SPAs for performance modelling: Lecture 6 — Collective Dynamics

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

15th April 2013

THE UNIVERSITY
of EDINBURGH

Outline

Outline

Introduction

- In the previous lecture we introduced continuous or fluid approximation as a means to tackle state space explosion.

Introduction

- In the previous lecture we introduced continuous or fluid approximation as a means to tackle state space explosion.
- The approach taken was a pragmatic one based on the activity matrix and no formal results were given about the correctness of such an approach.

Introduction

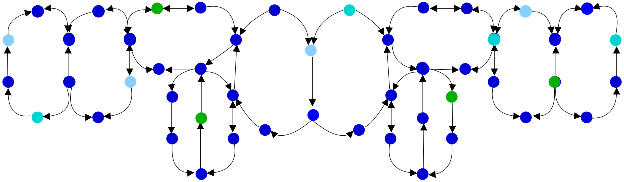
- In the previous lecture we introduced continuous or fluid approximation as a means to tackle state space explosion.
- The approach taken was a pragmatic one based on the activity matrix and no formal results were given about the correctness of such an approach.
- Here we reconsider the approach with a particular focus on the types of systems that it is well-suited for: systems with collective dynamics.

Introduction

- In the previous lecture we introduced continuous or fluid approximation as a means to tackle state space explosion.
- The approach taken was a pragmatic one based on the activity matrix and no formal results were given about the correctness of such an approach.
- Here we reconsider the approach with a particular focus on the types of systems that it is well-suited for: systems with collective dynamics.
- Moreover we give a more formal derivation of the system of ordinary differential equations that are used to approximate the discrete event system we are interested in.

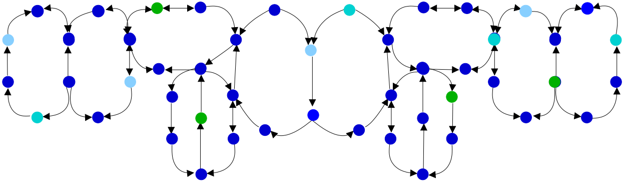
Collective Dynamics

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.

Collective Behaviour

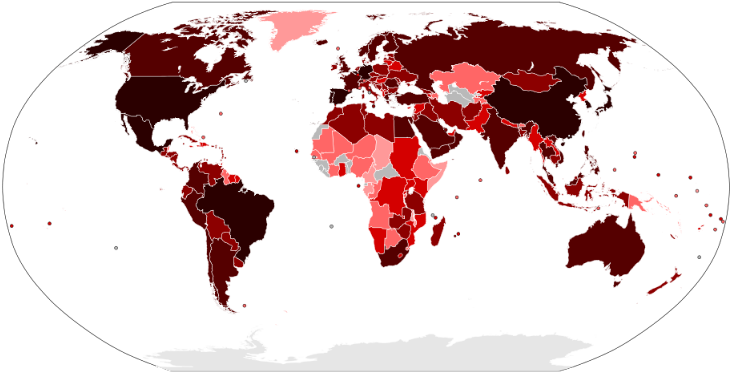
In the natural world there are many instances of collective behaviour and its consequences:

Collective Behaviour

In the natural world there are many instances of collective behaviour and its consequences:

Collective Behaviour

This is also true in the man-made and engineered world:



Spread of H1N1 virus in 2009

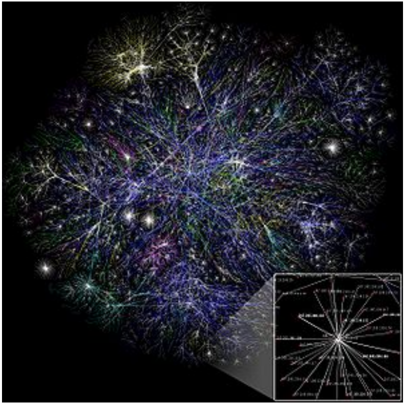
Collective Behaviour

This is also true in the man-made and engineered world:

