A Petri Net Translation of π-Calculus Terms

Raymond Devillers[1], Hanna Klaudel[2], and Maciej Koutny[3]

[1] Département d'Informatique, Université Libre de Bruxelles
CP212, B-1050 Bruxelles, Belgium
rdevil@ulb.ac.be
[2] IBISC, FRE 2873 CNRS, Université d'Evry, 91000 Evry, France
klaudel@ibisc.univ-evry.fr
[3] School of Computing Science, University of Newcastle
Newcastle upon Tyne, NE1 7RU, United Kingdom
maciej.koutny@newcastle.ac.uk

Abstract. In this paper, we propose a finite structural translation of possibly recursive π-calculus terms into Petri nets. This is achieved by using high level nets together with an equivalence on markings in order to model entering into recursive calls, which do not need to be guarded.

Keywords: mobility, process algebra, π-calculus, Petri nets, compositional translation, behavioural consistency.

1 Introduction

In our previous paper [8], we devised a structural way of translating terms from the finite fragment of the π-calculus into finite Petri nets. This translation relies on a combination of PBC [1] and M-net [2,10,11] features and its result is a Petri net whose semantics in terms of a labelled transition system is strongly bisimilar [17], and often even isomorphic, to that of the original π-term.

The translation in [8] only concerned terms without recursion (and replication, but the latter is equivalent to recursion), and in this paper we will show how to remove this restriction. The standard way of incorporating recursion in a PBC-like framework is to rely on net refinement and a fixpoint approach [1]. However, in the π-calculus framework, successive refinements would generally need to apply (somewhat arbitrary) alpha-conversions in order to keep the well-formedness of process expressions, making fixpoint approach much more complicated.

An alternative could be to use recursive Petri nets [14], but this would lead to a formalism, which in our view would be difficult to lift to the high level nets and to the specific framework needed to deal with recursive π-calculus terms (in particular, it is far from clear how one could support communications between different levels of recursion). Moreover, the various kinds of causality and concurrency semantics are not currently available in the recursive Petri net theory. We therefore decided to use instead a simpler and more direct approach inspired by [7] and used in the context of PBC.

We assume that the reader is familiar with the basics concepts of π-calculus and high-level Petri nets (all formal definitions and proofs can be found in [9]).

K. Barkaoui, A. Cavalcanti, and A. Cerone (Eds.): ICTAC 2006, LNCS 4281, pp. 138–152, 2006.

2 The π-Calculus and Its Indexed Operational Semantics

We start by recalling the syntax and semantics of the π-calculus [18], assuming that $\mathbb{C}$ is a countably infinite set of *channels* ranged over by the first few lower case Roman letters; and that $\mathbb{X} = \{X_1, \ldots, X_m\}$ is a finite set of *process variables*, each variable $X \in \mathbb{X}$ having a finite arity n_X. The concrete syntax we use is given below, where P denotes an *agent* (or π-expression).

$$\ell ::= \overline{a}b \mid ac \mid \tau \qquad \text{(output/input/internal prefixes)}$$
$$P ::= \mathsf{0} \mid \ell\,\mathtt{.}\,P \mid P + P \mid P|P \mid (\boldsymbol{\nu} c)P \mid X(a_1, \ldots, a_{n_X}) \qquad \text{(agents)}$$

The constructs $ac\,\mathtt{.}\,P$ (input) and $(\boldsymbol{\nu} c)P$ (restriction) *bind* the channel c in P, and we denote by $fn(P)$ the free channels of P. For each process variable $X \in \mathbb{X}$, there is exactly one definition D_X of the form $X(a_1, \ldots, a_{n_X}) \stackrel{\text{df}}{=} P_X$, where $a_i \neq a_j$ for $i \neq j$. We assume that $fn(P_X) \subseteq \{a_1, \ldots, a_{n_X}\}$, so that the free channels of P_X are *parameter bound*. Agents are defined up to the *alpha-conversion*, meaning that bound channels may be coherently renamed avoiding potential clashes. Moreover, $\{b/c, \ldots\}P$ will denote the agent obtained from P by replacing all free occurrences of c by b, etc, possibly after alpha-converting P in order to avoid name clashes; for example $\{b/c, f/a\}ab\,\mathtt{.}\,\overline{g}b\,\mathtt{.}\,X(d, c) = fe\,\mathtt{.}\,\overline{g}e\,\mathtt{.}\,X(d, b)$.

The semantical treatment of the π-calculus adopted in this paper is that expounded by Cattani and Sewell [5], where the usual transition steps are augmented with an explicit information about unrestricted channels:

$$A \vdash P \xrightarrow{\ell} B \vdash Q$$

where ℓ is a prefix and $A, B \subset \mathbb{C}$ are finite sets of *indexing* channels such that $fn(P) \subseteq A \subseteq B \supseteq fn(Q)$. Its intuitive meaning is that

> "in a state where the channels A may be known by agent P and by its environment, P can do ℓ to become agent Q and the channels B may be known to Q and its environment".

As a result, Q may know more channels than P as an input $\ell = ab$ adds b whenever $b \notin A$ (intuitively, such a b is a new channel communicated by the outside world – see the In rule in table 1), and an output $\ell = \overline{a}b$ adds b whenever $b \notin A$ (intuitively, such a b is a channel restricted in P which becomes a new known channel in Q – see the Open rule in table 1).

The operational semantics rules for the *indexed* π-expressions are shown in table 1 (in [5], the '$B \vdash$' parts of the rules are implicit). The complete behaviour of an expression $A \vdash P$, where $fn(P) \subseteq A$, is then given by a labelled transition system derived using these rules, and denoted $\mathsf{lts}_{A \vdash P}$.

As a running example, consider an expression $\{a\} \vdash X(a) + \tau\,\mathtt{.}\,\mathsf{0}$ with X defined by $X(e) \stackrel{\text{df}}{=} ec\,\mathtt{.}\,\overline{c}e\,\mathtt{.}\,\mathsf{0} + (\nu d)(X(d)|\overline{d}e\,\mathtt{.}\,\mathsf{0})$. It admits, e.g., the following executions:

$$\{a\} \vdash X(a) + \tau\,\mathtt{.}\,\mathsf{0} \xrightarrow{\tau} \{a\} \vdash (\nu d)((\overline{a}d\,\mathtt{.}\,\mathsf{0})|\mathsf{0})$$
$$\{a\} \vdash X(a) + \tau\,\mathtt{.}\,\mathsf{0} \xrightarrow{\tau} \{a\} \vdash \mathsf{0}\,.$$

Table 1. Operational semantics of π-calculus, where: $ns(\tau) \stackrel{\text{df}}{=} \varnothing$; $ns(ab) = ns(\overline{a}b) \stackrel{\text{df}}{=} \{a, b\}$; the notation A, c stands for the disjoint union $A \uplus \{c\}$; and $(\boldsymbol{\nu} c \setminus A)P$ is P if $c \in A$ and $(\boldsymbol{\nu} c)P$ otherwise. Symmetric versions of Sum, Par and Com are omitted.

