Mean-field Analysis of Continuous-Time Markov Chains

Fluid Approximation for Stochastic Process Algebra and Stochastic Model Checking

Fluid Approximation for Stochastic Process Algebras

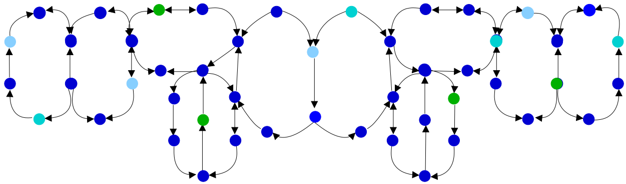
Jane Hillston

Laboratory for Foundations of Computer Science
University of Edinburgh

24th October 2012

Collective Dynamics

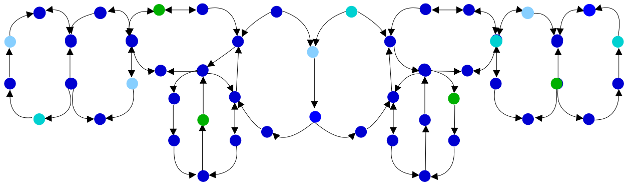
The behaviour of many systems can be interpreted as the result of the collective behaviour of a large number of interacting entities.



For such systems we are often as interested in the population level behaviour as we are in the behaviour of the individual entities.

Collective Dynamics

The behaviour of many systems can be interpreted as the result of the collective behaviour of a large number of interacting entities.



For such systems we are often as interested in the population level behaviour as we are in the behaviour of the individual entities.

Collective Behaviour

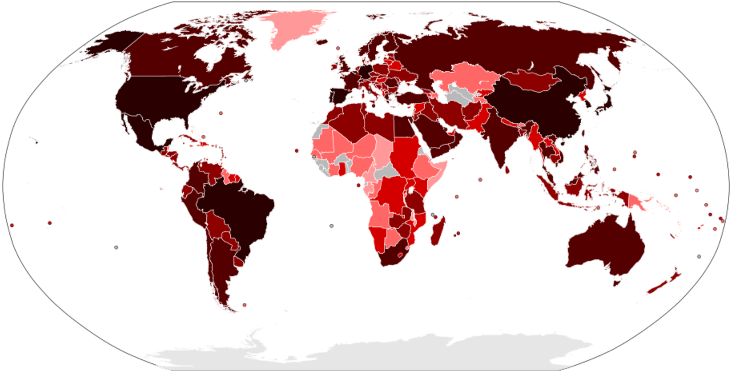
In the natural world there are many instances of collective behaviour and its consequences:

Collective Behaviour

In the natural world there are many instances of collective behaviour and its consequences:

Collective Behaviour

This is also true in the man-made and engineered world:



Spread of H1N1 virus in 2009

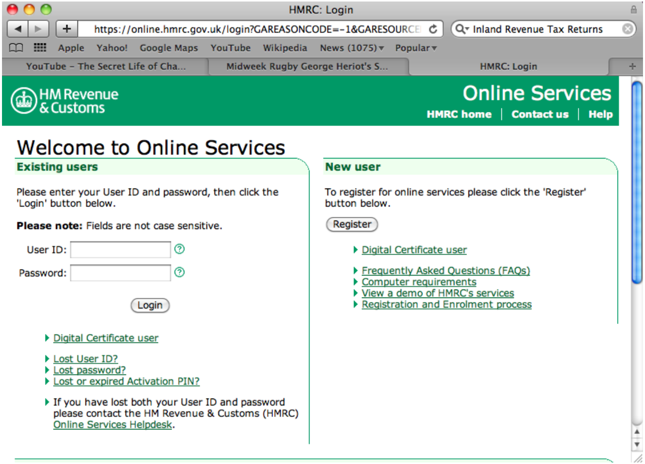
Collective Behaviour

This is also true in the man-made and engineered world:

Love Parade, Germany 2006

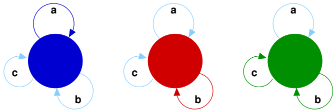
Collective Behaviour

This is also true in the man-made and engineered world:

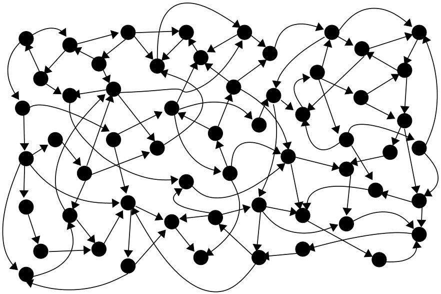


Self assessment tax returns 31st January each year

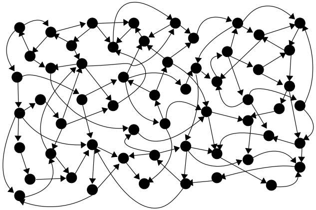
Solving discrete state models



With compositional modelling approaches we have a CTMC with global states determined by the local states of all the participating components.



Solving discrete state models

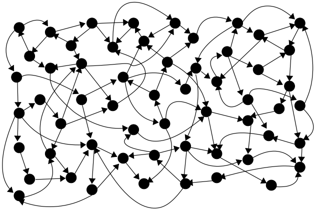


When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models

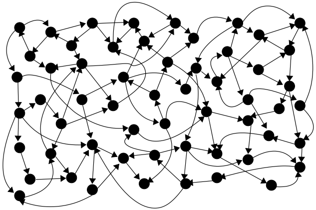


When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models



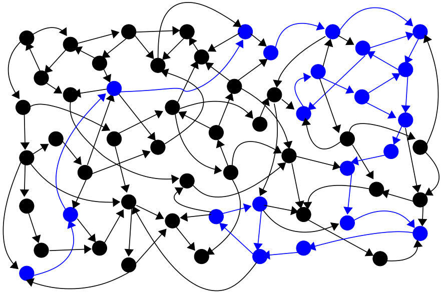
When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models

Alternatively they 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.



State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

In these cases we would like to take advantage of the mean field or fluid approximation techniques.

Use continuous state variables to approximate the discrete state space.

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

Fluid approximation-based approach

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.
- Instead the ODEs estimate the expected behaviour of the CTMC.

Appropriate for models in which there are large numbers of components of the same type, i.e. models of populations and situations of collective dynamics.

Fluid approximation-based approach

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.
- Instead the ODEs estimate the expected behaviour of the CTMC.

Appropriate for models in which there are large numbers of components of the same type, i.e. models of populations and situations of collective dynamics.

Fluid approximation-based approach

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.
- Instead the ODEs estimate the expected behaviour of the CTMC.

Appropriate for models in which there are large numbers of components of the same type, i.e. models of populations and situations of collective dynamics.

Fluid approximation-based approach

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.
- Instead the ODEs estimate the expected behaviour of the CTMC.

Appropriate for models in which there are large numbers of components of the same type, i.e. models of populations and situations of collective dynamics.

