# Formal languages for stochastic modelling

Jane Hillston

School of Informatics, University of Edinburgh

16th August 2017

# Outline

1. Introduction: Performance Modelling and Process Algebras
   - Performance Modelling
   - Stochastic Process Algebra
2. Tackling State Space Explosion
   - Lumpability and Bisimulation
   - Fluid Approximation
3. Beyond Performance Modelling

# The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov chains.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

# The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov chains.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

# The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov chains.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

# The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov chains.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

## The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov chains.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

# Outline

1. Introduction: Performance Modelling and Process Algebras
   - Performance Modelling
   - Stochastic Process Algebra
2. Tackling State Space Explosion
   - Lumpability and Bisimulation
   - Fluid Approximation
3. Beyond Performance Modelling

# Outline

1 Introduction: Performance Modelling and Process Algebras
- Performance Modelling
- Stochastic Process Algebra

2 Tackling State Space Explosion
- Lumpability and Bisimulation
- Fluid Approximation

3 Beyond Performance Modelling

# Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

# Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

# Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

# Does performance 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 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.

# Does performance 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 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.

# Does performance 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 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.

# Does performance 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 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.

# Continuous Time Markov Chains

There is a long association between queueing networks and continuous time Markov chains (CTMCs) more generally and performance modelling.

This dates back to Erlang's Loss Formula for the performance of telephone exchanges in the early 20th century.

In the 1960s and 1970s queueing networks were used extensively, but the advent of distributed systems in the 1980s meant that many systems no longer fit the assumptions of queueing networks.

Thus CTMC were used more directly, but the major challenge became one of state space explosion.

# Continuous Time Markov Chains

There is a long association between queueing networks and continuous time Markov chains (CTMCs) more generally and performance modelling.

This dates back to Erlang's Loss Formula for the performance of telephone exchanges in the early 20th century.

In the 1960s and 1970s queueing networks were used extensively, but the advent of distributed systems in the 1980s meant that many systems no longer fit the assumptions of queueing networks.

Thus CTMC were used more directly, but the major challenge became one of state space explosion.

# Continuous Time Markov Chains

There is a long association between queueing networks and continuous time Markov chains (CTMCs) more generally and performance modelling.

This dates back to Erlang's Loss Formula for the performance of telephone exchanges in the early 20th century.

In the 1960s and 1970s queueing networks were used extensively, but the advent of distributed systems in the 1980s meant that many systems no longer fit the assumptions of queueing networks.

Thus CTMC were used more directly, but the major challenge became one of state space explosion.

# Continuous Time Markov Chains

There is a long association between queueing networks and continuous time Markov chains (CTMCs) more generally and performance modelling.

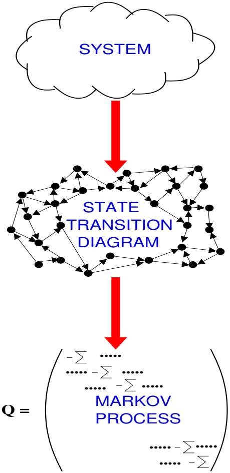
This dates back to Erlang's Loss Formula for the performance of telephone exchanges in the early 20th century.

In the 1960s and 1970s queueing networks were used extensively, but the advent of distributed systems in the 1980s meant that many systems no longer fit the assumptions of queueing networks.

Thus CTMC were used more directly, but the major challenge became one of state space explosion.

# Performance Modelling using CTMC

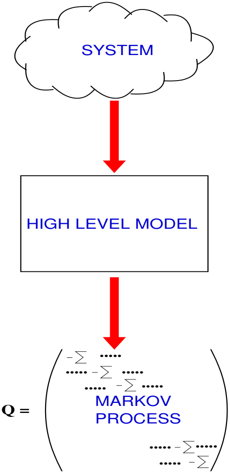
## Model Construction



# Performance Modelling using CTMC

## Model Construction

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



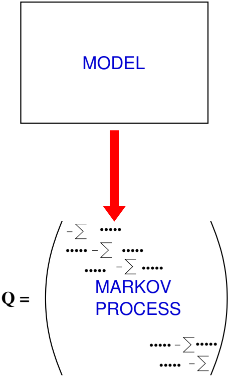
# Performance Modelling using CTMC

### Model Construction

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

### Model Manipulation

- model simplification
- model aggregation



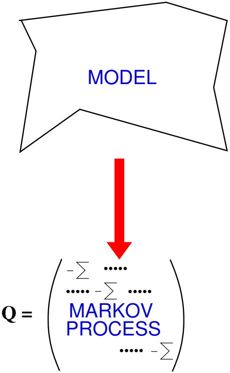
# Performance Modelling using CTMC

### Model Construction

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

### Model Manipulation

- model simplification
- model aggregation



# Performance Modelling using CTMC

## Model Construction

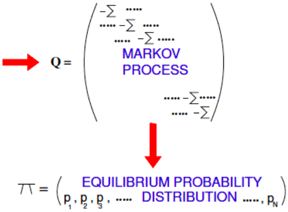
- describing the system using a high level modelling formalism
- generating the underlying CTMC

