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

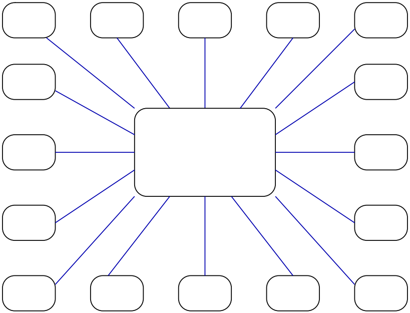
Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Performance Modelling

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.

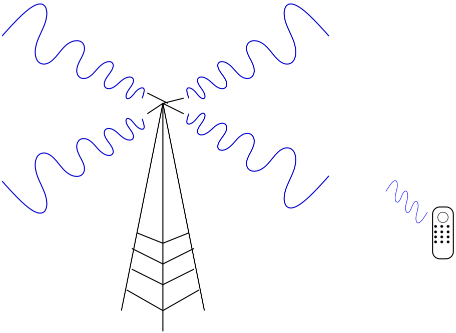
Performance Modelling: Motivation



Capacity planning

- How many clients can the existing server support and maintain reasonable response times?

Performance Modelling: Motivation

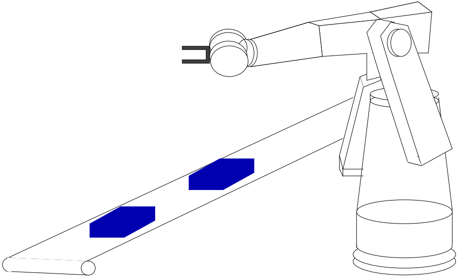


Mobile Telephone Antenna

System Configuration

- How many frequencies do you need to keep blocking probabilities low?

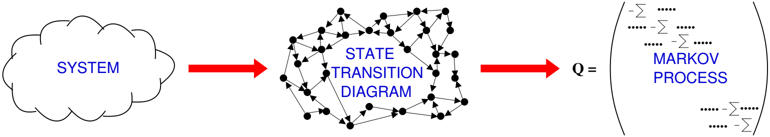
Performance Modelling: Motivation



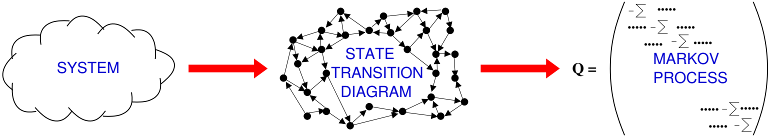
System Tuning

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

Performance Modelling using CTMC



Performance Modelling using CTMC



SYSTEM

STATE TRANSITION DIAGRAM

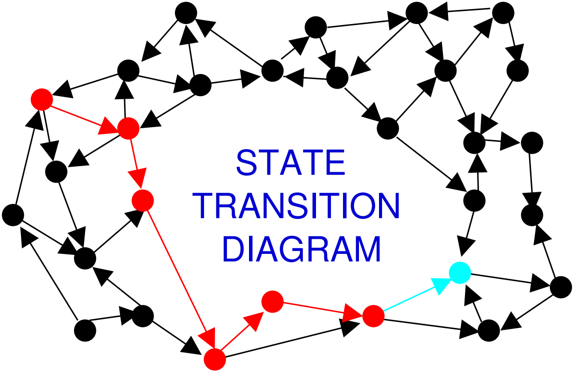
Q =

MARKOV PROCESS

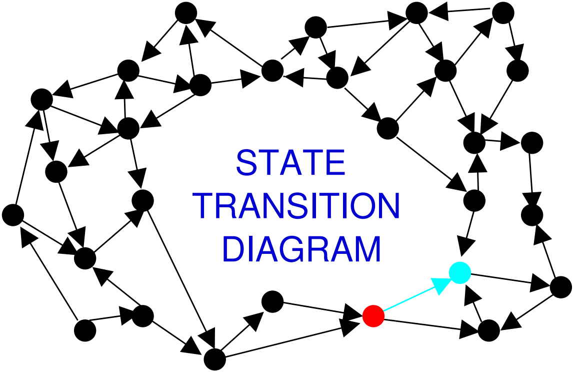
A stochastic process $X(t)$ is a Markov process iff for all $t_0 < t_1 <\ldots< t_n < t_{n+1}$, the joint probability distribution of $(X(t_0), X(t_1), \ldots, X(t_n), X(t_{n+1}))$ is such that

$$\Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_0) = s_{i_0}, \ldots, X(t_n) = s_{i_n}) = \Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_n) = s_{i_n})$$

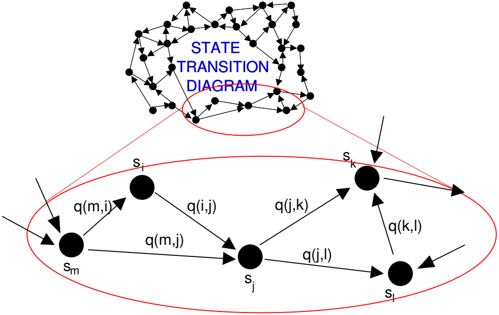
Performance Modelling using CTMC



Performance Modelling using CTMC

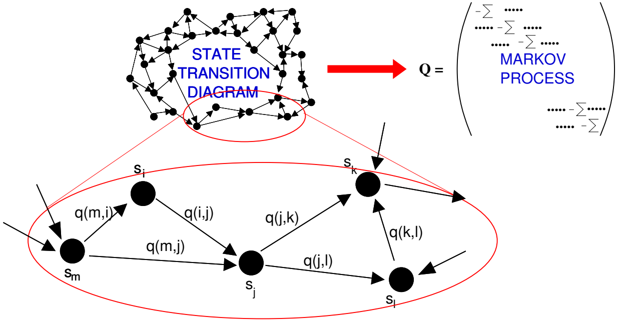


Performance Modelling using CTMC



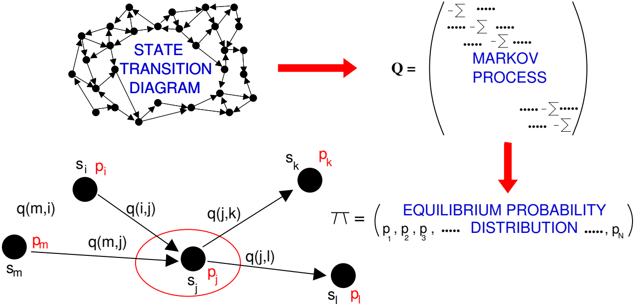
A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



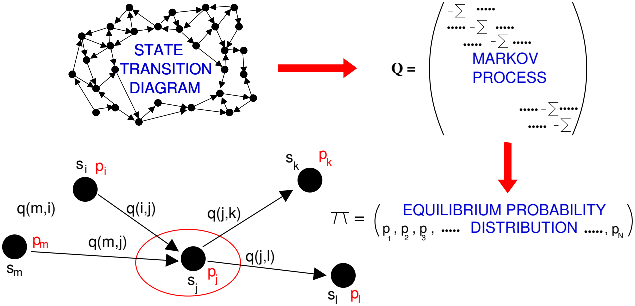
these parameters form the entries of the infinitesimal generator matrix Q

Performance Modelling using CTMC



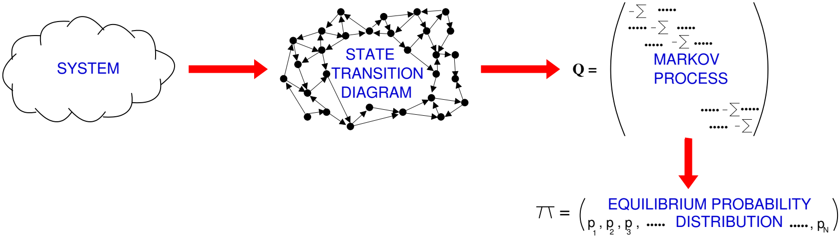
In steady state the probability flux out of a state is balanced by the flux in.