Fluid approximation-based approach

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.
- Instead the ODEs estimate the expected behaviour of the CTMC.

Appropriate for models in which there are large numbers of components of the same type, i.e. models of populations and situations of collective dynamics.

Fluid approximation theorem

Hypothesis

- $\overline{\mathbf{X}}^{(N)}(t)$: a sequence of normalized population CTMC, residing in $E \subset \mathbb{R}^n$
- $\exists \mathbf{x_0} \in S$ such that $\overline{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ in probability (initial conditions)
- $\mathbf{x}(t)$: solution of $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = F(\mathbf{x})$, $\mathbf{x}(0) = \mathbf{x_0}$, residing in E.

Theorem

For any finite time horizon $T < \infty$, it holds that:

$$\mathbb{P}\left(\sup_{0 \le t \le T} ||\overline{\mathbf{X}}^{(N)}(t) - \mathbf{x}(t)|| > \varepsilon\right) \to 0.$$

T.G.Kurtz. *Solutions of ordinary differential equations as limits of pure jump Markov processes.* Journal of Applied Probability, 1970.

Outline

Outline

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

- 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

- 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 $\xrightarrow{\text{SOS rules}}$

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 $\xrightarrow{\text{SOS rules}}$ Labelled transition system

A simple example: processors and resources

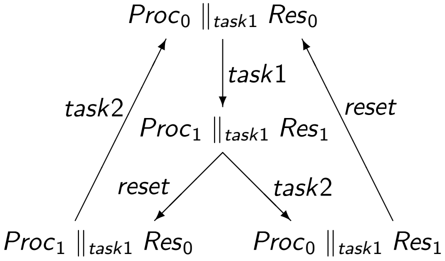
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} task1.Proc_1 \\
Proc_1 &\stackrel{def}{=} task2.Proc_0 \\
Res_0 &\stackrel{def}{=} task1.Res_1 \\
Res_1 &\stackrel{def}{=} reset.Res_0
\end{aligned}
$$

$$Proc_0 \parallel_{task1} Res_0$$

A simple example: processors and resources

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

$$Proc_0 \parallel_{task1} Res_0$$



Stochastic process algebras

Stochastic process algebra

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

Stochastic 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 CTMC for performance modelling.

Stochastic 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 CTMC for performance modelling.

Stochastic 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 CTMC for performance modelling.

Stochastic 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 CTMC for performance modelling.

Stochastic 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 CTMC for performance modelling.

Stochastic 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 CTMC for performance modelling.

SPA MODEL

Stochastic 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 CTMC for performance modelling.

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

Stochastic 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 CTMC for performance modelling.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM

Stochastic 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 CTMC for performance modelling.

$$\text{SPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}}$$

Stochastic 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 CTMC for performance modelling.

$$\text{SPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}} \text{CTMC } \mathbf{Q}$$

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
$P_1 \bowtie_L P_2$	Co-operation
P/L	Hiding
X	Variable

$P_1 \parallel P_2$ 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.

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

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
X	Variable

$P_1 \parallel P_2$ 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).

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

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

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 ($\alpha \notin L$)

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

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

Apparent Rate

$$
\begin{aligned}
r_\alpha((\beta, r).P) &= \begin{cases} r & \beta = \alpha \\ 0 & \beta \neq \alpha \end{cases} \\
r_\alpha(P + Q) &= r_\alpha(P) + r_\alpha(Q) \\
r_\alpha(A) &= r_\alpha(P) \qquad \text{where } A \stackrel{\mathit{def}}{=} P \\
r_\alpha(P \underset{L}{\bowtie} Q) &= \begin{cases} r_\alpha(P) + r_\alpha(Q) & \alpha \notin L \\ \mathit{min}(r_\alpha(P), r_\alpha(Q)) & \alpha \in L \end{cases} \\
r_\alpha(P/L) &= \begin{cases} r_\alpha(P) & \alpha \notin L \\ 0 & \alpha \in L \end{cases}
\end{aligned}
$$

Bounded capacity

We assume that components have bounded capacity: they cannot be made to go any faster than their local definition of rate for a shared activity.

Structured Operational Semantics: Hiding

Hiding

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

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

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

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

A simple example: processors and resources

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

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

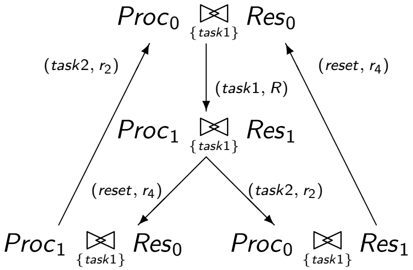
$$R = \min(r_1, r_3)$$

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

A simple example: processors and resources

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

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



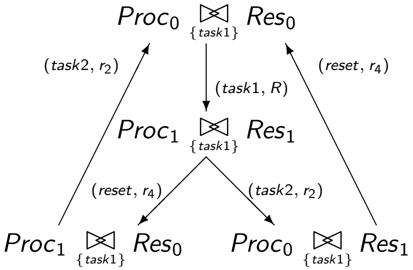
$$
R = \min(r_1, r_3)
$$

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

A simple example: processors and resources

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

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



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

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

Outline

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the recent CODA project we investigated the use of stochastic process algebras modelling and analysing the collective dynamics of large systems of interacting entities.

In the soon-to-start QUANTICOL project we will be extending techniques to spatially inhomogeneous systems.

Simple example revisited

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

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

Simple example revisited

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

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

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

Simple example revisited

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

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

- $task1$ 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

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

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

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = -\min(r_1\,x_1, r_3\,x_3) + r_2\,x_2$$

$$x_1 = \text{no. of } Proc_1$$

- $task1$ decreases $Proc_0$
- $task1$ is performed by $Proc_0$ and Res_0
- $task2$ increases $Proc_0$
- $task2$ is performed by $Proc_1$

Simple example revisited

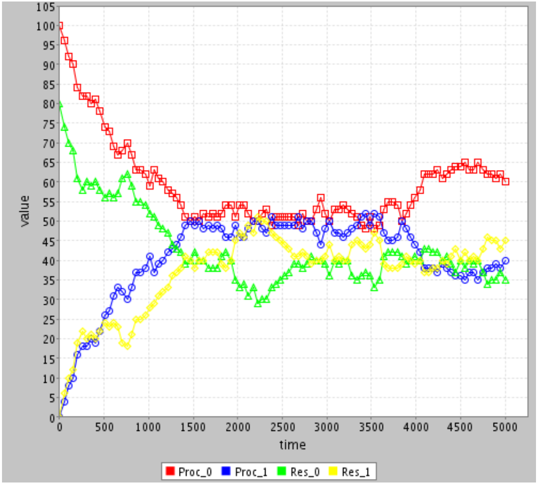
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