Love Parade, Germany 2006

Collective Behaviour

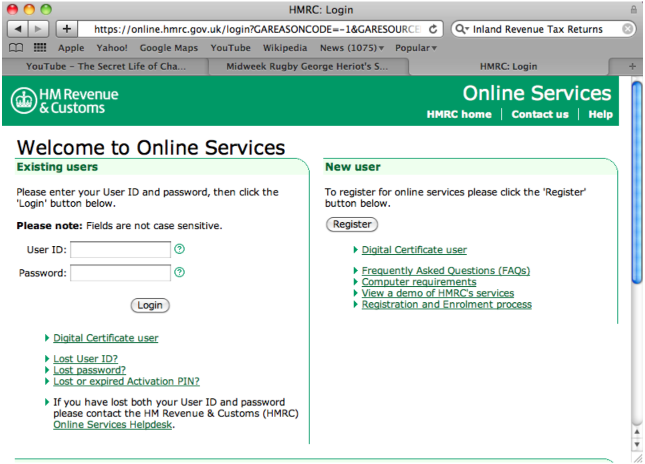
This is also true in the man-made and engineered world:



Map of the Internet 2009

Collective Behaviour

This is also true in the man-made and engineered world:



Self assessment tax returns 31st January each year

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.

Performance as an emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only the consider the system as a whole as an interaction of populations.

Performance as an emergent behaviour

A shift in perspective allows us to model the interactions between individual components but then only the 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.

Performance as an emergent behaviour

We must instead think about the performance of the collective point of view. Service providers often want to do this in any case.

Performance as an emergent behaviour

We must instead think about the performance of the collective point of view. Service providers often want to do this in any case.

For example making contracts in terms of service level agreements.

Example Service Level Agreement

90% of requests receive a response within 3 seconds.

Performance as an emergent behaviour

We must instead think about the performance of the collective point of view. Service providers often want to do this in any case.

For example making contracts in terms of service level agreements.

Example Service Level Agreement

90% of requests receive a response within 3 seconds.

Qualitative Service Level Agreement

Less than 1% of the responses received within 3 seconds will read "System is overloaded, try again later".

Novelty

The novelty of the CODA project has been twofold:

Novelty

The novelty of the CODA project has been twofold:

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

Novelty

The novelty of the CODA project has been 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 of the CODA project has been 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 of the CODA project has been 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 of the CODA project has been 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.

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.

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

Continuous Approximation

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

One approach to this problem is to use continuous state variables to approximate the discrete state space based on counting variables.

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.

Kurtz's Theorem

We seek to take advantage of Kurtz's Theorem from the 1970's which gives conditions under which a sequence of population Markov chains converges to a deterministic behaviour (within a given time horizon), i.e. $\forall t < T$ as $N \longrightarrow \infty$.

Kurtz's Theorem

We seek to take advantage of Kurtz's Theorem from the 1970's which gives conditions under which a sequence of population Markov chains converges to a deterministic behaviour (within a given time horizon), i.e. $\forall t < T$ as $N \longrightarrow \infty$.

Kurtz's result is given in terms of the size of the system, where size is often taken to be the total population size.

Kurtz's Theorem

We seek to take advantage of Kurtz's Theorem from the 1970's which gives conditions under which a sequence of population Markov chains converges to a deterministic behaviour (within a given time horizon), i.e. $\forall t < T$ as $N \longrightarrow \infty$.

Kurtz's result is given in terms of the size of the system, where size is often taken to be the total population size.

In terms of a PEPA model we assume that there is an initial population for each component type, and that all the subpopulations are scaled at the same rate.

Kurtz's Theorem

We seek to take advantage of Kurtz's Theorem from the 1970's which gives conditions under which a sequence of population Markov chains converges to a deterministic behaviour (within a given time horizon), i.e. $\forall t < T$ as $N \longrightarrow \infty$.

Kurtz's result is given in terms of the size of the system, where size is often taken to be the total population size.

In terms of a PEPA model we assume that there is an initial population for each component type, and that all the subpopulations are scaled at the same rate.

For example, a model $P[X_P] \underset{L}{\bowtie} Q[X_Q]$, is scaled as $P[n \times X_P] \underset{L}{\bowtie} Q[n \times X_Q]$ for increasing n.