Performance Modelling using CTMC



"Global balance equations" captured by $\pi Q = 0$ solved by linear algebra

Performance Modelling using CTMC



SYSTEM

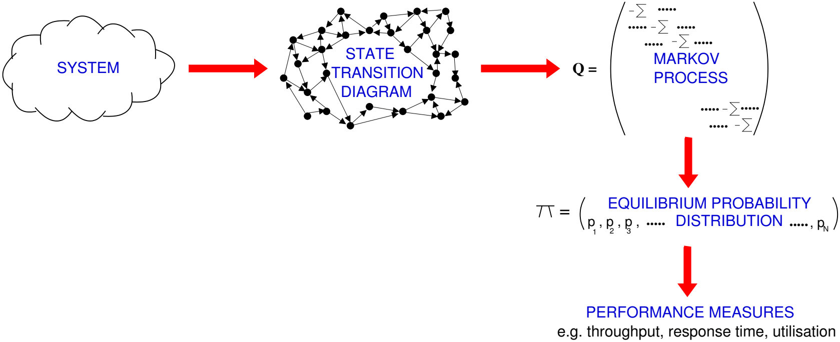
STATE TRANSITION DIAGRAM

Q = MARKOV PROCESS

π = (EQUILIBRIUM PROBABILITY DISTRIBUTION)

$p_1, p_2, p_3,$ ..... DISTRIBUTION ....., p_N

Performance Modelling using CTMC



SYSTEM

STATE TRANSITION DIAGRAM

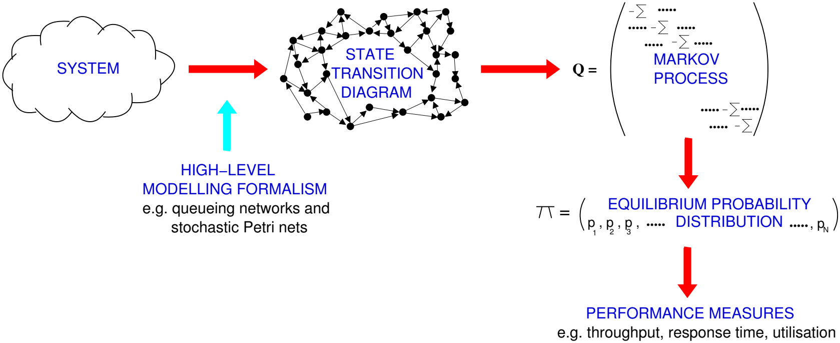
$$\mathbf{Q} = \begin{pmatrix} -\sum & \cdots & & & \\ \cdots & -\sum & \cdots & & \\ & \cdots & -\sum & \cdots & \\ & & \text{MARKOV PROCESS} & & \\ & & \cdots & -\sum & \cdots \\ & & & \cdots & -\sum \end{pmatrix}$$

$$\pi = \left(p_1 , p_2 , p_3 , \cdots \ \text{EQUILIBRIUM PROBABILITY DISTRIBUTION} \ \cdots , p_N \right)$$

PERFORMANCE MEASURES

e.g. throughput, response time, utilisation

Performance Modelling using CTMC



SYSTEM

HIGH–LEVEL
MODELLING FORMALISM
e.g. queueing networks and
stochastic Petri nets

STATE
TRANSITION
DIAGRAM

$Q =$ MARKOV PROCESS

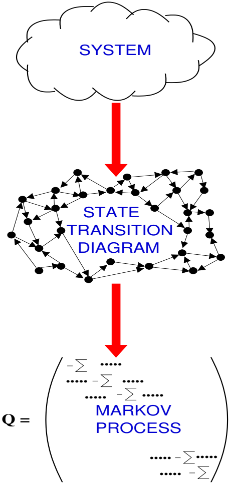
$\pi = (p_1, p_2, p_3, \ldots\ldots, p_N)$ EQUILIBRIUM PROBABILITY DISTRIBUTION

PERFORMANCE MEASURES
e.g. throughput, response time, utilisation

Performance Modelling using CTMC

Model Construction

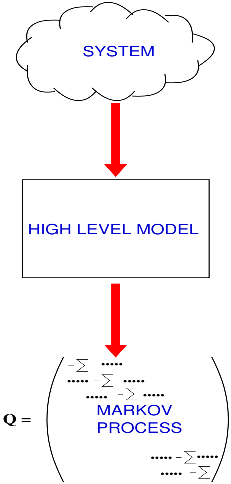
- describing the system using a high level modelling formalism
- generating the underlying CTMC



Performance Modelling using CTMC

Model Construction

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



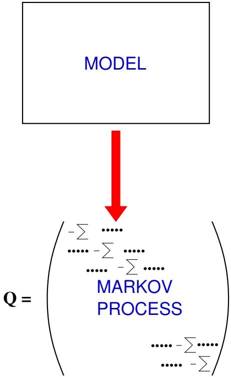
Performance Modelling using CTMC

Model Construction

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

Model Manipulation

- model simplification
- model aggregation



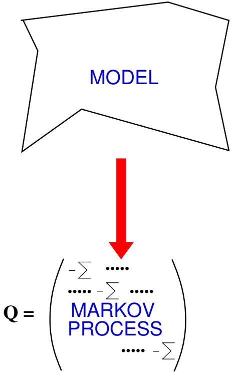
Performance Modelling using CTMC

Model Construction

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

Model Manipulation

- model simplification
- model aggregation



Performance Modelling using CTMC

Model Construction

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

Model Manipulation

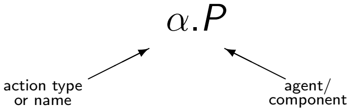
- model simplification
- model aggregation

Model Solution

- solving the CTMC to find steady state probability distribution
- deriving performance measures

Process Algebra

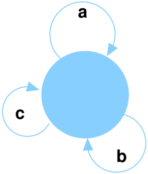
- Models consist of agents which engage in actions.



$\alpha.P$

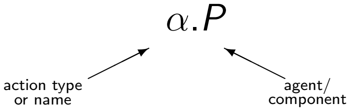
action type
or name

agent/
component



Process Algebra

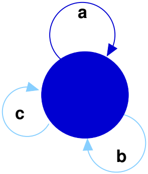
- Models consist of agents which engage in actions.



$\alpha.P$

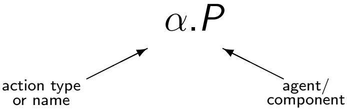
action type or name

agent/ component



Process Algebra

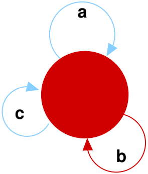
- Models consist of agents which engage in actions.



$\alpha.P$

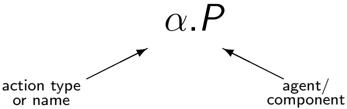
action type or name

agent/ component



Process Algebra

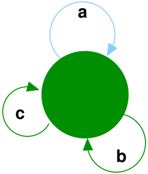
- Models consist of agents which engage in actions.



$\alpha.P$

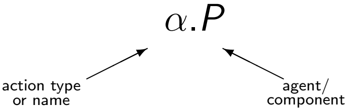
action type or name

agent/ component



Process Algebra

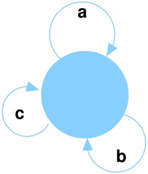
- Models consist of agents which engage in actions.



$\alpha.P$

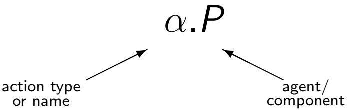
action type or name

agent/ component



Process Algebra

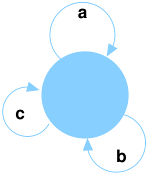
- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

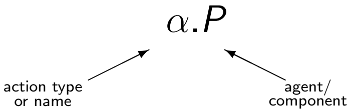
agent/ component



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

Process Algebra

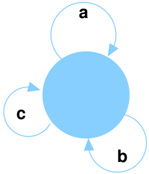
- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

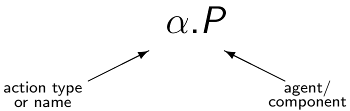


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model

Process Algebra

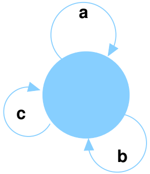
- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component



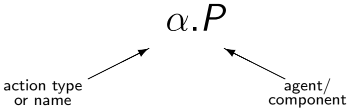
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

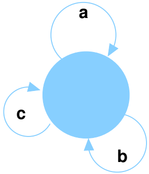
Process algebra model

SOS rules

Process Algebra

- Models consist of agents which engage in actions.



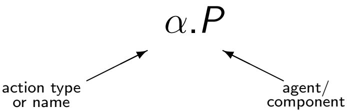


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model $\xrightarrow{\text{SOS rules}}$ Labelled transition system

Process Algebra

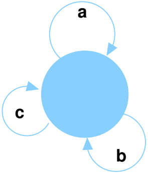
- Models consist of agents which engage in actions.