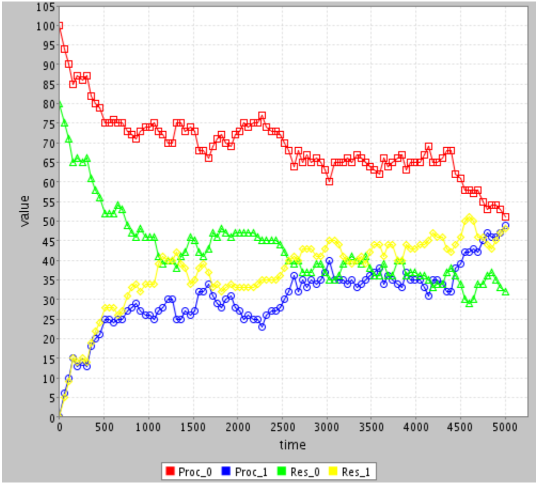
ODE interpretation

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

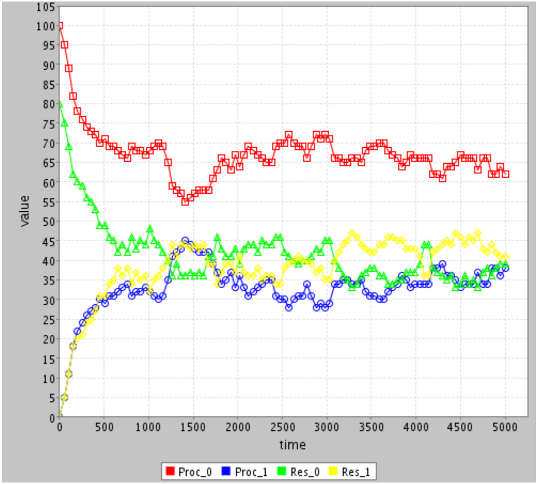
100 processors and 80 resources (simulation run A)



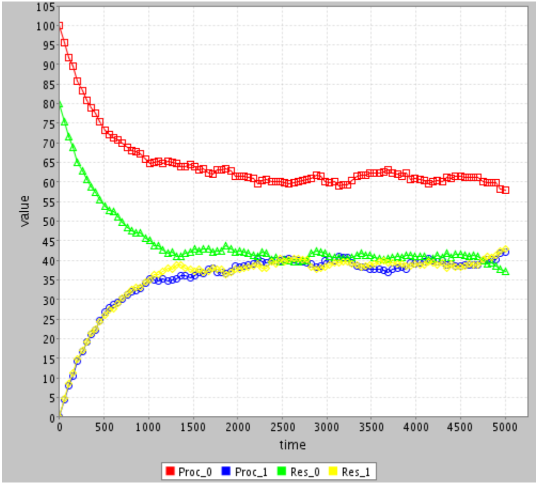
100 processors and 80 resources (simulation run B)



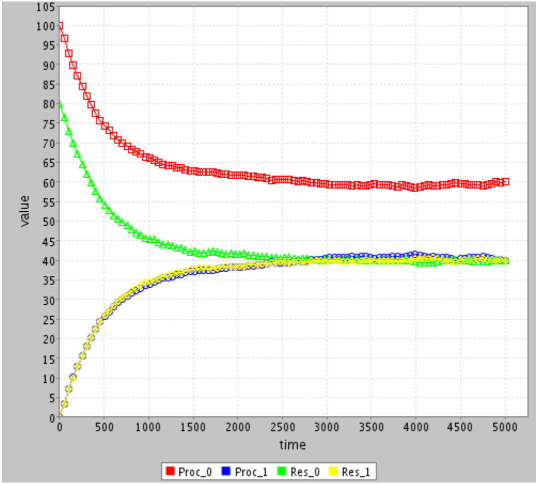
100 processors and 80 resources (simulation run C)



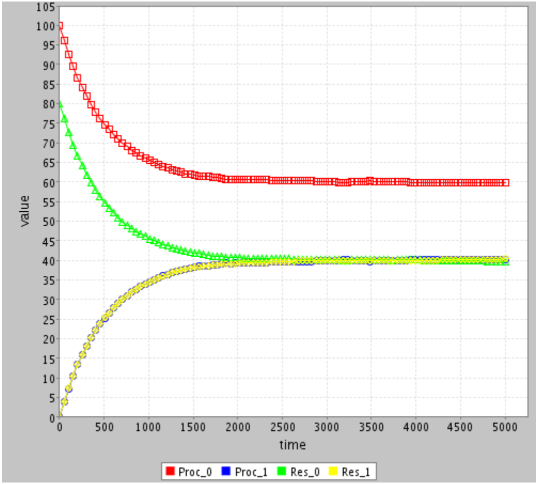
100 processors and 80 resources (average of 10 runs)



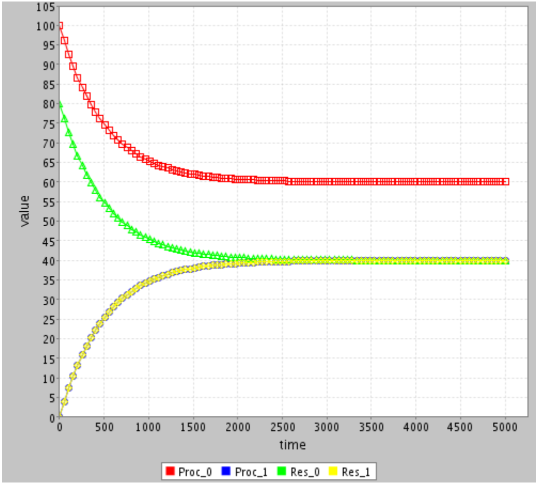
100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (ODE solution)



Outline

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The exisiting (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The exisiting (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The exisiting (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM $\xrightarrow[\text{diagram}]{\text{state transition}}$ CTMC $\mathbf{Q}$

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The exisiting (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

We define a structured operational semantics which defines the possible transitions of an arbitrary abstract state and from this derive the ODEs.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

Deriving a Fluid Approximation of a PEPA model

The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

The exisiting (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

We define a structured operational semantics which defines the possible transitions of an arbitrary abstract state and from this derive the ODEs.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components (Context Reduction)
2. Collect the transitions of the reduced context (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components (Context Reduction)
2. Collect the transitions of the reduced context (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components (Context Reduction)
2. Collect the transitions of the reduced context (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components (Context Reduction)
2. Collect the transitions of the reduced context (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Fluid Structured Operational Semantics

In order to get to the implicit representation of the CTMC we need to:

1. Remove excess components (Context Reduction)
2. Collect the transitions of the reduced context (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Once this is done we can extract the vector field $F_{\mathcal{M}}(x)$ from the jump multiset.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Context Reduction

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\Downarrow$$

$$\mathcal{R}(System) = \{Proc_0, Proc_1\} \underset{\{task1\}}{\bowtie} \{Res_0, Res_1\}$$

Population Vector

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

Context Reduction

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\Downarrow$$

$$\mathcal{R}(System) = \{Proc_0, Proc_1\} \underset{\{task1\}}{\bowtie} \{Res_0, Res_1\}$$

