A stochastic π-calculus semantics for PEPA nets

Linda Brodo [1]

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy

Stephen Gilmore [2]

Laboratory for Foundations of Computer Science
The University of Edinburgh
Edinburgh, Scotland

Jane Hillston [3]

Laboratory for Foundations of Computer Science
The University of Edinburgh
Edinburgh, Scotland

Corrado Priami [4]

Dipartimento di Informatica e Telecomunicazioni
Università di Trento
Povo, Italy

Abstract

We provide a semantics for a subset of the PEPA nets modelling language by mapping it into the stochastic π-calculus. The mapping provides an opportunity to contrast the well-known small-step operational semantics style with the definition induced by language embedding and to study the definitions of equivalences between models in the two styles.

[1] Email: `brodo@sci.univr.it`
[2] Email: `stg@dcs.ed.ac.uk`. Corresponding author.
[3] Email: `jeh@dcs.ed.ac.uk`
[4] Email: `priami@dit.unitn.it`

Preprint submitted to Elsevier Preprint *2 June 2003*

1 Introduction

The language of *PEPA nets* [1] is a performance modelling formalism which allows the modeller to describe a high-level performance model of a system as a Petri net where the tokens are model components of Hillston's PEPA stochastic process algebra [2]. PEPA nets evolve either by (local) *transitions* within a place or by (global) *firings* of the net, transferring tokens from one place to another. Together these two forms of evolution of the net describe a continuous-time Markov chain (CTMC) which can be solved to find the stationary probability distribution of the system. Performance measures on the system can be expressed in terms of this stationary probability distribution.

Moving stateful processes between the places in the net is used in modelling systems which have both a degree of process mobility and also the capacity to dynamically reconfigure the communication topology of the system. This points to a relationship with the canonical mobile stochastic process algebra, Priami's *Stochastic π-calculus* [3]. The present paper explores the relationship between these two formalisms.

If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets and the stochastic π-calculus would be considered equally expressive. That is to say, for a given CTMC C, it is possible to construct a high-level model in either formalism such that the underlying CTMC derived from the model is isomporphic to C.

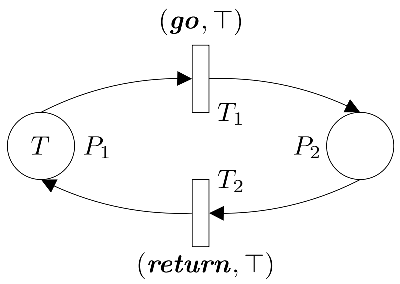
This is a fundamental agreement in expressive power, but it is a rather weak one, similar to the agreement that all programming languages are Turing-complete. Here we seek instead to relate these two languages via generating an encoding from PEPA nets into the stochastic π-calculus such that the model and its encoding would generate isomorphic CTMCs when evaluated against their respective language semantics. In general these isomorphic CTMCs would be derived from different intermediate labelled transition systems.

$$
\begin{array}{ccc}
Net & \leadsto & S\pi \\
\downarrow & & \downarrow \\
LTS_1 & & LTS_2 \\
\downarrow & & \downarrow \\
CTMC_1 & \equiv & CTMC_2
\end{array}
$$

The encoding which we seek is a *compositional* one, which does not trivially expand the given PEPA net to its full state space and then translate this into the stochastic π-calculus. Formally, the size of the stochastic π-calculus description which we generate must be proportional to the size of the description of the input PEPA net, not proportional to the size of its state space. In interleaving models of concurrent systems the size of the state space of a model can be exponentially larger than the size of the model itself so the difference between these two views is significant.

Because of differences in the modelling constructs which are provided by the two languages, it is the case in general that neither of the languages can be encoded into the other in a full compositional fashion. In the case of mapping the language of PEPA nets into the stochastic π-calculus there is no suitable encoding of the PEPA multi-way synchronisation into the two-way handshake synchronisation of the stochastic π-calculus. In the case of mapping the stochastic π-calculus into PEPA nets there is no suitable representation of channel-passing in PEPA nets. For these reasons we aim only to encode a subset of the language of PEPA nets into the stochastic π-calculus.

We illustrate the idea of mapping a PEPA net into the stochastic π-calculus with a simple example. Consider a net with two places and a single token which circulates between the places. The token is initially in place P_1. At each place there is a switch which controls a light which is either on or off (initially off). The token circulates around the net flipping the switch, moving to the other place, and then repeating this behaviour.



The description of the token and the places of the net as a PEPA net is below.

$$\begin{aligned}
\mathit{Token} &\stackrel{\mathit{def}}{=} (\mathit{on}, r_1).(\boldsymbol{go}, r_2).(\mathit{on}, r_3).(\boldsymbol{return}, r_4).\\
&\qquad (\mathit{off}, r_5).(\boldsymbol{go}, r_6).(\mathit{off}, r_7).(\boldsymbol{return}, r_8).\mathit{Token}
\end{aligned}$$

$$\mathit{Switch} \stackrel{\mathit{def}}{=} (\mathit{on}, \top).(\mathit{off}, \top).\mathit{Switch}$$

$$\begin{aligned}
P_1[t] &\stackrel{\mathit{def}}{=} \mathit{Switch} \underset{\{\mathit{on},\mathit{off}\}}{\bowtie} \mathit{Token}[t]\\
P_2[t] &\stackrel{\mathit{def}}{=} \mathit{Switch} \underset{\{\mathit{on},\mathit{off}\}}{\bowtie} \mathit{Token}[t]
\end{aligned}$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1[\mathit{Token}], P_2[\_])$$

We now present this example in the stochastic π-calculus. The translation of the *Token* component of the PEPA net gives rise to two subterms which describe, respectively, the token's behaviour and its movement.

