Choosing not to be discrete — The benefits of fluid approximations in dynamic modelling

Jane Hillston
LFCS, University of Edinburgh

27th April 2015

Outline

Outline

The Discrete World View

As computer scientists we generally take a discrete view of the world.

The Discrete World View

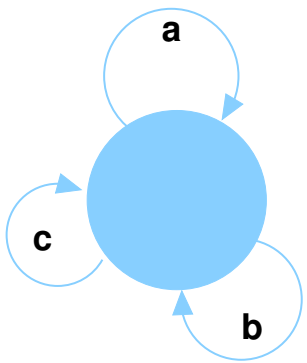
As computer scientists we generally take a discrete view of the world.

This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.

The Discrete World View

As computer scientists we generally take a discrete view of the world.

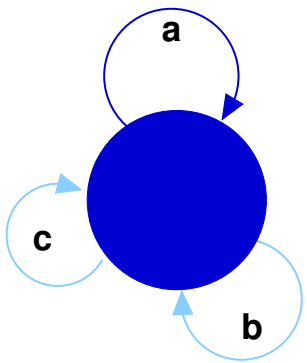
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

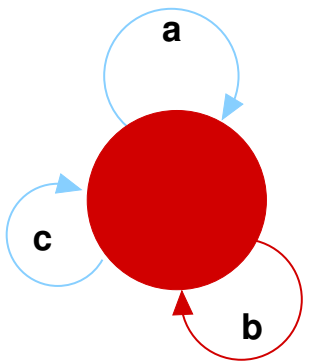
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

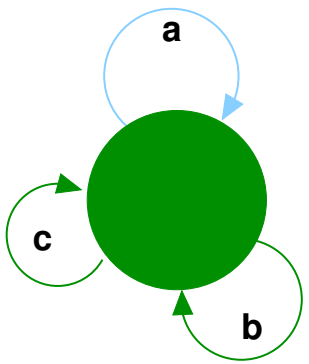
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

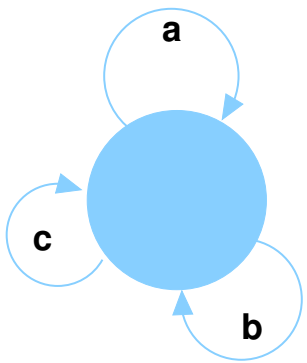
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.



The Discrete World View

As computer scientists we generally take a discrete view of the world.

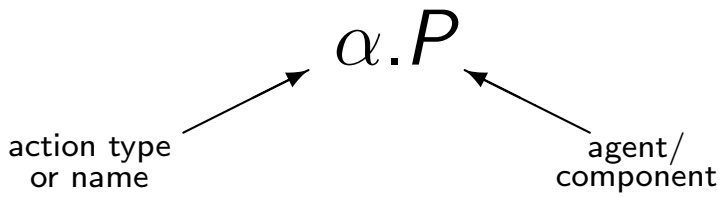
This is particularly true when we want to reason about the behaviour of systems, as most formalisms are built upon notions of states and transitions.

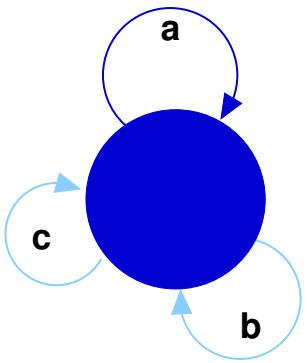


Various formalisms have been designed for capturing such behaviour.

Process Algebra

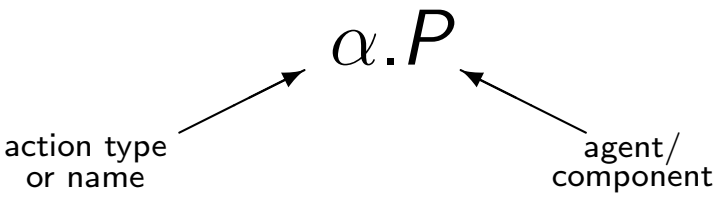
- Models consist of agents which engage in actions.

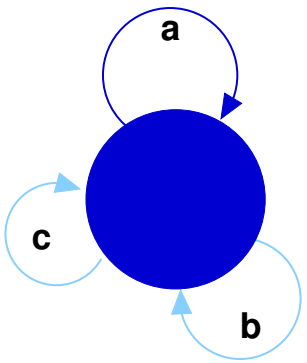




Process Algebra

- Models consist of agents which engage in actions.

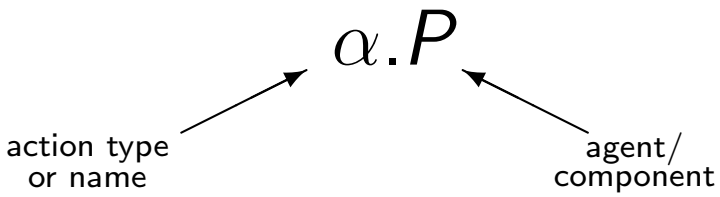


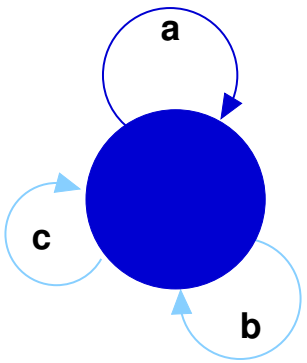


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process Algebra

- Models consist of agents which engage in actions.



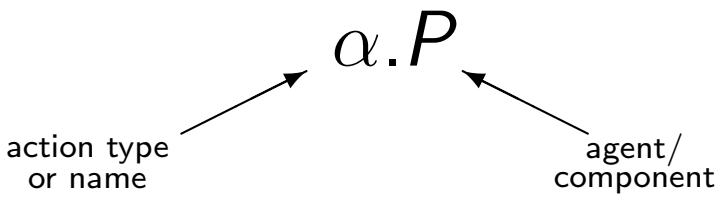


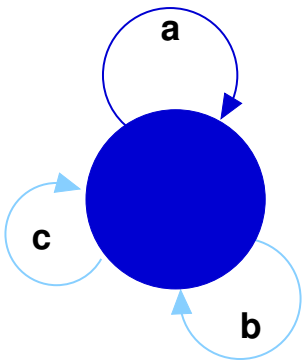
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra
model

Process Algebra

- Models consist of agents which engage in actions.



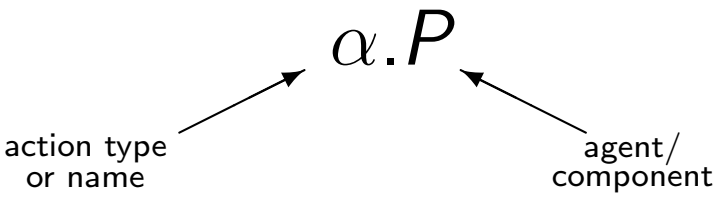


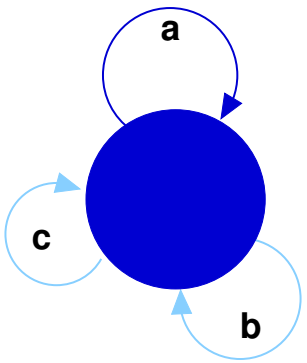
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

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

Process Algebra

- Models consist of agents which engage in actions.



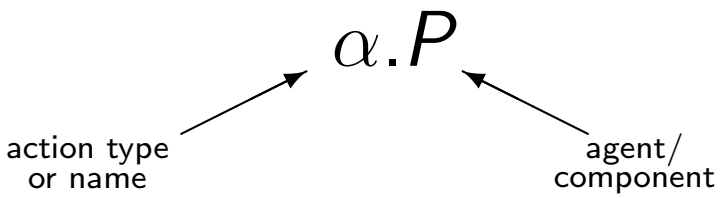


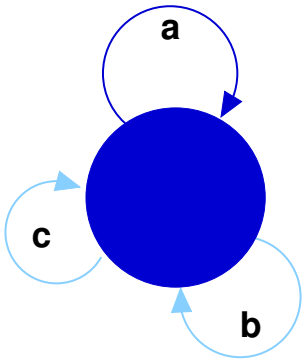
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model → SOS rules → Labelled transition system

Process Algebra

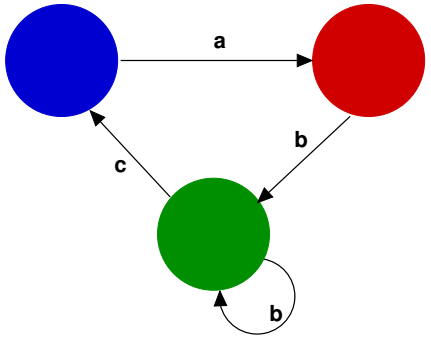
- Models consist of agents which engage in actions.





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

Process algebra model → SOS rules → Labelled transition system



A simple example: processors and resources

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

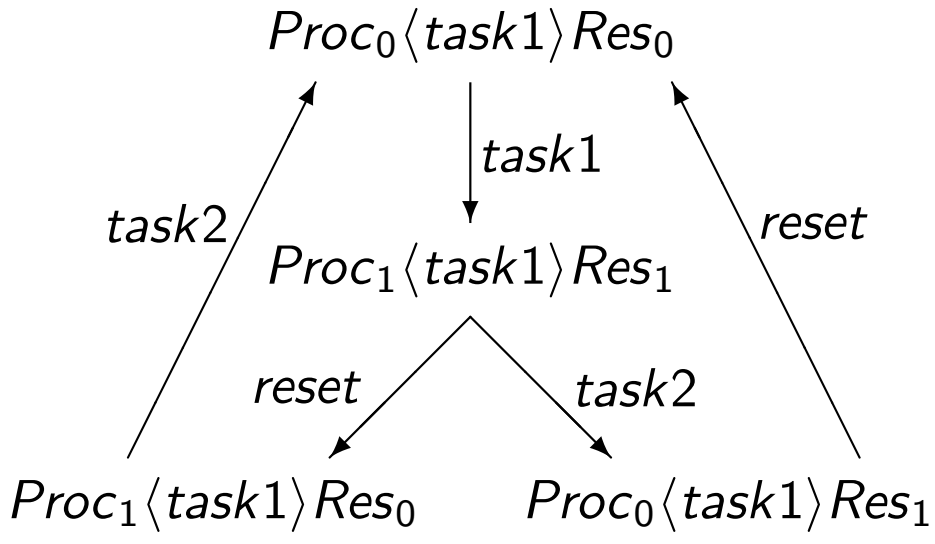
$$
\begin{aligned}
Res_0 &\stackrel{def}{=} task1.Res_1 \\
Res_1 &\stackrel{def}{=} reset.Res_0
\end{aligned}
$$

$$Proc_0 \langle task1 \rangle Res_0$$

A simple example: processors and resources

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

$$Proc_0 \langle task1 \rangle Res_0$$

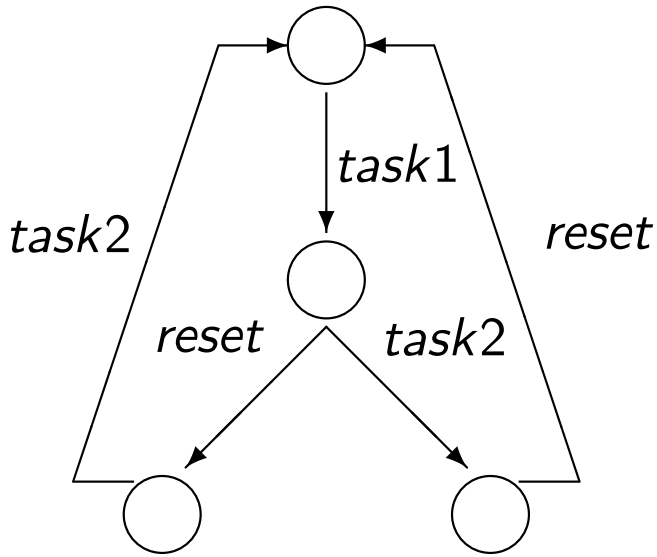


A simple example: processors and resources

$$Proc_0 \stackrel{def}{=} task1.Proc_1$$
$$Proc_1 \stackrel{def}{=} task2.Proc_0$$

$$Res_0 \stackrel{def}{=} task1.Res_1$$
$$Res_1 \stackrel{def}{=} reset.Res_0$$

$$Proc_0 \langle task1 \rangle Res_0$$

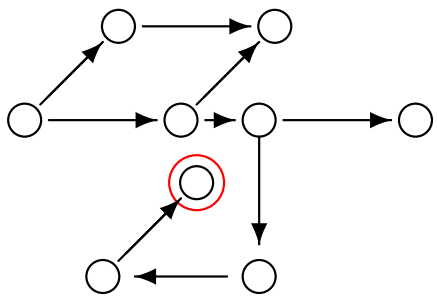


Qualitative Analysis

Qualitative Analysis

Reachability analysis

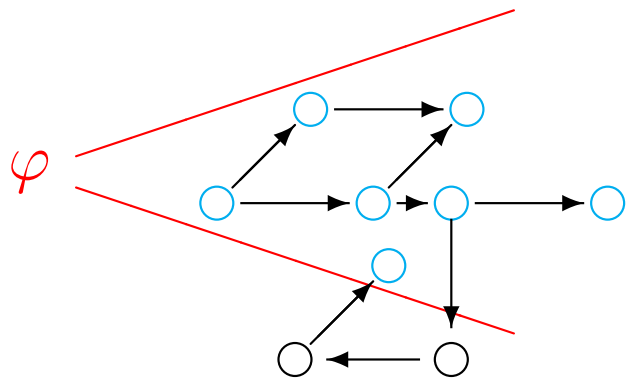
Will the system arrive in a particular state?



