Process Algebra for Collective Dynamics

Jane Hillston

Laboratory for Foundations of Computer Science
University of Edinburgh

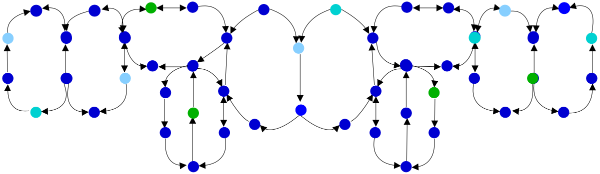
29th March 2009

Outline

Outline

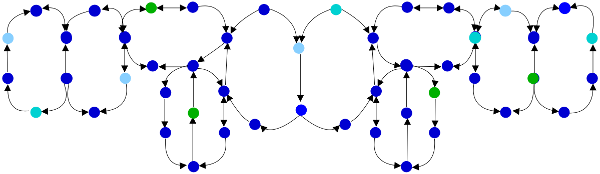
Collective Dynamics

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.



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

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

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

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

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

Novelty

The novelty in this project is twofold:

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Large scale software systems
Issues of scalability are important for user satisfaction and resource efficiency but such issues are difficult to investigate using discrete state models.

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

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

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

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

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

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

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 may be 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 may be used to generate a CTMC for performance modelling.

PEPA 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 may be used to generate a CTMC for performance modelling.

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

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 may be used to generate a CTMC for performance modelling.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM}$$

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 may be used to generate a CTMC for performance modelling.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}}$$

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 may be used to generate a CTMC for performance modelling.

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

Performance Evaluation Process Algebra

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

$(\alpha, f).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \bowtie_L P_2$	Co-operation
P/L	Hiding
X	Variable

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

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

Performance Evaluation Process Algebra

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

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

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

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

Performance Evaluation Process Algebra

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

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

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

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

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

A simple example: processors and resources

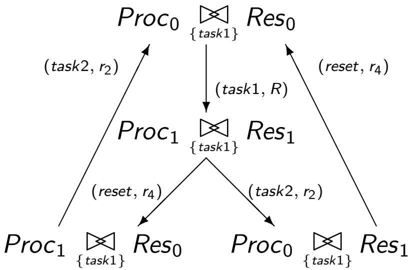
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

A simple example: processors and resources

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

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

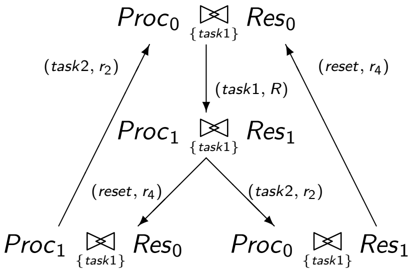


$$R = \min(r_1, r_3)$$

A simple example: processors and resources

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

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



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

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

Outline

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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

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.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

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

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.

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.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

○ ○ ○ ○ ○ ○ ○ ○ ○

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

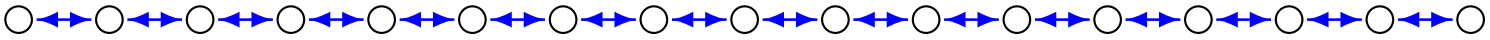
Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.



Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Use continuous state variables to approximate the discrete state space.

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

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.

.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.

.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.

.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.

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

Simple example revisited [QEST'05]

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

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

Simple example revisited [QEST'05]

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

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

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 [QEST'05]

