Process Algebra for Collective Dynamics

Jane Hillston

Laboratory for Foundations of Computer Science
University of Edinburgh

joint work with Stephen Gilmore and Mirco Tribastone

Outline

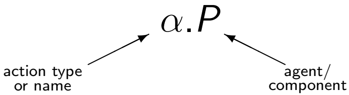
1. Introduction
   - Stochastic Process Algebra
   - Collective Dynamics
2. Continuous Approximation
   - State variables
   - Numerical illustration
3. Fluid-Flow Semantics
   - Fluid Structured Operational Semantics
4. Example
   - Scalable Web Services
5. Conclusions
   - Alternative Models

Outline

1. Introduction
   - Stochastic Process Algebra
   - Collective Dynamics
2. Continuous Approximation
   - State variables
   - Numerical illustration
3. Fluid-Flow Semantics
   - Fluid Structured Operational Semantics
4. Example
   - Scalable Web Services
5. Conclusions
   - Alternative Models

Process Algebra

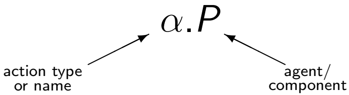
- Models consist of agents which engage in actions.



$\alpha.P$

Process Algebra

- Models consist of agents which engage in actions.

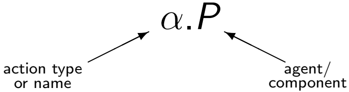


$\alpha.P$

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

Process Algebra

- Models consist of agents which engage in actions.

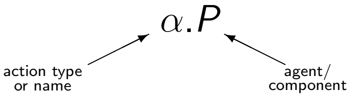


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

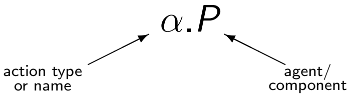
agent/ component

- 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.



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

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

A simple example: processors and resources

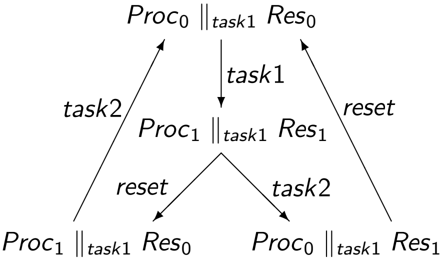
$$\begin{aligned}
Proc_0 &\overset{def}{=} task1.Proc_1 \\
Proc_1 &\overset{def}{=} task2.Proc_0 \\
Res_0 &\overset{def}{=} task1.Res_1 \\
Res_1 &\overset{def}{=} reset.Res_0
\end{aligned}$$

$$Proc_0 \parallel_{task1} Res_0$$

A simple example: processors and resources

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

$$Proc_0 \parallel_{task1} Res_0$$



$Proc_0 \parallel_{task1} Res_0$

$task1$

$task2$

$reset$

$Proc_1 \parallel_{task1} Res_1$

$reset$

$task2$

$Proc_1 \parallel_{task1} Res_0$

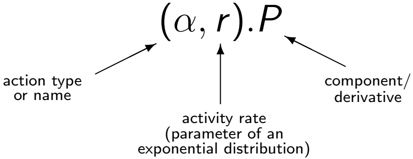
$Proc_0 \parallel_{task1} Res_1$

Stochastic process algebras

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

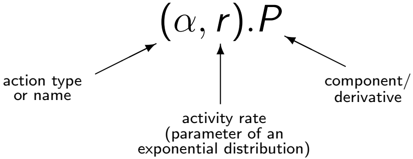
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type or name

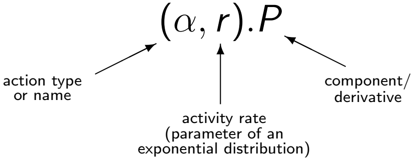
activity rate (parameter of an exponential distribution)

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.



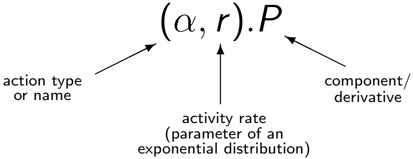
$$(\alpha, r).P$$

- 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)

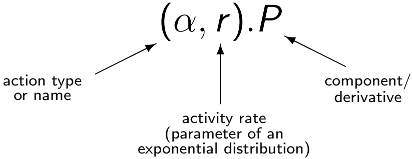
component/ derivative

- The language is used to generate a CTMC for performance modelling.

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



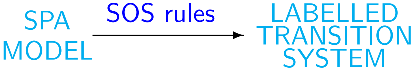
$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC for performance modelling.



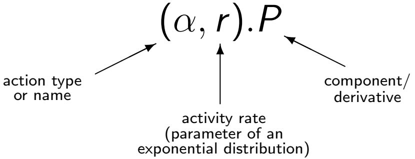
SPA MODEL

SOS rules

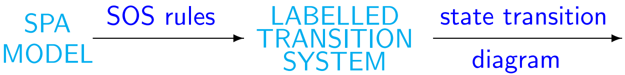
LABELLED TRANSITION SYSTEM

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

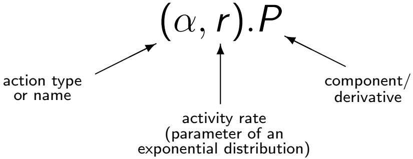


- The language is used to generate a CTMC for performance modelling.



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



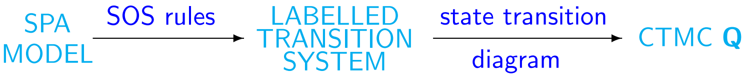
$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

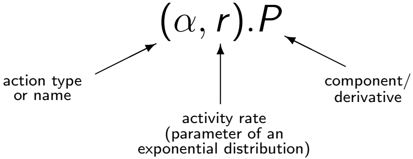
- The language is used to generate a CTMC for performance modelling.



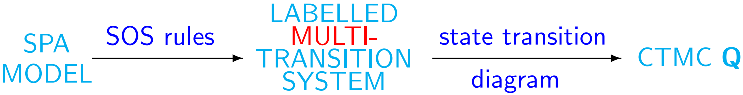
$$\text{SPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}} \text{CTMC } \mathbf{Q}$$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



- The language is used to generate a CTMC for performance modelling.



Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

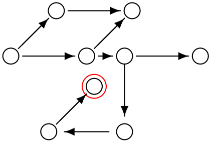
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

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

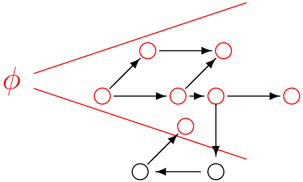


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

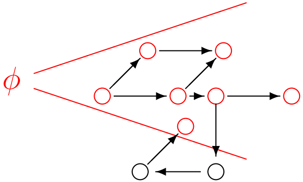


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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



Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
X	Variable

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

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

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

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

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

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 $P_1 \underset{\emptyset}{\bowtie} P_2$.

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.

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.

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

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions.

In PEPA the cooperation of components is assumed to give rise to shared actions and the rates of these shared actions are governed by the assumption of bounded capacity.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions.

In PEPA the cooperation of components is assumed to give rise to shared actions and the rates of these shared actions are governed by the assumption of bounded capacity.