Population Vector

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

Location Dependency

$$System \stackrel{def}{=} Proc_0[N'_C] \underset{\{task1\}}{\bowtie} Res_0[N_S] \parallel Proc_0[N''_C]$$

$$\Downarrow$$

$$\{Proc_0, Proc_1\} \underset{\{task1\}}{\bowtie} \{Res_0, Res_1\} \parallel \{Proc_0, Proc_1\}$$

Population Vector

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$$

Location Dependency

$$System \stackrel{def}{=} Proc_0[N'_C] \underset{\{task1\}}{\bowtie} Res_0[N_S] \parallel Proc_0[N''_C]$$

$$\Downarrow$$

$$\{Proc_0, Proc_1\} \underset{\{task1\}}{\bowtie} \{Res_0, Res_1\} \parallel \{Proc_0, Proc_1\}$$

Population Vector

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$$

Location Dependency

$$System \stackrel{def}{=} Proc_0[N'_C] \underset{\{task1\}}{\bowtie} Res_0[N_S] \parallel Proc_0[N''_C]$$

$$\Downarrow$$

$$\{Proc_0, Proc_1\} \underset{\{task1\}}{\bowtie} \{Res_0, Res_1\} \parallel \{Proc_0, Proc_1\}$$

Population Vector

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$$

Fluid Structured Operational Semantics by Example

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

$$
\frac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \quad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1}
$$

Fluid Structured Operational Semantics by Example

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

$$
\dfrac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1}
$$

Fluid Structured Operational Semantics by Example

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

$$
\dfrac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1}
$$

Fluid Structured Operational Semantics by Example

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$$

$$
\frac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1}
$$

Apparent Rate Calculation

$$\frac{\dfrac{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1'} \mathit{Proc_1}}{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1' \xi_1}_* \mathit{Proc_1}} \qquad \dfrac{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_3} \mathit{Res_1}}{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_3 \xi_3}_* \mathit{Res_1}}}{\mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} \xrightarrow{\mathit{task1}, r(\xi)}_* \mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_1}}$$

$$\begin{aligned} r(\xi) &= \frac{r_1 \xi_1}{r^\star_{\mathit{task1}}\left(\mathit{Proc_0}, \xi\right)} \frac{r_3 \xi_4}{r^\star_{\mathit{task1}}\left(\mathit{Res_0}, \xi\right)} \min\left(r^\star_{\mathit{task1}}\left(\mathit{Proc_0}, \xi\right), r^\star_{\mathit{task1}}\left(\mathit{Res_0}, \xi\right)\right) \\ &= \frac{r_1 \xi_1}{r_1 \xi_1} \frac{r_3 \xi_4}{r_3 \xi_4} \min\left(r_1 \xi_1, r_3 \xi_4\right) \\ &= \min\left(r_1 \xi_1, r_3 \xi_4\right) \end{aligned}$$

Apparent Rate Calculation

$$\dfrac{\dfrac{Proc_0 \xrightarrow{task1,r_1'} Proc_1}{Proc_0 \xrightarrow{task1,r_1'\xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1,r_3} Res_1}{Res_0 \xrightarrow{task1,r_3\xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1,r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1}$$

$$\begin{aligned}
r(\xi) &= \frac{r_1\xi_1}{r^\star_{task1}\left(Proc_0,\xi\right)} \; \frac{r_3\xi_4}{r^\star_{task1}\left(Res_0,\xi\right)} \min\left(r^\star_{task1}\left(Proc_0,\xi\right), r^\star_{task1}\left(Res_0,\xi\right)\right) \\
&= \frac{r_1\xi_1}{r_1\xi_1} \; \frac{r_3\xi_4}{r_3\xi_4} \min\left(r_1\xi_1, r_3\xi_4\right) \\
&= \min\left(r_1\xi_1, r_3\xi_4\right)
\end{aligned}$$

$f(\xi, l, \alpha)$ as the Generator Matrix of the *Lumped* CTMC

$$
(P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_0) \xrightarrow{r}
\begin{cases}
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} R_1 \parallel R_0 \parallel R_0) \\
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1)
\end{cases}
$$

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

$f(\xi, l, \alpha)$ as the Generator Matrix of the *Lumped* CTMC

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

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

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

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

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

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

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

$f(\xi, l, \alpha)$ as the Generator Matrix of the *Lumped* CTMC

$$(P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_0) \xrightarrow{r} \begin{cases} (P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} R_1 \parallel R_0 \parallel R_0) \\ (P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\ (P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1) \\ (P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0) \\ (P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\ (P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1) \end{cases}$$

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

$f(\xi, l, \alpha)$ as the Generator Matrix of the *Lumped* CTMC

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

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

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

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

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

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

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

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

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

Jump Multiset

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$r(\xi) = \min\left(r_1 \xi_1, r_3 \xi_3\right)$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

Jump Multiset

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$r(\xi) = \min\left(r_1\xi_1, r_3\xi_3\right)$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

Jump Multiset

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_{*} Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$r(\xi) = \min\left(r_1\xi_1, r_3\xi_3\right)$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2}_{*} Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_{*} Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

Equivalent Transitions

Some transitions may give the same information:

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$
$$Proc_1 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_0$$

i.e., Res_1 may perform an action independently from the rest of the system.

This is captured by the procedure used for the construction of the generator function $f(\xi, l, \alpha)$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

- Take $l = (0, 0, 0, 0)$
- Add -1 to all elements of l corresponding to the indices of the components in the lhs of the transition

$$l = (-1, 0, 0, -1)$$

- Add $+1$ to all elements of l corresponding to the indices of the components in the rhs of the transition

$$l = (-1 + 1, 0, +1, -1) = (0, 0, +1, -1)$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_{*} Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

- Take $l = (0, 0, 0, 0)$
- Add -1 to all elements of l corresponding to the indices of the components in the lhs of the transition

$$l = (-1, 0, 0, -1)$$

- Add $+1$ to all elements of l corresponding to the indices of the components in the rhs of the transition

$$l = (-1 + 1, 0, +1, -1) = (0, 0, +1, -1)$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

- Take $l = (0, 0, 0, 0)$
- Add -1 to all elements of l corresponding to the indices of the components in the lhs of the transition

$$l = (-1, 0, 0, -1)$$

- Add $+1$ to all elements of l corresponding to the indices of the components in the rhs of the transition

$$l = (-1 + 1, 0, +1, -1) = (0, 0, +1, -1)$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

