PEPA nets:

a dynamic stochastic modelling formalism

Stephen Gilmore
Laboratory for Foundations of Computer Science
The University of Edinburgh

Joint work with Jane Hillston, Leïla Kloul and Marina Ribaudo
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

• Performance modelling and PEPA
• Markov chains and performance modelling
• Petri nets and process algebras
• PEPA nets: transitions, firings, choices, markings
• Semantics of PEPA nets
• Example: a mobile agent system
• Relationship to other modelling formalisms
• Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

• Performance modelling and PEPA
• Markov chains and performance modelling
• Petri nets and process algebras
• **PEPA nets: transitions, firings, choices, markings**
• Semantics of PEPA nets
• Example: a mobile agent system
• Relationship to other modelling formalisms
• Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

• Performance modelling and PEPA
• Markov chains and performance modelling
• Petri nets and process algebras
• PEPA nets: transitions, firings, choices, markings
• Semantics of PEPA nets
• Example: a mobile agent system
• Relationship to other modelling formalisms
• Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Structure of this talk

- Performance modelling and PEPA
- Markov chains and performance modelling
- Petri nets and process algebras
- PEPA nets: transitions, firings, choices, markings
- Semantics of PEPA nets
- Example: a mobile agent system
- Relationship to other modelling formalisms
- Future work and conclusions
Performance modelling

- Despite impressive improvements in the computational power which is now available to end-users of computer systems, computer equipment is still expensive to purchase and maintain and it is important to make cost-effective use of computing resources.

- The analysis of computer systems through construction and solution of descriptive models is a hugely profitable activity: brief analysis of a model can provide as much insight as many hours of simulation and measurement.

- Simple models of a computer system can be constructed without any explicit notational support. However, as computer systems become more complex so do their models and the use of a high-level language to aid in their expression becomes essential.
**Performance Evaluation Process Algebra (PEPA)**

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r) \cdot P\]

- The language is used to generate a CTMC for performance modelling.
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r).P\]

- The language is used to generate a CTMC for performance modelling.
Performance Evaluation Process Algebra (PEPA)

• Models are constructed from *components* which engage in *activities*.

\[(\alpha, r). P\]

• The language is used to generate a CTMC for performance modelling.
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r). P\]

- The language is used to generate a CTMC for performance modelling.
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r).P\]

- The language is used to generate a CTMC for performance modelling.
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[ (\alpha, r) \cdot P \]

- The language is used to generate a CTMC for performance modelling.

PEPA MODEL

Stephen Gilmore  
PEPA nets — DEGAS project  
Rovereto, 12th February 2003
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r).P,\]

- The language is used to generate a CTMC for performance modelling.

PEPA MODEL \[\xrightarrow{\text{SOS rules}}\] PEPA nets — DEGAS project
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

- The language is used to generate a CTMC for performance modelling.

\[ (\alpha, r).P \]

**PEPA MODEL** \[\rightarrow\] **SOS rules** \[\rightarrow\] **LABELLED MULTI-TRANSITION SYSTEM**
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r).P\]

- The language is used to generate a CTMC for performance modelling.

PEPA MODEL \[\xrightarrow{\text{SOS rules}}\] LABELLED MULTI-TRANSITION SYSTEM \[\xrightarrow{\text{state transition diagram}}\]
Performance Evaluation Process Algebra (PEPA)

- Models are constructed from *components* which engage in *activities*.

\[(\alpha, r).P\]

- The language is used to generate a CTMC for performance modelling.