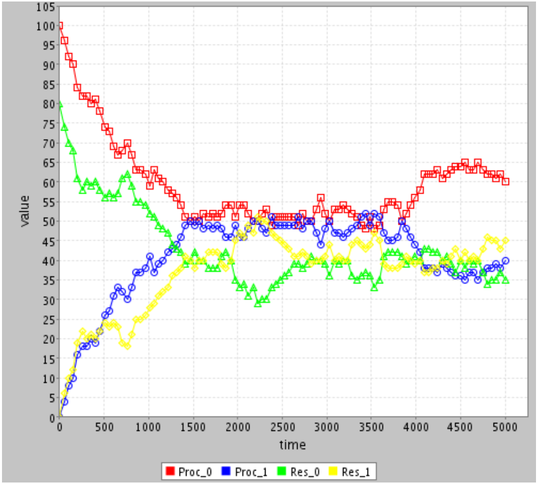
$$\begin{aligned} Proc_0 &\stackrel{\mathit{def}}{=} (task1, r_1).Proc_1 \\ Proc_1 &\stackrel{\mathit{def}}{=} (task2, r_2).Proc_0 \\ Res_0 &\stackrel{\mathit{def}}{=} (task1, r_1).Res_1 \\ Res_1 &\stackrel{\mathit{def}}{=} (reset, r_4).Res_0 \end{aligned}$$

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

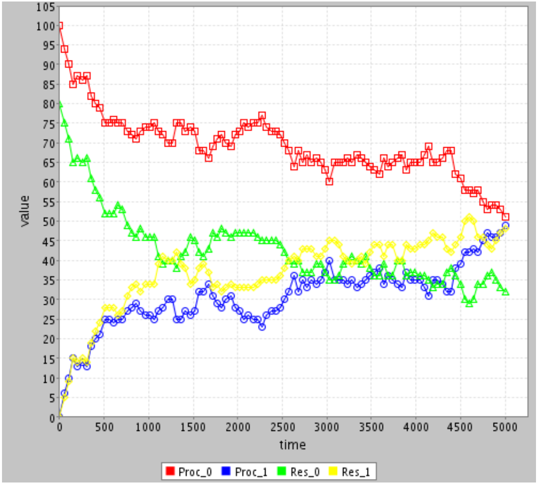
ODE interpretation

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

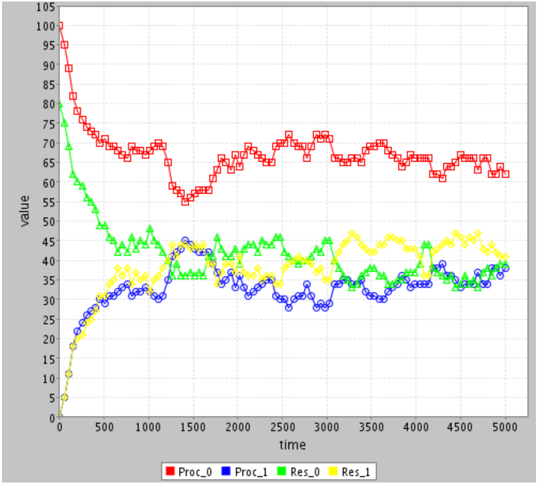
100 processors and 80 resources (simulation run A)



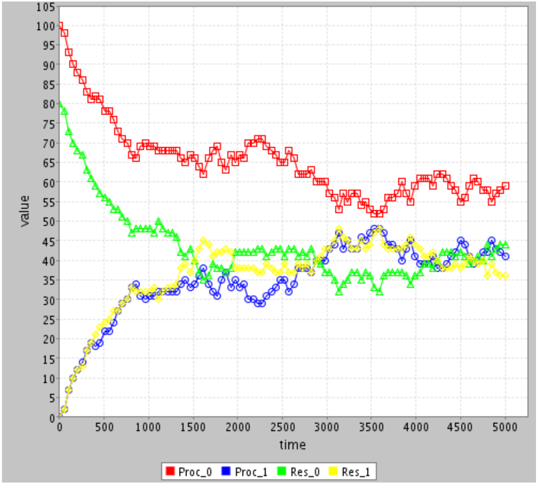
100 processors and 80 resources (simulation run B)