Qualitative Analysis

Model checking

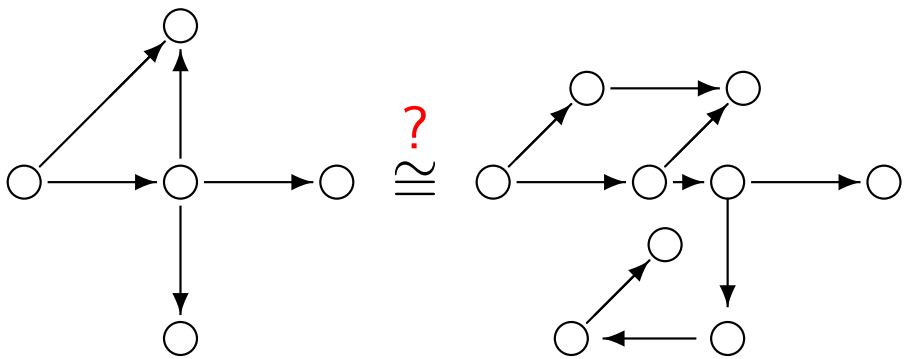
Does a given property φ hold within the system?



Qualitative Analysis

Specification matching

Does system behaviour match its specification?



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.

Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

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.

Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer and communication systems throughout 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.

Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer and communication systems throughout 20th century.

These capture the behaviour of a system as a Continuous Time Markov Chain (CTMC).

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.

Availability and reliability modelling consider the dynamic behaviour of systems with failures and breakdowns.

Markovian-based discrete event models have been applied to computer and communication systems throughout 20th century.

These capture the behaviour of a system as a Continuous Time Markov Chain (CTMC).

A CTMC is analogous to a labelled transition system or automaton which has additional information about the timing and probability associated with each action.

Formal Approaches to Quantitative Modelling

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

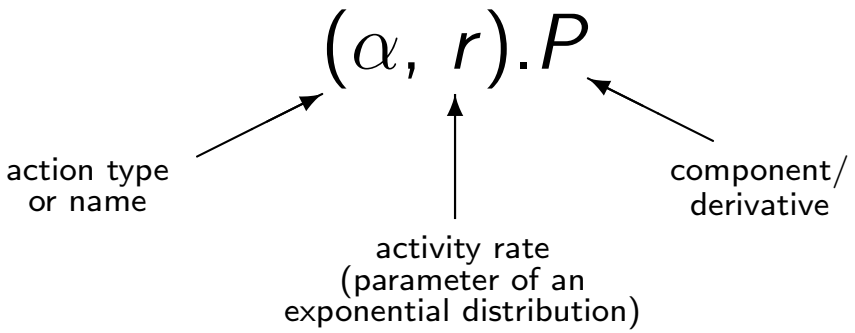
Formal Approaches to Quantitative Modelling

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

So just as with automata models, formal modelling techniques, now enhanced with information about timing and probability, are used to automatically generate the CTMCs of interest.

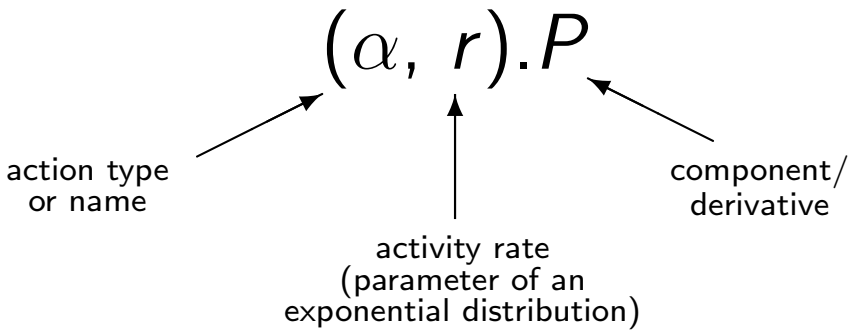
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



Stochastic Process Algebra

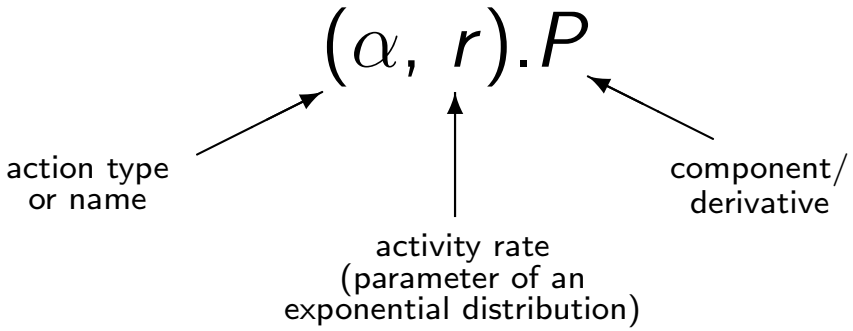
- Models are constructed from components which engage in activities.



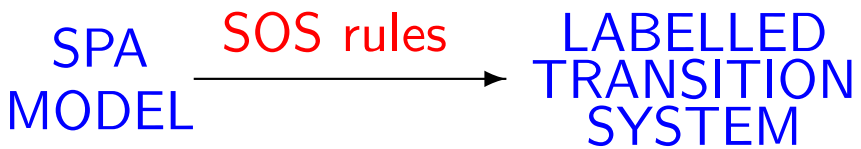
- The language is used to generate a CTMC for performance modelling.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

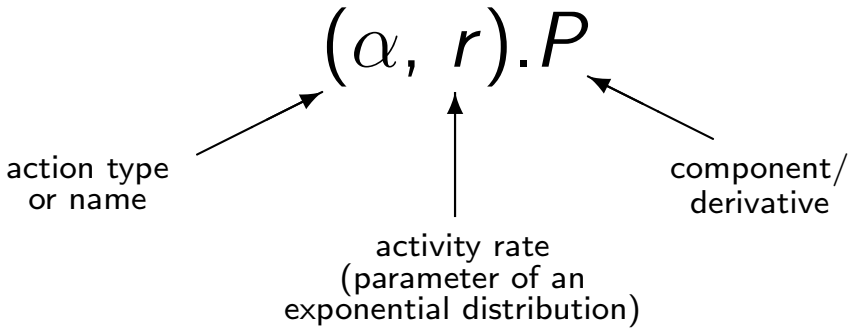


- The language is used to generate a CTMC for performance modelling.



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

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

Performance Evaluation Process Algebra

$(\alpha, f).P$	Prefix — designated first action
$P_1 + P_2$	Choice — alternative behaviours
$P_1 \underset{L}{\bowtie} P_2$	Co-operation — shared/constrained action
P/L	Hiding — abstraction
X	Constant — naming of behaviours

$P_1 \parallel P_2$ 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.

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

Processors and resources example revisited

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

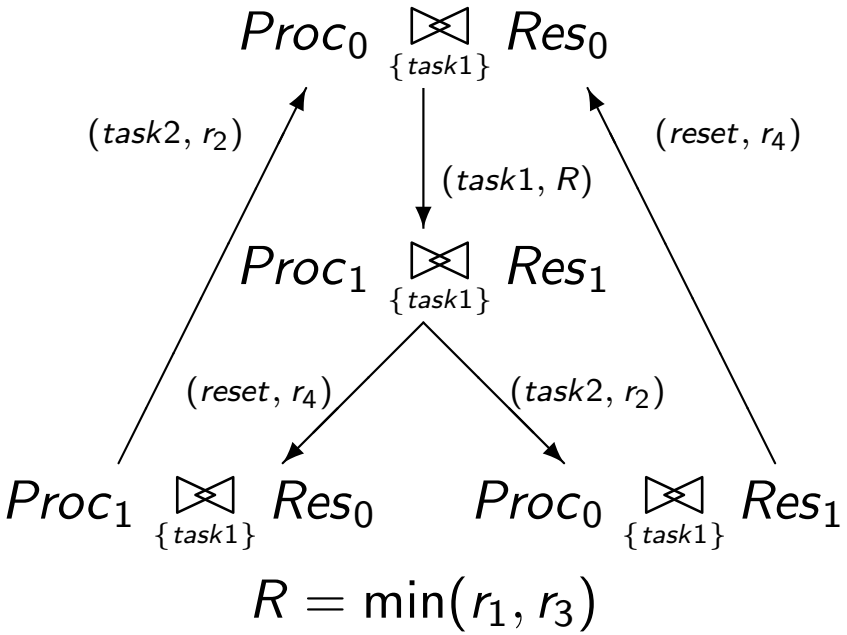
$$Proc_0 \underset{\{task1\}}{\bowtie} Res_0$$

Processors and resources example revisited

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

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

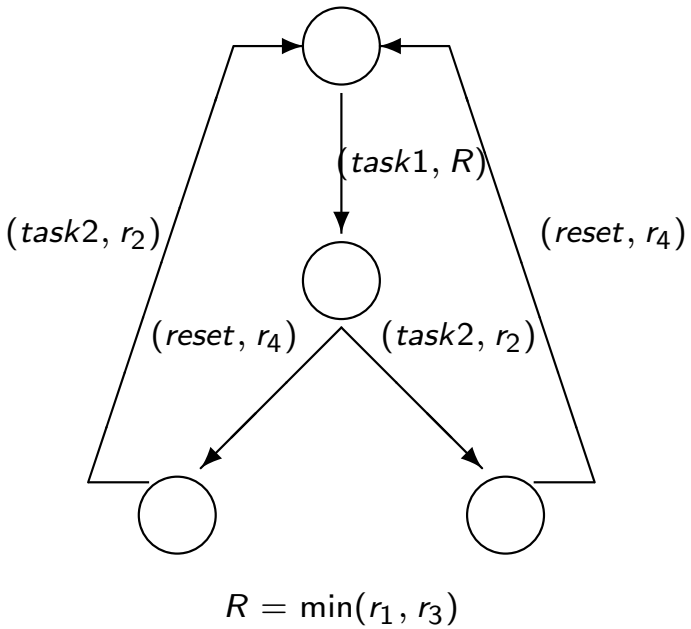


Processors and resources example revisited

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

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

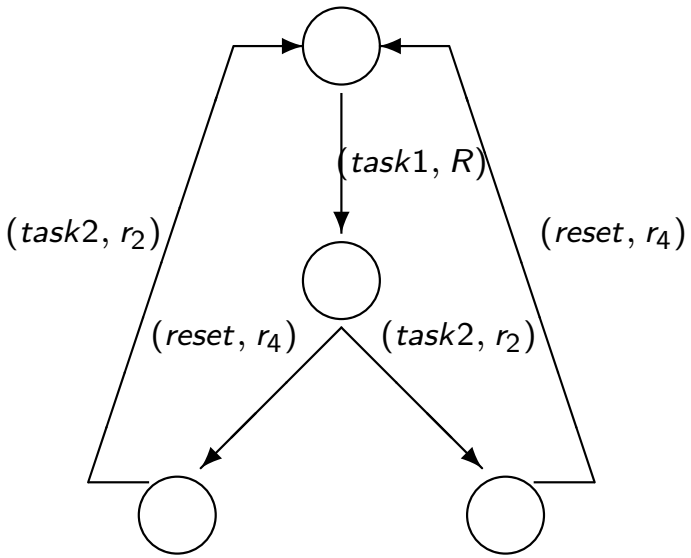


Processors and resources example revisited

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

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



$R = \min(r_1, r_3)$

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

Integrated analysis

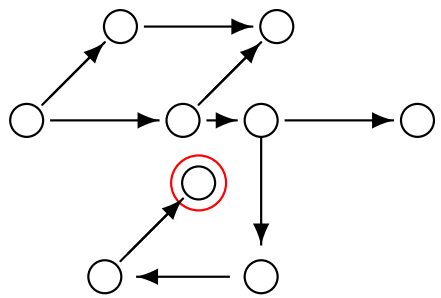
Qualitative verification can now be complemented by quantitative verification.

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?