$$\textsc{Tau}\ \frac{}{A \vdash \tau\,.\,P \xrightarrow{\ \tau\ } A \vdash P} \qquad \frac{}{A \vdash ac\,.\,P \xrightarrow{\ ab\ } A \cup \{b\} \vdash \{b/c\}P}\ \textsc{In}$$

$$\textsc{Out}\ \frac{}{A \vdash \overline{a}b\,.\,P \xrightarrow{\ \overline{a}b\ } A \vdash P} \qquad \frac{A, c \vdash P \xrightarrow{\ \overline{a}c\ } A, c \vdash P' \quad a \neq c}{A \vdash (\boldsymbol{\nu} c)P \xrightarrow{\ \overline{a}c\ } A \cup \{c\} \vdash P'}\ \textsc{Open}$$

$$\textsc{Par}\ \frac{A \vdash P \xrightarrow{\ \ell\ } A' \vdash P'}{A \vdash P|Q \xrightarrow{\ \ell\ } A' \vdash P'|Q} \qquad \frac{A, c \vdash P \xrightarrow{\ \ell\ } A', c \vdash P' \quad c \notin ns(\ell)}{A \vdash (\boldsymbol{\nu} c)P \xrightarrow{\ \ell\ } A' \vdash (\boldsymbol{\nu} c)P'}\ \textsc{Res}$$

$$\textsc{Sum}\ \frac{A \vdash P \xrightarrow{\ \ell\ } A' \vdash P'}{A \vdash P + Q \xrightarrow{\ \ell\ } A' \vdash P'} \qquad \frac{A \vdash P \xrightarrow{\ \overline{a}c\ } A' \vdash P' \quad A \vdash Q \xrightarrow{\ ac\ } A'' \vdash Q'}{A \vdash P|Q \xrightarrow{\ \tau\ } A \vdash (\boldsymbol{\nu} c \setminus A)(P'|Q')}\ \textsc{Com}$$

$$\textsc{ProcDef}\ \frac{A \vdash \{b_1/a_1, \ldots, b_{n_X}/a_{n_X}\}P \xrightarrow{\ \ell\ } A' \vdash P' \quad X(a_1, \ldots, a_{n_X}) \stackrel{\text{df}}{=} P}{A \vdash X(b_1, \ldots, b_{n_X}) \xrightarrow{\ \ell\ } A' \vdash P'}$$

Given an indexed π-expression $A \vdash P$, it is always possible to apply alpha-conversions to P and the process definitions so that no channel across P, $D_{X_1}, \ldots$, D_{X_m} is both free and bound, no such channel generates more than one binding, and no restricted channel occurs in A. Such an indexed π-expression will be called *well-formed*. We fix such a well-formed $A \vdash P$ for the rest of this paper.

Context-based representation. Before translating to nets, we give a term a presentation which separates its structure from the specific channels used to express what is visible from the outside and which channels are (input or parameter) bound or restricted. This also involves separating the features related to control flow of the term from those related to channel substitution and binding. For the resulting context based representation we need two fresh countably infinite sets of *restricted channels* $\mathbb{R}$ ranged over by the upper case Greek letters, and *channel holders* $\mathbb{H}$ ranged over by the first few lower case Greek letters. A *context* itself is a partial mapping $\varsigma : \mathbb{H} \to \mathbb{C} \uplus \mathbb{R}$ with a finite domain.

The aim is to represent an expression like $\{b, d\} \vdash ba\,.\,(\boldsymbol{\nu} c)\overline{a}c\,.\,\overline{c}b\,.\,\mathtt{0}$ as a context based expression $\mathcal{P}{:}\varsigma$, where $\mathcal{P} \stackrel{\text{df}}{=} \beta\alpha\,.\,\overline{\alpha}\gamma\,.\,\overline{\gamma}\beta\,.\,\mathtt{0}$ is a *restriction-free* agent based solely on channel holders and $\varsigma \stackrel{\text{df}}{=} [\beta \mapsto b, \delta \mapsto d, \gamma \mapsto \Delta]$ is a context allowing their interpretation. In this particular case ς implies that: (i) α is a channel holder bound by an input prefix (since α is not in the domain of the context mapping though it occurs in $\mathcal{P}$); (ii) β and δ correspond respectively to the known channels b and d; and (iii) γ is a channel holder corresponding to the restricted channel Δ, the detailed identity of this restricted channel being irrelevant.

Now, given our well-formed indexed expression $A \vdash P$ together with process definitions $D_{X_1}, \ldots, D_{X_m}$, we proceed as follows. For each channel name c

occurring in the indexed expression or process definitions, we choose a *distinct* channel holder α_c. The bodies of $P, D_{X_1}, \ldots, D_{X_m}$ are then transformed by first deleting all the instances of the restriction operator, and then replacing each occurrence of each channel by the corresponding channel holder, resulting in new holder-based terms: $\mathcal{P}, \mathcal{D}_{X_1}, \ldots, \mathcal{D}_{X_m}$.

We then construct contexts, $\varsigma, \varsigma_1, \ldots, \varsigma_m$. The context ς maps every α_c used in the body of P for which c was restriction bound into a distinct restricted channel Δ_c, and every α_c for which $c \in A$ into c. Each ς_i simply maps every α_c which is restriction bound in the body of D_{X_i} into a distinct Δ_c. We finally obtain a *main* expression $\mathcal{H} = \mathcal{P}{:}[\varsigma]$ and the modified definitions $\mathcal{D}_{X_1}{:}[\varsigma_1], \ldots, \mathcal{D}_{X_m}{:}[\varsigma_m]$, which will be used as an input to our translation into Petri nets.

For example, our running example can be rendered in the context-based scheme as: $X(\alpha) + \tau\,.\,\mathtt{0} : [\alpha \mapsto a]$ with $X(\epsilon) \stackrel{\text{df}}{=} \epsilon\gamma\,.\,\overline{\gamma}\epsilon\,.\,\mathtt{0} + (X(\delta)|\overline{\delta}\epsilon\,.\,\mathtt{0}) : [\delta \mapsto \Delta]$.

3 An Algebra of Nets

The development of our Petri net model, called *rp-nets*, has been inspired by the box algebra [1,2,10] and by the p-net algebra used in [8] to model the finite fragment of π-calculus. In particular, we shall use coloured tokens and read-arcs (allowing any number of transitions to simultaneously check for the presence of a resource stored in a place [6]). Transitions in rp-nets will have four different kinds of labels:

- UV, Uv and $\overline{U}V$ (where U, V and v are net variables) to specify communication with the external environment.
- τ to represent internal actions.
- *rcv* and *snd* to effect internal process communication.
- $X(\alpha_1, \ldots, \alpha_{n_X})$ to identify hierarchical transitions supporting recursion (we use gray rectangles to represent such transitions).

