Making Stochastic Process Algebras Count: Modelling Collective Dynamics

Jane Hillston

Laboratory for Foundations of Computer Science
and Centre for Systems Biology at Edinburgh
University of Edinburgh

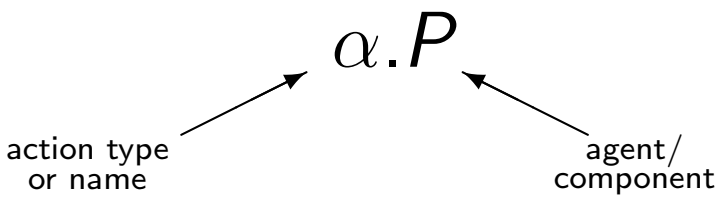
16th November 2010

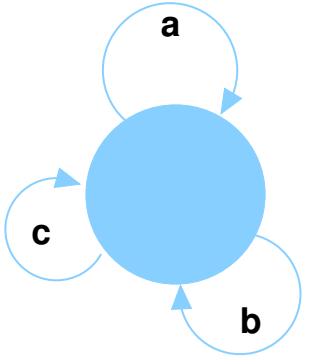
Outline

Outline

Process Algebra

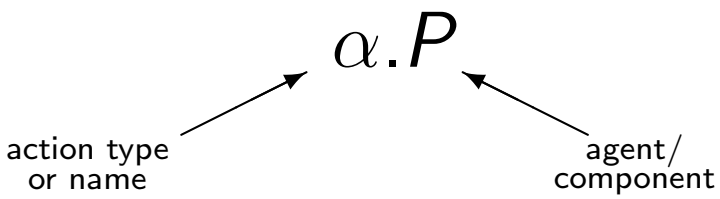
- Models consist of agents which engage in actions.

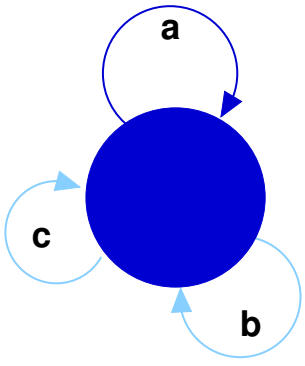




Process Algebra

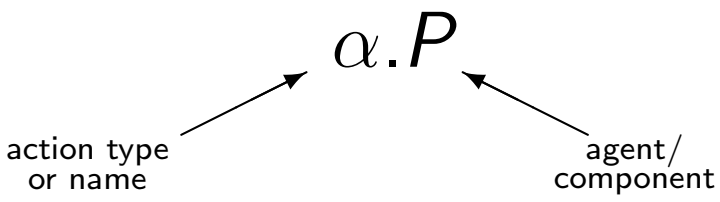
- Models consist of agents which engage in actions.

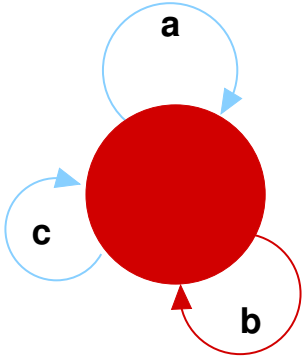




Process Algebra

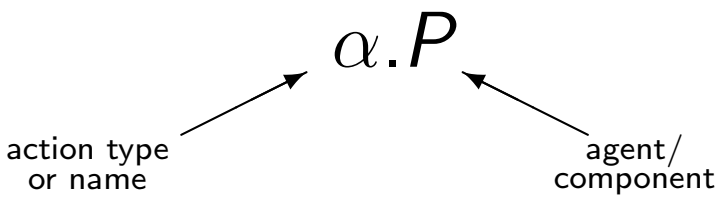
- Models consist of agents which engage in actions.

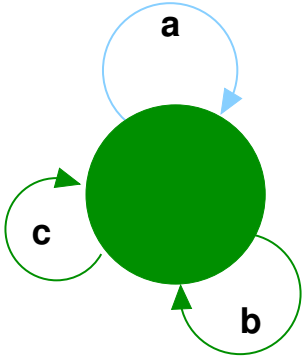




Process Algebra

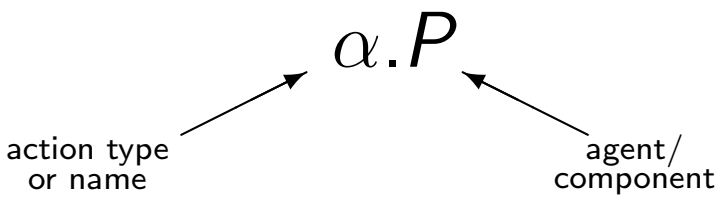
- Models consist of agents which engage in actions.

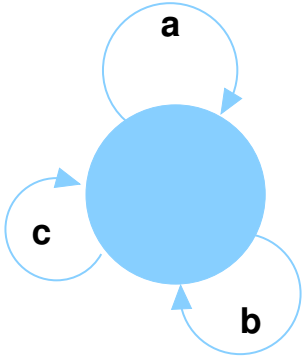




Process Algebra

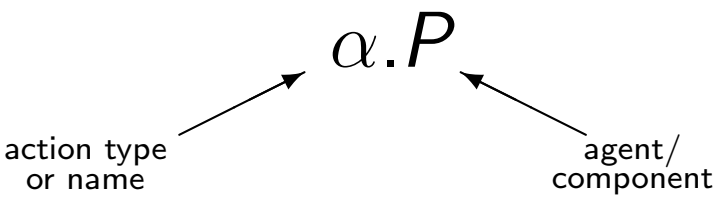
- Models consist of agents which engage in actions.

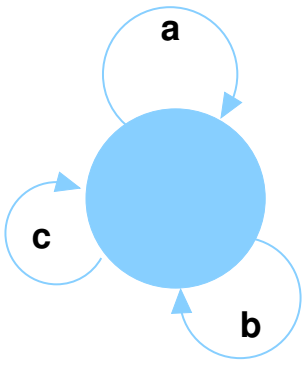




Process Algebra

- Models consist of agents which engage in actions.

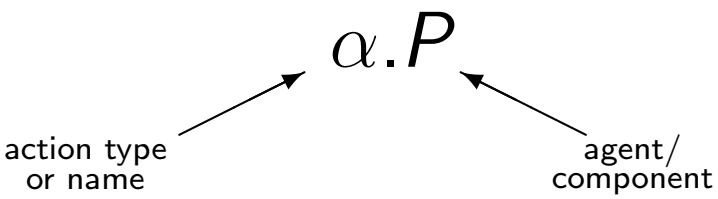


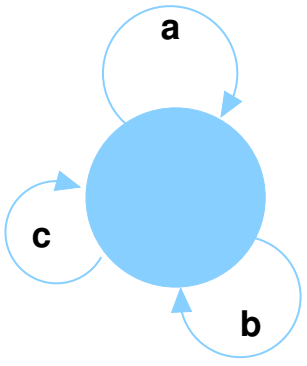


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process Algebra

- Models consist of agents which engage in actions.



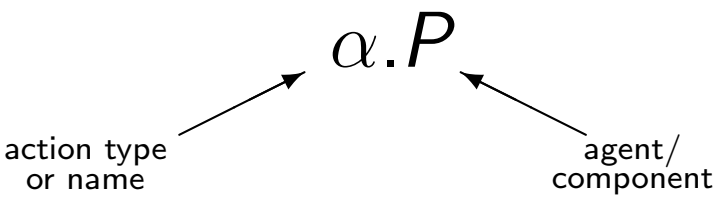


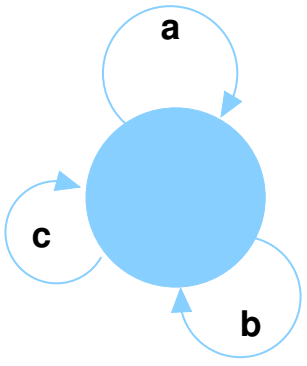
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra
model

Process Algebra

- Models consist of agents which engage in actions.



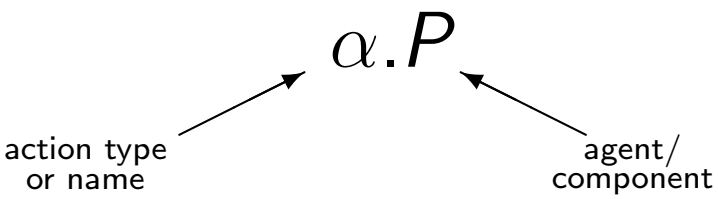


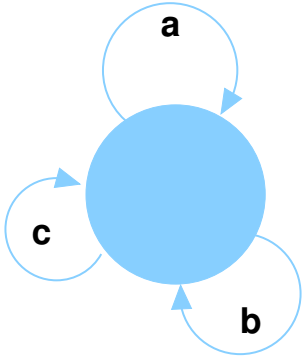
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model → SOS rules

Process Algebra

- Models consist of agents which engage in actions.



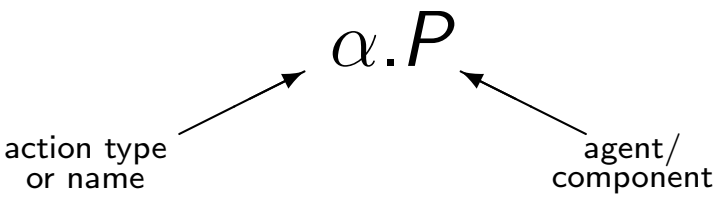


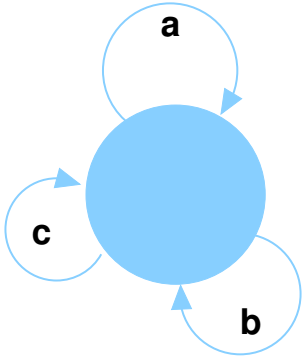
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model → SOS rules → Labelled transition system

Process Algebra

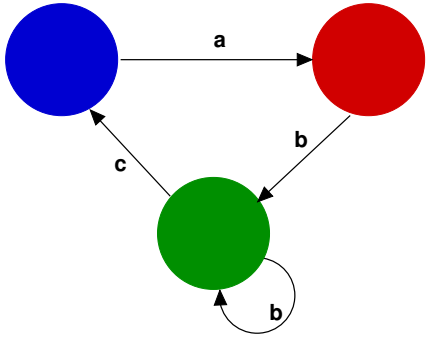
- Models consist of agents which engage in actions.





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

Process algebra model → SOS rules → Labelled transition system



A simple example: processors and resources

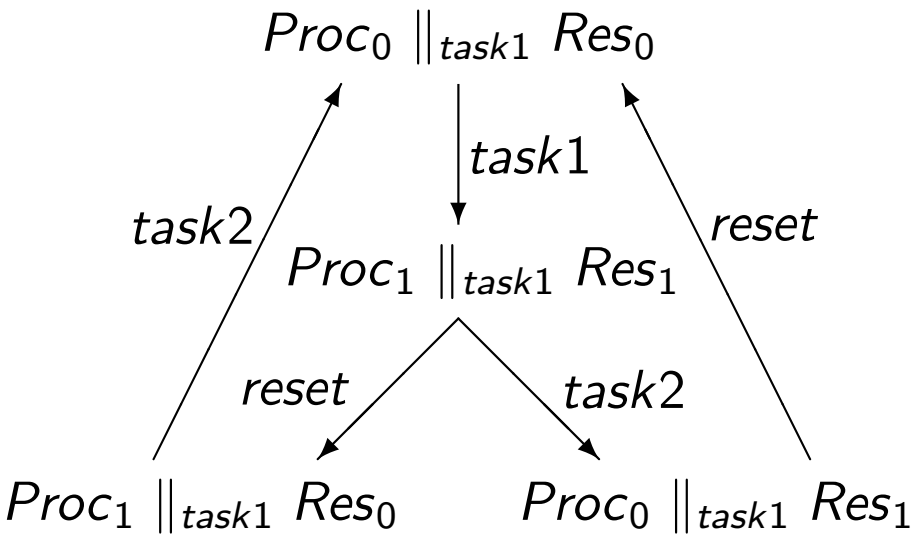
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} task1.Proc_1 \\
Proc_1 &\stackrel{def}{=} task2.Proc_0 \\
Res_0 &\stackrel{def}{=} task1.Res_1 \\
Res_1 &\stackrel{def}{=} reset.Res_0
\end{aligned}
$$

$$Proc_0 \parallel_{task1} Res_0$$

A simple example: processors and resources

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

$$Proc_0 \|_{task1} Res_0$$

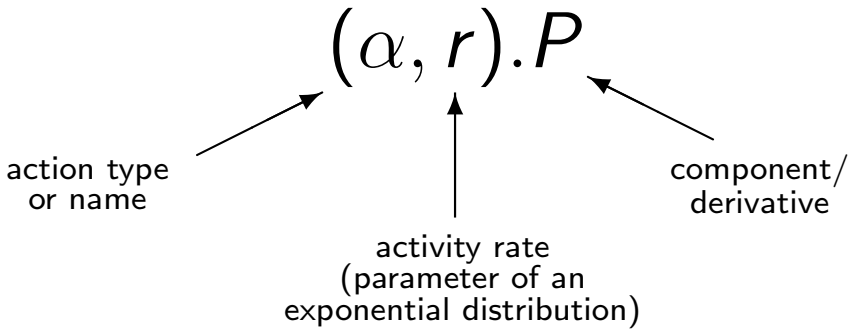


Stochastic process algebras

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

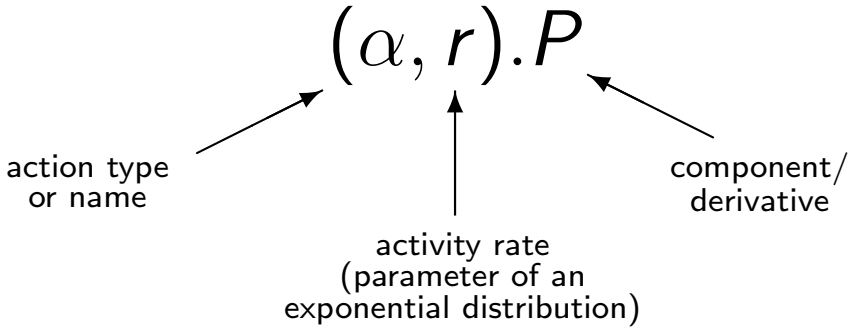
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



Stochastic Process Algebra

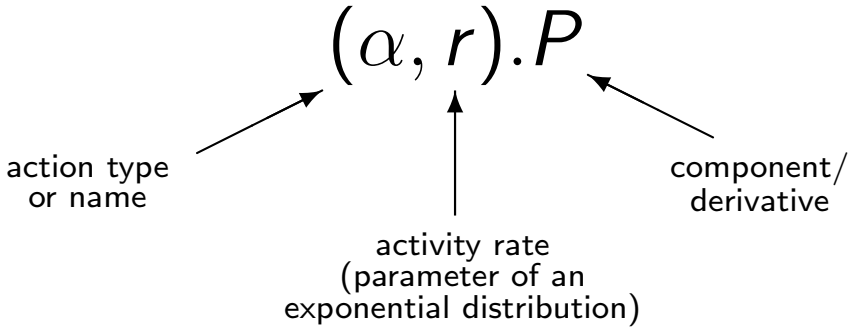
- Models are constructed from components which engage in activities.



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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

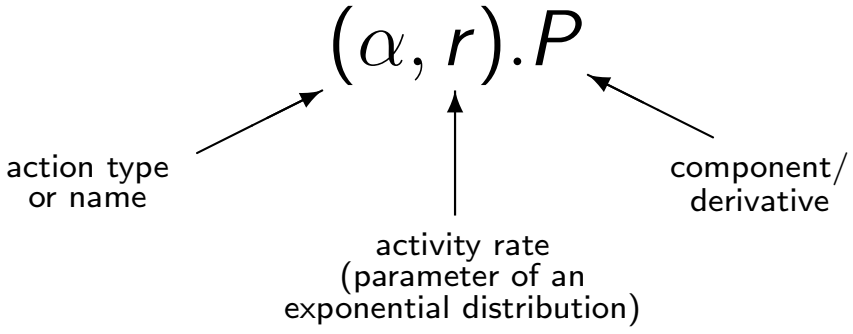


- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

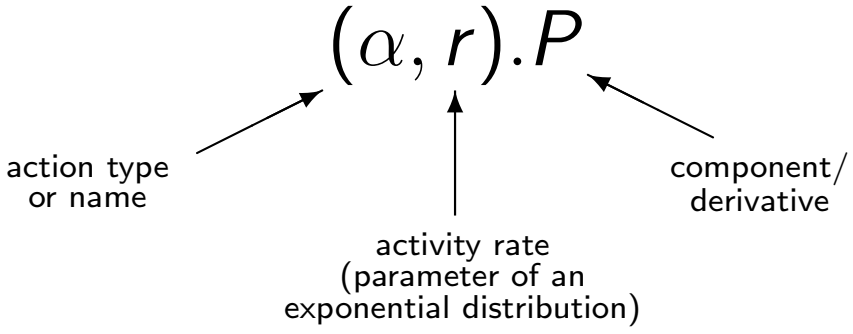


- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.

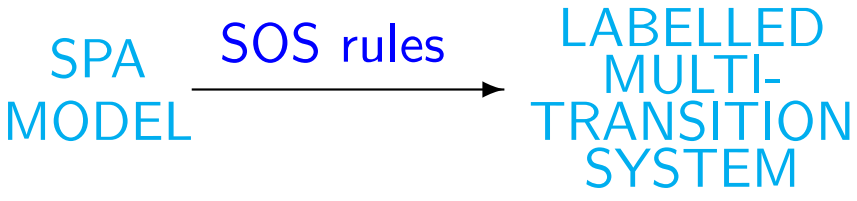
SPA MODEL $\xrightarrow{\text{SOS rules}}$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

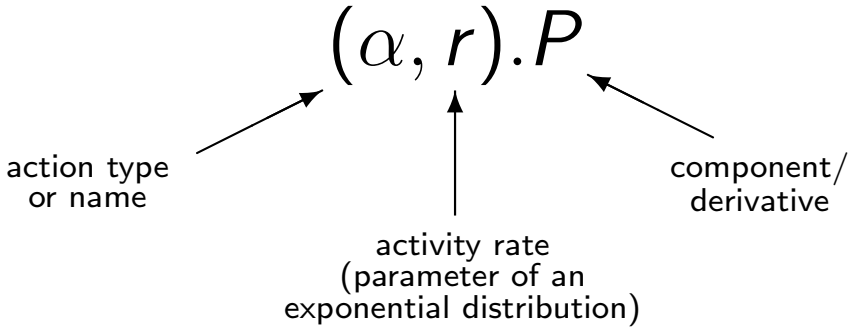


- The language is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.



Stochastic Process Algebra

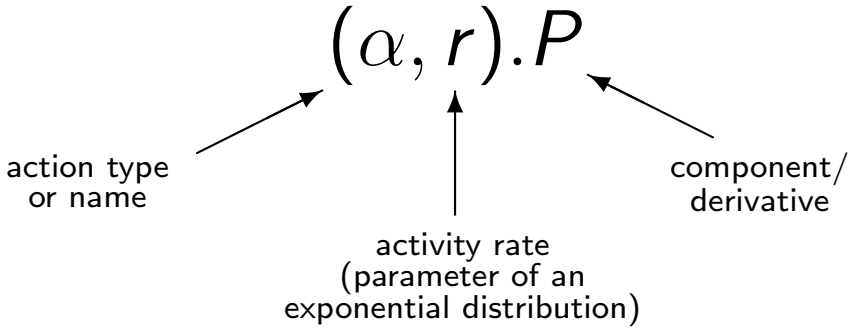
- Models are constructed from components which engage in activities.



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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

Integrated analysis

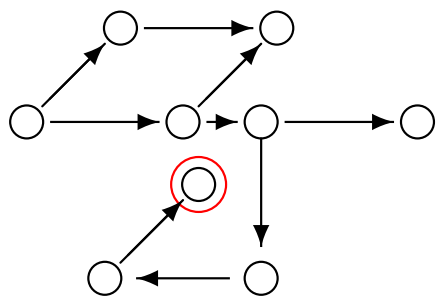
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

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

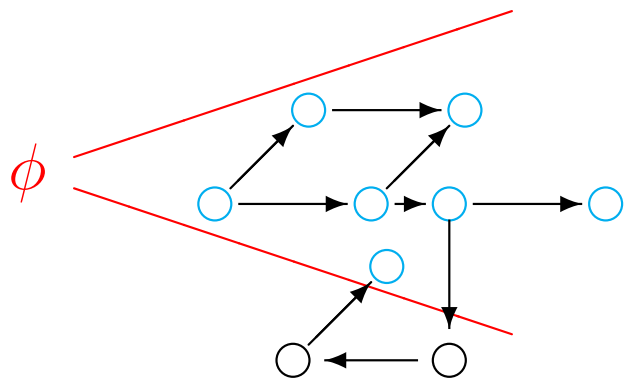


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

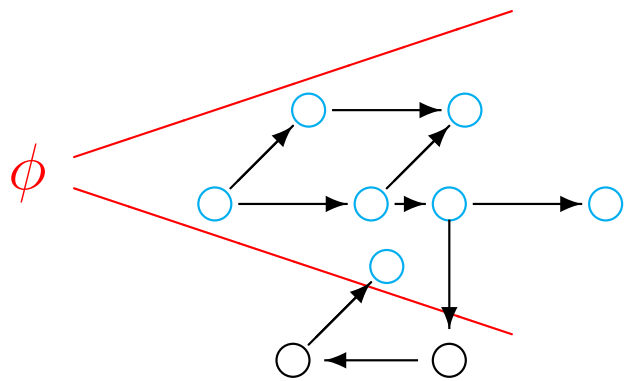


Integrated analysis

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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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)

$(\alpha, f).P$	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.

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

PEPA: some recent case studies

- QoS protocols for mobile devices
  (Wang, Lauriston and Hillston, Edinburgh)

PEPA: some recent case studies

- QoS protocols for mobile devices
  (Wang, Lauriston and Hillston, Edinburgh)
- Switch behaviour in active networks
  (Hillston, Kloul and Mokhtari, Edinburgh and Versailles)

PEPA: some recent case studies

- QoS protocols for mobile devices
  (Wang, Lauriston and Hillston, Edinburgh)
- Switch behaviour in active networks
  (Hillston, Kloul and Mokhtari, Edinburgh and Versailles)
- Task scheduling in a Grid-based processing system
  (Benoit, Cole, Gilmore and Hillston, Edinburgh)

PEPA: some recent case studies

- QoS protocols for mobile devices
  (Wang, Lauriston and Hillston, Edinburgh)
- Switch behaviour in active networks
  (Hillston, Kloul and Mokhtari, Edinburgh and Versailles)
- Task scheduling in a Grid-based processing system
  (Benoit, Cole, Gilmore and Hillston, Edinburgh)
- Probability of timely airbag deployment
  (Clark, Gilmore and Tribastone, Edinburgh)

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

Gary Linden, Amazon, quoted on `http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency`

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

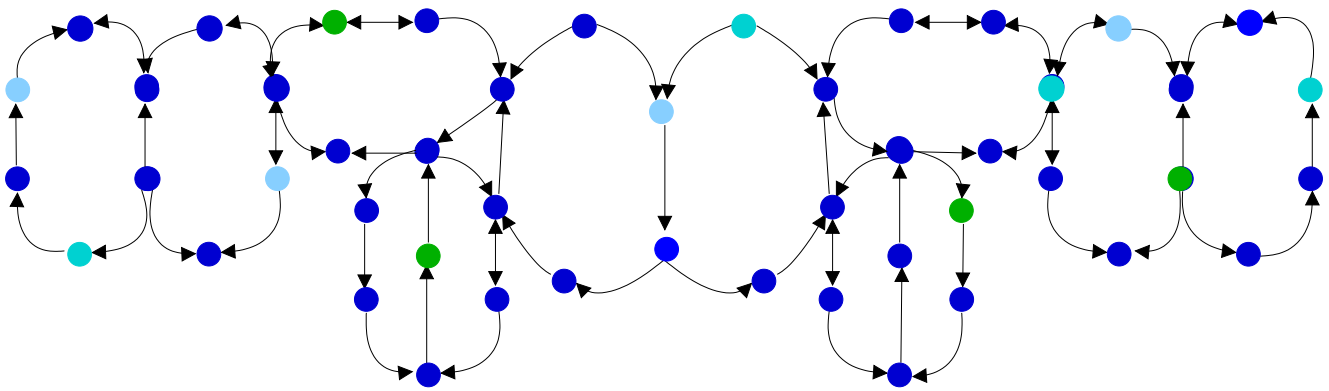
In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

Gary Linden, Amazon, quoted on `http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency`

AOL, Bing and Google report similar findings.

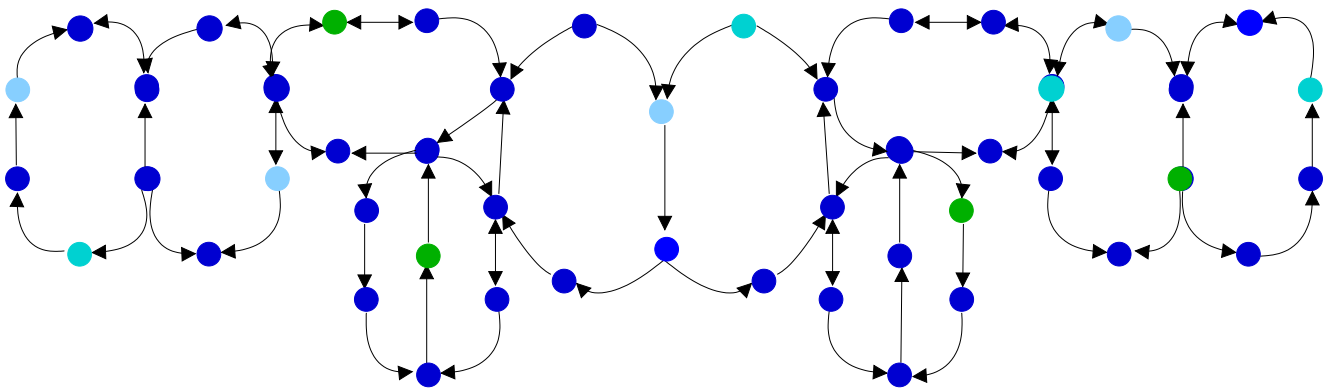
Collective Dynamics

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

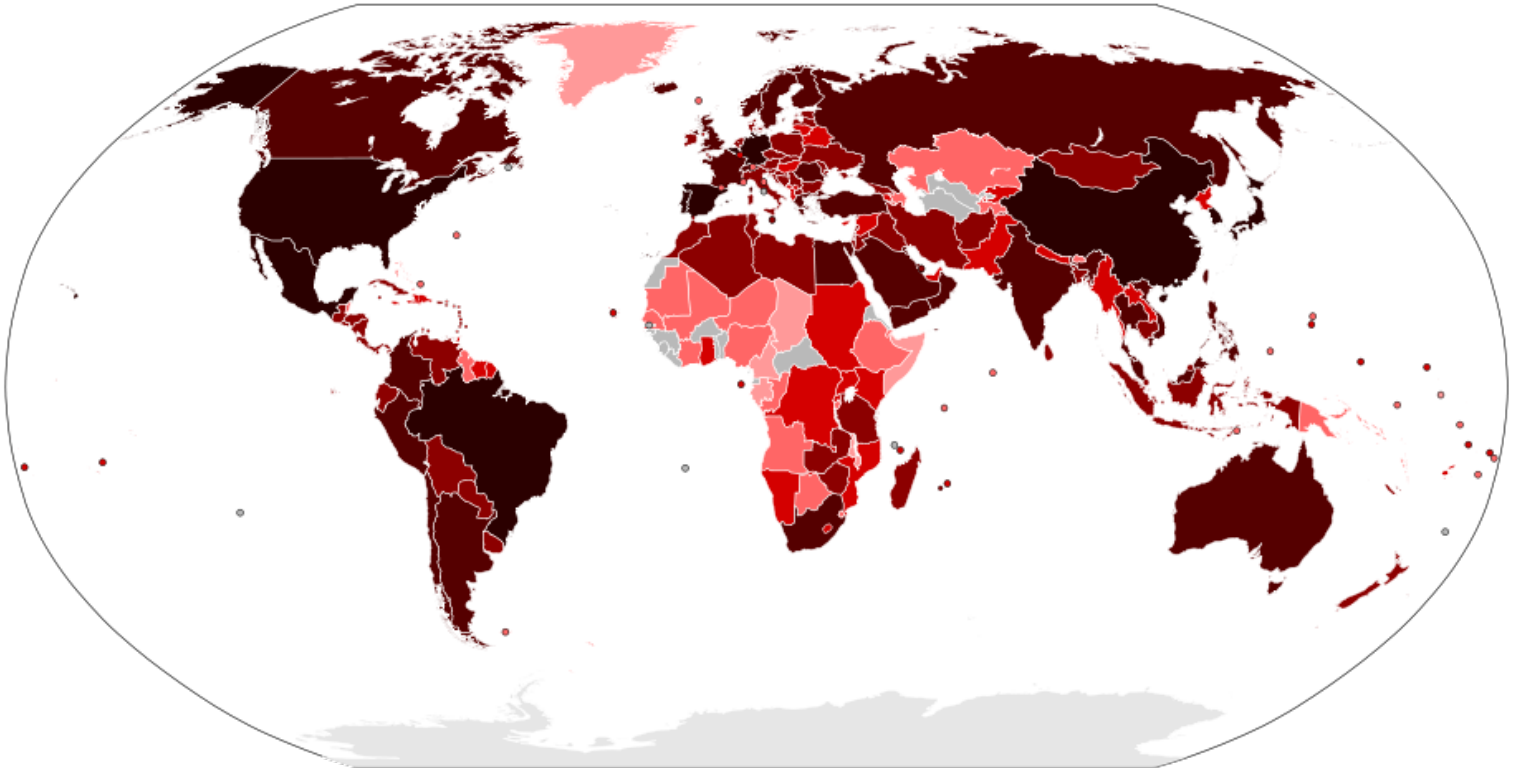
In the natural world there are many instances of collective behaviour and its consequences:

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

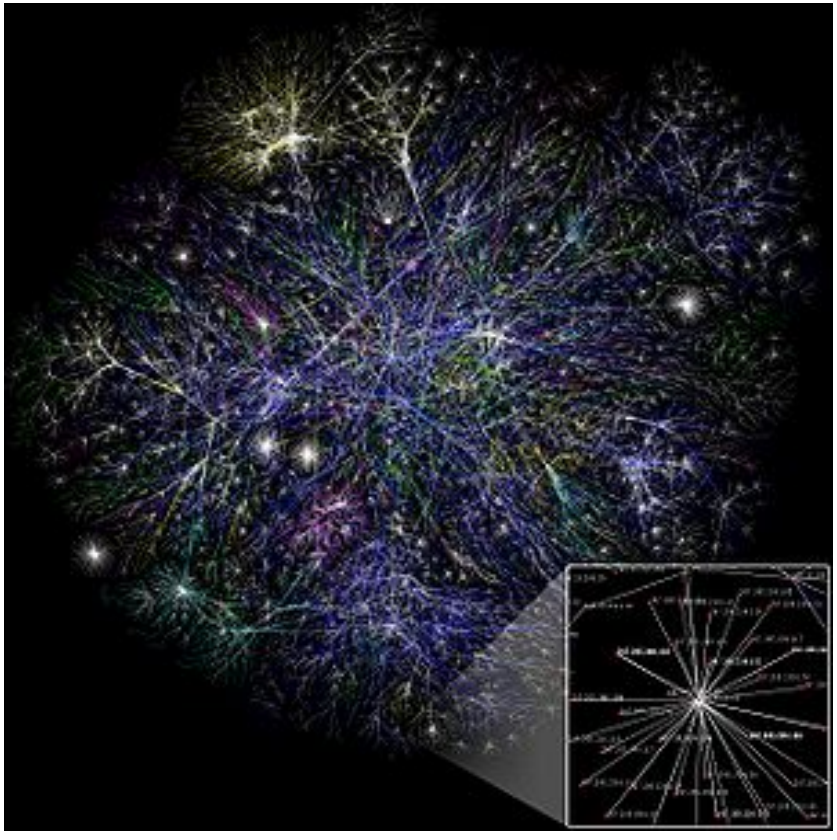
Collective Behaviour

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

Love Parade, Germany 2006

Collective Behaviour

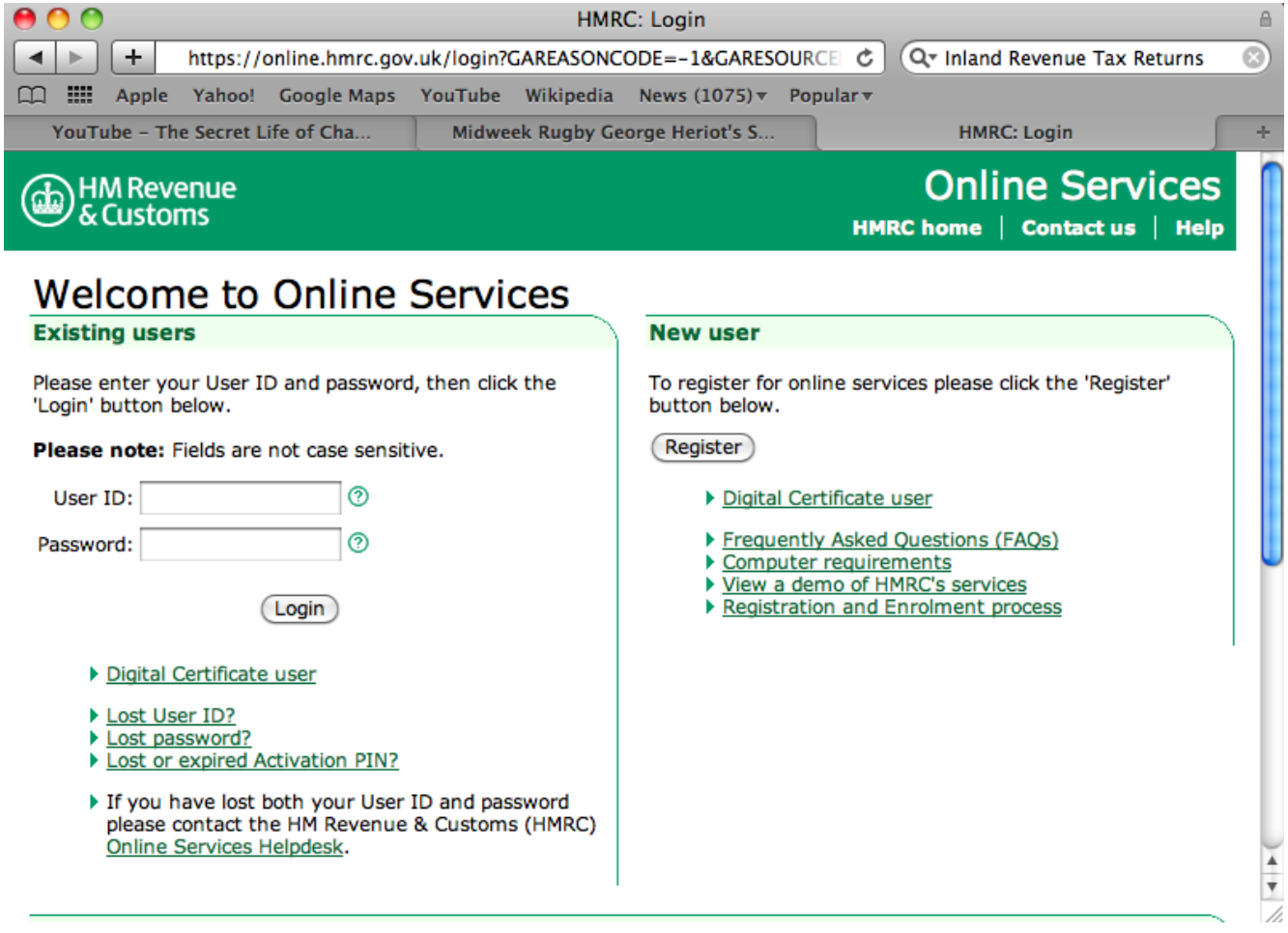
This is also true in the man-made and engineered world:



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 constructing models of such systems:

Process Algebra and Collective Dynamics

Process algebra are well-suited to constructing models of such systems:

- Developed to represent concurrent behaviour compositionally;

Process Algebra and Collective Dynamics

Process algebra are well-suited to constructing models of such systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;

Process Algebra and Collective Dynamics

Process algebra are well-suited to constructing models of such systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Stochastic extensions allow the dynamics of system behaviour to be captured;

Process Algebra and Collective Dynamics

Process algebra are well-suited to constructing models of such systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Stochastic extensions allow the dynamics of system behaviour to be captured;
- Incorporate formal apparatus for reasoning about the behaviour of systems.

Process Algebra and Collective Dynamics

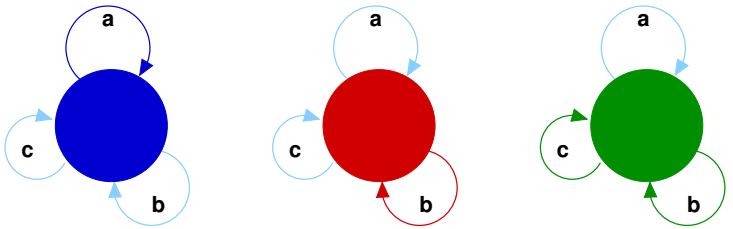
Process algebra are well-suited to constructing models of such systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Stochastic extensions allow the dynamics of system behaviour to be captured;
- Incorporate formal apparatus for reasoning about the behaviour of systems.

Unfortunately, whilst theoretically possible, the analysis of such systems through the standard process algebra approaches — explicitly building the state space — is not generally feasible.

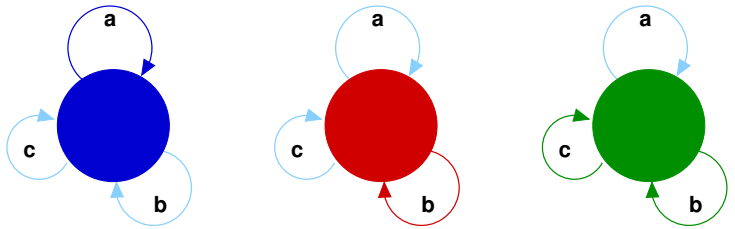
Solving discrete state models

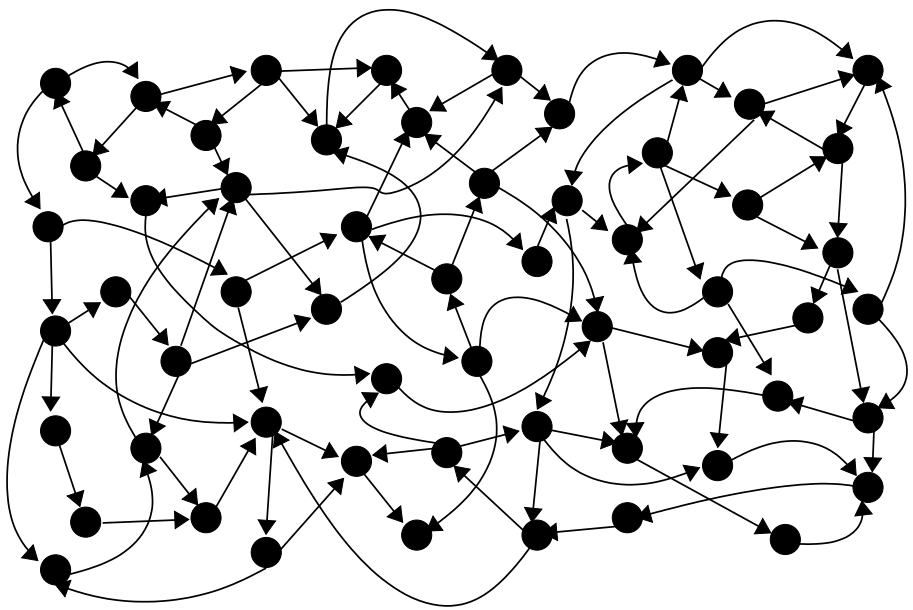
Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.



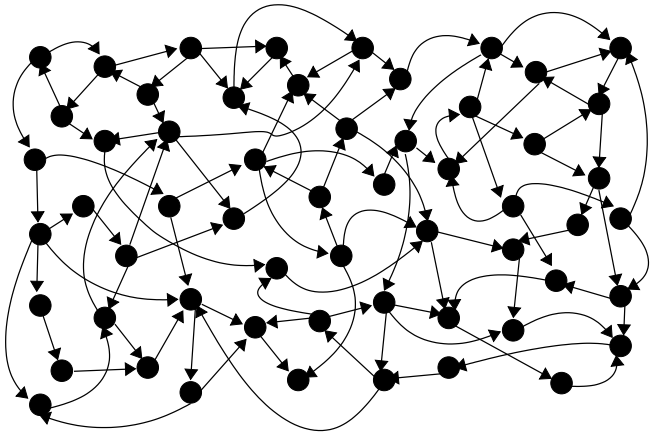
Solving discrete state models

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



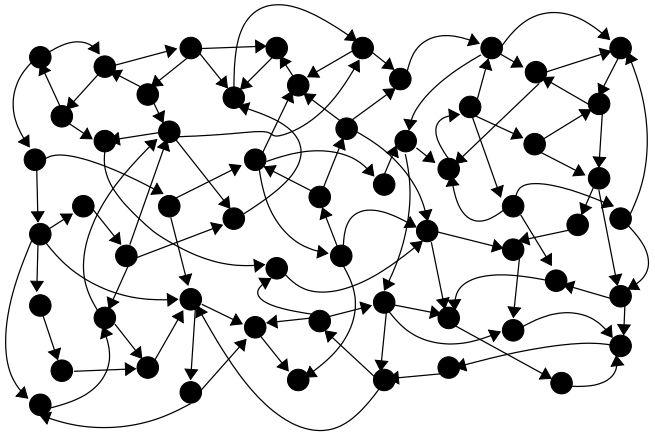


Solving discrete state models



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

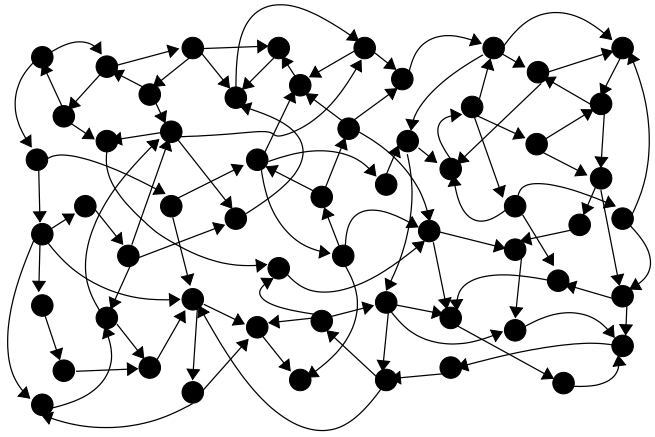
Solving discrete state models



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

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

Solving discrete state models



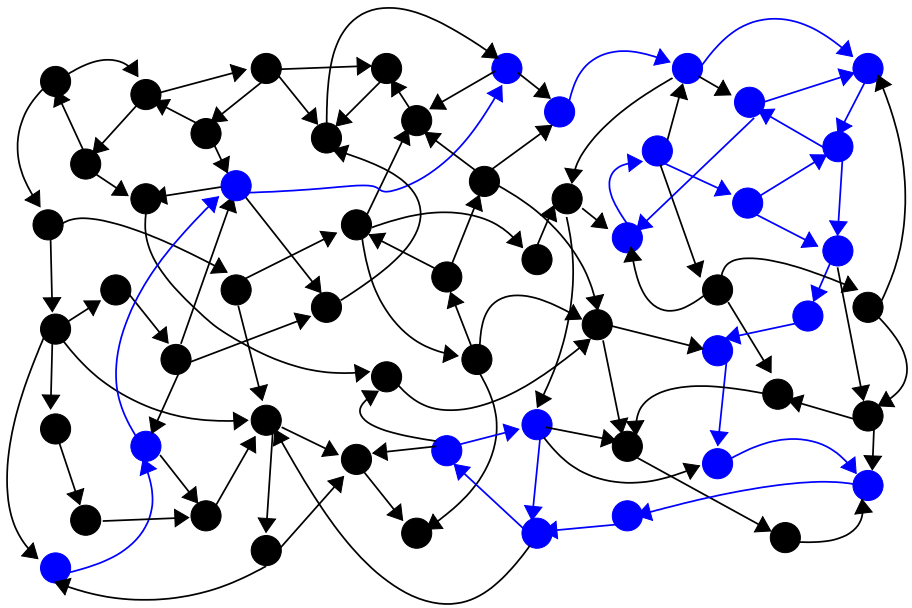
When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.



State space explosion

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

The CODA project

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

The CODA project

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

One approach to this is to keep the discrete state representation in the model and to evaluate it algorithmically rather than analytically, i.e. carry out a discrete event simulation of the model to explore its possible behaviours.

The CODA project

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

One approach to this is to keep the discrete state representation in the model and to evaluate it algorithmically rather than analytically, i.e. carry out a discrete event simulation of the model to explore its possible behaviours.

Another approach is to make a shift to population statistics.

Population statistics: emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

Population statistics: emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

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

Population statistics: emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

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

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

Population statistics: emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

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

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

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

Example: Performance as an emergent behaviour

In this framework we must think about the performance of a system from 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: Performance as an emergent behaviour

In this framework we must think about the performance of a system from 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.

Example: Performance as an emergent behaviour

In this framework we must think about the performance of a system from 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

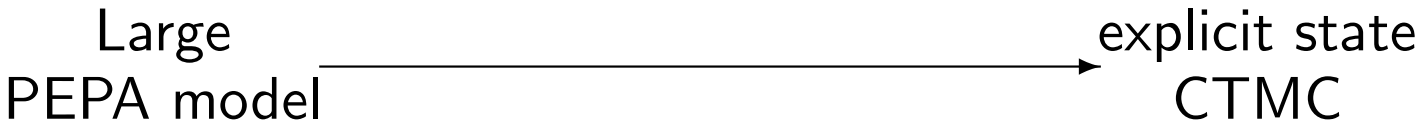
The novelty in this approach is twofold:

- 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:

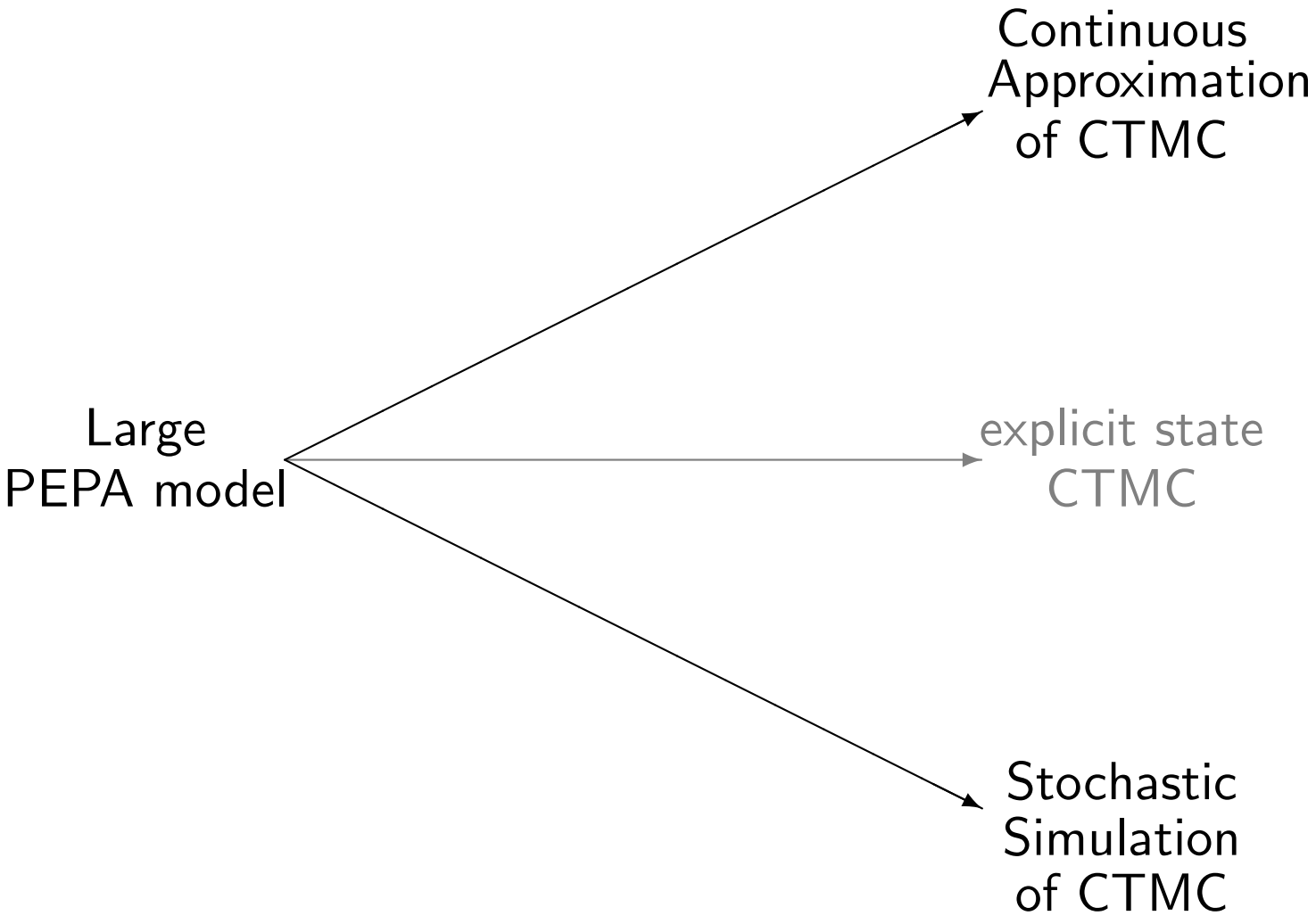
Crowd dynamics
Technology enhancement is creating new possibilities for directing crowd movements in buildings and urban spaces, for example for emergency egress, which are not yet well-understood.

Outline

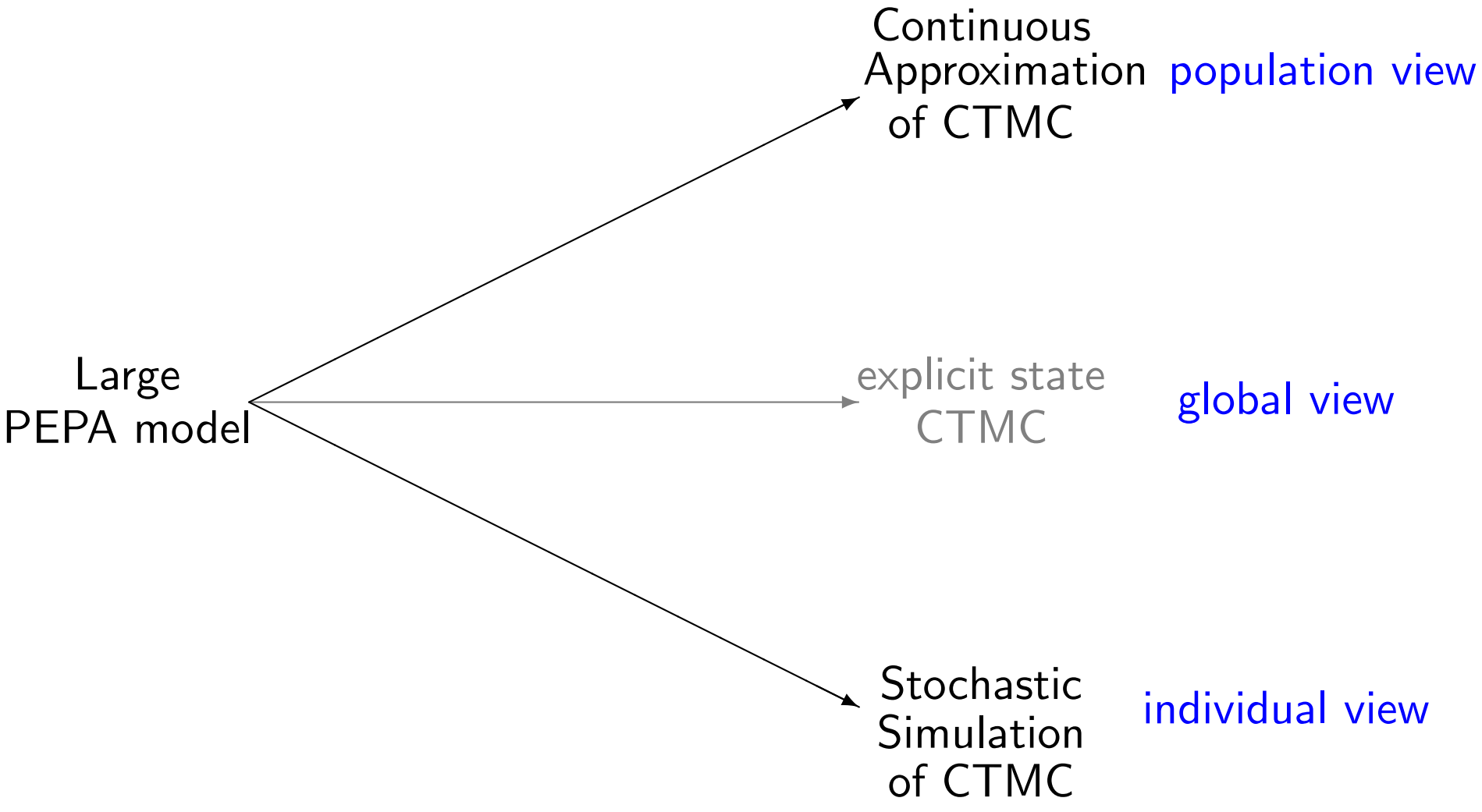
Alternative Representations



Alternative Representations



Alternative Representations



Continuous Approximation

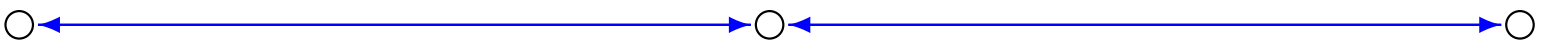
Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

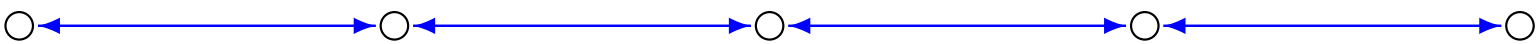


Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

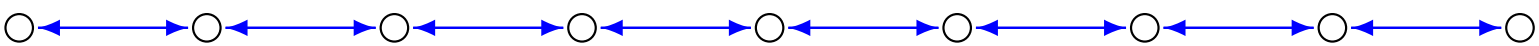


Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

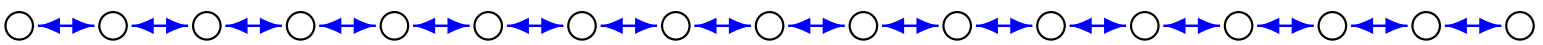


Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.



Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

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

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates 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.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Models suitable for counting abstraction

- In Bio-PEPA components are parameterised with a counting variable and the definition of an action records the impact that an action has on the counting variable.

Simple example revisited

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

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

Simple example revisited

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

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

CTMC interpretation

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

Simple example revisited

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

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

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

Simple example revisited

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

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

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_1\, x_1, r_3\, x_3) + r_2\, x_2 \\
x_1 &= \text{no. of } Proc_1
\end{aligned}
$$

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

Simple example revisited

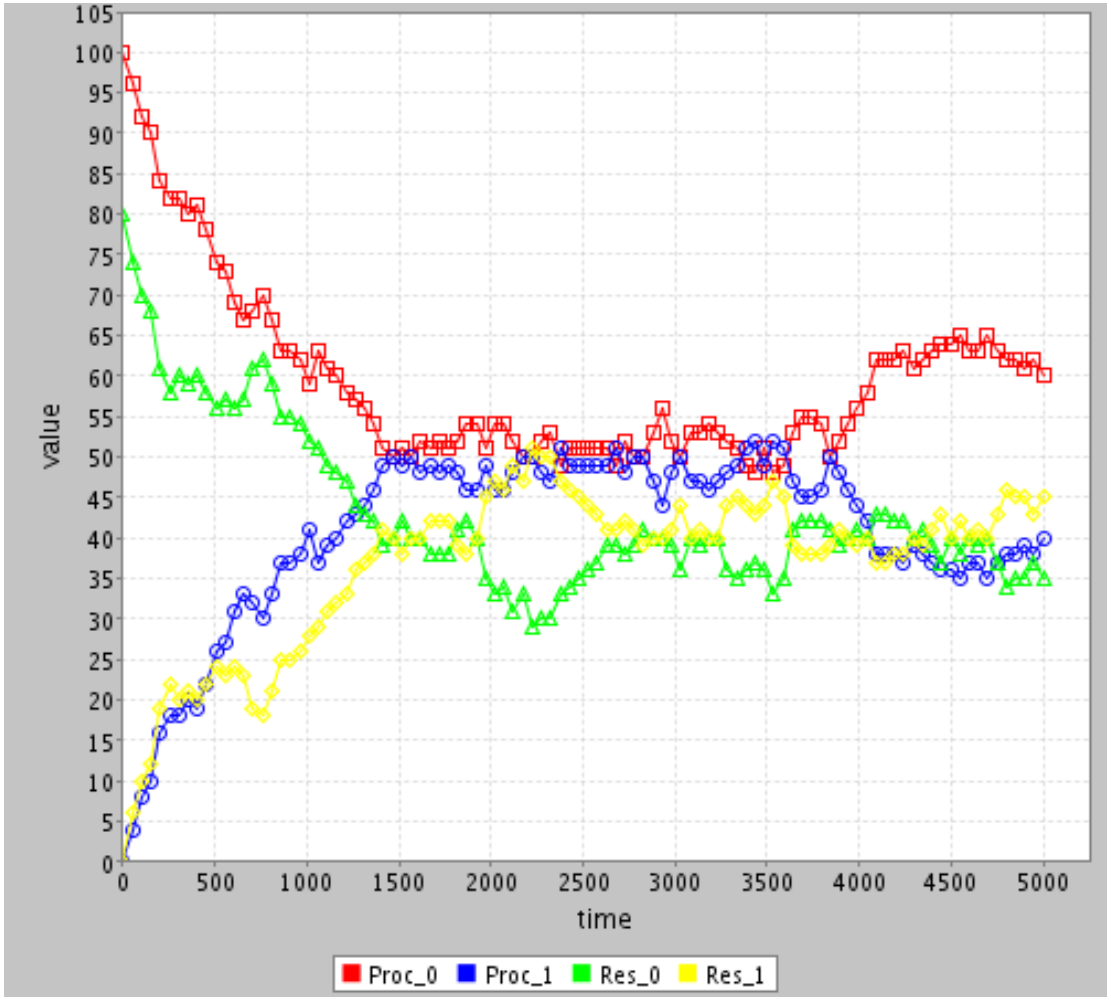
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

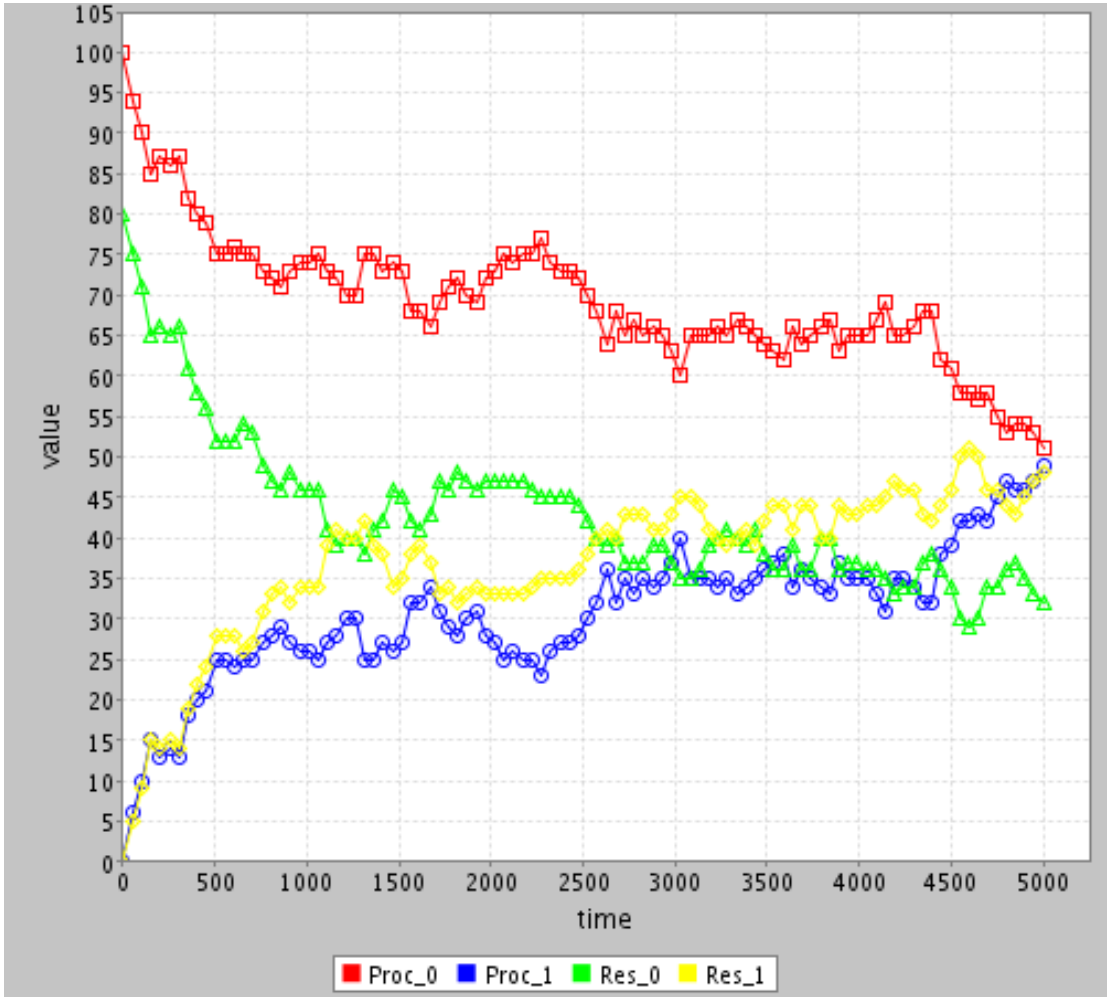
ODE interpretation

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

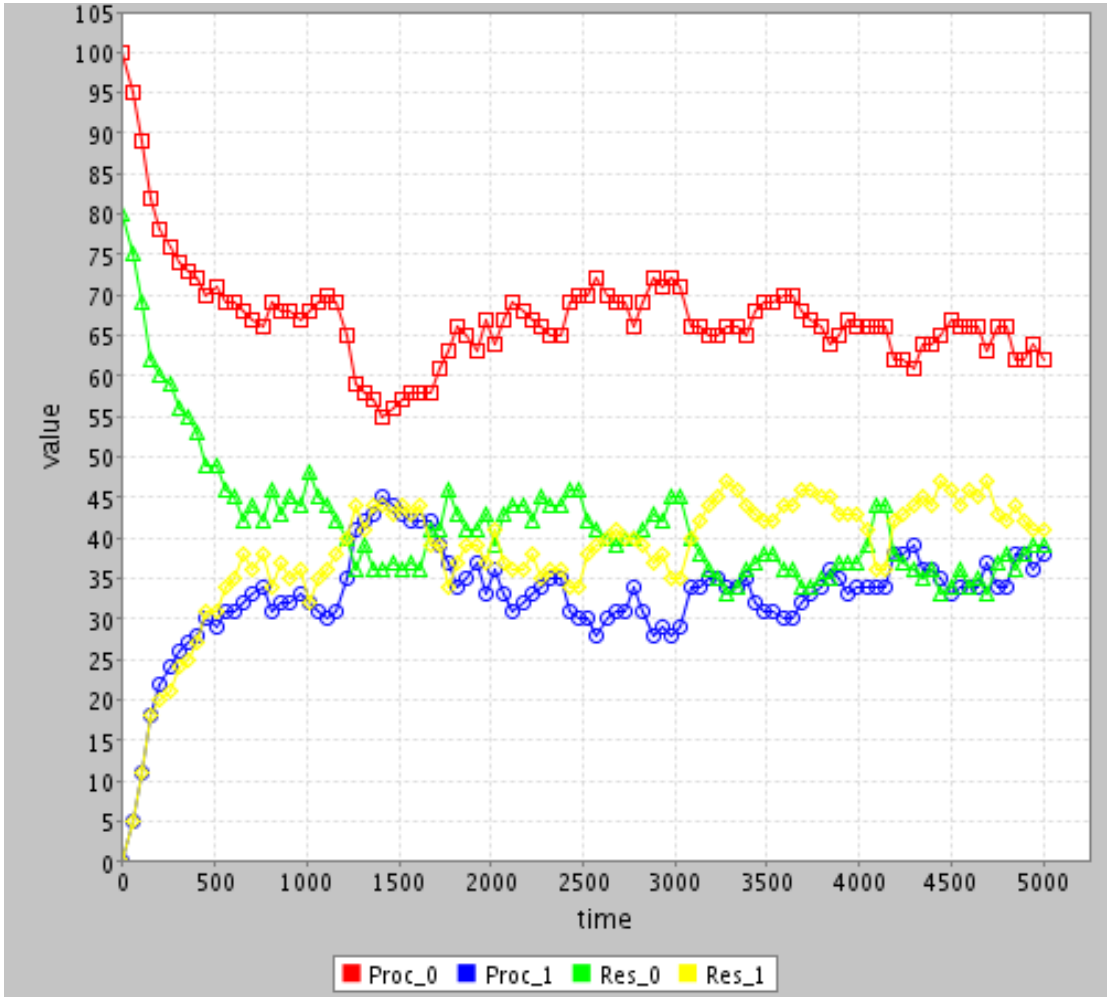
100 processors and 80 resources (simulation run A)



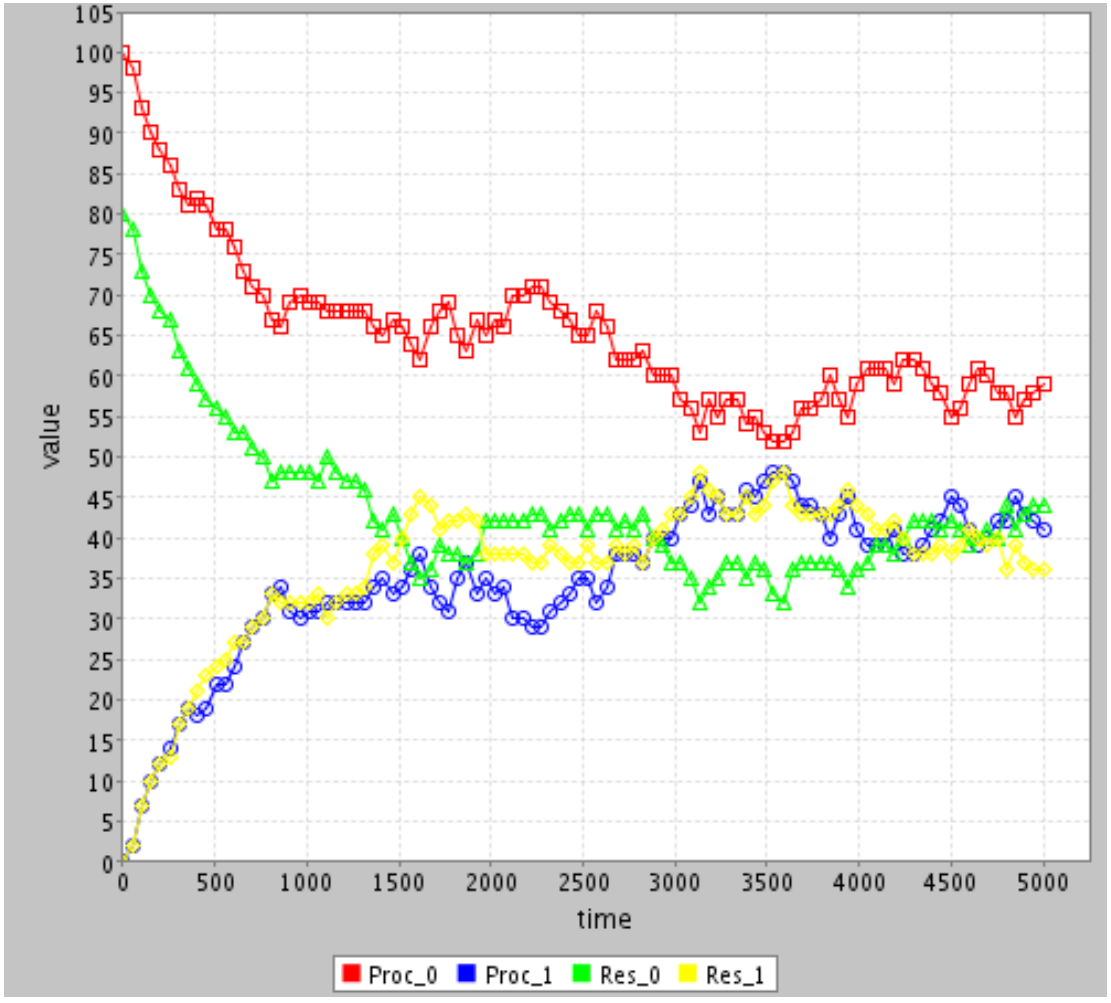
100 processors and 80 resources (simulation run B)



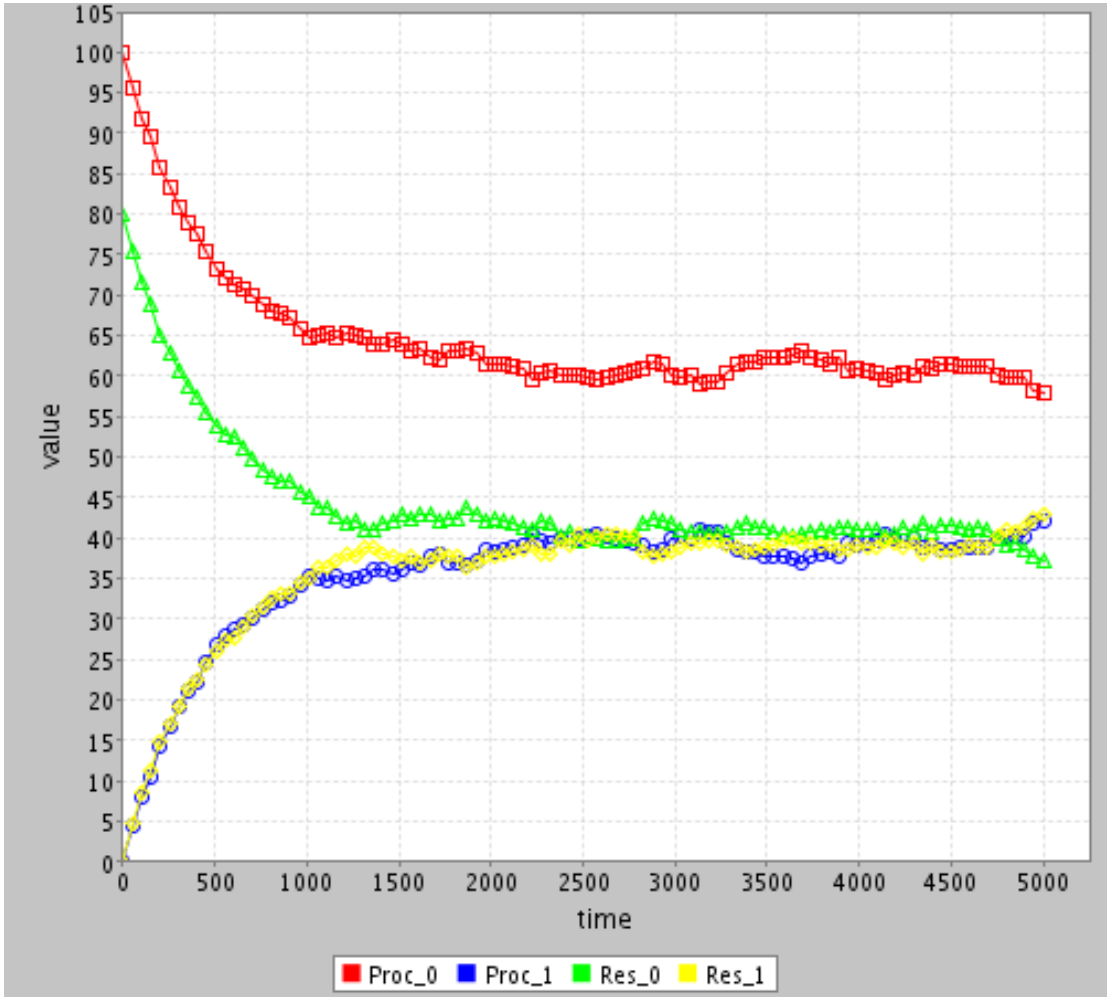
100 processors and 80 resources (simulation run C)



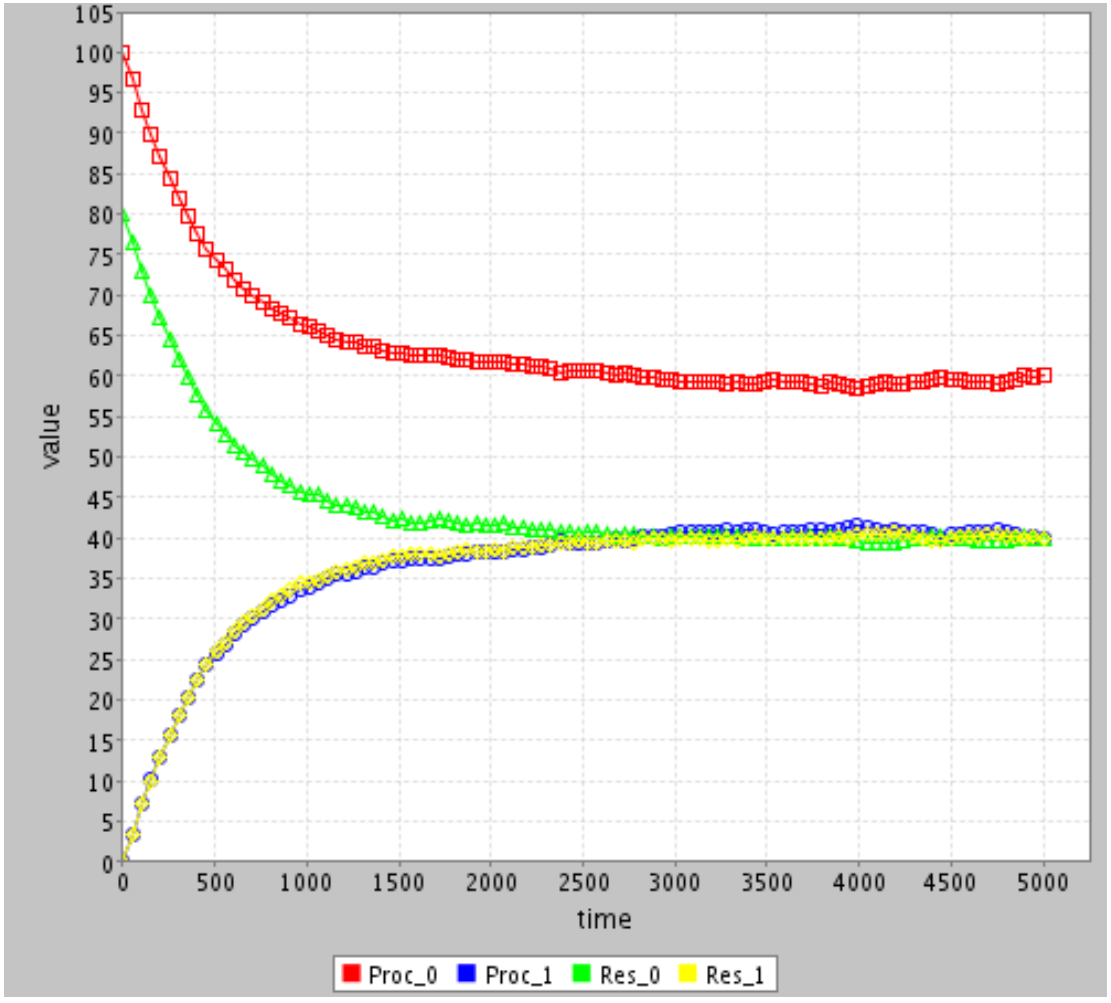
100 processors and 80 resources (simulation run D)



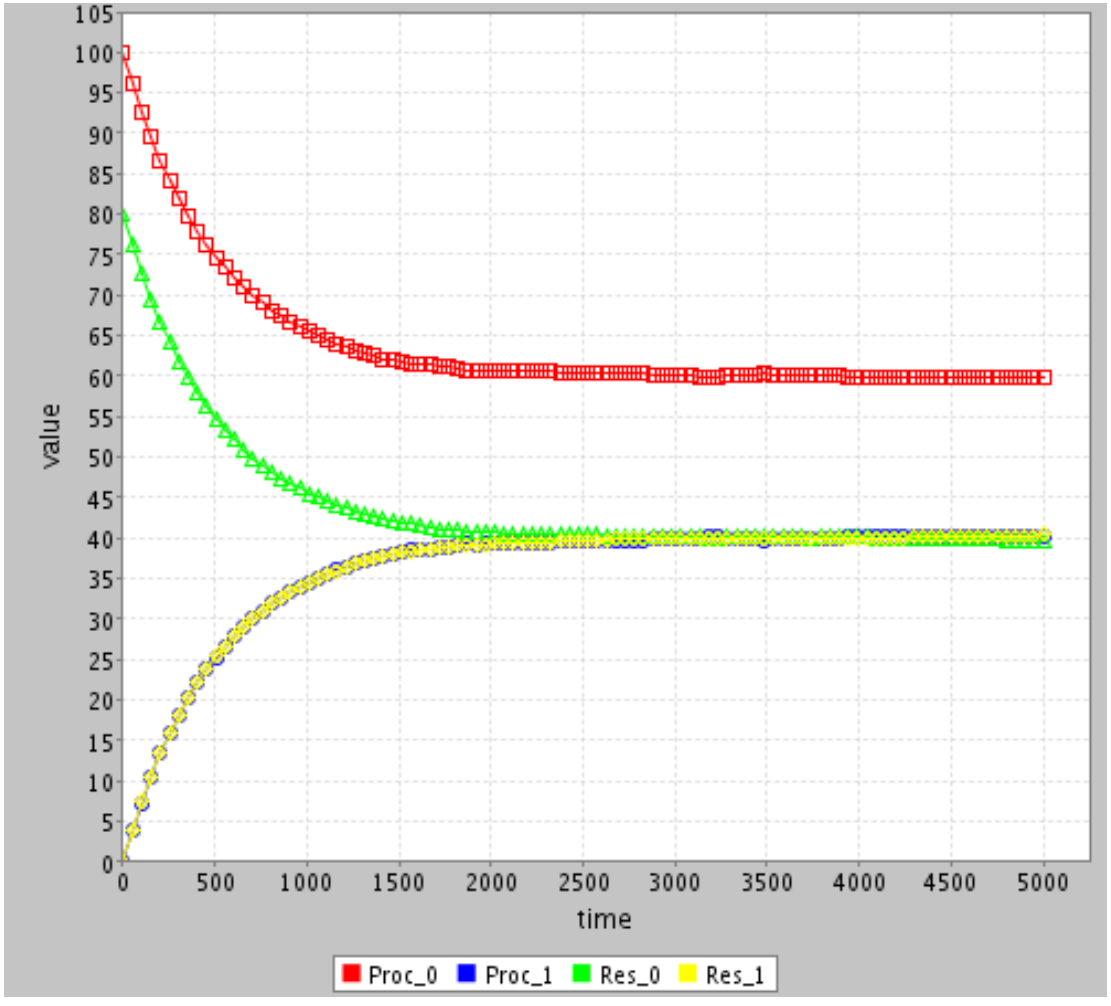
100 processors and 80 resources (average of 10 runs)



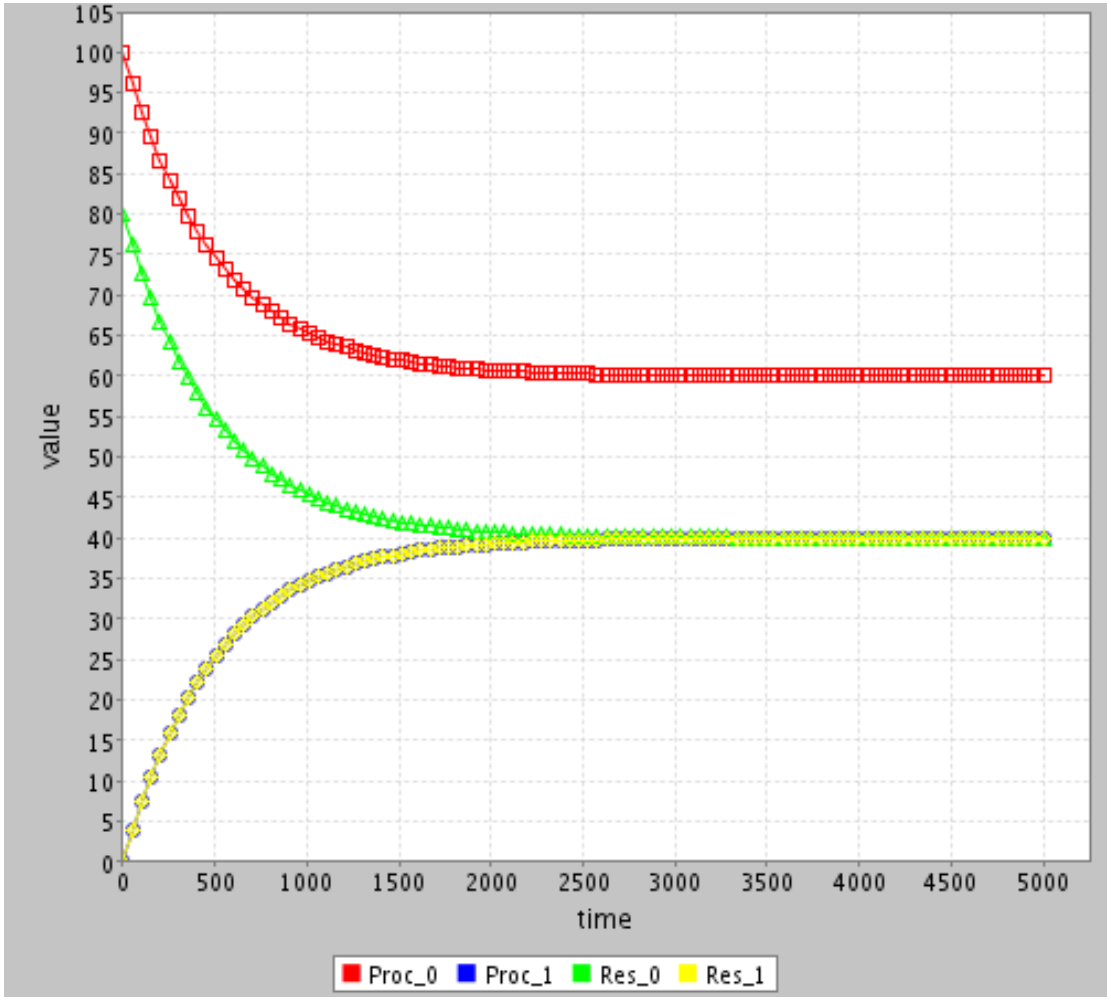
100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (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.

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.

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.

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. Make the counting abstraction (*Context Reduction*)

Fluid Structured Operational Semantics

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

1. Make the counting abstraction (*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. Make the counting abstraction (*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. Make the counting abstraction (*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

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

Location Dependency

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

$$\Downarrow$$

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

Location Dependency

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

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

Apparent Rate Calculation

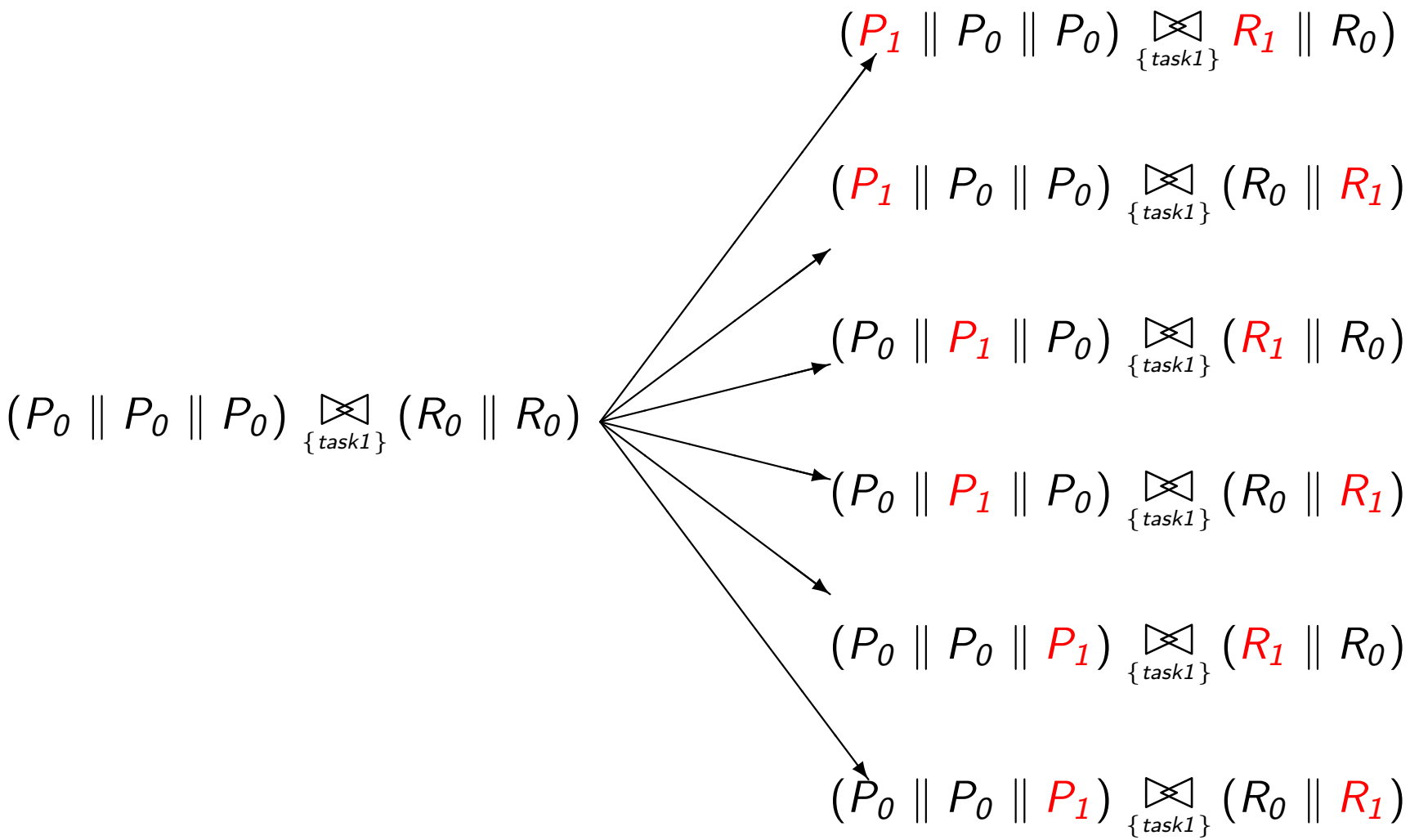
$$\frac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie}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

Apparent Rate Calculation

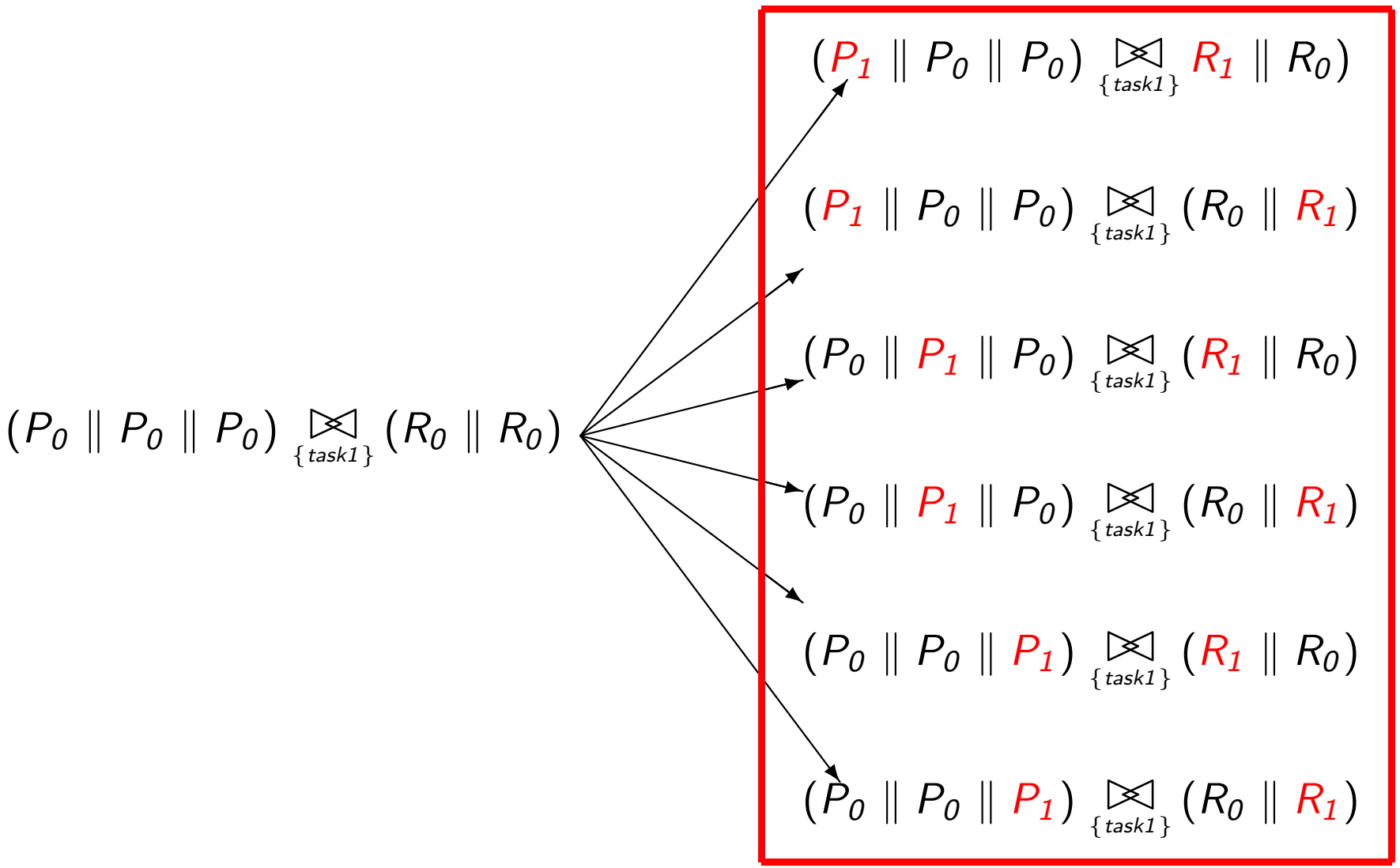
$$\frac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1\xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3\xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie}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

$$r(\xi) = \min\left(r_1\xi_1, r_3\xi_4\right)$$

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



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



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

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

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

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

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

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

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

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

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

Jump Multiset

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

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

Jump Multiset

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

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

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

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

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

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 $l = (0, 0, 0, 0)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1$$
$$Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task2, \xi_2 r_2'}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

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

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

$$
\begin{aligned}
Proc_0 \underset{\{task1\}}{\bowtie} Res_0 &\xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie} Res_1 \\
Proc_1 \underset{\{task1\}}{\bowtie} Res_0 &\xrightarrow{task2, \xi_2 r_2'}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \\
Proc_0 \underset{\{task1\}}{\bowtie} Res_1 &\xrightarrow{reset, \xi_4 r_4}_* Proc_0 \underset{\{task1\}}{\bowtie} Res_0
\end{aligned}
$$

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

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

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

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

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

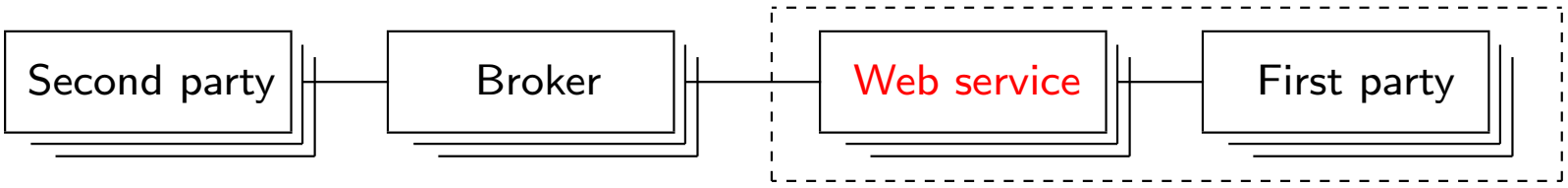
- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the 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.

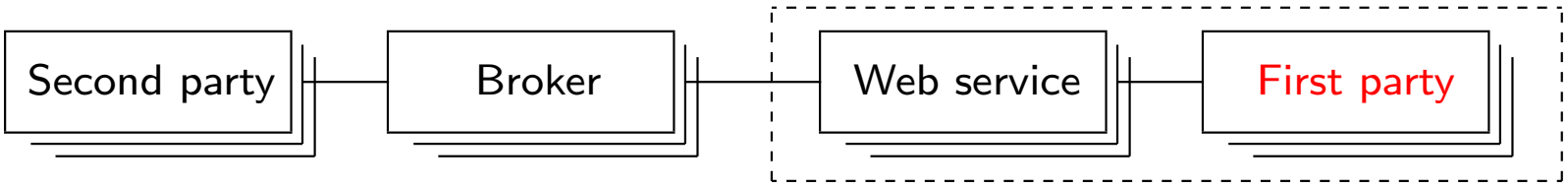
Outline

Example: Secure Web Service use



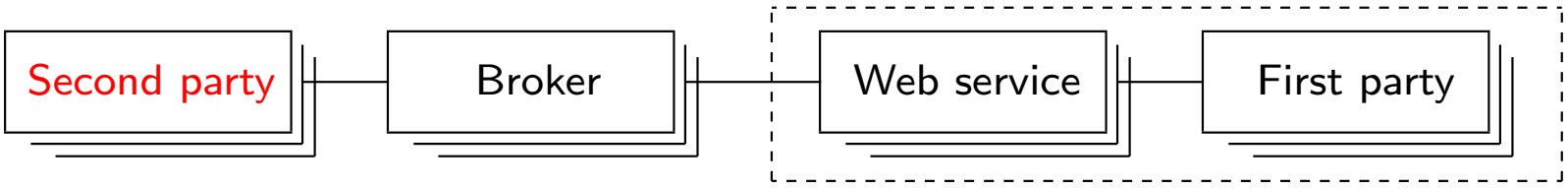
- The example which we consider is a Web service which has two types of clients:

Example: Secure Web Service use



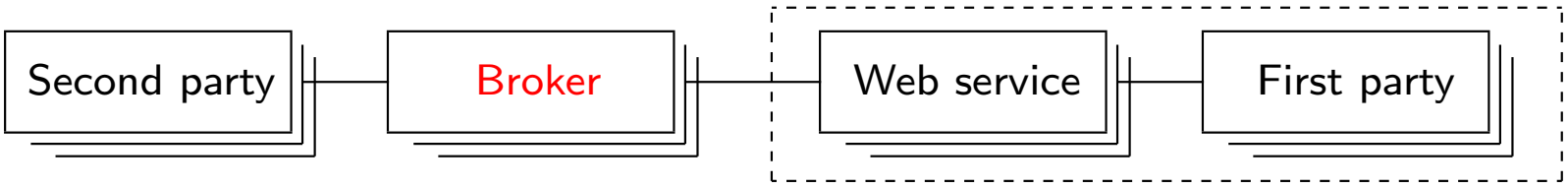
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and

Example: Secure Web Service use



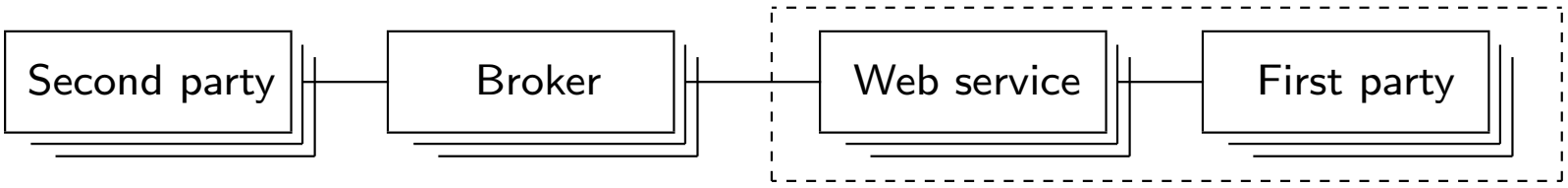
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.

Example: Secure Web Service use



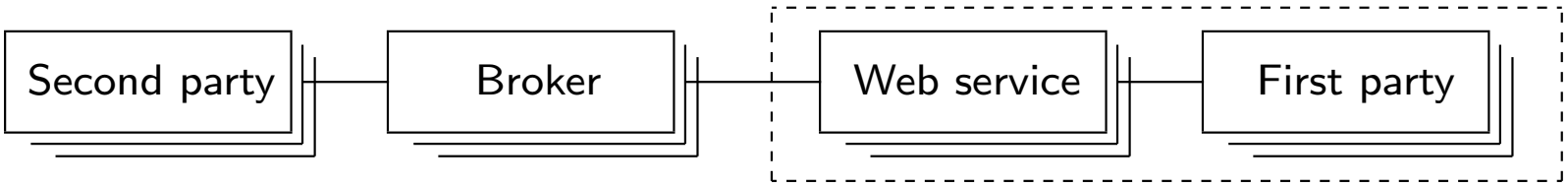
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

Scalability and replication



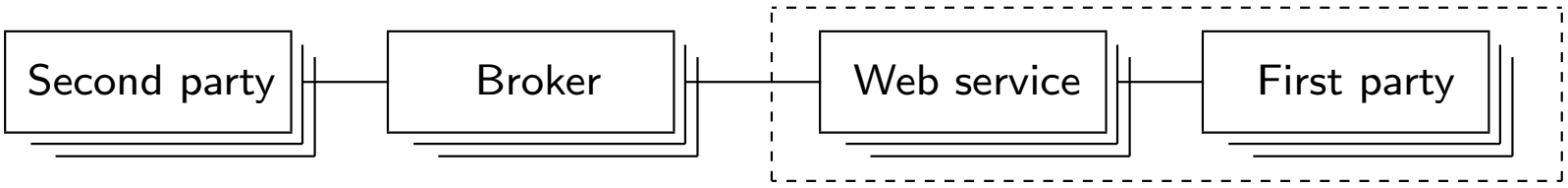
- To ensure scalability the Web service is replicated across multiple hosts.

Scalability and replication



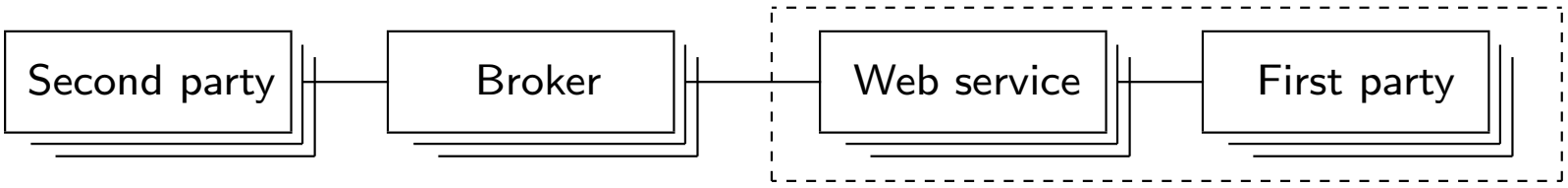
- To ensure scalability the Web service is replicated across multiple hosts.
- Multiple brokers are available.

Scalability and replication



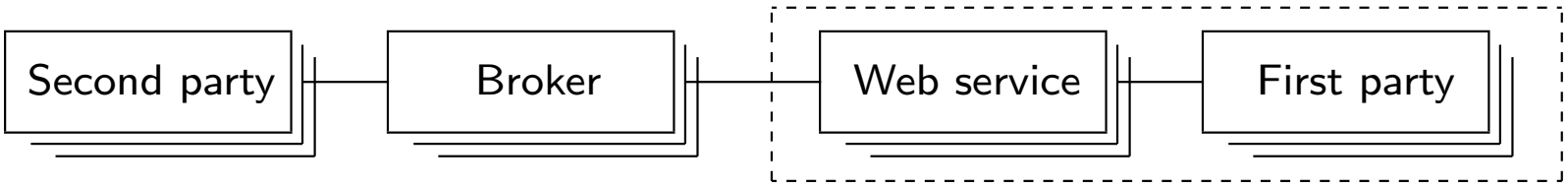
- To ensure scalability the Web service is replicated across multiple hosts.
- Multiple brokers are available.
- There are numerous first party clients behind the firewall using the service via remote method invocations across the secure intranet.

Scalability and replication



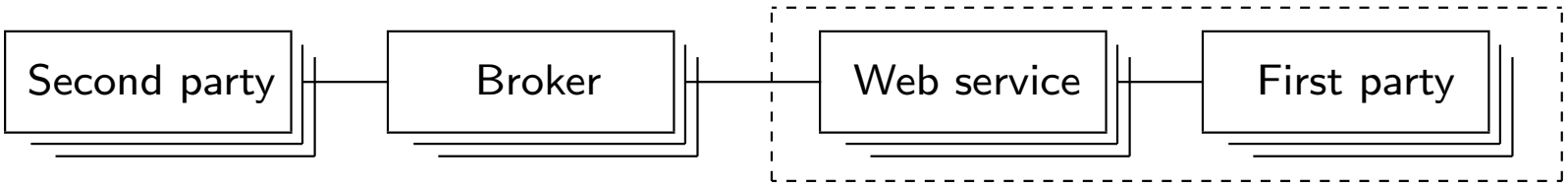
- To ensure scalability the Web service is replicated across multiple hosts.
- Multiple brokers are available.
- There are numerous first party clients behind the firewall using the service via remote method invocations across the secure intranet.
- There are numerous second party clients outside the firewall.

Security and use of encryption



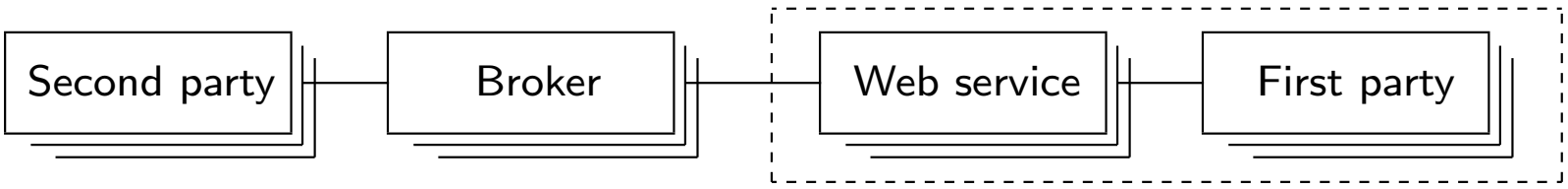
- Second party clients need to use encryption to ensure authenticity and confidentiality. First party clients do not.

Security and use of encryption



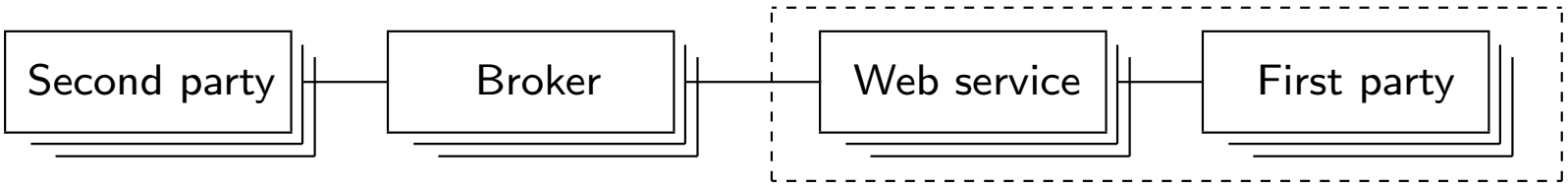
- Second party clients need to use encryption to ensure authenticity and confidentiality. First party clients do not.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.

Security and use of encryption



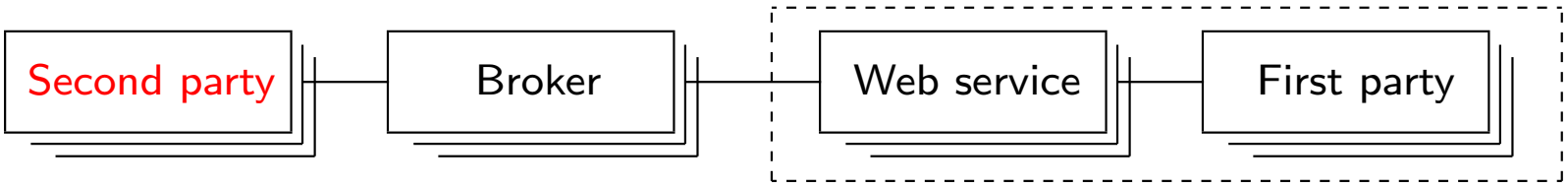
- Second party clients need to use encryption to ensure authenticity and confidentiality. First party clients do not.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.
  - When processing a request from a second party client brokers decrypt the request before re-encrypting it for the Web service.

Security and use of encryption



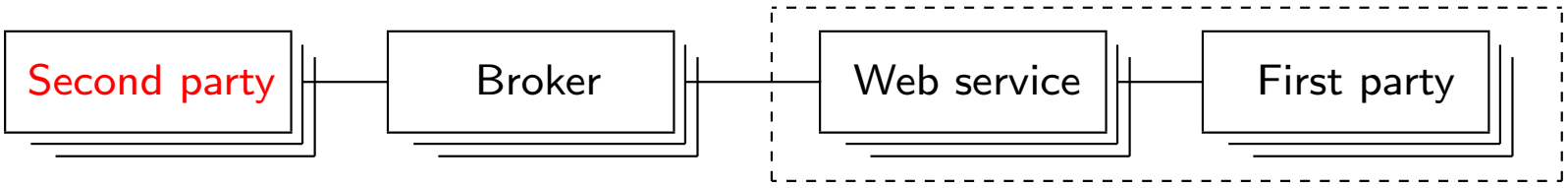
- Second party clients need to use encryption to ensure authenticity and confidentiality. First party clients do not.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.
  - When processing a request from a second party client brokers decrypt the request before re-encrypting it for the Web service.
  - When the response to a request is returned to the broker it decrypts the response before re-encrypting it for the client.

PEPA model: Second party clients



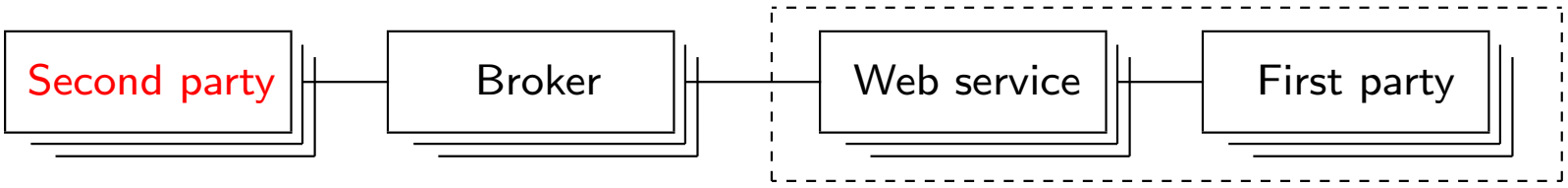
- A second party client composes service requests, encrypts these and sends them to its broker.

PEPA model: Second party clients



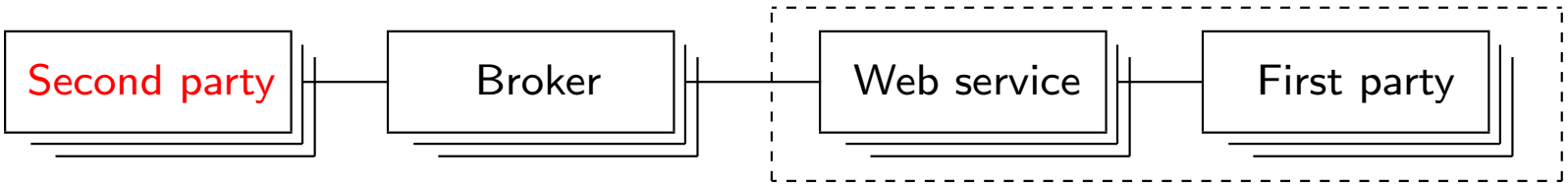
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.

PEPA model: Second party clients



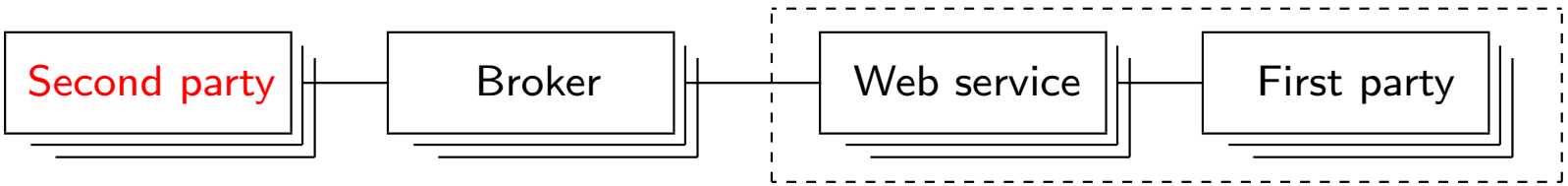
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- The rate at which the first three activities happen is under the control of the client.

PEPA model: Second party clients



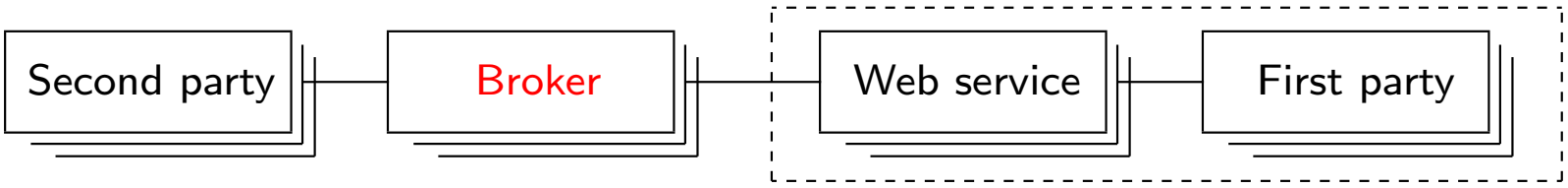
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- The rate at which the first three activities happen is under the control of the client.
- The rate at which responses are produced is determined by the interaction of the broker and the service endpoint.

PEPA model: Second party clients



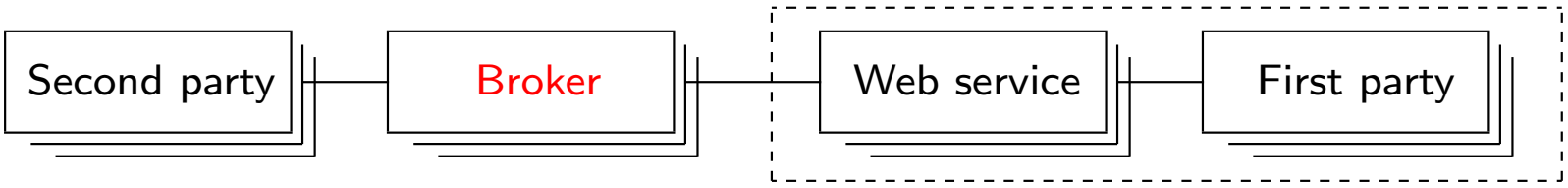
$$
\begin{aligned}
SPC_{idle} &\stackrel{def}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{def}{=} (encrypt_b, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{def}{=} (request_b, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{def}{=} (response_b, \top).SPC_{dec} \\
SPC_{dec} &\stackrel{def}{=} (decrypt_b, r_{sp\_decb}).SPC_{idle}
\end{aligned}
$$

PEPA model: Brokers



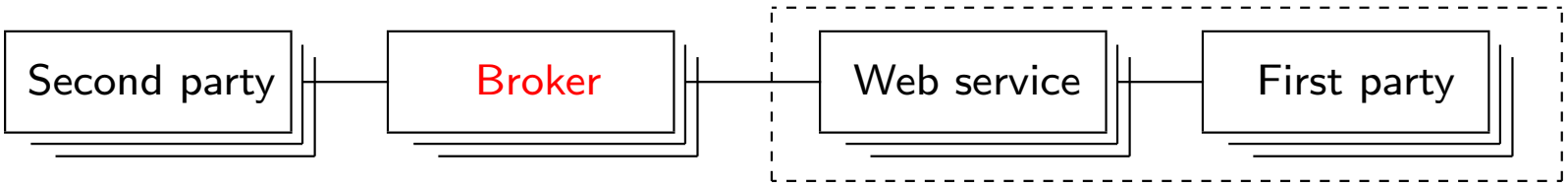
- The broker is inactive until it receives a request.

PEPA model: Brokers



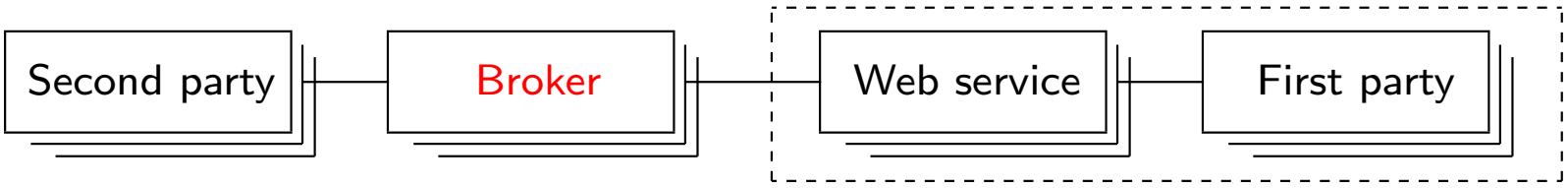
- The broker is inactive until it receives a request.
- It then decrypts the request before re-encrypting it for the Web service to ensure end-to-end security.

PEPA model: Brokers



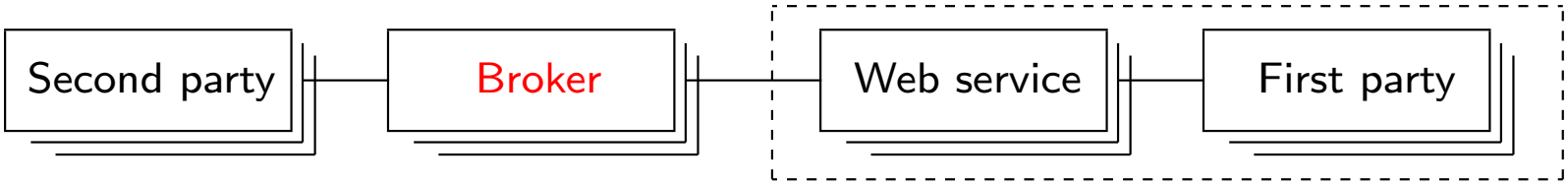
- The broker is inactive until it receives a request.
- It then decrypts the request before re-encrypting it for the Web service to ensure end-to-end security.
- It forwards the request to the Web service and then waits for a response.

PEPA model: Brokers



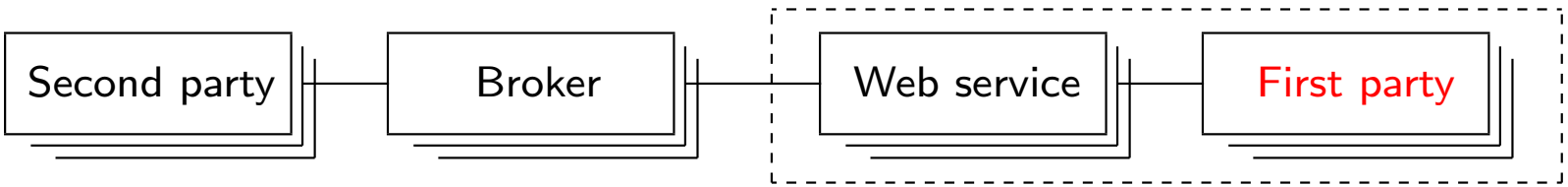
- The broker is inactive until it receives a request.
- It then decrypts the request before re-encrypting it for the Web service to ensure end-to-end security.
- It forwards the request to the Web service and then waits for a response.
- The corresponding decryption and re-encrytion are performed before returning the response to the client.

PEPA model: Brokers



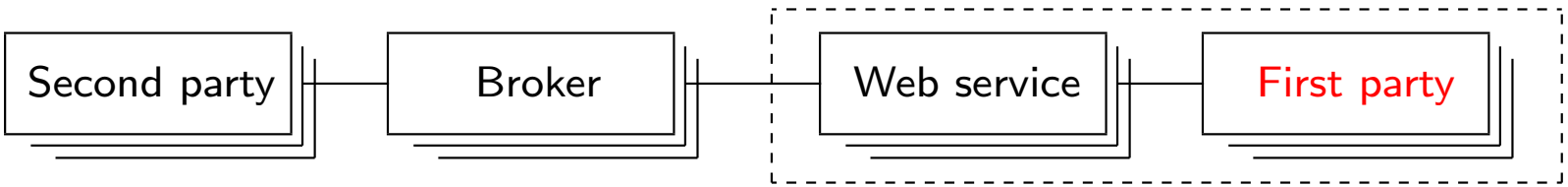
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=} (request_{ws}, r_{b\_req}).Broker_{waiting} \\
Broker_{waiting} &\stackrel{def}{=} (response_{ws}, \top).Broker_{dec\_resp} \\
Broker_{dec\_resp} &\stackrel{def}{=} (decrypt_{ws}, r_{b\_dec\_ws}).Broker_{enc\_resp} \\
Broker_{enc\_resp} &\stackrel{def}{=} (encrypt_{sp}, r_{b\_enc\_sp}).Broker_{replying} \\
Broker_{replying} &\stackrel{def}{=} (response_b, r_{b\_resp}).Broker_{idle}
\end{aligned}
$$

PEPA model: First party clients



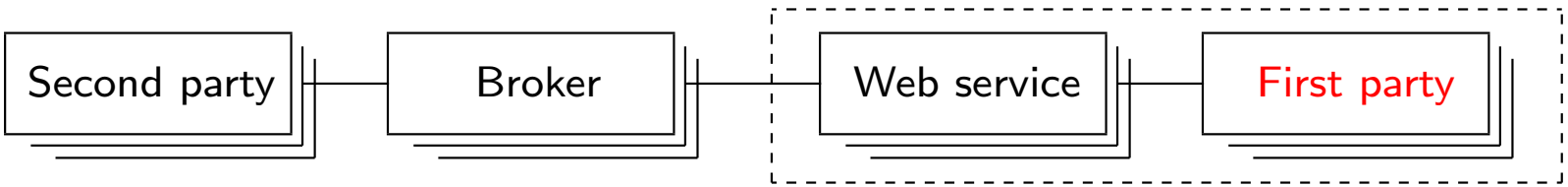
- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.

PEPA model: First party clients



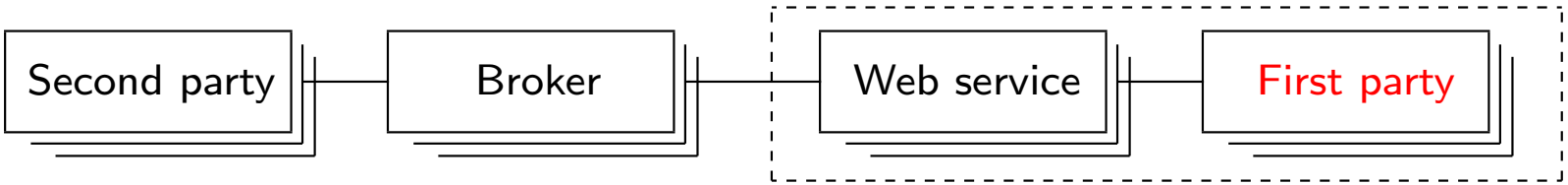
- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.
- Also the service may be invoked by a remote method invocation to the host machine instead of via HTTP.

PEPA model: First party clients



- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.
- Also the service may be invoked by a remote method invocation to the host machine instead of via HTTP.
- Thus the first party client experiences the Web service as a blocking remote method invocation.

PEPA model: First party clients

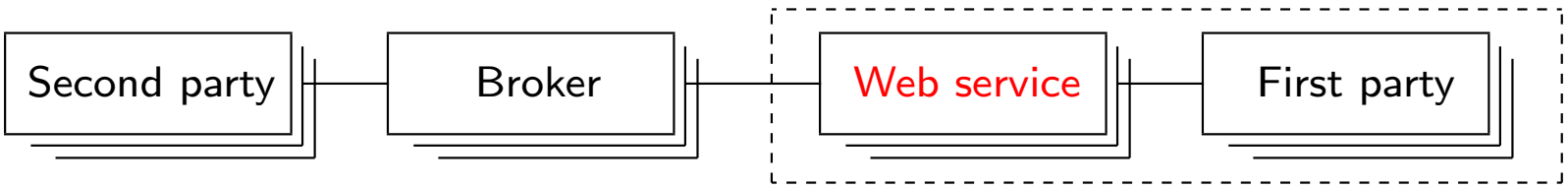


$$FPC_{idle} \stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling}$$

$$FPC_{calling} \stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked}$$

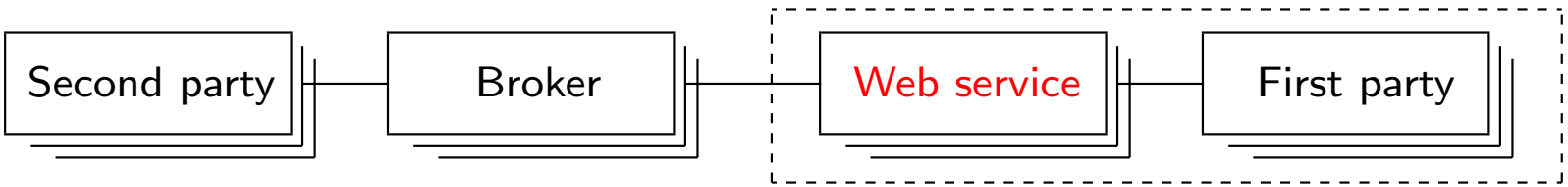
$$FPC_{blocked} \stackrel{def}{=} (result_{ws}, \top).FPC_{idle}$$

PEPA model: Web service



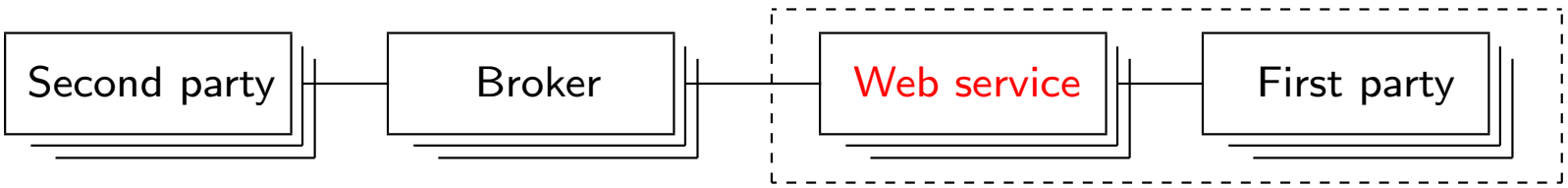
- There are two ways in which the service is executed, leading to a choice in the process algebra model taking the service process into one or other of its two modes of execution.

PEPA model: Web service



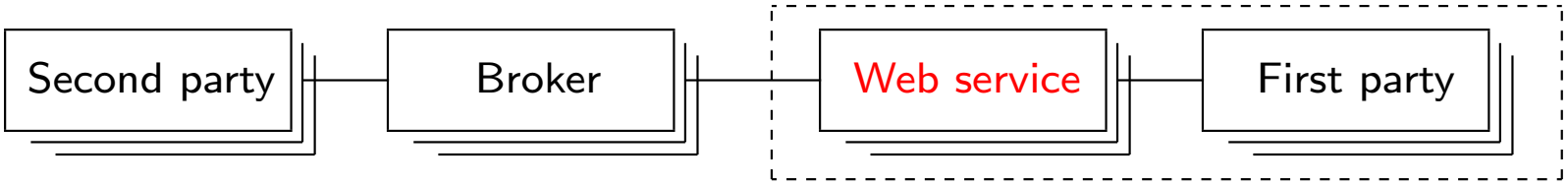
- There are two ways in which the service is executed, leading to a choice in the process algebra model taking the service process into one or other of its two modes of execution.
- In either case, the duration of the execution of the service itself is unchanged.

PEPA model: Web service



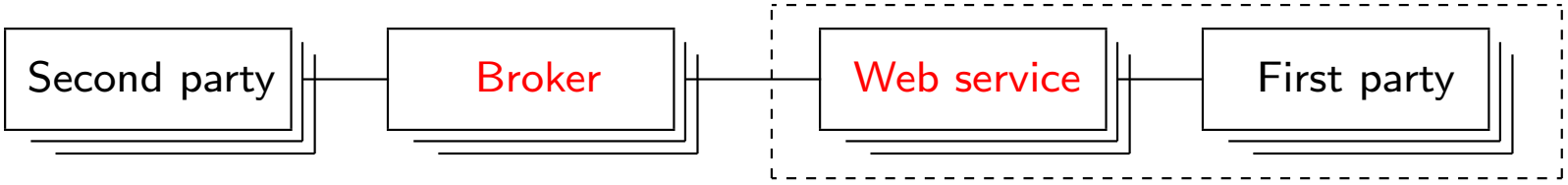
- There are two ways in which the service is executed, leading to a choice in the process algebra model taking the service process into one or other of its two modes of execution.
- In either case, the duration of the execution of the service itself is unchanged.
- The difference is only in whether encryption is needed and whether the result is delivered via HTTP or not.

PEPA model: Web service



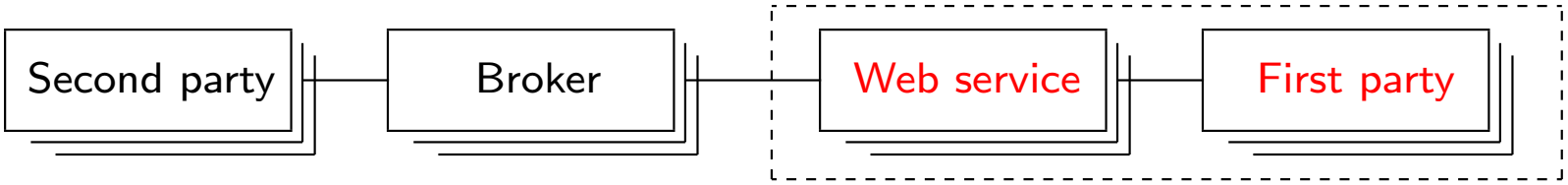
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



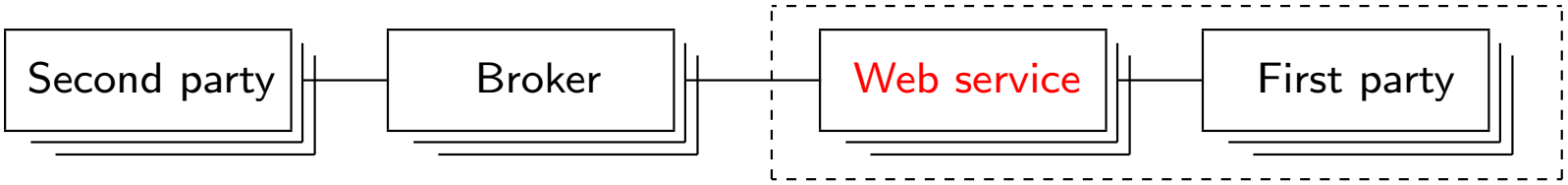
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: System composition

In the initial state of the system model we represent each of the four component types being initially in their idle state.

$$System \stackrel{def}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})$$

$$\begin{array}{lrcl} \text{where} & \mathcal{K} & = & \{ request_b, response_b \} \\ & \mathcal{L} & = & \{ request_{ws}, response_{ws} \} \\ & \mathcal{M} & = & \{ invoke_{ws}, result_{ws} \} \end{array}$$

PEPA model: System composition

In the initial state of the system model we represent each of the four component types being initially in their idle state.

$$System \stackrel{def}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})$$

$$\begin{array}{lrcl} \text{where} & \mathcal{K} & = & \{ request_b, response_b \} \\ & \mathcal{L} & = & \{ request_{ws}, response_{ws} \} \\ & \mathcal{M} & = & \{ invoke_{ws}, result_{ws} \} \end{array}$$

This model represents the smallest possible instance of the system, where there is one instance of each component type. We evaluate the system as the number of clients, brokers, and copies of the service increase.

Cost of analysis

- We compare fluid approximation, ODE-based, evaluation against other techniques which could be used to analyse the model.

Cost of analysis

- We compare fluid approximation, ODE-based, evaluation against other techniques which could be used to analyse the model.
- We compare against steady-state and transient analysis based on an explicit state representation,as implemented by the PRISM probabilistic model-checker (which provides PEPA as one of its input languages). We also compare against Monte Carlo Markov Chain simulation.

Comparison of analysis types

- We report only a single run of the simulation analysis. In practice, due to the stochastic nature of the analysis, this would need to be re-run multiple times to produce results comparable to the ODE-based analysis.

Comparison of analysis types

- We report only a single run of the simulation analysis. In practice, due to the stochastic nature of the analysis, this would need to be re-run multiple times to produce results comparable to the ODE-based analysis.
- Moreover, note that the number of ODEs is constant regardless of the number of components in the system, whilst the state space grows dramatically.

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77
1000	1000	1000	1000	–	–	–	–	–	5.44	2.77

Time series analysis via ODEs

- We now consider the results from our solution of the PEPA Web Service model as a system of ODEs with the number of clients of both kinds, brokers, and web service instances all 1000.

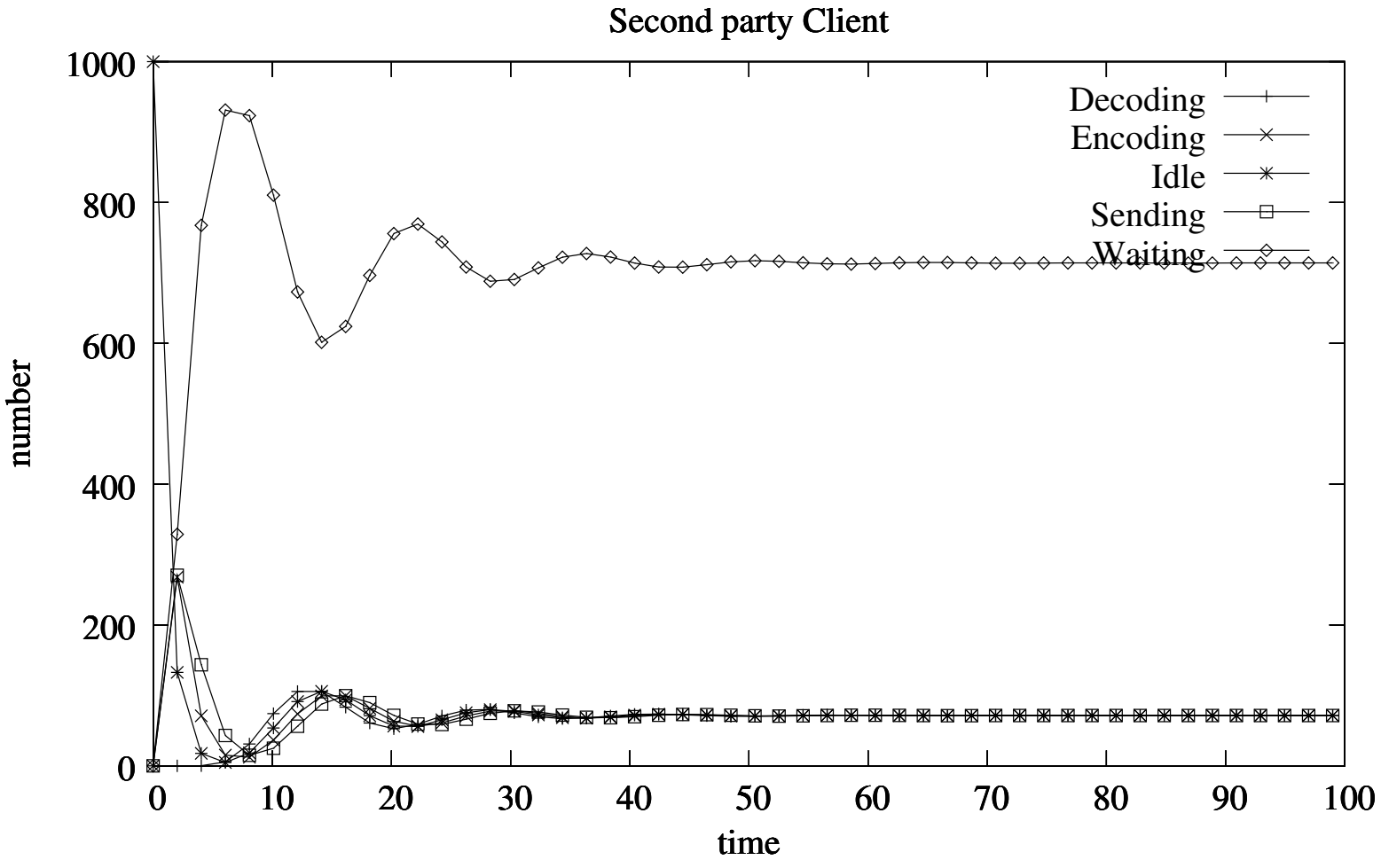
Time series analysis via ODEs

- We now consider the results from our solution of the PEPA Web Service model as a system of ODEs with the number of clients of both kinds, brokers, and web service instances all 1000.
- The results as presented from our ODE integrator are time-series plots of the number of each type of component behaviour as a function of time.

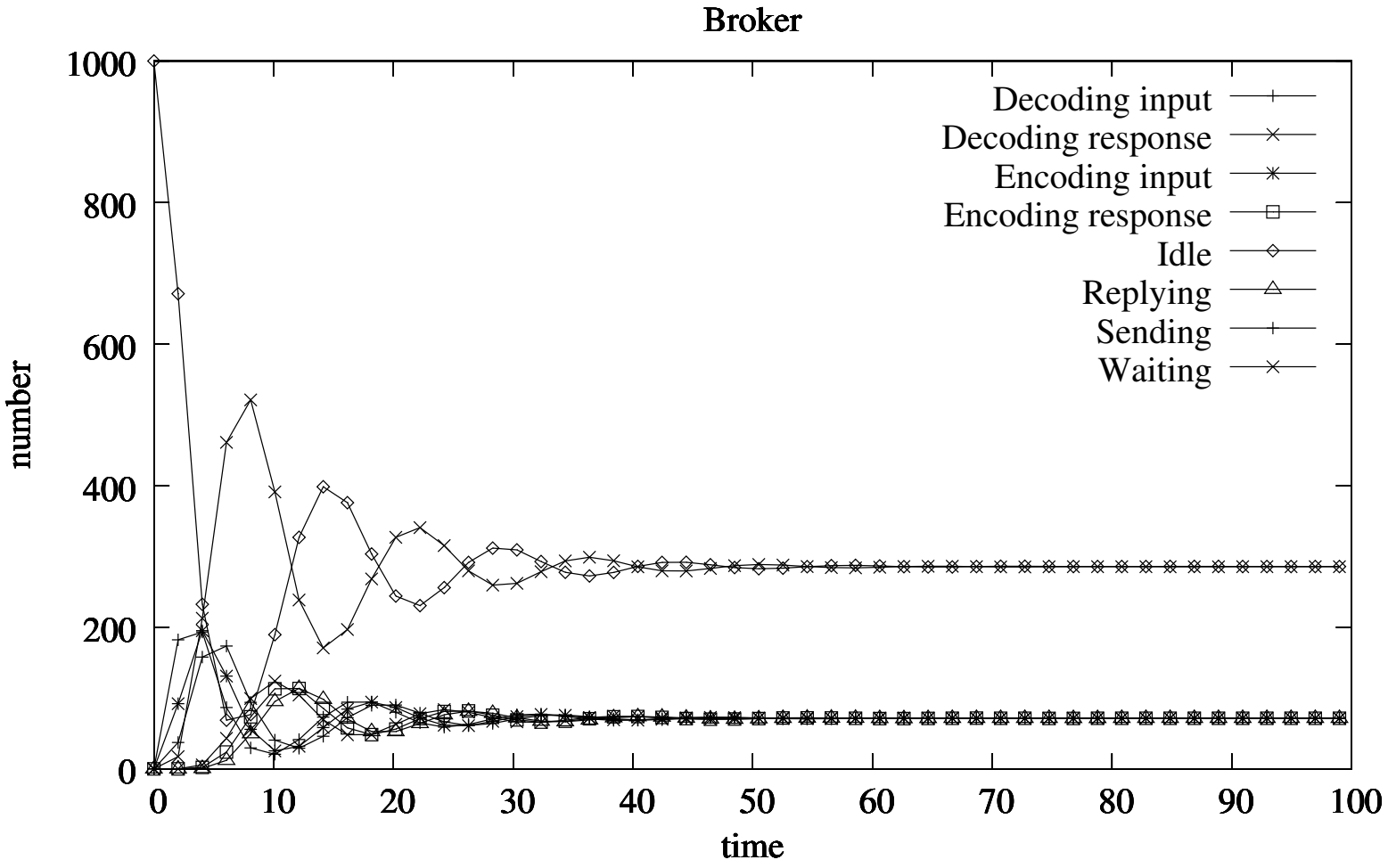
Time series analysis via ODEs

- We now consider the results from our solution of the PEPA Web Service model as a system of ODEs with the number of clients of both kinds, brokers, and web service instances all 1000.
- The results as presented from our ODE integrator are time-series plots of the number of each type of component behaviour as a function of time.
- We can observe an initial flurry of activity until the system stabilises into its steady-state equilibrium at time (around) $t = 50$.

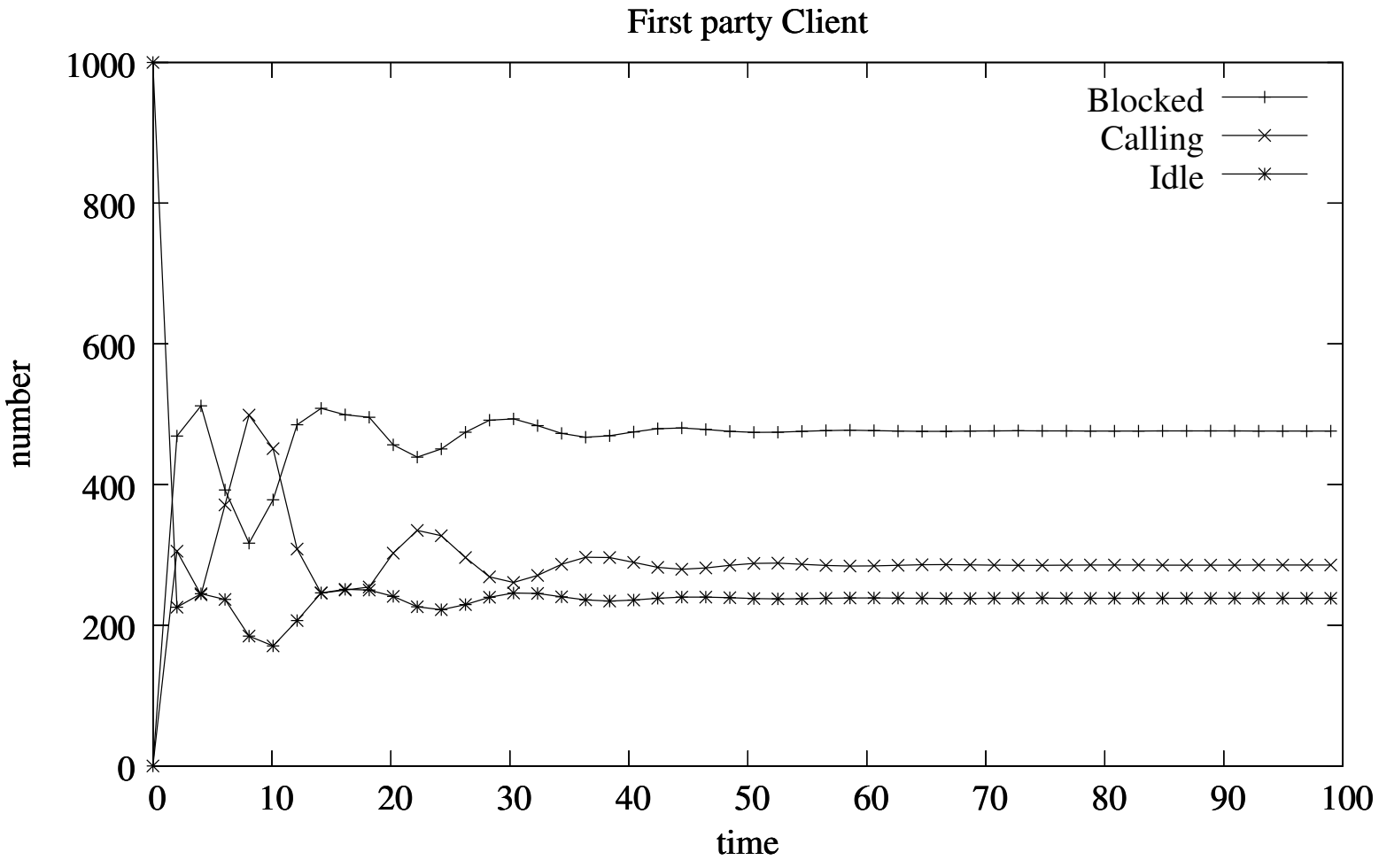
Second party clients



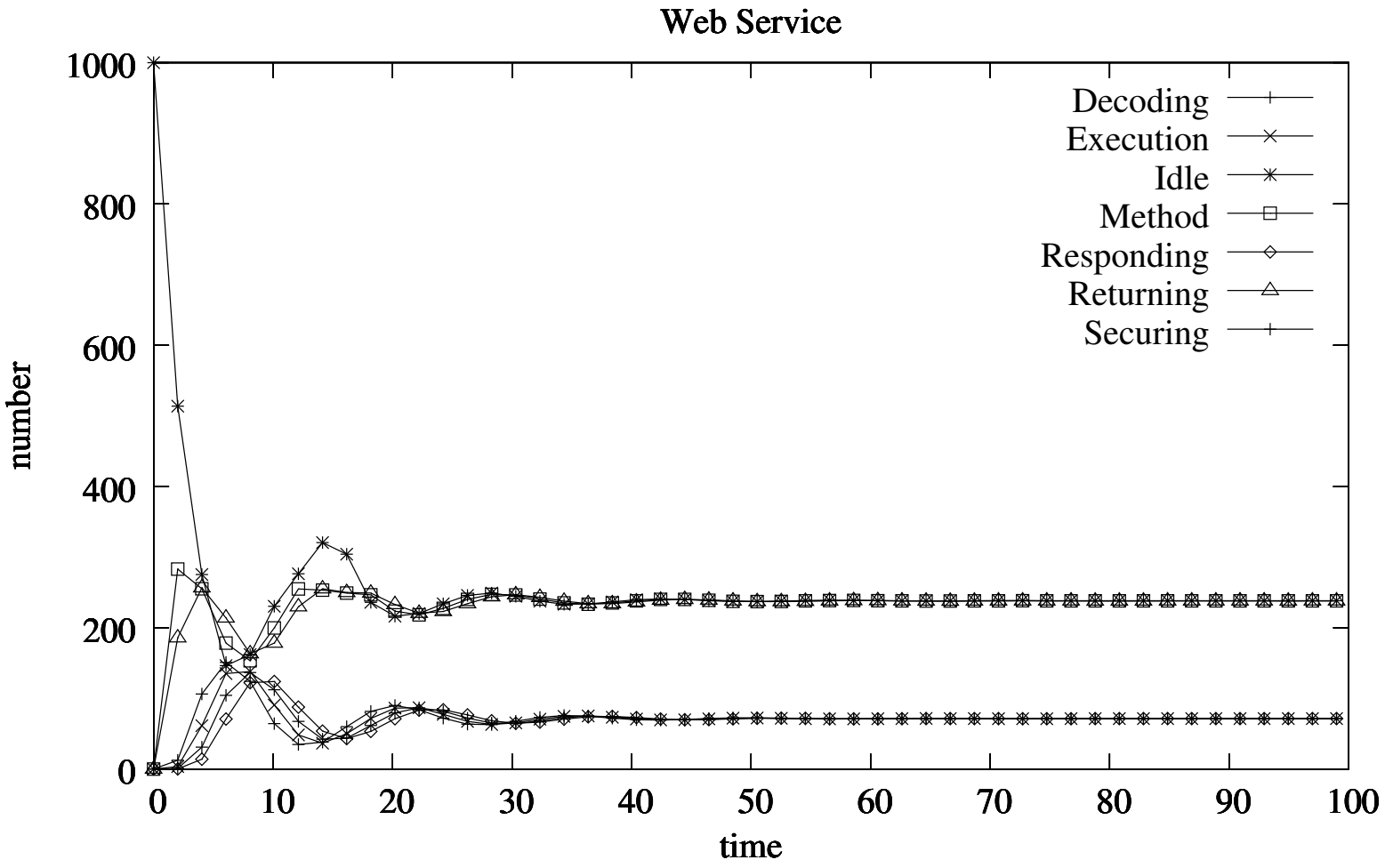
Brokers


Broker
number
time
0
200
400
600
800
1000
0
10
20
30
40
50
60
70
80
90
100
Decoding input
Decoding response
Encoding input
Encoding response
Idle
Replying
Sending
Waiting

First party clients



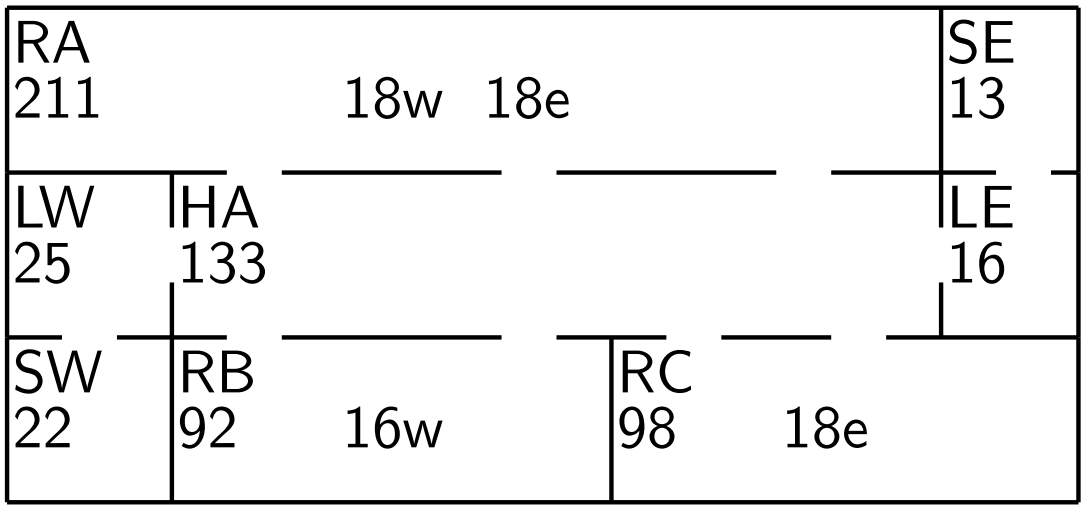
Web service


Web Service
number
time
0
200
400
600
800
1000
0
10
20
30
40
50
60
70
80
90
100
Decoding
Execution
Idle
Method
Responding
Returning
Securing

Designing for human crowd dynamics

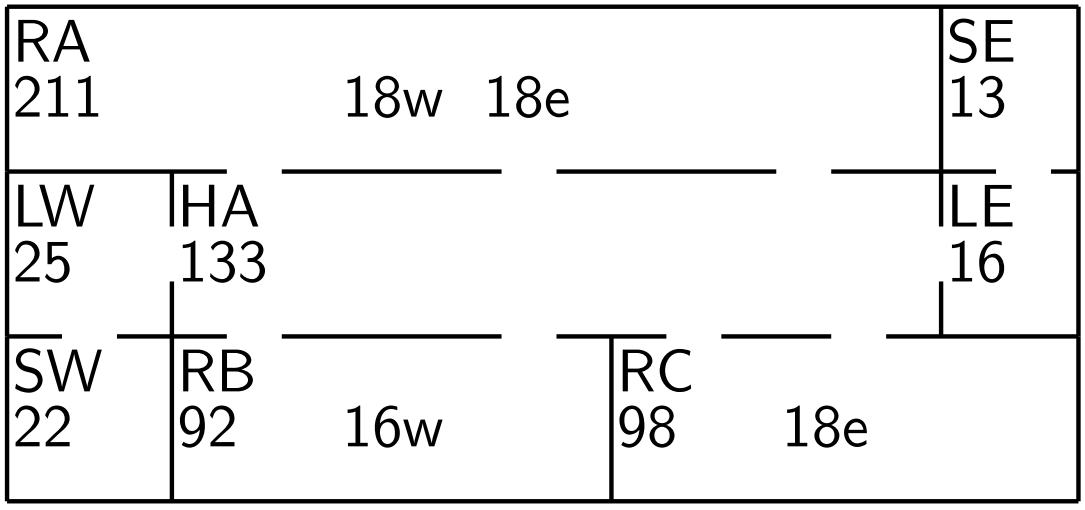
- Widespread take up of mobile and communicating computational devices is making ubiquitous systems a reality and creating new ways for us to interact with our environment.
- One application is to provide routing information to help people navigate through unfamiliar locations.
- In these cases the dynamic behaviour of the system as a whole is important to ensure the satisfaction of the users.
- Emergency egress can be regarded as a particular case, when the location may be familiar but circumstances may alter the usual topology and make efficient movement particularly important.

Example scenario



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

Example scenario



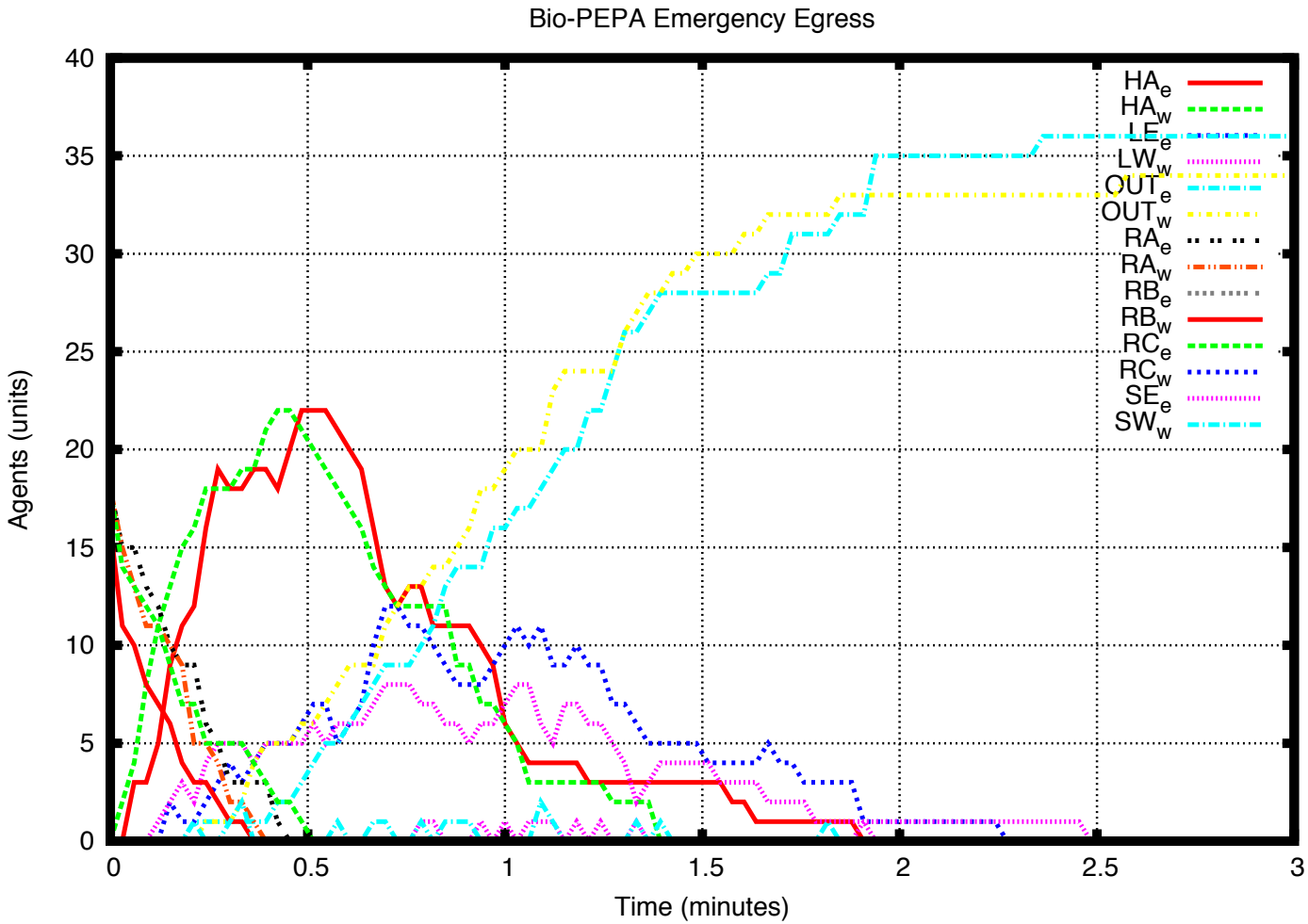
The layout of the building is described in terms of the arrangement of the rooms, hallways, landing and stairs. Each has a capacity and may have an initial occupancy.

Bio-PEPA components describe the behaviours of individuals, but also rooms and information dissemination.

Model specification

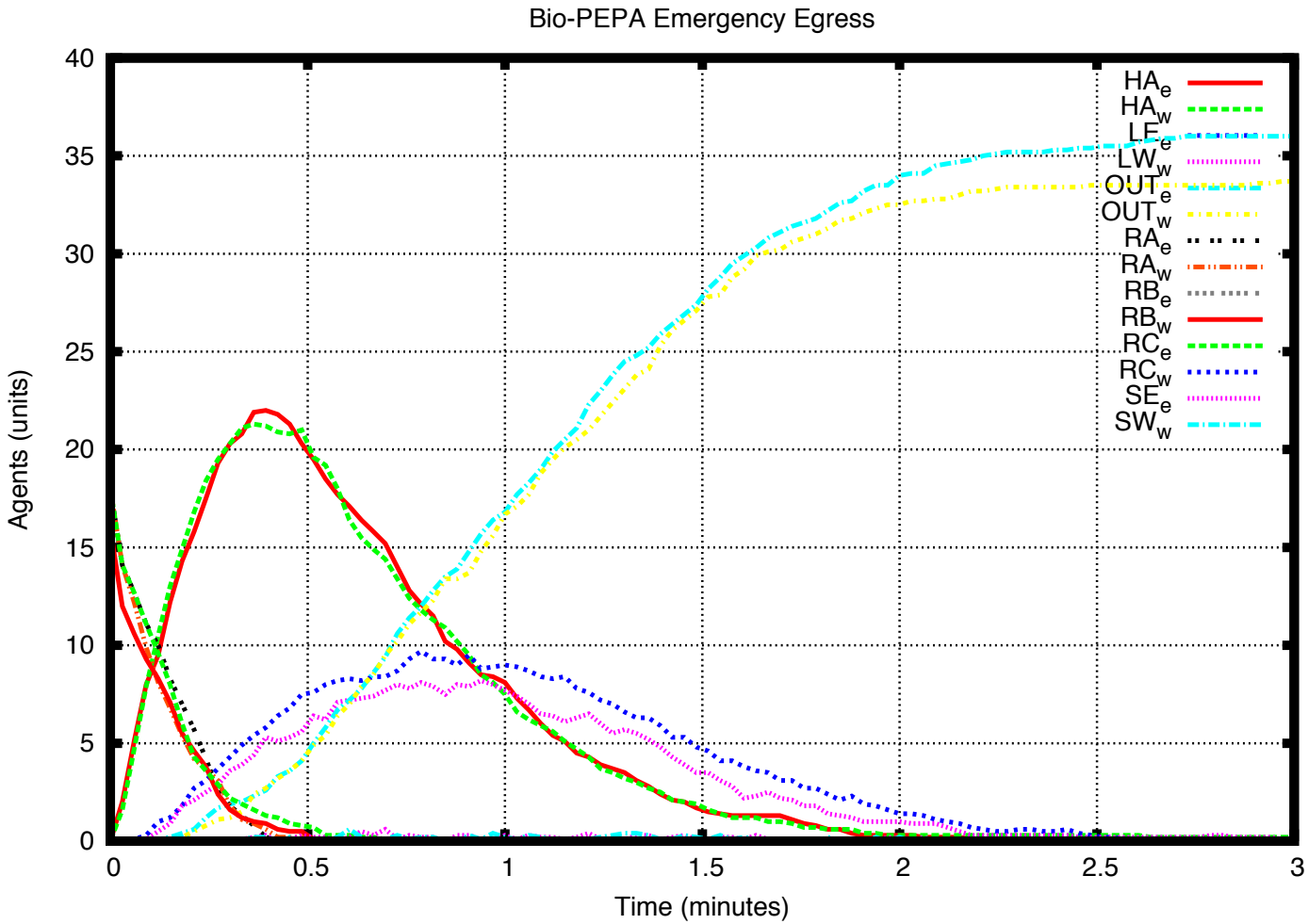
```
// BUILDING LAYOUT   (COMPARTMENTS)

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

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

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

// AGENT DEFINITIONS (SEQUENTIAL PROCESSES)

RA_e = (ra_e_in__d1_ra_ha, 1)     << RA_e@ra  +  ...
       (ra_e_out__d1_ra_ha, 1)    >> RA_e@ra  +  ...
RA_w =  ...
HA_e =  ...
...
```

Example results: room occupancy



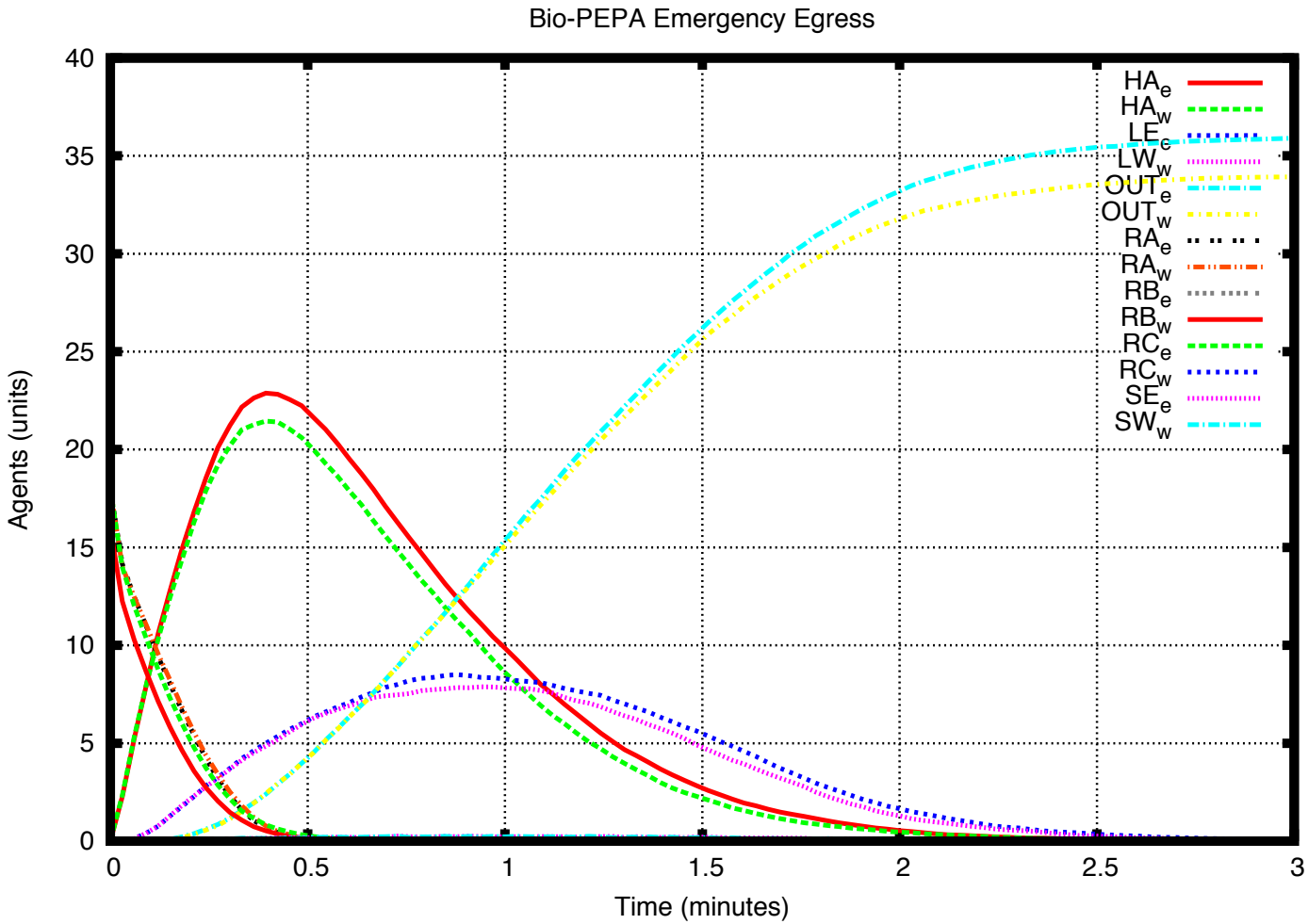
One stochastic simulation run

Example results: room occupancy



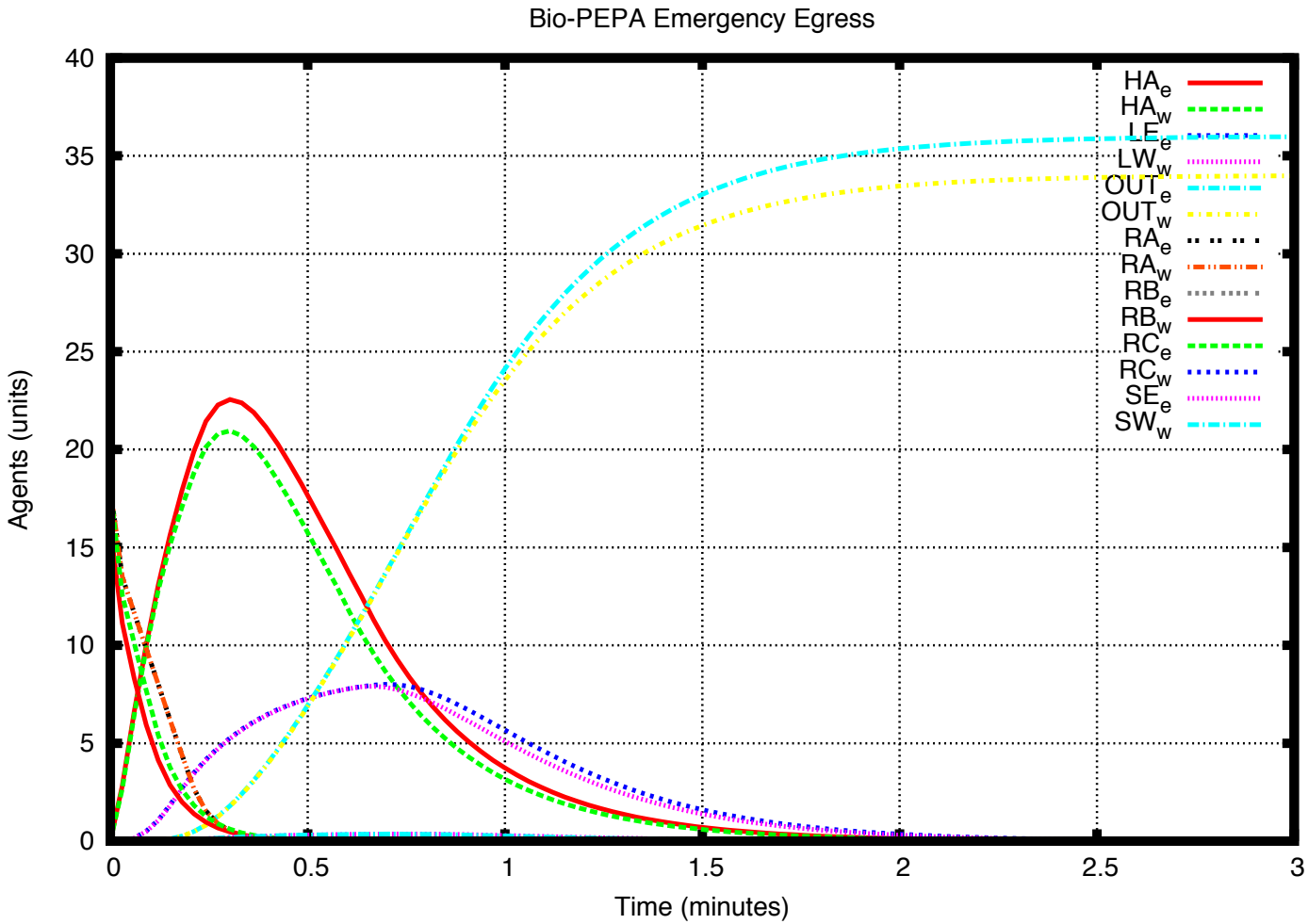
10 stochastic simulation runs

Example results: room occupancy



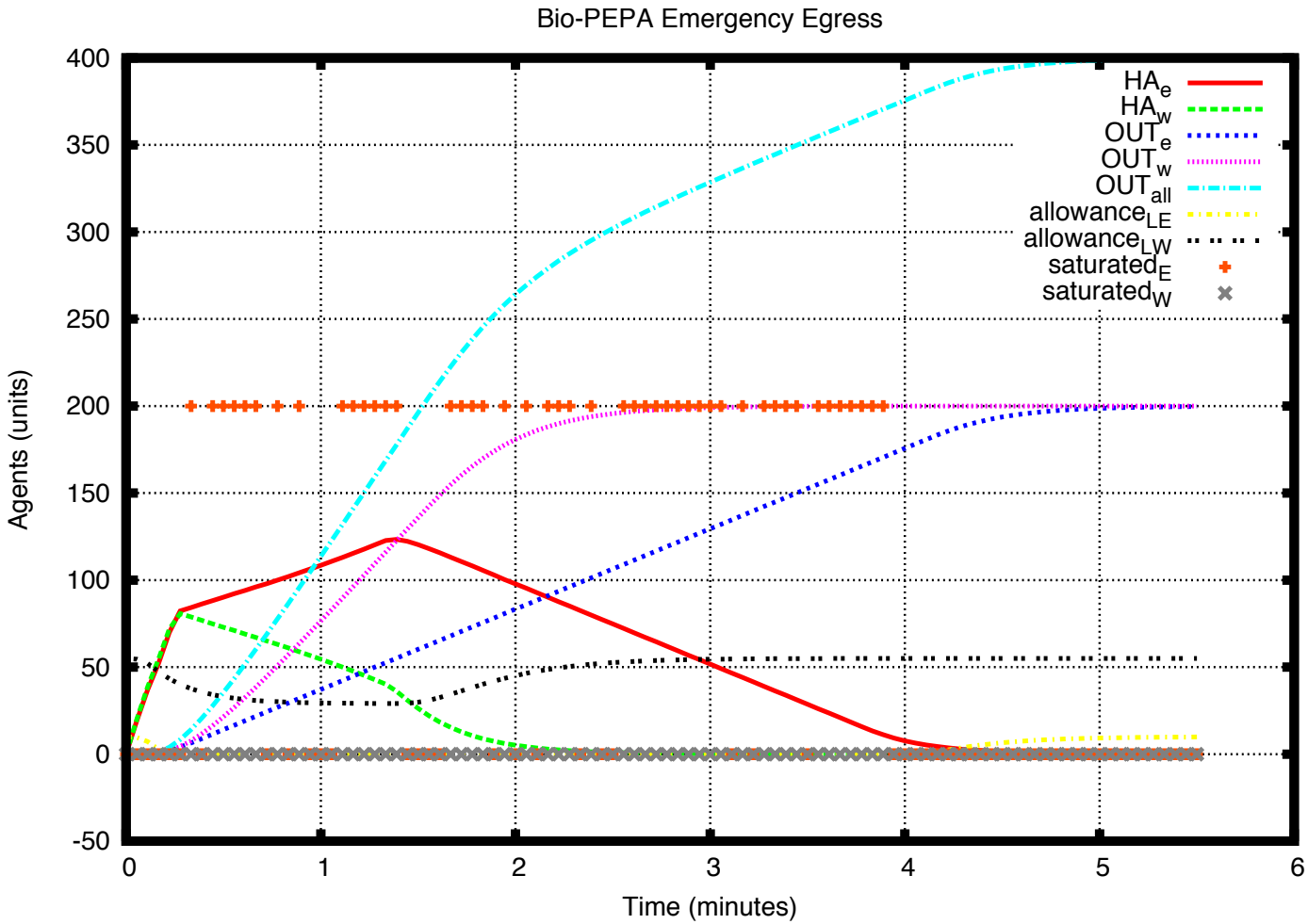
500 stochastic simulation runs

Example results: room occupancy



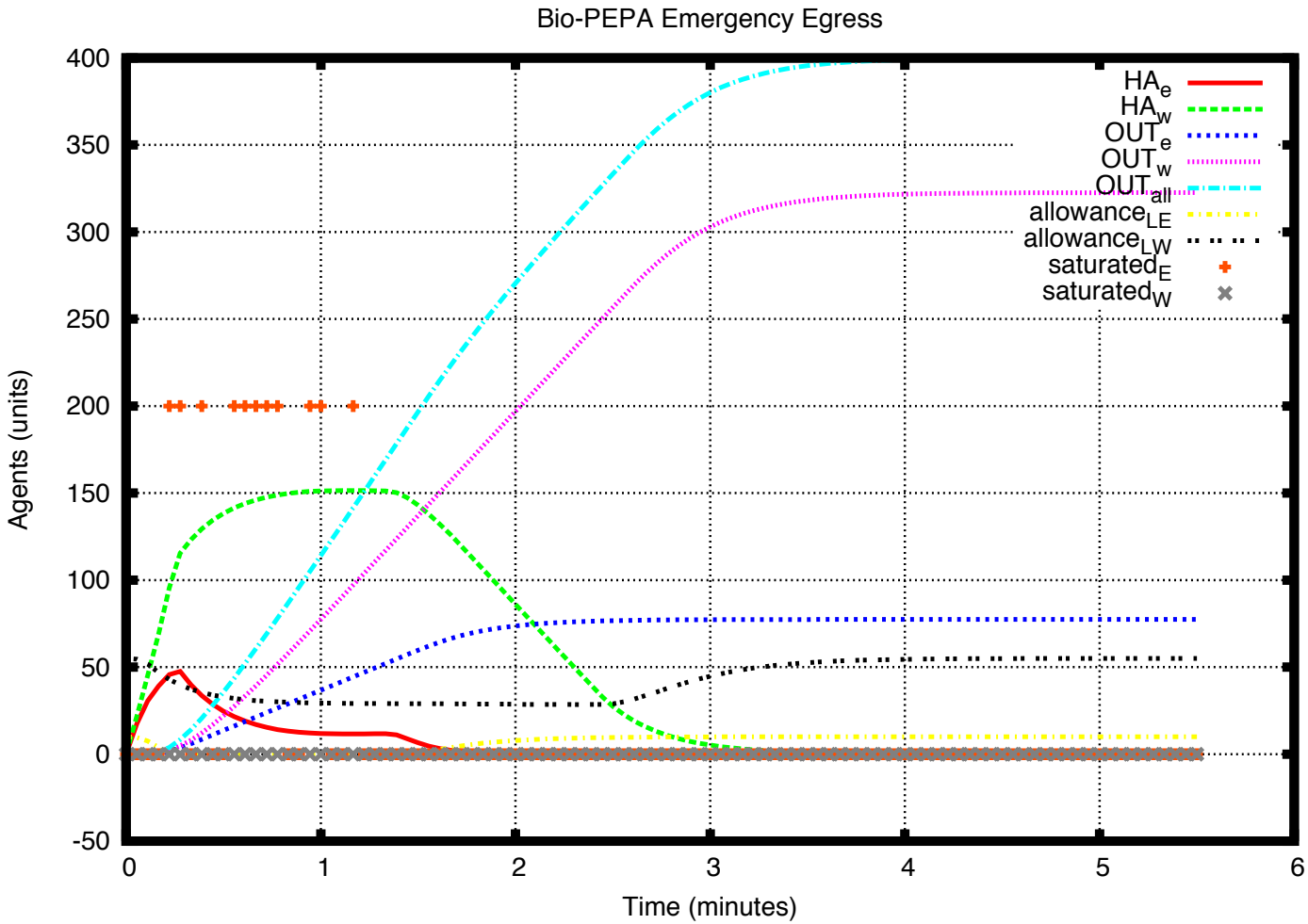
ODE numerical simulation

Example results: rerouting through mediation



Room occupancy over time without rerouting capability

Example results: rerouting through mediation



Room occupancy over time with rerouting capability

Outline

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 and Bio-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 and Bio-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 and Bio-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.

On-going work

- Time series plots counting the populations of components over time tell us a great deal about the dynamics of the system but are not necessary the information we require.

On-going work

- Time series plots counting the populations of components over time tell us a great deal about the dynamics of the system but are not necessary the information we require.
- Recent work has establish how performance measures such as throughput, and average response time can be derived from the ODE solutions.

On-going work

- Time series plots counting the populations of components over time tell us a great deal about the dynamics of the system but are not necessary the information we require.
- Recent work has establish how performance measures such as throughput, and average response time can be derived from the ODE solutions.
- On-going work is investigating the use of probes to query the model by adding components to the model whose sole purpose is to gather statistics.

On-going work

- Time series plots counting the populations of components over time tell us a great deal about the dynamics of the system but are not necessary the information we require.
- Recent work has establish how performance measures such as throughput, and average response time can be derived from the ODE solutions.
- On-going work is investigating the use of probes to query the model by adding components to the model whose sole purpose is to gather statistics.
- Using this technique we can now derive measures such as cumulative response time from the fluid approximation of the model.

Thanks!

Thanks!

Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Andrea Bracciali, Jeremy Bradley, Luca Bortolussi, Federica Ciocchetta, **Allan Clark**, Jie Ding, Adam Duguid, Vashti Galpin, **Stephen Gilmore**, Diego Latella, Mieke Massink, **Mirco Tribastone**, and others.

Thanks!

Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Andrea Bracciali, Jeremy Bradley, Luca Bortolussi, Federica Ciocchetta, **Allan Clark**, Jie Ding, Adam Duguid, Vashti Galpin, **Stephen Gilmore**, Diego Latella, Mieke Massink, **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!

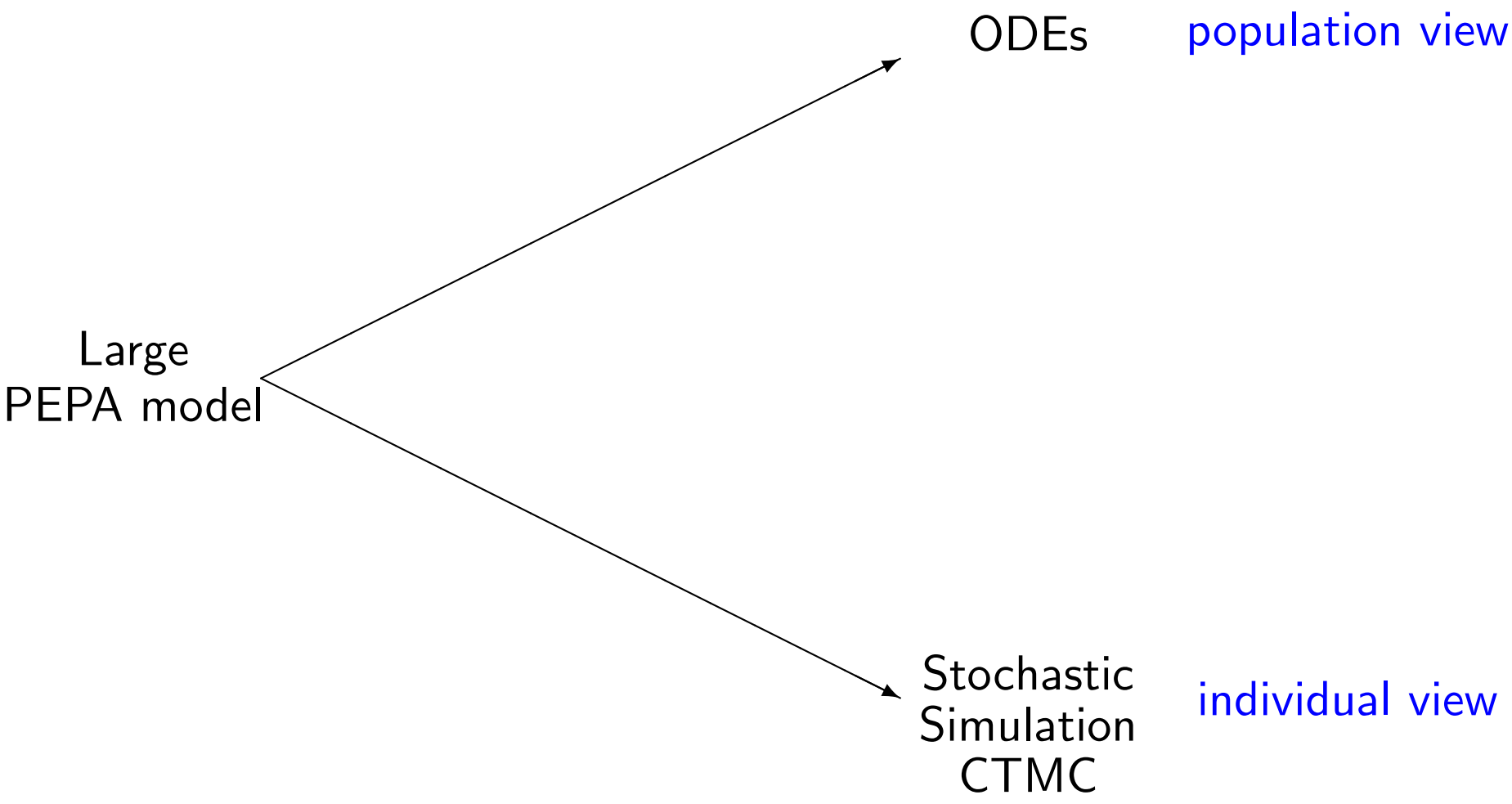
Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Andrea Bracciali, Jeremy Bradley, Luca Bortolussi, Federica Ciocchetta, **Allan Clark**, Jie Ding, Adam Duguid, Vashti Galpin, **Stephen Gilmore**, Diego Latella, Mieke Massink, **Mirco Tribastone**, and others.

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



Alternative Representations