Kurtz's Theorem

The models in this sequence of CTMCs have ever increasing state space as n grows.

Kurtz's Theorem

The models in this sequence of CTMCs have ever increasing state space as n grows.

Soit is difficult to compare the results obtained from the models because the population size is growing so in absolute terms the performance metrics will grow.

Kurtz's Theorem

The models in this sequence of CTMCs have ever increasing state space as n grows.

Soit is difficult to compare the results obtained from the models because the population size is growing so in absolute terms the performance metrics will grow.

Therefore in order to make the models within the sequence comparable we normalise the models, so that the counting variables now represent a proportion rather than an absolute count.

Kurtz's Theorem

The models in this sequence of CTMCs have ever increasing state space as n grows.

Soit is difficult to compare the results obtained from the models because the population size is growing so in absolute terms the performance metrics will grow.

Therefore in order to make the models within the sequence comparable we normalise the models, so that the counting variables now represent a proportion rather than an absolute count.

In the literature this is sometimes called the occupancy measure.

Scaling Conditions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs, for increasing total population N.

Scaling Conditions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs, for increasing total population N.
- We normalize such models, dividing variables by N: $\overline{\mathbf{X}}^{(N)} = \frac{\mathbf{X}}{N}$

Scaling Conditions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs, for increasing total population N.
- We normalize such models, dividing variables by N: $\overline{\mathbf{X}}^{(N)} = \frac{\mathbf{X}}{N}$
- We assume that each transition in the Markov chain is characterised by an update vector $\mathbf{v}$ (cf. the columns of the activity matrix)

Scaling Conditions

- We have a sequence $\mathbf{X}^{(N)}$ of population CTMCs, for increasing total population N.
- We normalize such models, dividing variables by N: $\overline{\mathbf{X}}^{(N)} = \frac{\mathbf{X}}{N}$
- We assume that each transition in the Markov chain is characterised by an update vector $\mathbf{v}$ (cf. the columns of the activity matrix)
- For each such transition τ, the normalized update is $\bar{\mathbf{v}} = \mathbf{v}/N$ and the rate function is $\bar{r}_\tau(\overline{\mathbf{X}}^{(N)}) = Nf_\tau(\overline{\mathbf{X}}^{(N)})$ (density dependence).

Fluid ODE

For a sequence of population CTMCs that satisfy these conditions we can define the Fluid ODE:

Fluid ODE

The fluid ODE is $\dot{\mathbf{x}} = F(\mathbf{x})$, where

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

Fluid approximation theorem

Hypothesis

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

Theorem

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

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

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

Simple example revisited

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

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

Simple example revisited

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

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

CTMC interpretation

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

Simple example revisited

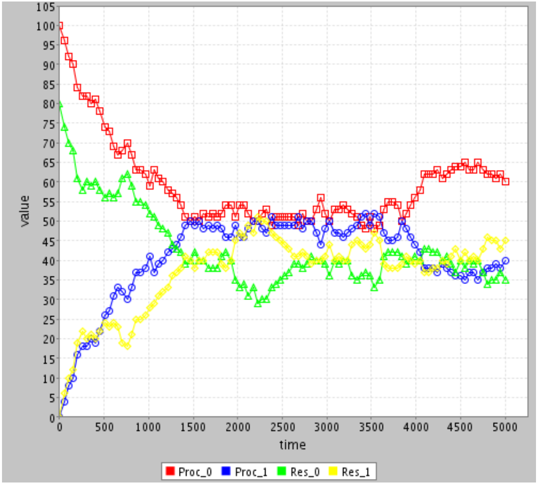
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_1).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