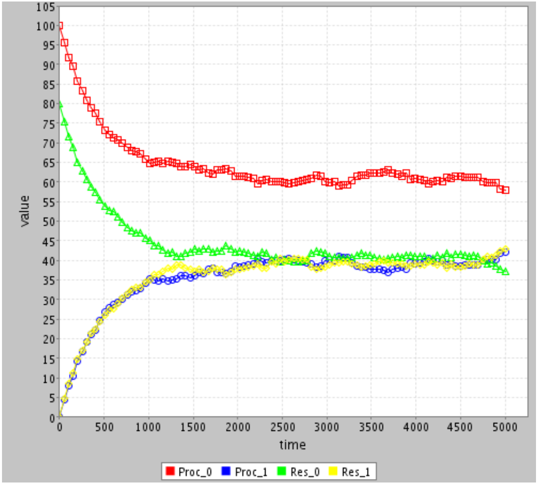
100 processors and 80 resources (simulation run C)



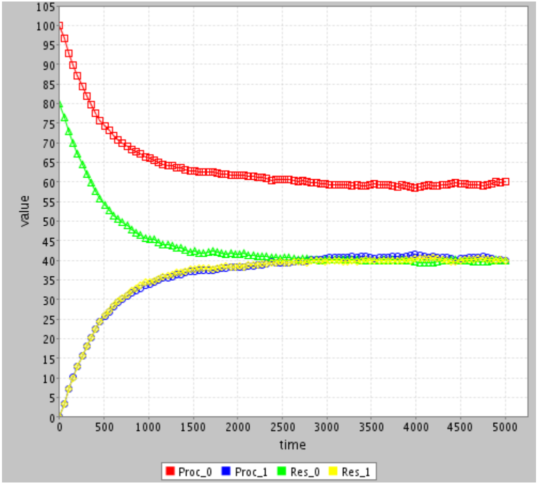
100 processors and 80 resources (simulation run D)



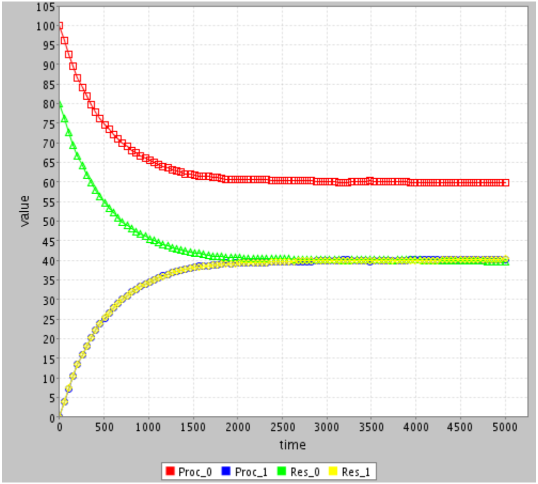
100 processors and 80 resources (average of 10 runs)



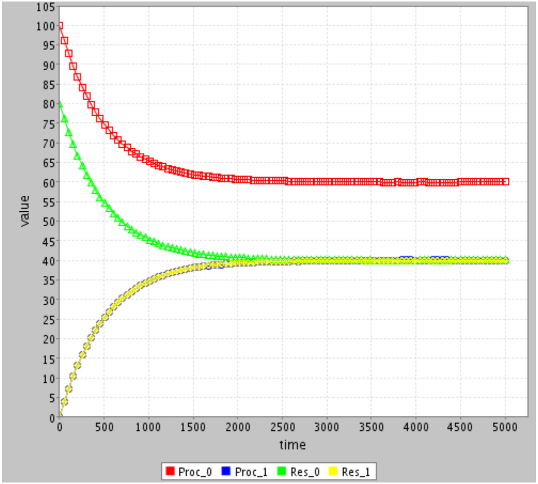
100 Processors and 80 resources (average of 100 runs)