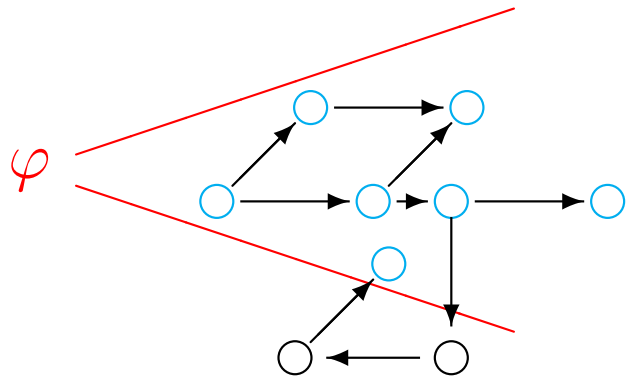


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

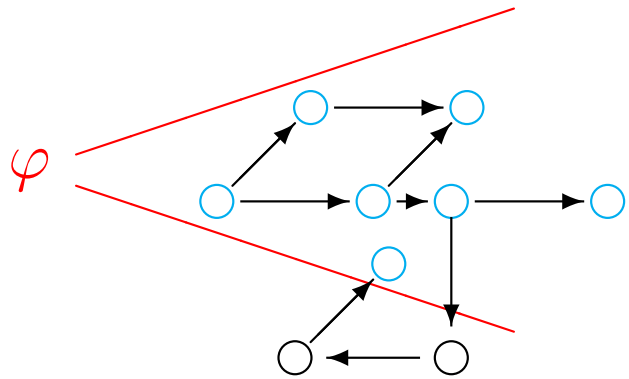


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

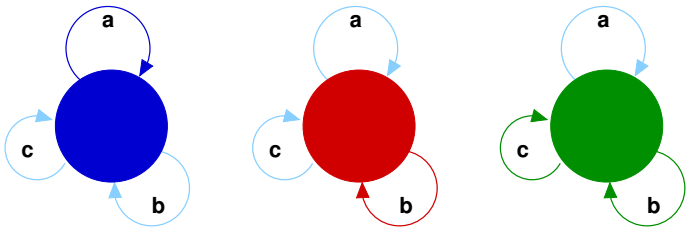
Model checking

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



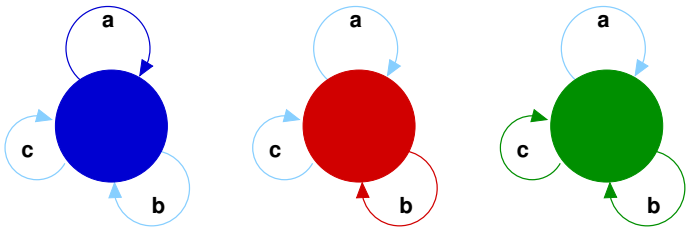
Solving discrete state models

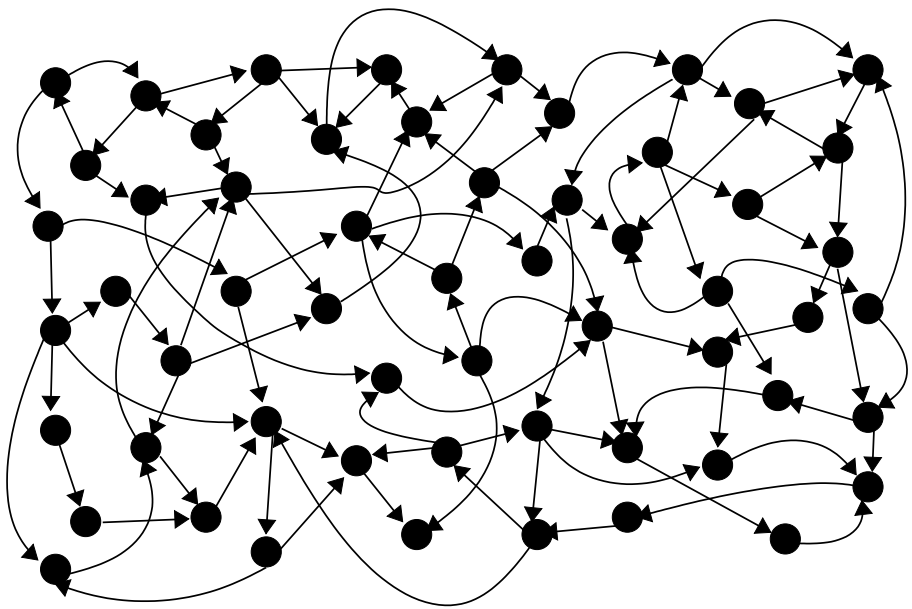
Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.



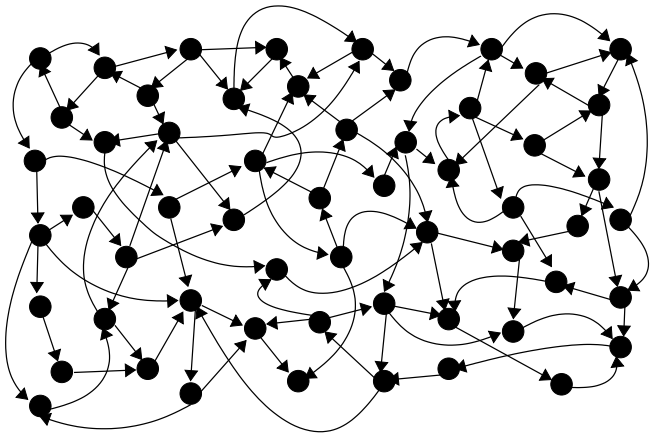
Solving discrete state models

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.



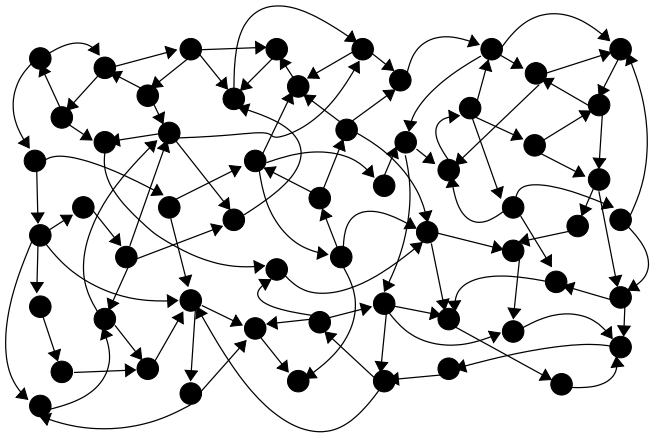


Solving discrete state models



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.

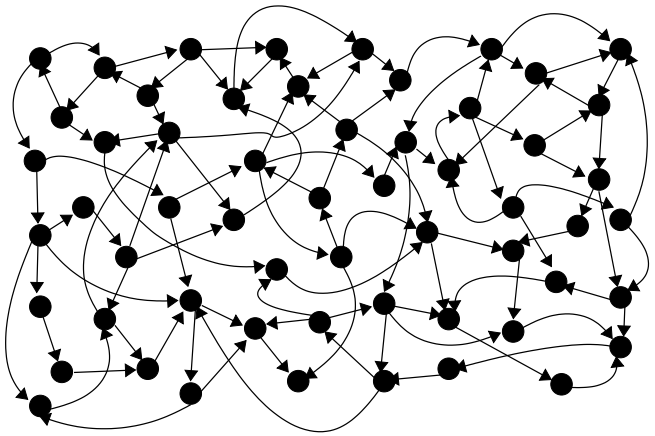
Solving discrete state models



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.

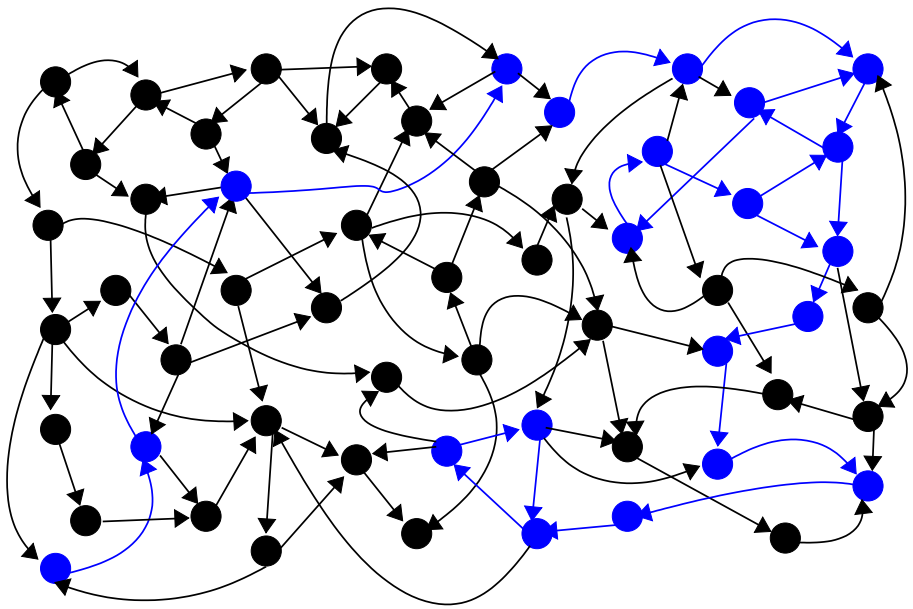
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

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

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

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.

State space explosion illustrated

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

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

State space explosion illustrated

$$Proc_0 \stackrel{def}{=} (task1, r_1).Proc_1$$
$$Proc_1 \stackrel{def}{=} (task2, r_2).Proc_0$$

$$Res_0 \stackrel{def}{=} (task1, r_3).Res_1$$
$$Res_1 \stackrel{def}{=} (reset, r_4).Res_0$$

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

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

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

For a large class of models, just as the size of the state space becomes unmanageable, the models become amenable to an efficient, scale-free approximation.

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

For a large class of models, just as the size of the state space becomes unmanageable, the models become amenable to an efficient, scale-free approximation.

These are models which consist of populations.

Outline

Collective Dynamics

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

Collective Dynamics

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

In these systems the entities within populations have limited or only competing interactions but seek to cooperate with/use entities in other populations.

Collective Dynamics

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

In these systems the entities within populations have limited or only competing interactions but seek to cooperate with/use entities in other populations.

For such systems we are often as interested in the population level behaviour as we are in the behaviour of the individual entities.

Collective Dynamics

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

In these systems the entities within populations have limited or only competing interactions but seek to cooperate with/use entities in other populations.

For such systems we are often as interested in the population level behaviour as we are in the behaviour of the individual entities.

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.

Process Algebra and Collective Dynamics

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

Process Algebra and Collective Dynamics

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

- Developed to represent concurrent behaviour compositionally;

Process Algebra and Collective Dynamics

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

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

Process Algebra and Collective Dynamics

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

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

Process Algebra and Collective Dynamics

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

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

Process Algebra and Collective Dynamics

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

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

The major challenge to analysing such models is the challenge of state space explosion.

Example Applications

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.

Spread of viruses and malware

Improved modelling of networks under attack could lead to improved detection and better security in computer systems.

Example Applications

Biochemical signalling pathways

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

Crowd dynamics

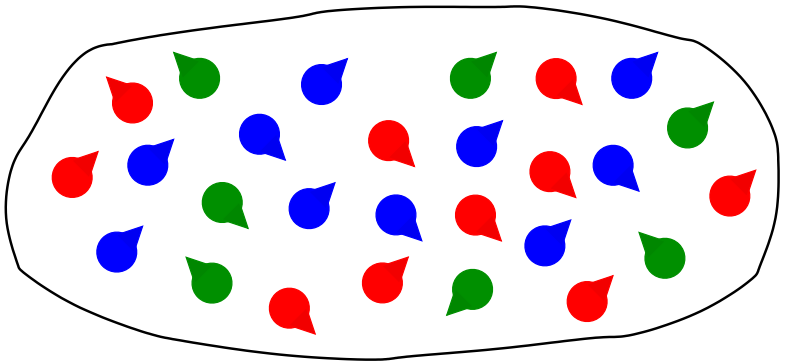
Technology enhancement is creating new possibilities for directing crowd movements in buildings and urban spaces, for example for emergency egress, which are not yet well-understood.

Identity and Individuality

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

Identity and Individuality

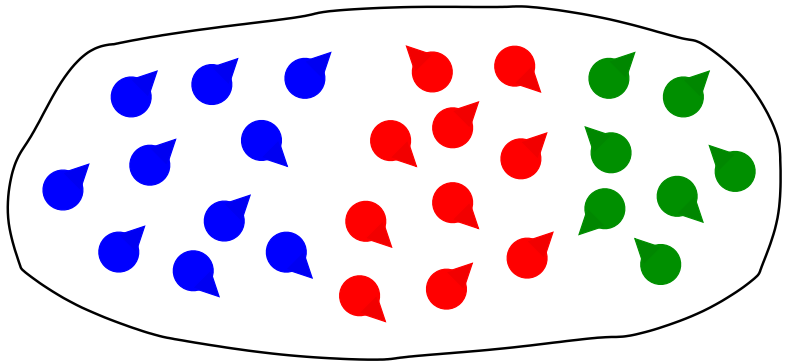
Population systems are constructed from many instances of a set of components.



