Performance Evaluation Process Algebra

Jane Hillston

School of Informatics
University of Edinburgh

8th April 2019
Outline

1. Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

2. Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration

3. Case study: smart building and active badges

4. Conclusions
   - Alternative interpretations
Outline

1. Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

2. Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration

3. Case study: smart building and active badges

4. Conclusions
   - Alternative interpretations
Performance Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.
Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

This often involves a trade-off between the interests of the users, who want more resource, and the interests of system operators, who want to minimise the resource.
Performance Modelling: Motivation

Capacity planning

- How many clients can the existing server support and maintain reasonable response times?
Performance Modelling: Motivation

System Configuration
- How many frequencies do you need to keep blocking probabilities low?
Performance Modelling: Motivation

System Tuning

- What speed of conveyor belt will minimize robot idle time and maximize throughput whilst avoiding lost widgets?
Performance Modelling using CTMC

\[
Q = \begin{pmatrix}
-\Sigma & \cdots & \cdots \\
\cdots & -\Sigma & \cdots \\
\cdots & \cdots & -\Sigma \\
\end{pmatrix}
\]
A stochastic process $X(t)$ is a Markov process iff for all $t_0 < t_1 < \ldots < t_n < t_{n+1}$, the joint probability distribution of $(X(t_0), X(t_1), \ldots, X(t_n), X(t_{n+1}))$ is such that

$$\Pr(X(t_{n+1}) = s_{i_{n+1}} | X(t_0) = s_{i_0}, \ldots, X(t_n) = s_{i_n}) = \Pr(X(t_{n+1}) = s_{i_{n+1}} | X(t_n) = s_{i_n})$$
Performance Modelling using CTMC
Performance Modelling using CTMC

STATE TRANSITION DIAGRAM
A negative exponentially distributed duration is associated with each transition.
Performance Modelling using CTMC

these parameters form the entries of the infinitesimal generator matrix $Q$
Performance Modelling using CTMC

In steady state the probability flux out of a state is balanced by the flux in.
Performance Modelling using CTMC

"Global balance equations" captured by $\sum Q = 0$ solved by linear algebra.
Performance Modelling using CTMC

\[
\begin{align*}
\text{SYSTEM} & \quad \rightarrow \quad \text{STATE TRANSITION DIAGRAM} & \rightarrow \quad Q = \begin{pmatrix}
\ldots & \ldots \\
\ldots & \ldots \\
\ldots \ldots & \ldots \\
\end{pmatrix} \\
\downarrow & \quad \rightarrow \quad \text{MARKOV PROCESS} & \rightarrow \quad \Pi = \left( \begin{array}{c}
p_1, p_2, p_3, \ldots, p_n \end{array} \right)
\end{align*}
\]
Performance Modelling using CTMC

- SYSTEM
- STATE TRANSITION DIAGRAM
- \( Q = \)
  \[ \begin{pmatrix}
  -\sum & \cdots & \cdots \\
  \cdots & -\sum & \cdots \\
  \cdots & \cdots & -\sum \\
  \end{pmatrix} \]
  
  MARKOV PROCESS

- EQUILIBRIUM PROBABILITY DISTRIBUTION
  \( \pi = (\pi_1, \pi_2, \pi_3, \cdots, \pi_n) \)

- PERFORMANCE MEASURES
  e.g. throughput, response time, utilisation
Performance Modelling using CTMC

- **SYSTEM**
- **STATE TRANSITION DIAGRAM**
- **Q**
- **EQUILIBRIUM PROBABILITY DISTRIBUTION**
- **PERFORMANCE MEASURES**

**HIGH-LEVEL MODELLING FORMALISM**
e.g. queueing networks and stochastic Petri nets
Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

\[
Q = \begin{pmatrix}
\ldots \\
\ldots \\
\ldots \\
\ldots \\
\ldots \\
\end{pmatrix}
\]

\[
\text{MARKOV PROCESS}
\]

\[
\text{TRANSITION}
\]

\[
\text{STATE}
\]

\[
\text{SYSTEM}
\]
Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC
**Performance Modelling using CTMC**

**Model Construction**
- describing the system using a high level modelling formalism
- generating the underlying CTMC

**Model Manipulation**
- model simplification
- model aggregation
Performance Modelling using CTMC

Model Construction
- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation
- model simplification
- model aggregation

\[
Q = \begin{pmatrix}
-\sum & \cdots \\
\cdots & -\sum & \cdots \\
\cdots & \cdots & -\sum
\end{pmatrix}
\]
Performance Modelling using CTMC

Model Construction
- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation
- model simplification
- model aggregation

Model Solution
- solving the CTMC to find steady state probability distribution
- deriving performance measures
Models consist of agents which engage in actions.

\[ \alpha.P \]

- Action type or name
- Agent/component

Graphical representation:
- Node labeled with a
- Node labeled with b
- Node labeled with c
- Arrows indicating connections between the nodes
Process Algebra

- Models consist of agents which engage in actions.

\[ \alpha.P \]

- action type or name
- agent/component

Diagram:
- Node labeled \( \alpha \) connected to labeled \( P \)
- Cycles labeled \( a \), \( b \), \( c \)
Process Algebra

- Models consist of agents which engage in actions.
Models consist of agents which engage in actions.

\[
\alpha.P
\]

- action type or name
- agent/component

![Diagram showing a, b, c actions]
Models consist of agents which engage in actions.

\[ \alpha.P \]

- action type or name
- agent/component
Models consist of agents which engage in actions.

\[ \alpha.P \]

The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.
Models consist of **agents** which engage in **actions**.

\[ \alpha \cdot P \]

- The structured operational (interleaving) semantics of the language is used to generate a **labelled transition system**.

