SPA for quantitative analysis: Lecture 5 — Collective Dynamics

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

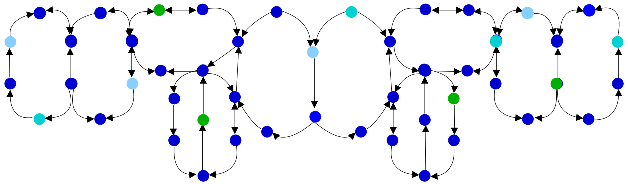
7th March 2013

Outline

Outline

Collective Dynamics

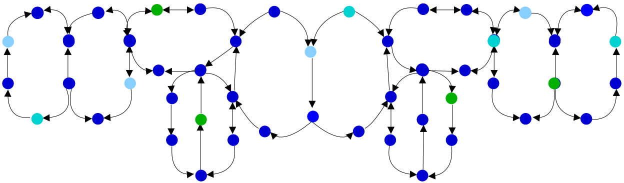
The behaviour of many systems can be interpreted as the result of the collective behaviour of a large number of interacting entities.



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

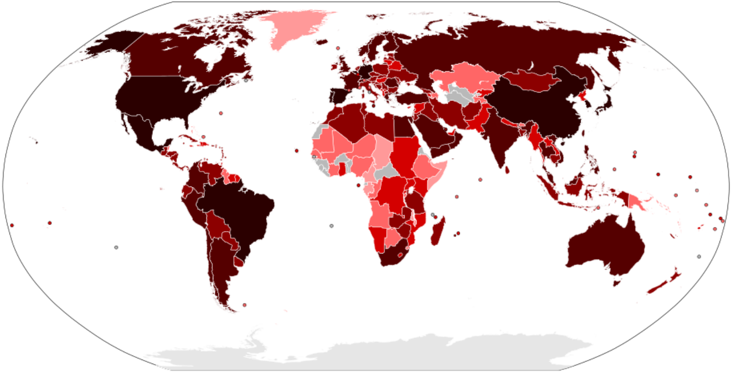
In the natural world there are many instances of collective behaviour and its consequences:

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

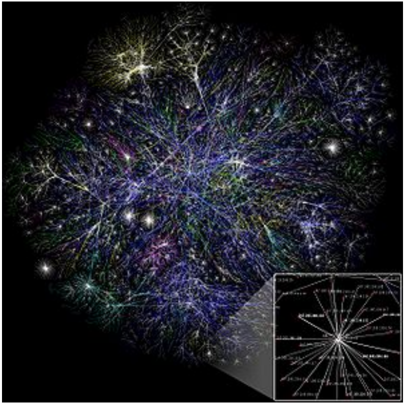
Collective Behaviour

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

Love Parade, Germany 2006

Collective Behaviour

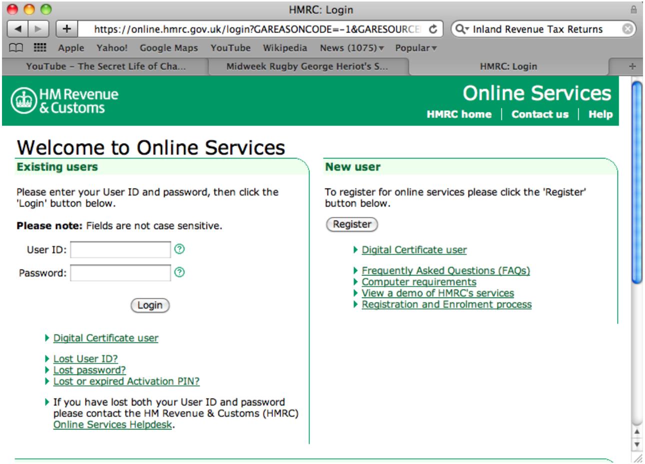
This is also true in the man-made and engineered world:



Map of the Internet 2009

Collective Behaviour

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



Self assessment tax returns 31st January each year

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.

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.

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.

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.

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.

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 CODA project we are developing stochastic process algebras and associated theory, tailored to the construction and evaluation of the collective dynamics of large systems of interacting entities.

Performance as an emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only the 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.

Performance as an emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only the 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.

Performance as an emergent behaviour

We must instead think about the performance of the collective point of view. Service providers often want to do this in any case. For example making contracts in terms of service level agreements.

Example Service Level Agreement

90% of requests receive a response within 3 seconds.

Qualitative Service Level Agreement

Less than 1% of the responses received within 3 seconds will read "System is overloaded, try again later".

Performance as an emergent behaviour

We must instead think about the performance of the collective point of view. Service providers often want to do this in any case. For example making contracts in terms of service level agreements.

Example Service Level Agreement

90% of requests receive a response within 3 seconds.

Qualitative Service Level Agreement

Less than 1% of the responses received within 3 seconds will read "System is overloaded, try again later".

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Large scale software systems
Issues of scalability are important for user satisfaction and resource efficiency but such issues are difficult to investigate using discrete state models.

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Biochemical signalling pathways
Understanding these pathways has the potential to improve the quality of life through enhanced drug treatment and better drug design.

Novelty

The novelty of the CODA project has been twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Epidemiological systems

Improved modelling of these systems could lead to improved disease prevention and treatment in nature and better security in computer systems.

Outline

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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.

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.

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.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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.

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.

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.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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.

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.

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.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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.

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.

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.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

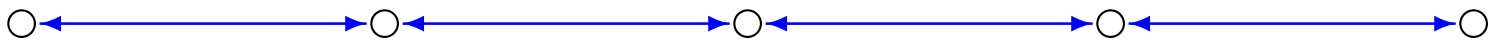
Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.



Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

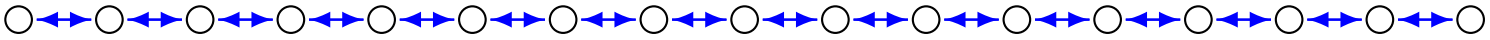
Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.



Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

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

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMC, for increasing total population N.
- We normalize such models, dividing variables by N: $\overline{\mathbf{X}}^{(N)} = \frac{\mathbf{X}}{N}$
- for each $\tau \in \mathcal{T}^{(N)}$, the normalized update is $\bar{\mathbf{v}} = \mathbf{v}/N$ and the rate function is $\bar{r}_\tau(\overline{\mathbf{X}}^{(N)}) = N f_\tau(\overline{\mathbf{X}}^{(N)})$ (density dependence).

Fluid ODE

The fluid ODE is $\dot{\mathbf{x}} = F(\mathbf{x})$, where

$$F(\mathbf{x}) = \sum_{\tau \in \mathcal{T}} \mathbf{v}_\tau f_\tau(\mathbf{x})$$

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:

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

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

New mathematical structures: differential equations

- 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.

New mathematical structures: differential equations

- 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.

New mathematical structures: differential equations

- 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.

New mathematical structures: differential equations

- 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.

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.

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

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

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

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

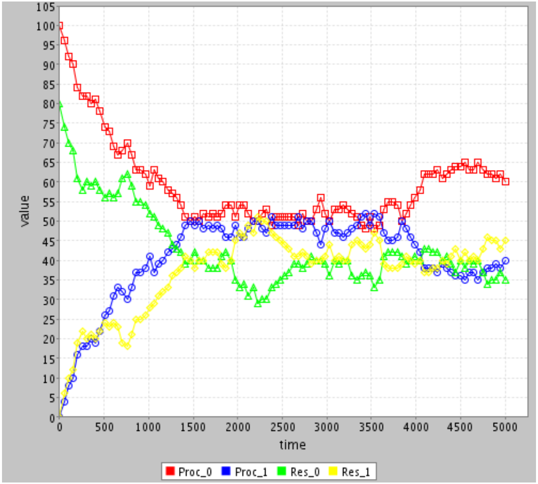
$$
\begin{aligned}
Proc_0 &\stackrel{\text{def}}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{\text{def}}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{\text{def}}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{\text{def}}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