1. $\textbf{rec } \mathit{Token}.(\overline{\mathit{on}@11}, r_1).(\boldsymbol{go}, r_2).(\overline{\mathit{on}@21}, r_3).(\boldsymbol{return}, r_4).$
   $(\overline{\mathit{off}@11}, r_5).(\boldsymbol{go}, r_6).(\overline{\mathit{off}@21}, r_7).(\boldsymbol{return}, r_8).\mathit{Token}$
2. $\textbf{rec } \mathit{TokenMob}.(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).\mathit{TokenMob}$

The translations of the two places of the net give rise to two subterms which describe the static components resident at the two places (the two copies of the Switch). The activity names of the switch have been renamed to allow them to be distinguished.

1. $\textbf{rec } P1. \textbf{ rec } \mathit{Switch}.(\mathit{on}@11, 1).(\mathit{off}@11, 1).\mathit{Switch}$
2. $\textbf{rec } P2. \textbf{ rec } \mathit{Switch}.(\mathit{on}@21, 1).(\mathit{off}@21, 1).\mathit{Switch}$

The complete system is built by composing these four subterms and restricting on their shared actions, as shown below:

$$
\begin{aligned}
\mathit{System} \stackrel{\mathit{def}}{=} (\nu\ & \mathit{on}@11, \mathit{off}@11, \mathit{on}@21, \mathit{off}@21) \\
& ((\nu\ \boldsymbol{go}, \boldsymbol{return}) \\
& \quad (\textbf{rec } \mathit{TokenMob}\ .(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1). \\
& \qquad \mathit{TokenMob} \\
& \quad | \\
& \quad \textbf{rec } \mathit{Token}\ .(\overline{\mathit{on}@11}, r_1).(\boldsymbol{go}, r_2).(\overline{\mathit{on}@21}, r_3).(\boldsymbol{return}, r_4). \\
& \qquad (\overline{\mathit{off}@11}, r_5).(\boldsymbol{go}, r_6).(\overline{\mathit{off}@21}, r_7).(\boldsymbol{return}, r_8).\mathit{Token} \\
& \quad) \\
& | \textbf{ rec } P1. \textbf{ rec } \mathit{Switch}.(\mathit{on}@11, 1).(\mathit{off}@11, 1).\mathit{Switch} \\
& | \textbf{ rec } P2. \textbf{ rec } \mathit{Switch}.(\mathit{on}@21, 1).(\mathit{off}@21, 1).\mathit{Switch} \\
&)
\end{aligned}
$$

Both of these models define a simple eight-state system which is viewed as a CTMC as below.

$$
\begin{array}{ccccccc}
(\bullet^+, \bullet) & \xrightarrow{r_1} & (\circ^+, \bullet) & \xrightarrow{r_2} & (\circ, \bullet^+) & \xrightarrow{r_3} & (\circ, \circ^+) \\
{}_{r_8}\uparrow & & & & & & \downarrow{}_{r_4} \\
(\bullet, \bullet^+) & \xleftarrow{r_7} & (\bullet, \circ^+) & \xleftarrow{r_6} & (\bullet^+, \circ) & \xleftarrow{r_5} & (\circ^+, \circ)
\end{array}
$$

2 Translation

We restrict the class of nets which we will translate to those which do not use multi-way synchronisation. The PEPA stochastic process algebra allows multi-way synchronisation but this is not present in the stochastic π-calculus.

We now describe the translation of a PEPA net into an equivalent stochastic π-calculus model. We highlight the following points:

Tokens: The tokens in a PEPA net use *names* whose interpretation will depend upon the location of the token and the scope of the cooperation sets it is within. There are two types of activities which need to be translated, those which cause a local transition of the net (with no movement of a token) and those which cause a global firing of the net (and a token moves from an input place to an output place).

The translation of a token in a PEPA net gives rise to two tightly-coupled processes in the resulting stochastic π-calculus definitions. One of these encodes the behaviour of the token (say, that it will perform α first and then β) and the other encodes information about the static structure of the net (say, that there is an α-labelled transition between place P_1 and P_2). Each token is actually composed with a subnet of the entire net, specialised to only those places which are reachable by a token of this type (that is, with this alphabet).

Static components: By their nature, static components do not perform firing activities. Their translation mirrors the translation of transition activities in tokens.

Places: There are two aspects to the translation of each place in the net:

(i) the instantiation of the static components of the place; and

(ii) the representation of the arcs from this place to other places.

In undertaking the first of these we specialise local transition names. This differentiates each copy of a component from other copies of this component at other places.

The second of these produces a collection of process definitions where each process performs only firing activities. These definitions mirror the structure of the net, with the representation of the places and the labelled net transitions between them.

The initial marking: The initial marking is formed by making the right choice of the names for the parameters of the tokens of the net and by composing in parallel each token with the definition of the place where it is initially, restricting on the names of the transition activities.

Finally, the remaining free names of the system are restricted.

3 Encoding function

In this section we will define a function to encode a PEPA net (*Pnet*) process into a Stochastic π-calculus ($S\pi$) process.

A *Pnet* is defined by the tuple $\langle S, T, F, (D, M)\rangle$, where S is the set of places, T is the set of transition names, $F \subseteq S \times (T \times \mathbb{R}) \times S$, is the flow relation between places, D the set of *Pnet* definitions and M is the initial marking of the net. The encoding function $\mathcal{E} : Pnet \rightarrow S\pi$ is expressed by:

$$\mathcal{E}_\sigma[\![M]\!] = P_\pi$$

where $\sigma = \langle S, T, F, D\rangle$ stores information about the *Pnet*.