A key idea behind our translation of a context-based expression using a set of process definitions is to view this system as consisting of a main program together with a number of procedure declarations. We then represent the control structure of the main program and the procedures using disjoint unmarked nets, one for the main program and one for each of the procedure declarations. The program is executed once, while each procedure can be invoked several times (even concurrently), each such invocation being uniquely identified by structured tokens which correspond to the sequence of recursive calls along the execution path leading to that invocation.[1] With this in mind, we will use the notion of a *trail* σ to denote a finite (possibly empty) sequence of hierarchical transitions of an rp-net. The places of the nets which are responsible for control flow will carry tokens which are simply trails. (The empty trail will be treated as the usual

[1] That this sequence is sufficient to identify an invocation will follow from the fact that a given hierarchical transition may be activated many times, but each time with a different sequence.

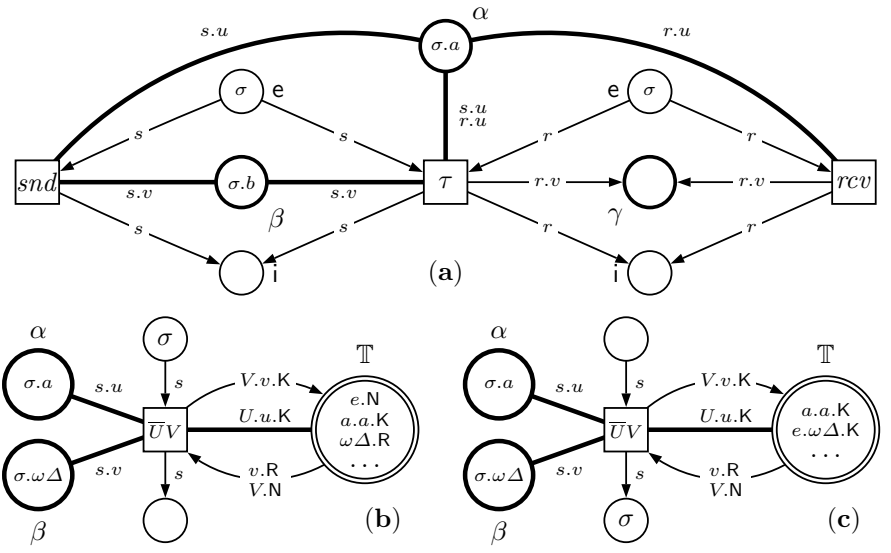

Fig. 1. Holder places and read arcs (**a**), and the usage of the tag-place (**b**, **c**), where K, R, N are constants, while the other symbols in arc inscriptions are net variables

'black' token.) Procedure invocation is then possible if each of the input places of a hierarchical transition t labelled with $X(\ldots)$ contains the same trail token σ, and it results in removing these tokens and inserting a new token σt in each initial (entry) place of the net corresponding to the definition of X.

Trailed channels and holder places. Places in rp-nets are labelled in ways reflecting their intended role. Those used to model control flow are labelled by their status symbols (*internal* places by i, and *interface* places by e and x, for entry and exit, respectively), and the tokens they carry are simply the trails σ. Another kind of places, called *holder* places, carry structured tokens representing channels used by different procedure invocations. Each such token, called a *trailed channel*, is of the form $\sigma.\xi$ where σ is a trail and ξ is a known channel in $\mathbb{C}$, or a restricted channel $\omega\Delta$ (ω is a trail and $\Delta \in \mathbb{R}$). Intuitively, its first part, σ, identifies the invocation in which the token is used, while the second part, a or $\omega\Delta$, provides its value. In the diagrams, holder places are labelled by the elements of $\mathbb{H}$ and have thick borders. (A third kind of places will be introduced later on.)

Referring to figure 1(a), a holder place can be accessed by directed arcs, which can deposit or remove tokens, as well as by read arcs (drawn as thick undirected edges), which *test* for the presence of specific tokens. The net itself may be seen as a fragment of the translation of a context-based process definition, $Y(\alpha, \beta) \stackrel{\text{df}}{=} (\overline{\alpha}\beta \ \ldots | \alpha\gamma \ \ldots) : [\ ]$, where the channel holders α, β and γ are represented by the corresponding holder places. The depicted procedure invocation has been activated by the trail σ, and two trailed channels, $\sigma.a$ and $\sigma.b$, have been inserted as actual parameters into the holder places labelled by α and β, respectively. On

the other hand, the γ-labelled holder place, corresponding to an input bound channel holder, remains empty until a communication or input action inserts a trailed channel into it.

Control flow places are connected using directed arcs, labelled with trail variables, s or r, while the holder places are connected using directed arcs and read arcs labelled with structured annotations, like $s.u$, with two variables directly matching the type of tokens allowed in holder places. To interpret arcs annotations, we use *bindings* $\flat$ assigning concrete values to the variables occurring there as well as those appearing in transition labels. In our setting, $\flat(s)$ and $\flat(r)$ return trails, $\flat(u)$ and $\flat(v)$ return channels (or trailing restricted channels), whereas $\flat(U)$ and $\flat(V)$ return channels. As usual in high-level Petri nets, bindings will yield tokens transferred/tested along arcs adjacent to executed transitions as well as the visible labels of the latter.

For the net depicted in figure 1(a), the *rcv*-labelled transition is enabled if the right entry place contains a trail (in our case, σ) and the α-labelled place contains a trailed channel with precisely the same 'trail' part (in our case, $\sigma.a$). More formally, there must be a binding $\flat$ such that $\flat(r)$ evaluates to σ and $\flat(r.u) \stackrel{\text{df}}{=} \flat(r).\flat(u)$ evaluates to $\sigma.a$. Indeed, such a binding can be constructed, by setting $\flat(r) = \sigma$ and $\flat(u) = a$. The firing of the *rcv*-labelled transition transforms the current marking in the following way: σ is removed from the right entry place and deposited in the right internal place, the token in the α-labelled place is left unchanged, and a trailed channel $\flat(r.v)$ (e.g., $\sigma.e$ or $\sigma.b$, depending on the choice of the binding which in this case is not unique) is inserted into the γ-labelled holder place. Similarly, the firing of the *snd*-labelled transition is also possible and results in a transfer of the trail σ from the left entry place to the left internal place. Now, if we look at the firing of the τ-labelled transition, which corresponds to the fusion of the two transitions considered previously, the binding with $\flat(v) = e$ is inconsistent with the only binding option for v (i.e., $\flat(v) = b$), and so a unique internal communication is possible through which the γ-labelled holder place acquires the trailed channel $\sigma.b$.