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

Identity and Individuality

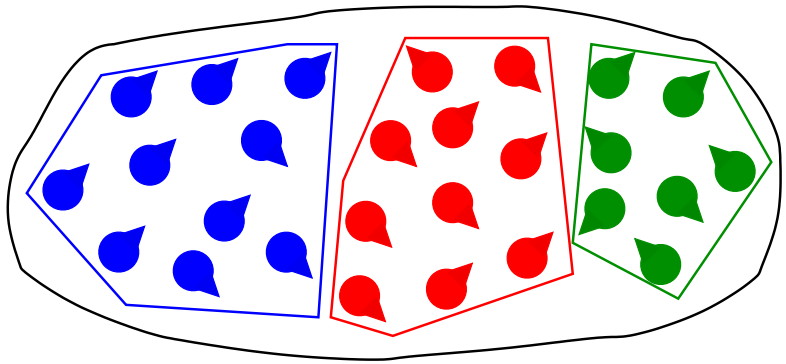
Population systems are constructed from many instances of a set of components.



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

Identity and Individuality

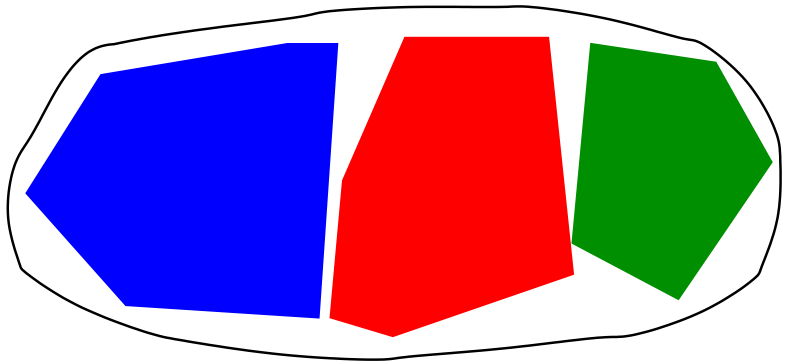
Population systems are constructed from many instances of a set of components.



Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space. We now disregard the identity of components.

Identity and Individuality

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



Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space. We now disregard the identity of components.

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

Population models

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 models

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 models

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 models

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 or fluid approximation of how the proportions vary over time.

Fluid Approximation

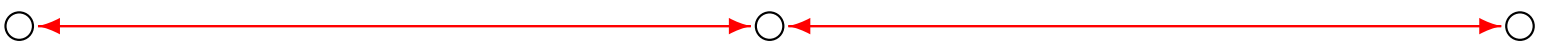
Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

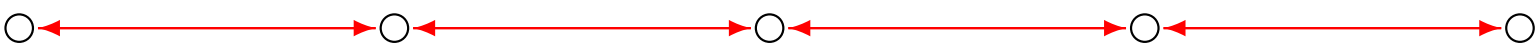


Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

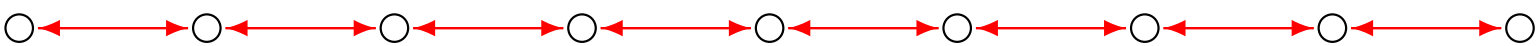


Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

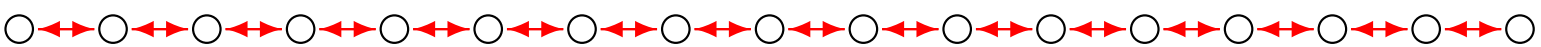


Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.



Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Fluid Approximation

Although in reality all state transitions are discrete, we can see that as the size of the population grows, the impact of each state change becomes smaller, and the error introduced by continuous approximation decreases.

Simple example revisited

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

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

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

Simple example revisited

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

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

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

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

Simple example revisited

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

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

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

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

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

Simple example revisited

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

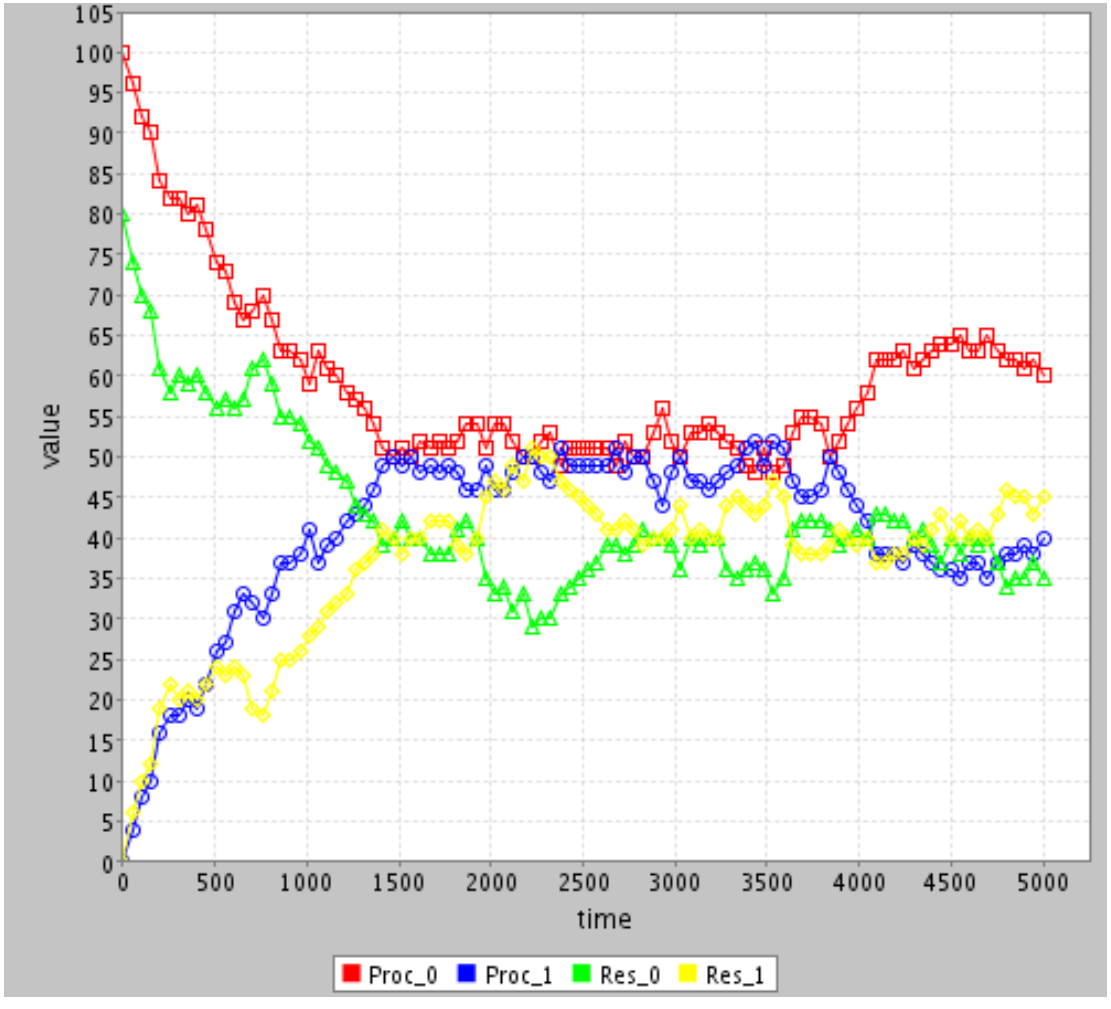
$$
\begin{aligned}
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

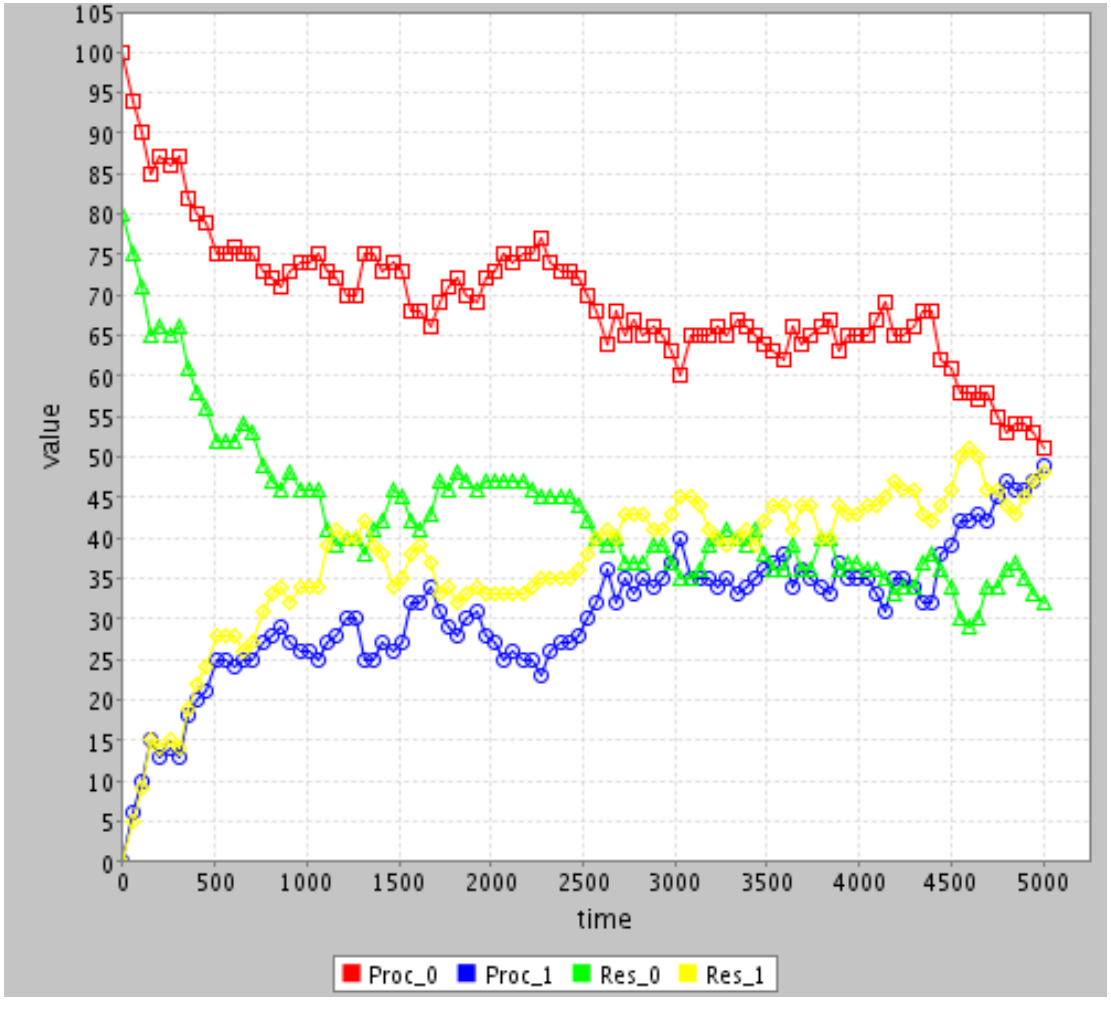
ODE interpretation

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

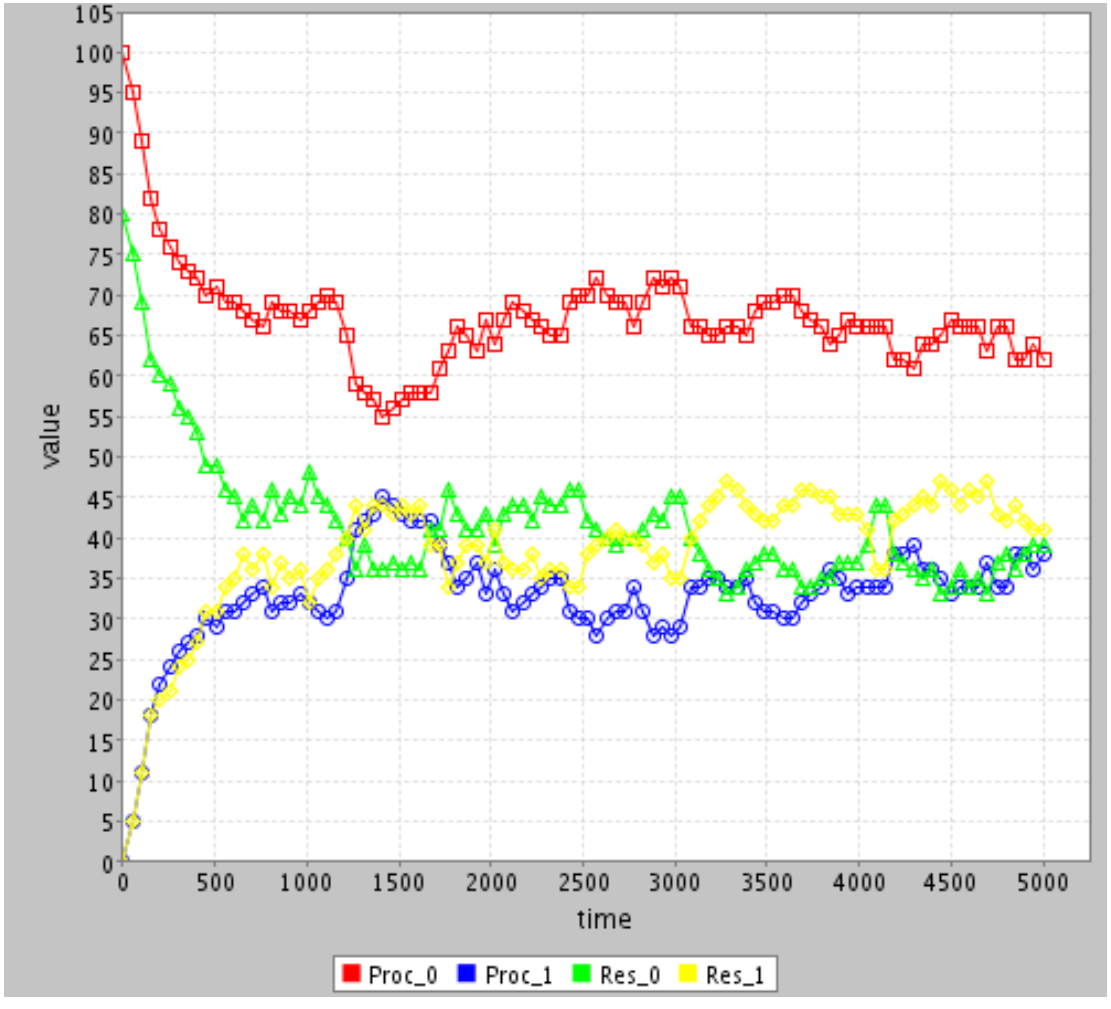
100 processors and 80 resources (simulation run A)