As all the place-processes and all the token-processes in a *Pnet* are distinct, we enumerate them and we will use this enumeration in referring to them. As outlined above, the mapping consists of three distinct maps translating static components, net-level structures and token components respectively.

$$\begin{aligned}&\mathcal{E}_\sigma[\![(P_1[A_1^1 \ldots, A_m^1], \ldots, P_N[A_{m+n}^N, \ldots, A_M^N])]\!] = \\ &\quad (\nu\ names)\ ((\Pi_{i=1}^N\ P_{\pi i})\ \mid\ \Pi_{j=1}^M\ (\nu\ names_j)(M_{\pi j}\ \mid\ T_{\pi j}))\end{aligned}$$

$$\begin{aligned}&\text{where } P_{\pi i} = \mathcal{E}S_\sigma^i[\![P_i]\!], M_{\pi j} = \mathcal{E}M_\sigma^i[\![A_j^i]\!], T_{\pi j} = \mathcal{E}T_\sigma^{(i,j)}[\![A_j^i]\!], names_j = fn_p(M_{\pi j}),\\ &\text{and } names = \{\alpha@ij \mid \alpha \in \mathsf{spec}\ A_j^k\ P_i\ \emptyset\ \ \forall i,k \in [1 \ldots N], \forall j \in [1 \ldots M]\} \cup \\ &\qquad\qquad \{\alpha@ij_1j_2 \mid \alpha \in \mathsf{spec}_{j_2}\ A_{j_1}^k\ P_i\ \emptyset\ \ \forall i,k \in [1 \ldots N], \forall j_1, j_2 \in [1 \ldots M]\}\end{aligned}$$

Some explanation of the above notation is required: the index i represents the ith place in a net and the index j represents the jth token.

The format which we use to translate *Pnet* name p is $p@n$, $n \in \mathbb{N}$ where the numeric suffix uniquely encodes two facts about the name. It records both the position of the name (in which place the name is used) and its use (who uses the name).

The function $\mathcal{E}S_\sigma^i[\![\_]\!] : \mathbb{N} \times Pnet \rightarrow S\pi$ translates the static components of the ith place; it is applied at each place of the PEPA net and gives as result the translation of the components of the places which are not tokens.

The function $\mathcal{E}M_\sigma^i[\![\_]\!] : \mathbb{N} \times Pnet \rightarrow S\pi$ translates only the firing actions of the jth token. It is applied to each token and gives as result a stochastic π-calculus process which synchronizes with the $\mathcal{E}T_\sigma$-translation of that token to allow it to move between places.

The function $\mathcal{E}T_\sigma^{(i,j)}[\![\_]\!] : \mathbb{N} \times \mathbb{N} \times Pnet \rightarrow S\pi$ translates all the transition actions of the jth token in the ith place.

The definitions of the functions $\mathcal{E}S$, $\mathcal{E}M$ and $\mathcal{E}T$ are given in Figures 1, 2 and 3, respectively. In Figure A.3 there is the definition of function spec, which calculates all the synchronisation activities a token-process uses in co-

$$
\begin{aligned}
\mathcal{E}S^i_\sigma[\![P \underset{L}{\bowtie} R]\!] &= (\nu\ L')(\mathcal{E}S^i_\sigma[\![P]\!] \mid \mathcal{E}S^i_\sigma[\![R]\!]) \\
&\qquad \text{where } L' = \{x@i \mid x \in (L \backslash \textstyle\bigcup_{j=1}^{M} \mathsf{spec}\ A^i_j\ P_i\ \emptyset)\} \\
\mathcal{E}S^i_\sigma[\![P/L]\!] &= \mathcal{E}S^i_\sigma[\![P]\!]\{^{x@new()}/_{x@i}\}\ \forall\ x \in L \\
\mathcal{E}S^i_\sigma[\![I[C]]\!] &= \mathbf{0} \\
\mathcal{E}S^i_\sigma[\![S_1 + S_2]\!] &= \mathcal{E}S^i_\sigma[\![S_1]\!] + \mathcal{E}S^i_\sigma[\![S_2]\!] \\
\mathcal{E}S^i_{\langle S,T,F,D\rangle}[\![I]\!] &= \begin{cases} \mathbf{rec}\ I.\mathcal{E}S^i_{\langle S,T,F,D \backslash \{(I \overset{def}{=} P)\}\rangle}[\![P]\!] & \text{if } (I \overset{def}{=} P) \in D \\ I & \text{otherwise} \end{cases} \\
\mathcal{E}S^i_\sigma[\![(\alpha, r).S]\!] &= \begin{cases} (\beta, 1).\mathcal{E}S^i_\sigma[\![S]\!] & \text{if } r = \top \\ (\overline{\beta}, r).\mathcal{E}S^i_\sigma[\![S]\!] & \text{otherwise} \end{cases} \\
&\qquad \text{where } \beta = \begin{cases} \alpha@ij \text{ if } \exists\ j \text{ s.t. } \alpha \in \mathsf{spec}\ P_i\ A^n_j\ \emptyset \\ \alpha@i \quad \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 1. Definition of $\mathcal{E}S$ function where P, R, S, are expressions, I is an identifier.

operations with static components of a place-process. In the same figure there is the definition of function spec_j, which calculates all the synchronisation activities a token-process uses in cooperations with the jth token-process in a specific place-process. In defining encoding functions we will use the function $new()$ which returns each time a fresh number.