**Process algebra model**
Models consist of **agents** which engage in **actions**.

The structured operational (interleaving) semantics of the language is used to generate a **labelled transition system**.
Models consist of agents which engage in actions.

The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.
Process Algebra

- Models consist of agents which engage in actions.

\[ \alpha.P \]

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model → SOS rules → Labelled transition system
A simple example: processors and resources

\[
\begin{align*}
Proc_0 & \overset{\text{def}}{=} task1.Proc_1 \\
Proc_1 & \overset{\text{def}}{=} task2.Proc_0 \\
Res_0 & \overset{\text{def}}{=} task1.Res_1 \\
Res_1 & \overset{\text{def}}{=} reset.Res_0 \\
\end{align*}
\]

\[\text{Proc}_0 \parallel_{\text{task}1} \text{Res}_0\]
A simple example: processors and resources

\[
\begin{align*}
\text{Proc}_0 & \overset{\text{def}}{=} \text{task1. Proc}_1 \\
\text{Proc}_1 & \overset{\text{def}}{=} \text{task2. Proc}_0 \\
\text{Res}_0 & \overset{\text{def}}{=} \text{task1. Res}_1 \\
\text{Res}_1 & \overset{\text{def}}{=} \text{reset. Res}_0 \\
\text{Proc}_0 \parallel_{\text{task1}} \text{ Res}_0
\end{align*}
\]
Stochastic process algebras

Process algebras where models are decorated with quantitative information used to generate a stochastic process are stochastic process algebras (SPA).
Process algebras where models are decorated with quantitative information used to generate a stochastic process are stochastic process algebras (SPA).

This extension was motivated by a desire to bring this formal and compositional approach to modelling to bear in performance analysis supporting the derivation of measures such as throughput, utilisation and response time.
Models are constructed from components which engage in activities.

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

- Action type or name
- Activity rate (parameter of an exponential distribution)
- Component/derivative
Models are constructed from components which engage in activities.

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

The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
- Models are constructed from components which engage in activities.

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

- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
- Models are constructed from components which engage in activities.

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

- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
- Models are constructed from components which engage in activities.

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

- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
Models are constructed from components which engage in activities.

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

The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
Models are constructed from components which engage in activities.

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

The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

How long will it take for the system to arrive in a particular state?
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

Does a given property $\phi$ hold within the system with a given probability?
Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

For a given starting state, how long is it until a given property $\phi$ holds?
Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.
PEPA components perform activities either independently or in co-operation with other components.

- $(\alpha, f).P$ \hspace{1cm} Prefix
- $P_1 + P_2$ \hspace{1cm} Choice
- $P_1 \boxtimes P_2$ \hspace{1cm} Co-operation
- $P/L$ \hspace{1cm} Hiding
- $C$ \hspace{1cm} Constant
PEPA components perform activities either independently or in co-operation with other components.

- \((\alpha, f).P\) Prefix
- \(P_1 + P_2\) Choice
- \(P_1 \otimes P_2\) Co-operation
- \(P/L\) Hiding
- \(C\) Constant
Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.

\[(\alpha, f).P\] Prefix
\[P_1 + P_2\] Choice
\[P_1 \boxplus P_2\] Co-operation
\[P / L\] Hiding
\[C\] Constant
PEPA components perform activities either independently or in co-operation with other components.

\[(\alpha, f).P\] Prefix

\[P_1 + P_2\] Choice

\[P_1 \parallel P_2\] Co-operation

\[P / L\] Hiding

\[C\] Constant
PEPA components perform activities either independently or in co-operation with other components.

\[(\alpha, f).P \quad \text{Prefix}\]
\[P_1 + P_2 \quad \text{Choice}\]
\[P_1 \parallel^L P_2 \quad \text{Co-operation}\]
\[P/L \quad \text{Hiding}\]
\[C \quad \text{Constant}\]
PEPA components perform activities either independently or in co-operation with other components.

$$(\alpha, f).P$$ Prefix

$${P_1} + {P_2}$$ Choice

$${P_1} \bowtie {P_2}$$ Co-operation

$${P/L}$$ Hiding

$$C$$ Constant
PEPA components perform activities either independently or in co-operation with other components.

\[(\alpha, f).P\] Prefix
\[P_1 + P_2\] Choice
\[P_1 \parallel P_2\] Co-operation
\[P / L\] Hiding
\[C\] Constant

\[P_1 \parallel P_2\] is a derived form for \[P_1 \boxtimes P_2\].
PEPA components perform activities either independently or in co-operation with other components.

