Mapping coloured stochastic Petri nets to stochastic process algebras

Linda Brodo [1]

Istituto Trentino di Cultura - IRST
Povo (Trento), 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 mapping which translates a model in a coloured stochastic Petri net notation (PEPA nets) into a term of a foundational stochastic process algebra (Stochastic CCS). Our mapping is a compositional static method which translates from one high-level model to another without needing to evaluate the state space of the given PEPA net. The result of the translation is a well-formed input for verification tools which manipulate process algebra terms but provide no support for coloured nets. We demonstrate the correctness of the translation by proving a correspondence result between the input and the image under translation.

[1] Email: brodo@itc.it
[2] Email: stg@inf.ed.ac.uk. Corresponding author.
[3] Email: jeh@inf.ed.ac.uk
[4] Email: priami@dit.unitn.it

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 coloured stochastic Petri net where the tokens of the net 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. Exponentially-distributed random delays are associated with both transitions and firings so that 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. Typical subjects for modelling with PEPA nets are mobile code software systems under a discipline of dynamic binding of names. The PEPA nets language enforces the property that communication between components at different places is not allowed. Remote communication is not an allowable atomic operation and must be implemented via a combination of migration and local communication.

There are several benefits of analysing communication rules such as these via a mapping into a foundational process algebra.

(i) The communication rules are rather new but the algebra is simple and well-understood. This gives us a way to consider the new language features using established proven analysis methods.

(ii) The encoding provides a route to process algebra-based verification tools such as model-checkers and theorem provers.

(iii) The encoding may expose previously unknown properties of the PEPA nets language which can be used profitably to develop new proof techniques.

(iv) The general approach may be applicable to untimed coloured Petri nets and classical process algebras.

We explore the relationship between coloured stochastic Petri net and process algebras by mapping PEPA nets into the foundational process algebra, Milner's CCS [3] enhanced with timing information to give *Stochastic CCS* (CCS_S).

If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets and CCS_S 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 CCS$_S$ 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.

$$\begin{array}{ccc} Net & \rightsquigarrow & CCS_S \\ \downarrow & & \downarrow \\ LTS_1 & & LTS_2 \\ \downarrow & & \downarrow \\ CTMC_1 & \equiv & CTMC_2 \end{array}$$

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 a sequential CCS$_S$ term. Formally, the size of the CCS$_S$ 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 not the case that the full PEPA nets language can be mapped into CCS$_S$ in a fully compositional fashion. To point to one reason for this, here is no suitable encoding of the PEPA multi-way synchronisation into the two-way handshake synchronisation of CCS$_S$. For this reason we aim only to encode a subset of the language of PEPA nets in CCS$_S$.

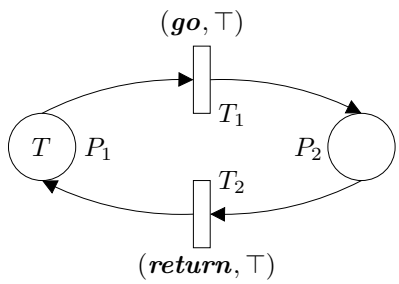
Structure of this paper

This paper is rather technical and has a number of formal definitions. To help the reader to understand the purpose of the definitions which come later we first present a small example in Section 2 which compares an input to our translation with the corresponding output. We then continue to expose more detail of the translation from PEPA nets to CCS$_S$ with a discussion of its high-level properties in Section 3. The translation itself is presented via a series of inter-related function definitions in Section 4. In Section 5 we present a larger example which exercises some of the subtleties of the translation which have been highlighted in the previous section. In Section 6 we present properties of the translation including our primary result, that our encoding of a PEPA net

generates a CCS_S process which describes a labelled continuous-time Markov chain which is isomorphic to that obtained from the input PEPA net. Related work is discussed in Section 7. Section 8 presents conclusions on the work. To make the paper self-contained, we provide an introduction to PEPA nets and CCS_S in Appendix A. Readers unfamiliar with these languages would benefit from reading Appendix A first.

2 An introductory example: lights on and off

We illustrate the idea of mapping a PEPA net into CCS_S 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 of the net is a cyclic sequence of eight activities (switch the light on; go to the other place; switch the light on there; return to the previous location; switch the light off there; go to the other place; switch the light off there; return to the previous location). We chose different rates for the eight activities $(r_1, \ldots, r_8)$.