$$
\begin{aligned}
\mathcal{E}M^i_\sigma[\![P \underset{L}{\bowtie} R]\!] &= (\nu\ L')(\mathcal{E}M^i_\sigma[\![P]\!] \mid \mathcal{E}M^i_\sigma[\![R]\!]), \quad L' = \{x@i \mid x \in L\} \\
\mathcal{E}M^i_\sigma[\![P/L]\!] &= \mathcal{E}P^i_\sigma[\![P]\!]\{^{x@new()}/_{x@i}\}\ \forall\ x \in L \\
\mathcal{E}M^i_\sigma[\![S_1 + S_2]\!] &= \mathcal{E}M^i_\sigma[\![S_1]\!] + \mathcal{E}M^i_\sigma[\![S_2]\!]
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{E}M^i_{\langle S,T,F,D\rangle}[\![I]\!] &= \begin{cases} \mathbf{rec}\ IMob.\mathcal{E}T^{(i,j)}_{\langle S,T,F,D \backslash \{(I \overset{def}{=} P)\}\rangle}[\![P]\!] \text{ if } (I \overset{def}{=} P) \in D \\ IMob \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \\
\mathcal{E}M^i_\sigma[\![(\alpha, r).S]\!] &= \begin{cases} \sum_{h=1}^{H}(\overline{\alpha}\ , 1).\mathcal{E}M^{z_h}_\sigma[\![S]\!] & \text{if } \alpha \in T \text{ with} \\ + & (i, (\alpha, \top), z_1) \ldots (i, (\alpha, \top), z_H) \in F, \\ \sum_{k=1}^{K}(\overline{\alpha}\ , \lambda_k).\mathcal{E}M^{z_k}_\sigma[\![S]\!] & (i, (\alpha, \lambda_1), z_1) \ldots (i, (\alpha, \lambda_K), z_K) \in F, \\ \mathcal{E}M^i_\sigma[\![S]\!] & \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 2. Definition of $\mathcal{E}M$ function where P, R, S, are expressions, I is an identifier.

$$
\begin{aligned}
\mathcal{E}T_\sigma^{(i,j)}[\![P \underset{L}{\bowtie} R]\!] &= (\nu\ L')(\mathcal{E}T_\sigma^{(i,j)}[\![P]\!] \mid \mathcal{E}T_\sigma^{(i,j)}[\![R]\!]),\ \ L' = \{x@Aj \mid x \in L\} \\
\mathcal{E}T_\sigma^{(i,j)}[\![P/L]\!] &= \mathcal{E}T_\sigma^{(i,j)}[\![P]\!]\{^{x@new()}/_{xi}\}\ \forall\ x \in L \\
\mathcal{E}T_\sigma^{(i,j)}[\![S_1 + S_2]\!] &= \mathcal{E}T_\sigma^{(i,j)}[\![S_1]\!] + \mathcal{E}T_\sigma^{(i,j)}[\![S_2]\!] \\
\mathcal{E}T_{\langle S,T,F,D\rangle}^{(i,j)}[\![I]\!] &= \begin{cases} \mathbf{rec}\ I.\mathcal{E}T^{(i,j)}_{\langle S,T,F,D\setminus\{(I \overset{def}{=} P)\}\rangle}[\![P]\!] & \text{if } (I \overset{def}{=} P) \in D \\ I & \text{otherwise} \end{cases} \\
\mathcal{E}T_\sigma^{(i,j)}[\![(\alpha,r).S]\!] &= \begin{cases} & \text{if } \alpha \in T \text{ with} \\ \sum_{h=1}^{H+K}(\alpha,r).\mathcal{E}T_\sigma^{(z_h,j)}[\![S]\!] & (i,(\alpha,\top),z_1)\dots(i,(\alpha,\top),z_H) \in F, \\ & (i,(\alpha,\lambda_1),z_1)\dots(i,(\alpha,\lambda_K),z_K) \in F, \\ (\overline{\beta},r).\mathcal{E}T_\sigma^{(i,j)}[\![S]\!] & \text{if } \alpha \notin T \text{ and } r \neq \top \\ (\beta,1).\mathcal{E}T_\sigma^{(i,j)}[\![S]\!] & \text{if } \alpha \notin T \text{ and } r = \top \end{cases}
\end{aligned}
$$

$$
\text{where } \beta = \begin{cases} \alpha@ij & \text{if } \alpha \in \mathsf{spec}\ P_i\ A_j^n\ \emptyset \\ \alpha@ijt & \text{if } \exists t \in [1\dots M] \text{ s.t. } \alpha \in \mathsf{spec}_t\ P_i\ A_j^n\ \emptyset \text{ and } j < t \\ \alpha@0j & \text{otherwise} \end{cases}
$$

Fig. 3. Definition of $\mathcal{E}T$ function where P, R, S, are expressions, I is an identifier.

4 Example

We illustrate the application of the above translation to the following PEPA net modelling a mobile software agent as described in [1].

$$
\begin{aligned}
& S = \{1,2,3\}, \qquad T = \{\boldsymbol{go}, \boldsymbol{return}\}, \\
& F = \{(1,\boldsymbol{return},2),(2,\boldsymbol{go},1),(2,\boldsymbol{go},3),(3,\boldsymbol{return},2)\}, \\
& D = \{P_1[A] \overset{def}{=} Agent[A] \underset{\{interrogate\}}{\bowtie} Probe, \\
& \qquad P_2[A] \overset{def}{=} Agent[A] \underset{\{dump\}}{\bowtie} Master, \\
& \qquad P_3[A] \overset{def}{=} Agent[A] \underset{\{interrogate\}}{\bowtie} Probe, \\
& \qquad Probe \overset{def}{=} (monitor, r_m).Probe + (interrogate, \top).Probe \\
& \qquad Master \overset{def}{=} (dump, \top).(analyse, r_a).Master \\
& \qquad Agent[A] \overset{def}{=} (\boldsymbol{go}.\lambda).(interrogate, r_i).(\boldsymbol{return}, \mu).(dump, r_d).Agent\}, \\
& M = (P1[\_], P_2[A_1^2], P_3[\_]).
\end{aligned}
$$

