Performance modelling
with PEPA nets and PRISM

S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo
Outline

- PEPA nets: informal introduction
- PEPA nets: few formal definitions
- Simple example: mobile agent
- From PEPA nets to PRISM
- Complex example: mobile IP
PEPA nets: informal introduction

Stochastic (coloured) Petri Nets

PEPA

PEPA nets
PEPA nets: informal introduction

The tokens are PEPA components that perform local activities and can move from one place to another ...

Transition names are labelled with activity names
PEPA nets: informal introduction

- PEPA components perform activities to represent state changes

- ... in PEPA nets we distinguish between two types of changes ...

  ✓ “local” changes
    (transitions of PEPA components)

  ✓ “global” changes
    (net firings)
PEPA nets: informal introduction

Firing?
Who moves?

Local changes

(go, λ)
(return, μ)
PEPA nets: informal introduction

- There is a PEPA context at each place of the net

- A PEPA context consists of
  - static components
  - cells [ ]
PEPA nets: informal introduction

Local changes

Q [Q] ⊗ R

Q [Q'] ⊗ R'

Q [_] ⊗ R''

P1

(go, λ)

P2

(return, μ)

Q [Q'']
PEPA nets: informal introduction

- Some assumptions

✓ Components can cooperate only when they are in the same place

✓ It is NOT possible for one component to cooperate with another component AND transfer to another place
**PEPA nets: syntax**

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

\[
P ::= P \bowtie_L P \quad \text{(cooperation)} \\
| \quad P/L \quad \text{(hiding)} \\
| \quad P[C] \\
| \quad I
\]

\[
C ::= \_ \quad \text{(empty)} \\
| \quad S \quad \text{(full)}
\]
PEPA nets: markings and places

\[ M \ ::= \ (M_P, \ldots) \quad (\text{marking}) \]
\[ M_P \ ::= \ P[C, \ldots] \quad (\text{place marking}) \]

\[ P[C, \ldots] \overset{\text{def}}{=} P[C] \otimes L P \quad (\text{place defn}) \]
Simple example: mobile agent

- A mobile software agent visits three sites, where it interacts with static software components

- In two sites the agent interrogates a network sensor for data (on recent patterns of network traffic)

- In the other site, the agent dumps the data to a master sensor
Simple example: Mobile agent

Agent\[\_\]_{L_1}^\text{Master}

P1

(go,T)

P2

(go,T)

P3

Agent\[\_\]_{L_2}^\text{Probe}

Agent\[\_\]_{L_2}^\text{Probe}
Simple example: Mobile agent

\[
\begin{align*}
\text{Agent} & \overset{\text{def}}{=} (\text{go}, \lambda).\text{Agent'} \\
\text{Agent'} & \overset{\text{def}}{=} (\text{interrogate}, r_i).\text{Agent'’} \\
\text{Agent’’} & \overset{\text{def}}{=} (\text{return}, \mu).\text{Agent’’’} \\
\text{Agent’’’} & \overset{\text{def}}{=} (\text{dump}, r_d).\text{Agent}
\end{align*}
\]

\[
\begin{align*}
\text{Master} & \overset{\text{def}}{=} (\text{dump}, T).\text{Master'} \\
\text{Master'} & \overset{\text{def}}{=} (\text{analyse}, r_a).\text{Master}
\end{align*}
\]

\[
\begin{align*}
\text{Probe} & \overset{\text{def}}{=} (\text{monitor}, r_m).\text{Probe} + \\
& (\text{interrogate}, T).\text{Probe}
\end{align*}
\]
PRISM

- **Probabilistic model checker**
  probabilistic temporal logic, PCTL and CSL

- **Supports three models**
  DTMC, MDP, CTMC

- **Compact state representation** (BDD)

- **Input to the PRISM tool**
  1. description of the system
  2. set of properties to be checked
A compiler exists for translating PEPA models (a subset of PEPA) into PRISM models

... then the models can be analysed with the PRISM tool

✓ The steady-state probability distribution for the underlying CTMC can be automatically derived

✓ Properties can be verified
From PEPA nets to PRISM

PEPA net description

PRISM input files

PRISM description

PRISM output
From PEPA nets to PEPA

- We need to map the net structure into (standard) PEPA components

- Problems

  ✓ What happens to different transitions with the same associated label?

  ✓ What happens to replica of the same static component, resident in different places

  ✓ What about cells?
Steps of the translation algorithm

0. Preprocessing
1. Translation of static components

In order to avoid wrong synchronisations we need to distinguish replicas of the same static component.

