Petri Net Semantics of the Finite π-Calculus

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

[1] Université Libre de Bruxelles, B-1050 Bruxelles, Belgium
rdevil@ulb.ac.be
[2] Université d'Évry, LaMI, 91000 Évry, France
klaudel@lami.univ-evry.fr
[3] University of Newcastle, NE1 7RU, U.K.
maciej.koutny@ncl.ac.uk

Abstract. In this paper we propose a translation into high level Petri nets of a finite fragment of the π-calculus. Our construction renders in a compositional way the control flow aspects present in π-calculus process expressions, by adapting the existing graph-theoretic net composition operators. Those aspects which are related to term rewriting, as well as name binding, are handled through special inscription of places, transitions and arcs, together with a suitable choice of the initial marking for a compositionally derived high level Petri net.

1 Introduction

In recent years, mobility has emerged as a key feature of many complex real life computing systems. In order to be able to model such a feature, dedicated process algebras have been designed, among which a central role is occupied by the π-calculus [13]. Within such a formalism, it is possible, for example, to model an interaction between a name server and a client willing to access a mobile provider known to the server (but not to the client), through which the physical address of the provider is acquired enabling a direct access to the provider by the client. The basic mechanism facilitating such a dynamic change is the ability to pass a *reference* (or a *channel*) through a communication on a previously known channel, allowing the recipient to use the new channel in future interactions.

The standard presentations of the π-calculus are based heavily on term rewriting and, as a result, tend to be difficult to translate into automata-based formalisms, such as Petri nets, which allow one to specify and reason about the causality and concurrency exhibited by a system. The main problem is that the standard term rewriting rules change the structure of an expression modelling the system, whereas an automata-based representation retains its structure over possible evolutions, and the changes of the state are represented explicitly (e.g., as net markings).

In this paper, we outline a compositional translation from the π-calculus to high-level Petri nets coping with this fundamental problem. Although, for brevity, we shall restrict ourselves to the finite fragment of the π-calculus without the match and mismatch constructs, the resulting theory is still rich enough to allow the description of non-trivial systems.

2 Finite π-Calculus

We start by briefly recalling the syntax and operational semantics of the π-calculus [13, 15], assuming that $\mathbb{C} \overset{\mathrm{df}}{=} \{a, b, c, \ldots\}$ is a countably infinite set of channels. The concrete syntax of the *finite* π-calculus we use is given below, where P denotes an arbitrary agent (or π-expression).

$$\begin{array}{rlll} \text{output/input/internal Prefixes} & \ell & ::= & \bar{a}b \mid ac \mid \tau \\ \text{Agents} & P & ::= & \mathsf{0} \mid \ell.P \mid P+P \mid P|P \mid (\boldsymbol{\nu}c)P \end{array}$$

The constructs $ac.P$ (input) and $(\boldsymbol{\nu}c)P$ (restriction) bind the channel c in P, and by $fn(P)$ we will denote the free channels of P. Agents are defined up to the *alpha-conversion* of bound channels, implying that the latter may be *coherently* replaced by *new* channels provided that name clashes are avoided. It is always possible to alpha-convert any agent in such a way that each bound channel only generates a single binding, and no channel is both free and bound; for instance, $ab.bb.ba.\bar{a}b.\mathsf{0}$ can be rewritten as $ab.bc.cd.\bar{d}c.\mathsf{0}$.

We will denote by $\{b/c\}P$ the agent obtained from P by replacing all the free occurrences of c by b, possibly after alpha-converting P in order to avoid channel clashes; for instance, $\{b/c\}\bar{a}c.cd.\mathsf{0} = \bar{a}b.bd.\mathsf{0}$ and $\{b/c\}ab.\bar{d}b.\bar{d}c.\mathsf{0} = ae.\bar{d}e.\bar{d}b.\mathsf{0}$.

Operational Semantics. There are several variants of the operational semantics of the π-calculus (see [15]), mainly due to different treatments of the restriction operator, which is generally considered to be the most intricate feature of the whole theory. Basically, it is possible to 'send to the outside world' a restricted channel c through a known channel a, but if c is captured by a receiving process then both the sender and receiver become part of an encompassing restriction (see the Com rule in table 1). To handle this situation correctly, one needs to know which channels are presently 'known' to the external environment, and which ones are migrating restricted channels. Unfortunately, this may not be determined just by looking at a sub-expression without considering its surrounding 'context'. As a result, we have found it advantageous to use the semantical presentation expounded in [6, 7], where the usual transition steps are augmented with an explicit information about unrestricted channels. More precisely, we use transitions of the form

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

where ℓ is a prefix and $A, B \subset \mathbb{C}$ are finite sets of *indexing* channels. Its intuitive meaning (see [7]) 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 provided that $b \notin A$ (intuitively,

Table 1. Operational semantics of π-expressions, where: $ns(\tau) \overset{\mathrm{df}}{=} \emptyset$; $ns(ab) = ns(\bar{a}b) \overset{\mathrm{df}}{=} \{a, b\}$; 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

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

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

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

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

such a b is a new channel communicated by the outside world – see the In rule in table 1), and an output $\ell = \bar{a}\ b$ adds b provided $b \notin A$ (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). We call $A \vdash P$ a *valid* indexed π-expression if P is a π-expression and $fn(P) \subseteq A$. Similarly as P, the indexed π-expression $A \vdash P$ will also be defined up to alpha-conversion (affecting P but not A). Hence we may assume that no bound channel of P is present in A, and that each bound channel only generates a single binding; in such a case, a valid indexed π-expression will be called *well-formed*. In order to simplify the presentation, we shall often omit the trailing $\mathsf{0}$'s in (indexed) π-expressions.

The operational semantics of indexed π-expressions is given in table 1 (in [6, 7], the '$B \vdash$' parts of the rules are implicit). It preserves the validity of expressions and, in combination with renaming and alpha-conversion, also their well-formedness. As usual, a complete behaviour of a valid indexed π-expression $\mathcal{P} = A \vdash P$ (defined up to alpha-conversion) can be represented by a labelled transition system (or lts) derived using the rules in table 1 and denoted $\mathsf{lts}_{\mathcal{P}}$.