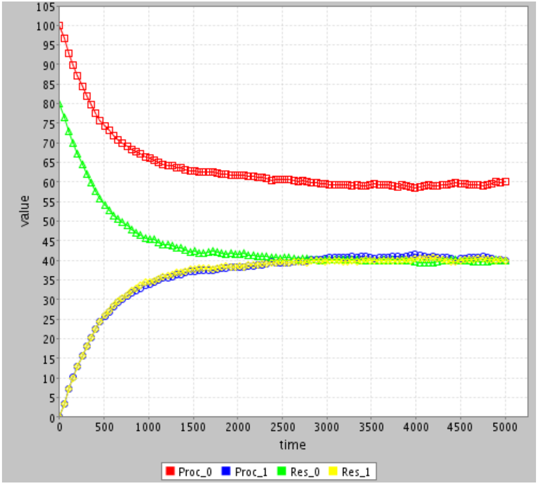
ODE interpretation

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

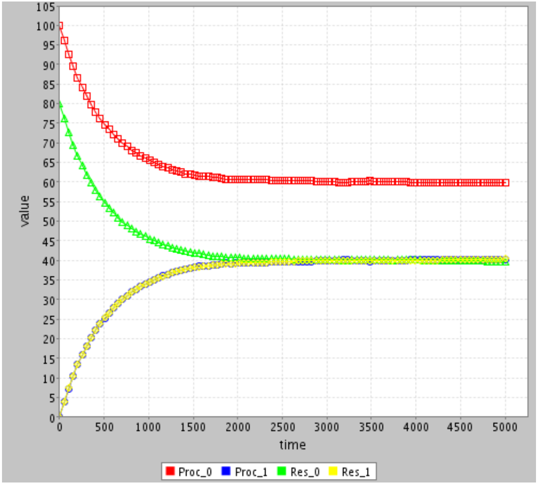
100 processors and 80 resources (simulation run A)



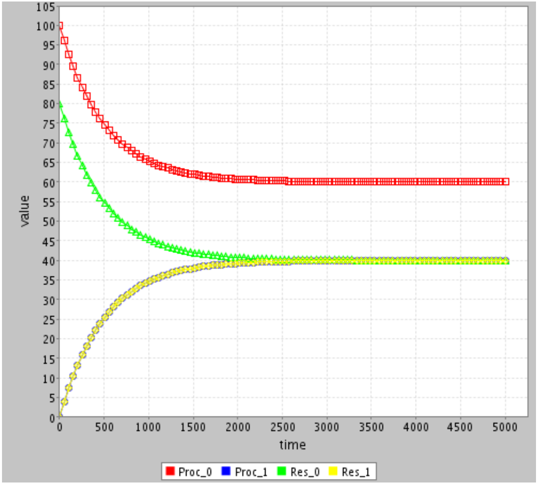
100 Processors and 80 resources (average of 100 runs)



100 processors and 80 resources (average of 1000 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 existing (CTMC) SOS semantics is not suitable for this purpose because it constructs the state space of the CTMC explicitly.

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

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

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

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

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

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

Generating functions

- The population-based semantics gives rise to generating functions, denoted by $f_\alpha(\xi, l)$, giving the rate at which an activity of type α is executed, and the state change due to its execution as a vector l.

Generating functions

- The population-based semantics gives rise to generating functions, denoted by $f_\alpha(\xi, l)$, giving the rate at which an activity of type α is executed, and the state change due to its execution as a vector l.

- Thus in the previous example the shared action `task1` is captured by the function

$$f_{task1}\big(\xi, (-1, 1, -1, 1)\big) = \min(r\,\xi_{Proc_0}, r\,\xi_{Res_0}),$$

`task1` decreases the population counts of $Proc_0$ and Res_0 and, correspondingly, increases the population counts of $Proc_1$ and Res_1 at a rate which is dependent upon the current state.

Generating functions

- The population-based semantics gives rise to generating functions, denoted by $f_\alpha(\xi, l)$, giving the rate at which an activity of type α is executed, and the state change due to its execution as a vector l.

- Thus in the previous example the shared action `task1` is captured by the function

$$f_{task1}\big(\xi, (-1, 1, -1, 1)\big) = \min(r\,\xi_{Proc_0}, r\,\xi_{Res_0}),$$

  `task1` decreases the population counts of $Proc_0$ and Res_0 and, correspondingly, increases the population counts of $Proc_1$ and Res_1 at a rate which is dependent upon the current state.

- This is just as we saw with the activity matrix construction but now obtained through SOS rules.

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.

Semantics by example

In these slides I will illustrate this approach to the scalable semantics using the previous Processor-Resource example.