Tag-places. The third, and last, kind of node in a rp-net is a special holder place, called the *tag-place*, which is always present and unique; it is $\mathbb{T}$-labelled and indicated in the diagrams by a double border. The tokens, called *bookkeeping tokens*, stored in this place are structured by being *tagged* with a member of the set $\mathbb{T} \stackrel{\text{df}}{=} \{\mathsf{K}, \mathsf{N}, \mathsf{R}\}$. The first tag, K, will be used to indicate the known channels (initially, those in $\varsigma(\mathbb{H}) \cap \mathbb{C}$). The second tag, N, will be used to indicate the new, yet unknown channels (initially, those in $\mathbb{C} \setminus \varsigma(\mathbb{H})$), and the third tag, R, will be used to indicate the restricted channels. The first case is slightly more complicated than the remaining two, for a restricted $\omega\Delta$ may be present with different preceding trails σ's in holder places, due to internal communications.[2] Now, if the restriction has been *opened*, $\omega\Delta$ should become a newly known channel c, but it is not possible to replace $\omega\Delta$ by c in all the relevant holder places

[2] More precisely, we may have various $\omega\Delta$'s in various holder places with the same trail σ due to internal communications, and with different σ's due to parameter passing.

without some global transformation of the net. Instead, we will indicate this fact by inserting a bookkeeping token $c.\omega\Delta.\mathsf{K}$ into the tag-place, and then consulting it whenever necessary (i.e., whenever we need to establish whether a restricted channel has been opened and what is its actual known value). Moreover, to keep the notation uniform, we shall use bookkeeping tokens $a.a.\mathsf{K}$ to denote all those known channels which were never restricted. To summarise, a bookkeeping token in the tag-place may be of the form:

- $a.\mathsf{N}$ meaning that a is a new channel.
- $\omega\Delta.\mathsf{R}$ meaning that Δ is a restricted channel for the incarnation of its defining process corresponding to trail ω.
- $a.a.\mathsf{K}$ meaning that a is a known channel (either a has always been known or a was initially new and then became known).
- $a.\omega\Delta.\mathsf{K}$ meaning that the restricted $\omega\Delta$ has become known as a.

The arcs adjacent to the tag-place (both directed and read ones) are labelled with annotations which are evaluated through bindings so that the tags are left intact; e.g., $\flat(V.\mathsf{N}) \stackrel{\text{df}}{=} \flat(V).\mathsf{N}$ and $\flat(U.u.\mathsf{K}) \stackrel{\text{df}}{=} \flat(U).\flat(u).\mathsf{K}$.

To explain the way a tag-place is used, we consider an rp-net fragment in figure 1(b), where the (irrelevant) labels of the two control flow places have been omitted. The marking in the tag-place indicates that $\omega\Delta$ is a restricted channel in the incarnation of some procedure definition identified by ω. Moreover, e is a new unknown channel, and a is a known one. The transition is enabled with the binding $\flat(u) = \flat(U) = a$, $\flat(v) = \omega\Delta$, $\flat(V) = e$ and $\flat(s) = \sigma$. Its firing produces the visible action $\flat(\overline{U}V) \stackrel{\text{df}}{=} \overline{\flat(U)}\flat(V) = \overline{a}e$ and leads to the marking in figure 1(c). This firing illustrates how a restricted channel becomes known (which is represented by the insertion of the bookkeeping token $e.\omega\Delta.\mathsf{K}$ in the tag-place), and corresponds to the OPEN rule in table 1.

Composing rp-nets. The operators we shall use to combine rp-nets can be seen as suitably adapted instances of those defined within PBC and its various extensions [1,10]. In particular, the way in which the holder places are handled when composing nets is directly inspired by the asynchronous communication construct of APBC [10].

The rp-net composition operators that we need are *prefixing*, $N.N'$, *choice*, $N + N'$, *parallel composition*, $N|N'$, and *scoping*, $\mathbf{sco}(N)$. The first three operators merge the tag-places, as well as the corresponding holder places (i.e., labelled by the same channel holder). This corresponds to the asynchronous links used in [10], and will allow one to mimic the standard rewriting mechanism of the π-calculus. For two operand nets, their transitions and control flow places are made disjoint before applying a composition operator in order to allow to properly handle the cases when, for example, $N = N'$.

- In the choice composition, the entry and exit places of N and N' are combined through a cartesian product together. This has the following effect: if we start from a situation where each entry place contains a copy of a common token σ, then either N or N' can be executed, mimicking the SUM rule and its symmetric counterpart.

- The prefixing operator combines the exit place (it will always be unique) of the prefix N with the entry places of N' into internal places, and the effect is that the execution of N after reaching the terminal marking, where the only exit place is marked, is followed by that of N'. Such a behaviour mimics the Tau, In and Out rules.
- The parallel composition of N and N' puts them side by side, allowing to execute both parts in parallel, as in the Par rule and its symmetric counterpart; and then merges all pairs of transitions labelled *rcv* and *snd*, resulting in τ-labelled transitions: the connectivity of the new transition is the combination of those of the composed transitions. This merging is illustrated in the middle of figure 1; it has an effect similar to the Com rule.
- Finally, the scoping operation erases all the *rcv*- and *snd*-labelled transitions.

4 Translating Context-Based Expressions into Rp-Nets

We now come back to our context-based expression $\mathcal{H} = \mathcal{P}{:}[\varsigma]$ and the process definitions $\mathcal{D}_{X_1}{:}[\varsigma_1], \ldots, \mathcal{D}_{X_m}{:}[\varsigma_m]$. The proposed rendering of $\mathcal{H}$ in terms of rp-nets is obtained in three phases. First, we compositionally translate $\mathcal{P}$ and each $\mathcal{D}_{X_i}$ into disjoint unmarked rp-nets $\mathbb{K}(\mathcal{P}), \mathbb{K}(\mathcal{D}_{X_1}), \ldots, \mathbb{K}(\mathcal{D}_{X_m})$. The resulting nets are then combined using parallel composition and scoping. Finally, using the contexts $\varsigma, \varsigma_1, \ldots, \varsigma_m$, we construct an initial marking, which results in the target rp-net $\mathbb{PN}(\mathcal{H})$.

Phase I. The translation $\mathbb{K}(\mathcal{P})$ (resp. $\mathbb{K}(\mathcal{P}_X)$), guided by the syntax tree of $\mathcal{P}$, consists in first giving the translation for the basic sub-terms (i.e., the basic process $\mathsf{0}$, the process calls, and the internal, input and output prefixes) shown in figure 2, and then applying rp-net operators following the syntax.

The translations of the basic process $\mathsf{0}$ and the internal prefix τ are very simple (they do not involve any manipulation on channels). The same is true of a process call $X(\alpha_1, \ldots, \alpha_{n_X})$, but the result employs a hierarchical transition, which will never fire but rather contribute to a marking equivalence.