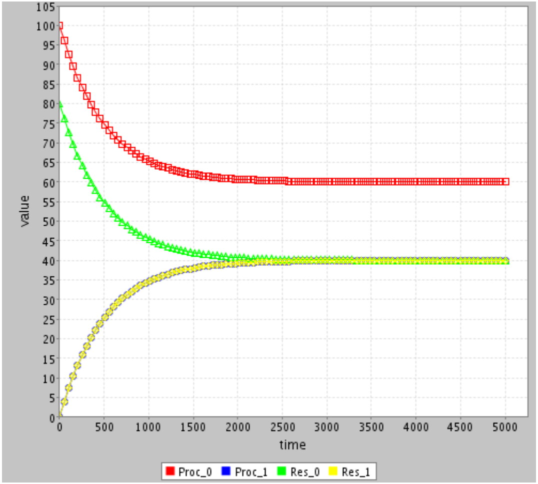
100 processors and 80 resources (average of 1000 runs)



100 processors and 80 resources (average of 10000 runs)



100 processors and 80 resources (ODE solution)



Outline

Deriving a Fluid Approximation of a PEPA model

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

Deriving a Fluid Approximation of a PEPA model

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

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

Deriving a Fluid Approximation of a PEPA model

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

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

Nevertheless we are able to define a structured operational semantics which defines the possible transitions of an abitrary abstract state and from this derive the ODEs.

Fluid Structured Operational Semantics

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

Fluid Structured Operational Semantics

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

1. Remove excess components (*Context Reduction*)

Fluid Structured Operational Semantics

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

1. Remove excess components (*Context Reduction*)
2. Collect the transitions of the reduced context (*Jump Multiset*)

Fluid Structured Operational Semantics

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

1. Remove excess components (*Context Reduction*)
2. Collect the transitions of the reduced context (*Jump Multiset*)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

Fluid Structured Operational Semantics

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

1. Remove excess components (*Context Reduction*)
2. Collect the transitions of the reduced context (*Jump Multiset*)
3. Calculate the rate of the transitions in terms of an arbitrary state of the CTMC.

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

Context Reduction

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

Location Dependency

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

$$\Downarrow$$

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

Location Dependency

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

$$\cfrac{\cfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{\color{red}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_{*} Proc_1}}}{}$$

Fluid Structured Operational Semantics by Example

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

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

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

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

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

Apparent Rate Calculation

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

$$\begin{aligned} r(\xi) &= \frac{r_1 \xi_1}{r^*_{task1}\left(Proc_0, \xi\right)} \; \frac{r_3 \xi_4}{r^*_{task1}\left(Res_0, \xi\right)} \min\left(r^*_{task1}\left(Proc_0, \xi\right), r^*_{task1}\left(Res_0, \xi\right)\right) \\ &= \frac{r_1 \xi_1}{r_1 \xi_1} \; \frac{r_3 \xi_4}{r_3 \xi_4} \min\left(r_1 \xi_1, r_3 \xi_4\right) \\ &= \min\left(r_1 \xi_1, r_3 \xi_4\right) \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$r = \frac{r_1}{2r_1}\frac{r_3}{3r_3}\min(2r_1, 3r_3) = \frac{1}{6}\min(2r_1, 3r_3)$$

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

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

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

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

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

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

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

$$r = \frac{r_1}{2r_1}\frac{r_3}{3r_3}\min(2r_1, 3r_3) = \frac{1}{6}\min(2r_1, 3r_3)$$

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

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

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

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

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

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

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

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

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

$$r = \frac{r_1}{2r_1}\frac{r_3}{3r_3}\min(2r_1, 3r_3) = \frac{1}{6}\min(2r_1, 3r_3)$$

Jump Multiset

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

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

Jump Multiset

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

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

Extraction of the ODE from f

Generator Function

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

Differential Equation

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

Outline

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.
- Worms like Nimbda, Slammer, Code Red, Sasser and Code Red 2 have caused the Internet to become unusable for many hours at a time until security patches could be applied and routers fixed.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.
- Worms like Nimbda, Slammer, Code Red, Sasser and Code Red 2 have caused the Internet to become unusable for many hours at a time until security patches could be applied and routers fixed.
- The estimated cost of computer worms and related activities is about $50 billion a year.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.
- Explicit state-based methods for calculating steady-state, transient or passage-time measures are limited to state-spaces of the order of 10^9.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.
- Explicit state-based methods for calculating steady-state, transient or passage-time measures are limited to state-spaces of the order of 10^9.
- By transforming our stochastic process algebra model into a set of ODEs, we can obtain a plot of model behaviour against time for models with global state spaces in excess of 10^{10000} states.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$\frac{\mathrm{d}s(t)}{\mathrm{d}t} = -\beta\, s(t)\, i(t)$$