PEPA MODEL \(\xrightarrow{\text{SOS rules}}\) LABELLED MULTI-TRANSITION SYSTEM \(\xrightarrow{\text{state transition diagram}}\) CONTINUOUS TIME MARKOV CHAIN \(Q\)

Stephen Gilmore
PEPA nets — DEGAS project
Rovereto, 12th February 2003
Markov chains and performance modelling

- For finite-state PEPA models whose derivation graph is strongly connected (an **ergodic Markov process**) the equilibrium probability distribution of the model, \( \Pi \), is found by solving the matrix equation

\[
\Pi Q = 0 \quad \text{where} \quad \sum \Pi = 1
\]

- Performance measures are derived by defining *reward structures* over a model. The performance measure is then defined as the total reward computed from the steady state probability distribution. I.e. if \( \rho_i \) is the reward associated with \( C_i \), the total reward \( R \) is

\[
R = \sum_i \rho_i \Pi(C_i)
\]
Performance results

Stephen Gilmore

PEPA nets — DEGAS project

Rovereto, 12th February 2003
Petri nets and process algebras

• 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.

• Stochastic process algebras lack the attractive graphical presentation of Petri nets. In contrast though, an *explicit compositional structure* is imposed on the model. This structure can be exploited for both qualitative and quantitative analysis.

• The present work considers using both Petri nets and process algebras together as a single, *structured* performance modelling formalism.
PEPA nets

• *Coloured Petri nets* are a high-level form of classical Petri nets. The plain (indistinguishable) tokens of a classical Petri net are replaced by arbitrary terms which are distinguishable.

• In *stochastic Petri nets* the transitions from one marking to another are associated with a random variable drawn from an exponential distribution. Here we consider *coloured stochastic Petri nets* where the colours used as the tokens of the net are PEPA components. We refer to these as *PEPA nets*.

• Petri nets have previously been combined with other performance modelling formalisms e.g. Bause’s *Queueing Petri nets* and Haverkort’s *Dynamic Queueing Networks*. Other extensions of (non-stochastic) Petri nets have programmable tokens, e.g. Valk’s *Elementary Object systems*. 
Transitions in a PEPA net

• A *transition* in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in co-operation with another component at the same place).

\[(\beta, r).P \rightarrow\]

• Transitions of PEPA components are used to model small-scale changes of state as components undertake activities. The PEPA net formalism does not allow components at different places in the net to co-operate on a shared activity so transitions have only *local* effect.
Transitions in a PEPA net

- A transition in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in co-operation with another component at the same place).

- Transitions of PEPA components are used to model small-scale changes of state as components undertake activities. The PEPA net formalism does not allow components at different places in the net to co-operate on a shared activity so transitions have only local effect.
Transitions in a PEPA net

- A *transition* in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in co-operation with another component at the same place).

- Transitions of PEPA components are used to model small-scale changes of state as components undertake activities. The PEPA net formalism does not allow components at different places in the net to co-operate on a shared activity so transitions have only *local* effect.
Transitions in a PEPA net

• A *transition* in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in co-operation with another component at the same place).

![Diagram of a transition in a PEPA net]

• Transitions of PEPA components are used to model small-scale changes of state as components undertake activities. The PEPA net formalism does not allow components at different places in the net to co-operate on a shared activity so transitions have only *local* effect.
Firings in a PEPA net

- *Firings* in a PEPA net are used to model macro-step changes of state such as breakdowns and repairs, one thread yielding to another, or a mobile software agent moving from one network host to another.

\[(\alpha, r).P \rightarrow \rightarrow \]

- A firing causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in both the *input place* (where existing co-operations now can no longer take place) and the *output place* (where previously disabled co-operations are now enabled).
Firings in a PEPA net

- *Firings* in a PEPA net are used to model macro-step changes of state such as breakdowns and repairs, one thread yielding to another, or a mobile software agent moving from one network host to another.

\[
(\alpha, r).P \rightarrow \rightarrow
\]

- A firing causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in both the *input place* (where existing co-operations now can no longer take place) and the *output place* (where previously disabled co-operations are now enabled).
Firings in a PEPA net

• *Firings* in a PEPA net are used to model macro-step changes of state such as breakdowns and repairs, one thread yielding to another, or a mobile software agent moving from one network host to another.

