i,j.\ \alpha_{ij} = r()$	$\sigma[\ ]\ +\ M \mid \overline{r}'s.Q + N$ with $\delta_0^k(r') = \overline{\sigma_{+k}(r)}$
$\exists i,j.\ \alpha_{ij} = \overline{r}s$	$\sigma[\ ]\ +\ M \mid r'().Q + N$ with $\delta_0^k(r') = \overline{\sigma_{+k}(r)}$
	$C[\ ] \mid r().Q + M \mid \overline{r}s.R + N$ $C[\ ]\ +\ r().Q \mid \overline{r}s.R + N$ $C[\ ]\ +\ \tau.Q$ $C[\ ] \mid \tau.Q + N$ $r().C[\ ] + M \mid \overline{r}s.Q + N$ $\tau.C[\ ] + N$

Table 3. π-calculus GIPO contexts.

5.1 Finitely Branching LTS's for π-calculus

First of all, notice that the substitution σ appearing in the GIPO context $\sigma[\ ]|r'().Q$ in the last raw of Table 4 is actually redundant, even if it cannot be eliminated using Proposition 1. However, by a direct reasoning, one can show that contexts of the shape $[\ ]|r'().Q$ are sufficient.

Process $P \equiv \nu^k(\Sigma_{j=1}^{m_1}\alpha_{1j}.P_{1j} \mid \ldots \mid \Sigma_{j=1}^{m_n}\alpha_{nj}.P_{nj})$	**Reduced GIPO Contexts**
$\exists i, j.\ \alpha_{ij} = \tau\ \vee$ $\exists i, j, i', j'.\ \alpha_{ij} = r_l() \ \wedge\ \alpha_{i'j'} = \overline{r}_l s$	$[\,]$
$\exists i, j, i', j'.\ \alpha_{ij} = r_h() \ \wedge\ \alpha_{i'j'} = \overline{r}_l s \ \wedge\ h \neq l$	$\sigma[\,]$ with σ singleton, $\sigma_{+k}(r_h) = \sigma_{+k}(r_l)$
$\exists i, j.\ \alpha_{ij} = r()$	$[\,] \mid \overline{r}' s.0$ with $\delta_0^k(r') = \overline{r}$
$\exists i, j.\ \alpha_{ij} = \overline{r} s$	$\sigma[\,] \mid r'().Q$ with $dom(\sigma) \subseteq fr(\nu Q)$, $\delta_0^k(r') = \overline{\sigma_{+k}(r)}$

Table 4. π-calculus reduced GIPO contexts.

Moreover, one can show that the GIPO bisimilarity that we have obtained coincides with the extension to the whole π-calculus of the *syntactical bisimilarity* introduced in [4] for the open π-calculus. In the syntactical bisimilarity one essentially observes input/output actions and name fusions allowing for a communication. The prefix and the communication rules of the LTS in [4] are represented as follows, in our setting:

$$\text{(pre)} \quad \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad\qquad \text{(com)} \quad \frac{P \xrightarrow{\overline{r} r_i} P' \quad Q \xrightarrow{r(\,)} Q'}{P \mid Q \xrightarrow{r=r'} P' \mid \nu(t_{0j+1} Q')}$$

The notion of syntactical bisimilarity is as defined by:

Definition 13 (Syntactical Bisimilarity). *A symmetric relation $\mathcal{R}$ is a* syntactical bisimulation *if whenever $P\mathcal{R}Q$ it holds that:*

- *if $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\alpha} Q'$,*
- *if $P \xrightarrow{r=r'} P'$ then $Q \xrightarrow{r=r'} Q'$ and $(\sigma P')\mathcal{R}(\sigma Q')$,*

where σ is a fusion that fuses r to r'.
The union of all syntactical bisimulations is syntactical bisimilarity*.*