\((\alpha, f).P\) Prefix

\(P_1 + P_2\) Choice

\(P_1 \bowtie P_2\) Co-operation

\(P/L\) Hiding

\(C\) Constant

\(P_1 \parallel P_2\) is a derived form for \(P_1 \bowtie_\emptyset P_2\).

When working with large numbers of entities, we write \(P[n]\) to denote an array of \(n\) copies of \(P\) executing in parallel.
PEPA components perform activities either independently or in co-operation with other components.

\[(\alpha, f).P\] Prefix

\[P_1 + P_2\] Choice

\[P_1 \boxdot P_2\] Co-operation

\[P/L\] Hiding

\[C\] Constant

\[P_1 \parallel P_2\] is a derived form for \[P_1 \boxdot_\emptyset P_2\].

When working with large numbers of entities, we write \[P[n]\] to denote an array of \(n\) copies of \(P\) executing in parallel.

\[P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)\]
Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).
Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).

Prefix

$$(\alpha, r).E \xrightarrow{(\alpha, r)} E$$
Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a "small step" semantics).

**Prefix**

\[(\alpha, r).E \xrightarrow{(\alpha, r)} E\]

**Choice**

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

\[E + F \xrightarrow{(\alpha, r)} E'\]

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

\[E + F \xrightarrow{(\alpha, r)} F'\]
Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

$$
\begin{align*}
E & \xrightarrow{(\alpha, r)} E' \\
E \mathrel{\boxtimes} L F & \xrightarrow{(\alpha, r)} E' \mathrel{\boxtimes} L F \\
F & \xrightarrow{(\alpha, r)} F' \\
E \mathrel{\boxtimes} L F & \xrightarrow{(\alpha, r)} E \mathrel{\boxtimes} L F'
\end{align*}
$$
Structured Operational Semantics: Cooperation \((\alpha \in L)\)

\[
\begin{align*}
\text{Cooperation} & \quad \frac{E \xrightarrow{(\alpha,r_1)} E'}{E \Join_L F \xrightarrow{(\alpha,R)} E' \Join_L F'} \\
& \quad \frac{F \xrightarrow{(\alpha,r_2)} F'}{E} \\
\end{align*}
\]

where \( R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F)) \)
Cooperation

What should be the impact of cooperation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.
- Have the rate limited by the slowest participant in terms of apparent rate. This is the approach adopted by PEPA.

PEPA assumes bounded capacity: a component cannot be made to perform an activity faster in cooperation than its own recorded capacity.
Structured Operational Semantics: Cooperation ($\alpha \in L$)

Cooperation

where $R = \frac{r_1}{r_{\alpha}(E)} \frac{r_2}{r_{\alpha}(F)} \min(r_{\alpha}(E), r_{\alpha}(F))$
Structured Operational Semantics: Hiding

Hiding

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

\[ E/\mathcal{L} \xrightarrow{(\alpha, r)} E'/\mathcal{L} \quad (\alpha \notin \mathcal{L}) \]

\[ E \xrightarrow{\tau, r} E' \]

\[ E/\mathcal{L} \xrightarrow{\tau, r} E'/\mathcal{L} \quad (\alpha \in \mathcal{L}) \]
Structured Operational Semantics: Constants

**Constant**

\[
\begin{align*}
E^{(\alpha, r)} & \rightarrow E' \\
A^{(\alpha, r)} & \rightarrow E' \\
A & \overset{\text{def}}{=} E
\end{align*}
\]
A simple example: processors and resources

\[\begin{align*}
Proc_0 & \overset{\text{def}}{=} (\text{task1}, r_1).Proc_1 \\
Proc_1 & \overset{\text{def}}{=} (\text{task2}, r_2).Proc_0 \\
Res_0 & \overset{\text{def}}{=} (\text{task1}, r_3).Res_1 \\
Res_1 & \overset{\text{def}}{=} (\text{reset}, r_4).Res_0
\end{align*}\]
A simple example: processors and resources

\[
\begin{align*}
\text{Proc}_0 & \overset{\text{def}}{=} (\text{task1}, r_1).\text{Proc}_1 \\
\text{Proc}_1 & \overset{\text{def}}{=} (\text{task2}, r_2).\text{Proc}_0 \\
\text{Res}_0 & \overset{\text{def}}{=} (\text{task1}, r_3).\text{Res}_1 \\
\text{Res}_1 & \overset{\text{def}}{=} (\text{reset}, r_4).\text{Res}_0 \\
\text{Proc}_0 & \Join \text{Res}_0
\end{align*}
\]