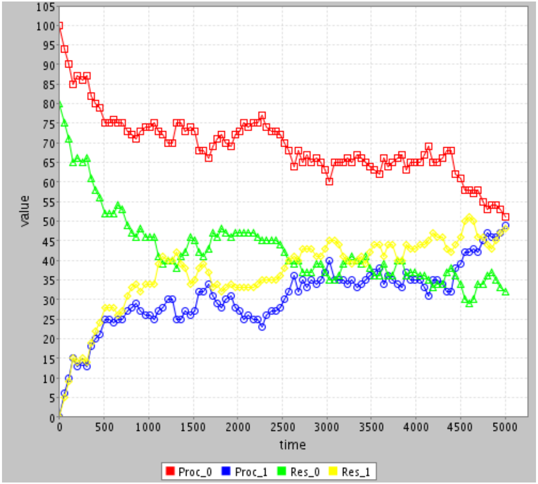
ODE interpretation

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

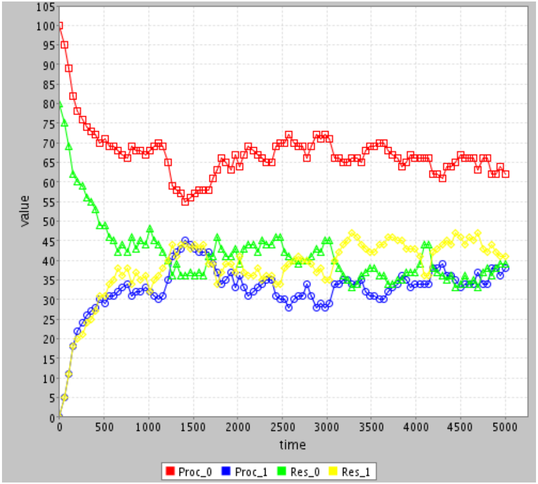
100 processors and 80 resources (simulation run A)



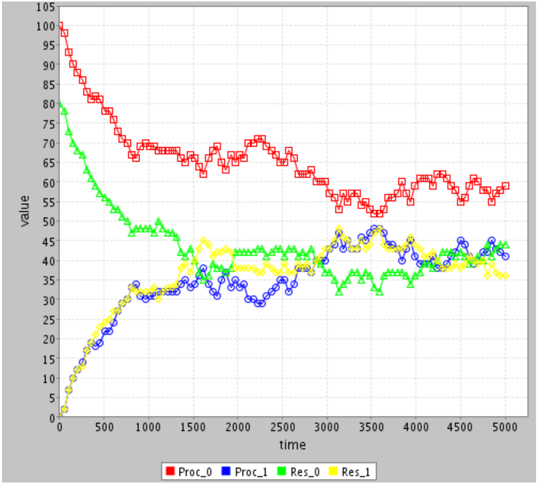
100 processors and 80 resources (simulation run B)



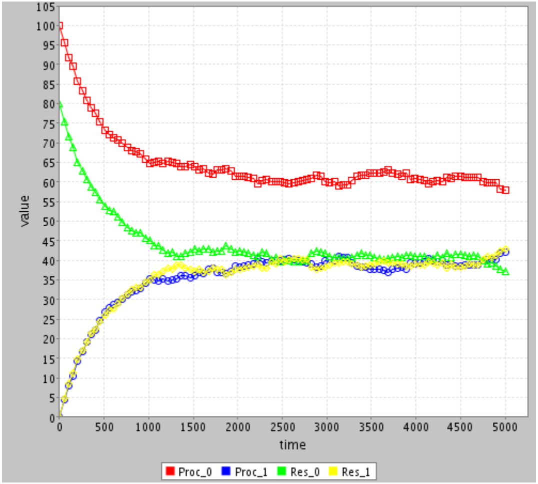
100 processors and 80 resources (simulation run C)



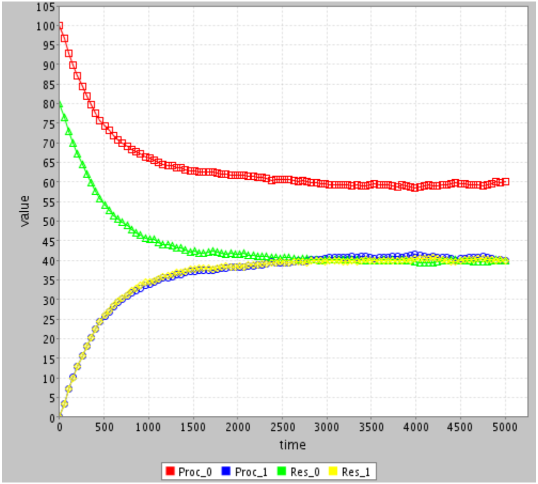
100 processors and 80 resources (simulation run D)



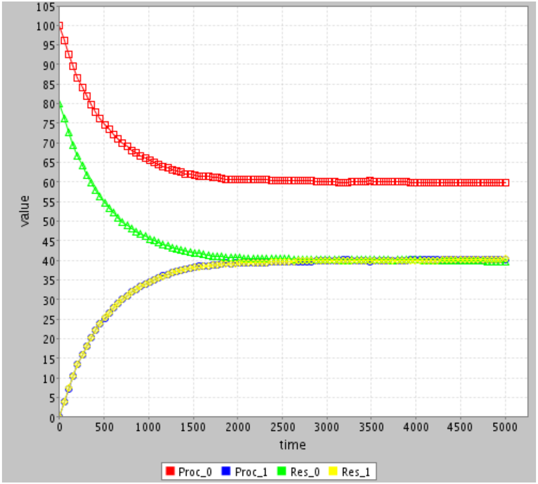
100 processors and 80 resources (average of 10 runs)



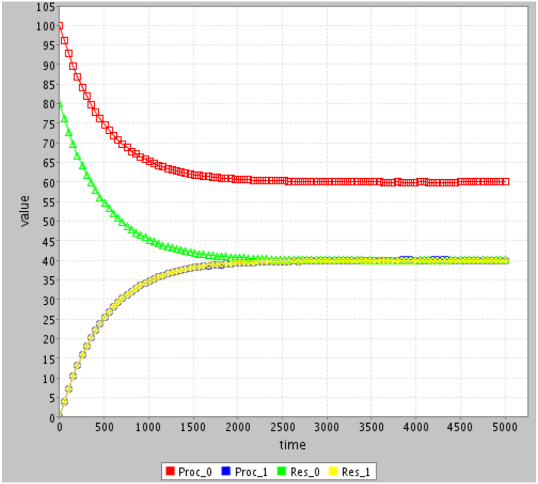
100 Processors and 80 resources (average of 100 runs)



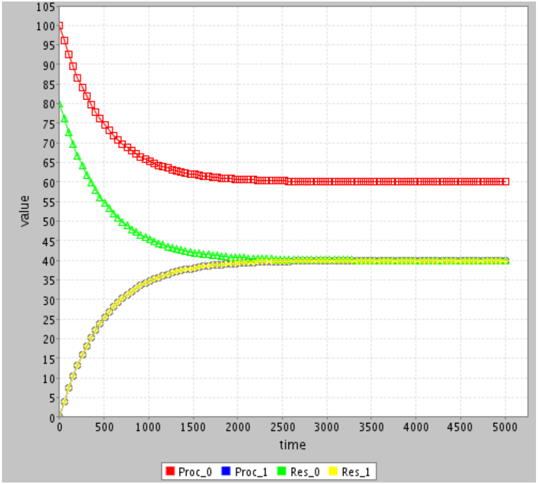
100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (average of 10000 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.

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

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

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

$$\Downarrow$$

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

Population Vector

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

Context Reduction

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

$$\Downarrow$$

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

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

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

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

$$
\begin{aligned}
r(\xi) &= \frac{r_1 \xi_1}{r^*_{task1}(Proc_0, \xi)} \frac{r_3 \xi_4}{r^*_{task1}(Res_0, \xi)} \min\left(r^*_{task1}(Proc_0, \xi), r^*_{task1}(Res_0, \xi)\right) \\
&= \frac{r_1 \xi_1}{r_1 \xi_1} \frac{r_3 \xi_3}{r_3 \xi_3} \min\left(r_1 \xi_1, r_3 \xi_3\right) \\
&= \min\left(r_1 \xi_1, r_3 \xi_3\right)
\end{aligned}
$$

Apparent Rate Calculation

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

$$
\begin{aligned}
r(\xi) &= \frac{r_1 \xi_1}{r^*_{task1}\left(Proc_0, \xi\right)} \; \frac{r_3 \xi_4}{r^*_{task1}\left(Res_0, \xi\right)} \min\left(r^*_{task1}\left(Proc_0, \xi\right), r^*_{task1}\left(Res_0, \xi\right)\right) \\
&= \frac{r_1 \xi_1}{r_1 \xi_1} \; \frac{r_3 \xi_3}{r_3 \xi_3} \min\left(r_1 \xi_1, r_3 \xi_3\right) \\
&= \min\left(r_1 \xi_1, r_3 \xi_3\right)
\end{aligned}
$$