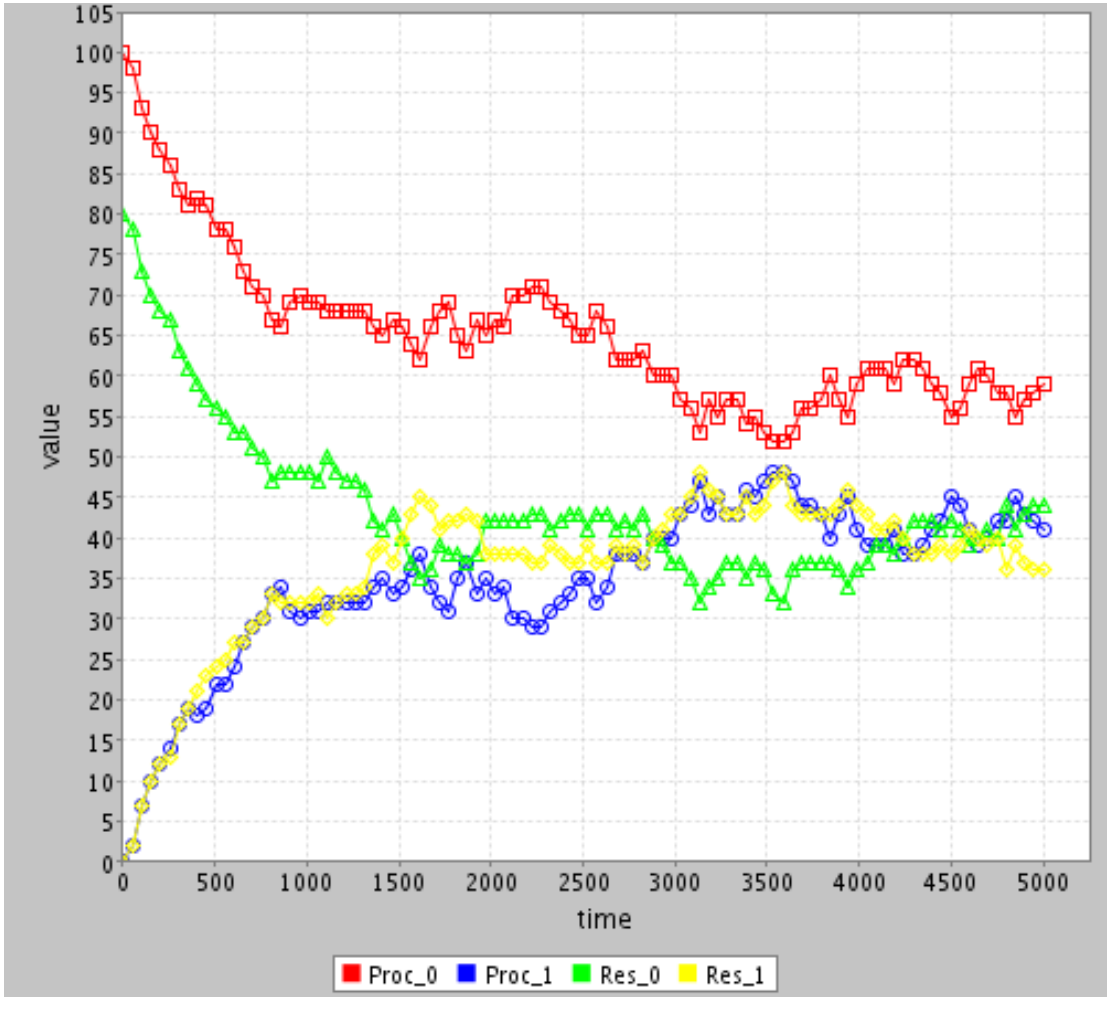
100 processors and 80 resources (simulation run B)



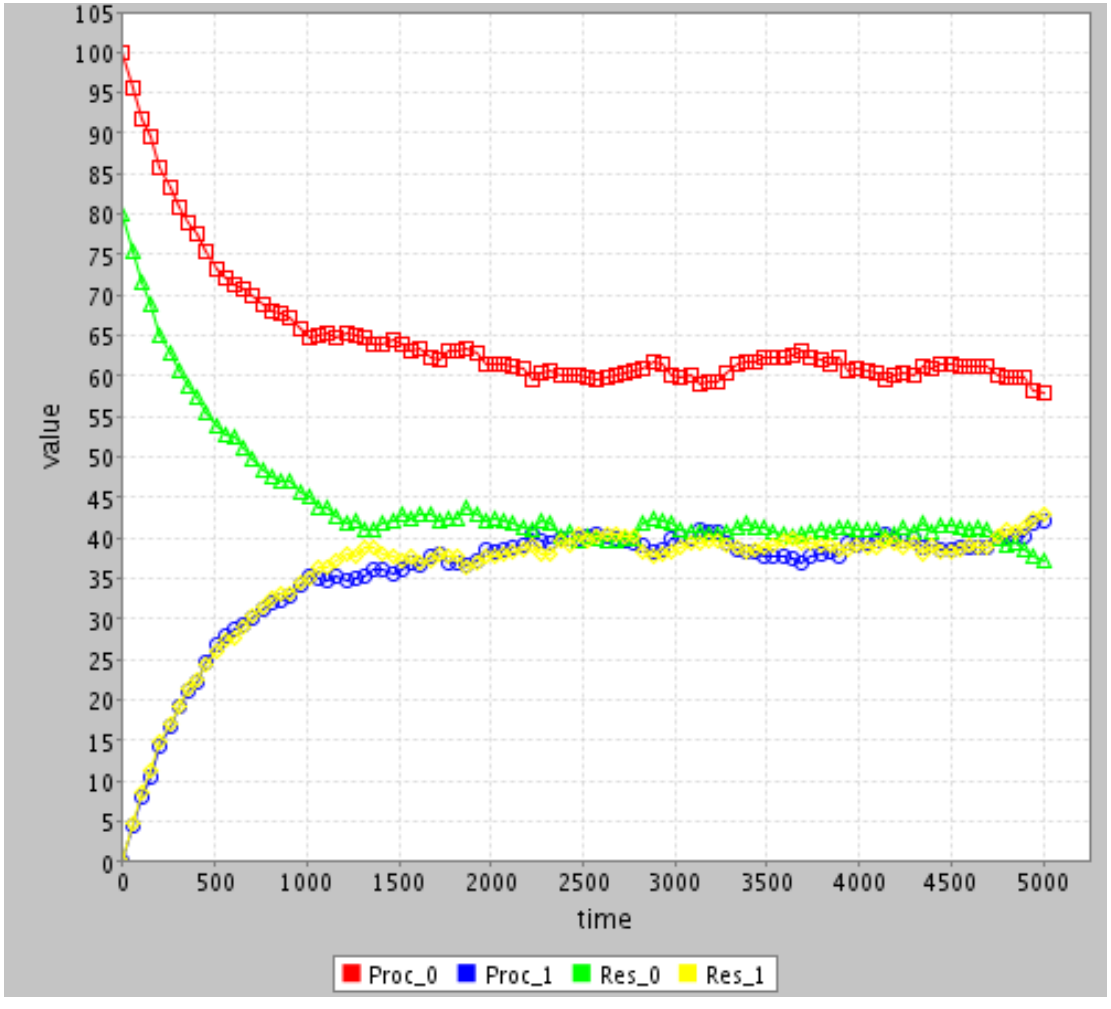
100 processors and 80 resources (simulation run C)



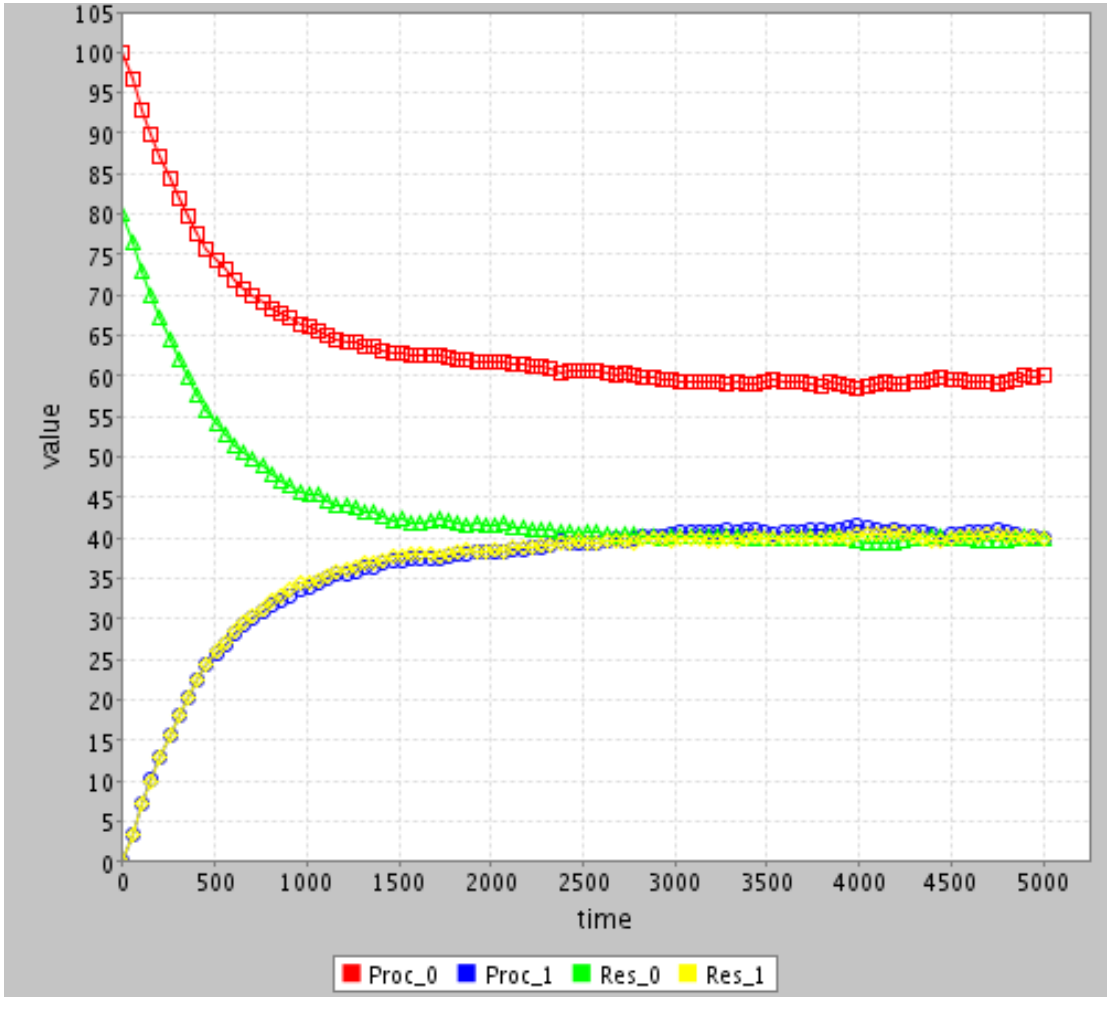
100 processors and 80 resources (simulation run D)