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$
\begin{aligned}
\frac{\mathrm{d}s(t)}{\mathrm{d}t} &= -\beta\, s(t)\, i(t) \\
\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= \beta\, s(t)\, i(t) - \gamma\, i(t)
\end{aligned}
$$

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$
\begin{aligned}
\frac{\mathrm{d}s(t)}{\mathrm{d}t} &= -\beta\, s(t)\, i(t) \\
\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= \beta\, s(t)\, i(t) - \gamma\, i(t) \\
\frac{\mathrm{d}r(t)}{\mathrm{d}t} &= \gamma\, i(t)
\end{aligned}
$$

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.
- An infected computer can be patched so that it is no longer infected or susceptible to infection.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.
- An infected computer can be patched so that it is no longer infected or susceptible to infection.
- This state is termed removed and is an absorbing state for that component in the system.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.
- This might be due to network contention or the lack of availability of a susceptible machine to infect.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.
- This might be due to network contention or the lack of availability of a susceptible machine to infect.
- As large scale worm infections tend not to waste time determining whether a given host is already infected or not, we assume that a certain number of infections will attempt to reinfect hosts; in this instance, the host is unaffected.

Susceptible-Infective-Removed over a network

$$
\begin{aligned}
S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\mathit{def}}{=} \mathit{Stop}
\end{aligned}
$$

Susceptible-Infective-Removed over a network

$$
\begin{aligned}
S &\stackrel{def}{=} (infectS, \top).I \\
I &\stackrel{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\stackrel{def}{=} Stop
\end{aligned}
$$

$$
\begin{aligned}
Net &\stackrel{def}{=} (infectI, \top).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

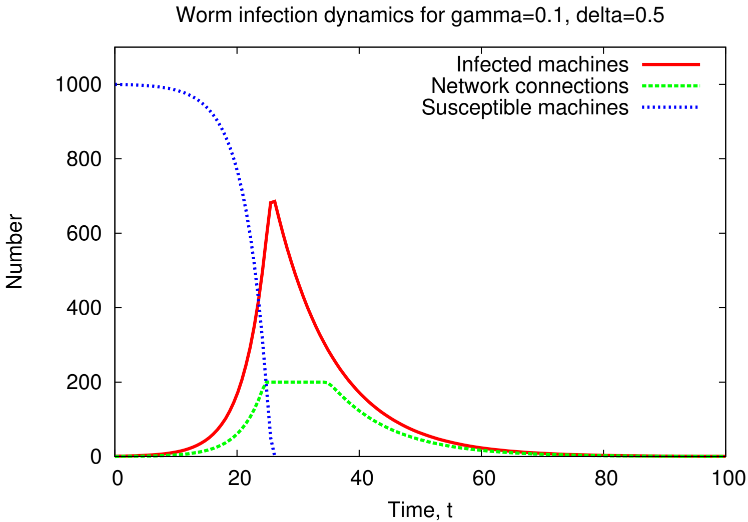
Susceptible-Infective-Removed over a network

$$
\begin{aligned}
S &\stackrel{\text{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\text{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\text{def}}{=} \mathit{Stop}
\end{aligned}
$$