\[
R = \min(r_1, r_3)
\]
A simple example: processors and resources

\[
\begin{align*}
\text{Proc}_0 & \overset{\text{def}}{=} (\text{task}1, r_1).\text{Proc}_1 \\
\text{Proc}_1 & \overset{\text{def}}{=} (\text{task}2, r_2).\text{Proc}_0 \\
\text{Res}_0 & \overset{\text{def}}{=} (\text{task}1, r_3).\text{Res}_1 \\
\text{Res}_1 & \overset{\text{def}}{=} (\text{reset}, r_4).\text{Res}_0
\end{align*}
\]

\[R = \min(r_1, r_3)\]
A simple example: processors and resources

\[
\begin{align*}
Proc_0 & \overset{\text{def}}{=} (\text{task1}, r_1).Proc_1 \\
Proc_1 & \overset{\text{def}}{=} (\text{task2}, r_2).Proc_0 \\
Res_0 & \overset{\text{def}}{=} (\text{task1}, r_3).Res_1 \\
Res_1 & \overset{\text{def}}{=} (\text{reset}, r_4).Res_0
\end{align*}
\]

\[\begin{pmatrix}
-R & R & 0 & 0 \\
0 & -(r_2 + r_4) & r_4 & r_2 \\
r_2 & 0 & -r_2 & 0 \\
r_4 & 0 & 0 & -r_4
\end{pmatrix}\]

\[R = \min(r_1, r_3)\]
Interplay with Performance Modelling

Model Construction: Compositionality leads to
- ease of construction
- reusable submodels
- easy to understand models
Interplay with Performance Modelling

Model Construction: **Compositionality** leads to
- ease of construction
- reusable submodels
- easy to understand models

Model Manipulation: **Equivalence relations** lead to
- term rewriting/state space reduction techniques
- aggregation techniques based on lumpability
Interplay with Performance Modelling

Model Construction: Compositionality leads to
- ease of construction
- reusable submodels
- easy to understand models

Model Manipulation: Equivalence relations lead to
- term rewriting/state space reduction techniques
- aggregation techniques based on lumpability

Model Solution: Formal semantics: lead to
- automatic identification of classes of models susceptible to efficient solution
- use of logics to express performance measures
Outline

1 Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

2 Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration

3 Case study: smart building and active badges

4 Conclusions
   - Alternative interpretations
Solving discrete state models

Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.
Solving discrete state models

Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.
Modelling at the level of individuals

\[(P_0 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_0)\]
Modelling at the level of individuals

\[ r = \frac{r_1}{3r_1} \frac{r_3}{2r_3} \min(3r_1, 2r_3) = \frac{1}{6} \min(3r_1, 2r_3) \]

\[(P_1 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_1 \parallel R_0)\]

\[(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_1)\]

\[(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_1 \parallel R_0)\]

\[(P_0 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_0)\]

\[(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_1 \parallel R_0)\]

\[(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_0 \parallel R_1)\]
Solving discrete state models

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.
Solving discrete state models

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

\[
Q = \begin{pmatrix}
q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\
q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
q_{N,1} & q_{N,2} & \cdots & q_{N,N}
\end{pmatrix}
\]

\[
\pi(t) = (\pi_1(t), \pi_2(t), \ldots, \pi_N(t))
\]
Solving discrete state models

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

$$Q = \begin{pmatrix}
q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\
q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\
\vdots & \vdots & & \vdots \\
q_{N,1} & q_{N,2} & \cdots & q_{N,N}
\end{pmatrix}$$

$$\pi(t) = (\pi_1(t), \pi_2(t), \ldots, \pi_N(t))$$
State space explosion

As the number of components, or the complexity of behaviour within components, grows the state space may become so large that it is infeasible to solve the underlying CTMC.

\[
\begin{align*}
Proc_0 & \overset{\text{def}}{=} (task_1, r_1).Proc_1 \\
Proc_1 & \overset{\text{def}}{=} (task_2, r_2).Proc_0 \\
Res_0 & \overset{\text{def}}{=} (task_1, r_3).Res_1 \\
Res_1 & \overset{\text{def}}{=} (reset, r_4).Res_0
\end{align*}
\]

\[Proc_0[N_P] \mathrel{\boxplus} \{task_1\} Res_0[N_R]\]
State space explosion

As the number of components, or the complexity of behaviour within components, grows the state space may become so large that it is infeasible to solve the underlying CTMC.

\[
\begin{align*}
\text{Proc}_0 & \quad \text{def} \quad (\text{task}1, r_1).\text{Proc}_1 \\
\text{Proc}_1 & \quad \text{def} \quad (\text{task}2, r_2).\text{Proc}_0 \\
\text{Res}_0 & \quad \text{def} \quad (\text{task}1, r_3).\text{Res}_1 \\
\text{Res}_1 & \quad \text{def} \quad (\text{reset}, r_4).\text{Res}_0
\end{align*}
\]

\[
\text{Proc}_0[N_P] \overset{\text{task1}}{\bowtie} \text{Res}_0[N_R]
\]

<table>
<thead>
<tr>
<th>Processors ($N_P$)</th>
<th>Resources ($N_R$)</th>
<th>States ($2^{N_P+N_R}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>256</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>512</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1024</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>2048</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4096</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8192</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>16384</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>32768</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>65536</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>131072</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>262144</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>524288</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1048576</td>
</tr>
</tbody>
</table>
 Achieving aggregation

- If we sacrifice looking at the **identity** of each component we can often achieve substantial state space reduction by aggregation.
Achieving aggregation

- If we sacrifice looking at the identity of each component we can often achieve substantial state space reduction by aggregation.