The result of the encoding function $\mathcal{E}_{\sigma=\langle S,T,F,D\rangle}[\![M]\!]$ is the $S\pi$-calculus process

$$
\begin{aligned}
P_\pi = \ & (\nu\ interrogate@31, dump@21, interrogate@11) \\
& (recP1 \mid recP2 \mid recP3 \mid \\
& \ (\nu\ \boldsymbol{go}, \boldsymbol{return})\ (recAgent \mid recAgentMob) \\
&)
\end{aligned}
$$

where $recP1$, $recP2$ and $recP3$ are the translations of the static components resident at the places of the net as follows:

$$
\begin{aligned}
recP1 = \mathcal{E}S^1_\sigma[\![P_1]\!] \ &= \ \textbf{rec}\ P1.\textbf{rec}\ Probe.\quad (\overline{monitor@1}, r_m).Probe + \\
&\qquad\qquad\qquad\qquad\qquad (interrogate@11, 1).Probe
\end{aligned}
$$

$$
\begin{aligned}
recP2 = \mathcal{E}S^2_\sigma[\![P_2]\!] \ &= \ \textbf{rec}\ P2.\textbf{rec}\ Master.\ (\overline{dump@21}, 1). \\
&\qquad\qquad\qquad\qquad\qquad (analyze@2, r_a).Master
\end{aligned}
$$

$$
\begin{aligned}
recP3 = \mathcal{E}S^3_\sigma[\![P_3]\!] \ &= \ \textbf{rec}\ P3.\textbf{rec}\ Probe.\quad (\overline{monitor@3}, r_m).Probe + \\
&\qquad\qquad\qquad\qquad\qquad (\overline{interrogate@31}, 1).Probe
\end{aligned}
$$

The definitions $recAgent$ and $recAgentMob$ are the translation of the token of the net.

$$
\begin{aligned}
recAgentMob = \mathcal{E}M^2_\sigma[\![Agent]\!] \ &= \textbf{rec}\ AgentMob. \\
&\quad ((\overline{\boldsymbol{go}}, \lambda_l).(\overline{\boldsymbol{return}}, \mu_l).AgentMob + \\
&\quad (\overline{\boldsymbol{go}}, \lambda_r).(\overline{\boldsymbol{return}}, \mu_r).AgentMob)
\end{aligned}
$$

$$
\begin{aligned}
recAgent = \mathcal{E}T^{(2,1)}_\sigma[\![Agent]\!] \ &= \textbf{rec}\ Agent. \\
&\quad \big((\boldsymbol{go}, \lambda).(interrogate@11, r_i). \\
&\quad\ (\boldsymbol{return}, \mu).(dump@21, r_d).Agent + \\
&\quad\ (\boldsymbol{go}, \lambda).(interrogate@31, r_i). \\
&\quad\ (\boldsymbol{return}, \mu).(dump@21, r_d).Agent\big)
\end{aligned}
$$

The restricted names are derived by the following computations:
$\{interrogate@31, dump@21, interrogate@11\} = \{\alpha@ij \mid \alpha \in \mathsf{spec}\ A^2_1\ P_i\ \emptyset\}$
$\{\boldsymbol{go}, \boldsymbol{return}\} = fn_p(recAgentMob)$

5 Equivalence

Building on the preceeding definitions, we are now able to provide a formal definition of equivalence between PEPA net processes and their stochastic π-calculus translations.

Definition 5.1 *Given a Pnet process $P_n = \langle S, T, F, (D, M)\rangle$ and a $S\pi$ process P_π, we say that P_n is equivalent to P_π, written as $P_n \sim P_\pi$, iff*

if $P_n \xrightarrow{(\alpha,r)} P'_n$ then $\exists\theta$ s.t. $P_\pi \xrightarrow{\theta} P'_\pi$ where $\ell(\theta) = (\alpha, r)$ and $P'_n \sim P'_\pi$

and

if $P_\pi \xrightarrow{\theta} P'_\pi$ then $\exists\alpha, r$ s.t. $P_n \xrightarrow{(\alpha,r)} P'_n$ where $\ell(\theta) = (\alpha, r)$ and $P'_n \sim P'_\pi$

This definition utilises a relabelling function which is similar to others used in other work in the enhanced operational semantics style [4].

Definition 5.2 *The function $\ell : \Theta \times \mathcal{N} \to \mathcal{N} \times \mathbb{R}$ returns the names of the channel used by the action together with its rate:*

$$
\begin{array}{llll}
\ell((x,r),J) & = (\beta, rate) & \ell((\overline{x},r),J) & = (\beta, rate) \\
\ell((\overline{x}(),r),J) & = (\beta, rate) & \ell((\tau,r),J) & = (\tau, rate) \\
\ell((\nu L)\vartheta\mu, J) & = \ell(\vartheta\mu, J \cup L) & \ell(\|_i\, \vartheta\mu, J) & = \ell(\vartheta\mu, J) \\
\ell(+_i\vartheta\mu, J \cup L) & = \ell(\vartheta\mu, J) & \ell(rec\, \vartheta\mu, J \cup L) & = \ell(\vartheta\mu, J) \\
\ell((\vartheta\langle\vartheta_0\mu_0, \vartheta_1\mu_1\rangle, r), J) & = (\beta, r), \text{ with } \ell(\mu_0) = (\beta, r_0) & &
\end{array}
$$

with $i \in \{0, 1\}$

$$
\beta = \begin{cases} x \text{ if } x \notin J \text{ and } \beta = x@n,\ n \in \mathbb{N} \\ \tau \text{ otherwise} \end{cases} \qquad rate = \begin{cases} \top \text{ if } r = 1 \\ r \ \text{ otherwise} \end{cases}
$$

Finally we come to the main theorem of the paper which records the equivalence between a PEPA net process and its stochastic π-calculus translation.

Theorem 5.3 *Given a Pnet process $P_n = \langle S, T, F, (D, M)\rangle$, then the following relation holds: $P_n \sim \mathcal{E}_\sigma[\![P_n]\!]$.*

Corollary 5.4 *The CTMC derived directly from P_n, via the PEPA net operational semantics, is isomorphic to the CTMC derived, using the late proved transition semantics, from $\mathcal{E}_\sigma[\![P_n]\!]$.*

6 Conclusions

We have presented a translation of a sublanguage of PEPA nets into the stochastic π-calculus, and defined an equivalence between PEPA net terms and stochastic π-calculus terms. Moreover, we have used the translation to show the equivalence of an input PEPA net and its image under translation. This implies isomorphism between their underlying CTMCs.