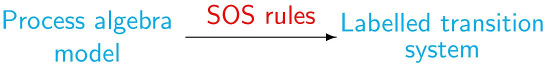
$\alpha.P$

action type or name

agent/ component



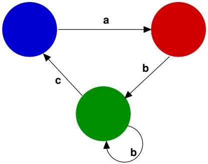
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.



Process algebra model

SOS rules

Labelled transition system



A simple example: processors and resources

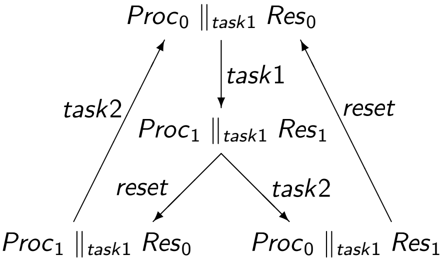
$$\begin{aligned}
Proc_0 &\overset{def}{=} task1.Proc_1 \\
Proc_1 &\overset{def}{=} task2.Proc_0 \\
Res_0 &\overset{def}{=} task1.Res_1 \\
Res_1 &\overset{def}{=} reset.Res_0
\end{aligned}$$

$$Proc_0 \parallel_{task1} Res_0$$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} task1.Proc_1 \\
Proc_1 &\stackrel{def}{=} task2.Proc_0 \\
Res_0 &\stackrel{def}{=} task1.Res_1 \\
Res_1 &\stackrel{def}{=} reset.Res_0
\end{aligned}
$$

$$Proc_0 \parallel_{task1} Res_0$$



$Proc_0 \parallel_{task1} Res_0$

$task1$

$Proc_1 \parallel_{task1} Res_1$

$task2$

$reset$

$reset$

$task2$

$Proc_1 \parallel_{task1} Res_0$

$Proc_0 \parallel_{task1} Res_1$

Stochastic process algebras

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

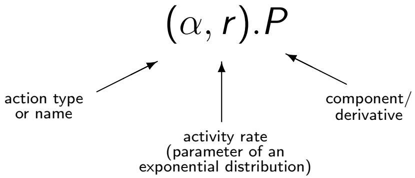
Stochastic process algebras

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, utlisation and response time.

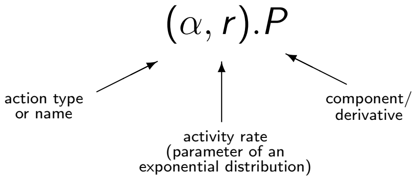
Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.


$(\alpha, r).P$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

action type
or name

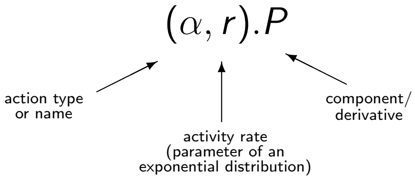
activity rate
(parameter of an
exponential distribution)

component/
derivative

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

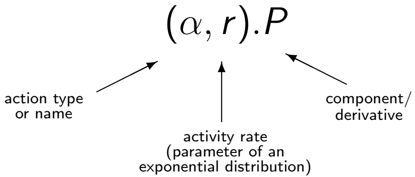
component/ derivative

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

PEPA MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



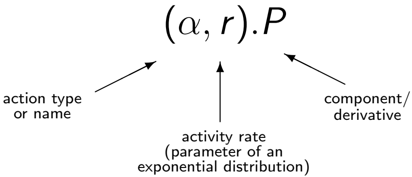
$$(\alpha, r).P$$

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

PEPA MODEL $\xrightarrow{\text{SOS rules}}$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



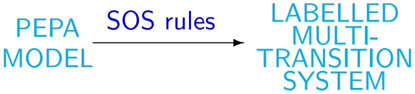
$$(\alpha, r).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

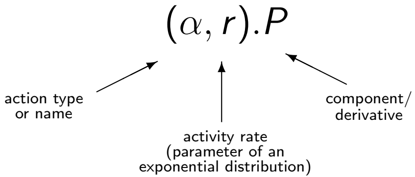
- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.



PEPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED MULTI-TRANSITION SYSTEM

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



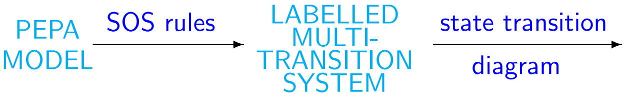
$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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



PEPA MODEL

SOS rules

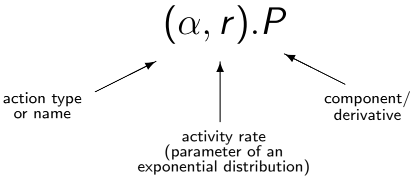
LABELLED MULTI-TRANSITION SYSTEM

state transition

diagram

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.



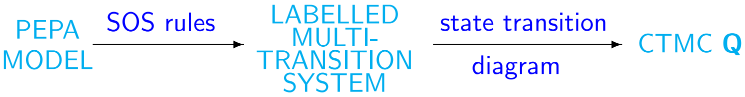
$(\alpha, r).P$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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



PEPA
MODEL

SOS rules

LABELLED
MULTI-
TRANSITION
SYSTEM

state transition

diagram

CTMC **Q**

Integrated analysis

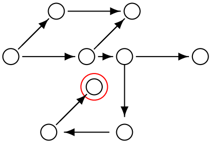
Qualitative verification can now be complemented by quantitative verification.

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?

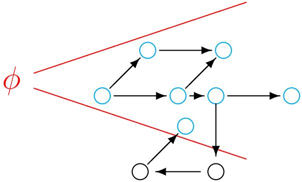


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

Does a given property ϕ hold within the system with a given probability?

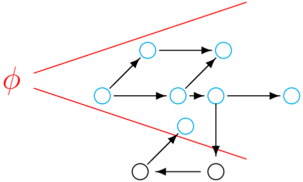


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

For a given starting state
how long is it until
a given property ϕ holds?



Performance Evaluation Process Algebra

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

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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	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
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

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
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

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

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
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
C	Constant

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

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

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Structured Operational Semantics

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

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'}$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \; (\alpha \notin L)$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \; (\alpha \notin L)$$

Structured Operational Semantics: Cooperation $(\alpha \in L)$

Cooperation $$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \ (\alpha \in L)$$

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

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
$$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L)$$

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

Structured Operational Semantics: Hiding

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \; (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \; (\alpha \in L)$$

Structured Operational Semantics: Constants

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} \,(A \overset{def}{=} E)$$

A simple example: processors and resources

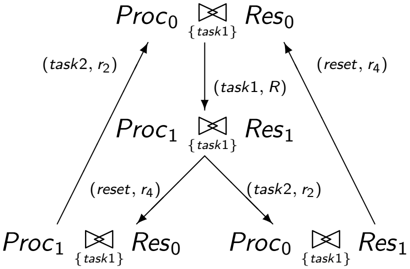
$$\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$



$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$

$(task1, R)$

$Proc_1 \underset{\{task1\}}{\bowtie} Res_1$

$(reset, r_4)$

$(task2, r_2)$

$Proc_1 \underset{\{task1\}}{\bowtie} Res_0$

$Proc_0 \underset{\{task1\}}{\bowtie} Res_1$

$(task2, r_2)$

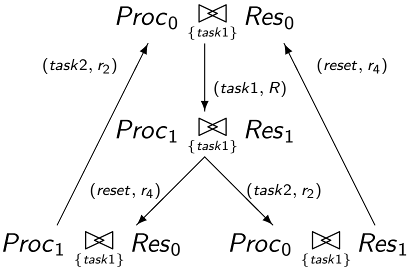
$(reset, r_4)$

$$R = \min(r_1, r_3)$$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

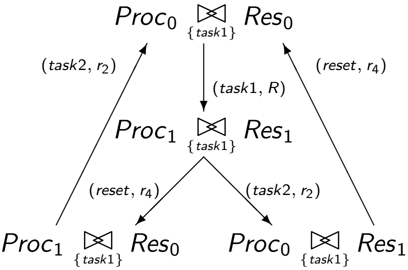

$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$
$(task2, r_2)$
$(task1, R)$
$(reset, r_4)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_1$
$(reset, r_4)$
$(task2, r_2)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_0$
$Proc_0 \underset{\{task1\}}{\bowtie} Res_1$

$R = \min(r_1, r_3)$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$


$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$
$(task2, r_2)$
$(task1, R)$
$(reset, r_4)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_1$
$(reset, r_4)$
$(task2, r_2)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_0$
$Proc_0 \underset{\{task1\}}{\bowtie} Res_1$