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

$$
(P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_0) \xrightarrow{r}
\left.\begin{array}{l}
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} R_1 \parallel R_0 \parallel R_0) \\
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\
(P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0) \\
(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0 \parallel R_1)
\end{array}\right.
$$

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

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

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

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

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

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

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

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

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

Numerical Vector Form

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} \frac{\mathrm{d}x_1}{\mathrm{d}t} &= -\min(r_1x_1, r_3x_3) + r_2x_2 \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= \min(r_1x_1, r_3x_3) - r_2x_2 \\ \frac{\mathrm{d}x_3}{\mathrm{d}t} &= -\min(r_1x_1, r_3x_3) + r_4x_4 \\ \frac{\mathrm{d}x_4}{\mathrm{d}t} &= \min(r_1x_1, r_3x_3) - r_4x_4 \end{aligned}$$

Density Dependence

Density dependence of parametric apparent rates

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

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

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

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

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

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

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

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

Lipschitz continuity of rate functions

If $P \xrightarrow{(\alpha, r(x))}_* P'$ then $r(x) \leq r_\alpha^*(P, x)$ 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 $r^*_\alpha(P, \xi)$ 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 $r(x) \leq r^*_\alpha(P, x)$ 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.

Kurtz's Theorem

Kurtz's Theorem for PEPA

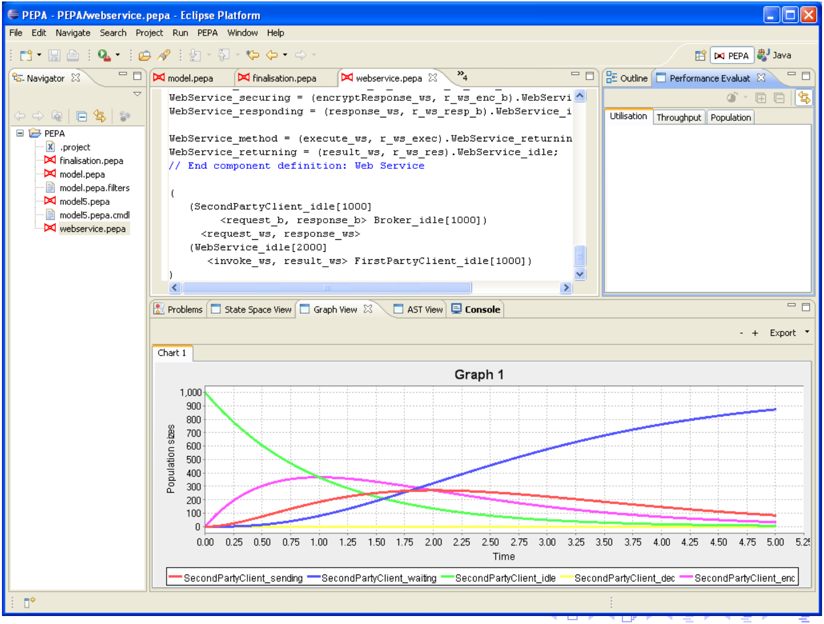
Let $x(t), 0 \leq t \leq T$ satisfy the initial value problem $\frac{dx}{dt} = F(x(t)),\ x(0) = \delta$, specified from a PEPA model.

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.

Eclipse Plug-in for PEPA



Outline

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.
- By agreement in the university, students may connect to any site to download content and use services, not just the one which is geographically closest.

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.
- By agreement in the university, students may connect to any site to download content and use services, not just the one which is geographically closest.

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.
- By agreement in the university, students may connect to any site to download content and use services, not just the one which is geographically closest.

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.
- By agreement in the university, students may connect to any site to download content and use services, not just the one which is geographically closest.

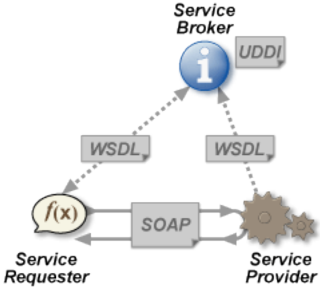
Case Study: A Virtual University

Location, Time, and Size

Replicating Web Services

Two viable approaches to cope with increasing user demand:

- use a service broker for routing

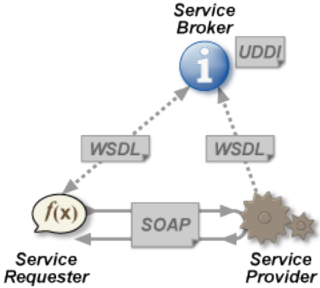


- decentralised routing

Replicating Web Services

Two viable approaches to cope with increasing user demand:

- use a service broker for routing

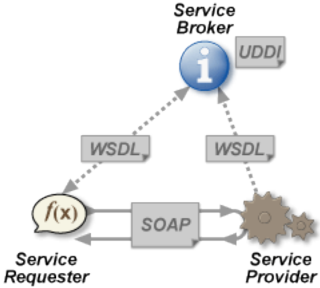


- decentralised routing

Replicating Web Services

Two viable approaches to cope with increasing user demand:

- use a service broker for routing

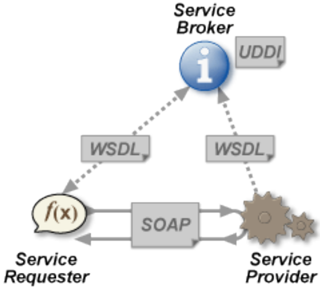


- decentralised routing

Replicating Web Services

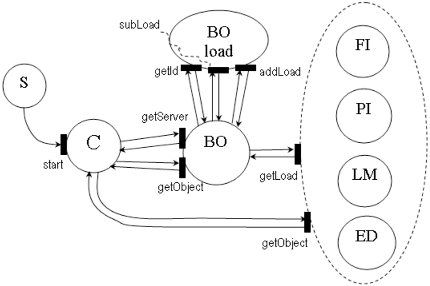
Two viable approaches to cope with increasing user demand:

- use a service broker for routing



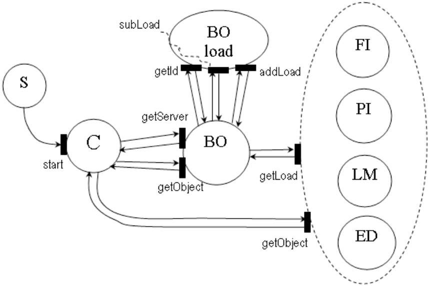
- decentralised routing

Decentralised Routing



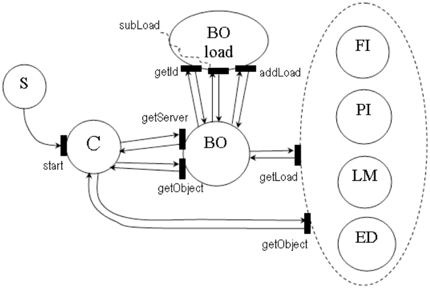
1. A client contacts a university site to download content.
2. The site either serves the request or forwards it to another site.
3. The decision in made in accord with the local service policy.

Decentralised Routing



1. A client contacts a university site to download content.
2. The site either serves the request or forwards it to another site.
3. The decision in made in accord with the local service policy.

Decentralised Routing



1. A client contacts a university site to download content.
2. The site either serves the request or forwards it to another site.
3. The decision in made in accord with the local service policy.

