Mapping coloured stochastic Petri nets to stochastic process algebras

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

28th June 2003

Joint work with Linda Brodo, Stephen Gilmore and Corrado Priami

Overview

- Introduction
- Mapping PEPA nets to Stochastic CCS
  - PEPA nets
  - Stochastic CCS
  - Example
- Translation functions
- Future Work and Conclusions

Stochastic Process Algebra

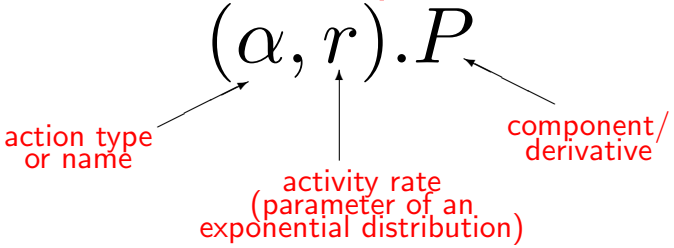
Attractive features of process algebras + Quantification

- Compositionality
- Formal definition
- Parsimony

- Actions have duration
- Probabilistic branching

Performance Evaluation Process Algebra (PEPA)

Models are constructed from components who engage in activities



PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA

$$S ::= (\alpha, r).S \mid S + S \mid A$$
$$P ::= S \mid P \underset{L}{\bowtie} P \mid P/L$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competition between components (race policy)
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*)
		$\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$
CONSTANT:	$A \stackrel{def}{=} S$	assigning names to components

PEPA nets (1)

- The *PEPA nets* formalism uses the stochastic process algebra PEPA as the inscription language for coloured Petri nets.

- The combination naturally represents applications with two classes of change of state (*global* and *local*).

- Firings at the net level represent global state changes, while PEPA transitions represent local state changes.

- Petri nets provide a graphical presentation of a model which has an easily accessible interpretation and they also have the advantage of being supported by an unambiguous formal interpretation. In contrast stochastic process algebras have an explicit compositional structure.

PEPA nets (2)

Net structure The net structure is represented in the usual way (state machines).

Tokens The tokens are PEPA components, which have a cyclic behaviour, made up of activities taken from two distinct alphabets: transitions and firings.

Places Each place is a PEPA context consisting of static components and cells. Cells are typed by the alphabet of the tokens which may be placed in the cell.

Initial Marking Which cells are occupied in the starting state.

Firing Rule A net level transition may fire if there is a token in the input place offering the firing action of the correct name and there is a unoccupied cell of the correct type in the output place.

The PEPA nets language enforces the property that communication between components at different places is not allowed — remote communication must be implemented via a combination of migration and local communication.

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C\|C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,\big(X = C\big)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C\|C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,(X = C)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C \vert C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,\big(X = C\big)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C \vert C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,(X = C)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C\|C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,(X = C)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C \vert C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,(X = C)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C \vert C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,(X = C)$	recursive process, where X is the recursion variable

Stochastic CCS (CCS_S)

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

NIL:	$\mathbf{0}$	
PREFIX:	$(\alpha, r).C$	designated first action
CHOICE:	$C + C$	competition between components (race policy)
COMPOSITION:	$C\|C$	parallel composition of processes
RESTRICTION:	$C \setminus L$	internalises the names in L and their complements.
RECURSION:	$\mathbf{fix}\,\big(X = C\big)$	recursive process, where X is the recursion variable

Benefits of the Mapping

We explore the relationship between coloured stochastic Petri net and process algebras by mapping PEPA nets into the foundational process algebra, Milner's CCS enhanced with timing information (CCS$_S$).

- The communication rules of PEPA nets 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.

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

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

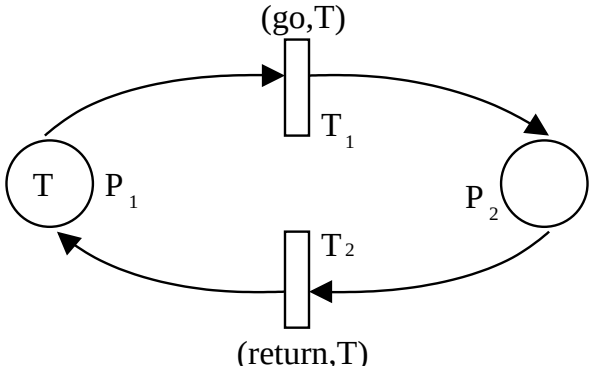
Quality of the Mapping

- The aim is to provide a mapping which respects the construction of the PEPA net, i.e. it would be unsatisfactory to trivially expand the given PEPA net to its full state space and then translate this into a sequential CCS_S term.

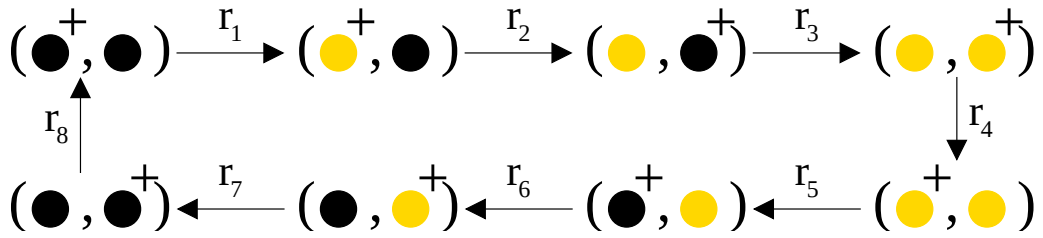
- We seek a compositional mapping in which the size of the CCS_S description which we generate is proportional to the size of the description of the input PEPA net, not proportional to the size of its state space.