![Diagram of firings in a PEPA net]

• A firing causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in both the *input place* (where existing co-operations now can no longer take place) and the *output place* (where previously disabled co-operations are now enabled).
Firings in a PEPA net

- **Firings** in a PEPA net are used to model macro-step changes of state such as breakdowns and repairs, one thread yielding to another, or a mobile software agent moving from one network host to another.

- A firing causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in both the **input place** (where existing co-operations now can no longer take place) and the **output place** (where previously disabled co-operations are now enabled).
Choice in a PEPA net

• Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

• The outcome of a choice is governed by a race condition.
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a race condition.
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a *race condition*. 

![Diagram of PEPA net showing choices and transitions]

---

Stephen Gilmore

PEPA nets — DEGAS project

Rovereto, 12th February 2003
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a *race condition*.
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a race condition.
Choice in a PEPA net

• Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

• The outcome of a choice is governed by a race condition.
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a race condition.
Choice in a PEPA net

- Choices in a PEPA net occur when the token has a choice of possible behaviours and a choice of possible output places. Choices are used to model decisions in a system.

- The outcome of a choice is governed by a race condition.
Markings in a PEPA net

- The marking of a PEPA net is made up of a list of PEPA contexts, one at each place in the net, where the system descriptions can also contain component cells.

- A cell is a slot to be filled by a component of a particular type.
  - Components which fill these cells can circulate as the tokens of the net.
  - Components which are not in a cell are static and cannot move.

- We use the notation $P[-]$ to denote a context or a place which could be filled by the PEPA component $P$ or one with the same alphabet.
Markings in a PEPA net

• The marking of a PEPA net is made up of a list of PEPA contexts, one at each place in the net, where the system descriptions can also contain component cells.

• A cell is a slot to be filled by a component of a particular type.
  – Components which fill these cells can circulate as the tokens of the net.
  – Components which are not in a cell are static and cannot move.

• We use the notation $P[\_]$ to denote a context or a place which could be filled by the PEPA component $P$ or one with the same alphabet.
Markings in a PEPA net

• The marking of a PEPA net is made up of a list of PEPA contexts, one at each place in the net, where the system descriptions can also contain component cells.

• A cell is a slot to be filled by a component of a particular type.
  – Components which fill these cells can circulate as the tokens of the net.
  – Components which are not in a cell are static and cannot move.

• We use the notation $P[-]$ to denote a context or a place which could be filled by the PEPA component $P$ or one with the same alphabet.
Markings in a PEPA net

- The marking of a PEPA net is made up of a list of PEPA contexts, one at each place in the net, where the system descriptions can also contain component cells.

- A cell is a slot to be filled by a component of a particular type.
  - Components which fill these cells can circulate as the tokens of the net.
  - Components which are not in a cell are static and cannot move.

- We use the notation $P[-]$ to denote a context or a place which could be filled by the PEPA component $P$ or one with the same alphabet.
The syntax of PEPA

\[ S ::= (\text{sequential components}) \]
\[ (\alpha, r).S \quad (\text{prefix}) \]
\[ S + S \quad (\text{choice}) \]
\[ I \quad (\text{identifier}) \]

\[ P ::= (\text{model components}) \]
\[ P \boxdot L P \quad (\text{cooperation}) \]
\[ P/L \quad (\text{hiding}) \]
\[ I \quad (\text{identifier}) \]
The syntax of PEPA with cells

\[ S ::= (\text{sequential components}) \]
\[ (\alpha, r).S \quad \text{(prefix)} \]
\[ S + S \quad \text{(choice)} \]
\[ I \quad \text{(identifier)} \]

\[ P ::= (\text{model components}) \]
\[ P \Join P \quad \text{(cooperation)} \]
\[ P/L \quad \text{(hiding)} \]
\[ I \quad \text{(identifier)} \]
\[ P[C] \quad \text{(cell)} \]
The syntax of PEPA with cells

\[
S ::= (\text{sequential components}) \\
\quad (\alpha, r).S \quad \text{ (prefix)} \\\n\quad S + S \quad \text{ (choice)} \\\n\quad I \quad \text{ (identifier)}
\]