Each output prefix $\overline{\alpha}\beta$, for $\alpha \neq \beta$, is translated into the rp-net $\mathbb{K}(\overline{\alpha}\beta)$ which may exhibit three kinds of behaviours, corresponding to the firing, under some binding $\flat$, of three specific transitions:

- t^k: *known output*. A known channel $\flat(V)$ is sent through a channel $\flat(U)$. The actual values of U and V are provided by the bookkeeping tokens present in the tag-place matching those in the holder places α and β, accessed through u and v and preceded by a trail which corresponds, thanks to the common annotation s, to a token in the entry place. This corresponds to the Out rule. That the channels $\flat(U)$ and $\flat(V)$ are known is determined by the presence in the tag-place of bookkeeping tokens tagged with K.
- t^n: *new output*. A new channel $\flat(V)$ is sent through a known channel $\flat(U)$, for some trail token, $\flat(s)$ as before. That the channels $\flat(v)$ and $\flat(V)$ are respectively restricted and new is determined by the presence in the tag-place of a bookkeeping token tagged with R for $\flat(v)$, and a bookkeeping

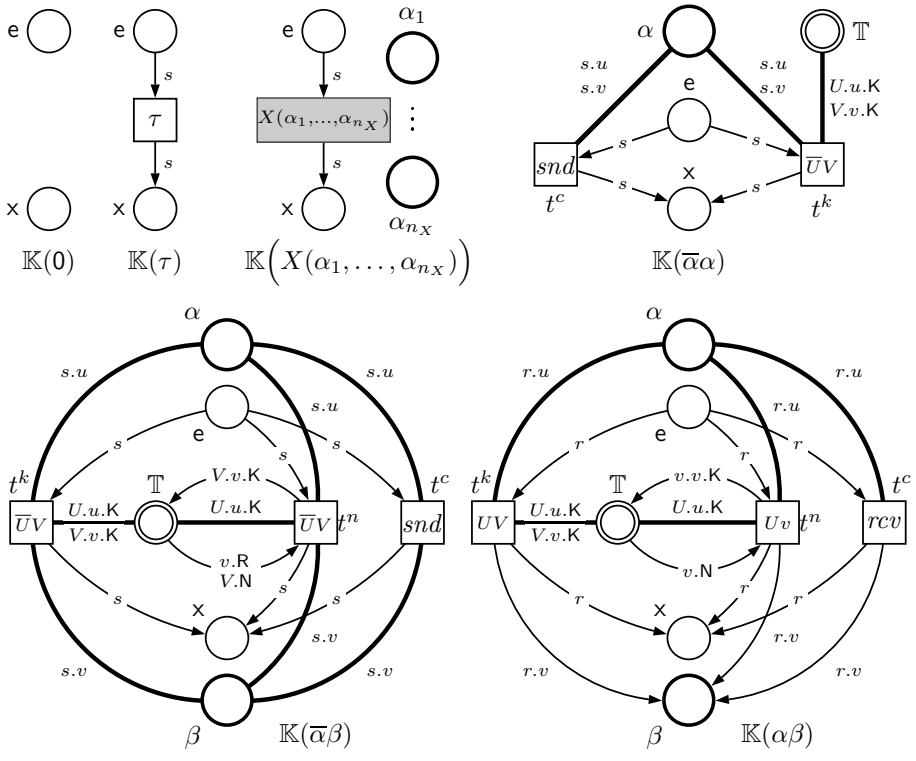

Fig. 2. The unmarked rp-nets for $\mathsf{0}$, τ, the process call and the three kinds of prefixes (the tag-place is omitted when disconnected)

token tagged with N for $\flat(V)$. After the firing of this transition, the restricted channel represented by v becomes known; this is indicated by inserting a bookkeeping token of the form $\flat(V.v.\mathsf{K})$ into the tag-place which now replaces $\flat(v.\mathsf{R})$ and $\flat(V.\mathsf{N})$. This corresponds to the Open rule.

– t^c: *communicating output*. It is intended to synchronise with a corresponding communication input in order to provide the transfer of a channel $\flat(v)$ through the channel $\flat(u)$, be it known or restricted. This models one of the main features of the π-calculus which is the ability of *passing* the channels around.

The special case of the output prefix $\overline{\alpha}\alpha$ has a simpler translation, since α may not be both known and restricted, so that t^n is irrelevant in this case. Even though the α-labelled holder place will never contain more than one token with the same trail part, it is not a problem to have two variables on the arcs adjacent to it since these are read arcs, and so transitions will be enabled by simply identifying $s.u$ and $s.v$ with the same token in the α-labelled place.

For an input prefix $\alpha\beta$, the translation is broadly the same as for the output prefix (notice that prefixes of the form $\alpha\alpha$ are excluded by the well-formedness

assumption). In particular, the known, new and communicating inputs should be interpreted in a similar way. Simply, $\flat(r.v)$ is now inserted into β instead of being checked, t^k corresponds to the rule In when b is already known (including the case of a previously restricted channel), and t^n to the same rule when b is new (it may not be restricted here). In the latter case, the variable V is not involved, and the transition is labelled Uv rather than UV. Notice also that, for t^k, while $\flat(v)$ is known as $\flat(V)$, it is the possibly restricted original value $\flat(v)$ which is written (together with the trailed token $\flat(s)$) into β, and not the corresponding known value for V. This is important in order to allow subsequent synchronisations between *rcv* (with $\flat(u)$) coming from β and *snd* (with $\flat(u)$) coming from another holder place and containing a copy of the original token.

For the compound sub-terms, we proceed compositionally: $\mathbb{K}(\mathcal{P}' \; op \; \mathcal{P}'') \stackrel{\text{df}}{=} \mathbb{K}(\mathcal{P}') \; op \; \mathbb{K}(\mathcal{P}'')$, where $op \in \{|, +, .\}$. There is however a simplification which may be applied to the above translation rule, which amounts to throwing away useless instances of the translation for $\mathsf{0}$. One simply has to first apply the following simplifications: $\mathbb{K}(P|\mathsf{0}) = \mathbb{K}(\mathsf{0}|P) \rightsquigarrow \mathbb{K}(P)$ and $\mathbb{K}(a.\mathsf{0}) \rightsquigarrow \mathbb{K}(a)$ (notice that we already have that $\mathbb{K}(P + \mathsf{0}) = \mathbb{K}(\mathsf{0} + P) = \mathbb{K}(P)$). Finally, we add a holder place for each channel holder occurring in the domain of ς but not in $\mathcal{P}$.

The translation of process definitions proceeds similarly. Assuming that $\mathcal{D}_X$ is of the form $X(\kappa_1^X, \ldots, \kappa_{n_X}^X) \stackrel{\text{df}}{=} \mathcal{P}_X$, we derive $\mathbb{K}(\mathcal{P}_X)$ following the scheme just described, and then add a holder place for each channel holder κ_i^X which does not occur in $\mathcal{P}_X$.