Model in PEPA

Clients

$$
\begin{array}{rcl}
Client_i & \stackrel{def}{=} & (connect_1, c_{1,i}).(download_1, d_{1,i}).Idle_i \\
& + & (connect_2, c_{2,i}).(download_2, d_{2,i}).Idle_i \\
& \ldots & \\
& + & (connect_m, c_{m,i}).(download_m, d_{m,i}).Idle_i \\
& + & (overload, \top).Client_i \\
Idle_i & \stackrel{def}{=} & (idle, r_{idle,i}).Client_i
\end{array}
$$

$$(1 \leq i \leq k)$$

Model in PEPA

Clients

$$
\begin{aligned}
\mathit{Client}_i &\stackrel{\mathit{def}}{=} (\mathit{connect}_1, c_{1,i}).(\mathit{download}_1, d_{1,i}).\mathit{Idle}_i \\
&+ (\mathit{connect}_2, c_{2,i}).(\mathit{download}_2, d_{2,i}).\mathit{Idle}_i \\
&\cdots \\
&+ (\mathit{connect}_m, c_{m,i}).(\mathit{download}_m, d_{m,i}).\mathit{Idle}_i \\
&+ (\mathit{overload}, \top).\mathit{Client}_i \\
\mathit{Idle}_i &\stackrel{\mathit{def}}{=} (\mathit{idle}, r_{\mathit{idle},i}).\mathit{Client}_i
\end{aligned}
$$

$(1 \leq i \leq k)$

Model in PEPA

Clients

$$
\begin{aligned}
Client_i &\stackrel{def}{=} (connect_1, c_{1,i}).(download_1, d_{1,i}).Idle_i \\
&+ (connect_2, c_{2,i}).(download_2, d_{2,i}).Idle_i \\
&\cdots \\
&+ (connect_m, c_{m,i}).(download_m, d_{m,i}).Idle_i \\
&+ (overload, \top).Client_i \\
Idle_i &\stackrel{def}{=} (idle, r_{idle,i}).Client_i
\end{aligned}
$$

$$(1 \leq i \leq k)$$

Model in PEPA

Content mirrors

$$
\begin{aligned}
Mirror_j &\stackrel{def}{=} (connect_j, f_j(s)).MirrorUploading_j \\
MirrorUploading_j &\stackrel{def}{=} (download_j, \top).Mirror_j
\end{aligned}
$$

$$(1 \leq j \leq m)$$

Model in PEPA

Content mirrors

$$
\begin{aligned}
Mirror_j &\stackrel{def}{=} (connect_j, f_j(s)).MirrorUploading_j \\
MirrorUploading_j &\stackrel{def}{=} (download_j, \top).Mirror_j
\end{aligned}
$$

$$(1 \leq j \leq m)$$

Service policies as functional rates in PEPA

The Bologna policy

Serve all requests while load is less than 75%. If more, and the loads at UNIFI, UPISA, LMU and UEDIN are at least 60%, 60%, 40% and 20% then serve the request if load is less than 95%.

$$f_{\mathrm{UNIBO}} = \begin{cases} \top & \textit{if MirrorUploading}_{\mathrm{UNIBO}} < 75 \\ \top & \textit{if MirrorUploading}_{\mathrm{UNIBO}} < 95, \\ & \quad \textit{MirrorUploading}_{\mathrm{UNIFI}} \geq 60, \\ & \quad\quad \textit{MirrorUploading}_{\mathrm{UPISA}} \geq 60, \\ & \quad\quad\quad \textit{MirrorUploading}_{\mathrm{LMU}} \geq 40, \\ & \quad\quad\quad\quad \textit{MirrorUploading}_{\mathrm{UEDIN}} \geq 20 \\ 0 & \textit{otherwise} \end{cases}$$

Service policies as functional rates in PEPA

The Bologna policy

Serve all requests while load is less than 75%. If more, and the loads at UNIFI, UPISA, LMU and UEDIN are at least 60%, 60%, 40% and 20% then serve the request if load is less than 95%.

$$f_{\mathrm{UNIBO}} = \begin{cases} \top & \text{if } \mathit{MirrorUploading}_{\mathrm{UNIBO}} < 75 \\ \top & \text{if } \mathit{MirrorUploading}_{\mathrm{UNIBO}} < 95, \\ & \quad \mathit{MirrorUploading}_{\mathrm{UNIFI}} \geq 60, \\ & \quad\quad \mathit{MirrorUploading}_{\mathrm{UPISA}} \geq 60, \\ & \quad\quad\quad \mathit{MirrorUploading}_{\mathrm{LMU}} \geq 40, \\ & \quad\quad\quad\quad \mathit{MirrorUploading}_{\mathrm{UEDIN}} \geq 20 \\ 0 & \textit{otherwise} \end{cases}$$

Model in PEPA

Dealing with overload

$$Overload \stackrel{def}{=} (overload, o(s)).Overload$$

$$o(s) = \begin{cases} \top & f_i(s) = 0, \quad 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

The system as a whole with client and mirror site populations

$$(Client_1[p_1] \parallel Client_2[p_2] \parallel \ldots \parallel Client_k[p_k]) \underset{L}{\bowtie} (Mirror_1[q_1] \parallel Mirror_2[q_2] \parallel \ldots \parallel Mirror_m[q_m])$$

Model in PEPA

Dealing with overload

$$Overload \stackrel{def}{=} (overload, o(s)).Overload$$

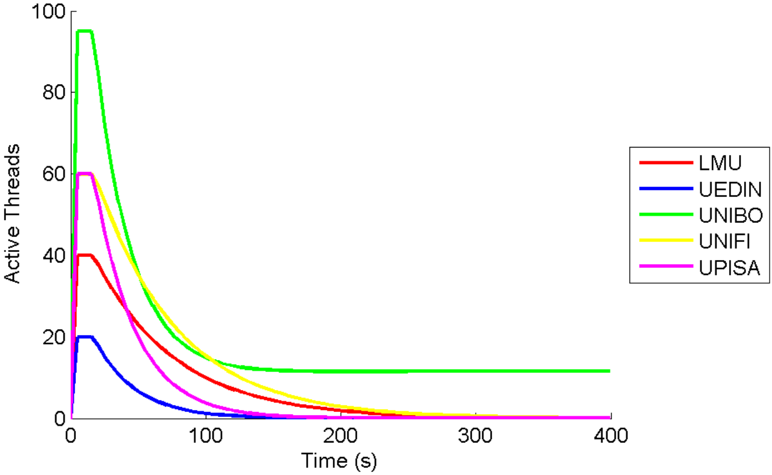
$$o(s) = \begin{cases} \top & f_i(s) = 0, \quad 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

The system as a whole with client and mirror site populations

$$\begin{aligned} &(Client_1[p_1] \parallel Client_2[p_2] \parallel \ldots \parallel Client_k[p_k]) \\ &\qquad \underset{L}{\bowtie} (Mirror_1[q_1] \parallel Mirror_2[q_2] \parallel \ldots \parallel Mirror_m[q_m]) \end{aligned}$$