- Take $l = (0, 0, 0, 0)$
- Add -1 to all elements of l corresponding to the indices of the components in the lhs of the transition

$$l = (-1, 0, 0, -1)$$

- Add $+1$ to all elements of l corresponding to the indices of the components in the rhs of the transition

$$l = (-1 + 1, 0, +1, -1) = (0, 0, +1, -1)$$

$$f\left(\xi, (0, 0, +1, -1), reset\right) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2'}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$f(\xi, (-1, +1, -1, +1), task1) = r(\xi)$$

$$f(\xi, (+1, -1, 0, 0), task2) = \xi_2 r_2$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_{*} Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2'}_{*} Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_{*} Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$f(\xi, (-1, +1, -1, +1), task1) = r(\xi)$$

$$f(\xi, (+1, -1, 0, 0), task2) = \xi_2 r_2$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Construction of $f(\xi, l, \alpha)$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$

$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2'}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

$$f(\xi, (-1, +1, -1, +1), task1) = r(\xi)$$

$$f(\xi, (+1, -1, 0, 0), task2) = \xi_2 r_2$$

$$f(\xi, (0, 0, +1, -1), reset) = \xi_4 r_4$$

Capturing behaviour in the Generator Function

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

Numerical Vector Form

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4) \in \mathbb{N}^4, \quad \xi_1 + \xi_2 = N_P \;\text{ and }\; \xi_3 + \xi_4 = N_R$$

Generator Function

$$
f(\xi, l, \alpha): \quad
\begin{aligned}
f(\xi, (-1, 1, -1, 1), task1) &= \min(r_1\xi_1, r_3\xi_3) \\
f(\xi, (1, -1, 0, 0), task2) &= r_2\xi_2 \\
f(\xi, (0, 0, 1, -1), reset) &= r_4\xi_4
\end{aligned}
$$

Capturing behaviour in the Generator Function

$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0 \\
System &\stackrel{def}{=} Proc_0[N_P] \underset{\{transfer\}}{\bowtie} Res_0[N_R]
\end{aligned}
$$

Numerical Vector Form

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4) \in \mathbb{N}^4, \quad \xi_1 + \xi_2 = N_P \text{ and } \xi_3 + \xi_4 = N_R$$

Generator Function

$$
f(\xi, l, \alpha) : \quad
\begin{aligned}
f(\xi, (-1, 1, -1, 1), task1) &= \min(r_1\xi_1, r_3\xi_3) \\
f(\xi, (1, -1, 0, 0), task2) &= r_2\xi_2 \\
f(\xi, (0, 0, 1, -1), reset) &= r_4\xi_4
\end{aligned}
$$

Capturing behaviour in the Generator Function

$$
\begin{array}{rcl}
\mathit{Proc_0} & \stackrel{def}{=} & (\mathit{task1}, r_1).\mathit{Proc_1} \\
\mathit{Proc_1} & \stackrel{def}{=} & (\mathit{task2}, r_2).\mathit{Proc_0} \\
\mathit{Res_0} & \stackrel{def}{=} & (\mathit{task1}, r_3).\mathit{Res_1} \\
\mathit{Res_1} & \stackrel{def}{=} & (\mathit{reset}, r_4).\mathit{Res_0} \\
\mathit{System} & \stackrel{def}{=} & \mathit{Proc_0}[N_P] \underset{\{\mathit{transfer}\}}{\bowtie} \mathit{Res_0}[N_R]
\end{array}
$$

Numerical Vector Form

$$\xi = (\xi_1, \xi_2, \xi_3, \xi_4) \in \mathbb{N}^4, \quad \xi_1 + \xi_2 = N_P \ \text{ and } \ \xi_3 + \xi_4 = N_R$$

Generator Function

$$
f(\xi, l, \alpha): \quad
\begin{array}{rcl}
f(\xi, (-1, 1, -1, 1), \mathit{task1}) & = & \min(r_1\xi_1, r_3\xi_3) \\
f(\xi, (1, -1, 0, 0), \mathit{task2}) & = & r_2\xi_2 \\
f(\xi, (0, 0, 1, -1), \mathit{reset}) & = & r_4\xi_4
\end{array}
$$

Extraction of the ODE from f

Generator Function

$$\begin{aligned}
f(\xi, (-1,1,-1,1), \mathit{task1}) &= \min(r_1\xi_1, r_3\xi_3) \\
f(\xi, (1,-1,0,0), \mathit{task2}) &= r_2\xi_2 \\
f(\xi, (0,0,1,-1), \mathit{reset}) &= r_4\xi_4
\end{aligned}$$

Differential Equations

$$\begin{aligned}
\frac{\mathrm{d}x}{\mathrm{d}t} = F_{\mathcal{M}}(x) &= \sum_{l \in \mathbb{Z}^d} l \sum_{\alpha \in \mathcal{A}} f(x, l, \alpha) \\
&= (-1,1,-1,1)\min(r_1x_1, r_3x_3) + (1,-1,0,0)r_2x_2 \\
&\quad + (0,0,1,-1)r_4x_4
\end{aligned}$$

Extraction of the ODE from f

Generator Function

$$
\begin{aligned}
f(\xi,(-1,1,-1,1),\mathit{task1}) &= \min(r_1\xi_1, r_3\xi_3)\\
f(\xi,(1,-1,0,0),\mathit{task2}) &= r_2\xi_2\\
f(\xi,(0,0,1,-1),\mathit{reset}) &= r_4\xi_4
\end{aligned}
$$

Differential Equations

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_1x_1, r_3x_3) + r_2x_2\\
\frac{dx_2}{dt} &= \min(r_1x_1, r_3x_3) - r_2x_2\\
\frac{dx_3}{dt} &= -\min(r_1x_1, r_3x_3) + r_4x_4\\
\frac{dx_4}{dt} &= \min(r_1x_1, r_3x_3) - r_4x_4
\end{aligned}
$$

Density Dependence

Density dependence of parametric apparent rates

Let $r^\star_\alpha(P,\xi)$ be the parametric apparent rate of action type α in process P. For any $n \in \mathbb{N}$ and $\alpha \in \mathcal{A}$,

$$r^\star_\alpha(P,\xi) = n \cdot r^\star_\alpha(P,\xi/n)$$

Density dependence of parametric transition rates

If $P \xrightarrow{(\alpha,r(\xi))}_* Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

Generating functions give rise to density dependent rates

Let $\mathcal{M}$ be a PEPA model with generating functions $f(\xi, l, \alpha)$ derived as demonstrated. Then the corresponding sequence of CTMCs will be density dependent.

Density Dependence

Density dependence of parametric apparent rates

Let $r^\star_\alpha(P,\xi)$ be the parametric apparent rate of action type α in process P. For any $n \in \mathbb{N}$ and $\alpha \in \mathcal{A}$,

$$r^\star_\alpha(P,\xi) = n \cdot r^\star_\alpha(P,\xi/n)$$

Density dependence of parametric transition rates

If $P \xrightarrow{(\alpha, r(\xi))}_* Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

Generating functions give rise to density dependent rates

Let $\mathcal{M}$ be a PEPA model with generating functions $f(\xi, l, \alpha)$ derived as demonstrated. Then the corresponding sequence of CTMCs will be density dependent.