- This is supported by a shift in how we view the state of a model, based on a counting abstraction.
Achieving aggregation

- If we sacrifice looking at the identity of each component we can often achieve substantial state space reduction by aggregation.

- This is supported by a shift in how we view the state of a model, based on a counting abstraction.

- The syntactic nature of PEPA (and other SPAs) makes models easily understood by humans, but not so convenient for computers to directly apply these tools and approaches.
Achieving aggregation

- If we sacrifice looking at the **identity** of each component we can often achieve substantial state space reduction by aggregation.

- This is supported by a shift in how we view the state of a model, based on a **counting abstraction**.

- The **syntactic** nature of PEPA (and other SPAs) makes models easily understood by humans, but not so convenient for computers to directly apply these tools and approaches.

- By shifting to a **numerical state representation** we can more readily exploit results such as aggregation and access to alternative mathematical interpretations (i.e. **fluid approximation**).
Counting abstraction to generate the *Lumped* CTMC

\[(P_0 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_0)\]

\[(P_1 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_1)\]

\[(P_1 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_0)\]

\[(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_1 \parallel R_0)\]

\[(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_1)\]

\[(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_1 \parallel R_0)\]

\[(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_0 \parallel R_1)\]
Counting abstraction to generate the \textit{Lumped} CTMC

\[ (P_0 \parallel P_0 \parallel P_0) \otimes \{\text{task1}\} (R_0 \parallel R_0) \]

\[ (P_1 \parallel P_0 \parallel P_0) \otimes \{\text{task1}\} (R_1 \parallel R_0) \]

\[ (P_1 \parallel P_0 \parallel P_0) \otimes \{\text{task1}\} (R_0 \parallel R_1) \]

\[ (P_0 \parallel P_1 \parallel P_0) \otimes \{\text{task1}\} (R_1 \parallel R_0) \]

\[ (P_0 \parallel P_1 \parallel P_0) \otimes \{\text{task1}\} (R_0 \parallel R_1) \]

\[ (P_0 \parallel P_0 \parallel P_1) \otimes \{\text{task1}\} (R_1 \parallel R_0) \]

\[ (P_0 \parallel P_0 \parallel P_1) \otimes \{\text{task1}\} (R_0 \parallel R_1) \]
Counting abstraction to generate the *Lumped* CTMC

\[
(3, 0, 2, 0) \xrightarrow{\min(3r_1, 2r_3)} (2, 1, 1, 1)
\]

\[
(P_0 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_0)
\]

\[
(P_1 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_1 \parallel R_0)
\]

\[
(P_1 \parallel P_0 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_1)
\]

\[
(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_1 \parallel R_0)
\]

\[
(P_0 \parallel P_1 \parallel P_0) \{\text{task1}\} (R_0 \parallel R_1)
\]

\[
(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_1 \parallel R_0)
\]

\[
(P_0 \parallel P_0 \parallel P_1) \{\text{task1}\} (R_0 \parallel R_1)
\]
Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.
Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.
Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states. [Gilmore, Hillston and Ribaudo, IEEE TSE 2001].
Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states. [Gilmore, Hillston and Ribaudo, IEEE TSE 2001].

The more recent approach uses the counting abstraction and a numerical representation of states and transitions.
Solving discrete state models

- Even with aggregation, the underlying CTMC may become too large to solve.
- Such models may be studied using stochastic simulation.
- Each run generates a single trajectory through the state space.
- Many runs are needed in order to obtain average behaviours.
100 processors and 80 resources (simulation run A)
100 processors and 80 resources (simulation run B)
100 processors and 80 resources (simulation run C)
100 processors and 80 resources (simulation run D)
100 processors and 80 resources (average of 10 runs)
Collective dynamics

For some SPA models we can make considerable gains in efficiency when solving the model if we take a collective dynamics view of the system.
Collective dynamics

For some SPA models we can make considerable gains in efficiency when solving the model if we take a collective dynamics view of the system.

Collective dynamics considers the behaviour of populations of similar entities which can interactive with each other in seemingly simple ways to produce phenomena at the population level.
Collective dynamics

For some SPA models we can make considerable gains in efficiency when solving the model if we take a collective dynamics view of the system.

Collective dynamics considers the behaviour of populations of similar entities which can interact with each other in seemingly simple ways to produce phenomena at the population level.

