Stochastic Process Algebras and Ordinary Differential Equations

Jane Hillston

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

5th September 2011

Outline

Outline

Process Algebra

- Models consist of agents which engage in actions.

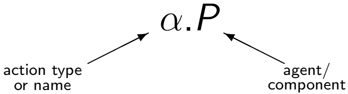
$$\alpha.P$$

action type or name → α

agent/ component → P

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.

$$\alpha.P$$

action type or name → α; agent/ component → P

- 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

Stochastic process algebras

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

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

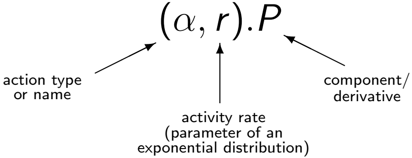
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

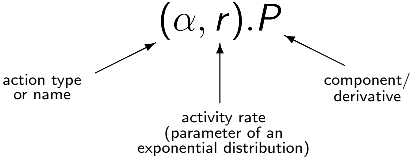
- Models are constructed from components which engage in activities.



- 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.

Why use a process algebra?

- High level description of the system eases the task of model construction.

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.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore 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.

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
$P_1 \bowtie_L P_2$	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
$P_1 \bowtie_{L} P_2$	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
$P_1 \bowtie_L P_2$	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.

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

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions. In PEPA cooperation between components gives rise to shared actions, the rate of which are governed by the assumption of bounded capacity.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions. In PEPA cooperation between components gives rise to shared actions, the rate of which are governed by the assumption of bounded capacity.

Bounded capacity

No component can be made to carry out an action in cooperation faster than its own defined rate for the action.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions. In PEPA cooperation between components gives rise to shared actions, the rate of which are governed by the assumption of bounded capacity.

Bounded capacity
No component can be made to carry out an action in cooperation faster than its own defined rate for the action.

Thus shared actions proceed at the minimum of the rates in the participating components.

Rates of interaction: bounded capacity

Stochastic process algebras differ in how they define the rate of synchronised actions. In PEPA cooperation between components gives rise to shared actions, the rate of which are governed by the assumption of bounded capacity.

Bounded capacity

No component can be made to carry out an action in cooperation faster than its own defined rate for the action.

Thus shared actions proceed at the minimum of the rates in the participating components.

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

A simple example: processors and resources

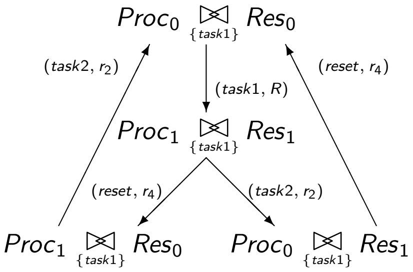
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

A simple example: processors and resources

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

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

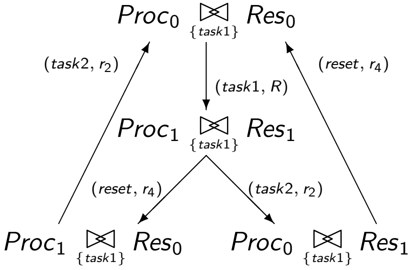


$R = \min(r_1, r_3)$

A simple example: processors and resources

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

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

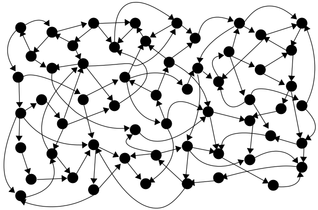


$R = \min(r_1, r_3)$

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

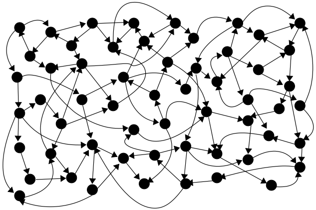
Solving discrete state models

As we have seen the SOS semantics of a SPA model is mapped to a CTMC with global states determined by the local states of the participating components.



Solving discrete state models

As we have seen the SOS semantics of a SPA model is mapped to a CTMC with global states determined by the local states of the participating components.

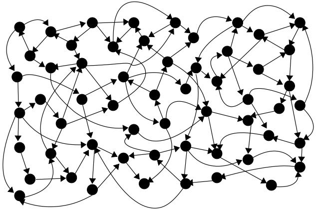


$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

Solving discrete state models

As we have seen the SOS semantics of a SPA model is mapped to a CTMC with global states determined by the local states of the participating components.

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

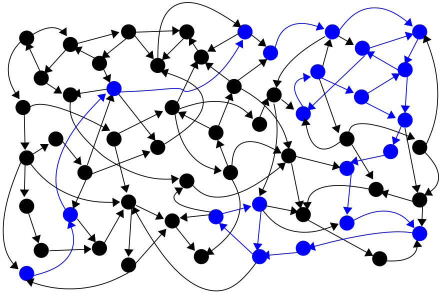


$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

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

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.



Benefits of process algebra

Beyond the clear benefits for model construction to be derived from using a high-level language with compositionality the formal nature of the process algebra specification has been exploited in a number of ways.

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.

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.

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 CSL family of temporal logics allows evaluation of quantified properties of the behaviour of the system

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

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

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

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

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

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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.

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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

Disadvantages of process algebra

The primary disadvantage of stochastic process algebras, shared by all discrete event modelling paradigms, is the problem of state space explosion, also known as the curse of dimensionality.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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.

A conundrum

Process algebras are well-suited to constructing such models:

A conundrum

Process algebras are well-suited to constructing such models:

- Developed to represent concurrent behaviour compositionally;

A conundrum

Process algebras are well-suited to constructing such models:

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

A conundrum

Process algebras are well-suited to constructing such models:

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

A conundrum

Process algebras are well-suited to constructing such models:

- 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.

A conundrum

Process algebras are well-suited to constructing such models:

- 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.

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

A conundrum

Process algebras are well-suited to constructing such models:

- 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.

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

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.

However, our main approach has been to use a counting abstraction in order to make a shift to population statistics and to develop a fluid approximation.

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.

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

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 formal models are analysed.

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 formal models are analysed.

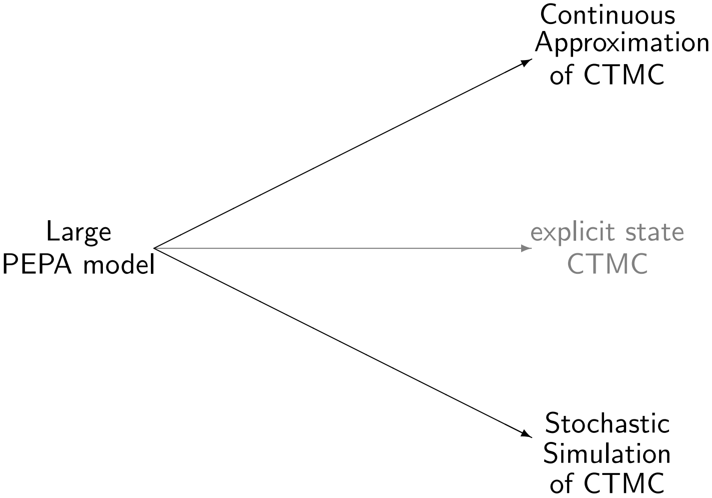
The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains which have traditionally worked directly at the level of fitting differential equation models.

Outline

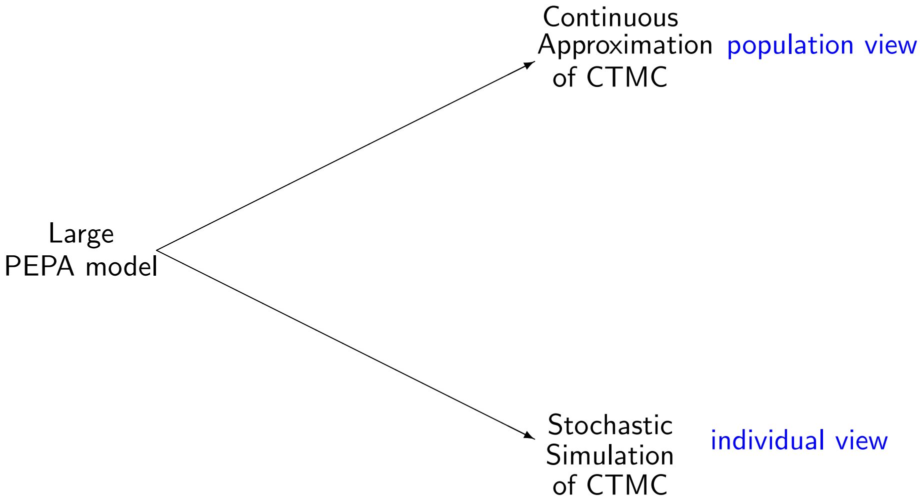
Alternative Representations

Large PEPA model → explicit state CTMC

Alternative Representations



Alternative Representations



A new approach to SPA semantics

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

A new approach to SPA semantics

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.

A new approach to SPA semantics

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.

A new approach to SPA semantics

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.

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.

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

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

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

- *task*1 decreases $Proc_0$ and Res_0
- *task*1 increases $Proc_1$ and Res_1
- *task*2 decreases $Proc_1$
- *task*2 increases $Proc_0$
- *reset* decreases Res_1
- *reset* increases Res_0

Simple example revisited

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

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

$$\frac{dx_1}{dt} = -\min(r_1\, x_1, r_3\, x_3) + r_2\, x_2$$

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

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

Simple example revisited

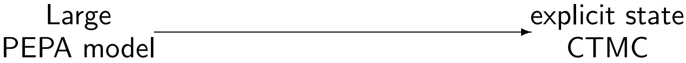
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

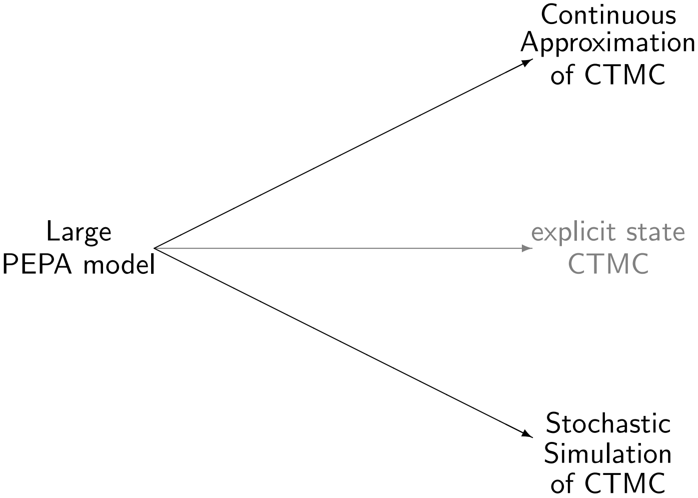
ODE interpretation

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

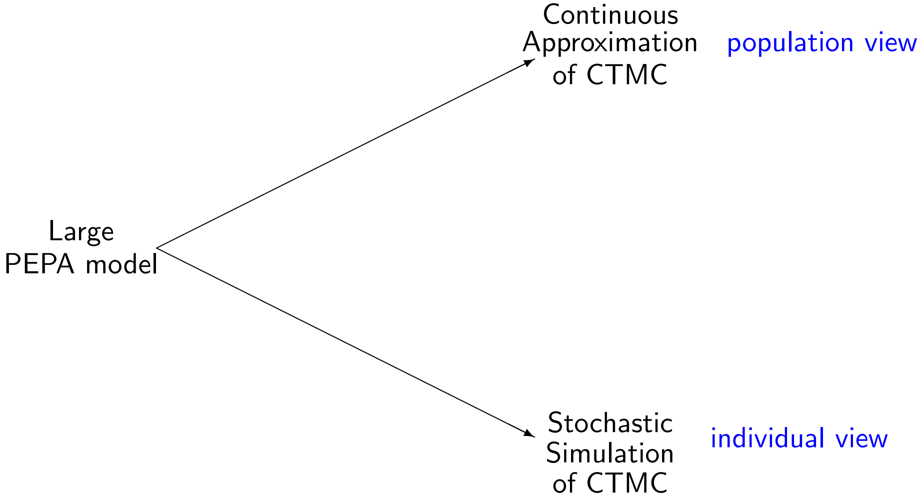
Alternative Representations



Alternative Representations



Alternative Representations



Alternative Representations

set of ODEs

Continuous Approximation of CTMC

population view

full generator matrix

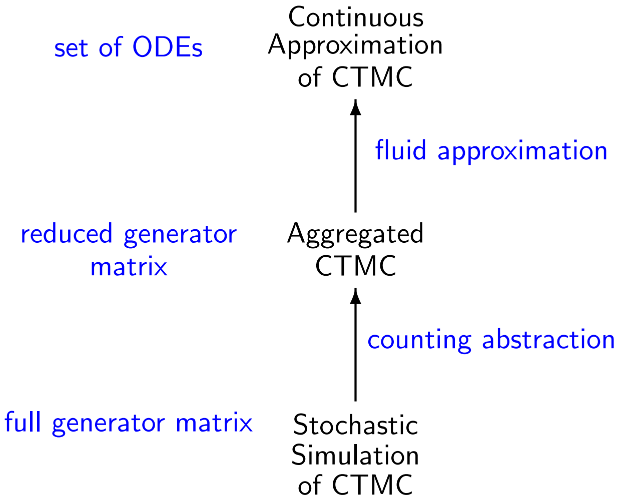
Stochastic Simulation of CTMC

individual view

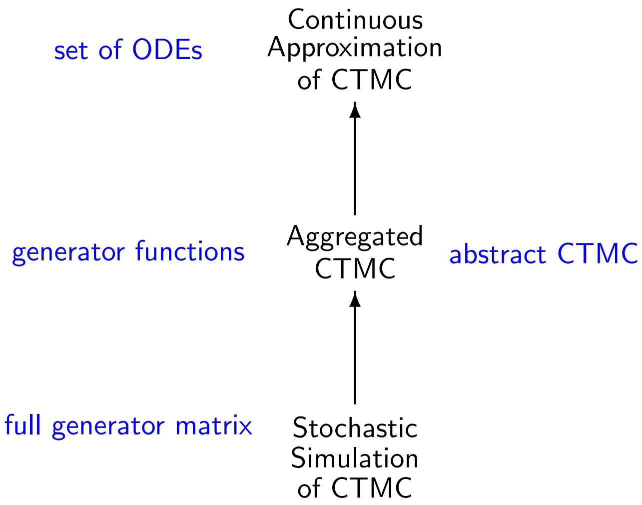
Alternative Representations

set of ODEs	Continuous Approximation of CTMC	population view
reduced generator matrix	Aggregated CTMC	
full generator matrix	Stochastic Simulation of CTMC	individual view

Alternative Representations



Alternative Representations



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 SOS semantics is not suitable 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 SOS semantics is not suitable because it constructs the state space of the CTMC explicitly.

Instead we define a structured operational semantics which defines the possible transitions of an abitrary abstract state, giving the CTMC implicitly as a set of generator functions, which directly give rise to 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 SOS semantics is not suitable because it constructs the state space of the CTMC explicitly.

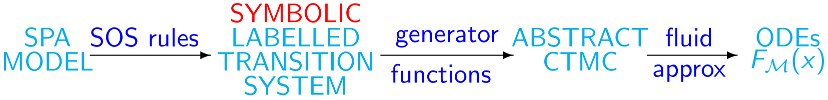
Instead we define a structured operational semantics which defines the possible transitions of an abitrary abstract state, giving the CTMC implicitly as a set of generator functions, which directly give rise to the ODEs.

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 SOS semantics is not suitable because it constructs the state space of the CTMC explicitly.

Instead we define a structured operational semantics which defines the possible transitions of an abitrary abstract state, giving the CTMC implicitly as a set of generator functions, which directly give rise to 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

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

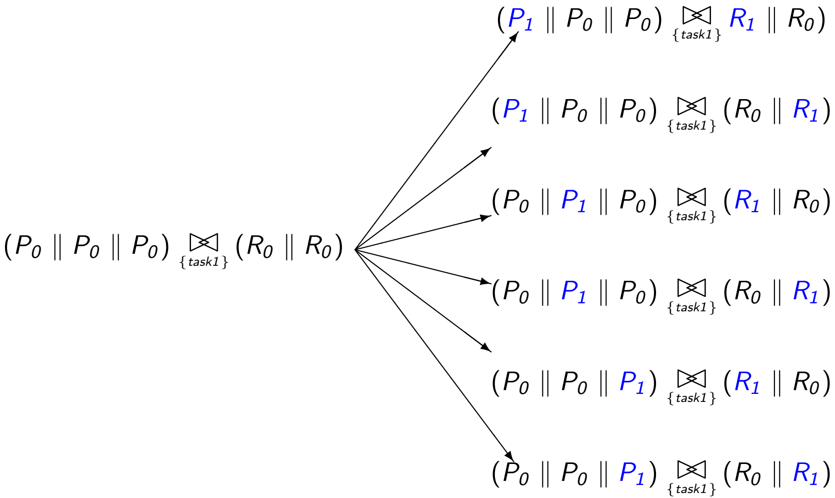
$$\dfrac{\dfrac{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1} \mathit{Proc_1}}{\mathit{Proc_0} \xrightarrow{\mathit{task1}, r_1 \xi_1}_* \mathit{Proc_1}} \qquad \dfrac{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_3} \mathit{Res_1}}{\mathit{Res_0} \xrightarrow{\mathit{task1}, r_3 \xi_3}_* \mathit{Res_1}}}{\mathit{Proc_0}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

Apparent Rate Calculation

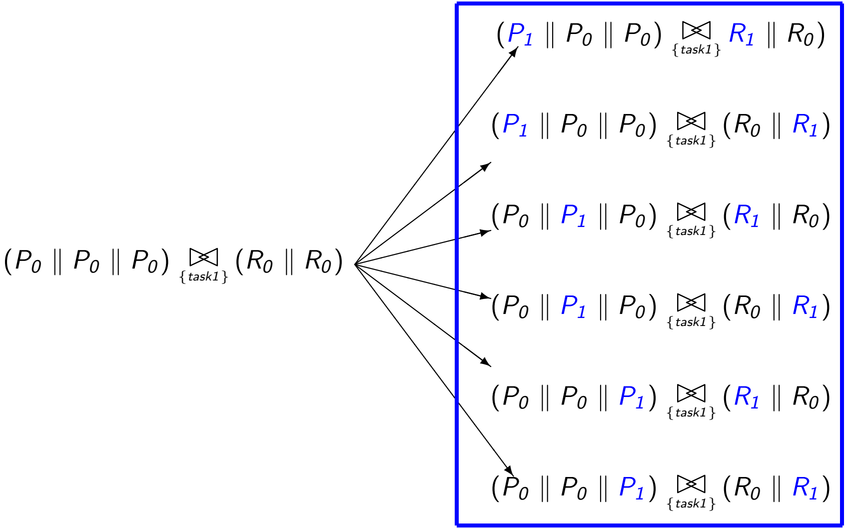
$$\dfrac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie}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

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

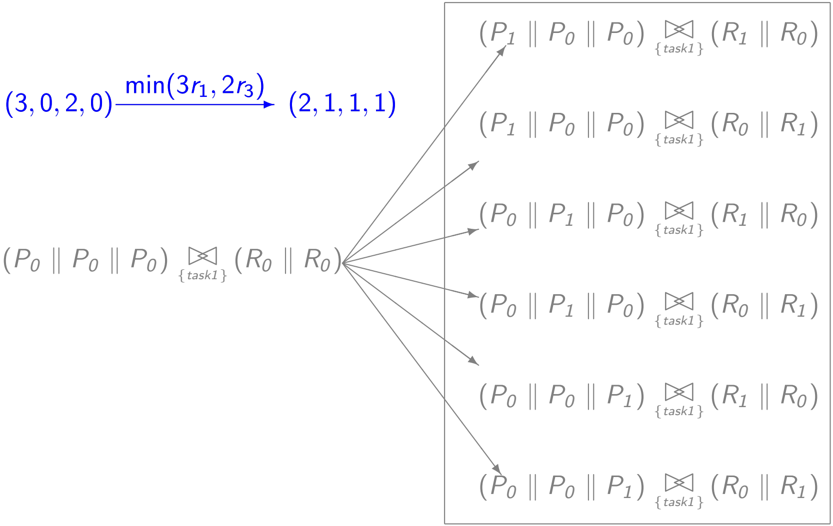
$f(\xi, l, \alpha)$ as the Generator Matrix of the *Lumped* CTMC



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



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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$f(\xi, (-1, +1, -1, +1), task1) = min(r_1\xi_1, r_3\xi_4)$$

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

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

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

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

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

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = -\min\left(r_1x_1, r_3x_3\right) + r_2x_2$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = \min\left(r_1x_1, r_3x_3\right) - r_2x_2$$

$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = -\min\left(r_1x_1, r_3x_3\right) + r_4x_4$$

$$\frac{\mathrm{d}x_4}{\mathrm{d}t} = \min\left(r_1x_1, r_3x_3\right) - r_4x_4$$

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

Why use a process algebra?

- High level description of the system eases the task of model construction.

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.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.

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.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore 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.

Conclusions

- Many interesting and important systems can be studied as a result of the link between stochastic process algebras and ODEs.

Conclusions

- Many interesting and important systems can be studied as a result of the link between stochastic process algebras and ODEs.
- Particularly, we can now consider examples of collective dynamics and emergent behaviour.

Conclusions

- Many interesting and important systems can be studied as a result of the link between stochastic process algebras and ODEs.
- Particularly, we can now consider examples of collective dynamics and emergent behaviour.
- Stochastic process algebras, 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 studied as a result of the link between stochastic process algebras and ODEs.
- Particularly, we can now consider examples of collective dynamics and emergent behaviour.
- Stochastic process algebras, 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.

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 necessarily 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 necessarily the information we require.
- Recent work has established the validity of performance measures such as throughput, and average response time derived from the ODE solutions [TDGH 2012, IEEE TSE].

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 necessarily the information we require.
- Recent work has established the validity of performance measures such as throughput, and average response time derived from the ODE solutions [TDGH 2012, IEEE TSE].
- 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 necessarily the information we require.
- Recent work has established the validity of performance measures such as throughput, and average response time derived from the ODE solutions [TDGH 2012, IEEE TSE].
- 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.
- Future work will also consider exploiting more of the formal structure of the process algebras to assist in the manipulation and analysis of the ODEs.

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!

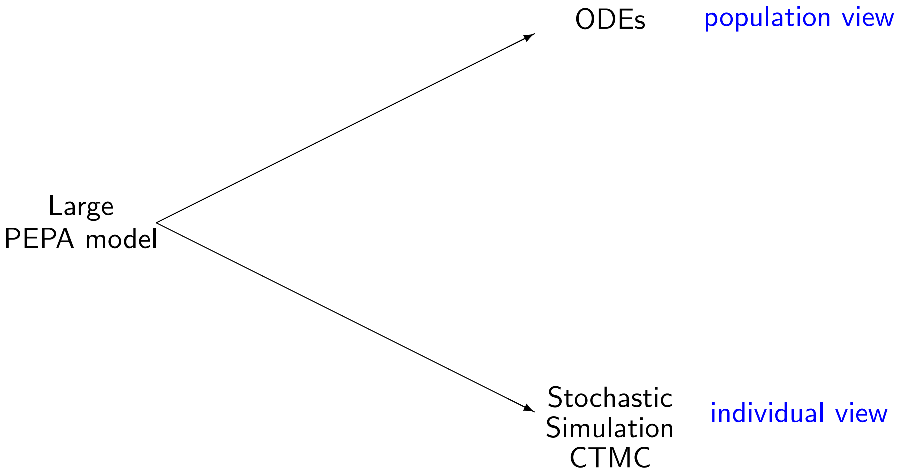
Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Andrea Bracciali, Jeremy Bradley, Luca Bortolussi, Federica Ciocchetta, Allan Clark, Jie Ding, Adam Duguid, Vashti Galpin, **Stephen Gilmore**, Diego Latella, Mieke Massink, **Mirco Tribastone**, and others.

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