Density Dependence

Density dependence of parametric apparent rates

Let $r^\star_\alpha(P,\xi)$ be the parametric apparent rate of action type α in process P. For any $n \in \mathbb{N}$ and $\alpha \in \mathcal{A}$,

$$r^\star_\alpha(P,\xi) = n \cdot r^\star_\alpha(P,\xi/n)$$

Density dependence of parametric transition rates

If $P \xrightarrow{(\alpha,r(\xi))}_* Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

Generating functions give rise to density dependent rates

Let $\mathcal{M}$ be a PEPA model with generating functions $f(\xi, l, \alpha)$ derived as demonstrated. Then the corresponding sequence of CTMCs will be density dependent.

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity of parametric apparent rates

Let $r^{\star}_{\alpha}(P, \xi)$ be the parametric apparent rate of action type α in process P. There exists a constant $L \in \mathbb{R}$ such that for all $x, y \in \mathbb{R}^d, x \neq y$,

$$\frac{\|r^{\star}_{\alpha}(P, x) - r^{\star}_{\alpha}(P, y)\|}{\|x - y\|} \leq L$$

Lipschitz continuity of rate functions

If $P \xrightarrow{(\alpha, r(x))}_{*} P'$ then $r(x) \leq r^{\star}_{\alpha}(P, x)$ and thus it follows that $r(x)$ is Lipschitz continuous.

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity of parametric apparent rates

Let $r^\star_\alpha(P, \xi)$ be the parametric apparent rate of action type α in process P. There exists a constant $L \in \mathbb{R}$ such that for all $x, y \in \mathbb{R}^d, x \neq y$,

$$\frac{\|r^\star_\alpha(P, x) - r^\star_\alpha(P, y)\|}{\|x - y\|} \leq L$$

Lipschitz continuity of rate functions

If $P \xrightarrow{(\alpha, r(x))}_* P'$ then $r(x) \leq r^\star_\alpha(P, x)$ and thus it follows that $r(x)$ is Lipschitz continuous.

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity of parametric apparent rates

Let $r^\star_\alpha(P,\xi)$ be the parametric apparent rate of action type α in process P. There exists a constant $L \in \mathbb{R}$ such that for all $x, y \in \mathbb{R}^d, x \neq y$,

$$\frac{\|r^\star_\alpha(P,x) - r^\star_\alpha(P,y)\|}{\|x-y\|} \leq L$$

Lipschitz continuity of rate functions

If $P \xrightarrow{(\alpha, r(x))}_* P'$ then $r(x) \leq r^\star_\alpha(P,x)$ and thus it follows that $r(x)$ is Lipschitz continuous.

Kurtz's Theorem

Kurtz's Theorem for PEPA

Let $x(t), 0 \leq t \leq T$ satisfy the initial value problem $\frac{dx}{dt} = F(x(t)),\ x(0) = \delta$, specified from a PEPA model.

Let $\{X_n(t)\}$ be a family of CTMCs with parameter $n \in \mathbb{N}$ generated as explained and let $X_n(0) = n \cdot \delta$. Then,

$$\forall \varepsilon > 0 \lim_{n \to \infty} \mathbb{P}\left(\sup_{t \leq T} \| X_n(t)/n - x(t) \| > \varepsilon \right) = 0.$$

Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Kurtz's Theorem

Kurtz's Theorem for PEPA

Let $x(t), 0 \leq t \leq T$ satisfy the initial value problem $\frac{dx}{dt} = F(x(t)),\ x(0) = \delta$, specified from a PEPA model.

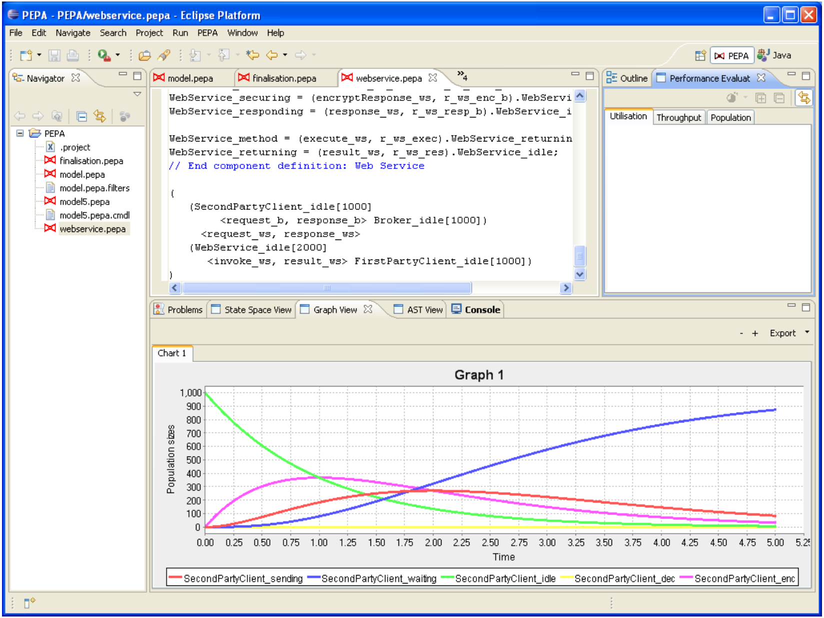
Let $\{X_n(t)\}$ be a family of CTMCs with parameter $n \in \mathbb{N}$ generated as explained and let $X_n(0) = n \cdot \delta$. Then,

$$\forall \varepsilon > 0 \lim_{n \to \infty} \mathbb{P}\left(\sup_{t \leq T} \| X_n(t)/n - x(t) \| > \varepsilon \right) = 0.$$

Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

Eclipse Plug-in for PEPA



Outline

Designing for human crowd dynamics

- Widespread take up of mobile and communicating computational devices is making pervasive systems a reality and creating new ways for us to interact with our environment, an informatic environment.
- One application is to provide routing information to help people navigate through unfamiliar locations.
- In theses case the dynamic behaviour of the system as a whole is important to ensure the satisfaction of the users.
- Using a stochastic process algebra allows quantified information, necessary for dynamic analysis, to be captured whilst also focussing on the behaviour of the individuals and their interactions with the environment.

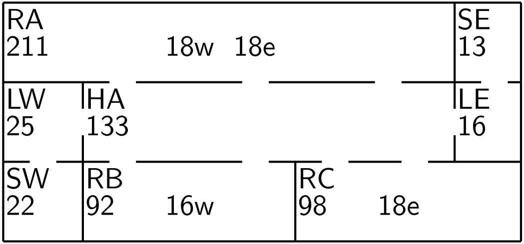
Example scenario: emergency egress

Emergency egress can be regarded as a particular case of crowd dynamics, when the location may be familiar but circumstances may alter the usual topology and make efficient movement particularly important.

Here technology mediation may mean that information about the best routes (possibly contradicting signage) can be supplied dynamically.

M.Massink, D.Latella, A.Bracciali, M.Harrison and J.Hillston. Scalable Context-dependent Analysis of Emergency Egress Models. FACS 2012.