Phase II. The various components are then connected by constructing the net $\mathbf{sc}\left(\mathbb{K}(\mathcal{P})|\mathbb{K}(\mathcal{D}_{X_1})|\ldots|\mathbb{K}(\mathcal{D}_{X_m})\right)$. This merges the various tag places and the pairs of *snd*- and *rcv*-labelled transitions, possibly from different components. All the *rcv*- and *snd*-labelled transitions are erased after that.

Phase III. Having applied the parallel composition and scoping, we add the initial marking, leading to the full translation $\mathbb{PN}(\mathcal{H})$, in the following way:

- An empty trail token $\bullet$ is inserted in each entry place of $\mathbb{K}(\mathcal{P})$.
- $\bullet.\varsigma(\alpha)$ trailed channel (in the diagrams represented as $\varsigma(\alpha)$, $\bullet$ representing the empty trail) is inserted into each α-labelled holder place, for $\alpha \in dom(\varsigma)$.
- $\omega.\omega\varsigma_i(\alpha)$ trailed channel is inserted into the α-labelled holder place, and $\omega\varsigma_i(\alpha).\mathsf{R}$ bookkeeping token is inserted into the tag-place, for each trail ω and $\alpha \in dom(\varsigma_i)$ $(1 \leq i \leq m)$.
- $a.a.\mathsf{K}$ bookkeeping token is inserted into the tag-place, for $a \in \varsigma(\mathbb{H}) \cap \mathbb{C}$.
- $e.\mathsf{N}$ bookkeeping token is inserted into the tag-place, for $e \in \mathbb{C} \setminus \varsigma(\mathbb{H})$.
- $\Delta.\mathsf{R}$ bookkeeping token is inserted into the tag-place, for $\Delta \in \varsigma(\mathbb{H}) \cap \mathbb{R}$.

Figure 3 (top) shows the rp-net resulting from the translation where, for clarity, all the arcs adjacent to the tag-place and holder places are omitted and arc annotations have been shortened or omitted.

5 Firing Rule and Marking Equivalence

To define the semantics of the resulting rp-net $\mathbb{PN}(\mathcal{H})$, we need to introduce the firing rule for non-hierarchical transitions, which may use a combination of annotated standard (oriented) arcs and read-arcs, and a marking equivalence corresponding to procedure calls.

For each transition t, we denote by $\iota(t)$ its label (a term with variables), by $\iota(s,t)$, $\iota(t,s)$ and $\iota(\{s,t\})$ the labels (sets of terms with variables) of its incoming arcs, out-going arcs and read-arcs, respectively. We shall assume that each variable has an associated domain of possible values, for instance, the domain of u, v, U and V is $\mathbb{C} \cup \{\omega\Gamma \mid \omega \text{ is a trail}, \Gamma \in \mathbb{R}\}$ and that of r and s is $\{\omega \mid \omega \text{ is a trail}\}$. For each transition t, if $\{u_1, ..., u_n\}$ denotes the variables occurring in the label of t and on the arcs adjacent to t, we shall denote by $\flat$ a binding assigning to each variable u_i a value in its domain. We shall only consider legal bindings, i.e., such that for each arc $\mathfrak{A}$ between t and an adjacent place s, if $\hbar$ is a function in $\iota(\mathfrak{A})$, the evaluation of $\hbar$ under the binding $\flat$ (denoted $\flat(\hbar)$) will deliver a value allowed in s. Moreover, the observed label of a transition fired under binding $\flat$ will be denoted by $\flat(\iota(t))$.

A *marking* $\mathcal{M}$ of a rp-net N is a function assigning to each place s a multiset of tokens belonging to its type. A marked rp-net will be denoted by $(N, \mathcal{M})$. Below we use $\oplus$ and $\ominus$ to denote respectively multiset sum and difference. Moreover, if $\mathcal{M}$ and $\mathcal{M}'$ are multisets over the same set of elements Z then $\mathcal{M} \geq \mathcal{M}'$ will mean that $\mathcal{M}(z) \geq \mathcal{M}'(z)$, for all $z \in Z$. We shall also denote by $z \in \mathcal{M}$ the fact that $\mathcal{M}(z) > 0$.

Let $\mathcal{M}$ be a marking of $\mathbb{PN}(\mathcal{H})$, t be any of its non–hierarchical transitions, and $\flat$ be a binding for t. Then we denote by $\mathcal{M}^{\flat}_{t,in}$ and $\mathcal{M}^{\flat}_{t,out}$ the two markings such that, for every place s,

$$\mathcal{M}^{\flat}_{t,in}(s) \stackrel{\text{df}}{=} \bigoplus_{\hbar \in \iota((s,t))} \{\flat(\hbar)\} \quad \text{and} \quad \mathcal{M}^{\flat}_{t,out}(s) \stackrel{\text{df}}{=} \bigoplus_{\hbar \in \iota((t,s))} \{\flat(\hbar)\} \ .$$

A non-hierarchical transition t will be *enabled* (i.e., allowed to be fired) under the binding $\flat$ and the marking $\mathcal{M}$ if, for every place s, $\mathcal{M}(s) \geq \mathcal{M}^{\flat}_{t,in}(s)$ and, moreover[3], $\flat(\hbar) \in \mathcal{M}(s)$ for every $\hbar \in \iota(\{s,t\})$. An enabled t may then be *fired*, which transforms the current marking $\mathcal{M}$ into a new marking $\mathcal{M}'$ in such a way that, for every place s:

$$\mathcal{M}'(s) \ = \ \mathcal{M}(s) \ominus \mathcal{M}^{\flat}_{t,in}(s) \oplus \mathcal{M}^{\flat}_{t,out}(s) \ .$$

This will be denoted by $(N, \mathcal{M}) \stackrel{\flat(\iota(t))}{\longrightarrow} (N, \mathcal{M}')$ and moves of this type will be used in the definition of labelled transition systems generated by rp-nets.

As already mentioned, hierarchical transitions do not fire; instead, they drive a marking equivalence $\equiv$ corresponding to procedure calls (this resembles to certain extent the approach presented in [16]). This relation is defined as the

[3] Notice that this allows to have $\mathcal{M}(z) = 1$ and $\hbar_1, \hbar_2 \in \iota(\{s,t\})$ with $\flat(\hbar_1) = z = \flat(\hbar_2)$.

smallest equivalence such that, for each transition t labelled $X(\alpha_1, \ldots, \alpha_{n_X})$ with a definition $X(\kappa_1^X, \ldots, \kappa_{n_X}^X) \stackrel{\text{df}}{=} \mathcal{P}_X{:}\varsigma_X$, and for each trailed channel $\sigma.\xi_i$ (for $1 \leq i \leq n_X$), we have

$$\mathcal{M} \oplus \mathcal{M}_t^\sigma \oplus \mathcal{M}_\alpha^{\sigma.\xi} \quad \equiv \quad \mathcal{M} \oplus \mathcal{M}_X^{\sigma t} \oplus \mathcal{M}_\alpha^{\sigma.\xi} \oplus \mathcal{M}_\kappa^{\sigma t.\xi}\,, \quad \text{where:} \tag{1}$$

- $\mathcal{M}_t^\sigma$ is the marking with a trail σ in each place s such that there is a directed arc from s to t (note that hierarchical transitions are only connected to control flow places), and nothing else.
- $\mathcal{M}_X^{\sigma t}$ is the marking with a trail σt in each entry place of $\mathbb{K}(\mathcal{P}_X)$, and nothing else.
- $\mathcal{M}_\alpha^{\sigma.\xi}$ (and $\mathcal{M}_\kappa^{\sigma t.\xi}$) is the marking with a trailed channel $\sigma.\xi_i$ (resp. $\sigma t.\xi_i$) in the holder place for α_i (resp. κ_i), for $i = 1 \ldots n_X$, and nothing else.