This is done by renaming action types and derivatives.
2. Translation of cells
A new PEPA component need to be defined for each cell \( i \) within each place \( P \)

\[
\begin{align*}
\text{Cell}_{i0} & \overset{\text{def}}{=} (a,r_1).\text{Cell}_{i1} \\
\text{Cell}_{i1} & \overset{\text{def}}{=} (b,r_2).\text{Cell}_{i0}
\end{align*}
\]
3. Translation of tokens

The movement of a token in a new place and its interaction with static components are considered.

To allow correct synchronisations, the new names introduced in the previous steps are are introduced in the token as well.
4. Building the system equation

All PEPA components built in the previous steps are put in parallel and forced to synchronise on common action types.
A simple example: preprocessing

Agent[Agent] Master

L1

P1

(go1, T)

P2

Q2

Agent[_] Probe

(go2, T)

P3

Agent[_] Probe

L2

(return, T)

(return, T)
A simple example: static components

\[
\begin{align*}
\text{Master} & \quad \text{def} \quad = (\text{dump}, T).\text{Master}' + \\
\text{Master}' & \quad \text{def} \quad = (\text{analyse}, r_a).\text{Master}
\end{align*}
\]

\[
\begin{align*}
\text{Probe}_1 & \quad \text{def} \quad = (\text{monitor}_1, r_m).\text{Probe}_1 + \\
& \quad \quad \quad \quad \quad (\text{interrogate}_1, T).\text{Probe}_1 \\
\text{Probe}_2 & \quad \text{def} \quad = (\text{monitor}_2, r_m).\text{Probe}_2 + \\
& \quad \quad \quad \quad \quad (\text{interrogate}_2, T).\text{Probe}_2
\end{align*}
\]
A simple example: cells

\[
\begin{align*}
\text{Cell}_{10} & \triangleq (\text{go}_1, T).\text{Cell}_{11} \\
\text{Cell}_{11} & \triangleq (\text{return}, T).\text{Cell}_{10} \\
\text{Cell}_{20} & \triangleq (\text{return}, T).\text{Cell}_{21} \\
\text{Cell}_{21} & \triangleq (\text{go}_1, T).\text{Cell}_{20} + (\text{go}_2, T).\text{Cell}_{20} \\
\text{Cell}_{30} & \triangleq (\text{go}_2, T).\text{Cell}_{31} \\
\text{Cell}_{31} & \triangleq (\text{return}, T).\text{Cell}_{30}
\end{align*}
\]
A simple example: tokens

<table>
<thead>
<tr>
<th>Agent</th>
<th>def</th>
<th>= (go₁, λ).Agent₁’ + (go₂, λ).Agent₂’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent₁’</td>
<td>def</td>
<td>= (interrogate₁, rᵢ).Agent”</td>
</tr>
<tr>
<td>Agent₂’</td>
<td>def</td>
<td>= (interrogate₂, rᵢ).Agent”</td>
</tr>
<tr>
<td>Agent’”</td>
<td>def</td>
<td>= (return, µ).Agent’”</td>
</tr>
<tr>
<td>Agent’”’</td>
<td>def</td>
<td>= (dump, rᵣ).Agent</td>
</tr>
</tbody>
</table>
A simple example: model equation

\[
\begin{align*}
\text{System} \ &= \\
&= (\text{Cell}_{10}^{K_1} (\text{Probe}_{1}^{K_2} (\text{Agent}^{K_3} (\text{Cell}_{21}^{K_4} \\
&\quad (\text{Master}^{K_5} (\text{Probe}_{2}^{K_6} \text{Cell}_{30}))))))))
\end{align*}
\]

\[
\begin{align*}
K_1 &= \{\text{go}_1, \text{return}\} \\
K_2 &= \{\text{interrogate}_1\} \\
K_3 &= \{\text{go}_1, \text{go}_2, \text{return}\} \\
K_4 &= \{\text{dump}\} \\
K_5 &= \{\text{interrogate}_2\} \\
K_6 &= \{\text{go}_2, \text{return}\}
\end{align*}
\]
1. Correspondent sends IP packets to the mobile node at his home address.
2. The Mobile Node sends its new IP address to the Home Agent
Complex example: mobile IP

3. The Home Agent forwards packets to the Mobile Node
Complex example: mobile IP

1. The Mobile Node sends its new IP address to the Correspondent
5. The Correspondent sends packets directly to the Mobile Node
Complex example: mobile IP

- 1 Mobile Node, 1 Correspondent, 1 Domain

  ✓ 2.8 million of states
  ✓ 16 million of transitions
  ✓ in 13.2 seconds, 1.6GHz Pentium IV with 256 MB of RAM
Conclusions

- PEPA nets is relatively new but we think that it can provide a framework for modelling systems characterised by some mobility

Future work

- Synchronisation over net transitions
- Movement of more than one token
- Graphical interface (done!)