$R = \min(r_1, r_3)$

$$
\mathbf{Q} = \begin{pmatrix}
-R & R & 0 & 0 \\
0 & -(r_2 + r_4) & r_4 & r_2 \\
r_2 & 0 & -r_2 & 0 \\
r_4 & 0 & 0 & -r_4
\end{pmatrix}
$$

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

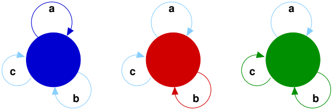
Model Solution: Formal semantics: lead to

- automatic identification of classes of models susceptible to efficient solution
- use of logics to express performance measures

Outline

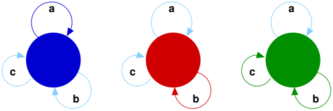
1. Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules
2. Semantics for the modelling language
   - Identity and Individuality
   - Collective Dynamics
   - Numerical illustration
3. Case study: smart building and active badges
4. Conclusions
   - Alternative interpretations

Solving discrete state models

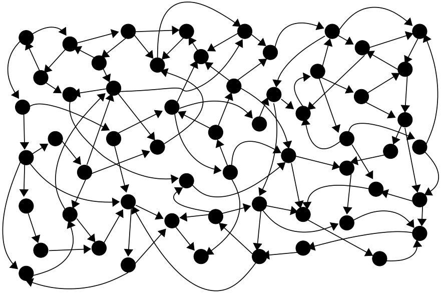


Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

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

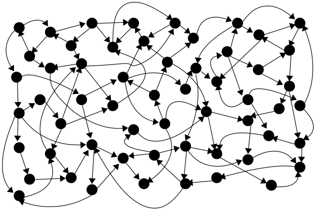
$$\begin{tikzcd}[column sep=huge]
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
(P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0) \arrow[uur, "r"] \arrow[ur, "r"] \arrow[r, "r"] \arrow[dr, "r"] \arrow[ddr, "r"] \arrow[dddr, "r"] & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)
\end{tikzcd}$$

Modelling at the level of individuals

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

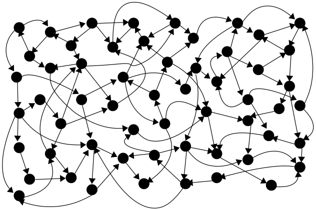
$$\begin{tikzcd}
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
 (P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0) \arrow[uur, "r"] \arrow[ur, "r"] \arrow[r, "r"] \arrow[dr, "r"] \arrow[ddr, "r"] \arrow[dddr, "r"] & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)
\end{tikzcd}$$

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.

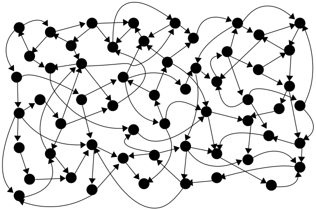
Solving discrete state models



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.

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

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.

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

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.

$$\begin{aligned} Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\ Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\ Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\ Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \end{aligned}$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

CTMC interpretation

Processors (N_P)	Resources (N_R)	States ($2^{N_P+N_R}$)
1	1	4
2	1	8
2	2	16
3	2	32
3	3	64
4	3	128
4	4	256
5	4	512
5	5	1024
6	5	2048
6	6	4096
7	6	8192
7	7	16384
8	7	32768
8	8	65536
9	8	131072
9	9	262144
10	9	524288
10	10	1048576

Achieving aggregration

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

Achieving aggregration

- 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.

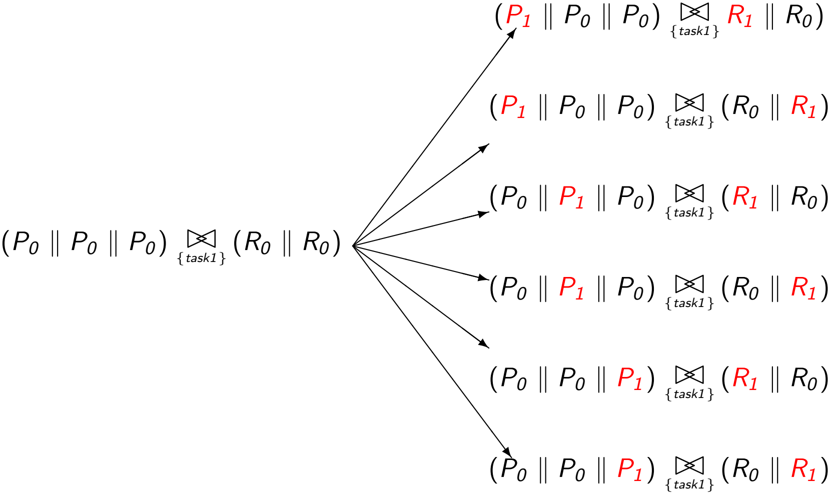
Achieving aggregration

- 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 aggregration

- 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



$(P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0)$

$(P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} R_1 \parallel R_0)$

$(P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$

$(P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0)$

$(P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$

$(P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0)$

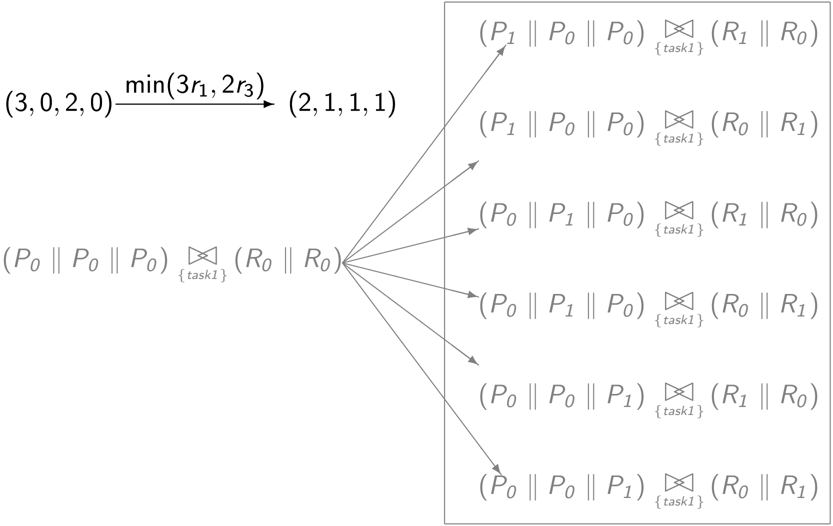
$(P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$

Counting abstraction to generate the *Lumped* CTMC

$$\begin{tikzcd}
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} R_1 \parallel R_0) \\
 & (P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
 & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
(P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0) \arrow[uuur] \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & (P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0) \\
 & (P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)
\end{tikzcd}$$

Counting abstraction to generate the *Lumped* CTMC

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



$$(P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0)$$

$$(P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0)$$

$$(P_1 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$$

$$(P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0)$$

$$(P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$$

$$(P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0)$$

$$(P_0 \parallel P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$$

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.

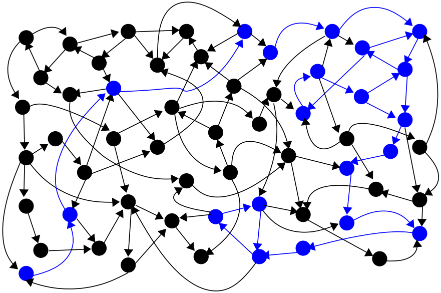
However in practice we would much rather construct the aggregated state space directly.

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

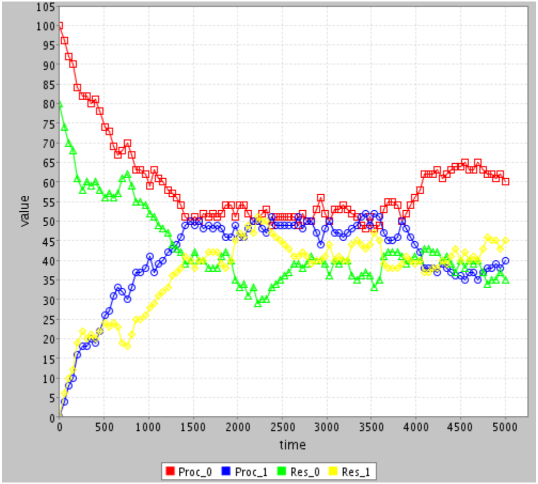
The more recent approach uses the counting abstraction and a numerical representation of states and transitions.