The complete behaviour of the rp-net $\mathbb{PN}(\mathcal{H})$, is then given by a labelled transition system derived using the above rules (the nodes of this system are $\equiv$-equivalence classes of net markings, and $(N, [\mathcal{M}_1]_\equiv) \xrightarrow{act} (N, [\mathcal{M}_2]_\equiv)$ iff there are $\mathcal{M}_1'$ and $\mathcal{M}_2'$ such that $\mathcal{M}_1 \equiv \mathcal{M}_1' \xrightarrow{act} \mathcal{M}_2' \equiv \mathcal{M}_2$), and denoted $\mathsf{lts}_{\mathbb{PN}(\mathcal{H})}$.

For the running example, a double application of the rule (1) above leads, from the initial marking shown in figure 3(a) to that in figure 3(b) (for clarity, we omitted all the arcs linking the transitions of the process definition part with the holder places). From there, the rightmost τ-transition may be fired, with the binding $\flat = \{r \mapsto tz, s \mapsto t, u \mapsto t\Delta, v \mapsto a\}$. The resulting marking is illustrated in figure 3(c), where all the arcs linking the executed transition with the holder places are shown and, for clarity, the other transitions appearing in the part of the net corresponding to the process definition have been left out.

6 Main Results

We proposed in this paper a translation from finite π-calculus specifications to rp-nets with finite structure, i.e., $\mathbb{PN}(\mathcal{H})$ is a net with finitely many places, transitions and arcs. Even though its marking is not finite, a closer examination of the possible reachable markings can reveal that they all may be obtained from a finite *condensed* form. For instance, in the rp-net for the running example, the marking of the holder place δ is constant and each token $\sigma.\sigma\Delta$ in it only becomes checkable if we enter a process instance corresponding to the trail σ, while the marking of the tag-place $\mathbb{T}$ is constantly such that the N-tagged and R-tagged tokens may be deduced from the K-tagged ones, which are a finite number. In other words, only a finite number of tokens is relevant at any given stage of evolution, and one can keep adding new tokens 'on demand' as the new instantiations of procedure calls are entered, and as new channels become known. Crucially, the proposed translation is *sound*. This means that, for every well-formed indexed π-expression $A \vdash P$, its labelled transition system $\mathsf{lts}_{A\vdash P}$ is strongly bisimilar [17] to the labelled transition system $\mathsf{lts}_{\mathbb{PN}(\mathcal{H})}$ of the corresponding rp-net.

Theorem 1. *$\mathbb{PN}(\mathcal{H})$ is an rp-net with finitely many places and transitions such that $\mathsf{lts}_{A\vdash P}$ and $\mathsf{lts}_{\mathbb{PN}(\mathcal{H})}$ are strongly bisimilar transition systems.*

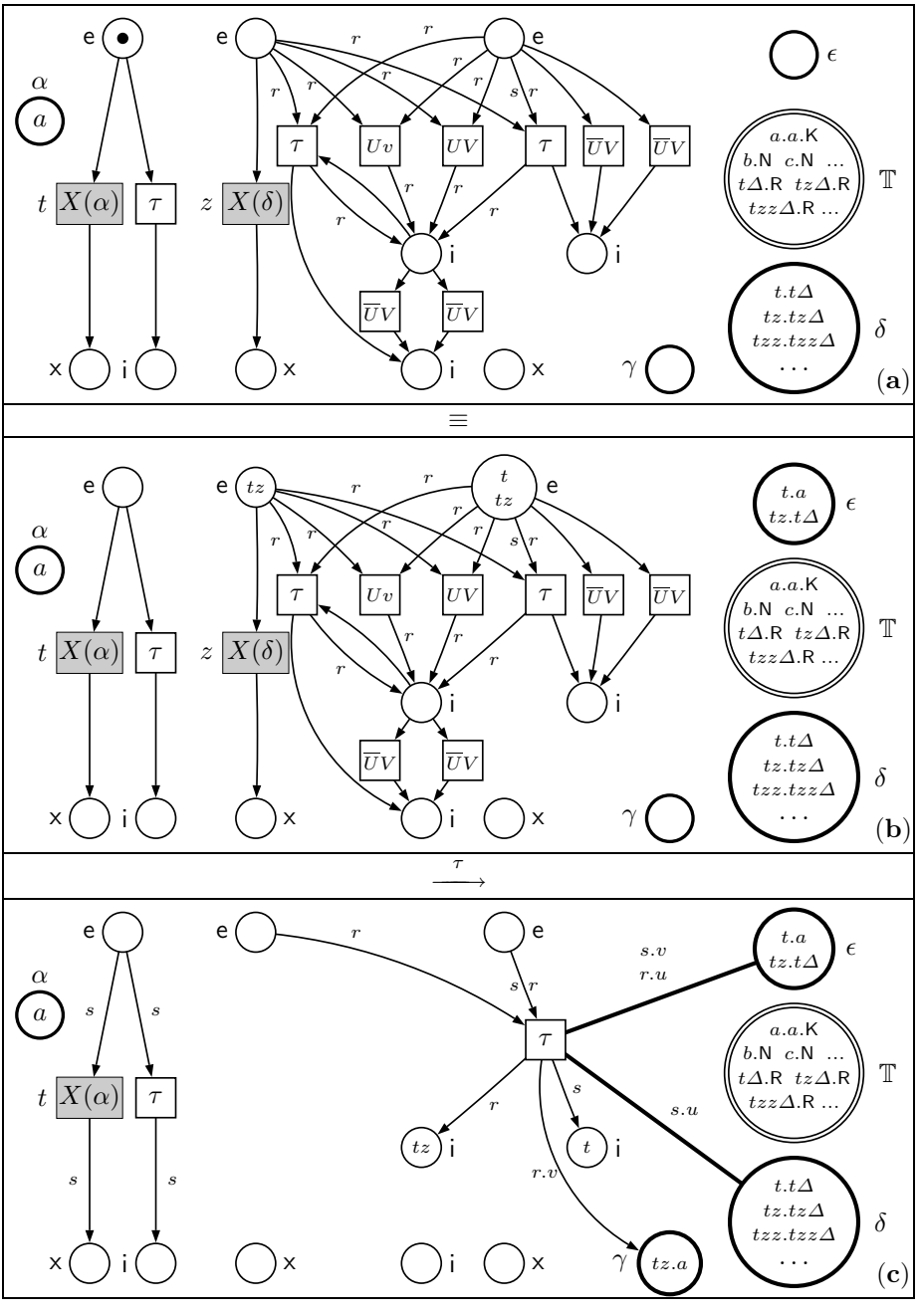

Fig. 3. Translation of the running example **(a)**; marking transformation **(b)**; and transition firing **(c)**. For clarity, in **(a,b)**, all the directed arcs annotated with s only are shown unannotated.