But the full SOS rules can be found in the paper:

Tribastone M, Gilmore S, Hillston J,
Scalable Differential Analysis of Process Algebra Models
Transactions on Software Engineering 38(1), 2012

Context Reduction

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

$$\Downarrow$$

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

Context Reduction

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

$$\Downarrow$$

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

Population Vector

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

Location Dependency

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

Location Dependency

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

$$\Downarrow$$

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

Location Dependency

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

$$\Downarrow$$

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

Population Vector

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

Fluid Structured Operational Semantics by Example

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Fluid Structured Operational Semantics by Example

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

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

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

Apparent Rate Calculation

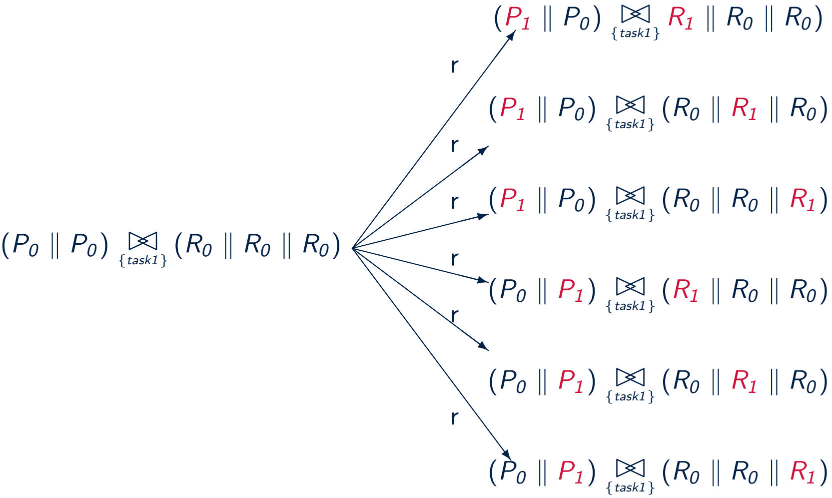
$$\dfrac{\dfrac{Proc_0 \xrightarrow{task1, r_1} Proc_1}{Proc_0 \xrightarrow{task1, r_1 \xi_1}_* Proc_1} \qquad \dfrac{Res_0 \xrightarrow{task1, r_3} Res_1}{Res_0 \xrightarrow{task1, r_3 \xi_3}_* Res_1}}{Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \xrightarrow{task1, r(\xi)}_* Proc_1 \underset{\{task1\}}{\bowtie}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

Apparent Rate Calculation

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

$$\begin{aligned} r(\xi) &= \frac{r_1 \xi_1}{r^*_{task1}\left(Proc_0, \xi\right)} \; \frac{r_3 \xi_4}{r^*_{task1}\left(Res_0, \xi\right)} \min\left(r^*_{task1}\left(Proc_0, \xi\right), r^*_{task1}\left(Res_0, \xi\right)\right) \\ &= \frac{r_1 \xi_1}{r_1 \xi_1} \; \frac{r_3 \xi_3}{r_3 \xi_3} \min\left(r_1 \xi_1, r_3 \xi_3\right) \\ &= \min\left(r_1 \xi_1, r_3 \xi_3\right) \end{aligned}$$

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Jump Multiset

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$f(\xi, (+1, -1, 0, 0), task2) = \xi_2 r_2$$

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

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

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

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

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

$$f(\xi, (+1, -1, 0, 0), task2) = \xi_2 r_2$$

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

Capturing behaviour in the Generator Function

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Capturing behaviour in the Generator Function

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

Numerical Vector Form

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

Generator Function

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

Extraction of the ODE from f

Generator Function

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

Differential Equations

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

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 Equations

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_1x_1, r_3x_3) + r_2x_2 \\
\frac{dx_2}{dt} &= \min(r_1x_1, r_3x_3) - r_2x_2 \\
\frac{dx_3}{dt} &= -\min(r_1x_1, r_3x_3) + r_4x_4 \\
\frac{dx_4}{dt} &= \min(r_1x_1, r_3x_3) - r_4x_4
\end{aligned}
$$