Solving discrete state models

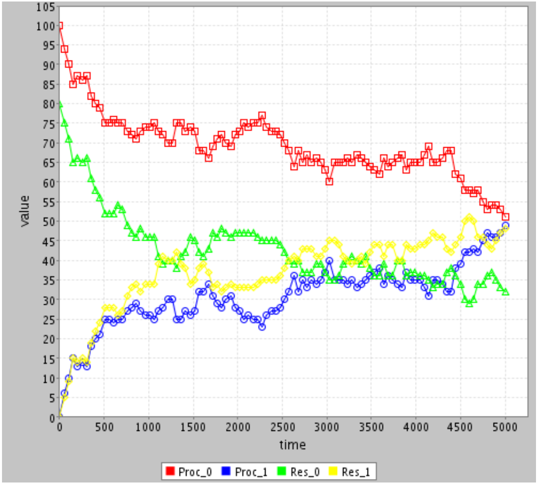
- Even with aggregation, the underlying CTMC may become too large to solve.
- Such models may be studied using stochastic simulation.
- Each run generates a single trajectory through the state space.
- Many runs are needed in order to obtain average behaviours.



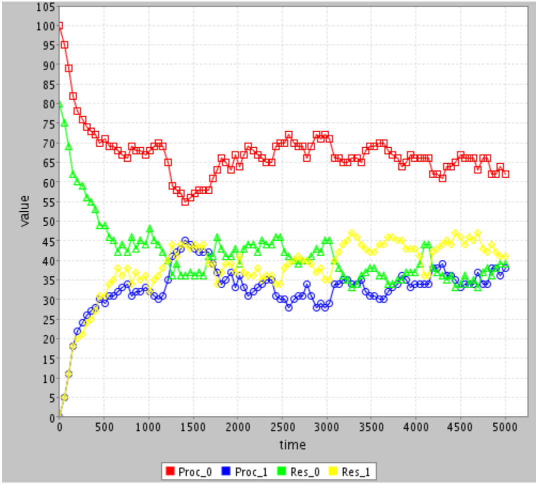
100 processors and 80 resources (simulation run A)



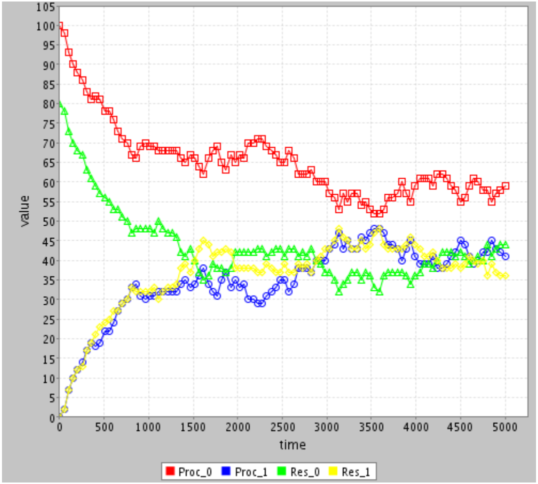
100 processors and 80 resources (simulation run B)



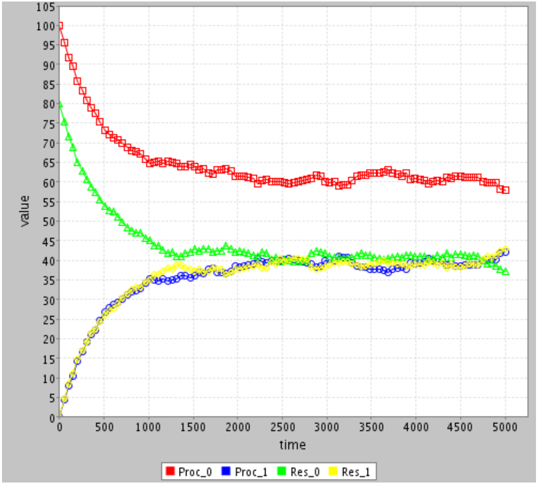
100 processors and 80 resources (simulation run C)



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 interactive 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.

Process Algebra and Collective Dynamics

Some large process algebra models can be considered to exhibit collective dynamics

Process Algebra and 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;

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

Process Algebra and 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;
- 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;

Process Algebra and 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;
- 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.

Population statistics: emergent behaviour

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.

Population statistics: emergent behaviour

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.

Population statistics: emergent behaviour

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.

Population statistics: emergent behaviour

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

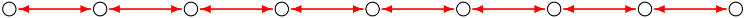
Use continuous state variables to approximate the discrete state space.

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.

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 ordinary differential equations to represent the evolution of those variables over time.

Simple example revisited

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

Simple example revisited

$$\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

- $task1$ decreases $Proc_0$ and Res_0
- $task1$ increases $Proc_1$ and Res_1
- $task2$ decreases $Proc_1$
- $task2$ increases $Proc_0$
- $reset$ decreases Res_1
- $reset$ increases Res_0

Simple example revisited

$$\begin{aligned} Proc_0 &\stackrel{\mathit{def}}{=} (task1, r_1).Proc_1 \\ Proc_1 &\stackrel{\mathit{def}}{=} (task2, r_2).Proc_0 \\ Res_0 &\stackrel{\mathit{def}}{=} (task1, r_3).Res_1 \\ Res_1 &\stackrel{\mathit{def}}{=} (reset, r_4).Res_0 \end{aligned}$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

$$\begin{aligned} \frac{\mathrm{d}x_1}{\mathrm{d}t} &= -\min(r_1\,x_1, r_3\,x_3) + r_2\,x_2 \\ x_1 &= \text{no. of } Proc_0 \end{aligned}$$

- *task*1 decreases $Proc_0$
- *task*1 is performed by $Proc_0$ and Res_0
- *task*2 increases $Proc_0$
- *task*2 is performed by $Proc_1$

Simple example revisited

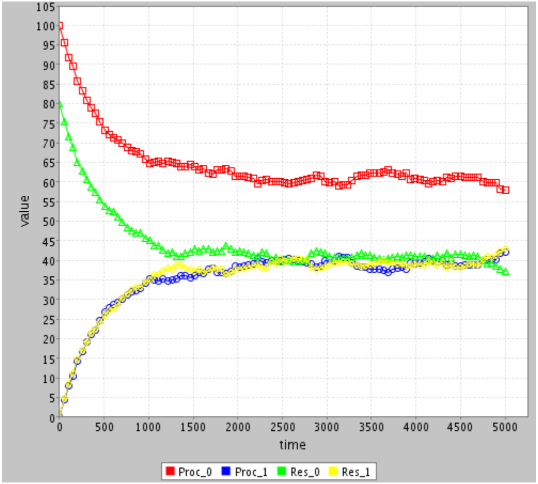
$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1\\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0\\
Res_0 &\overset{def}{=} (task1, r_3).Res_1\\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

$$Proc_0[N_P] \underset{\{task1\}}{\bowtie} Res_0[N_R]$$

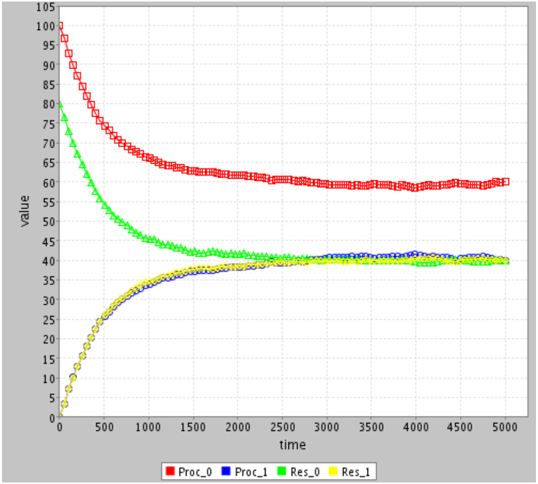
ODE interpretation

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_1\,x_1, r_3\,x_3) + r_2\,x_2\\
&\quad x_1 = \text{no. of } Proc_1\\
\frac{dx_2}{dt} &= \min(r_1\,x_1, r_3\,x_3) - r_2\,x_2\\
&\quad x_2 = \text{no. of } Proc_2\\
\frac{dx_3}{dt} &= -\min(r_1\,x_1, r_3\,x_3) + r_4\,x_4\\
&\quad x_3 = \text{no. of } Res_0\\
\frac{dx_4}{dt} &= \min(r_1\,x_1, r_3\,x_3) - r_4\,x_4\\
&\quad x_4 = \text{no. of } Res_1
\end{aligned}
$$