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

A switch simply enforces the discipline that it must alternately be switched on and off.

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

Places P_1 and P_2 of the net provide contexts which describe a (static, immobile) switch and a *cell* to contain a (dynamic, cycling) token. When the token is resident it synchronises with the switch to perform the activities *on* and *off*. When the token is not present the switch cannot change state because it is partnered with a token in a PEPA synchronisation and cannot perform the synchronised activities without its partner.

$$P_1[t] \stackrel{def}{=} Switch \underset{\{on,off\}}{\bowtie} Token[t]$$
$$P_2[t] \stackrel{def}{=} Switch \underset{\{on,off\}}{\bowtie} Token[t]$$

Finally, we describe the initial marking of the net in which the token is resident in place P_1 in its initial state ($Token$).

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

We now present this example in CCS_S. 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. $\mathbf{fix}\,(Token = (\overline{on@1}, r_1).(\boldsymbol{go}, r_2).(\overline{on@2}, r_3).(\boldsymbol{return}, r_4).$
   $(\overline{off@1}, r_5).(\boldsymbol{go}, r_6).(\overline{off@2}, r_7).(\boldsymbol{return}, r_8).Token)$
2. $\mathbf{fix}\,\big(TokenMob = (\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).TokenMob\big)$

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. $\mathbf{fix}\,\big(P_1 = \mathbf{fix}\,\big(Switch = (on@1, 1).(off@1, 1).Switch\big)\big)$
2. $\mathbf{fix}\,\big(P_2 = \mathbf{fix}\,\big(Switch = (on@2, 1).(off@2, 1).Switch\big)\big)$

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

$$\begin{aligned}
System \stackrel{def}{=} (&(\mathbf{fix}\,(TokenMob = (\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).(\overline{\boldsymbol{go}}, 1).(\overline{\boldsymbol{return}}, 1).\\
&\qquad TokenMob)\\
&\mid \mathbf{fix}\,(Token = (\overline{on@1}, r_1).(\boldsymbol{go}, r_2).(\overline{on@2}, r_3).(\boldsymbol{return}, r_4).\\
&\qquad (\overline{off@1}, r_5).(\boldsymbol{go}, r_6).(\overline{off@2}, r_7).(\boldsymbol{return}, r_8).Token)\\
&) \setminus \{\, \boldsymbol{go}, \boldsymbol{return} \,\}\\
&\mid \mathbf{fix}\,\big(P_1 = \mathbf{fix}\,\big(Switch = (on@1, 1).(off@1, 1).Switch\big)\big)\\
&\mid \mathbf{fix}\,\big(P_2 = \mathbf{fix}\,\big(Switch = (on@2, 1).(off@2, 1).Switch\big)\big)\\
&) \setminus \{\, on@1, off@1, on@2, off@2 \,\}
\end{aligned}$$

Both of these models define a simple eight-state system which is viewed as a CTMC as below. The symbol $\circ$ denotes a bulb which is lit and $\bullet$ denotes one

which is not. The plus superscript denotes the presence of the token.

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

3 High-level overview of the translation

We now describe the translation of a PEPA net into an equivalent Stochastic CCS 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 (where 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 CCS 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 activities of the specialised representations of the tokens of the net. Concretely, if the PEPA net specifies that a token

will first perform activity α and that it is initially in place P_i then the CCS_S translation will first perform $\alpha@i$.

We compose in parallel each token with the definition of the place where it is initially, restricting on the names of the transition activities. The remaining free names of the system are restricted.

3.1 Limitations on translatable PEPA nets

We require that our translation does not modify the continuous-time Markov chain which is generated from the CCS_S-encoding. Doing so would invalidate any analysis performed on the translated representation. This strong condition obliges us to limit the class of nets we will translate.

There are two kinds of limitations. The first kind deals with details of the PEPA stochastic process algebra language which is used to encode the tokens of the net. The second kind is more closely tailored to features of the PEPA nets formalism.

Concerning the PEPA language limitations we do not translate the multi-way synchronization of PEPA. We only translate binary synchronizations. These are usually between one active participant (specifying a rate via a real-valued random variable) and one passive participant (indicating that it does not specify the rate of the activity by using the distinguished symbol $\top$).

The limitations imposed on the PEPA nets formalism for the purposes of this encoding into CCS_S are listed below.

(i) All tokens are different. Cells in a PEPA net context specify the type of token which they can store so one consequence of this restriction is that each place has as many cells as all the tokens which can reach that place.