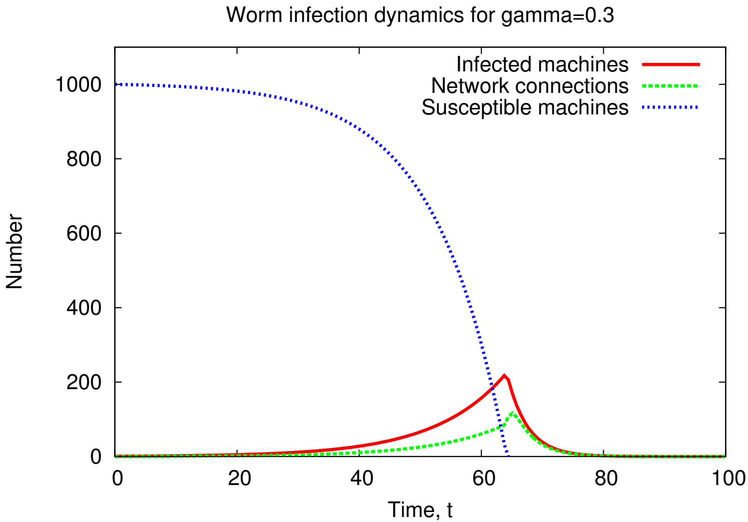
$$
\begin{aligned}
\mathit{Net} &\stackrel{\text{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\stackrel{\text{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net}
\end{aligned}
$$

$$
\mathit{Sys} \stackrel{\text{def}}{=} (S[N] \parallel I) \underset{L}{\bowtie} \mathit{Net}[M] \quad \text{where } L = \{\, \mathit{infectI}, \mathit{infectS} \,\}
$$

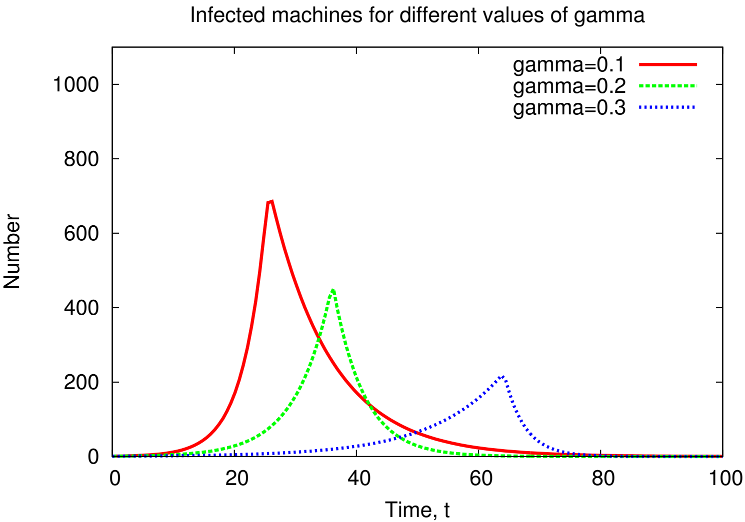
Patch rate $\gamma = 0.1$. Connection failure rate $\delta = 0.5$



Patch rate $\gamma = 0.3$. Connection failure rate $\delta = 0.5$



Increasing machine patch rate γ from 0.1 to 0.3



Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.

Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.
- A small modification in the process model of infection allows for removed computers to become susceptible again after a delay.

Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.
- A small modification in the process model of infection allows for removed computers to become susceptible again after a delay.
- We use this to model a faulty or incomplete security upgrade or the mistaken removal of security patches which had previously defended the machine against attack.

Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\mathit{def}}{=} (\mathit{unsecure}, \mu).S
\end{aligned}
$$

Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\mathit{def}}{=} (\mathit{unsecure}, \mu).S
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Net} &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net}
\end{aligned}
$$