In order to achieve a closer correspondence with the Petri net semantics, we slightly modify the Sum rule (as well as its symmetric version) into

$$\text{Sum}'\quad \frac{A \vdash P \xrightarrow{\ell} A' \vdash P'}{A \vdash P+Q \xrightarrow{\ell} A' \vdash P'+\mathsf{0}}$$

This has the advantage of better keeping track of the origin of the move. Indeed, if $Q = P$ then a move $A \vdash P+Q \xrightarrow{\ell} A' \vdash P'$ could well have been derived from an application of Sum as well as of its symmetric counterpart. On the other hand, a move $P+Q \xrightarrow{\ell} A' \vdash P'+\mathsf{0}$ clearly arises from Sum$'$, and $A \vdash P+Q \xrightarrow{\ell} A' \vdash \mathsf{0}+P'$ from its symmetric counterpart. We shall need this distinction since, in the Petri net translation of $A \vdash P{+}P$, one gets different (though in some sense equivalent) markings depending on which of the two translations of P has evolved. The above modification is harmless anyway, since we still have in the modified setup the standard π-calculus rules $P + \mathsf{0} \equiv P \equiv \mathsf{0} + P$.

A Running Example. We consider a system $\textsc{Dbase}$ modelling a simple database composed of a manager $\textsc{Manager} \overset{\mathrm{df}}{=} ab.(\bar{f}g + \bar{b}c)$, a file compressing process (zipper) $\textsc{Gzip} \overset{\mathrm{df}}{=} gd.\bar{a}d$, and a memory location $\textsc{Store} \overset{\mathrm{df}}{=} (\boldsymbol{\nu} h)(fe.\bar{e}h)$. Informally, the manager receives a request for a file on a visible channel a. If the file is not available, a negative response is sent back (using a newly acquired channel represented by b). Otherwise, the manager initiates a sequence of actions, the first being a sending on f to the store of a channel g (previously restricted to the manager and zipper processes) allowing a direct access to the zipper process. Upon receiving this message, the store uses the now available channel g to send a private (signified by the restriction $(\boldsymbol{\nu} h)$) file h to the zipper process. The file is then compressed and forwarded outside on channel a.

The database system is obtained by putting the three constituent processes in parallel as well as restricting the channel g connecting the manager with the zipper:

$$\textsc{Dbase} \overset{\mathrm{df}}{=} (\boldsymbol{\nu} g)(\textsc{Manager} \mid \textsc{Gzip}) \mid \textsc{Store} \ .$$

In concrete terms, the above scenario consists of the following four stages:

– The manager receives a request an for a data file from the external environment carrying a channel n which could be used for a negative response:

$$\begin{aligned}\{a,c,f\} \vdash \textsc{Dbase} &\xrightarrow{an} \\ &\{a,c,f,n\} \vdash (\boldsymbol{\nu} g)(\bar{f}g + \bar{n}c \mid gd.\bar{a}d) \mid (\boldsymbol{\nu} h)(fe.\bar{e}h) \ .\end{aligned} \tag{1}$$

  This move has been obtained using the following sequence of rules in table 1: In (where a previously unknown channel n is added to the indexing set), Par, Res and Par. Notice that the rule In could have been applied with b being substituted by any channel from the original indexing set; in such a case, the latter would have remained unchanged, e.g., $\ldots \xrightarrow{ac} \{a,c,f\} \vdash \ldots$.

– The manager can now either reply that the requested data is not available (using the $\bar{n}c$ branch) which is an option our scenario ignores, or initiate a positive response (using the other branch), as shown below:

$$\xrightarrow{\tau} \{a,c,f,n\} \vdash (\boldsymbol{\nu} m)(\mathsf{0} + \mathsf{0} \mid md.\bar{a}d \mid (\boldsymbol{\nu} h)(\bar{m}h)) \ . \tag{2}$$

  The above move is more complicated and it is built upon two sub-derivations. First, using the rules Out (after alpha-converting the bound channel g to a fresh channel m), Sum′, Par and Open, we get:

$$\{a,c,f,n\} \vdash (\boldsymbol{\nu} g)((\bar{f}g + \bar{n}c) \mid gd.\bar{a}d) \xrightarrow{\bar{f}m} \{a,c,f,n,m\} \vdash \mathsf{0} + \mathsf{0} \mid md.\bar{a}d \ .$$

Second, using the rule In followed by Res, we get:

$$\{a,c,f,n\} \vdash (\boldsymbol{\nu}h)(fe.\bar{e}h) \xrightarrow{fm} \{a,c,f,n,m\} \vdash (\boldsymbol{\nu}h)(\bar{m}h)\ .$$

The two derivations are then combined together using the Com rule. This exemplifies the *scope extrusion* which is a key concept of the π-calculus.

– Using the newly acquired restricted channel m, the store transfers the file h it held to the zipper:

$$\xrightarrow{\tau} \{a,c,f,n\} \vdash (\boldsymbol{\nu}m)(\mathsf{0}+\mathsf{0} \mid (\boldsymbol{\nu}h)(\bar{a}h \mid \mathsf{0}))\ . \tag{3}$$

This is another example of scope extrusion (for the restriction on h) and at the same time communication over the restricted channel m.

– The scenario concludes when the zipper sends off the compressed file to the external environment:

$$\xrightarrow{\bar{a}r} \{a,c,f,n,r\} \vdash (\boldsymbol{\nu}m)(\mathsf{0}+\mathsf{0} \mid \mathsf{0} \mid \mathsf{0})\ . \tag{4}$$

This step removes the restriction on h and adds r to the indexing set.

A Context-Based Representation of Indexed $\boldsymbol{\pi}$-Terms. The original syntactic definition of the π-calculus in table 1 is ill-suited for a compositional translation into the domain of Petri nets. For example, the scope of the restriction of a channel can change dynamically as a result of an application of the Open rule, and channels bound by input or restriction can be substituted by fresh ones. We address these problems by introducing an auxiliary representation in the form of indexed π-terms based on the separation of their static features (related to control flow) and dynamic features (related to channel substitution and channel binding).