(ii) Within a place it is possible to specify names for synchronising activites between tokens and static components. In addition it is possible to hide names in order to prevent further synchronisation. However, the names which are used to interact with tokens cannot be hidden. For example, consider the following place definition.

$$P[t] \stackrel{def}{=} \left(S \underset{L}{\bowtie} Token[t]\right)/L'$$

Here it must be the case that $L \cap L' = \emptyset$.

(iii) Within a place it is possible for two tokens to synchronise, if they are both resident in cells in the place. However, it is not possible to hide the names which are used to interact. Consider the following place definition.

$$P[t_1, t_2] \stackrel{def}{=} \left(Token_1[t_1] \underset{L}{\bowtie} Token_2[t_2]\right)/L'$$

Again it must be the case that $L \cap L' = \emptyset$.

(iv) A place may use several synchronisation sets to specify the interactions between tokens or between static components and tokens. For any place with N such synchronisation sets we can consider these to be an indexed family L_i for $i = 1, \ldots, N$. We require these sets to be pairwise disjoint so that $L_i \cap L_j = \emptyset$, for all $i, j \in 1, \ldots, N$.

(v) The tokens of the PEPA net must be sequential processes. They cannot be defined using PEPA synchronization.

(vi) In each place and in each token, there is only a single use of PEPA's hiding operator and it must be at the outermost level. This restriction corresponds to defining a *standard form* for writing the definition of a place and of a token.

Some clarifying comments are needed on the above. The difficulty throughout the encoding is in representing the range of synchronisation possibilities afforded in PEPA nets and mapping them into the CCS_S algebra. At the PEPA level we have an indexed family of synchronisation operators and a hiding operator ("$\bowtie$" and "/") and at the CCS level we have a single parallel composition operator and a restriction operator ("|" and "\").

On specific points above, concerning limitation (i) in CCS we cannot have a process use a name which encodes the current location of a token (using the $@i$ suffix) if many processes are using this name. This would create unintentional synchronisation points which were not in the original PEPA net.

Limitation (ii) characterizes the role of the hiding operator which creates a local environment where a token, when it enters this local environment, adopts the local hiding policy. This cannot be simulated in CCS without adding a *dynamic* restriction operator which the language does not currently have; names are statically bound in CCS. Limitation (iii) is needed for the same reason.

Limitation (iv) imposes a general discipline on the use of synchronization names inside a place which we need. This limitation derives from the fact that the effective interface of the tokens of a PEPA net dynamically changes depending on the position of the token. In contrast the CCS restriction operator has a fixed scope. We place the uses of restriction at the outermost position as each token may potentially interact with any other place/token process.

We want to abstract away the internal parallel structure of the tokens so limitation (v) requires any concurrent tokens to be re-expressed as sequential components. This is always possible by repeated applications of the expansion law of interleaving process algebras such as PEPA. Also, we can still rely on a compositional translation function.

Finally, in the last point we assume a unique name environment for each place and for each token. This point of view is familiar from calculi which allow α-conversion of restricted names.

We use a *restricted top-level* (RTL) form for CCS processes where all the restrictions are pushed to the top level of terms. To have processes in RTL-form, the congruence rules will be applied at the beginning and each time that recursive operators introduce new restrictions.

We term PEPA nets obeying the above limitations *limited PEPA nets*.

4 A functional encoding of the translation

In this section we will define a family of functions to encode a limited PEPA net (*Pnet*) process into a Stochastic CCS 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 $\mathcal{E} : Pnet \to CCS_S$ is expressed by:

$$\mathcal{E}_\sigma(M) = P_{CCS}$$

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.

$$\mathcal{E}_\sigma\big((P_1[A_1 \ldots, A_m], \ldots, P_N[A_{m+n}, \ldots, A_M])\big) = \\ \big((\Pi_{i=1}^{N}\ S_{CCSi})\ \mid\ \Pi_{j=1}^{M}\ (M_{CCSj}\ \mid\ T_{CCSj}) \setminus names_j\big) \setminus names$$

where

$$S_{CCSi} = \mathcal{E}S^i_\sigma(P_i),$$
$$M_{CCSj} = \mathcal{E}M_\sigma(A_j),$$
$$T_{CCSj} = \mathcal{E}T^i_\sigma(A_j),$$
$$names_j = \mathrm{fn}_p(M_{CCSj}),$$

and $names = \{\alpha@i \mid \alpha \in \mathsf{spec}\ A_j\ P_i\ \emptyset\ \ \forall i \in [1 \ldots N], \forall j \in [1 \ldots M]\}$

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 the position of the name, *i.e.* in which place the name is used.