The principle of bounded capacity means that a component cannot be made to carry out an action in cooperation faster than its own defined rate for the action. Thus shared actions proceed at the minimum of the rates in the participating components.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions.

In PEPA the cooperation of components is assumed to give rise to shared actions and the rates of these shared actions are governed by the assumption of bounded capacity.

The principle of bounded capacity means that a component cannot be made to carry out an action in cooperation faster than its own defined rate for the action. Thus shared actions proceed at the minimum of the rates in the participating components.

In contrast independent actions do not constrain each other and if there are multiple copies of a action enabled in independent concurrent components their rates are summed.

A simple example: processors and resources

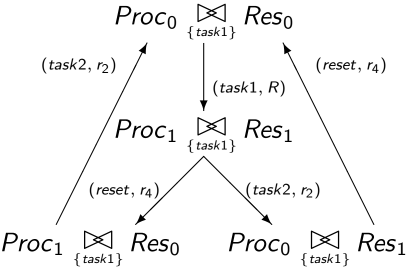
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

A simple example: processors and resources

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

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

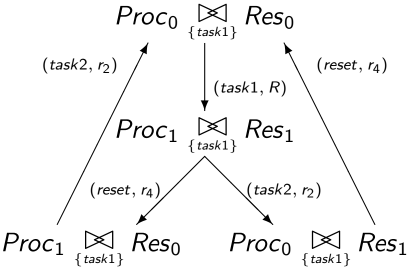

$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$
$(task2, r_2)$
$(task1, R)$
$(reset, r_4)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_1$
$(reset, r_4)$
$(task2, r_2)$
$Proc_1 \underset{\{task1\}}{\bowtie} Res_0$
$Proc_0 \underset{\{task1\}}{\bowtie} Res_1$

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

A simple example: processors and resources

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

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



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

$(task2, r_2)$

$(task1, R)$

$(reset, r_4)$

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

$(reset, r_4)$

$(task2, r_2)$

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

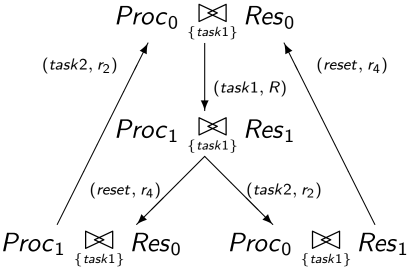
$Proc_0 \underset{\{task1\}}{\bowtie} Res_1$

$R = \min(r_1, r_3)$

A simple example: processors and resources

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

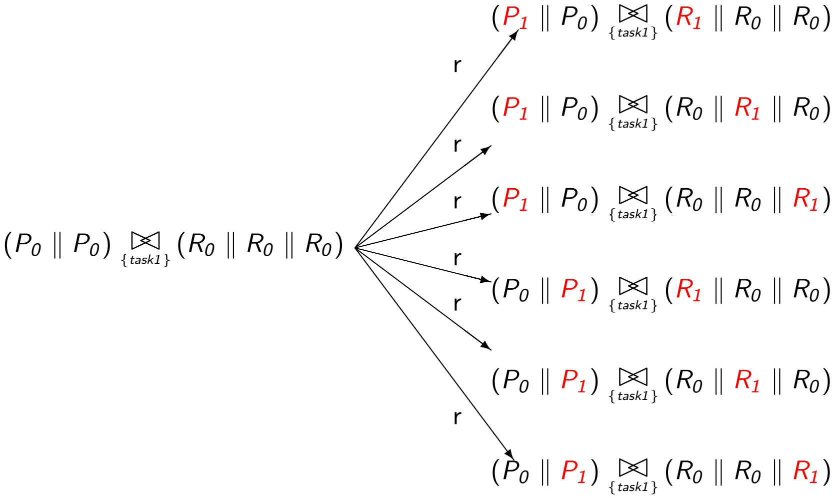
$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$


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

$R = \min(r_1, r_3)$

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

Calculating the rate of actions carried out in cooperation



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

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

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

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

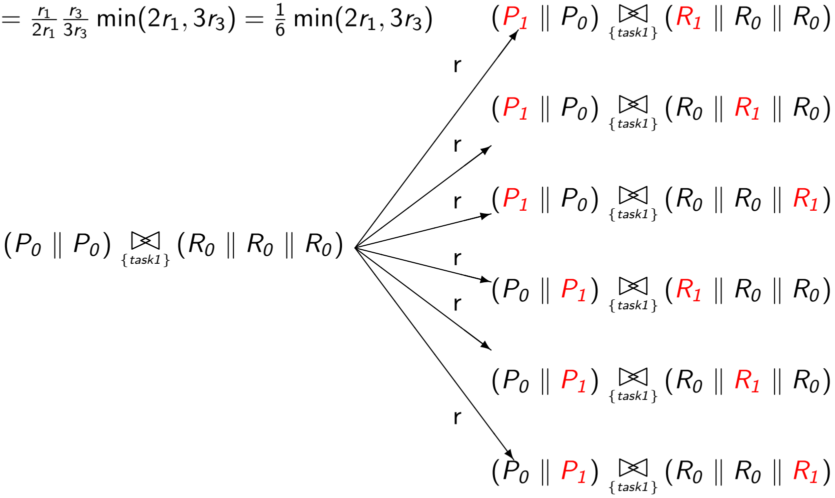
$\xrightarrow{r} (P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0)$

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

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

Calculating the rate of actions carried out in cooperation

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



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

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

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

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

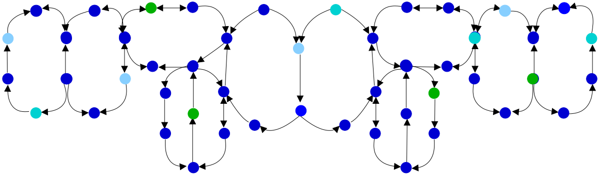
$(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0)$

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

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

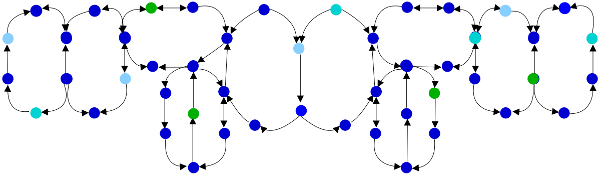
Collective Dynamics

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

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;

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;

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;

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.

Population statistics: 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.

Population statistics: 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.

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.

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 in this approach is twofold:

Novelty

The novelty in this approach is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.

Novelty

The novelty in this approach is 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 in this approach is 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 in this approach is 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 in this approach is 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.

Novelty

The novelty in this approach is 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:

Crowd dynamics

Technology enhancement is creating new possibilities for directing crowd movements in buildings and urban spaces, for example for emergency egress, which are not yet well-understood.

Outline

1 Introduction

- Stochastic Process Algebra
- Collective Dynamics

2 Continuous Approximation

- State variables
- Numerical illustration

3 Fluid-Flow Semantics

- Fluid Structured Operational Semantics

4 Example

- Scalable Web Services

5 Conclusions

- Alternative Models

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.

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.

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.

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.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.

.

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.

.