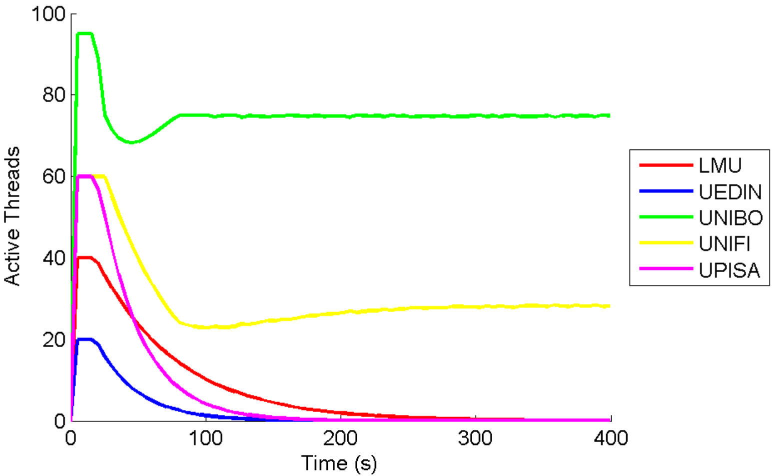
Numerical Results

$$r_{idle} = 0.001$$



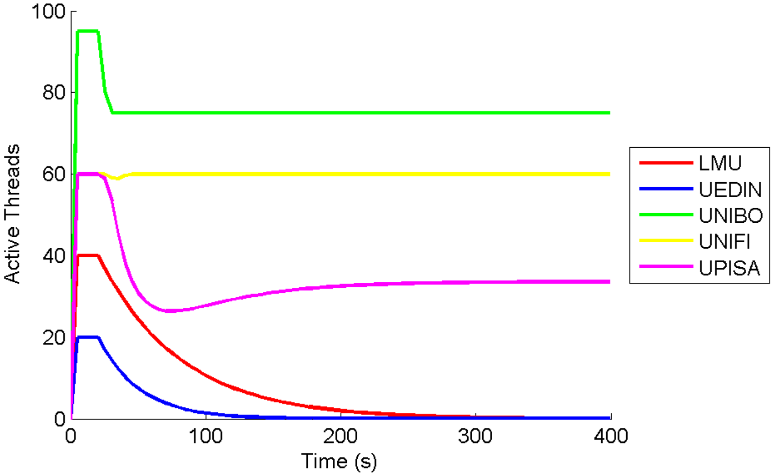
Numerical Results

$$r_{idle} = 0.01$$



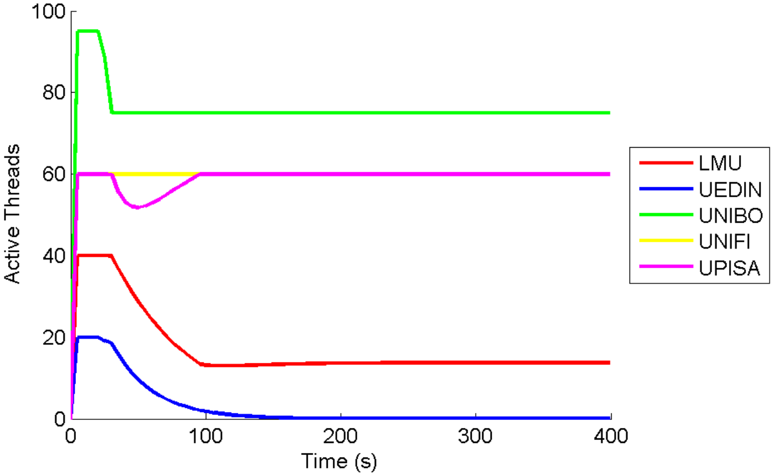
Numerical Results

$$r_{idle} = 0.02$$



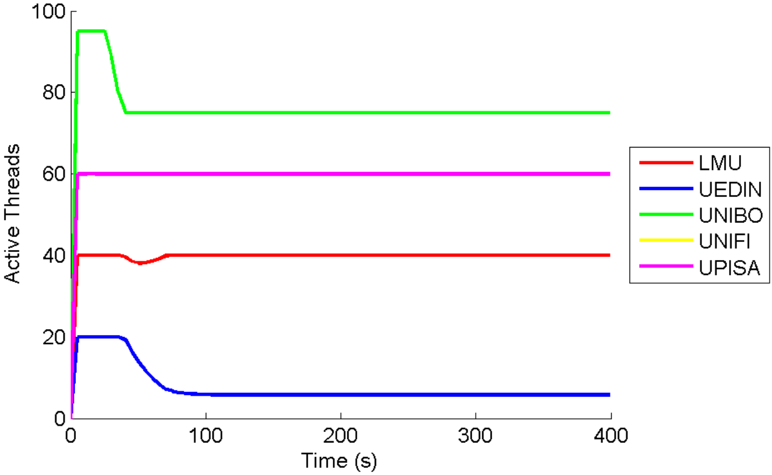
Numerical Results

$$r_{idle} = 0.03$$



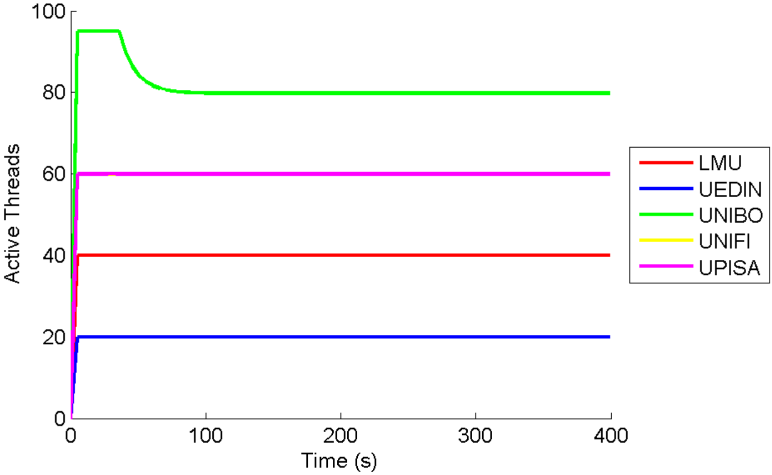
Numerical Results

$$r_{idle} = 0.04$$



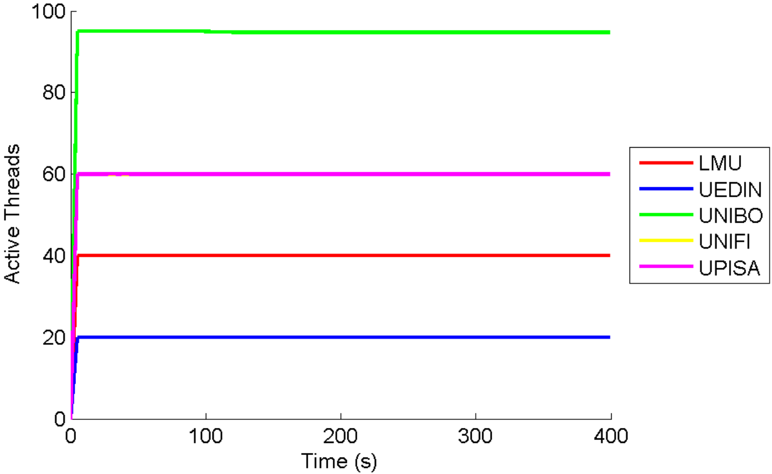
Numerical Results

$$r_{idle} = 0.05$$



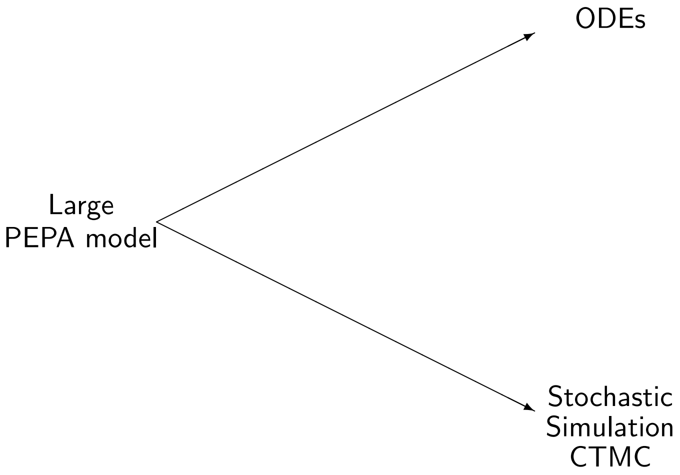
Numerical Results

$r_{idle} = 0.06$



Outline

Motivation: Alternative Representations



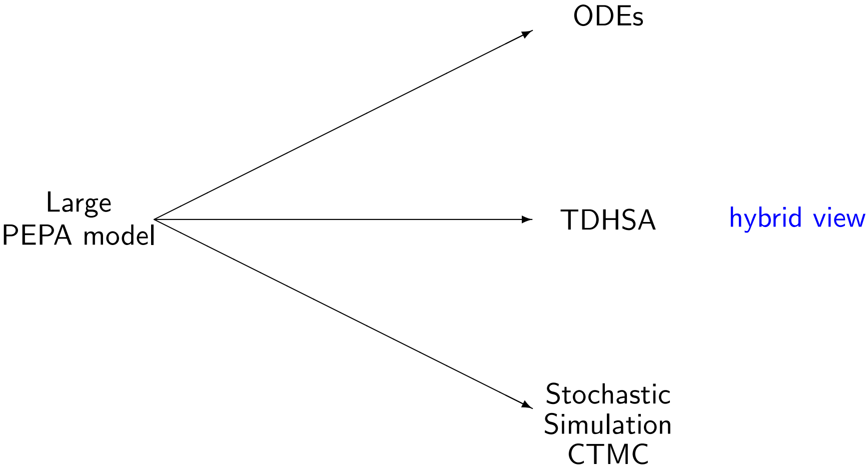
Motivation: Alternative Representations

Large PEPA model

ODEs — population view

Stochastic Simulation CTMC — individual view

Motivation: Alternative Representations



Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$

- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$

- mapping

$$
\begin{tikzcd}
P \arrow[d] & \stackrel{def}{=} & S_1 \arrow[d] & \underset{L_2}{\bowtie} & S_2 \arrow[d] & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \arrow[d] \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{tikzcd}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$\begin{tikzcd}
P \arrow[d] & \stackrel{def}{=} & S_1 \arrow[d] & \underset{L_2}{\bowtie} & S_2 \arrow[d] & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \arrow[d] \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{tikzcd}$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \stackrel{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \stackrel{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \stackrel{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \stackrel{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \stackrel{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \bigoplus_{L_2} & \mathcal{T}_2 & \bigoplus_{L_3} & \cdots & \bigoplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{tikzcd}
P \arrow[d] & \stackrel{\mathit{def}}{=} & S_1 \arrow[d] & \underset{L_2}{\bowtie} & S_2 \arrow[d] & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \arrow[d] \\
\mathcal{T} & = & \mathcal{T}_1 & \oplus_{L_2} & \mathcal{T}_2 & \oplus_{L_3} & \cdots & \oplus_{L_n} & \mathcal{T}_n
\end{tikzcd}
$$