In this case we lose the identity of components and even individuality, but for many models this is an approximation we are willing to make for the efficiency, or even tractability, of the models.
Some large process algebra models can be considered to exhibit collective dynamics
Some large process algebra models can be considered to exhibit collective dynamics

- Each component type captures the behaviour of one type of individual;
Some large process algebra models can be considered to exhibit collective dynamics

- Each component type captures the behaviour of one type of individual;
- The compositional structure of the model makes explicit interaction between component types;
Some large process algebra models can be considered to exhibit collective dynamics

- Each component type captures the behaviour of one type of individual;
- The compositional structure of the model makes explicit interaction between component types;
- When there are many instances of the individual component types these may be regarded as a population;
Some large process algebra models can be considered to exhibit collective dynamics

- Each component type captures the behaviour of one type of individual;

- The compositional structure of the model makes explicit interaction between component types;

- When there are many instances of the individual component types these may be regarded as a population;

- Through the interactions of these populations group or complex behaviours may emerge at the population level.
A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.
A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.
A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.
A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Furthermore we make a continuous approximation of how the counts vary over time.
Continuous Approximation

Use *continuous state variables* to approximate the discrete state space.
Continuous Approximation

Use *continuous state variables* to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.
Continuous Approximation

Use continuous state variables to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.
Continuous Approximation

Use continuous state variables to approximate the discrete state space.
Continuous Approximation

Use continuous state variables to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.
Use *continuous state variables* to approximate the discrete state space.
Continuous Approximation

Use **continuous state variables** to approximate the discrete state space.

Use **ordinary differential equations** to represent the evolution of those variables over time.
Simple example revisited

\[
\begin{align*}
\text{Proc}_0 & \overset{\text{def}}{=} (\text{task}1, r_1).\text{Proc}_1 \\
\text{Proc}_1 & \overset{\text{def}}{=} (\text{task}2, r_2).\text{Proc}_0 \\
\text{Res}_0 & \overset{\text{def}}{=} (\text{task}1, r_3).\text{Res}_1 \\
\text{Res}_1 & \overset{\text{def}}{=} (\text{reset}, r_4).\text{Res}_0 \\
\text{Proc}_0[N_P] & \overset{\text{task}1}{\bowtie} \text{Res}_0[N_R]
\end{align*}
\]
Simple example revisited

\[
\begin{align*}
\text{Proc}_0 & \overset{\text{def}}{=} (\text{task}_1, r_1).\text{Proc}_1 \\
\text{Proc}_1 & \overset{\text{def}}{=} (\text{task}_2, r_2).\text{Proc}_0 \\
\text{Res}_0 & \overset{\text{def}}{=} (\text{task}_1, r_3).\text{Res}_1 \\
\text{Res}_1 & \overset{\text{def}}{=} (\text{reset}, r_4).\text{Res}_0
\end{align*}
\]

- \text{task}_1 \text{ decreases } \text{Proc}_0 \text{ and } \text{Res}_0
- \text{task}_1 \text{ increases } \text{Proc}_1 \text{ and } \text{Res}_1
- \text{task}_2 \text{ decreases } \text{Proc}_1
- \text{task}_2 \text{ increases } \text{Proc}_0
- \text{reset} \text{ decreases } \text{Res}_1
- \text{reset} \text{ increases } \text{Res}_0
Simple example revisited

\[
\begin{align*}
    \text{Proc}_0 & \triangleq (\text{task}1, r_1).\text{Proc}_1 \\
    \text{Proc}_1 & \triangleq (\text{task}2, r_2).\text{Proc}_0 \\
    \text{Res}_0 & \triangleq (\text{task}1, r_3).\text{Res}_1 \\
    \text{Res}_1 & \triangleq (\text{reset}, r_4).\text{Res}_0 \\
    \text{Proc}_0[N_P] \otimes_{\{\text{task}1\}} \text{Res}_0[N_R]
\end{align*}
\]

\[
\frac{dx_1}{dt} = -\min(r_1 x_1, r_3 x_3) + r_2 x_2 \\
x_1 = \text{no. of } \text{Proc}_0
\]

- \text{ task}1 \text{ decreases } \text{Proc}_0
- \text{ task}1 \text{ is performed by } \text{Proc}_0 \text{ and } \text{Res}_0
- \text{ task}2 \text{ increases } \text{Proc}_0
- \text{ task}2 \text{ is performed by } \text{Proc}_1
Simple example revisited

\[
\text{Proc}_0 \overset{\text{def}}{=} (\text{task}_1, r_1).\text{Proc}_1 \\
\text{Proc}_1 \overset{\text{def}}{=} (\text{task}_2, r_2).\text{Proc}_0 \\
\text{Res}_0 \overset{\text{def}}{=} (\text{task}_1, r_3).\text{Res}_1 \\
\text{Res}_1 \overset{\text{def}}{=} (\text{reset}, r_4).\text{Res}_0
\]

\[
\text{Proc}_0[N_P] \mathbin{\boxdot} \text{Res}_0[N_R]
\]