\[
P ::= (\text{model components}) \\
\quad \text{model components} \quad \text{ (cooperation)} \\\n\quad P \times P \quad \text{ (cooperation)} \\\n\quad P \parallel P \quad \text{ (cooperation)} \\\n\quad \text{hiding} \quad \text{ (hiding)} \\\n\quad P \quad \text{ (identifier)} \\\n\quad P[C] \quad \text{ (cell)}
\]

\[
C ::= (\text{cell terms}) \\
\quad ' ' \quad \text{ (empty)} \\\n\quad P \quad \text{ (component)}
\]
Semantics

• The PEPA language is formally defined by a \textit{structured operational semantics} as used in the definition of Milner’s CCS and other process algebras.

• In order to describe the firing rule for PEPA nets formally we need a \textit{relational operator} which is to be used to express the fact that there exists a particular transition in the net.

• We use the notation

\[
P_1 \xrightarrow{(\alpha, r)} P_2
\]

to capture the information that there is an arc connecting input place \(P_1\) to a transition labelled by \((\alpha, r)\) which leads to output place \(P_2\).
Semantics

- The PEPA language is formally defined by a *structured operational semantics* as used in the definition of Milner’s CCS and other process algebras.

- In order to describe the firing rule for PEPA nets formally we need a *relational operator* which is to be used to express the fact that there exists a particular transition in the net.

- We use the notation

\[ P_1 \xrightarrow{(\alpha, r)} P_2 \]

to capture the information that there is an arc connecting input place \( P_1 \) to a transition labelled by \((\alpha, r)\) which leads to output place \( P_2 \).
Semantics

- The PEPA language is formally defined by a *structured operational semantics* as used in the definition of Milner’s CCS and other process algebras.

- In order to describe the firing rule for PEPA nets formally we need a *relational operator* which is to be used to express the fact that there exists a particular transition in the net.

- We use the notation

\[ P_1 \xrightarrow{(\alpha, r)} P_2 \]

\[ P_1 \xrightarrow{(\alpha, r)} P_2 \]

to capture the information that there is an arc connecting input place \( P_1 \) to a transition labelled by \((\alpha, r)\) which leads to output place \( P_2 \).
Semantics

- The PEPA language is formally defined by a *structured operational semantics* as used in the definition of Milner’s CCS and other process algebras.

- In order to describe the firing rule for PEPA nets formally we need a *relational operator* which is to be used to express the fact that there exists a particular transition in the net.

- We use the notation

  \[ P_1 \xrightarrow{(\alpha, r)} P_2 \]

  to capture the information that there is an arc connecting input place \( P_1 \) to a transition labelled by \((\alpha, r)\) which leads to output place \( P_2 \).
Semantics

Cell:

\[
P' \xrightarrow{(\alpha, r)} P'' \quad \text{(} P = a P' \text{)}
\]

\[
P[P'] \xrightarrow{(\alpha, r)} P[P'']
\]

Transition:

\[
P \xrightarrow{(\alpha, r)} P' \quad \text{(} \ldots, P, \ldots \text{)} \xrightarrow{(\alpha, r)} (\ldots, P', \ldots)
\]

\[
P \xrightarrow{(\alpha, r')} Q \quad \text{(} \ldots, P, \ldots \text{)} \xrightarrow{(\alpha, r')} (\ldots, P', \ldots)
\]

Firing:

\[
Q \xrightarrow{(\alpha, r_1)} Q' \quad P_i \xrightarrow{(\alpha, r_2)} P_j
\]

\[
(\ldots, P_i[Q], \ldots, P_j[-], \ldots) \xrightarrow{(\alpha, R)} (\ldots, P_i[-], \ldots, P_j[Q'], \ldots)
\]
Example: a mobile agent system

• In this example *a roving agent* visits three sites. It interacts with static software components at these sites and has two kinds of interactions.