The translation has several distinctive features. Most notably it does not use the channel-passing mechanism of the stochastic π-calculus to capture token mobility. In future work we intend to explore this issue further, in particular deriving alternative translations which do incorporate this feature.

The paper shows that PEPA nets and stochastic π-calculus have a common subset language which originates the same Markov model. We plan to characterize this subset to highlight the differences between the two formalisms and the implications of them on performance prediction.

This translation relates the two languages by embedding one in the other. A companion work compares the two languages by equipping PEPA nets with an enhanced operational semantics in the stochastic π-calculus style [5].

Acknowledgements

The authors are supported by the DEGAS (Design Environments for Global ApplicationS) project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing.

References

[1] S. Gilmore, J. Hillston, and M. Ribaudo. PEPA nets: A structured performance modelling formalism. In A.J. Field and P.G. Harrison, editors, *Proceedings of the 12th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, number 2324 in Lecture Notes in Computer Science, pages 111–130, London, UK, April 2002. Springer-Verlag.

[2] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[3] C. Priami. Stochastic π-calculus. In S. Gilmore and J. Hillston, editors, *Proceedings of the Third International Workshop on Process Algebras and Performance Modelling*, pages 578–589. Special Issue of *The Computer Journal*, 38(7), December 1995.

[4] P. Degano and C. Priami. Enhanced operational semantics: a tool for describing and analysing concurrent systems. *ACM Computing Surveys*, 33(2):135–176, 2001.

[5] S. Gilmore and J. Hillston. An enhanced operational semantics for PEPA nets. DEGAS internal document, 2002.

[6] R. Milner, J.G. Parrow, and D.J. Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–77, 1992.

A Background

In this appendix we provide an introduction to the stochastic π-calculus and PEPA nets modelling languages. This material is included here for completeness and can safely be disregarded by a reader who is familiar with papers such as [3] and [1].

A.1 Stochastic π-calculus

Definition A.2 *Let $\mathcal{N}$ be a countable infinite set of* names *ranged over by $a, b, \ldots, x, y, \ldots$ with $\mathcal{N} \cap \{\tau\} = \emptyset$. Let $\mathcal{A}$ be a set of* agent identifiers *ranged over by $A, A_1, \ldots$. Let $r \in \mathbb{R}$. Processes, denoted by $P, Q, R, \ldots$, are built according to the following syntax:*

$$P ::= \mathbf{0} \mid (\pi, r).P \mid P + P \mid P|P \mid (\nu x)P \mid \mathbf{rec}\ X.P$$

where π is either $x(y)$ for input, $\overline{x}z$ for output or τ for silent moves.

The prefix π is the first atomic action that the process $\pi.P$ can perform. The input prefix binds the name variable denoted by y in the prefixed process. Intuitively, some name z is received along the link named x. The output prefix does not bind the name or the process z which is sent along x. The silent prefix τ denotes an action which is invisible to an external observer of the system. Summation denotes non deterministic choice. The operator $|$ describes parallel composition of processes. The operator (νx) acts as a static binder for the name x in the process P that it prefixes. In other words, x is a unique name in P which is different from all the external names. Finally, $\mathbf{rec}\ X.P$ is the definition of recursion processes, where X is the recursive variable.

The labels of the transitions are defined in A.3. The possible actions (with metavariable μ) are (τ, r) for silent actions, $(x(y), r)$ for input, $(\overline{x}z, r)$ for free output, $(\overline{x}(y), r)$ for bound output.

Definition A.3 *Let $\mathcal{L} = \{||_0, ||_1\}$ and $\mathcal{O} = \{+_1, +_0, \mathbf{rec}, (\nu x)\}$, where $\vartheta \in (\mathcal{L} \cup \mathcal{O})^*$. Then the set Θ of* enhanced labels *(with metavariable θ) is defined by the following syntax*

$$\theta = \vartheta\mu \mid \vartheta\langle ||_0\vartheta_0\mu_0, ||_1\vartheta_1\mu_1\rangle$$

with $\mu_0 = x(y)$ if and only if $\mu_1 = \overline{x}z$ or $\mu_1 = \overline{x}(z)$, or vice versa.

Hereafter, we sometimes write (νJ) for $(\nu x_1), \ldots (\nu x_n)$ if $J = \{x_1, \ldots, x_n\}$. We use the notion of free names $fn(\mu)$, bound names $bn(\mu)$ and names $n(\mu) = fn(\mu) \cup bn(\mu)$ of a label μ, as well as the notions of free (fn_p) and bound (bn_p) names in a process.

μ	$fn(\mu)$	$bn(\mu)$
τ	$\emptyset$	$\emptyset$
$(\overline{x}\, y, r)$	$\{x,\ y\}$	$\emptyset$
$(x(y), r)$	$\{x\}$	$\{\ y\}$

Below we assume that the *structural congruence* $\equiv$ on processes is defined as the least congruence satisfying the following clauses:

- P and Q α-equivalent (they only differ in the choice of bound names) implies $P \equiv Q$,
- $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$, $(\nu x)(R \mid S) \equiv (\nu x)R \mid S$ if $x \notin fn(S)$, $(\nu x)(R \mid S) \equiv R \mid (\nu x)S$ if $x \notin fn(R)$, and $(\nu x)P \equiv P$ if $x \notin fn(P)$.

A *variant* of $P \xrightarrow{\mu} Q$ is a transition which only differs in that P and Q have been replaced by structurally congruent processes, and μ has been α-converted, where a name bound in μ includes Q in its scope [6]. The *apparent rate* [2] of an action a in a given process P will be denoted by $r_a(P)$.

The operational semantics of our calculus is in Figure A.1.