ODE interpretation

\[
\frac{dx_1}{dt} = -\min(r_1 x_1, r_3 x_3) + r_2 x_2 \\
x_1 = \text{no. of Proc}_1
\]

\[
\frac{dx_2}{dt} = \min(r_1 x_1, r_3 x_3) - r_2 x_2 \\
x_2 = \text{no. of Proc}_2
\]

\[
\frac{dx_3}{dt} = -\min(r_1 x_1, r_3 x_3) + r_4 x_4 \\
x_3 = \text{no. of Res}_0
\]

\[
\frac{dx_4}{dt} = \min(r_1 x_1, r_3 x_3) - r_4 x_4 \\
x_4 = \text{no. of Res}_1
\]
100 processors and 80 resources (average of 10 runs)
100 Processors and 80 resources (average of 100 runs)
100 processors and 80 resources (average of 1000 runs)
100 processors and 80 resources (ODE solution)
Outline

1 Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

2 Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration

3 Case study: smart building and active badges

4 Conclusions
   - Alternative interpretations
Case study: active badges

We have used the PEPA modelling language to analyse the configuration of a location tracking system based on active badges.

Active badges transmit unique infra-red signals which are detected by networked sensors. These report locations back to a central database.
The badges are battery-powered and the tradeoff in the system is between the conservation of battery power and the accuracy of the information harvested from the sensors.
The badges are battery-powered and the tradeoff in the system is between the conservation of battery power and the accuracy of the information harvested from the sensors.

When transmissions from badges collide, the badges sleep for a randomly determined time before retrying.
Active badges: the PEPA model

The PEPA model of this system tracks the progress of one badge-wearer around three connected corridors (numbered 14, 15 and 16).
The PEPA model of this system tracks the progress of one badge-wearer around three connected corridors (numbered 14, 15 and 16).

The activities which are performed in the system include the badge registering with a sensor (at rate $r$), the person moving to another corridor (at rate $m$) and a sensor reporting back to the central database (at rate $s$).
Active badges: the PEPA model

\[
\begin{align*}
P_{14} & \overset{\text{def}}{=} (\text{reg}_{14}, r).P_{14} + (\text{move}_{15}, m).P_{15} \\
P_{15} & \overset{\text{def}}{=} (\text{reg}_{15}, r).P_{15} + (\text{move}_{14}, m).P_{14} + (\text{move}_{16}, m).P_{16} \\
P_{16} & \overset{\text{def}}{=} (\text{reg}_{16}, r).P_{16} + (\text{move}_{15}, m).P_{15}
\end{align*}
\]
Active badges: the PEPA model

**Person**

\[
P_{14} \overset{\text{def}}{=} (\text{reg}_{14}, r).P_{14} + (\text{move}_{15}, m).P_{15}
\]

\[
P_{15} \overset{\text{def}}{=} (\text{reg}_{15}, r).P_{15} + (\text{move}_{14}, m).P_{14} + (\text{move}_{16}, m).P_{16}
\]

\[
P_{16} \overset{\text{def}}{=} (\text{reg}_{16}, r).P_{16} + (\text{move}_{15}, m).P_{15}
\]

**Sensor**

\[
S_{14} \overset{\text{def}}{=} (\text{reg}_{14}, \top).(\text{rep}_{14}, s).S_{14}
\]

\[
S_{15} \overset{\text{def}}{=} (\text{reg}_{15}, \top).(\text{rep}_{15}, s).S_{15}
\]

\[
S_{16} \overset{\text{def}}{=} (\text{reg}_{16}, \top).(\text{rep}_{16}, s).S_{16}
\]
Active badges: the PEPA model

### Database

\[
\begin{align*}
DB_{14} & \overset{\text{def}}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{15} & \overset{\text{def}}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{16} & \overset{\text{def}}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16}
\end{align*}
\]
Active badges: the PEPA model

### Database

\[
\begin{align*}
DB_{14} & \overset{\text{def}}{=} (\text{rep}_{14}, \top).DB_{14} + (\text{rep}_{15}, \top).DB_{15} + (\text{rep}_{16}, \top).DB_{16} \\
DB_{15} & \overset{\text{def}}{=} (\text{rep}_{14}, \top).DB_{14} + (\text{rep}_{15}, \top).DB_{15} + (\text{rep}_{16}, \top).DB_{16} \\
DB_{16} & \overset{\text{def}}{=} (\text{rep}_{14}, \top).DB_{14} + (\text{rep}_{15}, \top).DB_{15} + (\text{rep}_{16}, \top).DB_{16}
\end{align*}
\]

### System

\[
P_{14} \Join_{L} (S_{14} \parallel S_{15} \parallel S_{16}) \Join_{M} DB_{14}
\]

where

\[
L = \{ \text{reg}_{14}, \text{reg}_{15}, \text{reg}_{16} \} \\
M = \{ \text{rep}_{14}, \text{rep}_{15}, \text{rep}_{16} \}
\]
Probability that the database holds inaccurate information
Outline

1. Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

2. Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration

3. Case study: smart building and active badges

4. Conclusions
   - Alternative interpretations
Scalable Representations

- Continuous approximation of CTMC
- Aggregated CTMC
- Explicit state CTMC
Scalable Representations

PEPA model

Continuous approximation of CTMC
  fluid abstraction

Aggregated CTMC
  counting abstraction

Explicit state CTMC
  individual view
Scalable Representations

PEPA model

- Continuous approximation of CTMC
  - Set of ODEs

- Aggregated CTMC
  - Reduced generator matrix

- Explicit state CTMC
  - Full generator matrix
Eclipse Plug-in for PEPA

Robust tool support is essential to make develop techniques practical.
Other applications

PEPA, and the associated analysis techniques, were originally developed with the objective of studying computer systems.