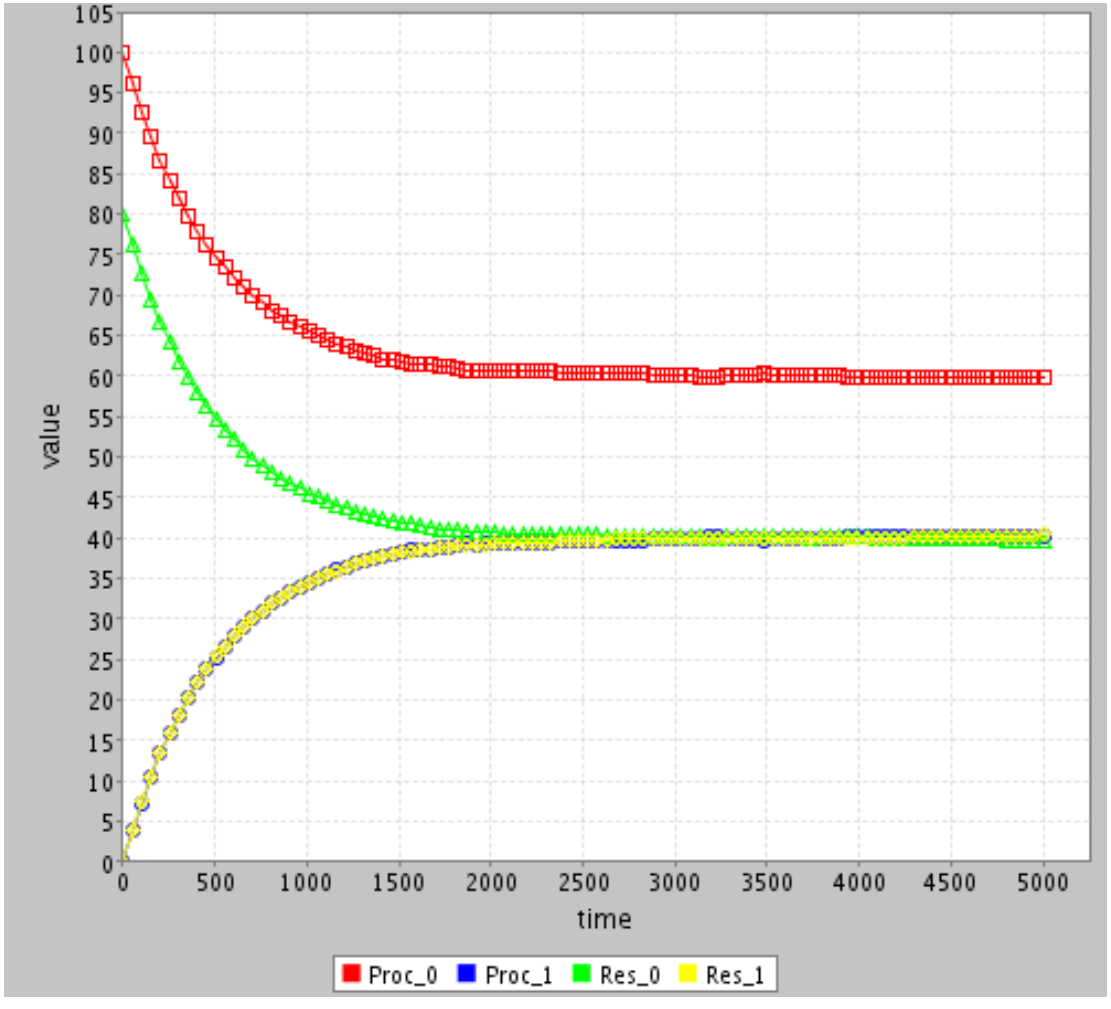
100 processors and 80 resources (average of 10 runs)



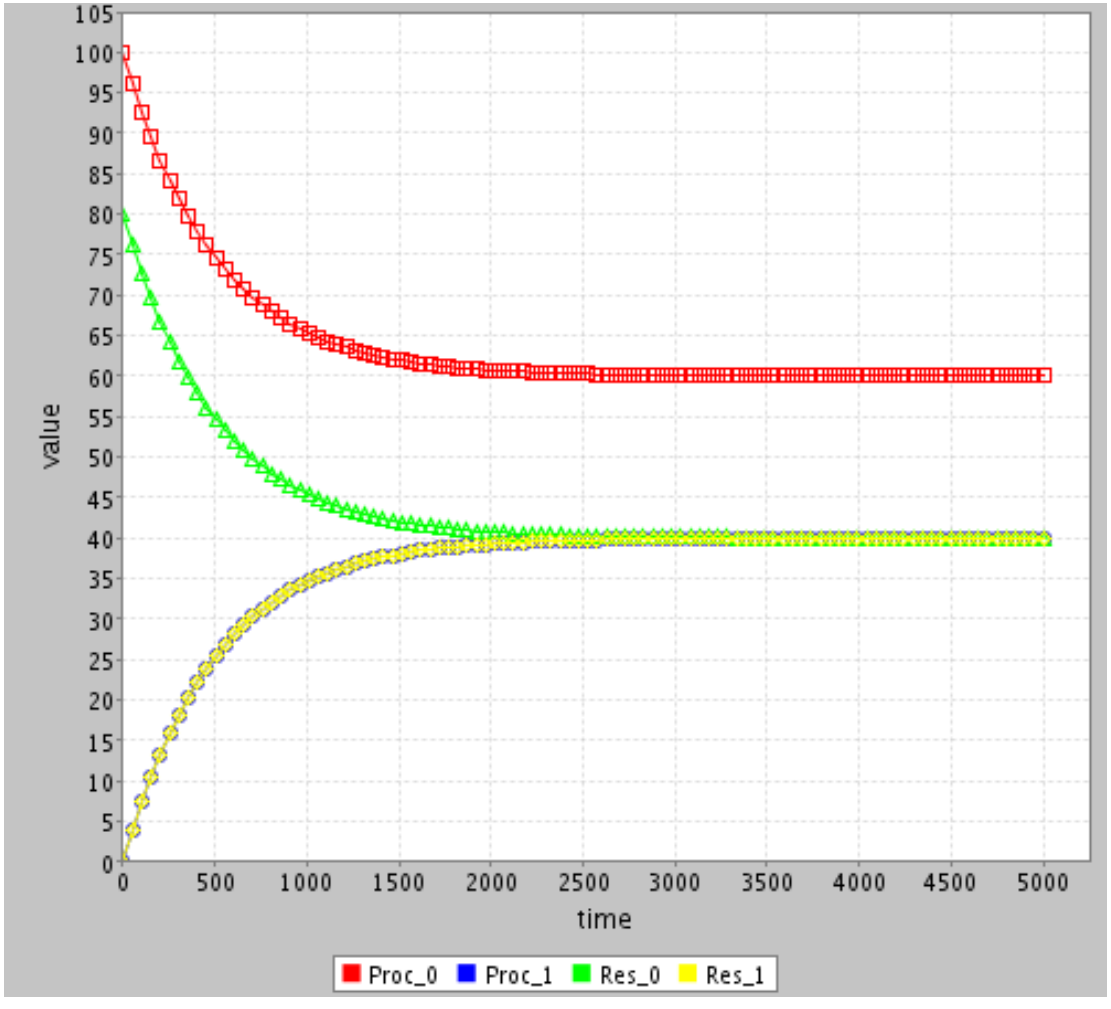
100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (ODE solution)



Outline

Pragmatism and Expediency

Through pragmatism and expediency the representation of inherently discrete systems by collections of ordinary differential equations has been adopted in many areas of science, e.g. cell biology, ecology and epidemiology.

Pragmatism and Expediency

Through pragmatism and expediency the representation of inherently discrete systems by collections of ordinary differential equations has been adopted in many areas of science, e.g. cell biology, ecology and epidemiology.

However we would like our approximation to be rigorous in the sense of being

- based on sound mathematical foundations, and
- systematically derived from the system description

Pragmatism and Expediency

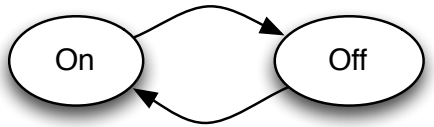
Through pragmatism and expediency the representation of inherently discrete systems by collections of ordinary differential equations has been adopted in many areas of science, e.g. cell biology, ecology and epidemiology.

However we would like our approximation to be rigorous in the sense of being

- based on sound mathematical foundations, and
- systematically derived from the system description

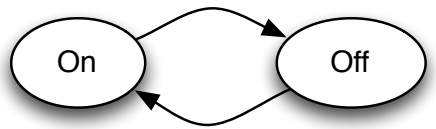
Fortunately, Kurtz's Deterministic Approximation Theorem, tells the circumstances under which it is valid to approximate a family of population CTMC models by a set of ODEs when the population grows to infinity.

Population models — intuition



$Y(t)$

Population models — intuition

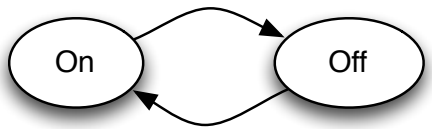


$Y(t)$

N copies: $Y_i^{(N)}$

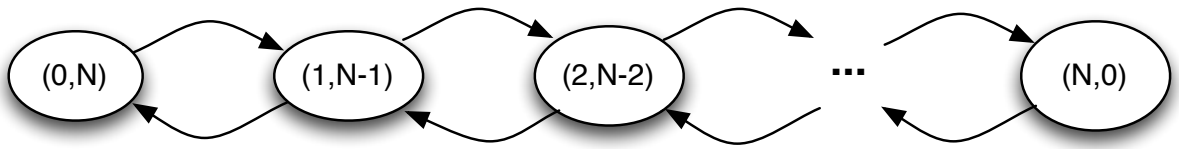
2^N states

Population models — intuition



$Y(t)$

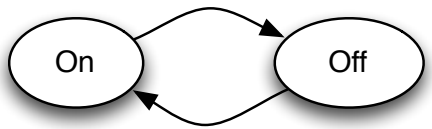
N copies: $Y_i^{(N)}$
2^N states



$\mathbf{X}^{(N)}(t)$

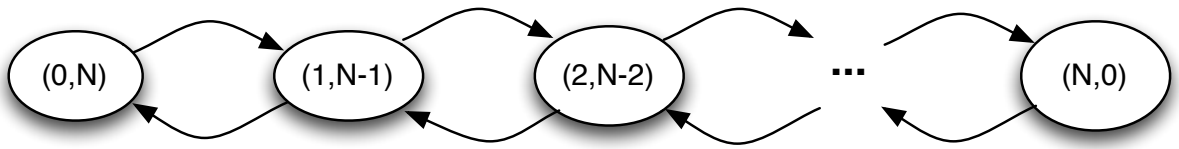
$N+1$ states

Population models — intuition



$Y(t)$

N copies: $Y_i^{(N)}$
2^N states

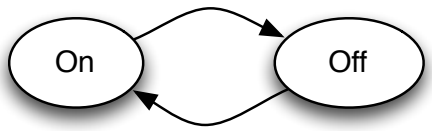


$\mathbf{X}^{(N)}(t)$

$N+1$ states

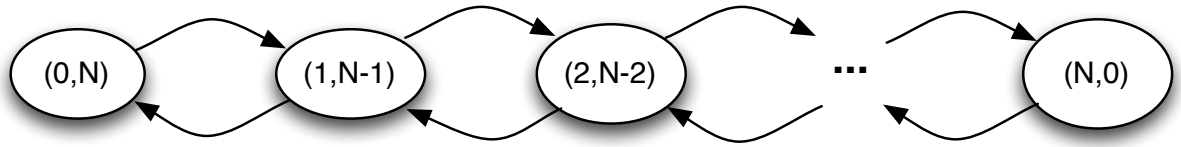
$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

Population models — intuition



$Y(t)$

N copies: $Y_i^{(N)}$
2^N states



$\mathbf{X}^{(N)}(t)$

$N+1$ states

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

- $Y(t)$, $Y_i^{(N)}(t)$ and $\mathbf{X}^{(N)}(t)$ are all CTMCs;
- As N increases we get a sequence of CTMCs, $\mathbf{X}^{(N)}(t)$

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.
- Each transition is specified by a rate function $r_\tau^{(N)}$, and by an update vector $\mathbf{v}_\tau$, specifying the impact of the event on the population vector.

Population transitions

- The dynamics of the population models is expressed in terms of a set of possible transitions, $\mathcal{T}^{(N)}$.
- Transitions are stochastic, and take an exponentially distributed time to happen.
- Their rate may depend on the current global state of the system.
- Each transition is specified by a rate function $r_\tau^{(N)}$, and by an update vector $\mathbf{v}_\tau$, specifying the impact of the event on the population vector.
- The infinitesimal generator matrix $Q^{(N)}$ of $\mathbf{X}^{(N)}(t)$ is defined as:

$$q_{\mathbf{x},\mathbf{x}'} = \sum \{ r_\tau(\mathbf{x}) \mid \tau \in \mathcal{T},\ \mathbf{x}' = \mathbf{x} + \mathbf{v}_\tau \}.$$

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \ldots, n\}$ in the system.

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \ldots, n\}$ in the system.

System variables

$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$, and $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$

Population models — summary of notation

Individuals

We have N individuals $Y_i^{(N)} \in S$, $S = \{1, 2, \ldots, n\}$ in the system.