The signature over which the context based representation is based is slightly richer than that of the original syntax. We assume that there are countably infinite disjoint sets of:

– *potentially known channels* $\mathbb{C}$ (as in the definition of the π-calculus, ranged over by the lower case letters, except u and v);
– *potentially restricted channels* $\mathbb{R}$ (ranged over by the upper case Greek letters);
– *channel holders* $\mathbb{H}$ (ranged over by the lower case Greek letters, except ς).

We then represent a well-formed indexed π-expression like $\{b,d\} \vdash ba.(\boldsymbol{\nu}c)\bar{a}c.\bar{c}b.0$ as a context based expression $\mathcal{C} \overset{\text{df}}{=} \mathcal{H}{:}\varsigma$, where $\mathcal{H} \overset{\text{df}}{=} \beta\alpha.\bar{\alpha}\gamma.\bar{\gamma}\beta.0$ is a restriction-free agent based solely on channel holders (the identity of the channel holders is irrelevant) and $\varsigma \overset{\text{df}}{=} [\beta \mapsto b, \delta \mapsto d, \gamma \mapsto \Delta]$ is a context allowing one to correctly interpret the channel holders. From the example ς, we may read that: α is presently a channel holder bound by an input (since α is not in the domain of the context mapping though is present in the expression $\mathcal{H}$); β and δ correspond respectively to the known channels b and d; and γ is a channel holder corresponding to the restricted channel Δ (again, the identity of this restricted channel is irrelevant).

In general, a *context* is a partial mapping $\varsigma : \mathbb{H} \to \mathbb{C} \cup \mathbb{R}$ with a finite domain. For each ς, we define:

$$\begin{array}{rcll} \mathit{known}(\varsigma) & \overset{\mathrm{df}}{=} & \varsigma(\mathbb{H}) \cap \mathbb{C} & \text{(known channels)} \\ \mathit{new}(\varsigma) & \overset{\mathrm{df}}{=} & \mathbb{C} \setminus \mathit{known}(\varsigma) & \text{(potentially known channels)} \\ \mathit{rstr}(\varsigma) & \overset{\mathrm{df}}{=} & \varsigma(\mathbb{H}) \cap \mathbb{R} & \text{(restricted channels).} \end{array}$$

And the syntax of the context based notation is defined as follows:

$$\begin{array}{llcl} \text{Prefixes} & p & ::= & \bar{\alpha}\beta \mid \alpha\beta \mid \tau \\ \text{Agents} & \mathcal{H} & ::= & \mathsf{0} \mid p.\mathcal{H} \mid \mathcal{H}+\mathcal{H} \mid \mathcal{H}|\mathcal{H} \\ \text{Expressions} & \mathcal{C} & ::= & \mathcal{H}{:}\varsigma \end{array}$$

A channel holder is *bound input* if it occurs in the second position of an input prefix. A context based expression $\mathcal{H}{:}\varsigma$ is *well-formed* if we have that: all the channel holders in $\mathcal{H}$ are uniquely used (no channel holder may be bound by more than one input prefix, and then its other usages are in the suffix of this prefix); and no bound input channel holder belongs to $\mathit{dom}(\varsigma)$ while all the remaining channel holders used in the expression do belong to $\mathit{dom}(\varsigma)$. In the following, we shall only consider well-formed expressions.

It is straightforward to translate a well-formed indexed π-expression $\mathcal{P} = A \vdash P$ into a corresponding context based one, $\mathcal{C} = \mathcal{H}{:}\varsigma$. First, for each channel c occurring in P and A, we introduce a unique channel holder α_c. Then we replace each occurrence of c within P by α_c, and delete all occurrences of the hiding operator, resulting in $\mathcal{H}$. The context mapping has as the domain all channel holders α_c such that c was not input bound in P, and is given by $\varsigma(\alpha_c) = c$ if c was not restricted, and $\varsigma(\alpha_c) = \Delta_c$ otherwise (we assume that $\Delta_c \neq \Delta_d$ for $c \neq d$). E.g., our running example can be rendered in the context-based representation as:

$$\textsc{Dbase}' \overset{\mathrm{df}}{=} (\textsc{Manager}' \mid \textsc{Gzip}' \mid \textsc{Store}') : \varsigma$$

where $\textsc{Manager}' \overset{\mathrm{df}}{=} \alpha\beta.(\bar{\gamma}\delta + \bar{\beta}\eta)$, $\textsc{Gzip}' \overset{\mathrm{df}}{=} \delta\phi.\bar{\alpha}\phi$ and $\textsc{Store}' \overset{\mathrm{df}}{=} \gamma\kappa.\bar{\kappa}\psi$ are the three processes expressed in the channel holder notation, and the context mapping is given by:

$$\varsigma \overset{\mathrm{df}}{=} [\alpha \mapsto a \ , \ \gamma \mapsto f \ , \ \delta \mapsto \Delta \ , \ \eta \mapsto c \ , \ \psi \mapsto \Omega] \ .$$

From the above we can read off the *known* (indexing) channels, $\mathit{known}(\varsigma) = \varsigma(\mathbb{H}) \cap \mathbb{C} = \{a, f, c\}$, and the *restricted* channels, $\mathit{rstr}(\varsigma) = \varsigma(\mathbb{H}) \cap \mathbb{R} = \{\Delta, \Omega\}$. Note that there is now no need to represent channel restriction within an expression, as in the original syntax, since the relevant information is conveyed by the context mapping.

Later on, we will see how the structure of a holder-based term yields a Petri net (with each channel holder being translated into a corresponding place), and how the initial marking of these places is derived using the context mapping.