The function $\mathcal{E}S^i_\sigma(\_) : \mathbb{N} \times Pnet \to CCS_S$ 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.

$$
\begin{aligned}
\mathcal{E}S^i_\sigma(P \underset{L}{\bowtie} R) &= (\mathcal{E}S^i_\sigma(P) \mid \mathcal{E}S^i_\sigma(R)) \setminus L' \\
&\quad \text{where } L' = \{x@i \mid x \in (L \setminus \textstyle\bigcup_{j=1}^{M} \mathsf{spec}\ A_j\ P_i\ \emptyset)\} \\
\mathcal{E}S^i_\sigma(P/L) &= \mathcal{E}S^i_\sigma(P)\{^{x@\mathrm{new}()}/_{x@i}\}\ \forall\ x \in L \\
\mathcal{E}S^i_\sigma(I[C]) &= \mathbf{0} \\
\mathcal{E}S^i_\sigma(S_1 + S_2) &= \mathcal{E}S^i_\sigma(S_1) + \mathcal{E}S^i_\sigma(S_2) \\
\mathcal{E}S^i_{\langle S,T,F,D\rangle}(I) &= \begin{cases} \mathbf{fix}\left(I = \mathcal{E}S^i_{\langle S,T,F,D\setminus\{I \stackrel{def}{=} P\}\rangle}(P)\right) & \text{if } (I \stackrel{def}{=} P) \in D \\ I & \text{otherwise} \end{cases} \\
\mathcal{E}S^i_\sigma((\alpha, r).S) &= \begin{cases} (\alpha@i, 1).\mathcal{E}S^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{E}S^i_\sigma(S) & \text{otherwise} \end{cases}
\end{aligned}
$$

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

The function $\mathcal{E}M_\sigma(\_) : \mathbb{N} \times Pnet \to CCS_S$ translates only the firing actions of the jth token. It is applied to each token and gives as result a CCS process which synchronizes with the $\mathcal{E}T_\sigma$-translation of that token to allow it to move between places.

The function $\mathcal{E}T^i_\sigma(\_) : \mathbb{N} \times \mathbb{N} \times Pnet \to CCS_S$ translates all the transition actions of the jth token in the ith place.

The definitions of the functions $\mathcal{E}S^i_\sigma()$, $\mathcal{E}M_\sigma()$ and $\mathcal{E}T^i_\sigma()$ are given in Figures 1, 2 and 3, respectively. In Figure A.5 there is the definition of function spec, which calculates all the synchronisation activities a token-process uses in cooperation with static components of a place-process. In defining encoding functions we will use the function new() which returns each time a fresh name.

5 A larger example: a mobile software agent

We illustrate the application of the above translation to the following PEPA net modelling a mobile software agent as described in [1]. In this example a mobile agent visits three sites. It interacts with static software components at the three sites and has two kinds of interactions. When visiting a site where a network probe is present it interrogates the probe to collect the data which it has gathered on recent patterns of network traffic. When the agents returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe analyses the data. The structure of the system allows the analysis to be temporally overlapped with the agent's

communication and data gathering.

We first present the place index set, the transition names and the firing relation.

$$S = \{1, 2, 3\}, \qquad T = \{\boldsymbol{go}, \boldsymbol{return}\},$$
$$F = \{(1, \boldsymbol{return}, 2), (2, \boldsymbol{go}, 1), (2, \boldsymbol{go}, 3), (3, \boldsymbol{return}, 2)\}$$

Now we progress to the definitions of the places, the static components and the token of the net. Collectively these make the definition set D. First we present the components: the probes, the master probe and the mobile agent.