Overview

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$
\begin{array}{ccccccccc}
P & \overset{def}{=} & S_1 & \underset{L_2}{\bowtie} & S_2 & \underset{L_3}{\bowtie} & \cdots & \underset{L_n}{\bowtie} & S_n \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow \\
\mathcal{T} & = & \mathcal{T}_1 & \bigoplus_{L_2} & \mathcal{T}_2 & \bigoplus_{L_3} & \cdots & \bigoplus_{L_n} & \mathcal{T}_n
\end{array}
$$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$
- multiset of continuous transitions:
  $(q, (z_1, \ldots, z_n), f, a)$ where $f : \mathbb{R}^n \to \mathbb{R}$
- multiset of stochastic transitions
  $(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), h, a)$ where $h : \mathbb{R}^n \to \mathbb{R}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$
- multiset of continuous transitions:
  $(q, (z_1, \ldots, z_n), f, a)$ where $f : \mathbb{R}^n \to \mathbb{R}$
- multiset of stochastic transitions
  $(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), h, a)$ where $h : \mathbb{R}^n \to \mathbb{R}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$
- multiset of continuous transitions:
  $(q, (z_1, \ldots, z_n), f, a)$ where $f : \mathbb{R}^n \to \mathbb{R}$
- multiset of stochastic transitions
  $(q_s, q_t, true, \bigwedge(X_k' = \rho_k(\mathbf{X})), h, a)$ where $h : \mathbb{R}^n \to \mathbb{R}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$
- multiset of continuous transitions:
  $(q, (z_1, \ldots, z_n), f, a)$ where $f : \mathbb{R}^n \to \mathbb{R}$
- multiset of stochastic transitions
  $(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), h, a)$ where $h : \mathbb{R}^n \to \mathbb{R}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$
- multiset of continuous transitions:
  $(q, (z_1, \ldots, z_n), f, a)$ where $f : \mathbb{R}^n \to \mathbb{R}$
- multiset of stochastic transitions
  $(q_s, q_t, \mathit{true}, \bigwedge(X'_k = \rho_k(\mathbf{X})), h, a)$ where $h : \mathbb{R}^n \to \mathbb{R}$

TDSHA behaviour

- continuous trace with stochastic jumps
- continuous behaviour in mode q described by ODEs

$$d\mathbf{X}/dt = \sum\{(z_1,\ldots,z_n)f(\mathbf{X}) \mid (q,(z_1,\ldots,z_n),f,a)\}$$

- stochastic transition from mode q_s and q_t with resets

$$(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), g, a)$$

happens with rate

$$\lambda(q,\mathbf{X}) = \sum\{h(\mathbf{X}) \mid (q_s, q_t, true, R, h, a)\}$$

and probability $g(\mathbf{X})/\lambda(q,\mathbf{X})$

TDSHA behaviour

- continuous trace with stochastic jumps
- continuous behaviour in mode q described by ODEs

$$d\mathbf{X}/dt = \sum\{(z_1, \ldots, z_n)f(\mathbf{X}) \mid (q, (z_1, \ldots, z_n), f, a)\}$$

- stochastic transition from mode q_s and q_t with resets

$$(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), g, a)$$

happens with rate

$$\lambda(q, \mathbf{X}) = \sum\{h(\mathbf{X}) \mid (q_s, q_t, true, R, h, a)\}$$

and probability $g(\mathbf{X})/\lambda(q, \mathbf{X})$

TDSHA behaviour

- continuous trace with stochastic jumps
- continuous behaviour in mode q described by ODEs

$$d\mathbf{X}/dt = \sum\{(z_1, \ldots, z_n)f(\mathbf{X}) \mid (q, (z_1, \ldots, z_n), f, a)\}$$

- stochastic transition from mode q_s and q_t with resets

$$(q_s, q_t, true, \bigwedge(X'_k = \rho_k(\mathbf{X})), g, a)$$

happens with rate

$$\lambda(q, \mathbf{X}) = \sum\{h(\mathbf{X}) \mid (q_s, q_t, true, R, h, a)\}$$

and probability $g(\mathbf{X})/\lambda(q, \mathbf{X})$

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$
- continuous transitions: extend vector to cover $\mathbf{X}$
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition from q_1 and q_2 with a and new function is PEPA cooperation rate (i.e. bounded capacity)
- stochastic transitions:
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition that both q_1 and q_2 have with a, new rate is PEPA cooperation rate and conjunction of resets is taken

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$
- continuous transitions: extend vector to cover $\mathbf{X}$
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition from q_1 and q_2 with a and new function is PEPA cooperation rate (i.e. bounded capacity)
- stochastic transitions:
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition that both q_1 and q_2 have with a, new rate is PEPA cooperation rate and conjunction of resets is taken

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$
- continuous transitions: extend vector to cover $\mathbf{X}$
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition from q_1 and q_2 with a and new function is PEPA cooperation rate (i.e. bounded capacity)
- stochastic transitions:
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition that both q_1 and q_2 have with a, new rate is PEPA cooperation rate and conjunction of resets is taken

Clients and servers example

- clients

$$
\begin{aligned}
\mathrm{Cr} &\stackrel{\mathit{def}}{=} (\mathrm{request}, r_{rq}).\mathrm{Ct} \\
\mathrm{Ct} &\stackrel{\mathit{def}}{=} (\mathrm{think}, r_{th}).\mathrm{Cr}
\end{aligned}
$$

- servers

Clients and servers example

- clients

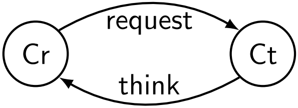
$$
\begin{aligned}
\mathrm{Cr} &\stackrel{\mathit{def}}{=} (\mathrm{request}, r_{rq}).\mathrm{Ct} \\
\mathrm{Ct} &\stackrel{\mathit{def}}{=} (\mathrm{think}, r_{th}).\mathrm{Cr}
\end{aligned}
$$

- servers

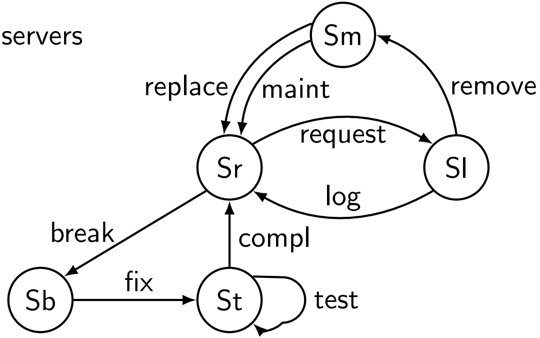
$$
\begin{aligned}
\mathrm{Sr} &\stackrel{\mathit{def}}{=} (\mathrm{request}, r_{rp}).\mathrm{Sl} + (\mathrm{break}, r_{bk}).\mathrm{Sb} \\
\mathrm{Sl} &\stackrel{\mathit{def}}{=} (\mathrm{log}, r_{lg}).\mathrm{Sr} + (\mathrm{remove}, r_{rm}).\mathrm{Sm} \\
\mathrm{Sm} &\stackrel{\mathit{def}}{=} (\mathrm{maint}, r_{mn}).\mathrm{Sr} + (\mathrm{replace}, r_{rc}).\mathrm{Sr} \\
\mathrm{Sb} &\stackrel{\mathit{def}}{=} (\mathrm{fix}, r_{fx}).\mathrm{St} \\
\mathrm{St} &\stackrel{\mathit{def}}{=} (\mathrm{test}, r_{ts}).\mathrm{St} + (\mathrm{compl}, r_{cm}).\mathrm{Sr}
\end{aligned}
$$