• When visiting a site where *a network probe* is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.

• When it returns to the central co-ordinating site it dumps the data which it has harvested to *the master probe*. The master probe performs a computationally expensive statistical analysis of the data.

• The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.
Example: a mobile agent system

- In this example a roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions.

- When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.

- When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data.

- The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.
Example: a mobile agent system

- In this example a roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions.

- When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.

- When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data.

- The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.
Example: a mobile agent system

• In this example a roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions.

• When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.

• When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data.

• The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.
Example: a mobile agent system

- In this example a roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions.

- When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.

- When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data.

- The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.
PEPA components

\[\begin{align*}
\text{Agent} & \overset{\text{def}}{=} (\text{go}, \lambda).\text{Agent}' \\
\text{Agent}' & \overset{\text{def}}{=} (\text{interrogate}, r_i).\text{Agent}'' \\
\text{Agent}'' & \overset{\text{def}}{=} (\text{return}, \mu).\text{Agent}''' \\
\text{Agent}''' & \overset{\text{def}}{=} (\text{dump}, r_d).\text{Agent} \\
\text{Master} & \overset{\text{def}}{=} (\text{dump}, \top).\text{Master}' \\
\text{Master}' & \overset{\text{def}}{=} (\text{analyse}, r_d).\text{Master} \\
\text{Probe} & \overset{\text{def}}{=} (\text{monitor}, r_m).\text{Probe} + (\text{interrogate}, \top).\text{Probe}
\end{align*}\]
A mobile agent system (1)

\[ Agent \overset{\text{def}}{=} (\text{go}, \lambda).Agent' \]
A mobile agent system (2)

Agent' \overset{\text{def}}{=} (\text{interrogate}, r_i).\text{Agent''}
A mobile agent system (3)

\[
\begin{align*}
\text{Agent''} & \overset{\text{def}}{=} (\text{return}, \mu).\text{Agent''''}
\end{align*}
\]
A mobile agent system (4)

\[ \text{Agent}''' = (\text{dump}, r_d).\text{Agent} \]
A mobile agent system (5)

\[
\text{Agent} \overset{\text{def}}{=} (\text{go}, \lambda).\text{Agent}'
\]

\[
\begin{aligned}
&P_1 \quad (\text{go}, \lambda_l) \quad T_1 \\
&P_2 \quad A \quad (\text{go}, \lambda_r) \\
&P_3

\]

\[
\begin{aligned}
&T_2 \quad (\text{return}, \mu_l) \\
&T_3 \\
&T_4 \quad (\text{return}, \mu_r)
\end{aligned}
\]
A mobile agent system (6)

\[
\begin{align*}
P_1 & \quad A' \quad (go, \lambda_l) \quad T_1 \quad P_2 \\
 & \quad T_2 \quad (return, \mu_l) \\
& \quad T_4 \quad (return, \mu_r) \\
& \quad T_3 \quad (go, \lambda_r) \\
& \quad P_3
\end{align*}
\]

\[
Agent' \overset{\text{def}}{=} (\text{interrogate}, r_i).Agent''
\]
A mobile agent system (7)

Agent''' $\overset{\text{def}}{=} (\text{return}, \mu).\text{Agent'''}$
A mobile agent system (8)

\[
\begin{array}{c}
\text{Agent}''' \overset{\text{def}}{=} (\text{dump}, r_d).\text{Agent}
\end{array}
\]
A mobile agent system (9)

Agent \overset{\text{def}}{=} (\text{go}, \lambda).\text{Agent}'
Relationship to other modelling formalisms

- If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets, stochastic Petri nets and the PEPA stochastic process algebra would be considered to be equally expressive.

  - For a given \( \text{CTMC}, C \), it is possible to construct a high-level model in each of these three formalisms such that the underlying CTMC derived from the model is isomorphic to \( C \).