$$\begin{aligned}
\mathcal{E}M_\sigma(P/L) &= \mathcal{E}M_\sigma(P) \\
\mathcal{E}M_\sigma(S_1 + S_2) &= \mathcal{E}M_\sigma(S_1) + \mathcal{E}M_\sigma(S_2)
\end{aligned}$$

$$\mathcal{E}M_{\langle S,T,F,D\rangle}(I) = \begin{cases} \mathbf{fix}\left(IMob = \mathcal{E}M_{\langle S,T,F,D\setminus\{I \stackrel{def}{=} P\}\rangle}(P)\right) \text{ if } I \stackrel{def}{=} P \in D \\ IMob \qquad\qquad \text{otherwise} \end{cases}$$

$$\mathcal{E}M_\sigma((\alpha, r).S) = \begin{cases} \sum_{h=1}^{H}(\overline{\alpha}\,, 1).\mathcal{E}M_\sigma(S) & \text{if } \alpha \in T \text{ with} \\ + & (i, (\alpha, \top), z_1) \ldots (i, (\alpha, \top), z_H) \in F, \\ \sum_{k=1}^{K}(\overline{\alpha}\,, \lambda_k).\mathcal{E}M_\sigma(S) & (i, (\alpha, \lambda_1), z_1) \ldots (i, (\alpha, \lambda_K), z_K) \in F, \\ \mathcal{E}M_\sigma(S) & \text{otherwise} \end{cases}$$

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

$$\begin{aligned}
\mathcal{E}T^i_\sigma(P/L) &= \mathcal{E}T^i_\sigma(P)\{^{\text{new}()@i}/_{xi}\}\ \forall\, x \in L \\
\mathcal{E}T^i_\sigma(S_1 + S_2) &= \mathcal{E}T^i_\sigma(S_1) + \mathcal{E}T^i_\sigma(S_2)
\end{aligned}$$

$$\mathcal{E}T^i_{\langle S,T,F,D\rangle}(I) = \begin{cases} \mathbf{fix}\left(I = \mathcal{E}T^i_{\langle S,T,F,D\setminus\{(I \stackrel{def}{=} P)\}\rangle}(P)\right) \text{ if } I \stackrel{def}{=} P \in D \\ I \qquad\qquad \text{otherwise} \end{cases}$$

$$\mathcal{E}T^i_\sigma((\alpha, r).S) = \begin{cases} \sum_{j=1}^{J}(\alpha, 1).\mathcal{E}T^{z_j}_\sigma(S) & \text{if } \alpha \in T \text{ and } r = \top \\ \sum_{j=1}^{J}(\alpha, r).\mathcal{E}T^{z_j}_\sigma(S) & \text{if } \alpha \in T \text{ and } r \neq \top \\ (\overline{\alpha@i}, r).\mathcal{E}T^i_\sigma(S) & \text{if } \alpha \notin T \text{ and } r \neq \top \\ (\alpha@i, 1).\mathcal{E}T^i_\sigma(S) & \text{if } \alpha \notin T \text{ and } r = \top \end{cases}$$
$$\text{where } \forall\, j \in J\ (i, (\alpha, r_j), z_j) \in F$$

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

The probe chooses (via the summation operator) to monitor the network or to be interrogated by the agent, if it is present.

$$\begin{aligned} \mathit{Probe} &\stackrel{def}{=} (\mathit{monitor}, r_m).\mathit{Probe} + (\mathit{interrogate}, \top).\mathit{Probe} \\ \mathit{Master} &\stackrel{def}{=} (\mathit{dump}, \top).(\mathit{analyse}, r_a).\mathit{Master} \\ \mathit{Agent}[A] &\stackrel{def}{=} (\boldsymbol{go}.\lambda).(\mathit{interrogate}, r_i).(\boldsymbol{return}, \mu).(\mathit{dump}, r_d).\mathit{Agent} \end{aligned}$$

Every place of the net contains a cell which the agent can inhabit. Places 1 and 3 contain network probes. Place 2 is the central "home" location where the master probe resides. The synchronisation sets are selected appropriately.

$$\begin{aligned} P_1[A] &\stackrel{def}{=} \mathit{Agent}[A] \underset{\{\mathit{interrogate}\}}{\bowtie} \mathit{Probe} \\ P_2[A] &\stackrel{def}{=} \mathit{Agent}[A] \underset{\{\mathit{dump}\}}{\bowtie} \mathit{Master} \\ P_3[A] &\stackrel{def}{=} \mathit{Agent}[A] \underset{\{\mathit{interrogate}\}}{\bowtie} \mathit{Probe} \end{aligned}$$

Finally we conclude with the initial marking of the net. The agent is initially in its home location, in its initial state.