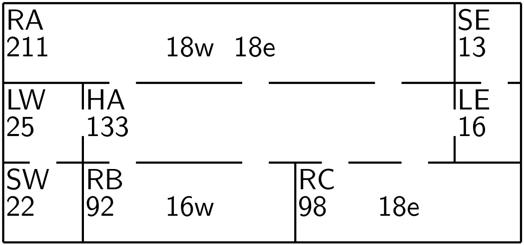
Example scenario



The layout of the building is described in terms of the arrangement of the rooms, hallways, landing and stairs. Each has a capacity and may have an initial occupancy.

Process algebra components describe the behaviours of individuals, but also rooms and information dissemination.

Example scenario



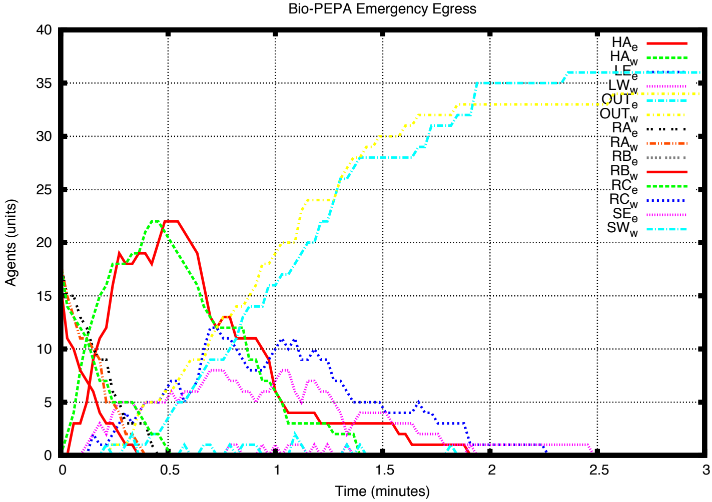
The layout of the building is described in terms of the arrangement of the rooms, hallways, landing and stairs. Each has a capacity and may have an initial occupancy.

Process algebra components describe the behaviours of individuals, but also rooms and information dissemination.

Model specification

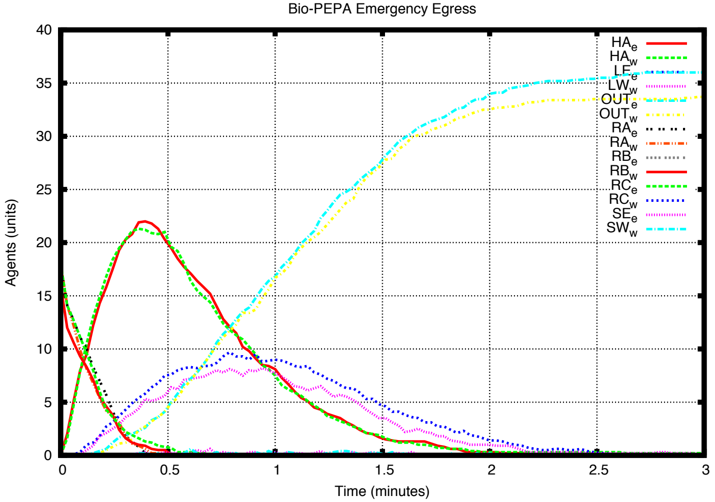
```
// BUILDING LAYOUT  (COMPARTMENTS)

location ra :        size = normal_room,  type= compartment;
location d1_ra_ha : size = normal_door,  type= compartment;
...
// PARAMETERS SET UP

to_ra_d1      =  6;                  // 3 + (60/7);
from_d1       =  door_exit_rate;
...
occupancy_d1 =  D1_ra_e@d1_ra_ha + D1_ra_w@d1_ra_ha + ... :
full_d1       =  H(capacity_d1 - occupancy_d1);
switch_d1     =  open_d1*full_d1;
...
// AGENT DYNAMICS    (FUNCTIONAL RATES)

//   From ra to ha through d1
kineticLawOf ra_e_in__d1_ra_ha   : fMA(to_ra_d1 * switch_d1 * ra_e_in_safe);
kineticLawOf ha_e_out__d1_ra_ha : fMA(from_d1 *  open_d1 * safeD1_ra_e * allowance_ha);
kineticLawOf ra_w_in__d1_ra_ha   : ...
kineticLawOf ha_w_out__d1_ra_ha : ...
//    ... and back
kineticLawOf ha_e_in__d1_ra_ha   : ...
...

// AGENT DEFINITIONS (SEQUENTIAL PROCESSES)

RA_e = (ra_e_in__d1_ra_ha, 1)     << RA_e@ra  +  ...
       (ra_e_out__d1_ra_ha, 1)    >> RA_e@ra  +  ...
RA_w =  ...
HA_e =  ...
...
D1_ra_e = (ra_e_in__d1_ra_ha, 1)  >> D1_ra_e@d1_ra_ha +
          (ha_e_out__d1_ra_ha, 1) << D1_ra_e@d1_ra_ha;
D1_ra_w = ...
```