System variables

$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$, and $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$

Dynamics (system level)

$\mathbf{X}^{(N)}$ is a CTMC with transitions $\tau \in \mathcal{T}$:

$$\tau\text{: } \mathbf{X}^{(N)} \text{ to } \mathbf{X}^{(N)} + \mathbf{v}_\tau \text{ at rate } r_\tau^{(N)}(\mathbf{X})$$

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{x}} = \frac{\mathbf{X}}{N}$$

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N} \qquad \text{occupancy measures}$$

Scaling Conditions

Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N} \qquad \text{occupancy measures}$$

- for each $\tau \in \mathcal{T}^{(N)}$
  - the normalised update is $\hat{\mathbf{v}} = \mathbf{v}/N$
  - there is a normalised rate function $\hat{r}_\tau(\hat{\mathbf{X}})$

Scaling Conditions

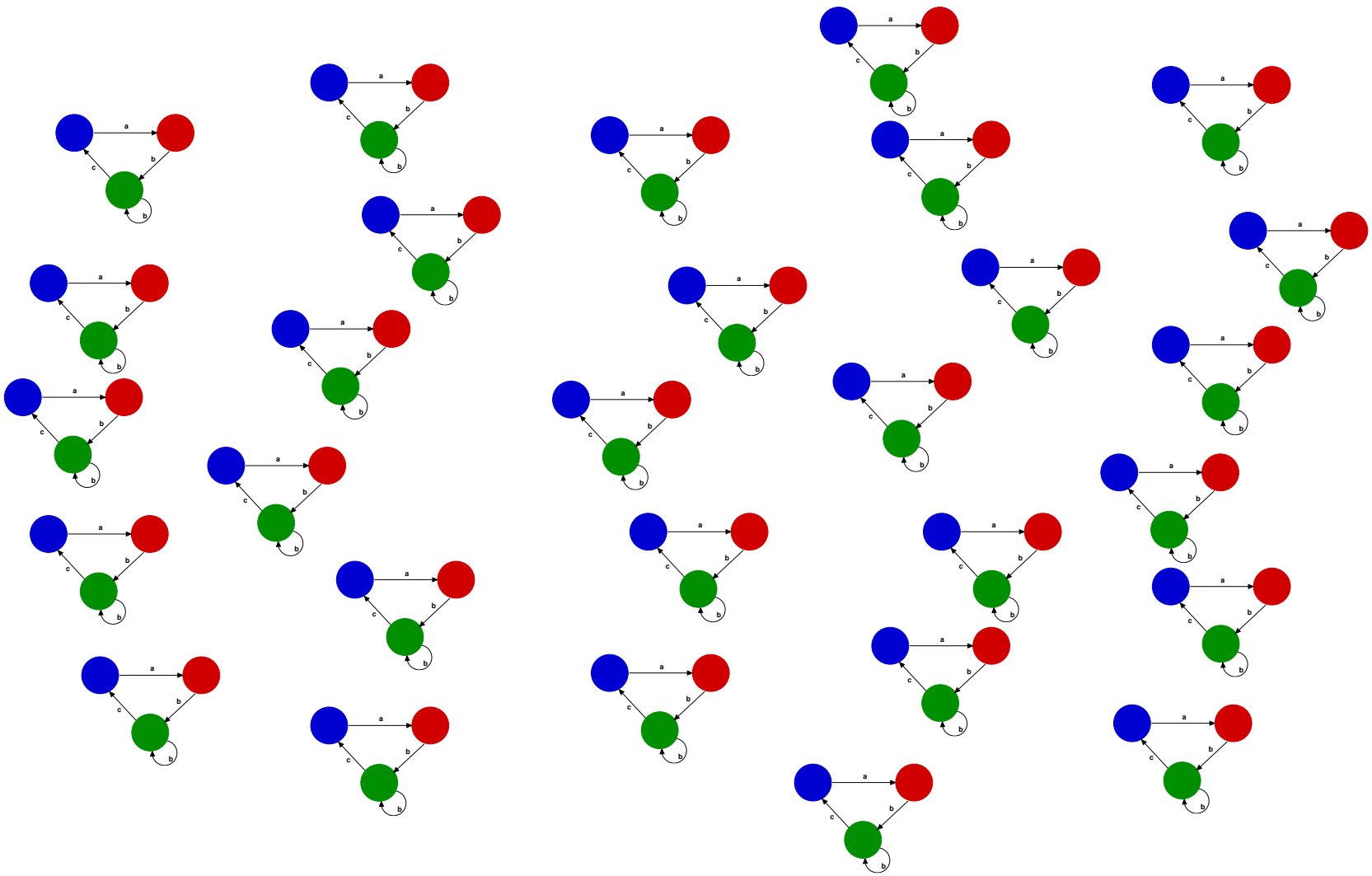
Scaling assumptions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs.
- We normalise such models, dividing variables by N:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{N} \quad \text{occupancy measures}$$

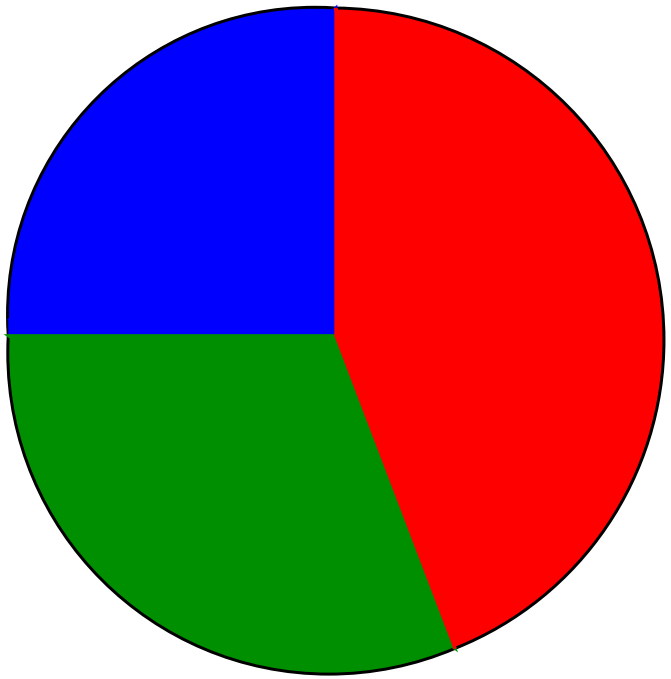
- for each $\tau \in \mathcal{T}^{(N)}$
  - the normalised update is $\hat{\mathbf{v}} = \mathbf{v}/N$
  - there is a normalised rate function $\hat{r}_\tau(\hat{\mathbf{X}})$
- $\forall \tau$ assume there exists a bounded and Lipschitz continuous function $f_\tau(\hat{\mathbf{X}})$, the limit rate function on normalised variables, independent of N, such that $\frac{1}{N}\,\hat{r}_\tau^{(N)}(\mathbf{x}) \to f_\tau(\mathbf{x})$ uniformly as $N \longrightarrow \infty$.

Normalised process — intuition



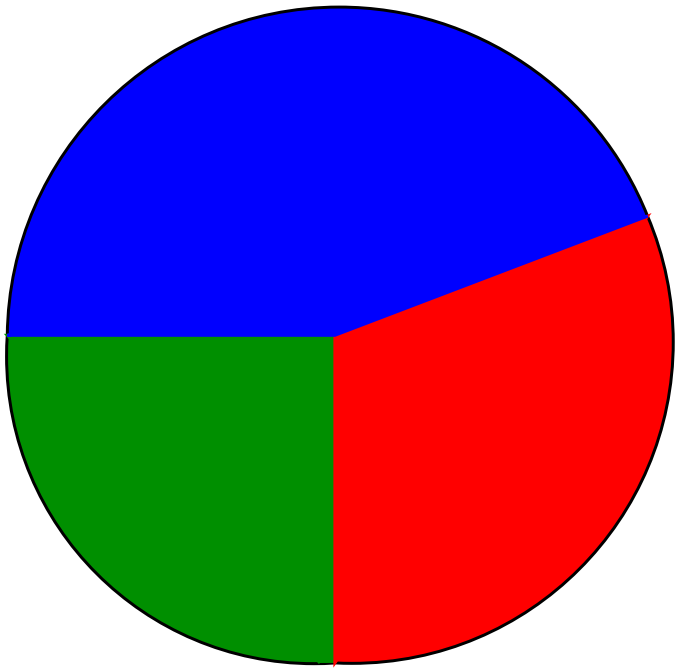
Normalised process — intuition

The whole population is represented as a single process.



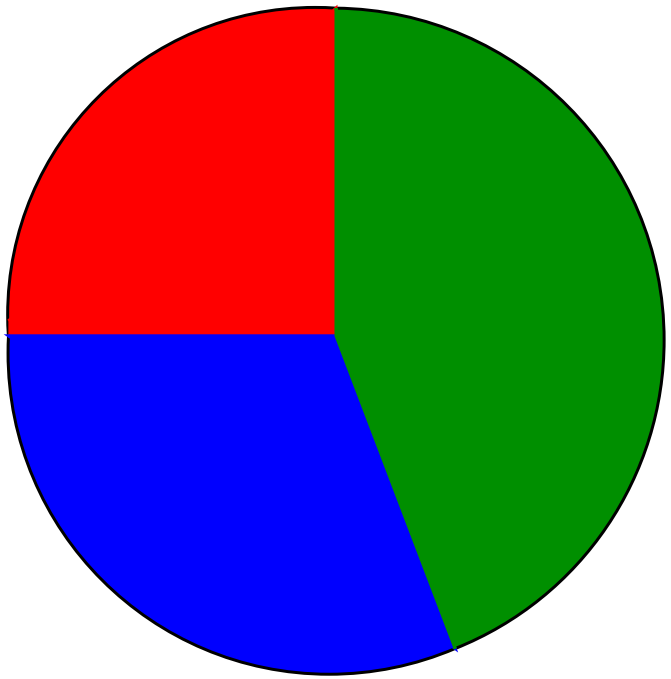
Normalised process — intuition

The whole population is represented as a single process.



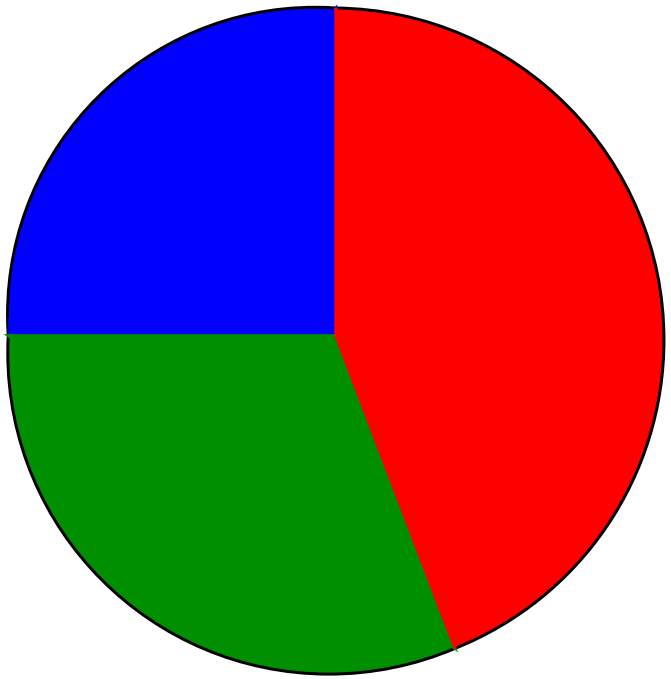
Normalised process — intuition

The whole population is represented as a single process.



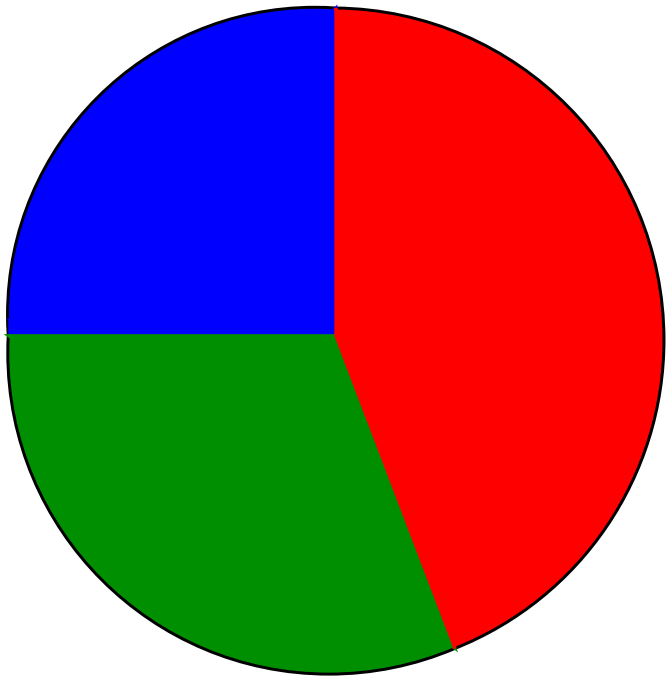
Normalised process — intuition

The whole population is represented as a single process.



Normalised process — intuition

The whole population is represented as a single process.



Even when the number of individuals varies ($N \longrightarrow \infty$) the processes remain comparable.