## Model Manipulation

- model simplification
- model aggregation

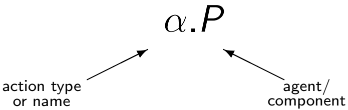
## Model Solution

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



# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

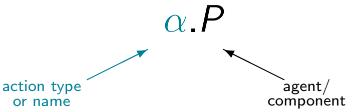
action type or name

agent/ component

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

# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

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

# Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

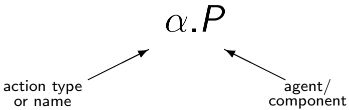
action type
or name

agent/
component

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

# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

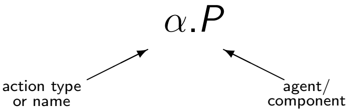
action type or name

agent/ component

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

# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

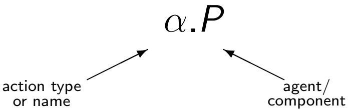
agent/ component

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

Process algebra model

# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

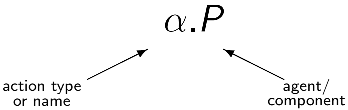
agent/ component

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

Process algebra model $\xrightarrow{\text{SOS rules}}$

# Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

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

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

# Process algebra operators

Process algebras generally have a number of different operators for combining actions and components, typically including:

- Prefix . – designated first action;
- Choice $+$ – selection between alternative components;
- Parallel composition $\parallel$ – components working concurrently;

These operators have rules associated with them such as

$$P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$$

and

$$P + P = P$$

# Bisimulation and congruence

Process algebras are usually equipped with an equivalence relation, termed a bisimulation, meaning that one component is equivalent to another if it can copy or simulate any action of the other component and vice versa.

Languages are designed so that these relations are designed so that these equivalence relations are congruences with respect to the operators of the language.

For example, if $P \sim Q$ then

- $\alpha.P \sim \alpha.Q$,
- $P + R \sim Q + R$ and
- $P \parallel R \sim Q \parallel R$.

# Bisimulation and congruence

Process algebras are usually equipped with an equivalence relation, termed a bisimulation, meaning that one component is equivalent to another if it can copy or simulate any action of the other component and vice versa.

Languages are designed so that these relations are designed so that these equivalence relations are congruences with respect to the operators of the language.

For example, if $P \sim Q$ then

- $\alpha.P \sim \alpha.Q$,
- $P + R \sim Q + R$ and
- $P \parallel R \sim Q \parallel R$.

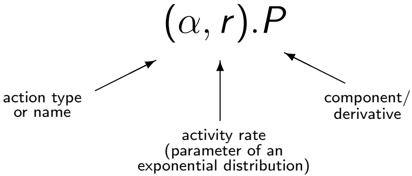
# Stochastic process algebras

## Stochastic process algebra

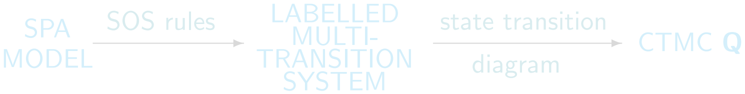
Process algebras where models are decorated with quantitative information used to generate a stochastic process are stochastic process algebras (SPA).

# Stochastic Process Algebra

- Models are constructed from components which engage in activities.

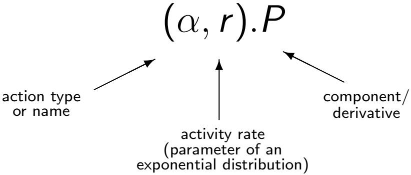

$(\alpha, r).P$

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



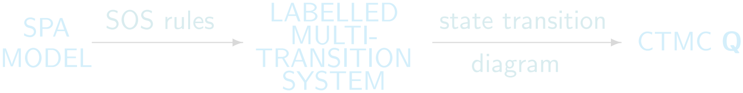
# Stochastic Process Algebra

- Models are constructed from components which engage in activities.



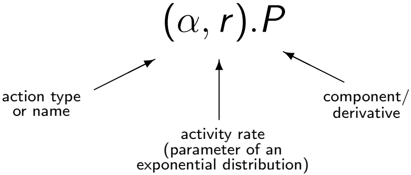
$$(\alpha, r).P$$

- 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.