$$M = (P1[\_], P_2[A_1], P_3[\_]).$$

The result of the encoding function $\mathcal{E}_{\sigma=\langle S,T,F,D\rangle}(M)$ is the CCS_S process

$$\begin{aligned} P_{CCS} = \big(& \mathit{recP1} \mid \mathit{recP2} \mid \mathit{recP3} \mid \\ & (\mathit{recAgent} \mid \mathit{recAgentMob}) \setminus \{\, \boldsymbol{go}, \boldsymbol{return} \,\} \\ \big) & \setminus \{\, \mathit{interrogate}@3, \mathit{dump}@2, \mathit{interrogate}@1 \,\} \end{aligned}$$

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

$$\begin{aligned} \mathit{recP1} = \mathcal{E}S^1_\sigma(P_1) \;=\; \mathbf{fix}\, \big(P1 = \mathbf{fix}\, \big(\mathit{Probe} = \;& (\overline{\mathit{monitor}@1}, r_m).\mathit{Probe} + \\ & (\mathit{interrogate}@1, 1).\mathit{Probe}\big)\big) \end{aligned}$$

$$\begin{aligned} \mathit{recP2} = \mathcal{E}S^2_\sigma(P_2) \;=\; \mathbf{fix}\, \big(P2 = \mathbf{fix}\, \big(\mathit{Master} = \;& (\overline{\mathit{dump}@2}, 1). \\ & (\mathit{analyze}@2, r_a).\mathit{Master}\big)\big) \end{aligned}$$

$$\begin{aligned} \mathit{recP3} = \mathcal{E}S^3_\sigma(P_3) \;=\; \mathbf{fix}\, \big(P3 = \mathbf{fix}\, \big(\mathit{Probe} = \;& (\overline{\mathit{monitor}@3}, r_m).\mathit{Probe} + \\ & (\overline{\mathit{interrogate}@3}, 1).\mathit{Probe}\big)\big) \end{aligned}$$

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

$$\begin{aligned} recAgentMob = \mathcal{E}M^2_\sigma(Agent) \;= \mathbf{fix}\; \big(&AgentMob = \\ &(\overline{\boldsymbol{go}}, \lambda_l).(\overline{\boldsymbol{return}}, \mu_l).AgentMob + \\ &(\overline{\boldsymbol{go}}, \lambda_r).(\overline{\boldsymbol{return}}, \mu_r).AgentMob\big) \end{aligned}$$

$$\begin{aligned} recAgent = \mathcal{E}T^{(2,1)}_\sigma(Agent) \;= \mathbf{fix}\; \big(&Agent = \\ &(\boldsymbol{go}, \lambda).(interrogate@1, r_i). \\ &(\boldsymbol{return}, \mu).(dump@2, r_d).Agent + \\ &(\boldsymbol{go}, \lambda).(interrogate@3, r_i). \\ &(\boldsymbol{return}, \mu).(dump@2, r_d).Agent\big) \end{aligned}$$

The restricted names are derived by the following computations: $\{interrogate@3, dump@2, interrogate@1\} = \{\alpha@i \mid \alpha \in \mathsf{spec}\ A^2_1\ P_i\ \emptyset\}$, $\{\boldsymbol{go}, \boldsymbol{return}\} = \mathrm{fn}_p(recAgentMob)$.

6 Properties of the translation

Property 6.1 (Coverage) *Let* $P_n = (P_1[\_], \ldots, P_i[Token], \ldots, P_n[\_])$, *and* $T = \mathcal{E}T^i_\sigma(Token)$. *Then for all* μ_j *where* $T \xrightarrow{\mu_j} T'$, *either* $\mu_j = (x@i, r)$ *or* $\mu_j = (\overline{x@i}, r)$ *or* $\mu_j = (x, r)$ *or* $\mu_j = (\overline{x}, r)$.

Proof. As *Token* processes can only be specified by sequential operators (see our initial assumptions), the continuation following each action, i.e. $(\alpha, r).P$ is the sole continuation of the complete process. □

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

Definition 6.1 *The function* $\ell : \Theta \times \mathcal{N} \to \mathbb{R} \cup \top$ *returns the names of the channel used by the action together with its rate:* $\ell(\alpha, 1) = \top, \ell(\alpha, r) = r$.