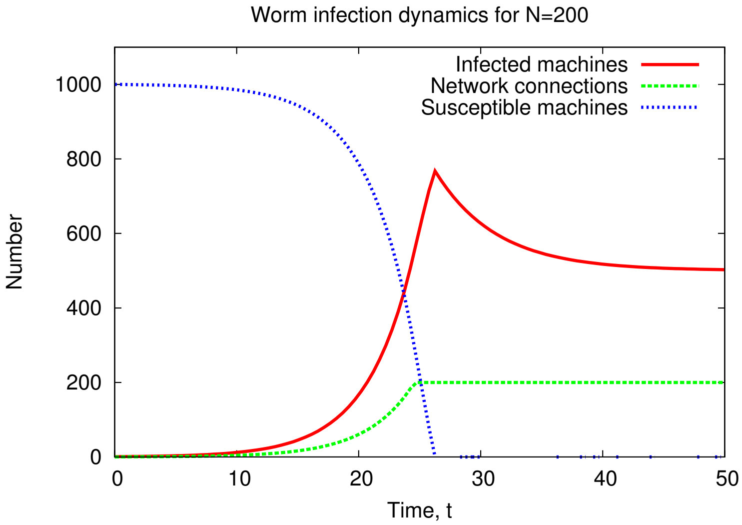
Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\stackrel{def}{=} (infectS, \top).I \\
I &\stackrel{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\stackrel{def}{=} (unsecure, \mu).S
\end{aligned}
$$

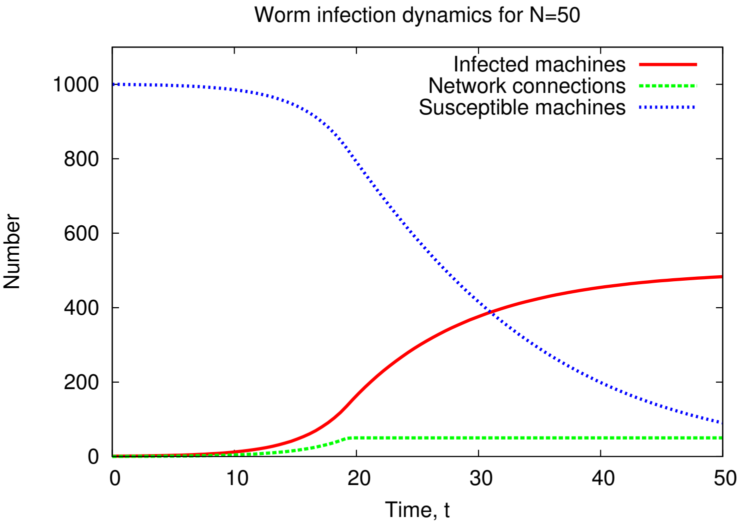
$$
\begin{aligned}
Net &\stackrel{def}{=} (infectI, \top).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

$$
\begin{aligned}
Sys &\stackrel{def}{=} (S[1000] \parallel I) \underset{L}{\bowtie} Net[M] \\
&\qquad where\ L = \{infectI, infectS\}
\end{aligned}
$$

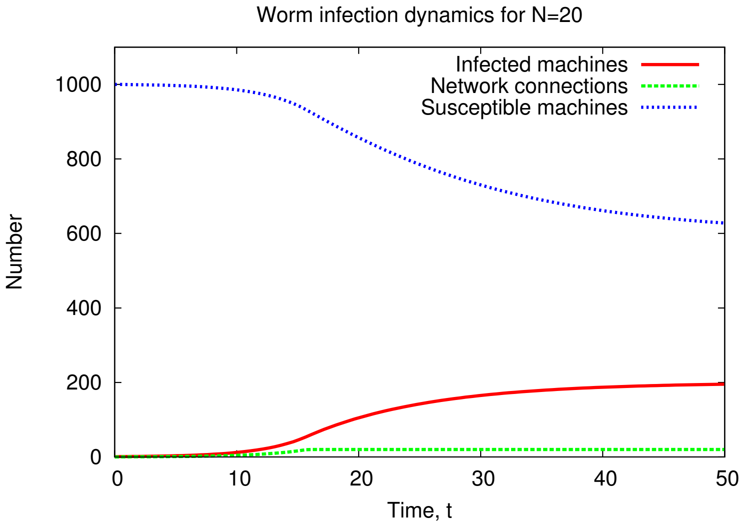
Unsecured SIR model (200 network channels)



Unsecured SIR model (50 network channels)



Unsecured SIR model (20 network channels)



Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.

Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then abstracted away into appropriate population-based representations.

Conclusions

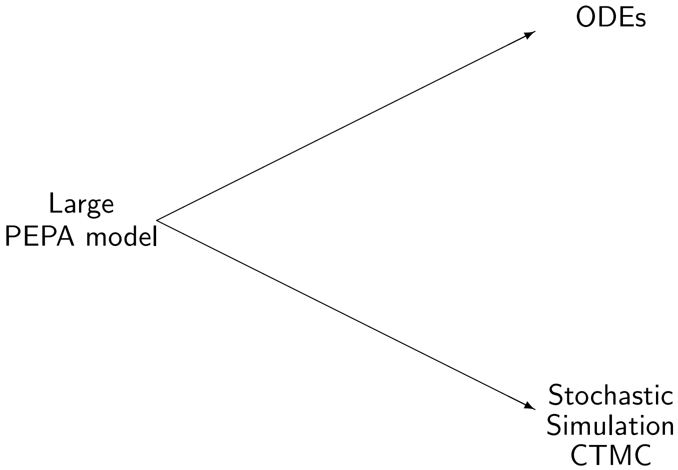
- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then abstracted away into appropriate population-based representations.
- The scale of problems which can be modelled in this way vastly exceeds those which are founded on explicit state representations.

Conclusions

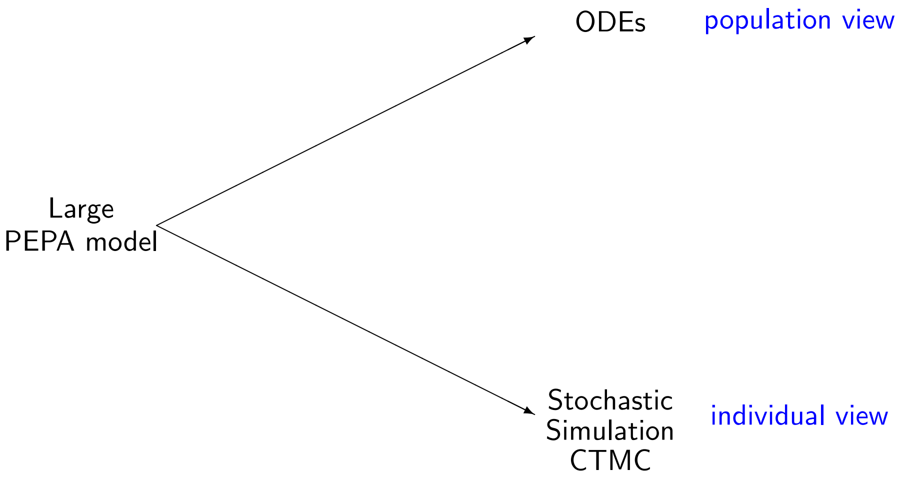
- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then abstracted away into appropriate population-based representations.
- The scale of problems which can be modelled in this way vastly exceeds those which are founded on explicit state representations.
- We believe the modelling methods exemplified here to be generally useful for analysing the behaviour of populations of interacting processes with complex dynamics.

Outline

Alternative Representations



Alternative Representations



Consistency results

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$
- 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
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$
- 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).
- Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- This alternative view of systems has opened up many and exciting new research directions.

Thanks!

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

Thanks!

Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

Acknowledgements: funding
Thanks to EPRSC for the Process Algebra for Collective Dynamics grant and the CEC IST-FET programme for the SENSORIA project which have supported this work.

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Nil Gesweiller, **Stephen Gilmore**, Marco Stenico, **Mirco Tribastone**, and others.

Acknowledgements: funding

Thanks to EPRSC for the Process Algebra for Collective Dynamics grant and the CEC IST-FET programme for the SENSORIA project which have supported this work.

More information:

http://www.dcs.ed.ac.uk/pepa