- In practice, the three languages present different sets of conceptual tools to the modeller so we can compare modelling with PEPA nets with modelling with Petri nets and the PEPA stochastic process algebra.
Relationship to other modelling formalisms

• If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets, stochastic Petri nets and the PEPA stochastic process algebra would be considered to be equally expressive.

  – For a given CTMC, $C$, it is possible to construct a high-level model in each of these three formalisms such that the underlying CTMC derived from the model is isomorphic to $C$.

• In practice, the three languages present different sets of conceptual tools to the modeller so we can compare modelling with PEPA nets with modelling with Petri nets and the PEPA stochastic process algebra.
Relationship to other modelling formalisms

- If they were to be viewed purely formally as high-level description languages for specifying continuous-time Markov chains, then PEPA nets, stochastic Petri nets and the PEPA stochastic process algebra would be considered to be equally expressive.

  – For a given CTMC, $C$, it is possible to construct a high-level model in each of these three formalisms such that the underlying CTMC derived from the model is isomorphic to $C$.

- In practice, the three languages present different sets of conceptual tools to the modeller so we can compare modelling with PEPA nets with modelling with Petri nets and the PEPA stochastic process algebra.
Mapping Petri nets to PEPA nets

• In a classical stochastic Petri net tokens are indistinguishable.
  – Have a single class of tokens with only one state.

• Petri net tokens should always permit firings of the net to take place.
  – Sum over all of the transition activity names in the net.

• Classical stochastic Petri nets have no immovable tokens.
  – Make no use of static components in the places in the net.

\[
\text{Token} = \sum_{t_n \in T} (t_n, \top).
\]

\[
\text{Token}_{P_i}[t_{k_1}, \ldots, t_{k_k}] \overset{\text{def}}{=} \text{Token}_{[t_{k_1}} \parallel \cdots \parallel \text{Token}_{[t_{k_k}].}
\]
Mapping Petri nets to PEPA nets

- In a classical stochastic Petri net tokens are indistinguishable.
Mapping Petri nets to PEPA nets

- In a classical stochastic Petri net tokens are indistinguishable.
  - Have a single class of tokens with only one state.
Mapping Petri nets to PEPA nets

- In a classical stochastic Petri net tokens are indistinguishable.
  - *Have a single class of tokens with only one state.*

- Petri net tokens should always permit firings of the net to take place.
Mapping Petri nets to PEPA nets

- In a classical stochastic Petri net tokens are indistinguishable.
  - *Have a single class of tokens with only one state.*

- Petri net tokens should always permit firings of the net to take place.
  - *Sum over all of the transition activity names in the net.*
Mapping Petri nets to PEPA nets

- In a classical stochastic Petri net tokens are indistinguishable.
  - Have a single class of tokens with only one state.

- Petri net tokens should always permit firings of the net to take place.
  - Sum over all of the transition activity names in the net.

- Classical stochastic Petri nets have no immovable tokens.
Mapping Petri nets to PEPA nets

• In a classical stochastic Petri net tokens are indistinguishable.
  – Have a single class of tokens with only one state.

• Petri net tokens should always permit firings of the net to take place.
  – Sum over all of the transition activity names in the net.

• Classical stochastic Petri nets have no immovable tokens.
  – Make no use of static components in the places in the net.
Mapping Petri nets to PEPA nets

• In a classical stochastic Petri net tokens are indistinguishable.
  – *Have a single class of tokens with only one state.*

• Petri net tokens should always permit firings of the net to take place.
  – *Sum over all of the transition activity names in the net.*

• Classical stochastic Petri nets have no immovable tokens.
  – *Make no use of static components in the places in the net.*

\[
\text{Token} \overset{\text{def}}{=} \sum_{t_n_i \in T} (t_n_i, \top). \text{Token}
\]
Mapping Petri nets to PEPA nets

• In a classical stochastic Petri net tokens are indistinguishable.
  – *Have a single class of tokens with only one state.*

• Petri net tokens should always permit firings of the net to take place.
  – *Sum over all of the transition activity names in the net.*

• Classical stochastic Petri nets have no immovable tokens.
  – *Make no use of static components in the places in the net.*

\[
\text{Token} \overset{\text{def}}{=} \sum_{tn_i \in T} (tn_i, \top) \cdot \text{Token} \\
\text{P}_i[tk_1, \ldots, tk_k] \overset{\text{def}}{=} \text{Token}[tk_1] \parallel \cdots \parallel \text{Token}[tk_k]
\]
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward. A PEPA net with one place and no transitions is a PEPA model.