A.4 PEPA nets

The semantic rules for PEPA nets are shown in Figure A.2. We present most of the rules for PEPA (**Prefix, Choice, Hiding** and **Constant**) without comment. The first two rules for **Cooperation** represent *individual activities* in which component proceed independently and concurrently. The final rule captures the case of *shared activities*, whose types appear in the cooperation set. Here the rate of the shared activity is adjusted to reflect the inability of the slower co-operand to function at the rate of the faster co-operand. The adjustment is made using the *apparent rates* of the components.We use the notation ${}^{P}_{r_1}\langle\alpha\rangle^{Q}_{r_2}$ to represent this adjustment to the rate when P would perform α at rate r_1 and Q would perform α at rate r_2.

$$ {}^{P}_{r_1}\langle\alpha\rangle^{Q}_{r_2} = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q)) $$

We write $P =_a Q$ to indicate that P and Q have equal alphabets.

Three new rules are added for PEPA nets. In order to present these we

$$Act : \mu.P \xrightarrow{\mu}_{\emptyset} P' \qquad Res : \frac{P \xrightarrow{\vartheta\mu} P'}{(\nu x)P \xrightarrow{\vartheta(\nu x)\mu} (\nu x)P'}, x \notin n(\ell(\mu))$$

$$Sum_0 : \frac{P \xrightarrow{\vartheta\mu} P'}{P + Q \xrightarrow{+_0\vartheta\mu} P'} \; Par_0 : \frac{P \xrightarrow{\vartheta\mu} P'}{P|Q \xrightarrow{\|_0\vartheta\mu} P'|Q}, (bn(\mu)) \cap fn(Q) = \emptyset$$

$$Sum_1 : \frac{Q \xrightarrow{\vartheta\mu} Q'}{P + Q \xrightarrow{+_1\vartheta\mu} Q'} \; Par_1 : \frac{Q \xrightarrow{\vartheta\mu} Q'}{P|Q \xrightarrow{\|_1\vartheta\mu} P|Q'}, (bn(\mu)) \cap fn(P) = \emptyset$$

$$Com_0 : \frac{P \xrightarrow{\vartheta_0(\overline{x}z, r_1)} P', Q \xrightarrow{\vartheta_1(x(y), r_2)} Q'}{P|Q \xrightarrow{(\langle \|_0\vartheta_0(\overline{x}z, r_1), \|_1\vartheta_1(x(y), r_2)\rangle, R)} P'|Q'\{z/y\}} \; Rec : \frac{rec\ X.P \xrightarrow{\vartheta} P'}{P \xrightarrow{\mathbf{rec}\ \vartheta} P'\{{}^{\mathbf{rec}\ X.P}/_X\}}$$

$$Com_1 : \frac{Q \xrightarrow{\vartheta_0(\overline{x}z, r_1)} Q' P \xrightarrow{\vartheta_1(x(y), r_2)} P'}{Q|P \xrightarrow{(\langle \|_0\vartheta_0(\overline{x}z, r_1), \|_1\vartheta_1(x(y), r_2)\rangle, R)} Q'|P'\{z/y\}} \; Open : \frac{P \xrightarrow{\vartheta\overline{x}y} P'}{(\nu y)P \xrightarrow{\vartheta\overline{x}(y)} P'}, x \neq y$$

$$Close_0 : \frac{P \xrightarrow{\vartheta_0(\overline{x}(z), r_1)} P', Q \xrightarrow{\vartheta_1(x(y), r_2)} Q'}{P|Q \xrightarrow{(\langle \|_0\vartheta_0(\overline{x}(z), r_1), \|_1\vartheta_1(x(y), r_2)\rangle, R)} (\nu z)(P'|Q'\{z/y\})}, \; y \notin fn(Q)$$

$$Close_1 : \frac{Q \xrightarrow{\vartheta_0(\overline{x}(z), r_1)} Q', P \xrightarrow{\vartheta_1(x(y), r_2)} P'}{Q|P \xrightarrow{(\langle \|_0\vartheta_0(\overline{x}(z), r_1), \|_1\vartheta_1(x(y), r_2)\rangle, R)} (\nu z)(Q'|P'\{z/y\})}, \; y \notin fn(P)$$

$$R = \frac{r_1}{r_{\overline{x}}(P)} \times \frac{r_2}{r_{\overline{x}}(Q)} \times min(r_{\overline{x}}(P), r_{\overline{x}}(Q))$$

Fig. A.1. Late proved transition system for Sπ-calculus.

make use of a relational operator

$$P_1 \overset{(\alpha, r)}{\longrightarrow\!\![]\!\!\longrightarrow} P_2$$

which expresses the existence of a transition in the net structure, connecting places P_1 and P_2 labelled by activity (α, r). The negation of this operator expresses the non-existence of such a transition.

The **Cell** rule conservatively extends the PEPA semantics to define that a cell with is filled by a component P has the same transitions as P itself. A typing judgement requires the context $P[]$ to be filled with a component which has the same alphabet as P, expressed by the alphabet equivalence $=_a$. There are no rules to infer transition for an empty cell because an empty cell enable no transitions.

The **Transition** rule state that the net has local transitions which change only a single component in the marking vector; moreover, these transitions agree with the transitions generated by the preceding rules. The second premise ensures that a local transition α can only occur in a place in a net which does not enable a net-level transition labelled by α. Note that this negative requirement is a static requirement related to the structure of the net, not the negation of the transition relation which is being defined. Thus, the rule cannot fail to be stratifiable.