3 An Algebra of Nets

Our target Petri net model, called *p-nets*, is inspired by the box algebra [2, 3, 11], designed with the aim of providing a compositional Petri net semantics of concurrent programming languages. In this paper, we shall modify the original model and, in particular, use coloured tokens together with suitably labelled transitions, arcs and read-arcs. The latter are a Petri net specific device allowing an arbitrary number of transitions to simultaneously check for the presence of a resource stored in a place [8].

Transitions in p-nets have three different kinds of labels:

- UV, Uv and $\bar{U}V$ to specify communication with the external environment;
- τ to represent an invisible action;
- uv and $\bar{u}v$ to effect synchronous interprocess interactions.

Places are also 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); the tokens they hold are the standard 'black' ones. *Holder* places carry coloured tokens representing channels; in the diagrams, their borders are thick, and they are labelled by the elements of $\mathbb{H}$. (A third kind of places will be introduced later on.)

Referring to figure 1, 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. For example, N_{out} and N_{in} may be seen as preliminary translations of two context-based expressions, $\bar{\alpha}\beta : \varsigma$ and $\alpha\gamma : \varsigma$, where $\varsigma \overset{\text{df}}{=} [\alpha \mapsto a, \beta \mapsto b]$ (note also that N_{out} and N_{in} correspond to the output and input prefixes, $\bar{a}b$ and ac, of the π-calculus). Furthermore, N_{com} represents the translation of $\bar{\alpha}\beta|\alpha\gamma : \varsigma$ (or $\bar{a}b|ac$ in the π-calculus syntax). The idea here is to represent a π-calculus channel a by a holder place labelled α, marked by a token coloured by the actual channel name replacing a. Initially, if a is known then the place will contain a single a-token; otherwise it will be empty (like the γ-labelled place) until a communication or an input prefix inserts a channel into it.[1]

In order to observe these mechanisms at work, consider the nets N_{out} and N_{in} in figure 1, each consisting of one transition, two interface places, and two holder places. Interface places are connected using the standard Petri net arcs, while holder places are connected using directed arcs and read arcs labelled with *channel variables*, u, U, v and V (note that, due to a strict naming discipline, no other channel variables will ever be needed). The initial marking on the holder places

[1] It is worth emphasising the importance of using read arcs in the proposed translation. In the interleaving semantics, each read arc in nets N_{out} and N_{in} could simply be replaced by a side loop (two arcs to and from the connected place and transition, with the same inscription). However, the transition in N_{com} coming from the synchronisation of N_{out} and N_{in}, would then not be executable because it would require *two* tokens a in the α-labelled place (whereas at most one is available).

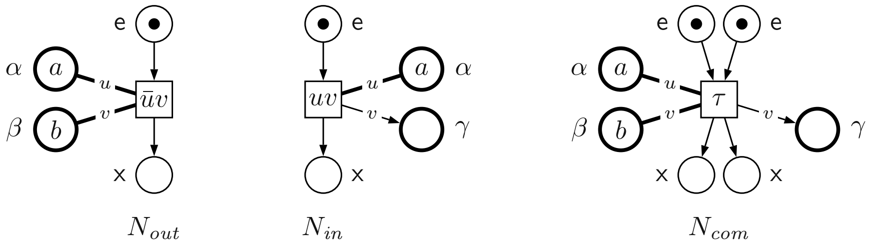

Fig. 1. Holder places and read arcs

is derived from the context ς which associates, in particular, a and b to the α-labelled and β-labelled holder places, respectively. (For control flow places, the default initial marking inserts a single black token into each entry place.)

The annotation of the various arcs by channel variables establishes a *binding* $\flat : \{u, U, v, V\} \rightarrow \mathbb{C}$ for the channels transferred/tested along arcs adjacent to a given transition. The transition of N_{in} is enabled if the entry place is non-empty and there exists a binding $\flat$ such that the α-labelled place contains at least one channel $\flat(u)$ (in our case, a). The execution of the transition transforms the marking in the following way: the black token is removed from the entry place and deposited in the exit one; the channel in the α-labelled place is left unchanged; and the channel $\flat(v)$ (e.g., channel e or channel b) is put into the γ-labelled holder place. With such a binding $\flat$, the firing generates a visible action $\flat(u)\flat(v)$ (e.g., ae or ab). Similarly, the firing of the transition in N_{out} generates the action $\bar{a}b$ (notice that no other visible action is possible for N_{out} as any binding enabling its only transition must satisfy $\flat(u) =$ and $\flat(v) = b$). Now, if we look at the firing of the τ-labelled transition in N_{com}, 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 in N_{out} (i.e., $\flat(v) = b$), and so the internal communication is effected through which the γ-labelled holder place acquires the channel b.

The third, and last, kind of node in a p-net is a special holder place, called the *tag-place*, which is always present (though it may well be disjoint from the rest of the net) and is indicated in the diagrams by a double border. The tokens stored in this place are coloured and structured by being *tagged* with a member of the set $\mathbb{T} \overset{\text{df}}{=} \{\mathsf{K}, \mathsf{N}, \mathsf{R}\}$; the place itself is $\mathbb{T}$-labelled. The first tag, K, will be used to indicate the channels in *known*(ς). The second tag, N, will be used to indicated the new (unknown) channels in *new*(ς). And the third tag, R, will be used to indicate the restricted channels in *rstr*(ς). The first case is slightly more complicated than the remaining two. For a restricted channel, say Δ, may be present in various holder places, due to synchronisation with various input prefixes. Then, if the restricted channel is *opened* (Δ becomes a newly known channel c), it is not possible to replace Δ by c in all the relevant holder places without some global transformation of the net. Instead, we shall indicate this fact by inserting a token $c.\Delta.\mathsf{K}$ in the tag-place, and then consulting it whenever