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.
- Instead the ODEs estimate the expected behaviour of the CTMC.

.

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.
- 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.

Simple example revisited

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

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

Simple example revisited

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

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

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

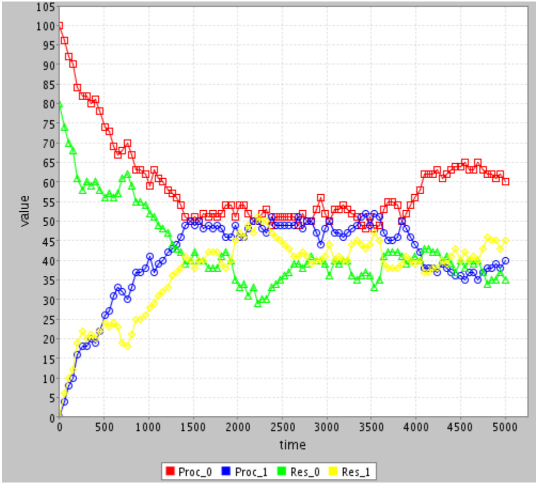
$$
\begin{aligned}
Proc_0 &\stackrel{\mathit{def}}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{\mathit{def}}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{\mathit{def}}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{\mathit{def}}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

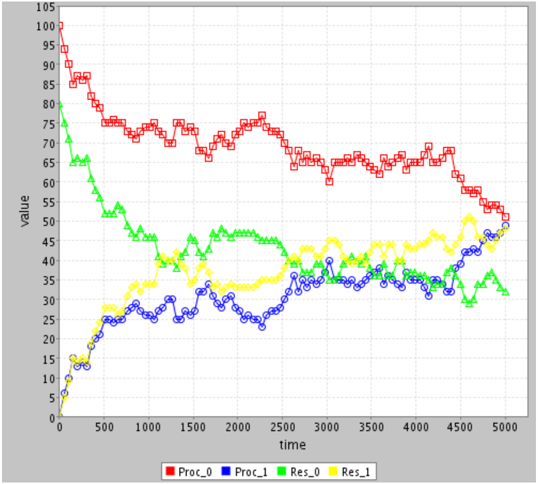
ODE interpretation

$$
\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= -\min(r_1 x_1, r_3 x_3) + r_2\, x_1 \\
& \quad x_1 = \text{no. of } Proc_0 \\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= r_1 \min(x_1, x_3) - r_2\, x_1 \\
& \quad x_2 = \text{no. of } Proc_1 \\
\frac{\mathrm{d}x_3}{\mathrm{d}t} &= -r_1 \min(x_1, x_3) + r_4\, x_4 \\
& \quad x_3 = \text{no. of } Res_0 \\
\frac{\mathrm{d}x_4}{\mathrm{d}t} &= r_1 \min(x_1, x_3) - r_4\, x_4 \\
& \quad x_4 = \text{no. of } Res_1
\end{aligned}
$$

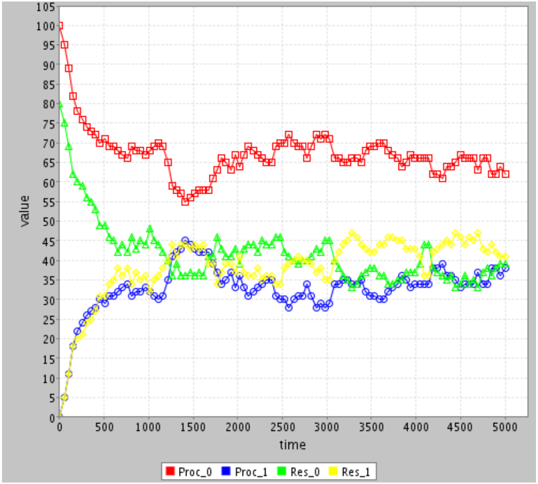
100 processors and 80 resources (simulation run A)



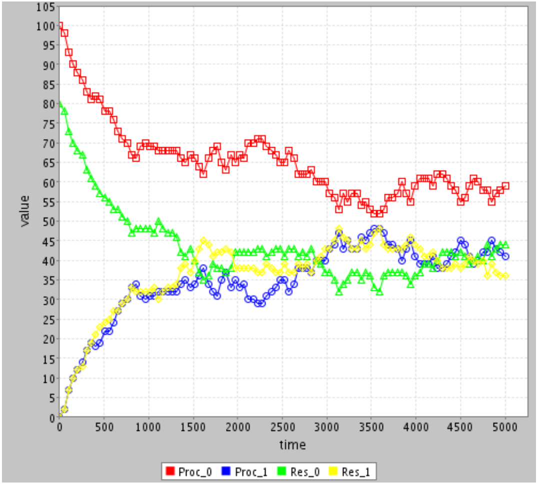
100 processors and 80 resources (simulation run B)



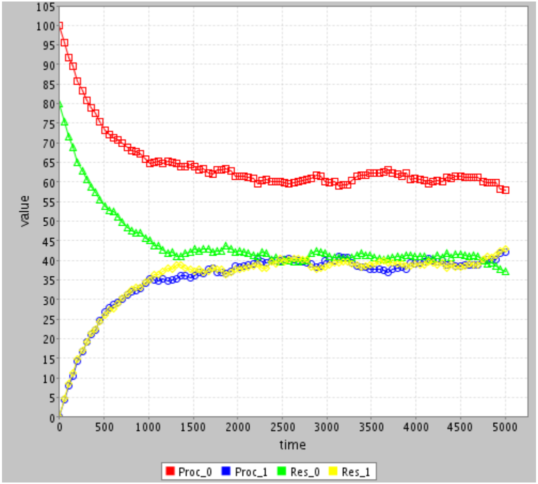
100 processors and 80 resources (simulation run C)



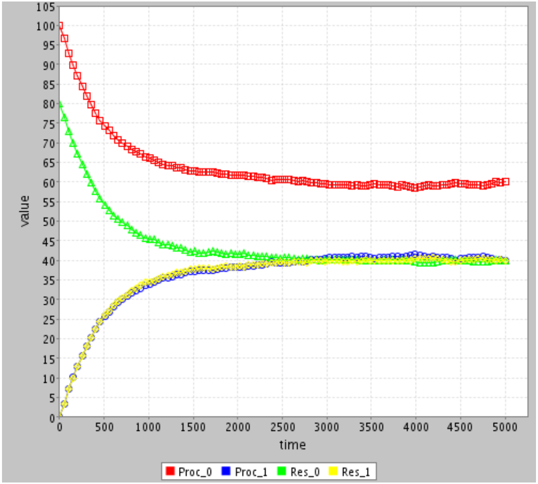
100 processors and 80 resources (simulation run D)



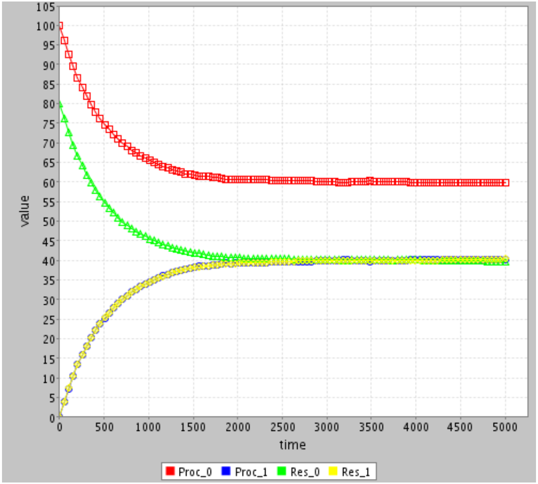
100 processors and 80 resources (average of 10 runs)