Example results: room occupancy



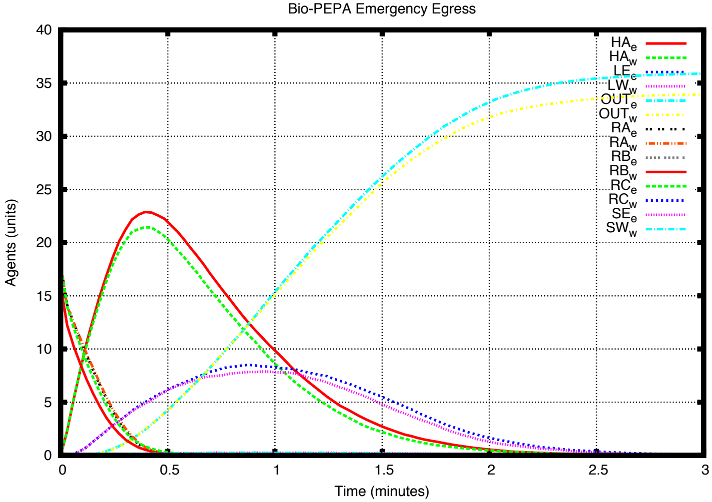
One stochastic simulation run

Example results: room occupancy



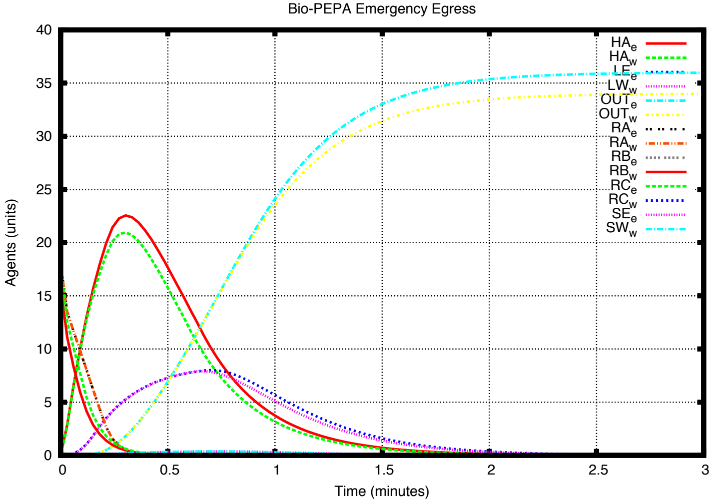
10 stochastic simulation runs

Example results: room occupancy



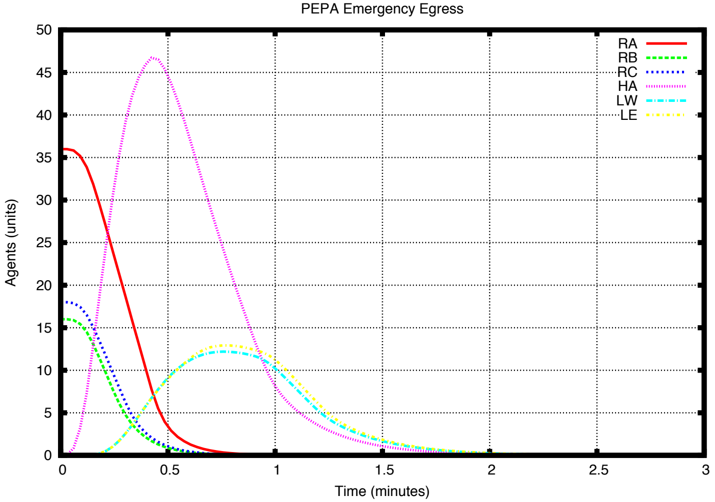
500 stochastic simulation runs

Example results: room occupancy



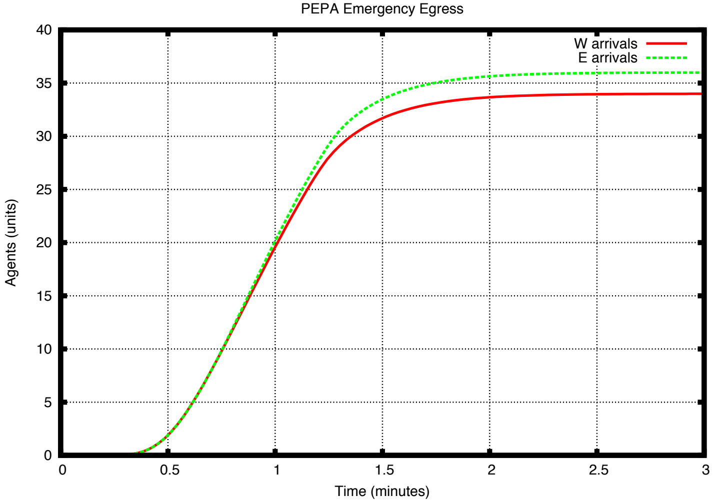
ODE numerical simulation

Results from PEPA model



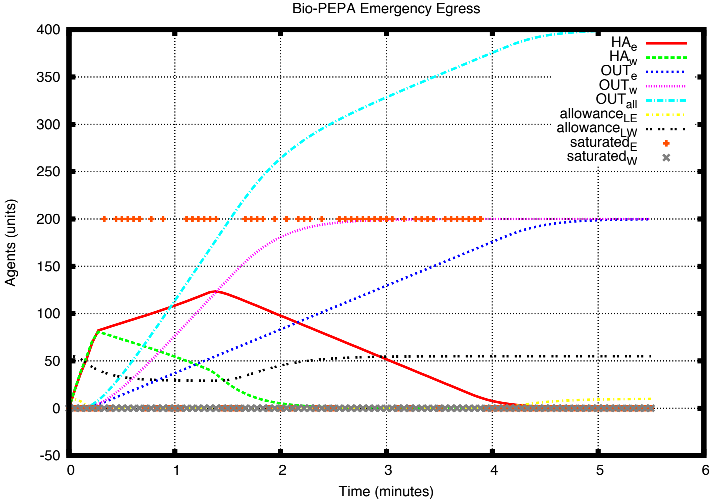
Room occupancy (PEPA model)

Results from PEPA model



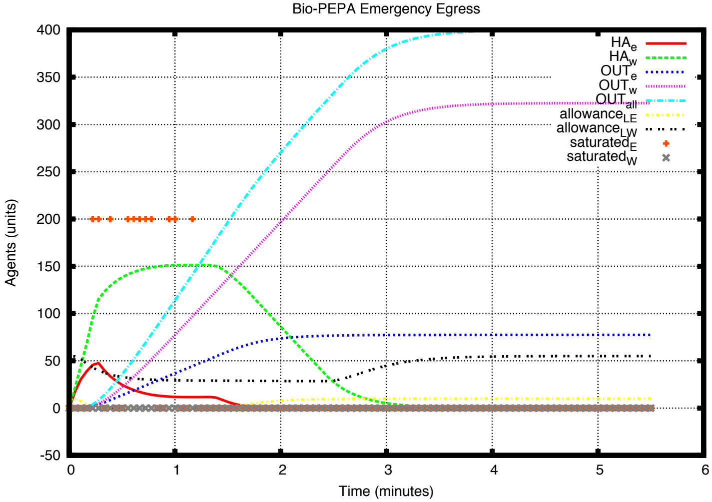
Number arrived (PEPA model)

Example results: rerouting through mediation



Room occupancy over time without rerouting capability

Example results: rerouting through mediation



Room occupancy over time with rerouting capability

Summary

Other examples we have considered include :

- Individualised routing in unfamiliar buildings such as hospitals, airports and museums.
- Crowd dynamics in cities — particularly the El Bottelon problem in squares in Spanish cities

On-going research issues:

- Good/appropriate representations of space.
- Relationship between the population level view and the individual view, particularly with respect to correctness.

Summary

Other examples we have considered include :

- Individualised routing in unfamiliar buildings such as hospitals, airports and museums.
- Crowd dynamics in cities — particularly the El Bottelon problem in squares in Spanish cities

On-going research issues:

- Good/appropriate representations of space.
- Relationship between the population level view and the individual view, particularly with respect to correctness.

Outline

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- Embedding the fluid approximation in the formal semantics of the language allows necessary conditions for the convergence to be established once and for all for the language rather than on a model-by-model basis.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- Embedding the fluid approximation in the formal semantics of the language allows necessary conditions for the convergence to be established once and for all for the language rather than on a model-by-model basis.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- Embedding the fluid approximation in the formal semantics of the language allows necessary conditions for the convergence to be established once and for all for the language rather than on a model-by-model basis.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- Embedding the fluid approximation in the formal semantics of the language allows necessary conditions for the convergence to be established once and for all for the language rather than on a model-by-model basis.