The **Firing** rule represents a PEPA activity which moves a PEPA component from the input place to the output place via a transition firing. This has the effect that two entries in the marking vector change. The rate at which the activity is performed is calculated as in the PEPA semantics for cooperation.

$$Pre : \frac{}{(\alpha, r).P \xrightarrow{(\alpha,r)} P}$$

$$Sum_0 : \frac{P \xrightarrow{(\alpha,r)} P'}{P + Q \xrightarrow{(\alpha,r)} P'} \qquad Par_0 : \frac{P \xrightarrow{(\alpha,r)} P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P' \underset{L}{\bowtie} Q} \; (\alpha \notin L)$$

$$Sum_1 : \frac{Q \xrightarrow{(\alpha,r)} Q'}{P + Q \xrightarrow{(\alpha,r)} Q'} \qquad Par_1 : \frac{Q \xrightarrow{(\alpha,r)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P \underset{L}{\bowtie} Q'} \; (\alpha \notin L)$$

$$Com : \frac{P \xrightarrow{(\alpha,r_1)} P' \quad Q \xrightarrow{(\alpha,r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,R)} P' \underset{L}{\bowtie} Q'} \; (\alpha \in L, R = {}^{P}_{r_1}\langle \alpha \rangle^{Q}_{r_2})$$

$$Hide_0 : \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\alpha,r)} P'/L} \; (\alpha \notin L) \quad Hide_1 : \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\tau,r)} P'/L} \; (\alpha \in L)$$

$$Const : \frac{P \xrightarrow{(\alpha,r)} P'}{A \xrightarrow{(\alpha,r)} P'} \; (A \stackrel{def}{=} P) \quad Cell : \frac{P' \xrightarrow{(\alpha, r)} P''}{P[P'] \xrightarrow{(\alpha, r)} P[P'']} \; (P =_a P')$$

$$Tra : \frac{P_i \xrightarrow{(\alpha, r)} P_i' \qquad P_i \not\xrightarrow{(\alpha, r')} P_j}{(\ldots, P_i, \ldots) \xrightarrow{(\alpha, r)} (\ldots, P_i', \ldots)}$$

$$Fire : \frac{Q \xrightarrow{(\alpha, r_1)} Q' \qquad P_i \xrightarrow{(\alpha, r_2)} P_j}{(P_i[Q], \ldots, P_j[\_]) \xrightarrow{(\alpha,R)} (P_i[\_], \ldots, P_j[Q'])} \; (R = {}^{Q}_{r_1}\langle \alpha \rangle^{P_i}_{r_2})$$

Fig. A.2. The structured operational semantics of PEPA nets

$$
\begin{array}{llll}
\mathsf{alph}\ ((\alpha, r).R) = \{\alpha\} \cup \mathsf{alph}\ R & & \mathsf{alph}\ (R/L) & = \mathsf{alph}\ R \backslash L \\
\mathsf{alph}\ ((\alpha, \top).R) = \{\alpha\} \cup \mathsf{alph}\ R & & \mathsf{alph}\ (R \bowtie_L Q) & = \mathsf{alph}\ R \cup \mathsf{alph}\ Q \\
\mathsf{alph}\ (R + Q) = \mathsf{alph}\ R \cup \mathsf{alph}\ Q & & \mathsf{alph}\ (R[R]) & = \mathsf{alph}\ R
\end{array}
$$

$$
\begin{array}{ll}
\mathsf{spec}\ P\ (a.R)\ K & = \emptyset \\
\mathsf{spec}\ P\ (R + S)\ K & = \emptyset \\
\mathsf{spec}\ P\ (P[C]/L)\ K & = ((K \backslash L) \cap \mathsf{alph}\ P) \\
\mathsf{spec}\ P\ (Q[C]/L)\ K & = \emptyset \\
\mathsf{spec}\ P\ (Q/L)\ K & = \emptyset \\
\mathsf{spec}\ P\ (R/L)\ K & = \mathsf{spec}\ P\ R\ (K \backslash L) \\
\mathsf{spec}\ P\ (P \bowtie_L S)\ K & = (L \cap \mathsf{alph}\ P) \cup (K \cap \mathsf{alph}\ P) \cup \mathsf{spec}\ P\ S\ (K \cup L) \\
\mathsf{spec}\ P\ (Q \bowtie_L S)\ K & = \mathsf{spec}\ P\ S\ (K \cup L) \\
\mathsf{spec}\ P\ (R \bowtie_L S)\ K & = \mathsf{spec}\ P\ R\ (K \cup L) \cup \mathsf{spec}\ P\ S\ (K \cup L)
\end{array}
$$

$$
\begin{array}{ll}
\mathsf{spec}_t\ P\ (a.R)\ K & = \emptyset \\
\mathsf{spec}_t\ P\ (R + S)\ K & = \emptyset \\
\mathsf{spec}_t\ P\ (P[C]/L)\ K & = \emptyset \\
\mathsf{spec}_t\ P\ (Q[C]/L)\ K & = \emptyset \\
\mathsf{spec}_t\ P\ (Q/L)\ K & = \emptyset \\
\mathsf{spec}_t\ P\ (R/L)\ K & = \mathsf{spec}_t\ P\ R\ (K \backslash L) \\
\mathsf{spec}_t\ P\ (P \bowtie_L S)\ K & = (L \cap \mathsf{alph}\ P) \cup (K \cap \mathsf{alph}\ P) \cup \mathsf{spec}\ A_t^i\ S\ (K \cup L) \\
\mathsf{spec}_t\ P\ (Q \bowtie_L S)\ K & = \mathsf{spec}_t\ P\ S (K \backslash L) \\
\mathsf{spec}_t\ P\ (A_t^i \bowtie_L S)\ K & = (L \cap \mathsf{alph}\ A_t^i) \cup (K \cap \mathsf{alph}\ A_t^i) \cup \mathsf{spec}\ P\ S\ (K \cup L) \\
\mathsf{spec}_t\ P\ (R \bowtie_L S)\ K & = \mathsf{spec}_t\ P\ R\ (K \cup L) \cup \mathsf{spec}_t\ P\ S\ (K \cup L)
\end{array}
$$

P, Q are distinct identifiers, R, S expressions, K, L sets of actions, C is a metavariable which stands for _ (empty space) or for an identifier.

Fig. A.3. Definition of spec and spec_t function.