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

Small Example: lights on and off



Consider a net with two places and a single token which circulates between the places, starting 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.



Example as a PEPA net

$$
\begin{aligned}
Token &\stackrel{def}{=} (on, r_1).(\mathbf{go}, r_2).(on, r_3).(\mathbf{return}, r_4). \\
&\quad\ (off, r_5).(\mathbf{go}, r_6).(off, r_7).(\mathbf{return}, r_8).Token \\
Switch &\stackrel{def}{=} (on, \top).(off, \top).Switch \\
P_1[t] &\stackrel{def}{=} Switch \underset{\{on, off\}}{\bowtie} Token[t] \\
P_2[t] &\stackrel{def}{=} Switch \underset{\{on, off\}}{\bowtie} Token[t] \\
System &\stackrel{def}{=} (P_1[Token], P_2[\_])
\end{aligned}
$$

Example in CCS$_S$

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

Example in CCS$_S$

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

Example in CCS$_S$

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

Example in CCS$_S$

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

Example in CCS$_S$

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

Example in CCS$_S$

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

Example in CCS$_S$

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

The mapping (1)

- In order to keep information about the net structure within the flat process algebra structure we introduce differentiated names for activities, to indicate the location within which an activity takes place. Thus an on activity occurring in place P_1 becomes, on translation, $on@1$.

- Similarly we need to represent the structure of the net (from each token's perspective), as another process algebra component, which can be thought of as storing the itinerary of the token.

- The mapping is compositional because we do form a component for each place, and two components for each token, reflecting the structure of the PEPA net description.

The mapping (2)

Tokens Each token gives rise to two tightly-coupled CCS_S processes: one encodes the behaviour of the token, the other encodes the locations in which the token may be found.

Static Components By their nature these components do not participate in firings, and the transitions can be straightforwardly translated.

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

1. the instantiation of the static components of the place, which involves specialising local transition names; and
2. the representation of the arcs from this place to other places as a collection of process definitions in which only firing activities are performed.

Initial Marking The right choice of names must be made so that specialised activities reflecting the initial marking are enabled in the starting state.

Current limitations of the mapping

- Various limitations must be placed on the PEPA nets considered in the mapping to avoid the use of multiway synchronisation which cannot be represented by the CCS composition operator. For example, we restrict that tokens must be sequential components.

- Since we wish to impose the physical restriction on communication which is fundamental to PEPA nets, it is essential that activities which might arise in different places of the net are differentiated as explained earlier. Moreover it is necessary to consider PEPA nets in which all tokens are different.

- Restrictions are also placed on the use of the PEPA hiding operator within places. These are needed to make a mapping to the restriction operator possible (names by which tokens interact cannot be hidden).

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (1)

The encoding function $\mathcal{E} : Pnet \rightarrow CCS_S$ is expressed by: $\mathcal{E}_\sigma(M) = P_{CCS}$ where $\sigma = \langle S, T, F, D \rangle$ stores information about the $Pnet$.

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

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

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{{}^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \{\!|\, \bigcup_{j=1}^{M} \text{spec } A_j \ P_i \ \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{^{x@\text{new}()}/_{x@i}\} \ \forall\, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$

$$\text{where } L' = \{x@i \mid x \in (L \setminus \{\!|\, \bigcup_{j=1}^{M} \text{spec } A_j \ P_i \ \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{^{x@\text{new}()}/_{x@i}\} \ \forall\, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L' \quad \text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{{}^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{{}^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Functional encoding of the mapping (2)

$$\mathcal{ES}^i_\sigma(P \underset{L}{\bowtie} R) = (\mathcal{ES}^i_\sigma(P) \mid \mathcal{ES}^i_\sigma(R)) \setminus L'$$
$$\text{where } L' = \{x@i \mid x \in (L \setminus \lfloor \rfloor_{j=1}^{M} \text{spec } A_j \; P_i \; \emptyset)\}$$

$$\mathcal{ES}^i_\sigma(P/L) = \mathcal{ES}^i_\sigma(P)\{{}^{x@\text{new}()}/_{x@i}\} \; \forall \, x \in L$$

$$\mathcal{ES}^i_\sigma(I[C]) = \mathbf{0}$$

$$\mathcal{ES}^i_\sigma(S_1 + S_2) = \mathcal{ES}^i_\sigma(S_1) + \mathcal{ES}^i_\sigma(S_2)$$

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

$$\mathcal{ES}^i_\sigma((\alpha, r).S) = \begin{cases} (\alpha@i, 1).\mathcal{ES}^i_\sigma(S) & \text{if } r = \top \\ (\overline{\alpha@i}, r).\mathcal{ES}^i_\sigma(S) & \text{otherwise} \end{cases}$$

Conclusions

- A compositional mapping which gives rise to isomorphic continuous time Markov chains has been developed.
- Representation of mobility in PEPA nets and the stochastic π-calculus are orthogonal.
- Several limitations, of varying restrictions needed to be placed on the source language in order to map into the CCS-style target. However these reflect basic process algebra differences of style (complementary actions vs multiway synchronisation) rather than the key elements, PEPA net communication patterns, which we were wishing to study.
- The translation to CCS_S has formed the basis of a compiler which allows us to use the PRISM model-checker to verify properties of PEPA net models.