Dynamics

Drift

The drift $F^{(N)}(\hat{\mathbf{X}})$ — the mean instantaneous increment of model variables in state $\hat{\mathbf{X}}$ — is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$$

Dynamics

Drift

The drift $F^{(N)}(\hat{\mathbf{X}})$ — the mean instantaneous increment of model variables in state $\hat{\mathbf{X}}$ — is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$$

Limit Drift

Let f_τ be the limit rate functions.

The limit drift of the model $\hat{\mathcal{X}}^{(N)}$ is

$$F(\mathbf{x}) = \sum_{\tau \in \hat{\mathcal{T}}} \mathbf{v}_\tau f_\tau(\mathbf{x}),$$

and $F^{(N)}(\mathbf{x}) \to F(\mathbf{x})$ uniformly as $N \longrightarrow \infty$.

Fluid approximation theorem and fluid ODEs

Deterministic Approximation Theorem (Kurtz)

Assume that $\exists\, \mathbf{x_0} \in S$ such that $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ in probability. Then, for any finite time horizon $T < \infty$, it holds that as $N \longrightarrow \infty$:

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

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

Fluid ODE

The fluid ODE is

$$\frac{d\mathbf{x}}{dt} = F(\mathbf{x}), \quad \textit{with } \mathbf{x}(0) = \mathbf{x_0} \in S.$$

Since F is Lipschitz (all f_τ are), this ODE has a unique solution $\mathbf{x}(t)$ starting from $\mathbf{x_0}$.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

We focus on the second interpretation — a functional version of the Law of Large Numbers.

Fluid Approximation ODEs

The fluid approximation ODEs can be interpreted in two different ways:

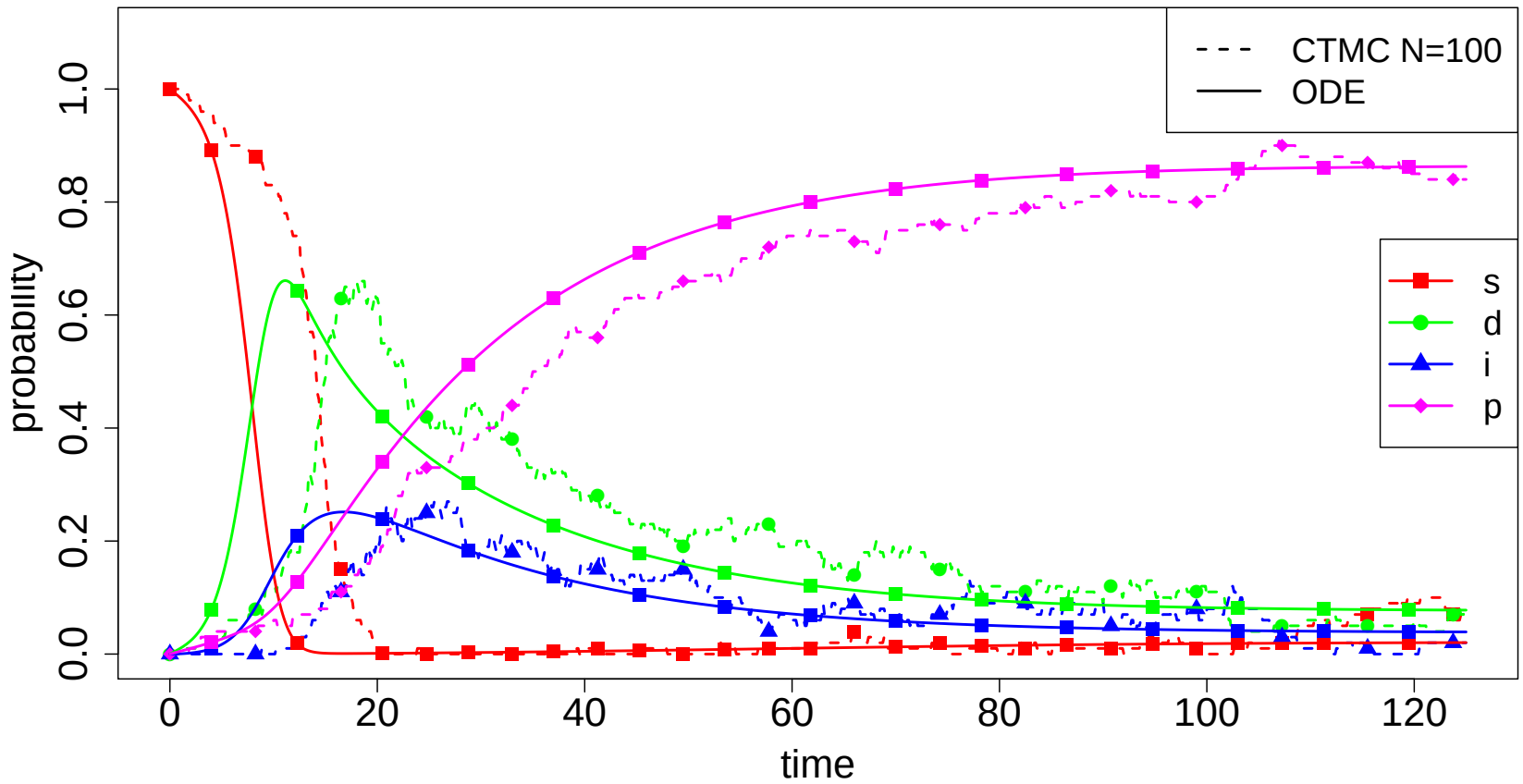
- as an approximation of the average of the system (usually a first order approximation). This is often termed a mean field approximation.
- as an approximate description of system trajectories for large populations.

We focus on the second interpretation — a functional version of the Law of Large Numbers.

Instead of having a sequence of random variables, converging to a deterministic value, here we have a sequence of CTMCs for increasing population size, which converge to a deterministic trajectory, the solution of the fluid ODE.

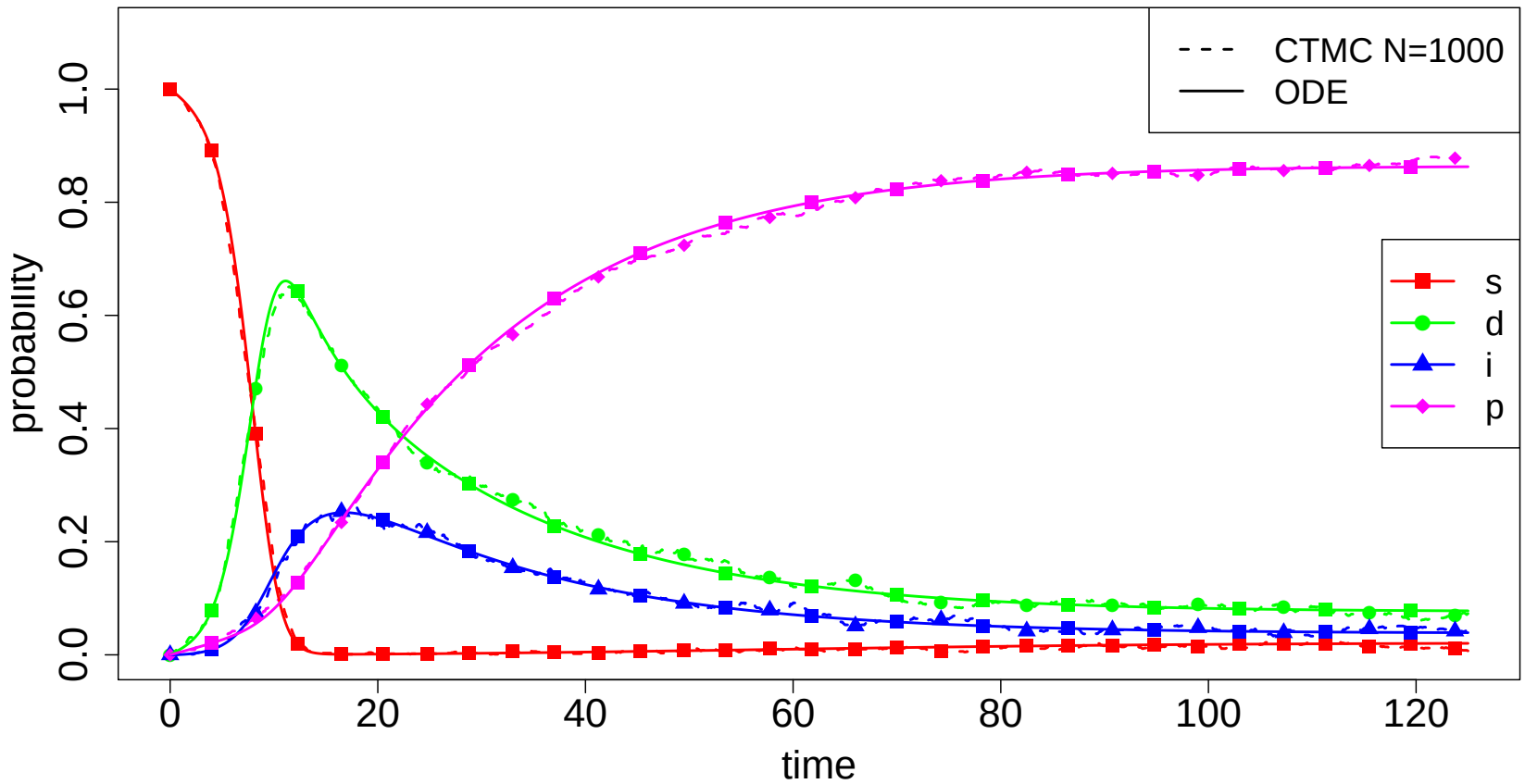
Illustrative trajectories

Limit fluid ODE and single stochastic trajectory of a network epidemics example for $N = 100$



Illustrative trajectories

Limit fluid ODE and single stochastic trajectory of a network epidemics example for $N = 1000$



Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for constructing CTMC models offers quantitative scalable analysis which is immune to state space explosion.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for constructing CTMC models offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the model grows.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for constructing CTMC models offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the model grows.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for constructing CTMC models offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the model grows.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.

Fluid semantics for Stochastic Process Algebras

- Incorporating fluid approximation into a formal high-level language used for constructing CTMC models offers quantitative scalable analysis which is immune to state space explosion.
- Indeed, accuracy increases as the size of the model grows.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.
- Moreover the derivation of the ODEs can be automated in the implementation of the language.

Defining agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

Defining agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

Defining agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

We construct a single agent class in the population CTMC but partition the state space S into subsets, each of which represents the states of a distinct component, and such that there are no transitions between subsets.

Defining agents

Kurtz's Theorem is based on the notion of a single agent class — many instances of one sequential component.

But in a process algebra model we typically work with multiple components composed to evolve concurrently.

We construct a single agent class in the population CTMC but partition the state space S into subsets, each of which represents the states of a distinct component, and such that there are no transitions between subsets.

The agents whose initial state is in each subset correspond to that component.

Definition transitions

The existing SOS semantics defines all the possible transitions by constructing the state space of the CTMC explicitly.

Definition transitions

The existing SOS semantics defines all the possible transitions by constructing the state space of the CTMC explicitly.

Definition transitions

The existing SOS semantics defines all the possible transitions by constructing the state space of the CTMC explicitly.

We define a structured operational semantics which defines the rates and updates associated with all possible transitions in terms of an arbitrary abstract state:

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

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

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

Apparent Rate Calculation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$f(\xi, (-1, +1, -1, +1), task1) = \min(r_1\xi_1, r_3\xi_3)$$

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

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

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

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

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

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

Extraction of the ODE from f

Differential Equation

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

$$= (-1, 1, -1, 1) \min(r_1 x_1, r_3 x_3) + (1, -1, 0, 0) r_2 x_2 + (0, 0, 1, -1) r_4 x_4$$

Differential Equation

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

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

$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = -\min(r_1 x_1, r_3 x_3) + r_4 x_4$$

$$\frac{\mathrm{d}x_4}{\mathrm{d}t} = \min(r_1 x_1, r_3 x_3) - r_4 x_4$$

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.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.

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.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.
- The generated ODEs are the fluid limit of the family of CTMCs and so approximate the discrete behaviour as the size of the system grows.

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.
- Thus the hypotheses of the Deterministic Approximation Theorem are satisfied.
- The generated ODEs are the fluid limit of the family of CTMCs and so approximate the discrete behaviour as the size of the system grows.
- 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.

Outline

Conclusions

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

Conclusions

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

Conclusions

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

Conclusions

- Collective systems of interacting populations are an interesting and challenging class of systems to design and construct.
- Their role within infrastructure, such as smart cities, make it essential that quantitive aspects of behaviour are taken into consideration, as well as functional correctness.
- Fluid approximation based analysis offers hope for scalable quantitative analysis techniques.
- Nevertheless there remain many interesting and challenging problems to be solved: populations that do not scale proportionally, systems with entities in distinct locations,...

Thanks

Thanks

Thanks to my collaborators, especially Luca Bortolussi, Stephen Gilmore and Mirco Tribastone.

This work has been funded by EPSRC through the CODA project, and by the EU through the ongoing FET-Proactive QUANTICOL project

quanticol

www.quanticol.eu