The above theorem yields a bisimilarity property between the labelled transition systems of the original π-agents and their Petri net translations. That the result is not formulated in terms of isomorphism is due to the *amnesia* feature of the SUM rule, complemented by a similar phenomenon for the prefix rules (TAU, IN and OUT). The loss of the past in the prefix rules is not essential by itself, since it corresponds to the progression of the control flow token(s) in the corresponding net. However, when combined with the SUM rule, which completely forgets the non-chosen branch, this may lead to differences in the labelled transition systems. Indeed, if we have a choice between two branches with a reachable common suffix then, after some time, the two branches will lead to the same intermediate expression in the π-semantics, while this never happens in the Petri net semantics, where the two branches correspond to two disjoint subnets, hence two different markings depending on which branch has been chosen.

The proposed translation is also a conservative extension of that in [8].

Theorem 2. *If the indexed π-expression $A \vdash P$ is recursion-free (i.e., $m = 0$) then $\mathbb{PN}(\mathcal{H})$ is the same as that in [8] up to some dead transitions and the suppression of all trails and trail variables.*

7 Related Work and Concluding Remarks

A first paper giving a Petri net semantics for the π-calculus is [12]. However, it only considers the so-called 'small' π-calculus (without the choice composition) provided with the reduction semantics (addressing only the communications between parallel components). Due to these limited aims, the problem is greatly simplified as restrictions may be managed syntactically. While not based on nets, [3] already considers the causality structures of the π-calculus, and distinguishes structural and link dependencies (the former mainly due to prefixing and communications, and the latter due to extrusion). A graph-rewriting system is proposed in [19] as a semantic model for a fragment of the π-calculus mainly addressing the concurrency feature of systems. Although it is not the objective of the present paper, we intend to look at concurrency issues, and in this respect we may notice a discrepancy between our approach and [19] in the handling of restriction. More precisely, [19] allows parallel opening for expressions like $(\nu y)(\overline{x}y|P|\overline{z}y)$ by letting the actions $\overline{x}y$ and $\overline{z}y$ to occur in parallel, while in our approach they must in some sense agree on their common exportation, so that only one of them is in fact an opening. The translation of π-terms into Petri nets of [4] uses (low-level) labelled Petri nets extended with inhibitor arcs, while we use high-level nets with read-arcs. Moreover, the way compositionality is obtained is different from that used in our approach, relying to a construction of a general infrastructure, with places corresponding to all the possible sequential π-terms with all possible transitions between those places, and a compositionally defined initial marking corresponding to each π-term.

We outlined a translation of recursive π-calculus process expressions into high-level Petri domain. The next step is to incorporate it into suitable computer

aided tools, such as PEP [13], to allow the verification and simulation of π-calculus specifications using techniques found in the Petri net domain (for the finite π-calculus translation of [8] this has been already been done in [15]).

Acknowledgements. We thank the anonymous referees for their helpful comments. This research was supported by the EC IST grant 511599 (RODIN).

References

1. Best, E., Devillers, R., Koutny, M.: Petri Net Algebra. EATCS Monographs on TCS, Springer (2001).
2. Best, E., Fraczak, W., Hopkins, R.P., Klaudel, H., Pelz, E.: M-nets: an Algebra of High Level Petri Nets, with an Application to the Semantics of Concurrent Programming Languages. Acta Informatica **35** (1998) 813–857
3. Boreale, M., Sangiorgi, D.: A Fully Abstract Semantics for Causality in the π-calculus. Proc. of STACS'95, Springer, LNCS 900 (1995) 243–254
4. Busi, N., Gorrieri, R.: A Petri net Semantics for π-calculus. Proc. of CONCUR'95, LNCS 962 (1995) 145–159
5. Cattani, G.L., Sewell, P.: Models for Name-Passing Processes: Interleaving and Causal. Proc. of LICS'2000, IEEE CS Press (2000) 322–333
6. Christensen, S., Hansen, N.D.: Coloured Petri Nets Extended with Place Capacities, Test Arcs and Inhibitor Arcs. Proc. of ICATPN'93, Springer, LNCS 691 (1993) 186–205
7. Devillers, R., Klaudel, H.: Solving Petri Net Recursions through Finite Representation. Proc. of IASTED'04, ACTA Press (2004) 145–150
8. Devillers, R., Klaudel, H., Koutny, M.: Petri Net Semantics of the Finite π-Calculus Terms. Fundamenta Informaticae **70** (2006) 1–24
9. Devillers, R., Klaudel, H., Koutny, M.: A Petri Net Translation of pi-calculus Terms. Report CS-TR 887, University of Newcastle (2005)
10. Devillers, R., Klaudel, H., Koutny, M., Pommereau, F.: Asynchronous Box Calculus. Fundamenta Informaticae **54** (2003) 295–344
11. Devillers, R., Klaudel, H., Riemann, R.-C.: General Parameterised Refinement and Recursion for the M-net Calculus. Theoretical Comp. Sci. **300** (2003) 235–258
12. Engelfriet, J.: A Multiset Semantics for the π-calculus with Replication. Theoretical Comp. Sci. **153** (1996) 65–94
13. Grahlmann, B., Best, E.: PEP - More than a Petri net tool. Proc. of TACAS'96, Springer, LNCS 1055 (1996) 397–401
14. Haddad, S., Poitrenaud, D.: Modelling and Analyzing Systems with Recursive Petri Nets. Proc. of WODES'00, Kluwer Academics Publishers (2000) 449–458
15. Khomenko, V., Koutny, M., Niaouris, A.: Applying Petri Net Unfoldings for Verification of Mobile Systems. Report CS-TR 953, University of Newcastle (2006)
16. Kiehn, A.: Petri Net Systems and their Closure Properties. In: Advances in Petri Nets 1989, Rozenberg, G (Ed.). Springer, LNCS 424 (1990) 306–328
17. Milner, R.: Communication and Concurrency. Prentice Hall (1989).
18. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes. Information and Computation **100** (1992) 1–77
19. Montanari, U., Pistore, M.: Concurrent Semantics for the π-calculus. Proc. of MFPS'95, Electronic Notes in Computer Science 1, Elsevier (1995) 1–19