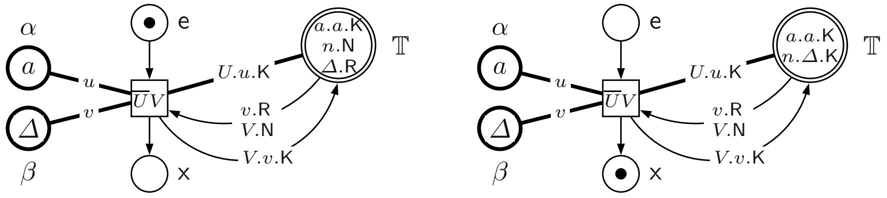

Fig. 2. Example of the usage of the tag-place in the net N_{res}

necessary (i.e., whenever we need to establish whether a restricted channel has been opened and what is the corresponding known value). In order to make the notation uniform, we shall use tokens $a.a.\mathsf{K}$ to denote all those known channels which were never restricted. To summarise, a token in the tag-place may be of the form:

- $a.\mathsf{N}$ meaning that a is a new channel;
- $\Delta.\mathsf{R}$ meaning that Δ is a restricted channel;
- $a.a.\mathsf{K}$ meaning that a is a known channel (either a has always been known or a was initially new and then became known);
- $a.\Delta.\mathsf{K}$ meaning that the restricted channel Δ has become known as a.

The arcs adjacent to the tag-place (both directed and read ones) are labelled with annotations of the form $U.u.\mathsf{K}$, $v.v.\mathsf{K}$, $V.v.\mathsf{K}$, $V.\mathsf{N}$, $v.\mathsf{N}$ or $v.\mathsf{R}$. For a given binding $\flat$, such annotations evaluate respectively to $\flat(U).\flat(u).\mathsf{K}$, $\flat(v).\flat(v).\mathsf{K}$, $\flat(V).\flat(v).\mathsf{K}$, $\flat(V).\mathsf{N}$, $\flat(v).\mathsf{N}$ and $\flat(v).\mathsf{R}$. Notice that the channel variables (U, V, u and v) will also be used in transition labels.

Consider the p-net N_{res} on the left in figure 2 (only some tokens in the tag-place are shown). Such a net gives a translation of a context based expression $\bar{\alpha}\beta$, assuming that $\alpha \mapsto a$ and $\beta \mapsto \Delta$ (in other words, of the π-calculus prefix $\bar{a}d$ when d is a restricted channel). The marking in the tag-place indicates that Δ is a restricted channel, n is an unknown channel and a a known one. The transition is enabled with the binding $\flat$ such that $\flat(u) = \flat(U) = a$, $\flat(v) = \Delta$ and $\flat(V) = n$. Its execution produces the visible action $\flat(\overline{U}V) = \bar{a}n$ and leads to the marking exhibited in the net on the right. This execution illustrates how a restricted channel becomes known (which is represented by the insertion of the token $n.\Delta.\mathsf{K}$ in the tag-place), and corresponds to the Open rule in table 1.

The p-net composition operators that we need are *prefixing*, $N.N'$, *choice* $N + N'$, and *parallel composition*, $N|N'$. All 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 [11], and will allow one to mimic the rewriting mechanism of the standard 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 places of N and N' are combined, and so are their exit places. For example, 'combining' the entry places means performing a cartesian product of these two sets, and connecting a new entry place (s, s') to a transition t from N (or from N') in the same way as it was connected to s in N (resp. s' in N'). This corresponds to the choice operation in the box algebra [3] and has the following effect: if we start from the initial marking (i.e., one black token is inserted into each entry place), then either N or N' can be executed, mimicking the Sum (or Sum$'$) rule and its symmetric counterpart.

When applying the prefixing operator, there is only one exit place in N which is combined with the entry places of N', all the resulting places becoming internal. This corresponds to the sequence operator in the box algebra, 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.

Finally, when composing N and N' in parallel, the p-nets are placed side by side and the pairs of transitions labelled uv and $\bar{u}v$ (one coming from N and the other from N') are synchronised, resulting in τ-labelled transitions. E.g., in figure 1, the net N_{com} can be derived as the parallel composition N_{out} and N_{in}, and the τ-labelled transition results from synchronisation of the two transitions labelled by uv and $\bar{u}v$ (which will be wiped out at the end of the translation). This, in particular, corresponds to the synchronisation operation in the M-net theory [2]. Putting the nets side by side allows to execute both parts in parallel, as in the Par rule and its symmetric counterpart, and the synchronisation of transition has effect similar to that of the Com rule.

4 Translating Context-Based Expressions into p-Nets

Given a context-based expression $\mathcal{C} \overset{\text{df}}{=} \mathcal{H} : \varsigma$, our translation into p-nets is done in two phases. First, we compositionally translate $\mathcal{H}$ into an unmarked p-net $\mathbb{K}(\mathcal{H})$ and then, using the context ς, we fill holder places with appropriate channels, set the default initial marking on the control flow places, and delete some transitions which are no longer needed. This results in the target p-net denoted by $\mathbb{PN}(\mathcal{C})$.

Phase I The translation $\mathbb{K}(\mathcal{H})$, guided by the syntax tree of $\mathcal{H}$, consists in first giving the translation for the basic sub-terms (i.e., $\mathsf{0}$ and the internal, input and output prefixes) shown in figure 3, and then applying p-net operators.

The basic process $\mathsf{0}$ is translated into a net consisting of a single entry place, a single exit place, and a single tag-place. As a result, we shall have up to isomorphism that $\mathbb{K}(N + \mathsf{0}) = \mathbb{K}(N) = \mathbb{K}(\mathsf{0} + N)$, and that $\mathbb{K}(N|\mathsf{0}) = \mathbb{K}(\mathsf{0}|N)$ only differ from $\mathbb{K}(N)$ by two isolated places and so have identical behaviours. Hence the standard π-calculus rules $N \equiv N + \mathsf{0} \equiv \mathsf{0} + N \equiv N|\mathsf{0} \equiv \mathsf{0}|N$ are also observed in the net model.

The translation of the internal prefix τ is very simple as it does not involve any manipulation on channels.