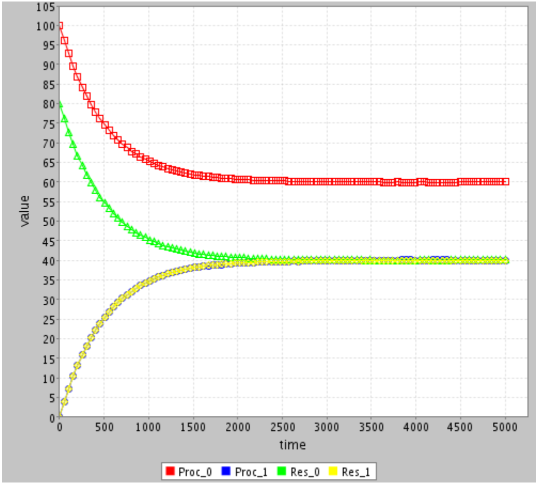
100 Processors and 80 resources (average of 100 runs)



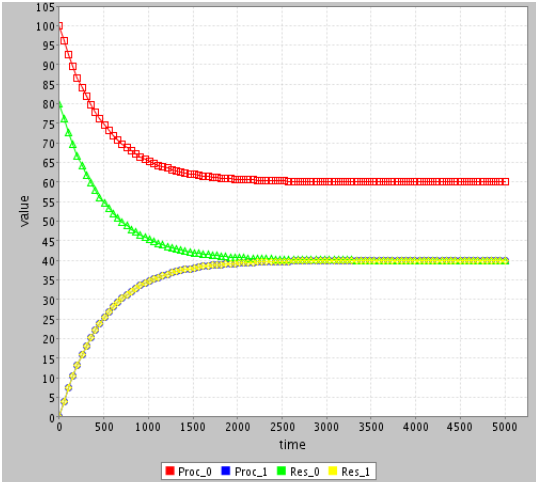
100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (average of 10000 runs)



100 processors and 80 resources (ODE solution)



Outline

1 Introduction
- Stochastic Process Algebra
- Collective Dynamics

2 Continuous Approximation
- State variables
- Numerical illustration

3 Fluid-Flow Semantics
- Fluid Structured Operational Semantics

4 Example
- Scalable Web Services

5 Conclusions
- Alternative Models

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.

Deriving a Fluid Approximation of a PEPA model

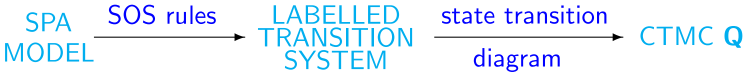
The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

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

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

SOS rules

LABELLED TRANSITION SYSTEM

state transition diagram

CTMC **Q**

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.

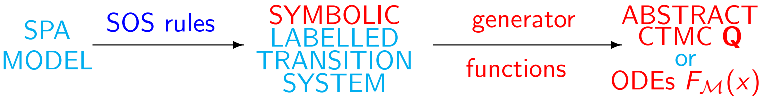
Nevertheless we are able to define a structured operational semantics which defines the possible transitions of an abitrary abstract state and from this derive the ODEs.

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.

Nevertheless we are able to define a structured operational semantics which defines the possible transitions of an abitrary abstract state and from this derive the ODEs.



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

generator functions

ABSTRACT CTMC **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:

Fluid Structured Operational Semantics

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

1. Remove excess components (*Context Reduction*)

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*)

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.

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.

Context Reduction

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

Location Dependency

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

$$\Downarrow$$

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

Location Dependency

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

$$\frac{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1} \mathit{Proc_1}}{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1 \xi_1}_{*} \mathit{Proc_1}}$$

Fluid Structured Operational Semantics by Example

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

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

$$\dfrac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{}$$

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

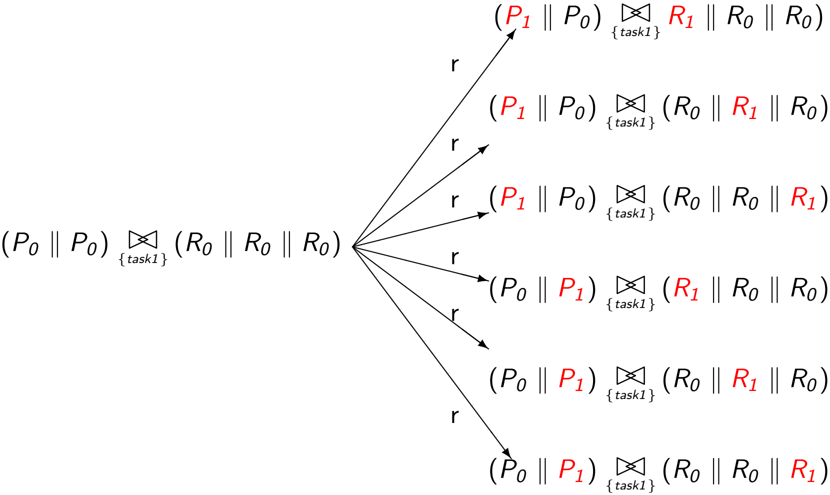
$$\frac{\dfrac{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r'_{\mathit{1}}} \mathit{Proc_1}}{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r'_{\mathit{1}}\xi_1}_{*} \mathit{Proc_1}} \qquad \dfrac{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_{\mathit{3}}} \mathit{Res_1}}{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_{\mathit{3}}\xi_3}_{*} \mathit{Res_1}}}{\mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} \xrightarrow{\mathit{task1}, r(\xi)}_{*} \mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_1}}$$

Apparent Rate Calculation

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

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

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



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

r

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

r

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

r

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

r

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

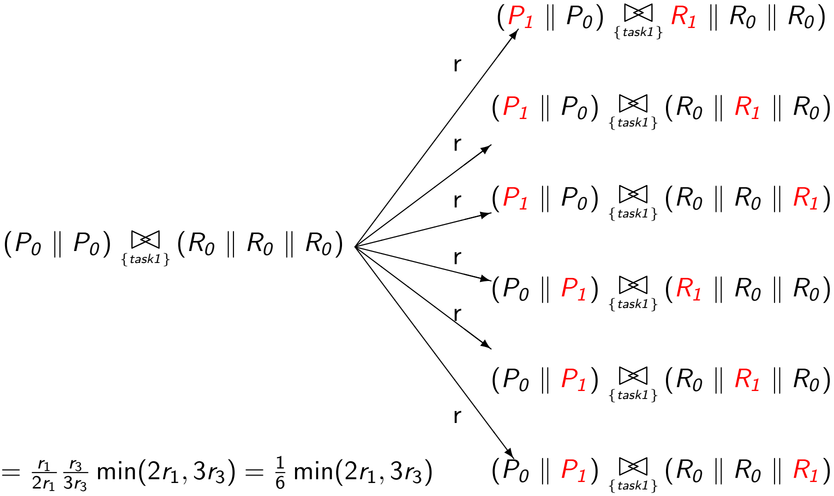
r

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

r

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

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



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

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

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

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

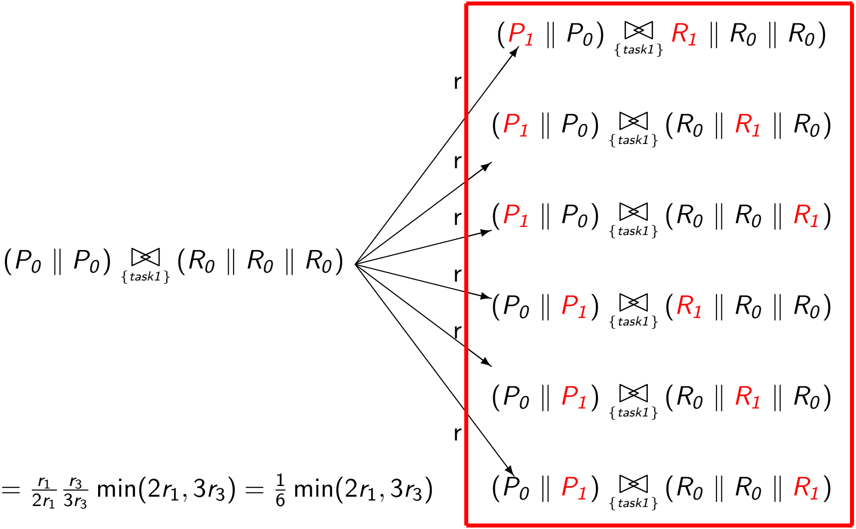
$(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_1 \parallel R_0 \parallel R_0)$

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

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

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

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



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

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

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

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

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

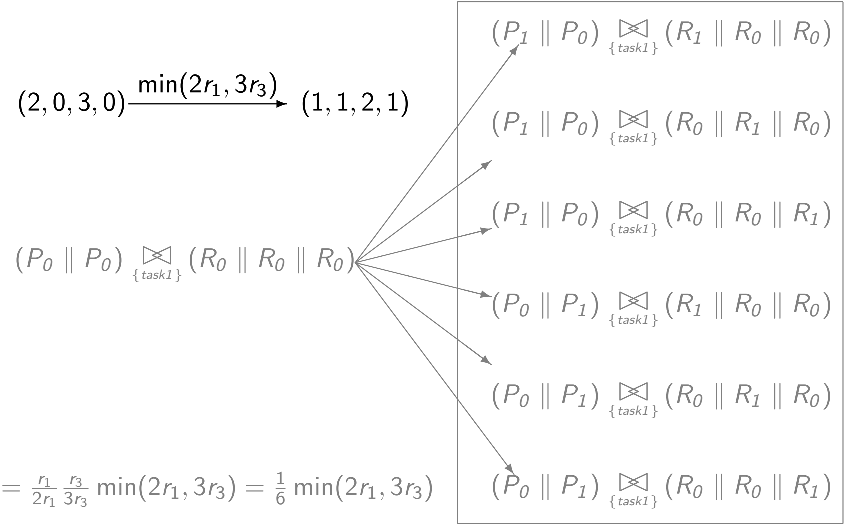
$(P_0 \parallel P_1) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1 \parallel R_0)$

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

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

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

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

$$r(\xi) = \min\left(r_{\mathit{1}}\xi_1, r_{\mathit{3}}\xi_3\right)$$

Jump Multiset

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

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

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

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:

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

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

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

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

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

- Take $I = (0, 0, 0, 0)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{array}{rcl}
\mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} & \xrightarrow{\mathit{task1}, r(\xi)}_{*} & \mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_1} \\
\mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} & \xrightarrow{\mathit{task2}, \xi_2 r'_2}_{*} & \mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0}
\end{array}
$$

$$
\begin{array}{rcl}
f(\xi, (-1, +1, -1, +1), \mathit{task1}) & = & r(\xi) \\
f(\xi, (+1, -1, 0, 0), \mathit{task2}) & = & \xi_2 r_2
\end{array}
$$

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

$$
\begin{array}{ccc}
\mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} & \xrightarrow{\mathit{task1}, r(\xi)}_{*} & \mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_1} \\
\mathit{Proc_1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} & \xrightarrow{\mathit{task2}, \xi_2 r'_2}_{*} & \mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0} \\
\mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_1} & \xrightarrow{\mathit{reset}, \xi_4 r_4}_{*} & \mathit{Proc_0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res_0}
\end{array}
$$

$$
\begin{array}{rcl}
f(\xi, (-1, +1, -1, +1), \mathit{task1}) & = & r(\xi) \\
f(\xi, (+1, -1, 0, 0), \mathit{task2}) & = & \xi_2 r_2 \\
f(\xi, (0, 0, +1, -1), \mathit{reset}) & = & \xi_4 r_4
\end{array}
$$

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

$$
\begin{array}{rcl}
\mathit{Proc}_0 & \stackrel{\mathit{def}}{=} & (\mathit{task1}, r_1).\mathit{Proc}_1 \\
\mathit{Proc}_1 & \stackrel{\mathit{def}}{=} & (\mathit{task2}, r_2).\mathit{Proc}_0 \\
\mathit{Res}_0 & \stackrel{\mathit{def}}{=} & (\mathit{task1}, r_3).\mathit{Res}_1 \\
\mathit{Res}_1 & \stackrel{\mathit{def}}{=} & (\mathit{reset}, r_4).\mathit{Res}_0 \\
\mathit{System} & \stackrel{\mathit{def}}{=} & \mathit{Proc}_0[N_P] \underset{\{\mathit{transfer}\}}{\bowtie} \mathit{Res}_0[N_R]
\end{array}
$$

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

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 Equation

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

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 Equation

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

Outline

1 Introduction
- Stochastic Process Algebra
- Collective Dynamics

2 Continuous Approximation
- State variables
- Numerical illustration

3 Fluid-Flow Semantics
- Fluid Structured Operational Semantics

4 Example
- Scalable Web Services

5 Conclusions
- Alternative Models

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.

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.

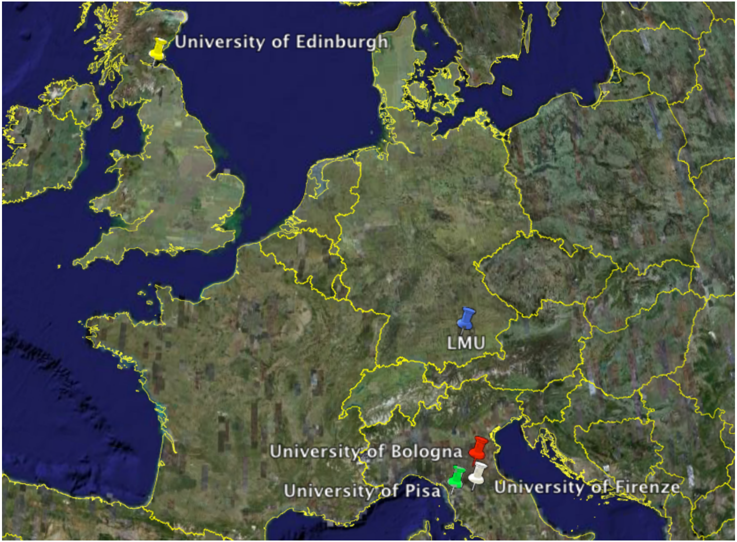
Virtual University Scenario

- 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.

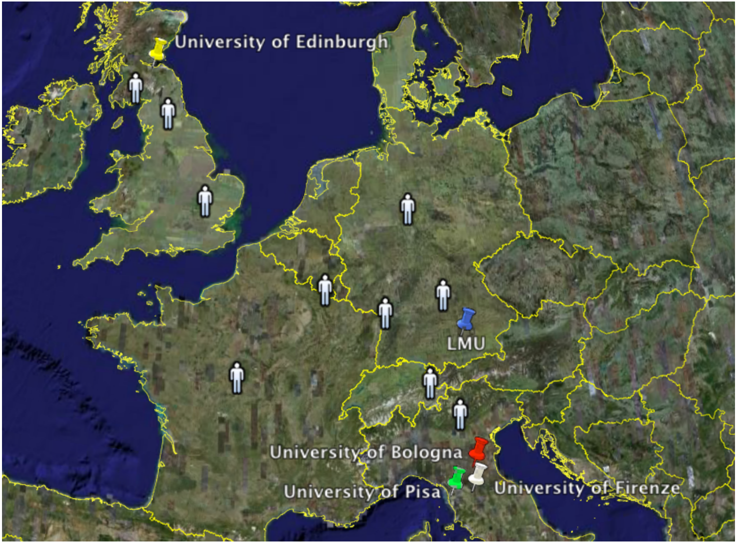
Virtual University Scenario

- 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



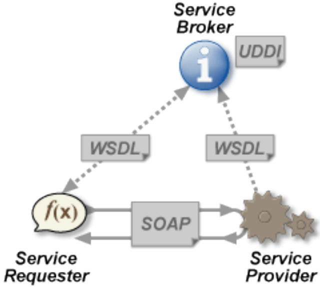
Replicating Web Services

Two viable approaches to cope with increasing user demand:

Replicating Web Services

Two viable approaches to cope with increasing user demand:

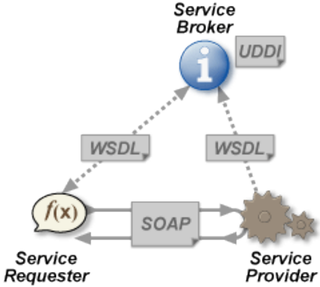
- use a service broker for routing



Replicating Web Services

Two viable approaches to cope with increasing user demand:

- use a service broker for routing

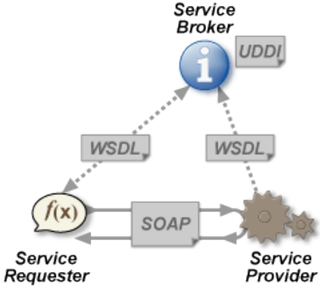


- decentralised routing

Replicating Web Services

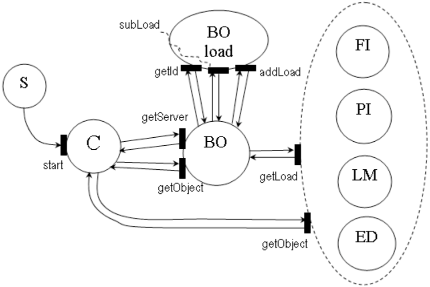
Two viable approaches to cope with increasing user demand:

- use a service broker for routing



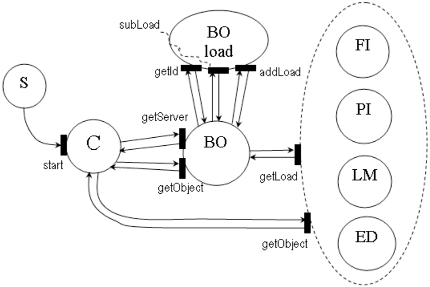
- decentralised routing

Decentralised Routing



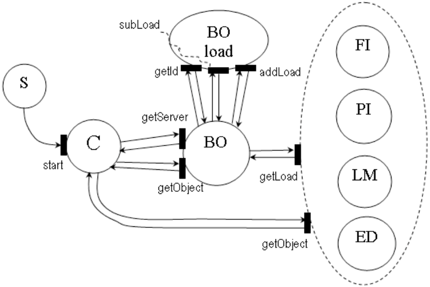
1. A client contacts a university site to download content.

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.

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

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

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

Model in PEPA

Clients

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

$(1 \leq i \leq k)$

Model in PEPA

Clients

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

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

Model in PEPA

Content mirrors

$$
\begin{array}{rcl}
\mathit{Mirror}_j & \stackrel{\mathit{def}}{=} & \big(\mathit{connect}_j, f_j(s)\big).\mathit{MirrorUploading}_j \\
\mathit{MirrorUploading}_j & \stackrel{\mathit{def}}{=} & \big(\mathit{download}_j, \top\big).\mathit{Mirror}_j
\end{array}
$$

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

Model in PEPA

Content mirrors

$$
\begin{aligned}
\mathit{Mirror}_j &\stackrel{\mathit{def}}{=} \big(\mathit{connect}_j, f_j(s)\big).\mathit{MirrorUploading}_j \\
\mathit{MirrorUploading}_j &\stackrel{\mathit{def}}{=} \big(\mathit{download}_j, \top\big).\mathit{Mirror}_j
\end{aligned}
$$

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

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%.

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%.

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

Model in PEPA

Dealing with overload

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

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

Model in PEPA

Dealing with overload

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

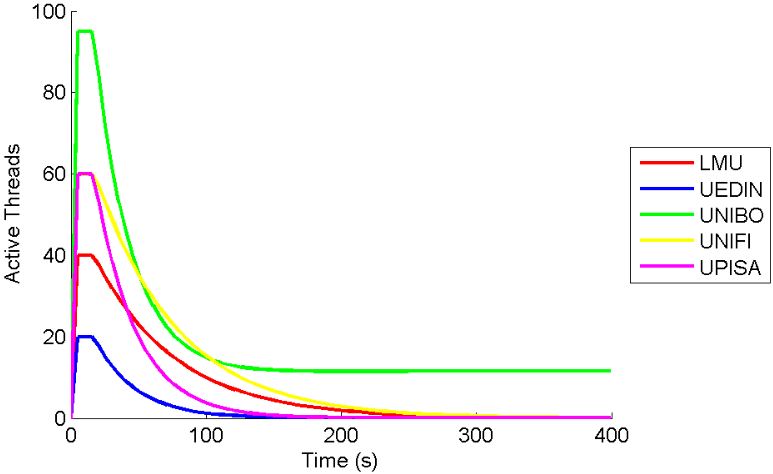
$$o(s) = \begin{cases} \top & f_i(s) = 0, \quad 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

The system as a whole with client and mirror site populations

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

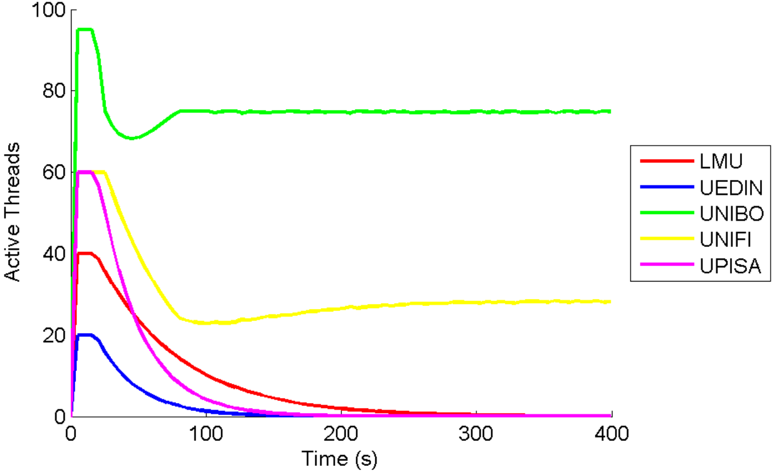
Numerical Results

$$r_{idle} = 0.001$$



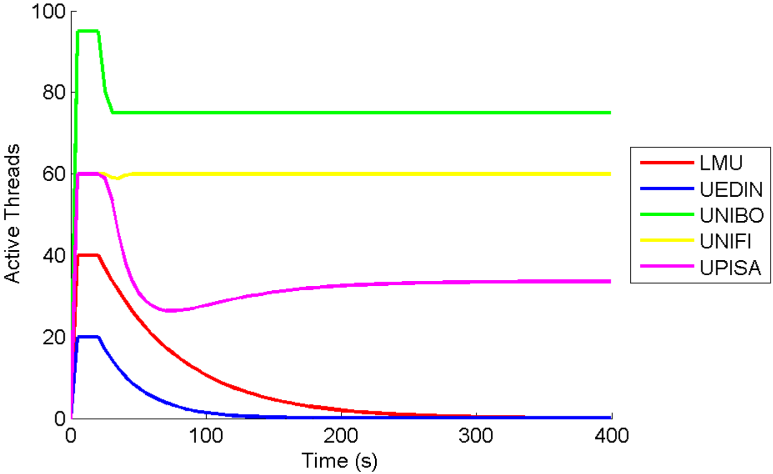
Numerical Results

$r_{idle} = 0.01$



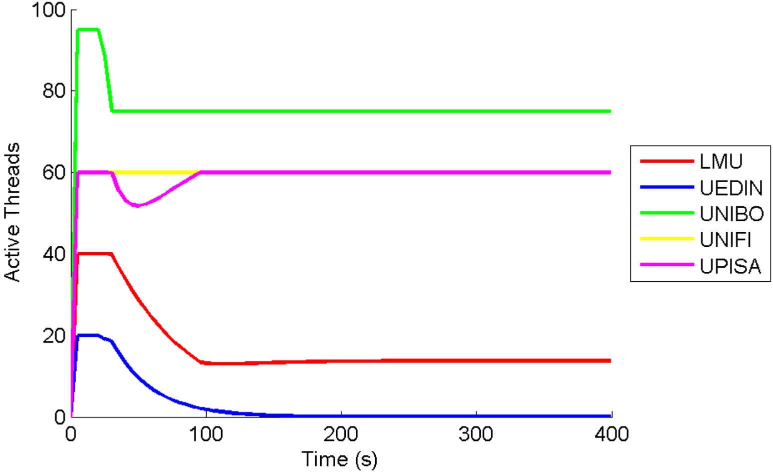
Numerical Results

$r_{idle} = 0.02$



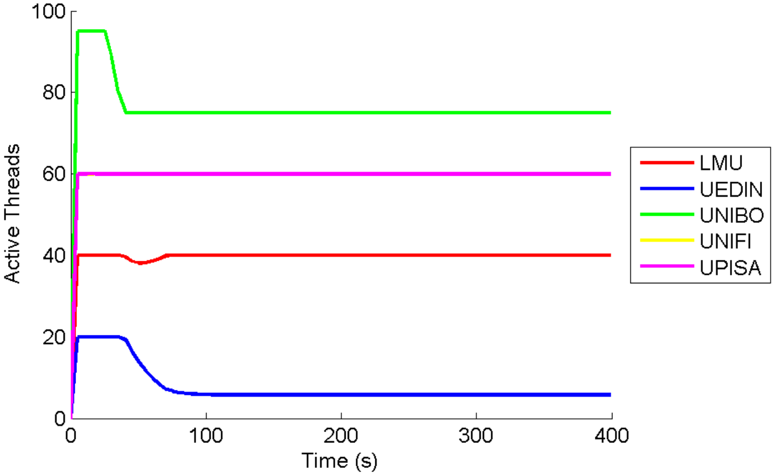
Numerical Results

$r_{idle} = 0.03$



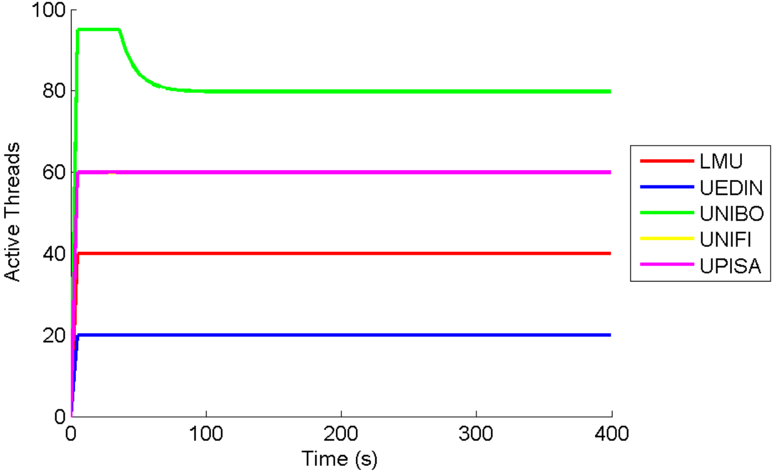
Numerical Results

$r_{idle} = 0.04$



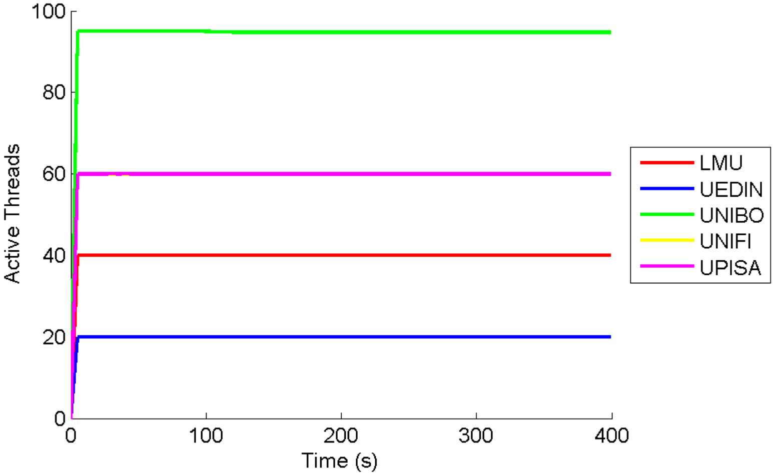
Numerical Results

$r_{idle} = 0.05$



Numerical Results

$r_{idle} = 0.06$



Outline

1 Introduction

- Stochastic Process Algebra
- Collective Dynamics

2 Continuous Approximation

- State variables
- Numerical illustration

3 Fluid-Flow Semantics

- Fluid Structured Operational Semantics

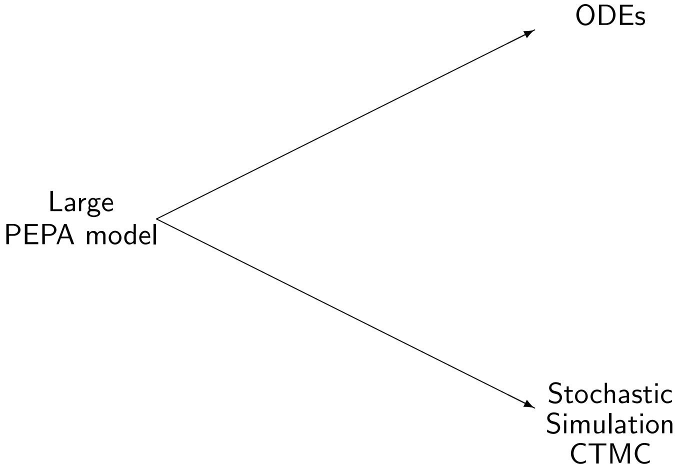
4 Example

- Scalable Web Services

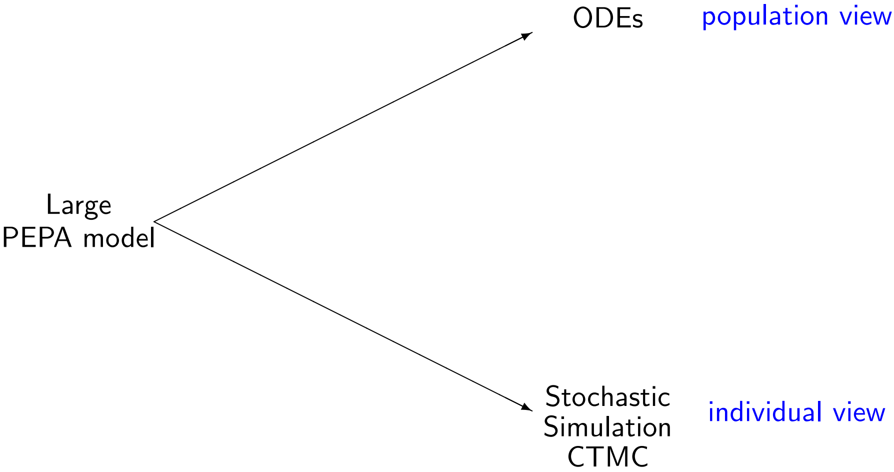
5 Conclusions

- Alternative Models

Alternative Representations



Alternative Representations



Large PEPA model

ODEs

population view

Stochastic Simulation CTMC

individual view

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.

- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.

- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n

- We can prove this using Kurtz's theorem: *Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes*, T.G. Kurtz, J. Appl. Prob. (1970).

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.

- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n

- We can prove this using Kurtz's theorem: *Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes*, T.G. Kurtz, J. Appl. Prob. (1970).

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

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.

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.

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.

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.
- This alternative view of systems has opened up many and exciting new research directions.

Thanks!

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Luca Bortolussi, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Luca Bortolussi, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

Acknowledgements: funding

Thanks to EPRSC for the Process Algebra for Collective Dynamics grant and the CEC IST-FET programme for the SENSORIA project which have supported this work.

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Luca Bortolussi, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

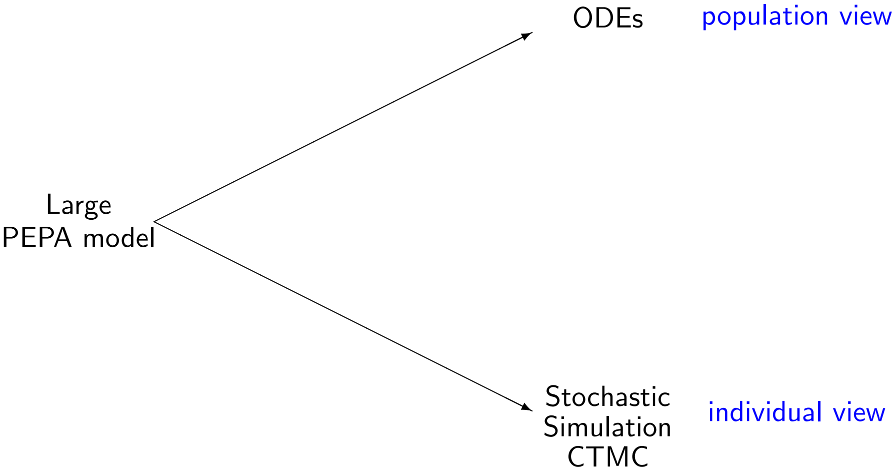
Acknowledgements: funding

Thanks to EPRSC for the Process Algebra for Collective Dynamics grant and the CEC IST-FET programme for the SENSORIA project which have supported this work.

More information:

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

Alternative Representations



Large PEPA model

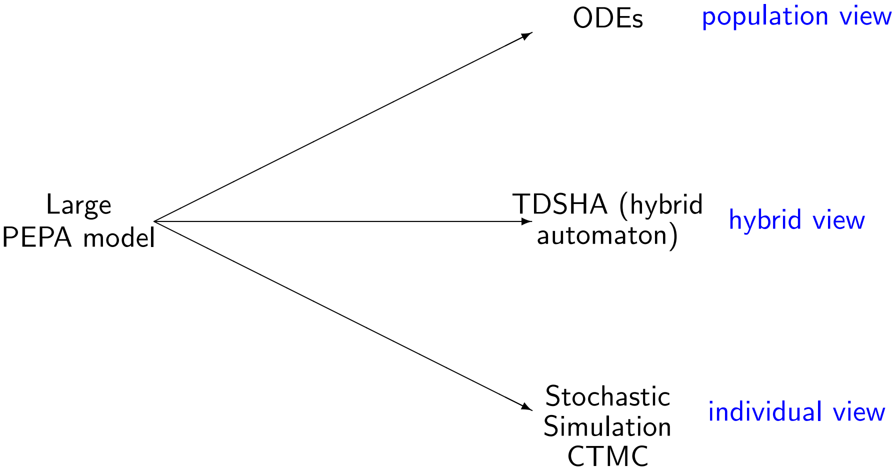
ODEs

population view

Stochastic Simulation CTMC

individual view

Alternative Representations



Large PEPA model

ODEs

population view

TDSHA (hybrid automaton)

hybrid view

Stochastic Simulation CTMC

individual view