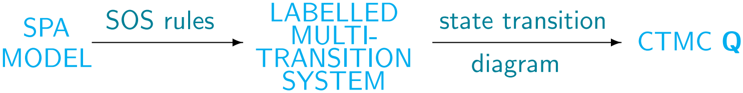
$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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



SPA MODEL

SOS rules

LABELLED MULTI-TRANSITION SYSTEM

state transition diagram

CTMC **Q**

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

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

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

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

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

# Performance Evaluation Process Algebra

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

| | |
|---|---|
| $(\alpha, r).P$ | 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 $P_1 \underset{\emptyset}{\bowtie} P_2$.

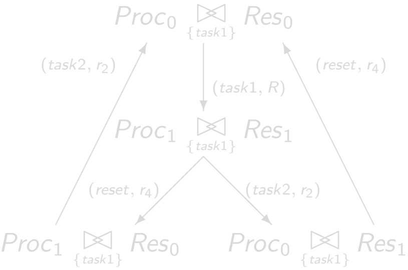
When working with large numbers of entities, we write $P[n]$ to denote an array of $n$ copies of $P$ executing in parallel.

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

# A simple example: processors and resources

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

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


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

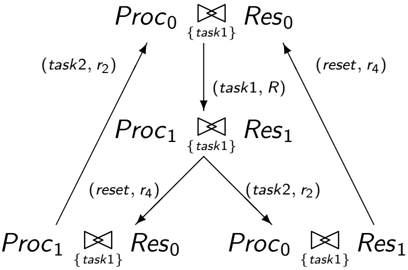
$$R = \min(r_1, r_3)$$

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

# A simple example: processors and resources

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

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



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

$(task2, r_2)$

$(task1, R)$

$(reset, r_4)$

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

$(reset, r_4)$

$(task2, r_2)$

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

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

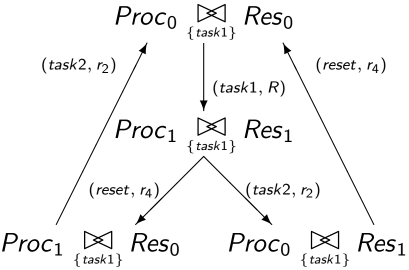
$$R = \min(r_1, r_3)$$

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

# A simple example: processors and resources

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

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


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

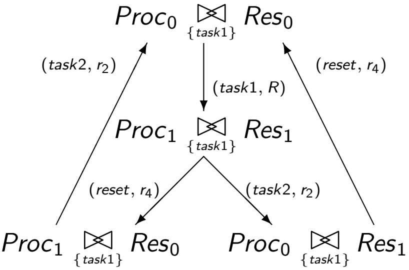
$R = \min(r_1, r_3)$

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

# A simple example: processors and resources

$$
\begin{aligned}
\mathit{Proc}_0 &\overset{\mathit{def}}{=} (\mathit{task}1, r_1).\mathit{Proc}_1\\
\mathit{Proc}_1 &\overset{\mathit{def}}{=} (\mathit{task}2, r_2).\mathit{Proc}_0\\
\mathit{Res}_0 &\overset{\mathit{def}}{=} (\mathit{task}1, r_3).\mathit{Res}_1\\
\mathit{Res}_1 &\overset{\mathit{def}}{=} (\mathit{reset}, r_4).\mathit{Res}_0
\end{aligned}
$$

$$\mathit{Proc}_0 \underset{\{\mathit{task}1\}}{\bowtie} \mathit{Res}_0$$


$\mathit{Proc}_0 \underset{\{\mathit{task}1\}}{\bowtie} \mathit{Res}_0$
$(\mathit{task}2, r_2)$
$(\mathit{task}1, R)$
$(\mathit{reset}, r_4)$
$\mathit{Proc}_1 \underset{\{\mathit{task}1\}}{\bowtie} \mathit{Res}_1$
$(\mathit{reset}, r_4)$
$(\mathit{task}2, r_2)$
$\mathit{Proc}_1 \underset{\{\mathit{task}1\}}{\bowtie} \mathit{Res}_0$
$\mathit{Proc}_0 \underset{\{\mathit{task}1\}}{\bowtie} \mathit{Res}_1$

$R = \min(r_1, r_3)$

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

# Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

# Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

# Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

# Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

# Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

# Benefits of process algebra

For example,

- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

# Benefits of process algebra

For example,

- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

# Benefits of process algebra

For example,

- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

# Outline

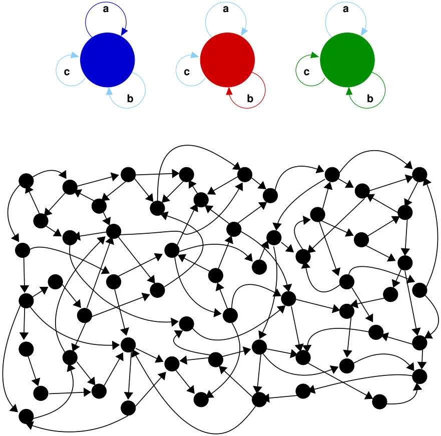
1 Introduction: Performance Modelling and Process Algebras
- Performance Modelling
- Stochastic Process Algebra

2 Tackling State Space Explosion
- Lumpability and Bisimulation
- Fluid Approximation

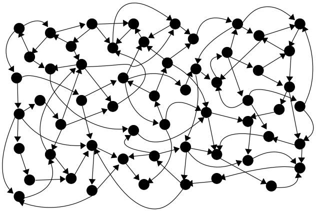
3 Beyond Performance Modelling

# Deriving performance measures

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.



# Deriving performance measures



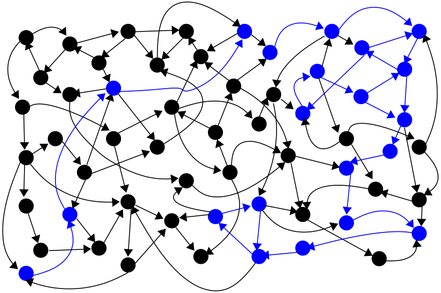
When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

# Deriving performance measures

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.

# Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

# Model Manipulation

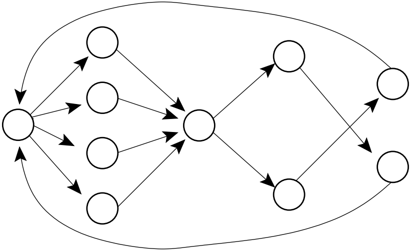
Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

# Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

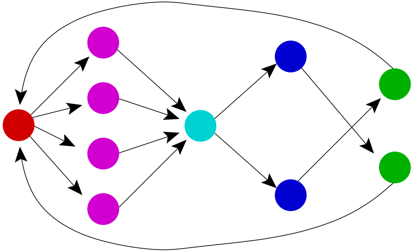
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



# Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

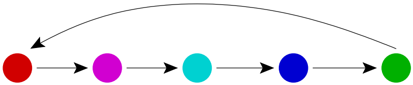
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



# Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



# Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to aggregate a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates.
- However checking the conditions typically involves constructing the complete Markov chain first.

J.Kemeny and J.Snell. *Finite Markov Chains.* Van Nostrand (1960)

# Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to aggregate a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates.
- However checking the conditions typically involves constructing the complete Markov chain first.

J.Kemeny and J.Snell. *Finite Markov Chains.* Van Nostrand (1960)

# Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to aggregate a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates.
- However checking the conditions typically involves constructing the complete Markov chain first.

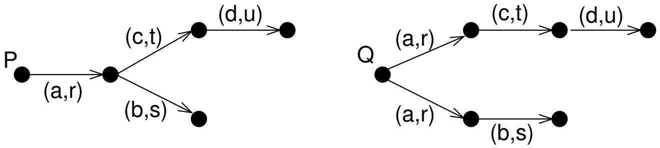
J.Kemeny and J.Snell. *Finite Markov Chains.* Van Nostrand (1960)

# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:

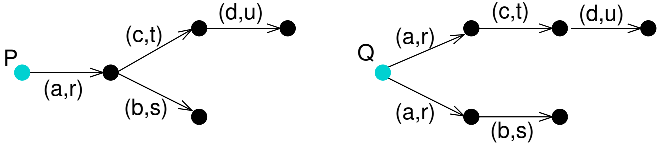
# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:



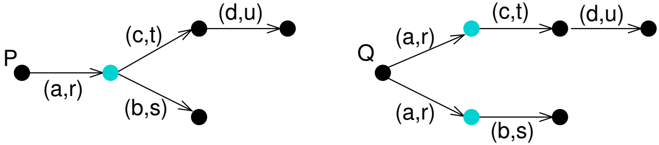
# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:



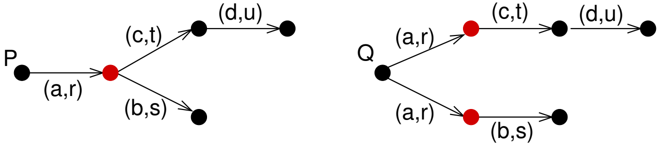
# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:



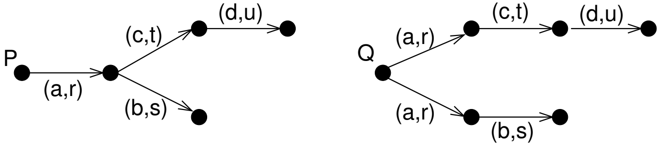
# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:



# Equivalence Relations

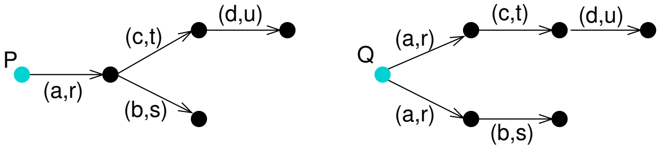
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

# Equivalence Relations

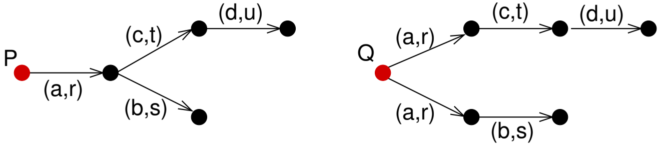
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

# Equivalence Relations

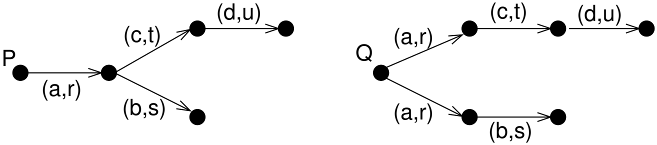
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:

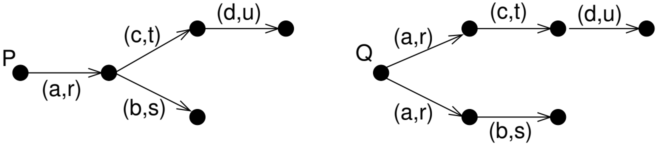


In PEPA observation is assumed to include the ability to record timing information over a number of runs.

The resulting equivalence relation is a bisimulation and coincides with the Markov process notion of lumpability.

# Equivalence Relations

In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

The resulting equivalence relation is a bisimulation and coincides with the Markov process notion of lumpability.

The formal definition means this can be applied automatically and compositionally.

J.Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press (1995)

# State space explosion

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

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

Use continuous state variables to approximate the discrete state space and ordinary differential equations to represent the evolution of those variables over time.

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.

# State space explosion

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

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

Use continuous state variables to approximate the discrete state space and ordinary differential equations to represent the evolution of those variables over time.

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.

# State space explosion

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

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

Use continuous state variables to approximate the discrete state space and ordinary differential equations to represent the evolution of those variables over time.

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.

# State space explosion

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

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

Use continuous state variables to approximate the discrete state space and ordinary differential equations to represent the evolution of those variables over time.

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

# Fluid approximation theorem

## Hypothesis

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

(Density dependent CTMCs are a special case.)

## Theorem

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

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

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

# Simple example revisited

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

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

# Simple example revisited

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

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

## CTMC interpretation

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

# Simple example revisited

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

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

- $task1$ 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

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

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

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

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

# Simple example revisited

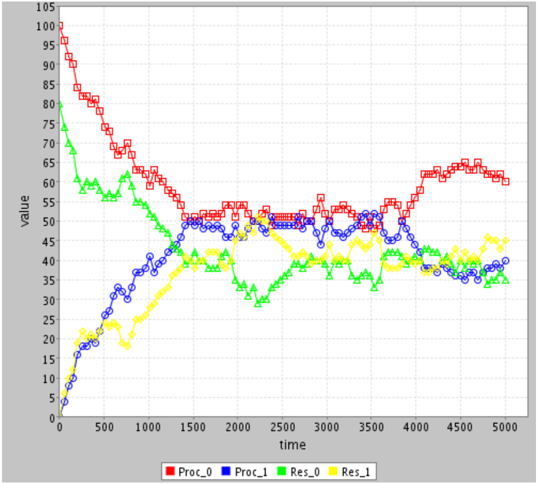
$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_1).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

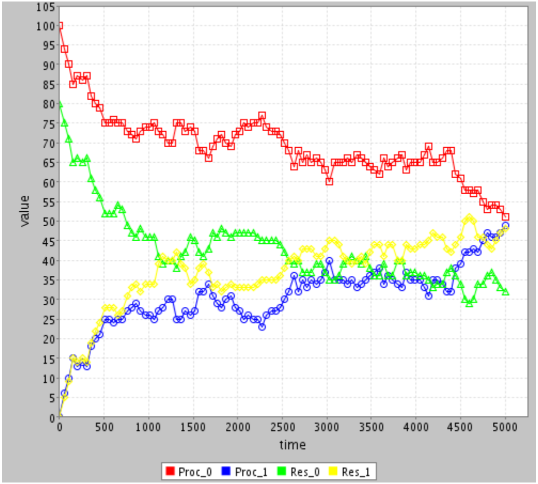
ODE interpretation

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

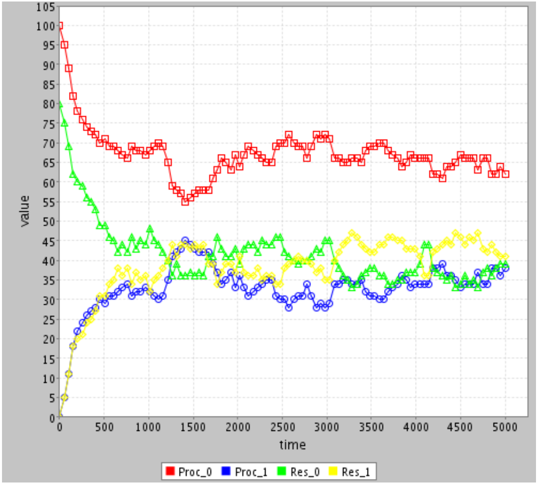
# 100 processors and 80 resources (simulation run A)



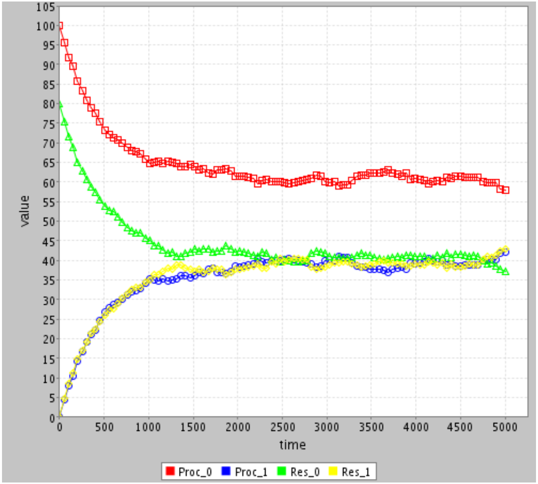
# 100 processors and 80 resources (simulation run B)



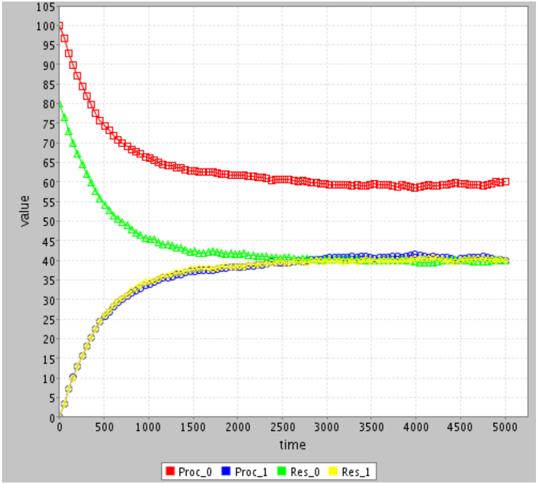
# 100 processors and 80 resources (simulation run C)



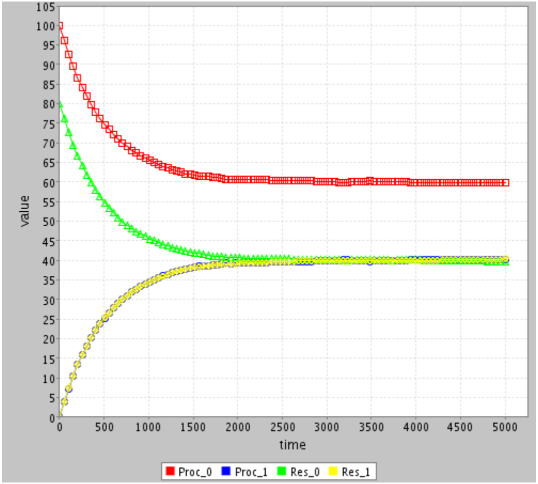
# 100 processors and 80 resources (average of 10 runs)



# 100 Processors and 80 resources (average of 100 runs)



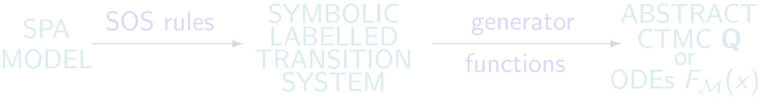
# 100 processors and 80 resources (average of 1000 runs)



# Deriving a Fluid Approximation of a PEPA model

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

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



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

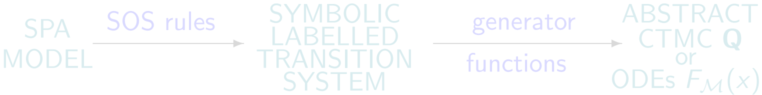
generator functions

ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

# Deriving a Fluid Approximation of a PEPA model

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

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



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

generator

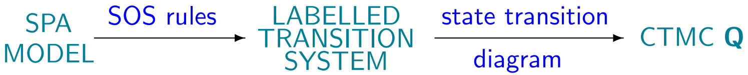
functions

ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

# Deriving a Fluid Approximation of a PEPA model

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

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



SPA MODEL

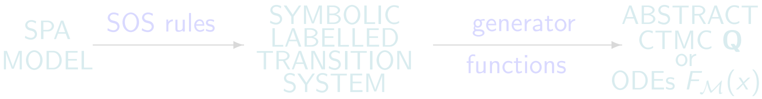
SOS rules

LABELLED TRANSITION SYSTEM

state transition

diagram

CTMC $\mathbf{Q}$



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

generator

functions

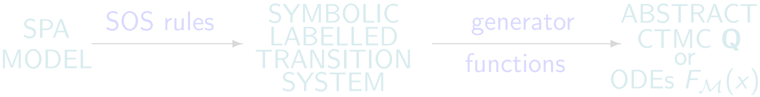
ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

# Deriving a Fluid Approximation of a PEPA model

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

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

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



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

generator functions

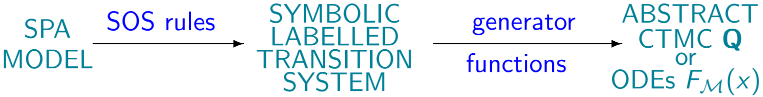
ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

# Deriving a Fluid Approximation of a PEPA model

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

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

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



SPA MODEL

SOS rules

SYMBOLIC LABELLED TRANSITION SYSTEM

generator functions

ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

# Fluid Structured Operational Semantics

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

1. Context Reduction: Remove excess components to find the abstract state representation $\xi$.
2. Jump Multiset: Collect the transitions $\alpha$ of the reduced context in terms of update vectors $l$.
3. Generating Functions: Calculate the rate of the transitions in terms of an arbitrary state of the CTMC, $f(\xi, l, \alpha)$.

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

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

# Fluid Structured Operational Semantics

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

1. Context Reduction: Remove excess components to find the abstract state representation $\xi$.
2. Jump Multiset: Collect the transitions $\alpha$ of the reduced context in terms of update vectors $l$.
3. Generating Functions: Calculate the rate of the transitions in terms of an arbitrary state of the CTMC, $f(\xi, l, \alpha)$.

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

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

# Fluid Structured Operational Semantics

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

1. Context Reduction: Remove excess components to find the abstract state representation $\xi$.
2. Jump Multiset: Collect the transitions $\alpha$ of the reduced context in terms of update vectors $l$.
3. Generating Functions: Calculate the rate of the transitions in terms of an arbitrary state of the CTMC, $f(\xi, l, \alpha)$.

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

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

# Fluid Structured Operational Semantics

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

1. Context Reduction: Remove excess components to find the abstract state representation $\xi$.
2. Jump Multiset: Collect the transitions $\alpha$ of the reduced context in terms of update vectors $l$.
3. Generating Functions: Calculate the rate of the transitions in terms of an arbitrary state of the CTMC, $f(\xi, l, \alpha)$.

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

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

# Fluid Structured Operational Semantics

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

1. Context Reduction: Remove excess components to find the abstract state representation $\xi$.
2. Jump Multiset: Collect the transitions $\alpha$ of the reduced context in terms of update vectors $l$.
3. Generating Functions: Calculate the rate of the transitions in terms of an arbitrary state of the CTMC, $f(\xi, l, \alpha)$.

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

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

# Rate properties of PEPA models

## Density dependence of parametric transition rates

The transition rates scale in the same way as the population, i.e. if $P \xrightarrow{(\alpha, r(\xi))}_{*} Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

## Generating functions give rise to density dependent rates

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

## Lipschitz continuity of parametric apparent rates

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

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

# Rate properties of PEPA models

## Density dependence of parametric transition rates

The transition rates scale in the same way as the population, i.e. if $P \xrightarrow{(\alpha, r(\xi))}_{*} Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

## Generating functions give rise to density dependent rates

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

## Lipschitz continuity of parametric apparent rates

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

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

# Rate properties of PEPA models

### Density dependence of parametric transition rates

The transition rates scale in the same way as the population, i.e. if $P \xrightarrow{(\alpha, r(\xi))}_{*} Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

### Generating functions give rise to density dependent rates

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

### Lipschitz continuity of parametric apparent rates

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

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

# Kurtz's Theorem

## Kurtz's Theorem for PEPA

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

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

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

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

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

# Kurtz's Theorem

## Kurtz's Theorem for PEPA

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

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

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

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

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

# Outline

1. Introduction: Performance Modelling and Process Algebras
   - Performance Modelling
   - Stochastic Process Algebra
2. Tackling State Space Explosion
   - Lumpability and Bisimulation
   - Fluid Approximation
3. Beyond Performance Modelling

# Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems, spatial temporal systems and ecological processes.

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

# Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems, spatial temporal systems and ecological processes.