Intuitively, our (reduced) GIPO LTS corresponds to the one of [4]. Notice in particular that, when P exposes an output action, in the LTS of [4] we have a transition $P \xrightarrow{\overline{r}s} P'$, where we recall both the output channel $\overline{r}$ and the object s in the label, while in our LTS we have $P \xrightarrow{[\,] \mid r(\,).Q} P' \mid \nu(\sigma Q)$, where we keep track of the object s not in the label, but in the substitution applied to Q. Summarizing, we have:

Theorem 2. *The GIPO bisimilarity on π-calculus coincides with the syntactical bisimilarity.*

As it has been observed in [4], the above bisimilarity is strictly included in Sangiorgi's open bisimilarity (where there is the extra freedom of matching a fusion transition of a process with a τ-transition of the other).

The LTS of [4] provides a finitely branching characterization of our GIPO bisimilarity. However, it is possible to get a more direct finitary characterization of the GIPO equivalence, by working in the setting of *categories of second-order term contexts*, introduced in [3]. In this setting one can represent *parametric transitions* such as $P \xrightarrow{[\,]\mid r(\,).X} P' \mid \nu(\sigma X)$, where X is a second-order variable representing a generic term, which will be possibly instantiated in the future (with the most general substitution allowing for a reaction). In this way we can avoid to have infinitely many ground transitions. We aim to present the whole construction in a further work.

6 Conclusions and Directions for Further Work

In this paper, we have refined Leifer-Milner construction for deriving LTS's from reaction systems, by studying general conditions under which we can prune the GIPO LTS. Then we have carried out two fundamental case studies in process calculi, by working in categories of term contexts. In order to deal properly with structural rules, we had to work in the setting of G-categories. For CCS the result is quite satisfactory, since we have obtained as GIPO LTS exactly Milner's original LTS together with strong bisimilarity. There are other works in the literature, where the case study of CCS has been considered, but often a graph encoding is used and furthermore the original LTS is not directly obtained from the general construction, but it is recovered only a posteriori, using an ad hoc reasoning. A similar observation applies also to π-calculus, to which the RPO approach has not been previously applied directly to its reaction semantics, but to a (often ad hoc) enriched semantics. Under this perspective, it would be interesting to develop in all details also the π-calculus case study in the second-order setting, as hinted at the end of Section 5.1.

Finally, it would be interesting to compare our work with [9], where it is presented a LTS for the π-calculus whose labels are taken to be contexts on a higher order syntax. However, that work does not apply the Leifer-Milner technique to derive the LTS, and the higher-order syntax does not coincide with the one proposed in [3] for second-order contexts. It is not clear how the proposed LTS is related to the one obtained by the GIPO technique in the second-order setting.

References

1. F. Bonchi, F. Gadducci, and B. König. Process bisimulation via a graphical encoding. In *ICGT*, volume 4178 of *LNCS*, pages 168–183. Springer, 2006.
2. F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *LICS*, pages 69–80. IEEE Computer Society, 2006.
3. P. Di Gianantonio, F. Honsell, and M. Lenisa. RPO, second-order contexts, and lambda-calculus. In R. M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 334–349. Springer, 2008. extended version available at `http://www.dimi.uniud.it/pietro/papers/`.
4. G. L. Ferrari, U. Montanari, and E. Tuosto. Model checking for nominal calculi. In V. Sassone, editor, *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2005.

5. F. Gadducci and U. Montanari. Observing reductions in nominal calculi via a graphical encoding of processes. In *Processes, Terms and Cycles*, volume 3838 of *LNCS*, pages 106–126. Springer, 2005.
6. J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *CONCUR*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.
7. V. Sassone and P. Sobocinski. Deriving bisimulation congruences using 2-categories. *Nord. J. Comput.*, 10(2):163–, 2003.
8. P. Sobocinski. *Deriving process congruences from reduction rules.* PhD thesis, University of Aarhus, 2004.
9. P. Sobocinski. A well-behaved lts for the pi-calculus: (abstract). *Electr. Notes Theor. Comput. Sci.*, 192(1):5–11, 2007.