- As $p$ travels right, it moves out of the scope of the co-operation set $L$.

- Actions previously in co-operation can now be performed individually.
Relating PEPA nets to PEPA

• The relationship between PEPA nets and PEPA is straightforward
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A *PEPA net with one place and no transitions is a PEPA model.*
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A PEPA net with one place and no transitions is a PEPA model.
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A PEPA net with one place and no transitions is a PEPA model.
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A PEPA net with one place and no transitions is a PEPA model.
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A PEPA net with one place and no transitions is a PEPA model.
Relating PEPA nets to PEPA

- The relationship between PEPA nets and PEPA is straightforward
  - A PEPA net with one place and no transitions is a PEPA model.

- As $p$ travels right it moves out of the scope of the co-operation set $L$. 
Relating PEPA nets to PEPA

• The relationship between PEPA nets and PEPA is straightforward
  – A PEPA net with one place and no transitions is a PEPA model.

\[
\begin{align*}
  P[-] & \mathbin{\boxtimes}_L Q \\
  \rightarrow \quad (g \circ r) \quad \rightarrow \\
  & \quad P[p]
\end{align*}
\]

• As \( p \) travels right it moves out of the scope of the co-operation set \( L \).

• Actions previously in co-operation can now be performed individually.
Implementation

- The PEPA stochastic process algebra is supported by a range of tools including the *PEPA Workbench*, the *Möbius Modelling Framework*, and *PRISM*. We have extended the PEPA Workbench to implement PEPA nets.

- Möbius implements an extension of PEPA called *PEPA*$_k$. PEPA$_k$ extends PEPA with formal parameters, guards and integer value passing. We can encode a PEPA net cell in place $j$ as a sum of PEPA$_k$ guarded components as shown below. A similar translation could work for PRISM.

\[
\begin{align*}
& P[j] \\
& \text{PEPA net cell at place } j, \\
& \text{tokens } = \{ P_0, \ldots, P_n \} \\
\sim \sum_{i=0}^{n} [(\text{place } = j) \land (\text{colour } = i)] \Rightarrow P_i \\
& \text{PEPA}_k \text{ expression in atomic submodel } j
\end{align*}
\]
Future work

- PEPA net models are currently solved using traditional sparse matrix methods such as the preconditioned biconjugate gradient method but other solution methods might be appropriate for PEPA nets such as time-scale decomposition.

- We are currently investigating the relationship between PEPA nets and other modelling formalisms such as Priami’s Stochastic $\pi$-calculus. We have developed an enhanced operational semantics for PEPA nets and (together with Linda Brodo) an embedding of PEPA nets in the Stochastic $\pi$-calculus.

- PEPA net models lend themselves readily to decomposition methods so a possible future direction of work is mapping PEPA nets into the Kronecker representation format of the APNN Toolbox. This could provide an illuminating comparison with the MTBDD representation used by PRISM.
Conclusions

• The PEPA nets formalism is new and, as yet, relatively unproven. It is our belief that it can provide a suitable framework for the description of performance models of systems which have *distinct notions of changes of state* which we represent by transitions and firings.

• Our experience with PEPA has been that the combination of a well-defined *formal semantics* for the language and *a range of tools* for the language enabled us and others to use it effectively. By following a similar development path we would hope that the PEPA nets formalism could also prove to be useful.
Acknowledgements

This work has been supported by the DEGAS (Design Environments for Global ApplicationS) project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing.