Fluid Approximation for the Analysis of Collective Systems

Jane Hillston
LFCS, University of Edinburgh

July 23rd 2014

Outline

1. Introduction
   - Collective Systems
   - Quantitative Analysis
   - Stochastic Process Algebra
2. Quantitative Analysis of Collective Systems
   - Model construction
   - Mathematical analysis: fluid approximation
   - Numerical illustration
   - Deriving properties: fluid model checking
3. Conclusions

Outline

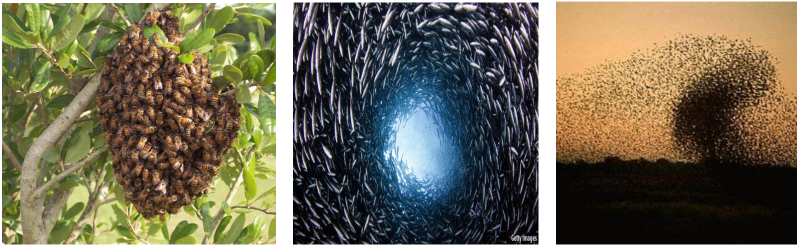
Collective Systems

We are surrounded by examples of collective systems:

Collective Systems

We are surrounded by examples of collective systems:

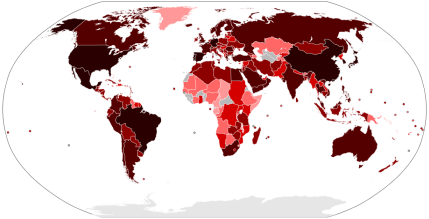
in the natural world ....



Collective Systems

We are surrounded by examples of collective systems:

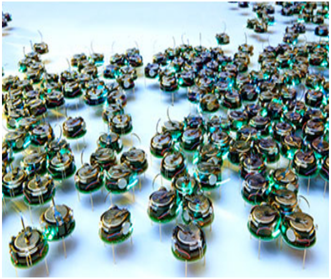
.... and in the man-made world



Collective Systems

We are surrounded by examples of collective systems:

.... and in the man-made world



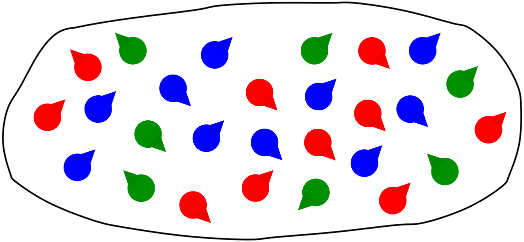
Collective Systems

We are surrounded by examples of collective systems:

an informatic environment

Collective Systems

From a computer science perspective these systems can be viewed as being made up of a large number of interacting entities.

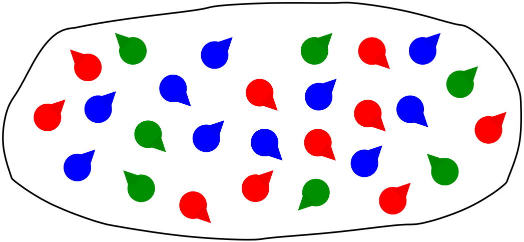


Each entity may have its own properties, objectives and actions.

At the system level these combine to create the collective behaviour.

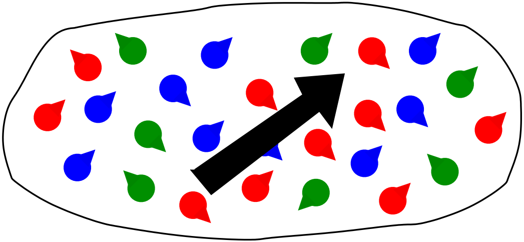
Collective Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



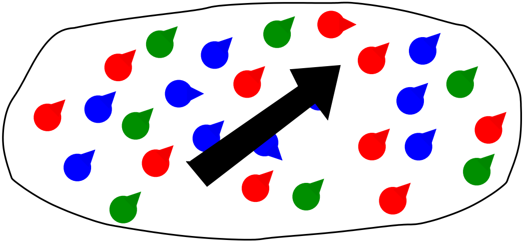
Collective Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



Collective Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



And the behaviour of the individuals will be influenced by the state of the overall system.

The Informatic Environment

Robin Milner coined the term of informatics environment, in which pervasive computing elements are embedded in the human environment, invisibly providing services and responding to requirements.

The Informatic Environment

Robin Milner coined the term of informatics environment, in which pervasive computing elements are embedded in the human environment, invisibly providing services and responding to requirements.

Such systems are now becoming the reality, and many form collective adaptive systems, in which large numbers of computing elements collaborate to meet the human need.

The Informatic Environment

Robin Milner coined the term of informatics environment, in which pervasive computing elements are embedded in the human environment, invisibly providing services and responding to requirements.

Such systems are now becoming the reality, and many form collective adaptive systems, in which large numbers of computing elements collaborate to meet the human need.

For instance, may examples of such systems can be found in components of Smart Cities, such as smart urban transport and smart grid electricity generation and storage.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed.

Quantitative Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the efficient and equitable sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed.

These techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems, and this is even more true of systems with collective behaviour.

Performance Modelling: Motivation

Capacity Planning

- How many clients can the existing server support and maintain reasonable response times?
- How many buses do I need to maintain service at peak time in a smart urban transport system?

Performance Modelling: Motivation

Capacity Planning

- How many clients can the existing server support and maintain reasonable response times?
- How many buses do I need to maintain service at peak time in a smart urban transport system?

System Configuration

- Where should I place base stations in a network to keep blocking probabilities low?
- What capacity do I need at bike stations to minimise the movement of bikes by truck?

Performance Modelling: Motivation

Capacity Planning

- How many clients can the existing server support and maintain reasonable response times?
- How many buses do I need to maintain service at peak time in a smart urban transport system?

System Configuration

- Where should I place base stations in a network to keep blocking probabilities low?
- What capacity do I need at bike stations to minimise the movement of bikes by truck?

System Tuning

- In an automated factory what speed of conveyor belt will minimize robot idle time and jamming but maximize throughput?
- What strategy can I use to maintain supply-demand balance within a smart electricity grid?

Performance Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

Performance Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

Performance Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

From these high-level system descriptions the underlying mathematical model (Continuous Time Markov Chain (CTMC)) can be automatically generated.

Performance Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

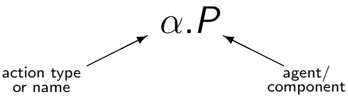
From these high-level system descriptions the underlying mathematical model (Continuous Time Markov Chain (CTMC)) can be automatically generated.

Primary examples include:

- Stochastic Petri Nets and
- Stochastic/Markovian Process Algebras.

Process Algebra

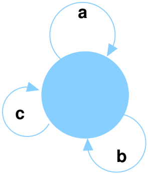
- Models consist of agents which engage in actions.



$\alpha.P$

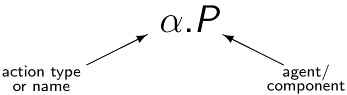
action type or name

agent/ component



Process Algebra

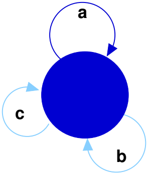
- Models consist of agents which engage in actions.



$\alpha.P$

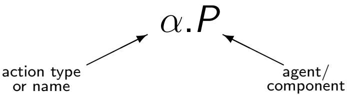
action type or name

agent/ component



Process Algebra

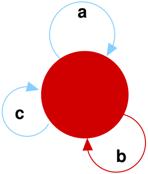
- Models consist of agents which engage in actions.



$$\alpha.P$$

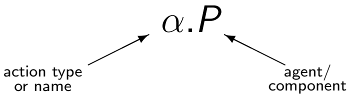
action type
or name

agent/
component



Process Algebra

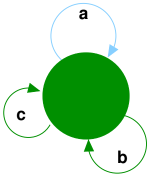
- Models consist of agents which engage in actions.



$\alpha.P$

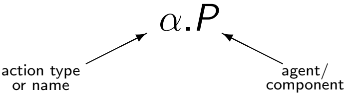
action type or name

agent/ component



Process Algebra

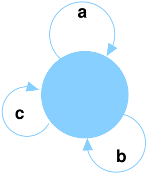
- Models consist of agents which engage in actions.



$\alpha.P$

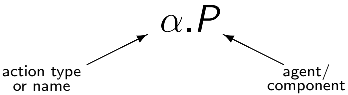
action type or name

agent/ component



Process Algebra

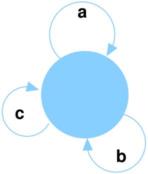
- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

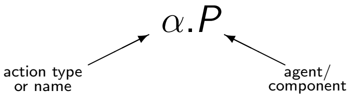
agent/ component



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

Process Algebra

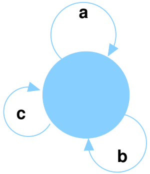
- Models consist of agents which engage in actions.



$\alpha.P$

action type
or name

agent/
component

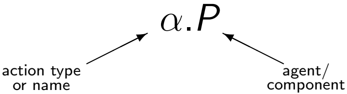


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra
model

Process Algebra

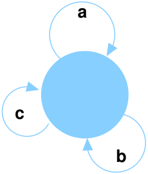
- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

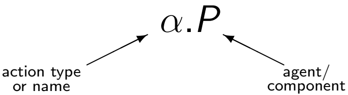


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

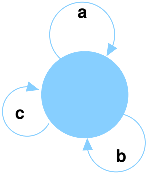
Process algebra model $\xrightarrow{\text{SOS rules}}$

Process Algebra

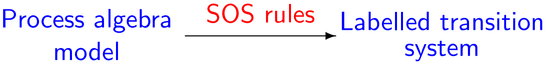
- Models consist of agents which engage in actions.



$\alpha.P$



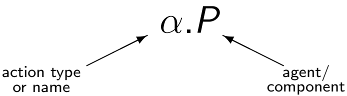
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

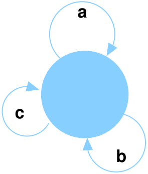


Process algebra model $\xrightarrow{\text{SOS rules}}$ Labelled transition system

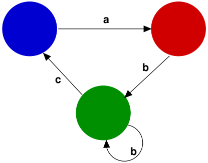
Process Algebra

- Models consist of agents which engage in actions.


$\alpha.P$



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

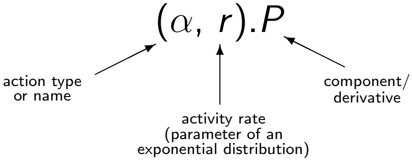


Stochastic process algebras

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

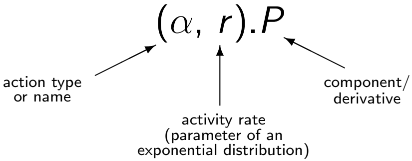
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

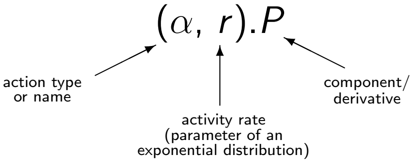
activity rate (parameter of an exponential distribution)

component/ derivative

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type
or name

activity rate
(parameter of an
exponential distribution)

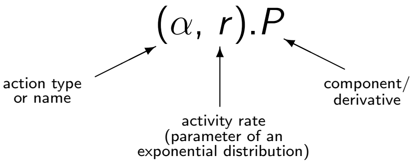
component/
derivative

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

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



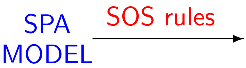
$(\alpha, r).P$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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

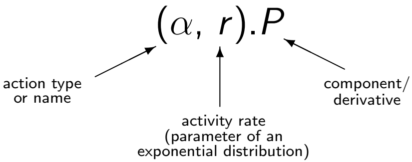


SPA
MODEL

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

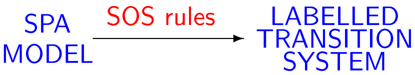
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



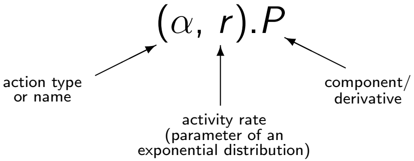
$$(\alpha, r).P$$

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



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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

SPA
MODEL

SOS rules

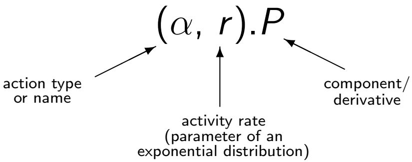
LABELLED
TRANSITION
SYSTEM

state transition

diagram

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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

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

Integrated analysis

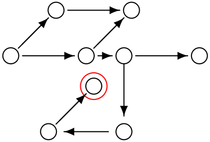
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

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

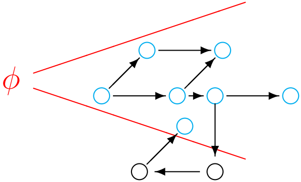


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

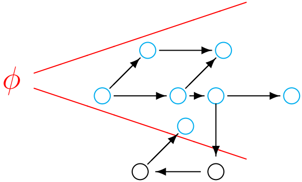


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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



Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

$P_1 \parallel P_2$ 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.

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

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a "small step" semantics).

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Structured Operational Semantics

PEPA is defined using a Plotkin-style structured operational semantics (a “small step” semantics).

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'}$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \, (\alpha \notin L)$$

Structured Operational Semantics: Cooperation ($\alpha \notin L$)

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \quad (\alpha \notin L)$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \quad (\alpha \notin L)$$

Structured Operational Semantics: Cooperation ($\alpha \in L$)

Cooperation $$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L)$$

Structured Operational Semantics: Cooperation ($\alpha \in L$)

Cooperation

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L)$$

$$\text{where} \quad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} min(r_\alpha(E), r_\alpha(F))$$

Apparent Rate

$$
\begin{aligned}
r_\alpha((\beta, r).P) &= \begin{cases} r & \beta = \alpha \\ 0 & \beta \neq \alpha \end{cases} \\
r_\alpha(P + Q) &= r_\alpha(P) + r_\alpha(Q) \\
r_\alpha(A) &= r_\alpha(P) \qquad \text{where } A \overset{def}{=} P \\
r_\alpha(P \underset{L}{\bowtie} Q) &= \begin{cases} r_\alpha(P) + r_\alpha(Q) & \alpha \notin L \\ min(r_\alpha(P), r_\alpha(Q)) & \alpha \in L \end{cases} \\
r_\alpha(P/L) &= \begin{cases} r_\alpha(P) & \alpha \notin L \\ 0 & \alpha \in L \end{cases}
\end{aligned}
$$

Structured Operational Semantics: Hiding

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \, (\alpha \notin L)$$

Structured Operational Semantics: Hiding

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \ (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \ (\alpha \in L)$$

Structured Operational Semantics: Constants

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} \, (A \overset{\mathit{def}}{=} E)$$

A simple example: processors and resources

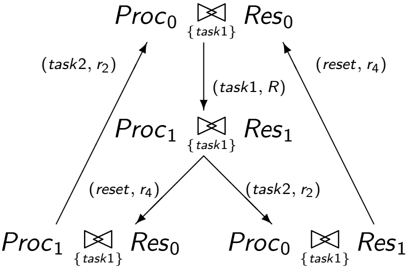
$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_3).Res_1 \\
Res_1 &\overset{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

A simple example: processors and resources

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

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



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

$(task1, R)$

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

$(reset, r_4)$

$(task2, r_2)$

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

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

$(task2, r_2)$

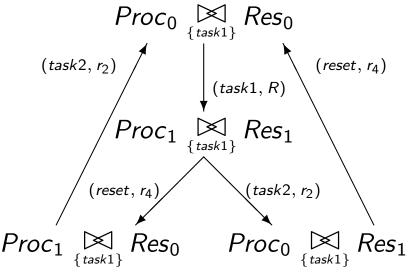
$(reset, r_4)$

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

A simple example: processors and resources

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

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



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

$(task2, r_2)$

$(task1, R)$

$(reset, r_4)$

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

$(reset, r_4)$

$(task2, r_2)$

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

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

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

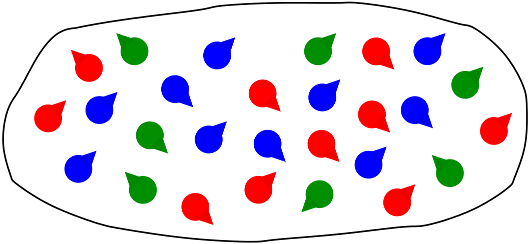
$$
\mathbf{Q} = \begin{pmatrix}
-R & R & 0 & 0 \\
0 & -(r_2 + r_4) & r_4 & r_2 \\
r_2 & 0 & -r_2 & 0 \\
r_4 & 0 & 0 & -r_4
\end{pmatrix}
$$

Outline

1. Introduction
   - Collective Systems
   - Quantitative Analysis
   - Stochastic Process Algebra
2. Quantitative Analysis of Collective Systems
   - Model construction
   - Mathematical analysis: fluid approximation
   - Numerical illustration
   - Deriving properties: fluid model checking
3. Conclusions

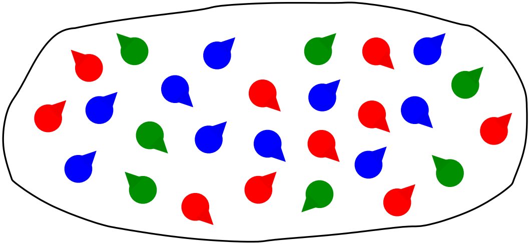
Modelling collective behaviour

A key feature of collective systems is the existence of populations of entities who share certain characteristics.



Modelling collective behaviour

A key feature of collective systems is the existence of populations of entities who share certain characteristics.



High-level modelling formalisms allow this repetition to be captured at the high-level rather than explicitly.

Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

- Developed to represent concurrent behaviour compositionally;

Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

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

Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;

Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

- Developed to represent concurrent behaviour compositionally;
- Represent the interactions between individuals explicitly;
- Populations are readily and rigorously identified;
- Stochastic extensions allow the dynamics of system behaviour to be captured;

Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

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

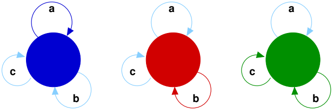
Process Algebra for Collective Systems

Process algebra are well-suited for constructing models of collective systems:

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

Recent advances in analysis techniques for process algebras have made it possible to study such systems even when the number of entities and activities become huge.

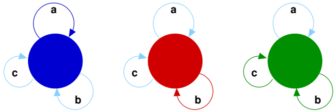
Solving discrete state models

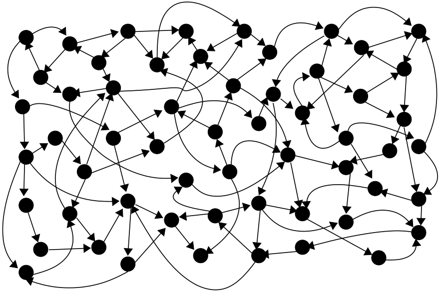


Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

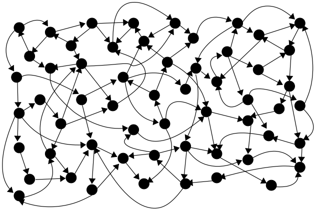
Solving discrete state models

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



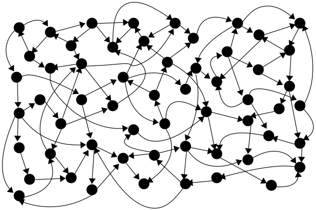


Solving discrete state models



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

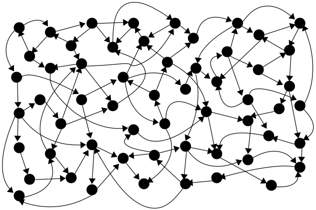
Solving discrete state models



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

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

Solving discrete state models



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

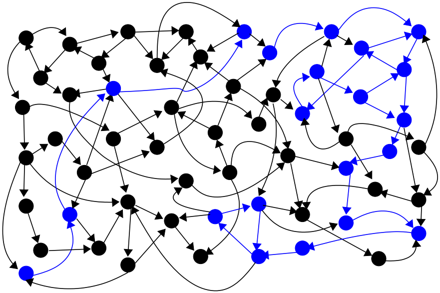
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

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

$$\pi(\infty)Q = 0$$

Solving discrete state models

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



State space explosion

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

Identity and Individuality

Collective systems are constructed from many instances of a set of components.

Identity and Individuality

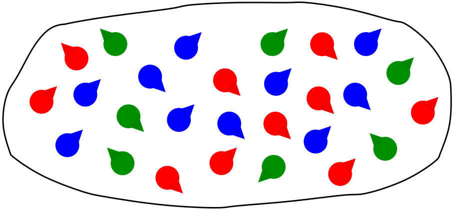
Collective systems are constructed from many instances of a set of components.

If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.

Identity and Individuality

Collective systems are constructed from many instances of a set of components.

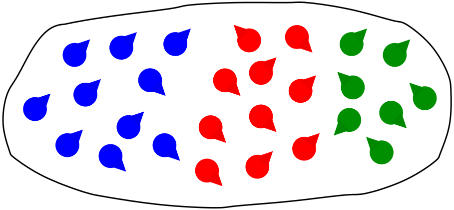
If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.



Identity and Individuality

Collective systems are constructed from many instances of a set of components.

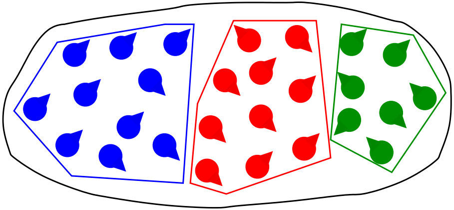
If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.



Identity and Individuality

Collective systems are constructed from many instances of a set of components.

If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.

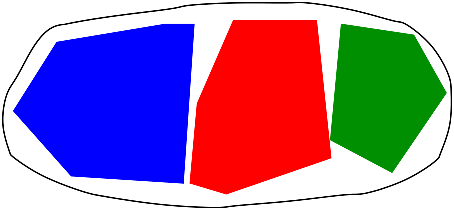


We may choose to disregard the identity of components.

Identity and Individuality

Collective systems are constructed from many instances of a set of components.

If we cease to distinguish between instances of components we can form an aggregation or counting abstraction to reduce the state space.



We may choose to disregard the identity of components.

Even better reductions can be achieved when we no longer regard the components as individuals.

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 calculate the proportion 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 calculate the proportion of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

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

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

Continuous Approximation

This means shifting to a different mathematical representation, where we no longer keep track of the individual states of each entity.

As we are focussed instead of proportions within populations we now treat these variables as continuous rather than discrete.

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

Simple example revisited

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

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

Simple example revisited

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

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

CTMC interpretation

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

Simple example revisited

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

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

- $task1$ 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

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

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

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = -\min(r_1\, x_1, r_3\, x_3) + r_2\, x_2$$

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

- $task1$ 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

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

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

ODE interpretation

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

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

$$\frac{dx_2}{dt} = \min(r_1\, x_1, r_3\, x_3) - r_2\, x_2$$

$$x_2 = \text{no. of } Proc_2$$

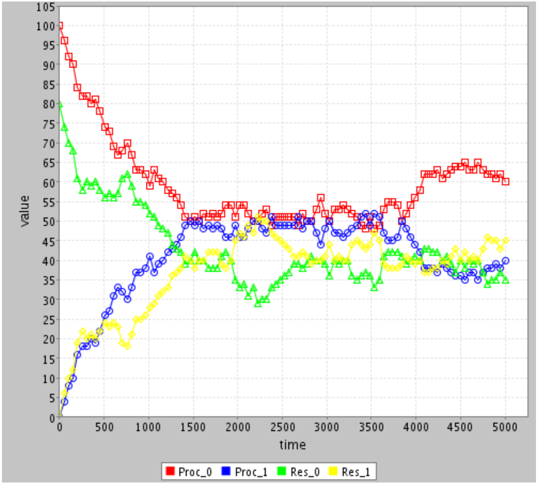
$$\frac{dx_3}{dt} = -\min(r_1\, x_1, r_3\, x_3) + r_4\, x_4$$

$$x_3 = \text{no. of } Res_0$$

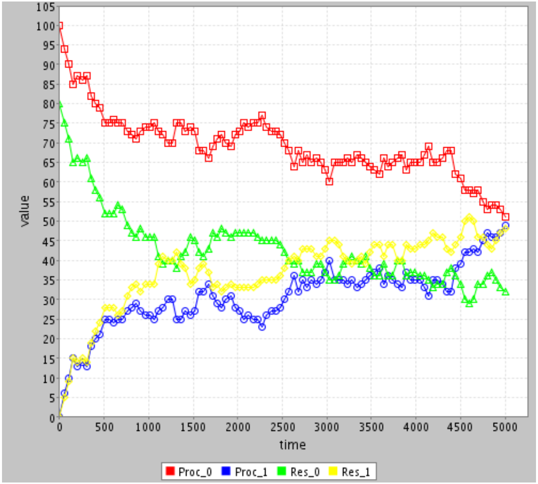
$$\frac{dx_4}{dt} = \min(r_1\, x_1, r_3\, x_3) - r_4\, x_4$$

$$x_4 = \text{no. of } Res_1$$

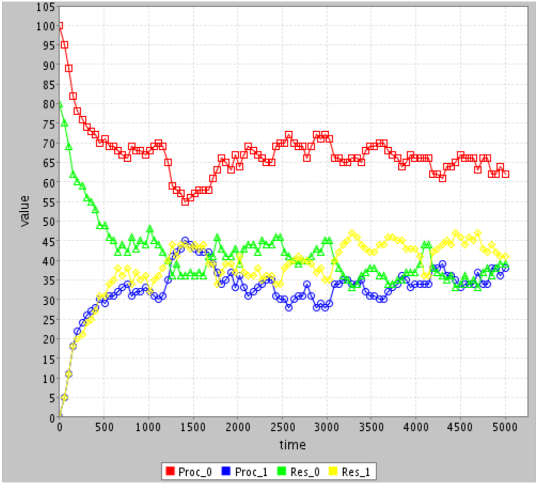
100 processors and 80 resources (simulation run A)



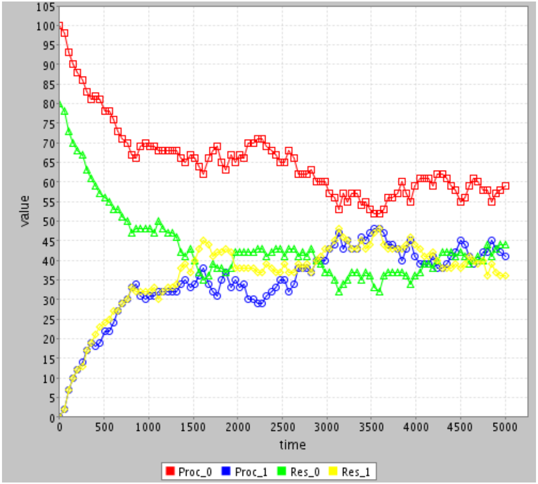
100 processors and 80 resources (simulation run B)



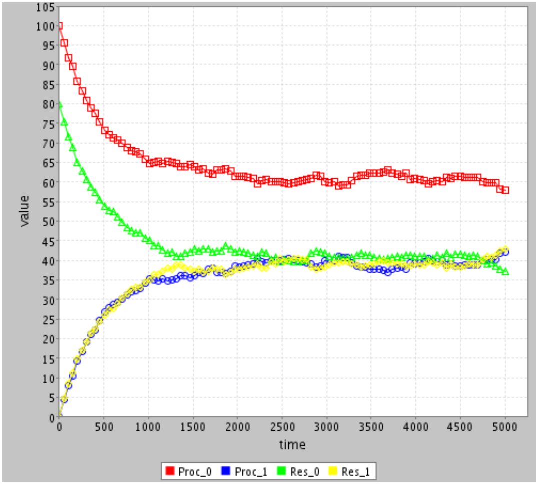
100 processors and 80 resources (simulation run C)



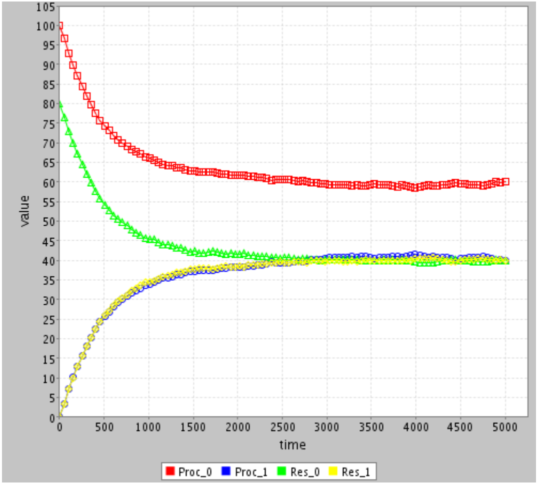
100 processors and 80 resources (simulation run D)



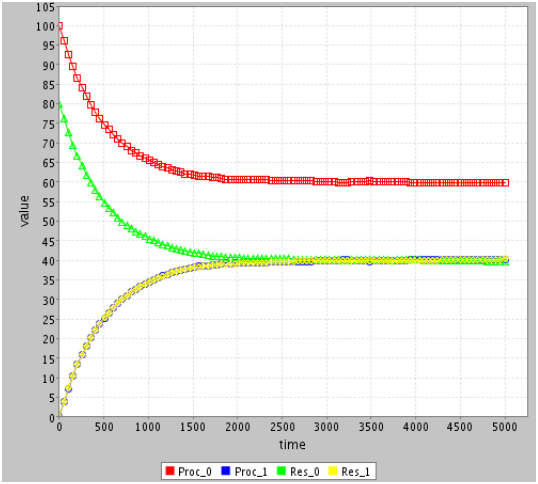
100 processors and 80 resources (average of 10 runs)



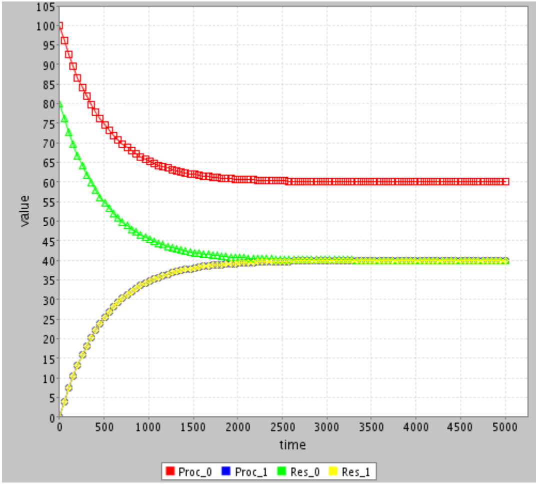
100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (ODE solution)



Deriving a Fluid Approximation of a SPA 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 SPA 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 existing (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

Deriving a Fluid Approximation of a SPA 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 existing (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

SPA MODEL

SOS rules

LABELLED TRANSITION SYSTEM

state transition diagram

CTMC **Q**

Deriving a Fluid Approximation of a SPA 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 existing (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.

Deriving a Fluid Approximation of a SPA 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 existing (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 **Q** or ODEs $F_{\mathcal{M}}(x)$

Fluid Structured Operational Semantics

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

Fluid Structured Operational Semantics

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

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)

Fluid Structured Operational Semantics

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

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (Jump Multiset)

Fluid Structured Operational Semantics

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

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (Jump Multiset)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Fluid Structured Operational Semantics

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

1. Remove excess components to identify the counting abstraction of the process (Context Reduction)
2. Collect the transitions of the reduced context as symbolic updates on the state representation (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, under the assumption that the population size tends to infinity.

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

Context Reduction

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

Location Dependency

$$\mathit{System} \overset{\mathit{def}}{=} \mathit{Proc}_0[N'_C] \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res}_0[N_S] \parallel \mathit{Proc}_0[N''_C]$$

$$\Downarrow$$

$$\{\mathit{Proc}_0, \mathit{Proc}_1\} \underset{\{\mathit{task1}\}}{\bowtie} \{\mathit{Res}_0, \mathit{Res}_1\} \parallel \{\mathit{Proc}_0, \mathit{Proc}_1\}$$

Location Dependency

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

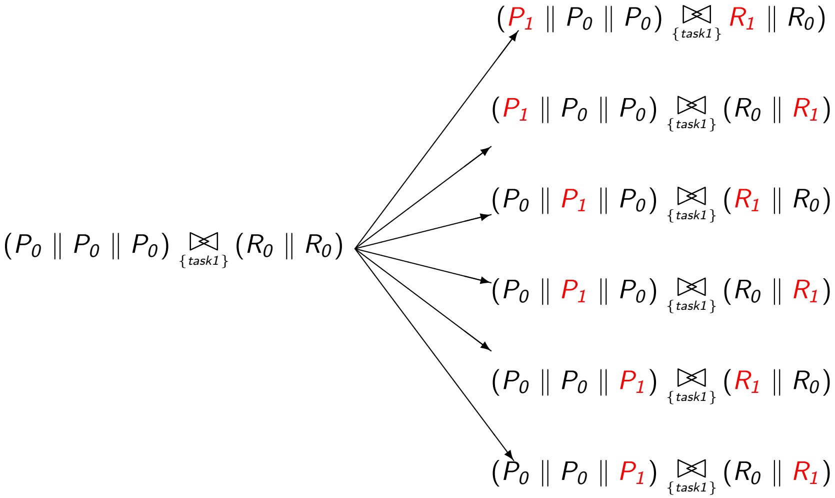
$$\dfrac{\dfrac{\mathit{Proc0} \xrightarrow{\mathit{task1}, r_1} \mathit{Proc1}}{\mathit{Proc0} \xrightarrow{\mathit{task1}, r_1\xi_1}_* \mathit{Proc1}} \qquad \dfrac{\mathit{Res0} \xrightarrow{\mathit{task1}, r_3} \mathit{Res1}}{\mathit{Res0} \xrightarrow{\mathit{task1}, r_3\xi_3}_* \mathit{Res1}}}{\mathit{Proc0} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res0} \xrightarrow{\mathit{task1}, r(\xi)}_* \mathit{Proc1} \underset{\{\mathit{task1}\}}{\bowtie} \mathit{Res1}}$$

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



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

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

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

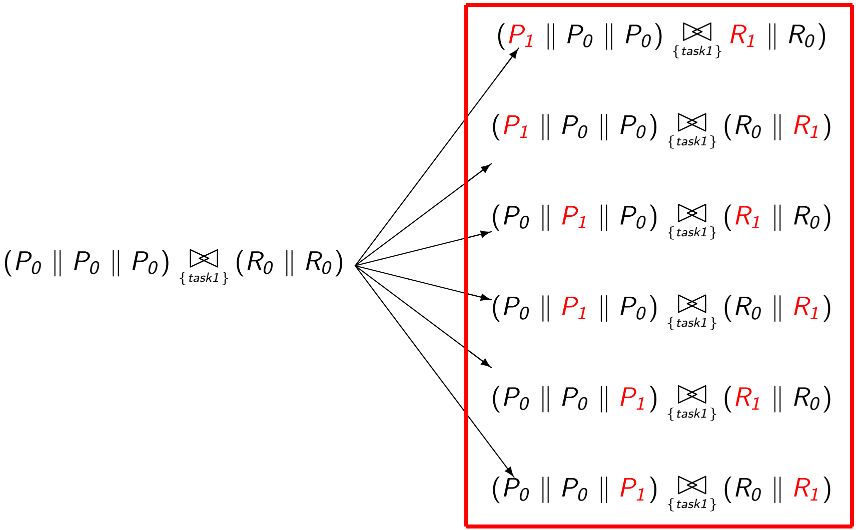
$(P_0 \parallel P_0 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_0)$

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

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

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

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



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

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

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

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

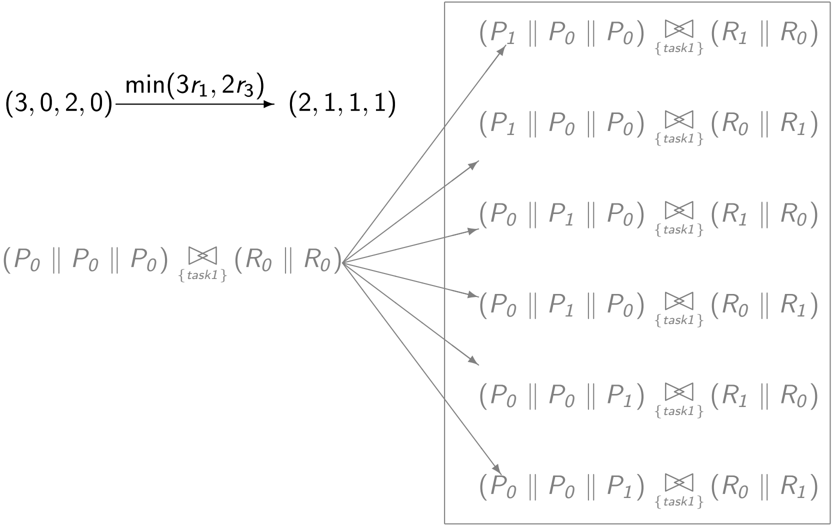
$$(P_0 \parallel P_1 \parallel P_0) \underset{\{task1\}}{\bowtie} (R_0 \parallel R_1)$$

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

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

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

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



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

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

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

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

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

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

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

Jump Multiset

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

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

Jump Multiset

$$\mathit{Proc}_0 \underset{\{task1\}}{\bowtie} \mathit{Res}_0 \xrightarrow{task1,\ r(\xi)}_{*} \mathit{Proc}_1 \underset{\{task1\}}{\bowtie} \mathit{Res}_1$$

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

$$\mathit{Proc}_1 \underset{\{task1\}}{\bowtie} \mathit{Res}_0 \xrightarrow{task2,\ \xi_2 r_2}_{*} \mathit{Proc}_0 \underset{\{task1\}}{\bowtie} \mathit{Res}_0$$

Jump Multiset

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

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

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

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

Equivalent Transitions

Some transitions may give the same information:

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

i.e., Res_1 may perform an action independently from the rest of the system.

This is captured by the procedure used for the construction of the generator function $f(\xi, l, \alpha)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = F_{\mathcal{M}}(x) = \sum_{l\in\mathbb{Z}^d} l \sum_{\alpha\in\mathcal{A}} f(x,l,\alpha)$$

$$\begin{aligned} &= (-1,1,-1,1)\min\left(r_1x_1, r_3x_3\right) + (1,-1,0,0)r_2x_2 \\ &\qquad + (0,0,1,-1)r_4x_4 \end{aligned}$$

Extraction of the ODE from f

Generator Function

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

Differential Equation

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

Consistency results

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

Consistency results

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

- The generated ODEs are the fluid limit of the family of CTMCs: 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: 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: 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.

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

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

- Vector fields have been defined to approximate higher moments.

R.A.Hayden and J.T.Bradley. A fluid analysis framework for a Markovian process algebra. TCS 2010.

Quantitative properties

The derived vector field $\mathcal{F}(x)$, gives an approximation of the expected count for each population over time.

This has been extended in a number of ways:

- Fluid rewards which can be safely calculated from the fluid expectation trajectories.

M.Tribastone, J.Ding, S.Gilmore and J.Hillston. Fluid Rewards for a Stochastic Process Algebra. IEEE TSE 2012.

- Vector fields have been defined to approximate higher moments.

R.A.Hayden and J.T.Bradley. A fluid analysis framework for a Markovian process algebra. TCS 2010.

- Fluid approximation of passage times have been defined.

R.A.Hayden, A.Stefanek and J.T.Bradley. Fluid computation of passage-time distributions in large Markov models. TCS 2012.

Fluid model checking

Since the vector field records only deterministic behaviour, LTL model checking can be used over a trace to give boolean results.

Fluid model checking

Since the vector field records only deterministic behaviour, LTL model checking can be used over a trace to give boolean results.

But for the systems we are interested in we would like some more quantified answers, in the style of stochastic model checking.

Fluid model checking

Since the vector field records only deterministic behaviour, LTL model checking can be used over a trace to give boolean results.

But for the systems we are interested in we would like some more quantified answers, in the style of stochastic model checking.

Work on this is on-going but there are initial results for:

- CSL properties of a single agent within a population.

L.Bortolussi and J.Hillston. Fluid model checking. CONCUR 2012.

- The fraction of a population that satisfies a property expressed as a one-clock deterministic timed automaton.

L.Bortolussi and R.Lanciani. Central Limit Approximation for Stochastic Model Checking. QEST 2013.

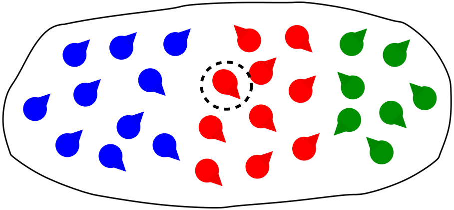
CSL model checking of a single agent

We consider properties of a single agent within a population, expressed in the Continuous Stochastic Logic (CSL), usually used for model checking CTMCs.

CSL model checking of a single agent

We consider properties of a single agent within a population, expressed in the Continuous Stochastic Logic (CSL), usually used for model checking CTMCs.

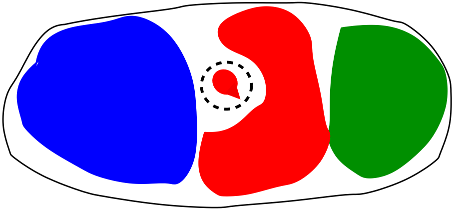
We consider an arbitrary member of the population.



CSL model checking of a single agent

We consider properties of a single agent within a population, expressed in the Continuous Stochastic Logic (CSL), usually used for model checking CTMCs.

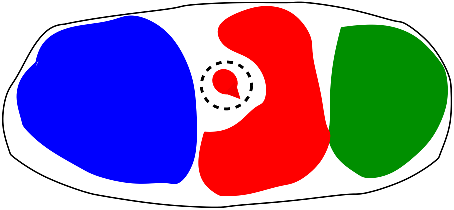
This agent is kept discrete, making transitions between its discrete states, but all other agents are treated as a mean-field influencing the behaviour of this agent.



CSL model checking of a single agent

We consider properties of a single agent within a population, expressed in the Continuous Stochastic Logic (CSL), usually used for model checking CTMCs.

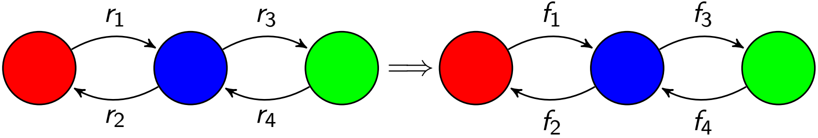
Essentially we keep a detailed discrete-event representation of the one agent and make a fluid approximation of the rest of the population.

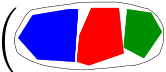


Inhomogeneous CTMC

The transition rates within the discrete-event representation will depend on the rest of the population.

i.e. it will depend on the vector field capturing the behaviour of the residual population.

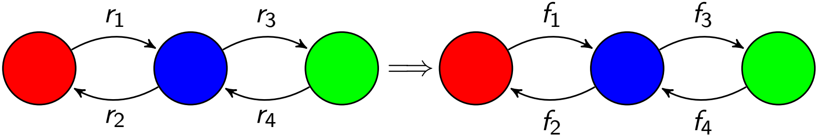


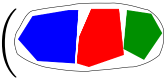
where $f_i = f\Big($  $\Big)$

Inhomogeneous CTMC

The transition rates within the discrete-event representation will depend on the rest of the population.

i.e. it will depend on the vector field capturing the behaviour of the residual population.



where $f_i = f\Bigg($  $\Bigg)$

It is an inhomogeneous continuous time Markov chain.

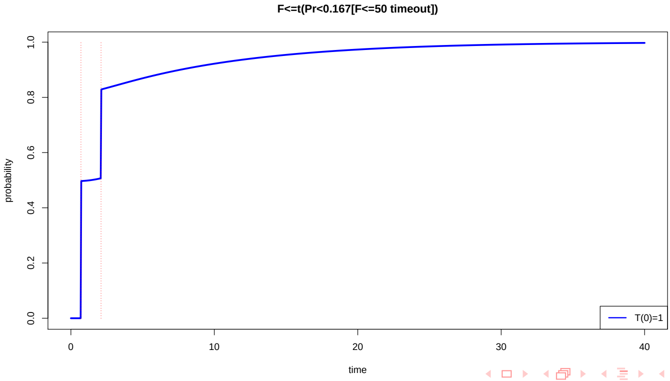
Model checking the ICTMC

Care is needed to model check the ICTMC, which proceeds by explicitly calculating the reachability probability probabilities for states of interest (analogously to CSL model checking on CTMCs).

Model checking the ICTMC

Care is needed to model check the ICTMC, which proceeds by explicitly calculating the reachability probability probabilities for states of interest (analogously to CSL model checking on CTMCs).

The inhomogeneous time within the model means that truth values may change with respect to time.



Outline

Conclusions

- Collective Systems are an interesting and challenging class of systems to design and construct.

Conclusions

- Collective Systems are an interesting and challenging class of systems to design and construct.
- Their role within infrastructure, such as within smart cities, make it essential that quantitive aspects of behaviour is taken into consideration, as well as functional correctness.

Conclusions

- Collective Systems are an interesting and challenging class of systems to design and construct.
- Their role within infrastructure, such as within smart cities, make it essential that quantitive aspects of behaviour is taken into consideration, as well as functional correctness.
- Fluid approximation based analysis offers hope for scalable quantitative analysis techniques, but there remain many interesting and challenging problems to be solved.

Conclusions

- Collective Systems are an interesting and challenging class of systems to design and construct.
- Their role within infrastructure, such as within smart cities, make it essential that quantitive aspects of behaviour is taken into consideration, as well as functional correctness.
- Fluid approximation based analysis offers hope for scalable quantitative analysis techniques, but there remain many interesting and challenging problems to be solved.
- In particular we currently seek to bring the fluid approximation techniques to systems with distinct locations.

Thank you!

Thank you!

Thanks to the other members of the QUANTICOL project

quanticol

www.quanticol.eu