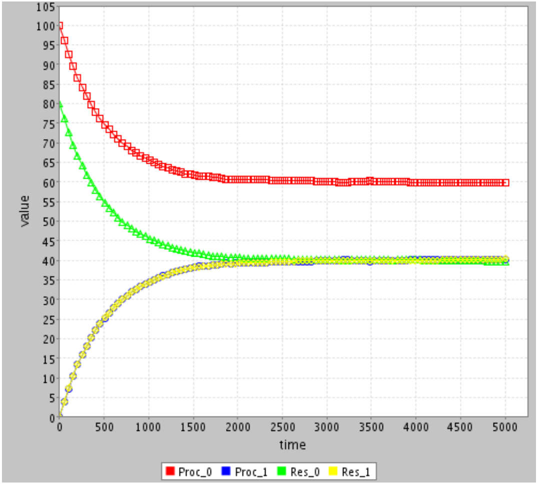
100 processors and 80 resources (average of 10 runs)



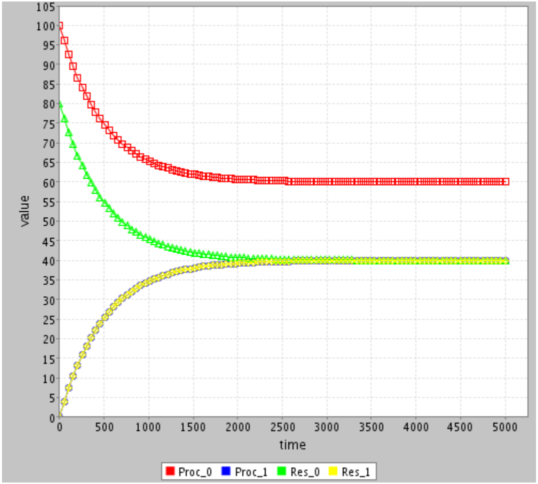
100 Processors and 80 resources (average of 100 runs)



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.

Case study: active badges

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.

Case study: active badges

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).

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 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

Person

$$\begin{aligned}
P_{14} &\overset{def}{=} (reg_{14}, r).P_{14} + (move_{15}, m).P_{15} \\
P_{15} &\overset{def}{=} (reg_{15}, r).P_{15} + (move_{14}, m).P_{14} + (move_{16}, m).P_{16} \\
P_{16} &\overset{def}{=} (reg_{16}, r).P_{16} + (move_{15}, m).P_{15}
\end{aligned}$$

Active badges: the PEPA model

Person

$$\begin{aligned}
P_{14} &\overset{def}{=} (reg_{14}, r).P_{14} + (move_{15}, m).P_{15} \\
P_{15} &\overset{def}{=} (reg_{15}, r).P_{15} + (move_{14}, m).P_{14} + (move_{16}, m).P_{16} \\
P_{16} &\overset{def}{=} (reg_{16}, r).P_{16} + (move_{15}, m).P_{15}
\end{aligned}$$

Sensor

$$\begin{aligned}
S_{14} &\overset{def}{=} (reg_{14}, \top).(rep_{14}, s).S_{14} \\
S_{15} &\overset{def}{=} (reg_{15}, \top).(rep_{15}, s).S_{15} \\
S_{16} &\overset{def}{=} (reg_{16}, \top).(rep_{16}, s).S_{16}
\end{aligned}$$

Active badges: the PEPA model

Database

$$\begin{aligned} DB_{14} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\ DB_{15} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\ DB_{16} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \end{aligned}$$

Active badges: the PEPA model

Database

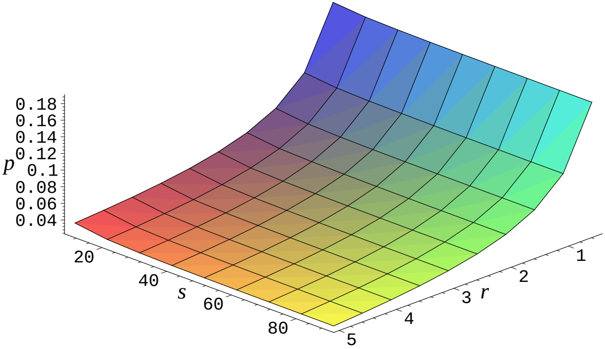
$$\begin{aligned}
DB_{14} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{15} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16} \\
DB_{16} &\overset{def}{=} (rep_{14}, \top).DB_{14} + (rep_{15}, \top).DB_{15} + (rep_{16}, \top).DB_{16}
\end{aligned}$$

System

$$P_{14} \underset{L}{\bowtie} (S_{14} \parallel S_{15} \parallel S_{16}) \underset{M}{\bowtie} DB_{14}$$

where $L = \{ reg_{14}, reg_{15}, reg_{16} \}$
$M = \{ rep_{14}, rep_{15}, rep_{16} \}$

Probability that the database holds inaccurate information



Outline

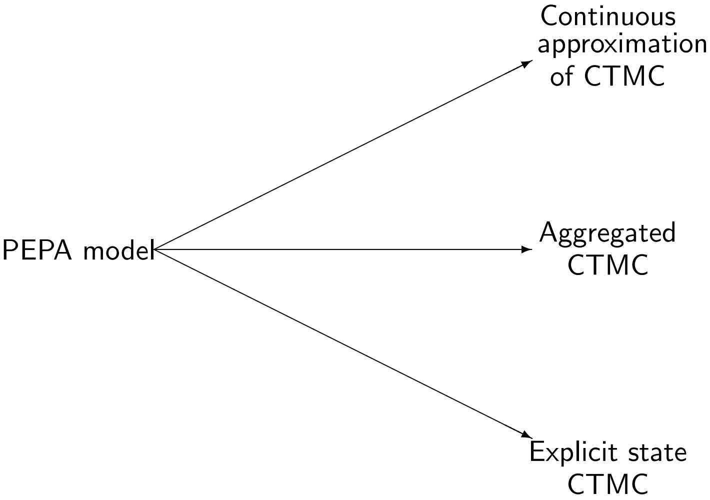
1. Introduction
   - Performance Evaluation
   - Stochastic Process Algebra
   - SOS rules

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

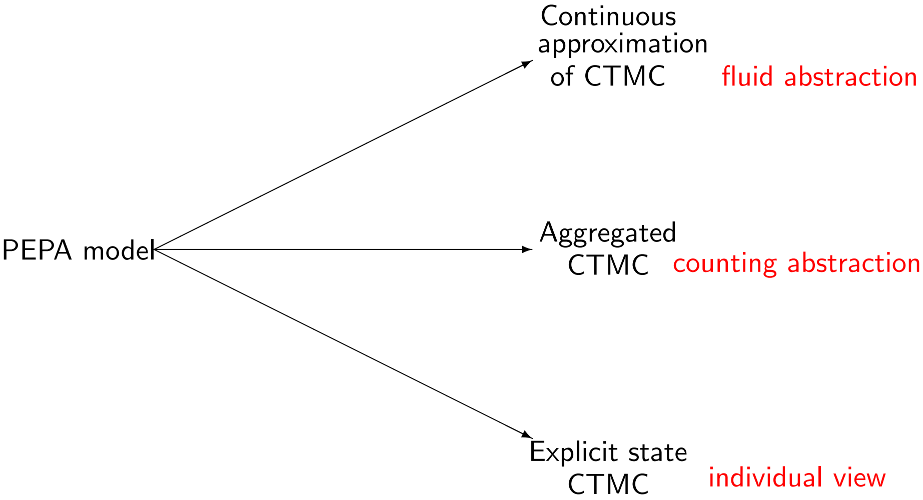
3. Case study: smart building and active badges