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

# Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems, spatial temporal systems and ecological processes.

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

# Molecular processes as concurrent computations

| **Concurrency** | **Molecular Biology** | **Metabolism** | **Signal Transduction** |
|---|---|---|---|
| Concurrent computational processes | Molecules | Enzymes and metabolites | Interacting proteins |
| Synchronous communication | Molecular interaction | Binding and catalysis | Binding and catalysis |
| Transition or mobility | Biochemical modification or relocation | Metabolite synthesis | Protein binding, modification or sequestration |

A. Regev and E. Shapiro *Cells as computation*, Nature 419, 2002.

# Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

# Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

# Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

# Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

# The abstraction

- Each species $i$ is described by a species component $C_i$
- Each reaction $j$ is associated with an action type $\alpha_j$ and its dynamics is described by a specific function $f_{\alpha_j}$

The species components (now quantified) are then composed together to describe the behaviour of the system.

# The abstraction

- Each species $i$ is described by a species component $C_i$
- Each reaction $j$ is associated with an action type $\alpha_j$ and its dynamics is described by a specific function $f_{\alpha_j}$

The species components (now quantified) are then composed together to describe the behaviour of the system.

# The abstraction

- Each species $i$ is described by a species component $C_i$
- Each reaction $j$ is associated with an action type $\alpha_j$ and its dynamics is described by a specific function $f_{\alpha_j}$

The species components (now quantified) are then composed together to describe the behaviour of the system.

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

## Sequential component (species component)

$$S ::= (\alpha, \kappa)\ \mathsf{op}\ S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

## Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action $\alpha_j$ is associated with a rate $f_{\alpha_j}$

The list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

# The semantics

The semantics is defined by two transition relations:

- First, a capability relation — is a transition possible?
- Second, a stochastic relation — gives rate of a transition, derived from the parameters of the model.

The labelled transition system generated by the stochastic relation formally defines the underlying CTMC.

F.Ciocchetta & J.Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. TCS 2009.

## The semantics

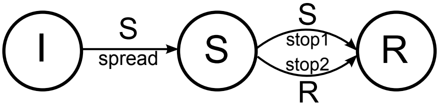
The semantics is defined by two transition relations:

- First, a capability relation — is a transition possible?
- Second, a stochastic relation — gives rate of a transition, derived from the parameters of the model.

The labelled transition system generated by the stochastic relation formally defines the underlying CTMC.

F.Ciocchetta & J.Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. TCS 2009.

# Example — in Bio-PEPA



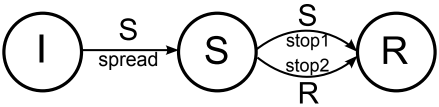
```
k_s = 0.5;
k_r = 0.1;

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\mathtt{I[10]} \bowtie_{*} \mathtt{S[5]} \bowtie_{*} \mathtt{R[0]}$$

# Example — in Bio-PEPA



```
k_s = 0.5;
k_r = 0.1;

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\texttt{I[10]} \underset{*}{\bowtie} \texttt{S[5]} \underset{*}{\bowtie} \texttt{R[0]}$$

# Conclusions

- Stochastic process algebras provide high-level description languages which can eases the task of model construction for large CTMC models.
- The formal nature of the langauge allows for unambiguous interpretation and automatic CTMC generation.
- Properties of the underlying mathematical structure can be detected at the syntax level and proof obligations can be carried out once and for all in the semantics of the langauge.
- Languages can be tailored to particular application domains making it easier for non-experts to build and analyse Markovian models.

## Conclusions

- Stochastic process algebras provide high-level description languages which can eases the task of model construction for large CTMC models.
- The formal nature of the langauge allows for unambiguous interpretation and automatic CTMC generation.
- Properties of the underlying mathematical structure can be detected at the syntax level and proof obligations can be carried out once and for all in the semantics of the langauge.
- Languages can be tailored to particular application domains making it easier for non-experts to build and analyse Markovian models.

## Conclusions

- Stochastic process algebras provide high-level description languages which can eases the task of model construction for large CTMC models.
- The formal nature of the langauge allows for unambiguous interpretation and automatic CTMC generation.
- Properties of the underlying mathematical structure can be detected at the syntax level and proof obligations can be carried out once and for all in the semantics of the langauge.
- Languages can be tailored to particular application domains making it easier for non-experts to build and analyse Markovian models.

# Conclusions

- Stochastic process algebras provide high-level description languages which can eases the task of model construction for large CTMC models.
- The formal nature of the langauge allows for unambiguous interpretation and automatic CTMC generation.
- Properties of the underlying mathematical structure can be detected at the syntax level and proof obligations can be carried out once and for all in the semantics of the langauge.
- Languages can be tailored to particular application domains making it easier for non-experts to build and analyse Markovian models.

# Thank you