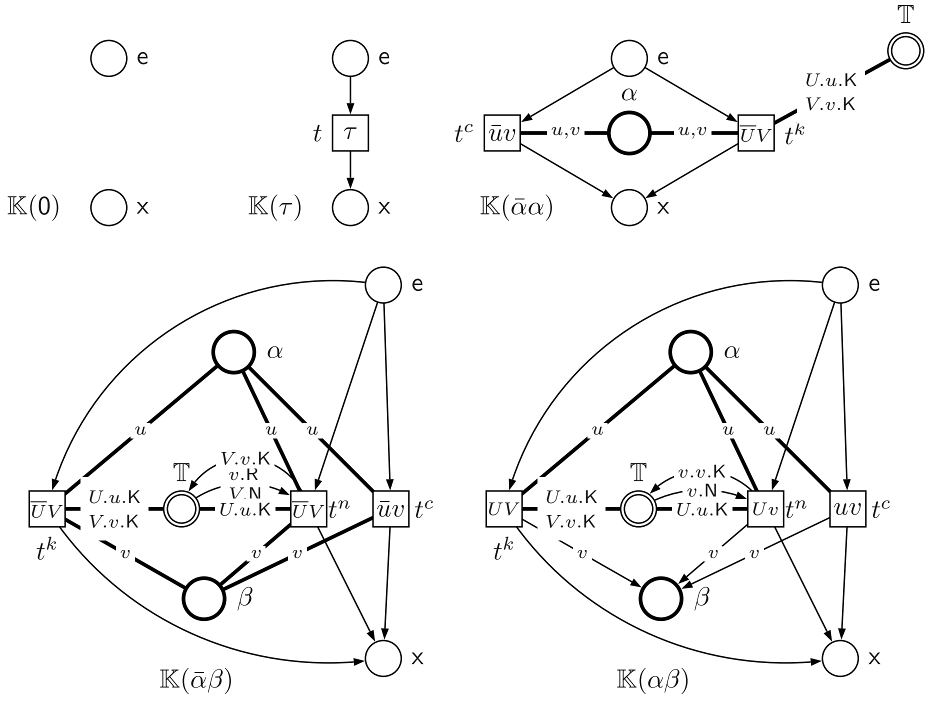

Fig. 3. The unmarked p-nets for 0 and the three kinds of prefixes (the tag-place is omitted when disconnected)

Each output prefix $\bar{\alpha}\beta$, when $\alpha \neq \beta$, is translated into the p-net $\mathbb{K}(\bar{\alpha}\beta)$ which may exhibit three kinds of behaviours, corresponding to the firing of three specific transitions:

- t^k: known output. A known channel (matching) V is sent through a channel (matching) U. The actual values of U and V are provided by the tokens present in the tag-place matching those in the holder places α and β, accessed through u and v. This corresponds to the Out rule. That the channels matching U and V are known is determined by the presence in the tag-place of tokens tagged with K. Notice that even if the entry place is marked, it may happen that this transition is not enabled; this may happen if α (and/or β) is unmarked (it is bound input and the binding prefix has not been executed yet), or it is marked by a restricted channel which has not been opened (yet).
- t^n: new output. A new channel V is sent through a known channel U. That the channels v and V are respectively restricted and new is determined by the presence in the tag-place of a channel tagged with R for v, and a channel tagged with N for V. After the execution of this transition, the restricted channel represented by v becomes known as the channel value of V; this is indicated by a token of the form $V.v.\mathsf{K}$ inserted into the tag-place to replace $v.\mathsf{R}$ and $V.\mathsf{N}$. This corresponds to the Open rule. Again, this transition may

be blocked (possibly temporarily), if α or β is unmarked, or if α is marked by a restricted and unopened channel, or if β is marked by a known or previously restricted but now opened channel.

– t^c: communicating output. It is intended to synchronize with a corresponding communication input in order to provide the transfer of a channel v through the channel u, be it known or restricted.

The special case of the output prefix $\bar{\alpha}\alpha$ has a simpler translation, since α may not be both known and restricted, so that t^n is unnecessary. Even though the α-labelled holder place will never contain more than one token, 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 u and v with the same token in the α-labelled place.

For an input prefix $\alpha\beta$, when $\alpha \neq \beta$, the translation is broadly the same as for the output prefix. In particular, the known, new and communicating inputs should be interpreted in a similar way. Simply, v is now put into β instead of being read (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 v is known as V, it is the possibly restricted original value v which is written into β, and not the corresponding known value V. This is important in order to allow subsequent synchronisations between uv with the u coming from this place and $\bar{u}v$ with the u coming from another holder place and containing a copy of the original token. Finally, prefixes of the form $\alpha\alpha$ are excluded by the well-formedness assumption (no channel holder can be both known and bound input).

For the compound sub-terms, we proceed compositionally:

$$\begin{aligned}
\mathbb{K}(p.\mathcal{H}) &\overset{\text{df}}{=} \mathbb{K}(p).\mathbb{K}(\mathcal{H}) \\
\mathbb{K}(\mathcal{H} + \mathcal{H}') &\overset{\text{df}}{=} \mathbb{K}(\mathcal{H}) + \mathbb{K}(\mathcal{H}') \\
\mathbb{K}(\mathcal{H} \mid \mathcal{H}') &\overset{\text{df}}{=} \mathbb{K}(\mathcal{H}) \mid \mathbb{K}(\mathcal{H}')\,.
\end{aligned}$$

As an illustration of this phase of translation, we show in figure 4 the p-net obtained for STORE′. It is composed of the upper and lower parts corresponding respectively to the p-nets for input and output prefixes $\gamma.\kappa$ and $\kappa.\psi$ (the concatenation with the implicit trailing $\mathsf{0}$ has been omitted for simplicity without changing the essence of the overall picture). These parts have been composed using the prefixing operation which merged the exit place of the first p-net with the entry place of the second p-net (leading to an internal place), the holder places labelled κ from the first and second p-nets as well as their tag-places. The transitions labelled uv and $\bar{u}v$ are intended for synchronisation during subsequent parallel compositions and will be deleted in phase II.