4. Conclusions
   - Alternative interpretations

Scalable Representations



Scalable Representations



PEPA model

Continuous approximation of CTMC

fluid abstraction

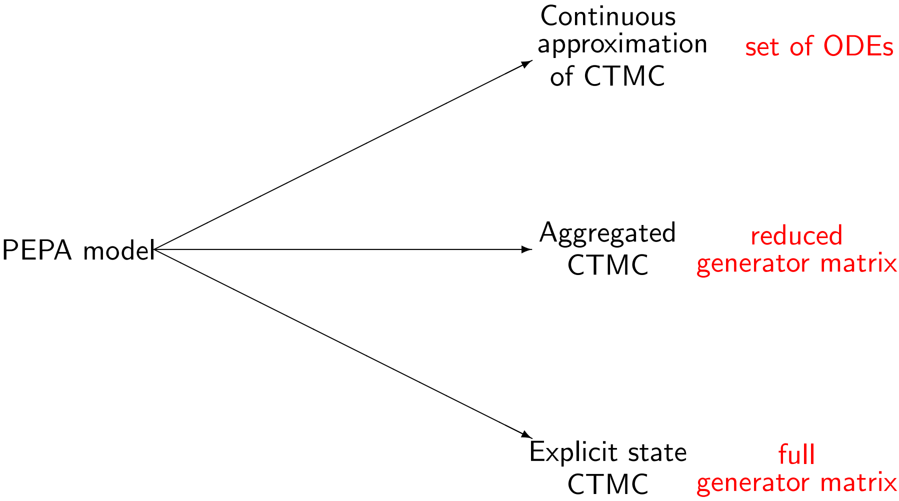
Aggregated CTMC

counting abstraction

Explicit state CTMC

individual view

Scalable Representations



PEPA model

Continuous approximation of CTMC

set of ODEs

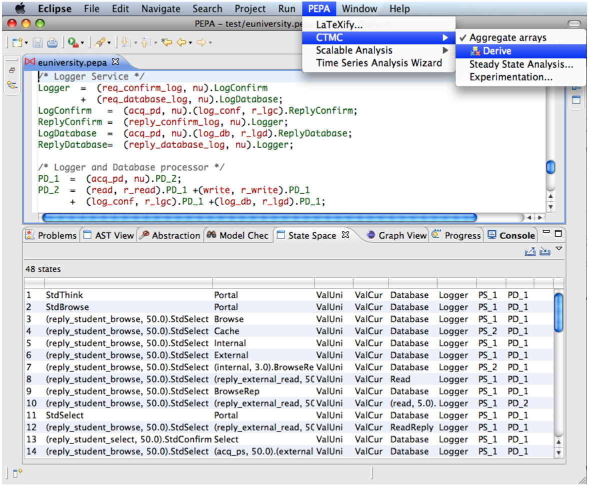
Aggregated CTMC

reduced generator matrix

Explicit state CTMC

full generator matrix

Eclipse Plug-in for PEPA


Eclipse File Edit Navigate Search Project Run PEPA Window Help
PEPA – test/euniversity.pe
LaTeXify...
CTMC
Scalable Analysis
Time Series Analysis Wizard
✓ Aggregate arrays
Derive
Steady State Analysis...
Experimentation...
euniversity.pepa
Problems AST View Abstraction Model Chec State Space Graph View Progress Console

```
/* Logger Service */
Logger   =  (req_confirm_log, nu).LogConfirm
         +  (req_database_log, nu).LogDatabase;
LogConfirm   =  (acq_pd, nu).(log_conf, r_lgc).ReplyConfirm;
ReplyConfirm =  (reply_confirm_log, nu).Logger;
LogDatabase  =  (acq_pd, nu).(log_db, r_lgd).ReplyDatabase;
ReplyDatabase=  (reply_database_log, nu).Logger;

/* Logger and Database processor */
PD_1  =  (acq_pd, nu).PD_2;
PD_2  =  (read, r_read).PD_1 +(write, r_write).PD_1
      +  (log_conf, r_lgc).PD_1 +(log_db, r_lgd).PD_1;
```

48 states

1	StdThink	Portal	ValUni	ValCur	Database	Logger	PS_1	PD_1
2	StdBrowse	Portal	ValUni	ValCur	Database	Logger	PS_1	PD_1
3	(reply_student_browse, 50.0).StdSelect	Browse	ValUni	ValCur	Database	Logger	PS_1	PD_1
4	(reply_student_browse, 50.0).StdSelect	Cache	ValUni	ValCur	Database	Logger	PS_2	PD_1
5	(reply_student_browse, 50.0).StdSelect	Internal	ValUni	ValCur	Database	Logger	PS_1	PD_1
6	(reply_student_browse, 50.0).StdSelect	External	ValUni	ValCur	Database	Logger	PS_1	PD_1
7	(reply_student_browse, 50.0).StdSelect	(internal, 3.0).BrowseRe	ValUni	ValCur	Database	Logger	PS_2	PD_1
8	(reply_student_browse, 50.0).StdSelect	(reply_external_read, 5(	ValUni	ValCur	Read	Logger	PS_1	PD_1
9	(reply_student_browse, 50.0).StdSelect	BrowseRep	ValUni	ValCur	Database	Logger	PS_1	PD_1
10	(reply_student_browse, 50.0).StdSelect	(reply_external_read, 5(	ValUni	ValCur	(read, 5.0).	Logger	PS_1	PD_2
11	StdSelect	Portal	ValUni	ValCur	Database	Logger	PS_1	PD_1
12	(reply_student_browse, 50.0).StdSelect	(reply_external_read, 5(	ValUni	ValCur	ReadReply	Logger	PS_1	PD_1
13	(reply_student_select, 50.0).StdConfirm	Select	ValUni	ValCur	Database	Logger	PS_1	PD_1
14	(reply_student_browse, 50.0).StdSelect	(acq_ps, 50.0).(external	ValUni	ValCur	Database	Logger	PS_1	PD_1

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.

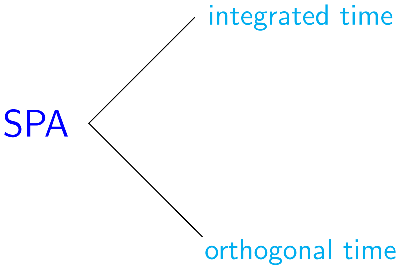
However, it has also be adopted by modelling a wide-range of other types of system:

- 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

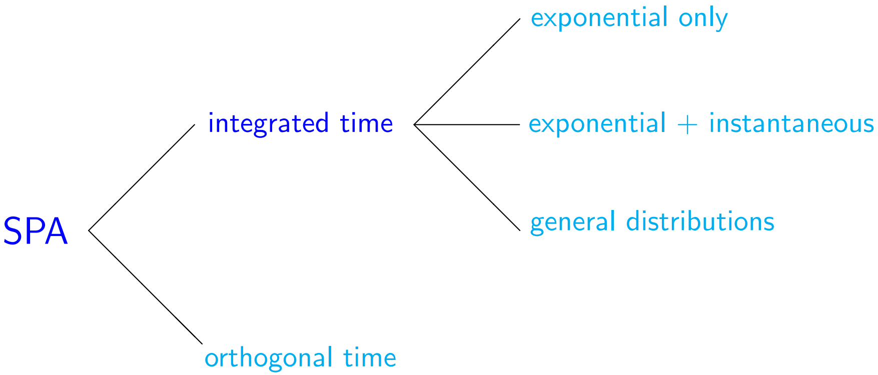


SPA

integrated time

orthogonal time

SPA Languages



SPA

integrated time

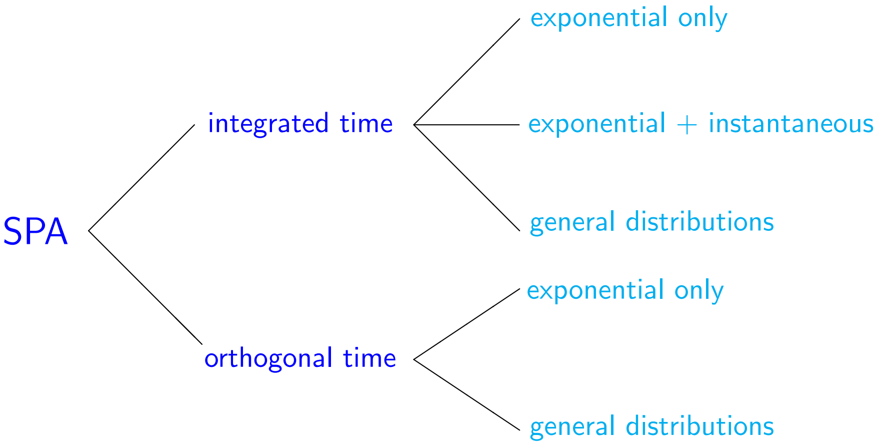
exponential only

exponential + instantaneous

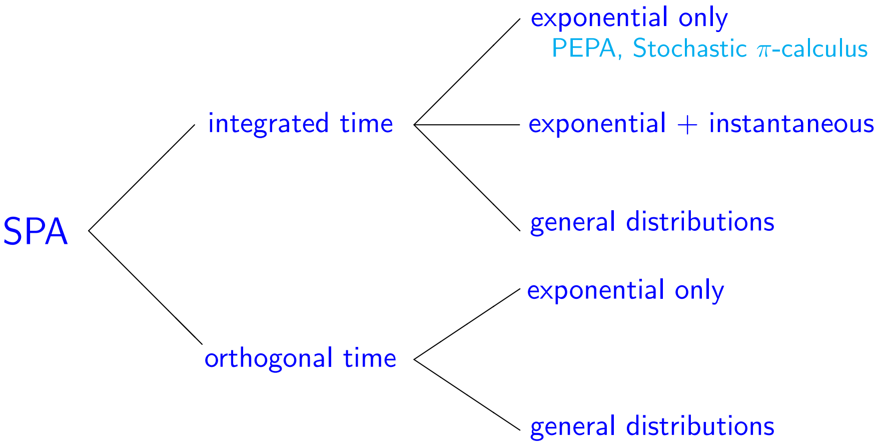
general distributions

orthogonal time

SPA Languages



SPA Languages



SPA

integrated time

exponential only

PEPA, Stochastic π-calculus

exponential + instantaneous

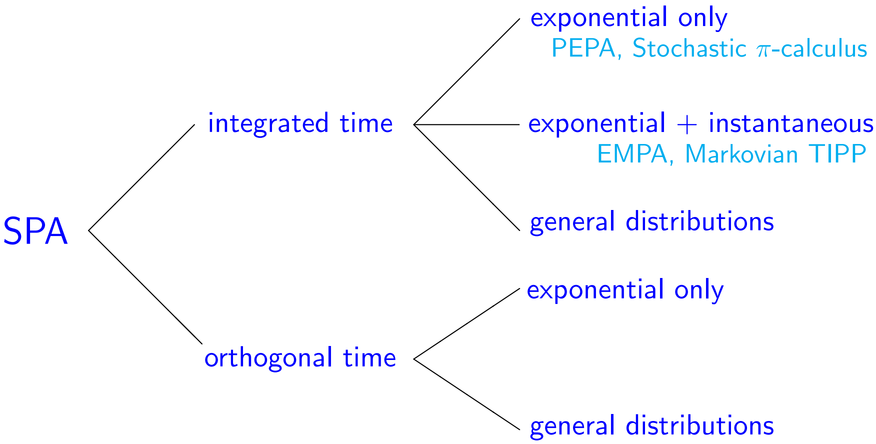
general distributions

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Stochastic π-calculus

exponential + instantaneous

EMPA, Markovian TIPP

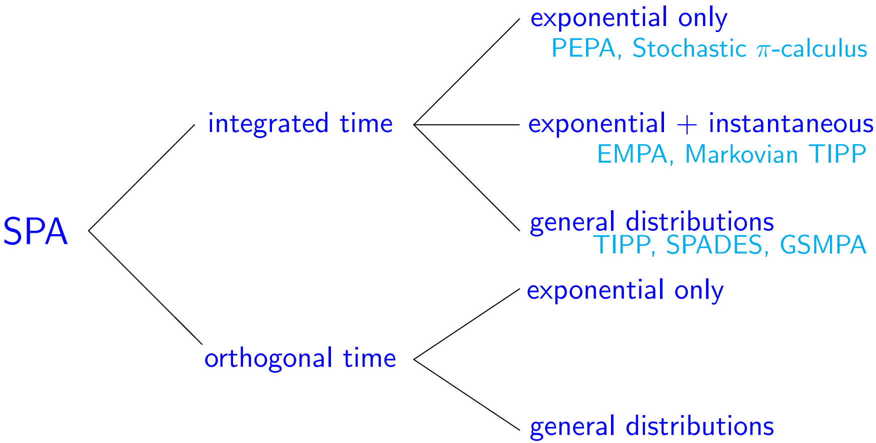
general distributions

orthogonal time

exponential only

general distributions

SPA Languages


SPA

integrated time

exponential only
PEPA, Stochastic π-calculus

exponential + instantaneous
EMPA, Markovian TIPP

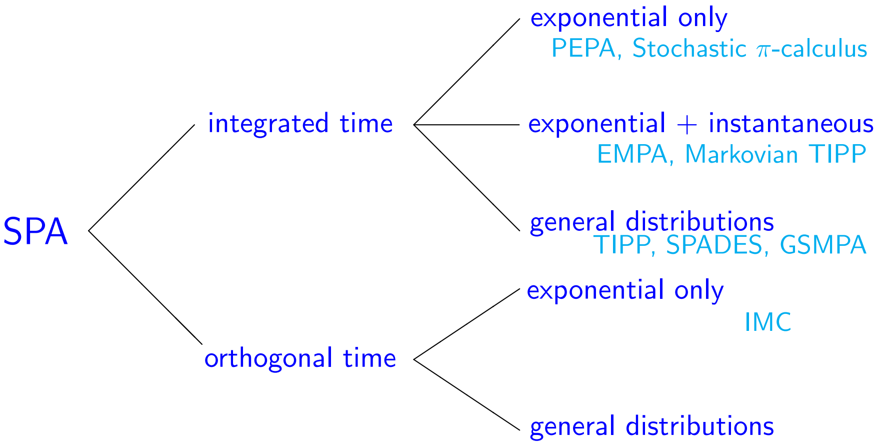
general distributions
TIPP, SPADES, GSMPA

orthogonal time

exponential only

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Stochastic π-calculus

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

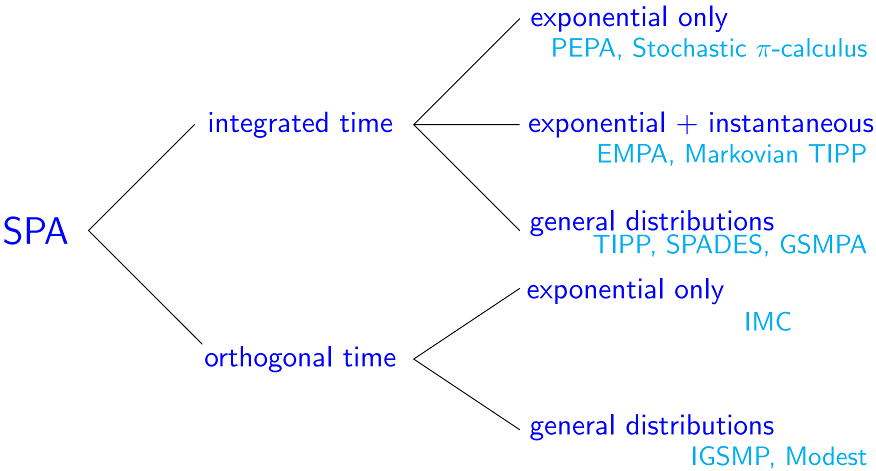
orthogonal time

exponential only

IMC

general distributions

SPA Languages



SPA

integrated time

exponential only

PEPA, Stochastic π-calculus

exponential + instantaneous

EMPA, Markovian TIPP

general distributions

TIPP, SPADES, GSMPA

orthogonal time

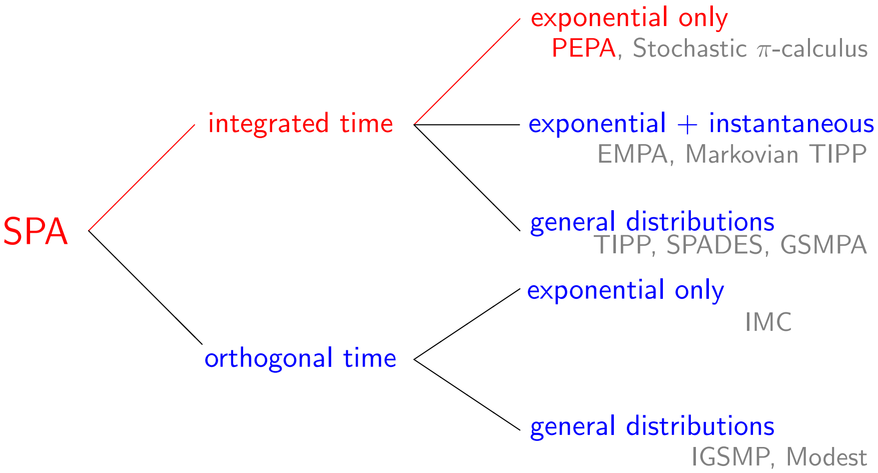
exponential only

IMC

general distributions

IGSMP, Modest

SPA Languages



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