Theorem 6.2 (Correspondence of CTMCs) *Let* $P_n = (P_1, \ldots, P_n)$ *be a restricted PEPA net, then*

$$P_n \xrightarrow{(\alpha,r)}_n P'_n \quad \textit{iff} \quad \mathcal{E}_\sigma(P_n) \xrightarrow{\theta}_{CCS} \mathcal{E}_\sigma(P'_n), \quad \textit{with } \ell(\theta) = r.$$

Proof. [Short form.] Both directions can be proved by cases on the rule applied and by induction on the length of the derivation of its premises. □

7 Related work

We view this work as part of a continuing programme to understand the connections between stochastic Petri nets and stochastic process algebras initiated by Donatelli, Hillston and Ribaudo [5] and continued by [6]. Like those earlier works, the present paper is largely focussed on theoretical concerns. Practical engineering insights on the similarities and differences between Petri nets and process algebras have come from attempts to use the two modelling formalisms together in an integrated approach to performance modelling. Examples of the latter approach include [7] and [8].

The contribution of this paper has taken the form of an encoding of a coloured stochastic Petri net language into a foundational process algebra. A complementary approach is to select a process algebra and equip it with a Petri net semantics as in the work of Ribaudo, which compares both representation issues [9] and analysis techniques [10].

8 Conclusions

We have presented a translation of a sublanguage of PEPA nets into Stochastic CCS. 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.

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 CCS_S style [11].

Even when applied to simple examples, the results of the encoding demonstrate that the input net is considerably simpler than its image under the encoding. This suggests that the added features of the PEPA nets language are a useful conceptual tool for performance modellers.

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] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[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. Donatelli, J. Hillston, and M. Ribaudo. A comparison of Performance Evaluation Process Algebra and Generalized Stochastic Petri Nets. In *Proc. 6th International Workshop on Petri Nets and Performance Models*, Durham, North Carolina, 1995.

[6] J. Hillston, L. Recalde, M. Ribaudo, and M. Silva. A comparison of the expressiveness of SPA and bounded SPN models. In B. Haverkort and R. German, editors, *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, Aachen, Germany, September 2001. IEEE Computer Science Press.

[7] G. Clark and W.H. Sanders. Implementing a stochastic process algebra within the Möbius modeling framework. In L. de Alfaro and S. Gilmore, editors, *Proceedings of the first joint PAPM-PROBMIV Workshop*, volume 2165 of *Lecture Notes in Computer Science*, pages 200–215, Aachen, Germany, September 2001. Springer-Verlag.

[8] M. Bernardo, N. Busi, and M. Ribaudo. Integrating TwoTowers and GreatSPN. In R. Gorrieri et al., editor, *ICALP Workshops 2000 — Proceedings of the Satellite Workshops of the 27th International Colloquium on Automata, Languages and Programming (Proceedings of the Eighth Annual Workshop on Process Algebra and Performance Modelling)*, number 8 in Proceedings in Informatics, pages 551–563, Geneva, Switzerland, September 2000. Carleton Scientific.

[9] M. Ribaudo. Stochastic Petri net semantics for stochastic process algebras. In *Proc. 6th International Workshop on Petri Nets and Performance Models*, Durham, North Carolina, 1995.

[10] M. Ribaudo. On the aggregation techniques in stochastic Petri nets and stochastic process algebras. In S. Gilmore and J. Hillston, editors, *Proceedings of the Third International Workshop on Process Algebras and Performance Modelling*, pages 600–611. Special Issue of *The Computer Journal*, 38(7), December 1995.

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

A Background on Stochastic CCS and PEPA nets

A.1 Stochastic CCS

Definition A.1 *Let $\mathcal{A}$ be an infinite set of* names *ranged over by $a, b, \ldots, x, y, \ldots$ and let $\overline{\mathcal{A}}$ be the set of co-names ranged over by $\overline{a}, \overline{b}, \ldots, \overline{x}, \overline{y}, \ldots$. Let $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ be the set of labels. Let $L \subseteq \mathcal{L}, \alpha \in L$. Let $\mathcal{K}$ be a set of* agent identifiers *ranged over by $A, A_1, \ldots$. Let $r \in \mathbb{R}$. Processes, denoted by $T, Q, R, \ldots$, are built according to the following syntax:*

$$T ::= \mathbf{0} \mid (\alpha, r).T \mid T + T \mid T|T \mid T \setminus L \mid \mathbf{fix}\,\big(X = T\big)$$

The prefix α is the activity type of the first activity that the process $(\alpha, r).T$ can perform. The operator $|$ describes parallel composition of processes. The restriction $\setminus L$ internalises the names in L and their complements. The singleton restriction $\setminus\{x\}$ is often abbreviated to $\setminus x$. Finally, $\mathbf{fix}\,\big(X = T\big)$ is the definition of a recursive processes, where X is the recursion variable which typically recurs within T.