Clients and servers example

- clients



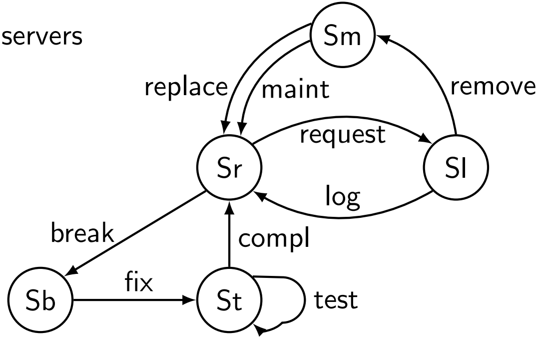
- servers



Clients and servers example

- clients

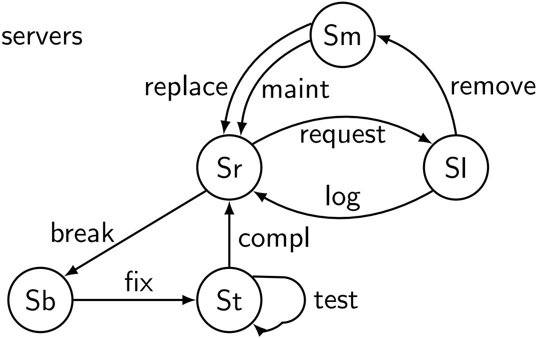
- servers



Clients and servers example

- clients

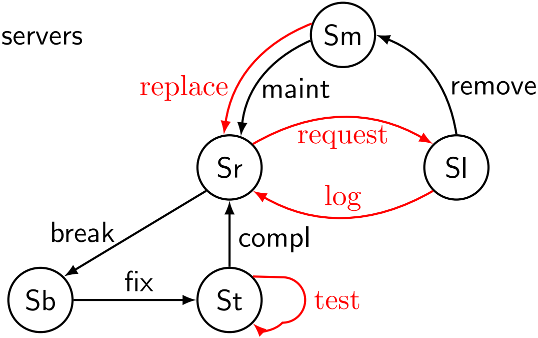
- servers



Clients and servers example

- clients

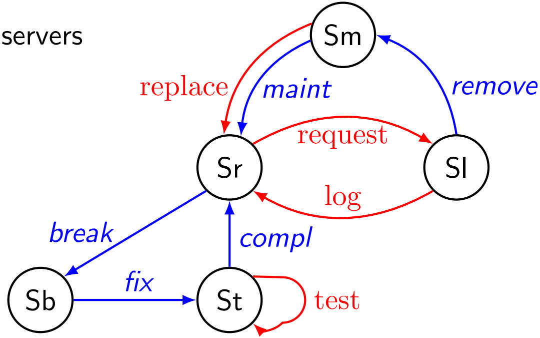
- servers



Clients and servers example

- clients

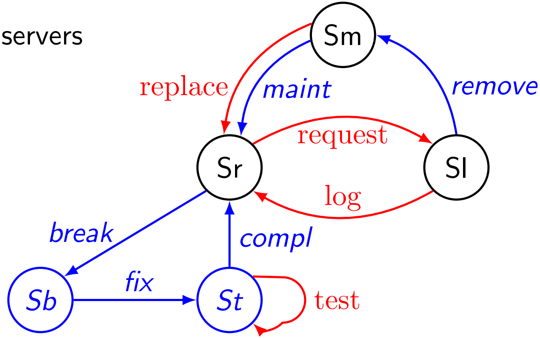
- servers



Clients and servers example

- clients

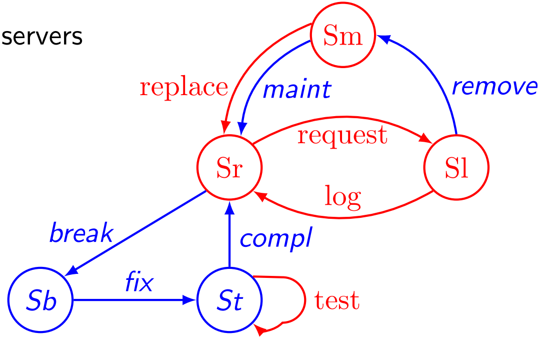
- servers



Clients and servers example

- clients

- servers



Clients and servers example

- clients

$$
\begin{aligned}
\mathrm{Cr} &\stackrel{def}{=} (\mathrm{request}, r_{rq}).\mathrm{Ct} \\
\mathrm{Ct} &\stackrel{def}{=} (\mathrm{think}, r_{th}).\mathrm{Cr}
\end{aligned}
$$

- servers

$$
\begin{aligned}
\mathrm{Sr} &\stackrel{def}{=} (\mathrm{request}, r_{rp}).\mathrm{Sl} + (\mathit{break}, r_{bk}).\mathit{Sb} \\
\mathrm{Sl} &\stackrel{def}{=} (\mathrm{log}, r_{lg}).\mathrm{Sr} + (\mathit{remove}, r_{rm}).\mathrm{Sm} \\
\mathrm{Sm} &\stackrel{def}{=} (\mathit{maint}, r_{mn}).\mathrm{Sr} + (\mathrm{replace}, r_{rc}).\mathsf{Sr} \\
\mathit{Sb} &\stackrel{def}{=} (\mathit{fix}, r_{fx}).\mathit{St} \\
\mathit{St} &\stackrel{def}{=} (\mathrm{test}, r_{ts}).\mathit{St} + (\mathit{compl}, r_{cm}).\mathrm{Sr}
\end{aligned}
$$

Mapping to TDSHA

- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: Sb, St
- population vector: $(\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{Sm}, \#Sb, \#St)$
- PEPA is conservative: both $N_C = \#\mathrm{Cr} + \#\mathrm{Ct}$ and $N_S = \#\mathrm{Sr} + \#\mathrm{Sl} + \#\mathrm{Sm} + \#Sb + \#St$ are invariant
- TDSHA
  - modes: $(\#Sb, \#St) \in \{0, \ldots, N_S\} \times \{0, \ldots, N_S\}$
  - variables: $(X_{\mathrm{Cr}}, X_{\mathrm{Ct}}, X_{\mathrm{Sr}}, X_{\mathrm{Sl}}, X_{\mathrm{Sm}})$
  - initial state: $((\#Sb, \#St), (\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{St}))$
  - continuous and stochastic transitions

Mapping to TDSHA

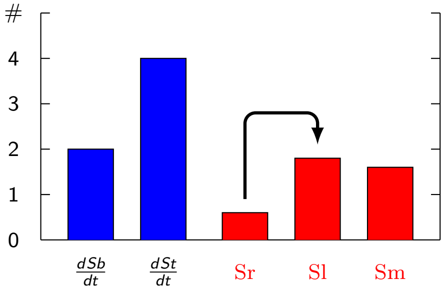
- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: Sb, St
- population vector: $(\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{Sm}, \#Sb, \#St)$
- PEPA is conservative: both $N_C = \#\mathrm{Cr} + \#\mathrm{Ct}$ and $N_S = \#\mathrm{Sr} + \#\mathrm{Sl} + \#\mathrm{Sm} + \#Sb + \#St$ are invariant
- TDSHA
  - modes: $(\#Sb, \#St) \in \{0, \ldots, N_S\} \times \{0, \ldots, N_S\}$
  - variables: $(X_{\mathrm{Cr}}, X_{\mathrm{Ct}}, X_{\mathrm{Sr}}, X_{\mathrm{Sl}}, X_{\mathrm{Sm}})$
  - initial state: $((\#Sb, \#St), (\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{St}))$
  - continuous and stochastic transitions

Mapping to TDSHA

- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: Sb, St
- population vector: $(\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{Sm}, \#Sb, \#St)$
- PEPA is conservative: both $N_C = \#\mathrm{Cr} + \#\mathrm{Ct}$ and $N_S = \#\mathrm{Sr} + \#\mathrm{Sl} + \#\mathrm{Sm} + \#Sb + \#St$ are invariant
- TDSHA
  - modes: $(\#Sb, \#St) \in \{0, \ldots, N_S\} \times \{0, \ldots, N_S\}$
  - variables: $(X_{\mathrm{Cr}}, X_{\mathrm{Ct}}, X_{\mathrm{Sr}}, X_{\mathrm{Sl}}, X_{\mathrm{Sm}})$
  - initial state: $((\#Sb, \#St), (\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{St}))$
  - continuous and stochastic transitions