Phase II Having derived $\mathbb{K}(\mathcal{H})$, we construct the target p-net by first removing all the transitions labelled by uv and $\bar{u}v$, and then inserting one black token into each entry place as well as the following channels into holder places:

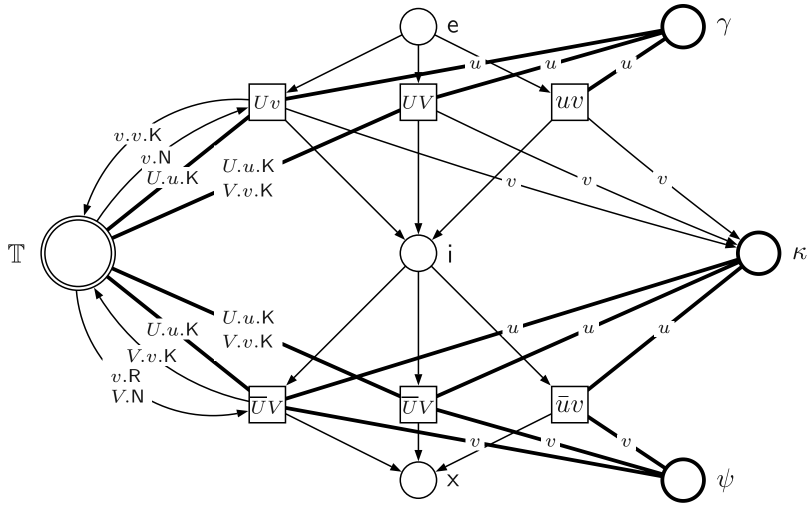

Fig. 4. The p-net resulting from phase I for STORE$' \overset{\mathrm{df}}{=} \gamma\kappa.\bar{\kappa}\psi$

- $\varsigma(\alpha)$ into each α-labelled holder place such that $\alpha \in \mathit{dom}(\varsigma)$.
- $a.a.\mathsf{K}$ into the tag-place for each $a \in \mathit{known}(\varsigma)$.
- $n.\mathsf{N}$ into the tag-place for each $n \in \mathbb{C} \setminus \mathit{known}(\varsigma)$.
- $\Delta.\mathsf{R}$ into the tag-place for each $\Delta \in \mathit{rstr}(\varsigma)$.

Phase II is illustrated in figure 5 which shows the part of p-net for the context-based expression DBASE$'$: ς corresponding to the scenario used in our running example (still omitting the trailing $\mathsf{0}$'s). The control-flow places form three vertical lines respectively corresponding to the control-flow places of the sub-expressions MANAGER$'$ (on the left), STORE$'$ (in the middle) and GZIP$'$ (on the right). The τ-labelled transitions come from the parallel composition (which includes synchronisation) between MANAGER$'$ and STORE$'$ (the upper one) and between STORE$'$ and GZIP$'$ (the lower one). All the entry places are marked, the tag-place and the holder places corresponding to the known or restricted channels are marked accordingly to the context ς.

At the initial marking, the Uv-labelled transition may be fired under a binding satisfying $\flat(u) = \flat(U) = a$ and $\flat(v) = n$ since:

- a is present in α-labelled holder place and $a.a.\mathsf{K}$ is present in the tag-place (which means that a is a known channel);
- $n.\mathsf{N}$ is present in the tag-place, which means that n is an unknown (fresh) channel.

The firing of this transition:

- consumes $\bullet$ from the input entry place and $n.\mathsf{N}$ from the tag-place;
- generates the visible action an;

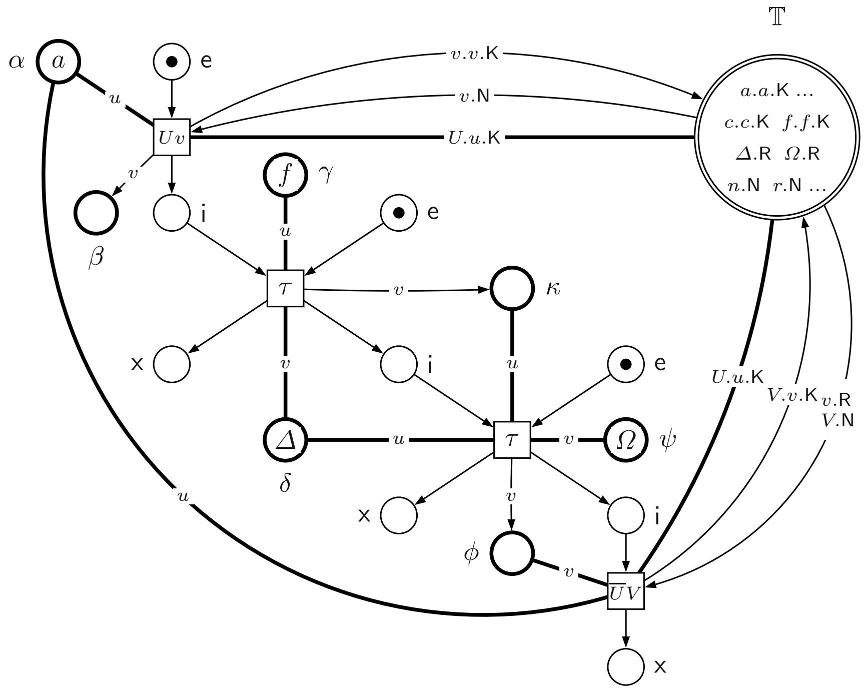

Fig. 5. The relevant fragment of the p-net resulting from phase II for the running example

– produces a black token in the internal output place, n in the β-labelled holder place and $n.n.$K in the tag-place (which means that n is now a known channel).

Next, the firing of the first τ-labelled transition is possible. It produces, in particular, the restricted Δ in the κ-labelled holder place which implements the scope extrusion of the π-calculus.

After that the second τ-labelled transition can be fired. It represents the transfer of the restricted channel Ω between Store′ and Gzip′ through the private (restricted) channel Δ. It results, in particular, in putting Ω in the ϕ-labelled holder place.