The labels of the transitions are defined in Defn. A.2. The possible actions (with metavariable μ) are (x, r) for action, $(\overline{x}, r)$ for co-action.

Definition A.2 *The set Θ of* enhanced labels *(with metavariable θ) is defined by the following syntax.*

$$\theta = \mu \mid \langle(\mu_0, \mu_1), r\rangle$$

with $\mu_0 = (x, r_0)$ if and only if $\mu_1 = (\overline{x}, r_1)$, or vice versa. The notion of free, fn*, and bound names,* bn*, are the standard ones.*

Fig. A.1 presents the structured operational semantics up to α-equivalence.

$$T \setminus x \setminus y \equiv T \setminus y \setminus x \qquad (R \,|\, S) \setminus x \equiv (R \setminus x) \,|\, S \text{ if } x \notin \mathrm{fn}\,(S)$$

$$(R \,|\, S) \setminus x \equiv R \,|\, (S \setminus x) \text{ if } x \notin \mathrm{fn}\,(R) \qquad T \setminus x \equiv T \text{ if } x \notin \mathrm{fn}\,(T)$$

$$T_0|T_1 \equiv T_1|T_0 \qquad T_0 + T_1 \equiv T_1 + T_0$$

$$\mathbf{fix}\,\big(X = T\big) \equiv T\{\mathbf{fix}\,\big(X = T\big)/X\}$$

Fig. A.1. Structural congruence for CCS

A *variant* of $T \xrightarrow{\mu} Q$ is a transition which only differs in that T and Q have been replaced by structurally congruent processes, and μ has been α-converted. 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 Fig. A.2.

$$Act : \mu.T \xrightarrow{\mu}_{\emptyset} T' \qquad Com : \frac{T \xrightarrow{(\overline{x},r_1)} T', Q \xrightarrow{(x,r_2)} Q'}{T|Q \xrightarrow{(\langle(\overline{x},r_1),(x,r_2)\rangle,R)} T'|Q'}$$

$$Sum : \frac{T \xrightarrow{\theta} T'}{T + Q \xrightarrow{\theta} T'} \qquad Par : \frac{T \xrightarrow{\theta} T'}{T|Q \xrightarrow{\theta} T'|Q}$$

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

$$CCS1 : \frac{P \xrightarrow{(\alpha,r)} P'}{P \setminus I \xrightarrow{(\alpha,r)}_{CCS} P' \setminus I}, \text{ if the name in } \alpha \text{ is not in } I$$

$$CCS2 \frac{P \xrightarrow{\langle((\alpha_0,r_0),(\alpha_1,r_1)),R\rangle} P'}{P \setminus I \xrightarrow{\langle((\alpha_0,r_0),(\alpha_1,r_1)),R\rangle}_{CCS} P' \setminus I}, \text{ if the name in } \alpha_0 \text{ is in } I$$

Fig. A.2. Transition system for Stochastic CCS.

A.2 PEPA nets

The semantic rules for PEPA nets are shown in Fig. A.4. 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

make use of a relational operator

$$P_1 \xrightarrow{(\alpha, r)}\!\!\!\!\!\!\square\longrightarrow P_2$$

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.

The semantics of PEPA nets is given in terms of the congruence rules in Fig. A.2 and in terms of semantic rules in Fig. A.4.

$$T|Q \equiv Q|T \qquad\qquad P + Q \equiv Q + P$$
$$A \equiv P \text{ if } A \stackrel{def}{=} P$$

Fig. A.3. Congruence rules for PEPA nets processes.

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

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

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

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

$$Cell : \frac{P' \xrightarrow{(\alpha,r)} P''}{P[P'] \xrightarrow{(\alpha,r)}_n P[P'']} \ (P =_a P')$$

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

$$Fire : \frac{Q \xrightarrow{(\alpha,r_1)} Q' \qquad P_i \not\xrightarrow{(\alpha,r_2)} P_j}{(P_i[Q], \ldots, P_j[\_]) \xrightarrow{(\alpha,R)}_n (P_i[\_], \ldots, P_j[Q'])} \ (R = {}^{Q}_{r_1}\langle\alpha\rangle^{P_i}_{r_2})$$

Fig. A.4. The structured operational semantics of PEPA nets

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

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

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.5. Definition of spec function.