However, it has also been adopted by modelling a wide-range of other types of system:

- **Locks and movable bridges** in inland shipping in Belgium (Knapen, Hasselt)
- **Automotive on-board diagnostics** expert systems (Console, Picardi and Ribaudo)
- **Biological cell signalling pathways** (Calder, Duguid, Gilmore and Hillston)
- **Crowd dynamics** in informatic environments (Harrison, Latella and Massink)
SPA Languages

SPA
SPA Languages

SPA

- integrated time
- orthogonal time

IMC

general distributions

IGSMP, Modest

PEPA, Stochastic \( \pi \)-calculus

exponential only

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

exponential only

@QQQQ
SPA Languages

- **Integrated Time**
  - Exponential only
  - Exponential + instantaneous
  - General distributions

- **Orthogonal Time**
  - General distributions
SPA Languages

 SPA

<table>
<thead>
<tr>
<th>Integrated time</th>
<th>Exponential only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential + instantaneous</td>
<td>General distributions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orthogonal time</th>
<th>Exponential only</th>
</tr>
</thead>
<tbody>
<tr>
<td>General distributions</td>
<td></td>
</tr>
</tbody>
</table>

Conclusions Alternative interpretations

St Andrews, 08/04/19
SPA Languages

Conclusions

Alternative interpretations

SPA

- **integrated time**
  - exponential only
    - PEPA, Stochastic $\pi$-calculus
  - exponential + instantaneous
  - general distributions

- **orthogonal time**
  - exponential only
  - general distributions
SPA Languages

SPA

- integrated time
  - exponential only
    - PEPA, Stochastic $\pi$-calculus
  - exponential + instantaneous
    - EMPA, Markovian TIPP
- orthogonal time
  - general distributions
  - exponential only
  - general distributions
SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic $\pi$-calculus
  - orthogonal time
    - exponential + instantaneous
      - EMPA, Markovian TIPP
    - general distributions
      - TIPP, SPADES, GSMPA
    - exponential only
    - general distributions
SPA Languages

SPA

- integrated time
  - exponential only
    - PEPA, Stochastic $\pi$-calculus
  - exponential + instantaneous
    - EMPA, Markovian TIPP
    - TIPP, SPADES, GSMPA
    - exponential only
      - IMC
    - general distributions

- orthogonal time
  - general distributions
SPA Languages

- **integrated time**
  - exponential only
    - PEPA, Stochastic $\pi$-calculus
  - exponential + instantaneous
    - EMPA, Markovian TIPP
  - general distributions
    - TIPP, SPADES, GSMPA

- **orthogonal time**
  - exponential only
    - IMC
  - general distributions
    - IGSMP, Modest
SPA Languages

- **Integrated time**
  - **Exponential only**
    - PEPA, Stochastic $\pi$-calculus
  - **Exponential + instantaneous**
    - EMPA, Markovian TIPP
- **Orthogonal time**
  - **General distributions**
    - TIPP, SPADES, GSMPA
  - **Exponential only**
    - IMC
  - **General distributions**
    - IGSMP, Modest
Conclusions

- Stochastic Process Algebras like PEPA, provide a high-level modelling language for performance modelling with many benefits.
Conclusions

- Stochastic Process Algebras like PEPA, provide a high-level modelling language for performance modelling with many benefits.
- The semantics are encoded in software, so the underlying CTMC (or ODE) is generated automatically.
Conclusions

- Stochastic Process Algebras like PEPA, provide a high-level modelling language for performance modelling with many benefits.

- The semantics are encoded in software, so the underlying CTMC (or ODE) is generated automatically.

- Similarly various model reduction techniques can be characterised by the syntax of the language, meaning that the validity of the reduction is proven for the language rather than on a model-by-model basis.
Conclusions

- Stochastic Process Algebras like **PEPA**, provide a high-level modelling language for performance modelling with many benefits.

- The semantics are encoded in software, so the underlying CTMC (or ODE) is generated automatically.

- Similarly various model reduction techniques can be characterised by the syntax of the language, meaning that the validity of the reduction is proven for the language rather than on a model-by-model basis.

- **PEPA** has been used for a wide variety of applications, most recently to detect information leakage for secure computations.
Thanks!
Thanks!

More information:
http://www.dcs.ed.ac.uk/pepa