At this stage, the firing of the last transition is possible with a binding satisfying $\flat'(u) = \flat'(U) = a$, $\flat'(v) = \Omega$ and $\flat'(V) = r$. It generates the visible action $\bar{a}r$, and replaces $\Omega.$R and $r.$N with $r.\Omega.$K inside the tag-place. This means that Ω is not longer restricted and that r has became known.

Main Result. It turns out that the proposed translation is sound, in a rather strict sense, which is expressed by the following theorem, the proof of which may be found in the technical report version of this paper [9].

Theorem. *For every indexed π-expression $\mathcal{P}$, its labelled transition system $\mathsf{lts}_{\mathcal{P}}$ (considered up to alpha-conversion) is isomorphic to the labelled transition system $\mathsf{lts}_{\mathbb{PN}(\mathcal{C})}$ of the p-net $\mathbb{PN}(\mathcal{C})$, where $\mathcal{C}$ is any well-formed context based expression corresponding to $\mathcal{P}$.*

5 Related Work

A first paper dedicated to giving a Petri net semantics for the π-calculus appears to be [10]. 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 (in fact, eliminated by renaming the restricted channels by fresh ones).

While not based on nets, [4] 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). However, the authors only capture the first kind of causality in their semantic model, while we taken both of them into account.

A graph-rewriting system is proposed in [14] as a semantic model for a rather restricted fragment of the π-calculus. This approach mainly addresses the concurrency feature of systems whereas we concentrated here on their interleaving semantics, since our aim was to show that our translation leads to nets with the same sequential behaviour as that defined by the standard interleaving semantics of the π-calculus. One of our immediate goal is to look at concurrency issues, and in this respect we may notice a fundamental discrepancy between our approach and [14] in the handling of restriction. More precisely, [14] allows parallel opening for expressions like $(\nu y)(\bar{x}y|P|\bar{z}y)$ by letting the actions $\bar{x}y$ and $\bar{z}y$ to occur independently (in parallel), while in our approach they must in some sense agree on their common exportation, so that only one of them (chosen arbitrarily) is in fact an opening, the other one accepting the choice already made.

The most frequently cited translation of π-terms into Petri nets is probably [5], which 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. Indeed, the approach from [5] proceeds by first establishing a general infrastructure, with places corresponding to all the possible sequential π-terms decorated with some conflicting set, and to all possible restrictions and conflicts, with all possible transitions between those places. It then defines compositionally the initial marking corresponding to each π-term (here, guarded recursions are allowed). As a consequence, even for a very simple process expression $\tau.\mathsf{0}$, this leads to a net with infinitely many places and transitions (but only a single token in a single place).

6 Conclusions

In this paper, we outlined a Petri net based translation of the finite π-calculus, as well as an intermediate process algebra. Both approaches are based on the notions of a channel holder and context. Our development has been motivated by a desire to provide a compositional translation of the original π-calculus to the domain of automata-based models of computation (in particular, Petri nets). We therefore needed to address problems relating to the fact that a number of fundamental features of the former are based on purely language theoretic concepts (such as alpha-conversion and channel restriction). One of possible practical applications of our translation will be in making it possible to use Petri net based verification techniques to prove properties of mobility systems.

We now plan to investigate the extension of the theory to infinite behaviours through replication or recursion, as in the full π-calculus, or through iteration, as in the box algebra.

Acknowledgement

We would like to thank the anonymous referees for their helpful comments.

References

[1] E.Best and R.Devillers: Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* 55 (1988) 87–136.

[2] E.Best, W.Frączak, R. P.Hopkins, H.Klaudel and E.Pelz: 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. 315, 318

[3] E.Best, R.Devillers and M.Koutny: *Petri Net Algebra.* EATCS Monographs on TCS, Springer (2001). 315, 318

[4] M.Boreale and D.Sangiorgi: A Fully Abstract Semantics for Causality in the π-calculus. Proc. of *STACS'95*, Springer, LNCS 900 (1995) 243–254. 323

[5] N.Busi and R.Gorrieri: A Petri net Semantics for π-calculus. Proc. of *Concur'95*, LNCS 962 (1995) 145–159. 323

[6] G. L.Cattani and P.Sewell: Models for Name-Passing Processes: Interleaving and Causal. Proc. of *LICS'2000*, IEEE CS Press (2000) 322–333. 310, 311

[7] G. L.Cattani and P.Sewell: Models for Name-Passing Processes: Interleaving and Causal. Technical Report TR-505, University of Cambridge (2000). 310, 311

[8] S.Christensen and N. D.Hansen: Coloured Petri Nets Extended with Place Capacities, Test Arcs and Inhibitor Arcs. Proc. of *ICATPN'93*, Springer, LNCS 691 (1993) 186–205. 315

[9] R.Devillers, H.Klaudel and M.Koutny: Petri net semantics of the finite π-calculus CS-TR-846 University of Newcastle 2004 322

[10] J.Engelfriet: A Multiset Semantics for the π-calculus with Replication. *Theoretical Computer Science* 153 (1996) 65–94. 323

[11] H.Klaudel and F.Pommereau: Asynchronous links in the PBC and M-nets. Proc. of *ASIAN'99*, Springer, LNCS 1742 (1999) 190–200. 315, 317

[12] R.Milner: *Communication and Concurrency.* Prentice Hall (1989).

[13] R.Milner, J.Parrow and D.Walker: A Calculus of Mobile Processes. *Information and Computation* 100 (1992) 1–77. 309, 310

[14] U.Montanari and M.Pistore: Concurrent Semantics for the π-calculus. Proc. of *MFPS'95*, Elsevier (1995) Electronic Notes in Computer Science 1. 323

[15] J.Parrow: An Introduction to the π-calculus. In: *Handbook of Process Algebra*, Bergstra, Ponse, Smolka (Eds.). Elsevier (2001) 479–543. 310

[16] G. D.Plotkin: A Structural Approach to Operational Semantics. Technical Report FN-19, Computer Science Department, University of Aarhus (1981).

[17] W.Vogler: Partial Order Semantics and Read Arcs. Proc. of *MFCS'97*, Springer, LNCS 1295 (1997) 508–517.