Continuous transitions between continuous components

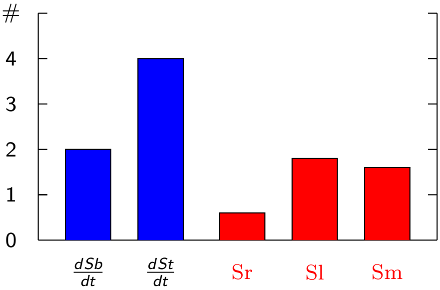
- $\mathrm{Sr} \xrightarrow{(\mathrm{request}, r_{rp} \cdot \#\mathrm{Sr})}_{\star} \mathrm{Sl}$
- continuous transition: flow is determined by ODEs



- $((\#Sb, \#St), (0, 0, -1, 1, 0), r_{rp} \cdot \#\mathrm{Sr}, \mathrm{request})$

Continuous transition at a discrete component

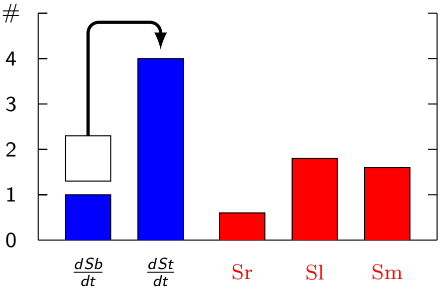
- $St \xrightarrow{(\text{test}, r_{ts} \cdot \#St)}_{\star} St$
- continuous transition: no flow because single component



- $((\#Sb, \#St), (0, 0, 0, 0, 0), r_{ts} \cdot \#St, \text{request})$

Discrete transitions between discrete components

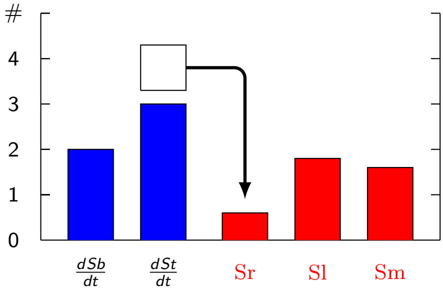
- $Sb \xrightarrow{(fix, r_{fx} \cdot \#Sb)}_{\star} St$
- stochastic transition: unit quantity is shifted



- $((\#Sb, \#St), (\#Sb - 1, \#St + 1), true, true, r_{fx} \cdot \#Sb, fix)$

Discrete transition from discrete to continuous component

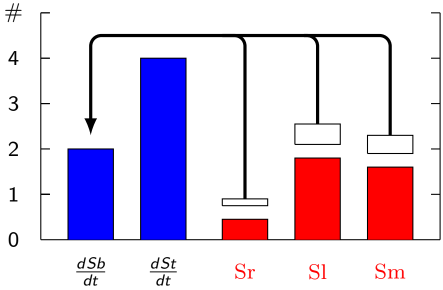
- $St \xrightarrow{(compl, r_{cm} \cdot \# St)}_{\star} \mathrm{Sr}$
- stochastic transition: unit quantity is shifted



- $((\# Sb, \# St), (\# Sb, \# St - 1), true, R, r_{cm} \cdot \# St, compl)$ with $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} + 1)$

Discrete transition from continuous to discrete component

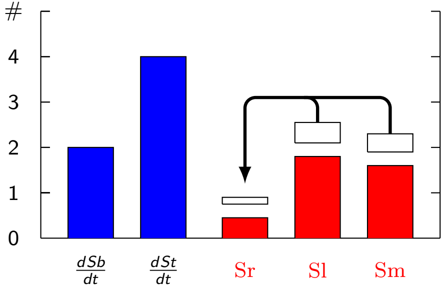
- $\mathrm{Sr} \xrightarrow{(\mathit{break}, r_{bk} \cdot \#\mathrm{Sr})}_{\star} \mathit{Sb}$
- stochastic transition: unit quantity is shifted proportionally



- $((\#Sb, \#St), (\#Sb + 1, \#St), true, R, r_{bk} \cdot \#\mathrm{Sr}, break)$ with $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} - z_r) \wedge (X'_{\mathrm{Sl}} = X_{\mathrm{Sl}} - z_l) \wedge (X'_{\mathrm{Sm}} = X_{\mathrm{Sm}} - z_m)$ and $z_r + z_l + z_m = 1$

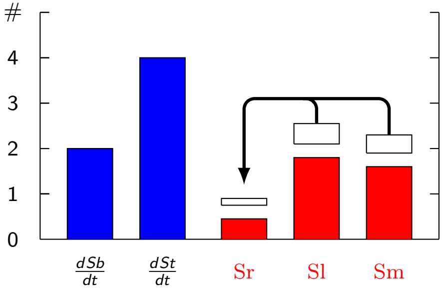
Discrete transition between continuous components

- $\mathrm{Sm} \xrightarrow{(\mathit{maint}, r_{mn} \cdot \#\mathrm{Sm})}_{\star} \mathrm{Sr}$
- stochastic transition: unit quantity is shifted proportionally

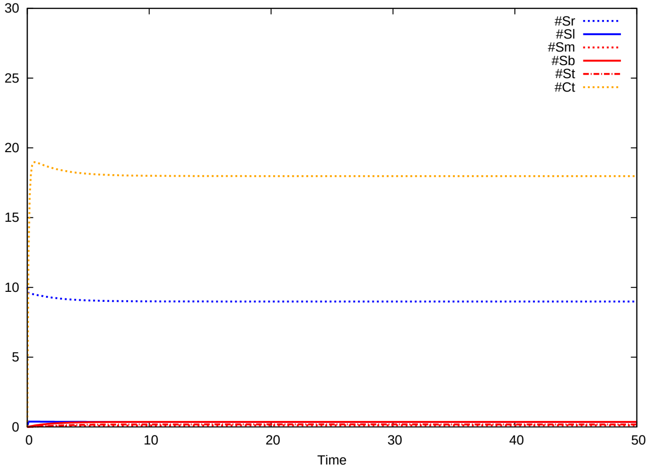


Discrete transition between continuous components

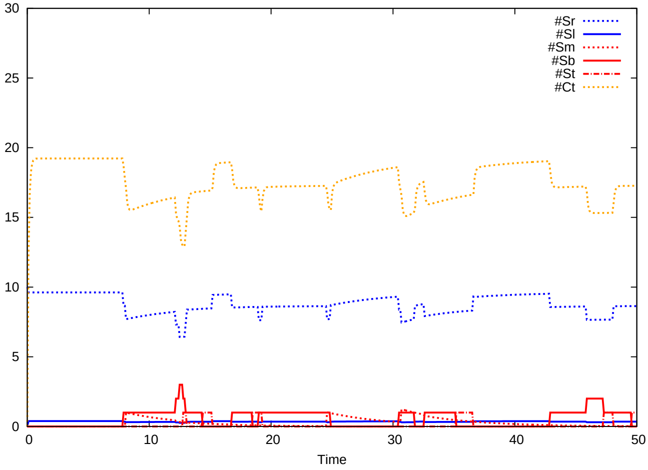
- $((\#Sb, \#St), (\#Sb, \#St), true, R, r_{mn} \cdot \#\mathrm{Sm}, maint)$ where $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} - z_r + 1) \wedge (X'_{\mathrm{Sl}} = X_{\mathrm{Sl}} - z_l) \wedge (X'_{\mathrm{Sm}} = X_{\mathrm{Sm}} - z_m)$ and $z_r + z_l + z_m = 1$



Continuous determinstic simulation



Hybrid simulation



References

- J. Hillston, *Fluid Flow Approximations of PEPA Models*, in Proc. of Intl. Conference on Quantitative Evaluation of Systems (QEST) 2005, Computer Society Press, pp. 33–42, 2005.
- J.T. Bradley, S.T. Gilmore and J. Hillston, *Analysing distributed Internet worm attacks using continuous state-space approximation of process algebra models*, in Journal of Computer and System Sciences, 74(6), pp. 1013–1032, 2008.
- M. Tribastone, S. Gilmore and J. Hillston, *Scalable Differential Analysis of Process Algebra Models*, in IEEE Transactions on Software Engineering, 38(1), pp. 205–219, 2012.
- L. Bortolussi, V. Galpin, J. Hillston and M. Tribastone, *Hybrid semantics for PEPA*, in QEST 2010, Williamsburg, USA, Computer Society Press, pp. 181–190, 2010.