Density Dependence

Density dependence of parametric apparent rates

Let $r^{*}_{\alpha}(P, \xi)$ be the parametric apparent rate of action type α in process P. For any $n \in \mathbb{N}$ and $\alpha \in \mathcal{A}$,

$$r^{*}_{\alpha}(P, \xi) = n \cdot r^{\star}_{\alpha}(P, \xi / n)$$

Density Dependence

Density dependence of parametric apparent rates

Let $r^{*}_{\alpha}(P, \xi)$ be the parametric apparent rate of action type α in process P. For any $n \in \mathbb{N}$ and $\alpha \in \mathcal{A}$,

$$r^{*}_{\alpha}(P, \xi) = n \cdot r^{\star}_{\alpha}(P, \xi/n)$$

Density dependence of parametric transition rates

If $P \xrightarrow{(\alpha, r(\xi))}_{*} Q$ then, for any $n \in \mathbb{N}$, $r(\xi) = n \cdot r(\xi/n)$

Generating functions give rise to density dependent rates

Generating functions give rise to density dependent rates

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

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity of parametric apparent rates

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

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

Lipschitz continuity

Since Lipschitz continuity is preserved by summation, in order to verify that the vector field $F_{\mathcal{M}}(x)$ is Lipschitz it suffices to prove that any parametric rate generated by the semantics is Lipschitz.

Lipschitz continuity of parametric apparent rates

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

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

Lipschitz continuity of rate functions

If $P \xrightarrow{(\alpha, r(x))}_{*} P'$ then $r(x) \leq r_{\alpha}^{*}(P, x)$ and thus it follows that $r(x)$ is Lipschitz continuous.

Kurtz's Theorem

Kurtz's Theorem for PEPA

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

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

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

Kurtz's Theorem

Kurtz's Theorem for PEPA

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

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

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

Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem. This allows the time horizon to be extended to ∞.

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

A note about passive actions

The scalable semantics which have been presented do not have rules for passive actions.

A note about passive actions

The scalable semantics which have been presented do not have rules for passive actions.

The reason is that the passive partner within a model will act as a switch.

- When it is present in any number the rate of the action will proceed at the rate determined by the activity rate and the population of the other partner.
- When it is absent the rate will become zero.

A note about passive actions

The scalable semantics which have been presented do not have rules for passive actions.

The reason is that the passive partner within a model will act as a switch.

- When it is present in any number the rate of the action will proceed at the rate determined by the activity rate and the population of the other partner.
- When it is absent the rate will become zero.

This will cause a discontinuity in the rate of the activity meaning that the Lipschitz continuity condition required to apply Kurtz's theorem will no longer hold.

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 at least $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

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

Susceptible-Infective-Removed over a network

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

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

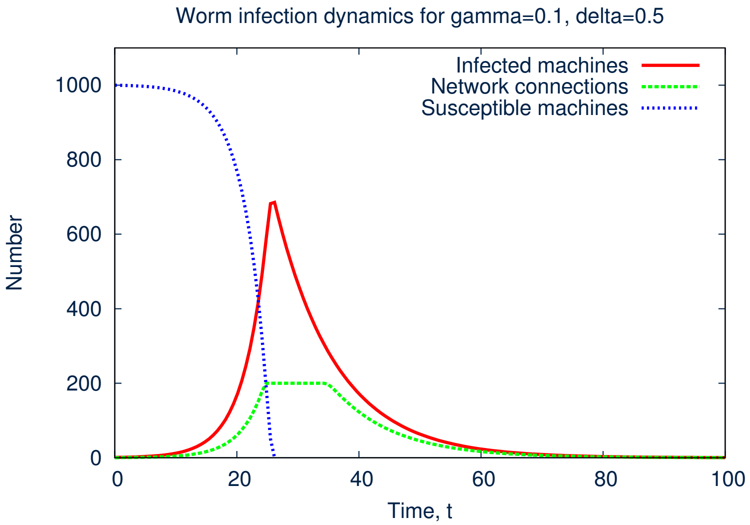
Susceptible-Infective-Removed over a network

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

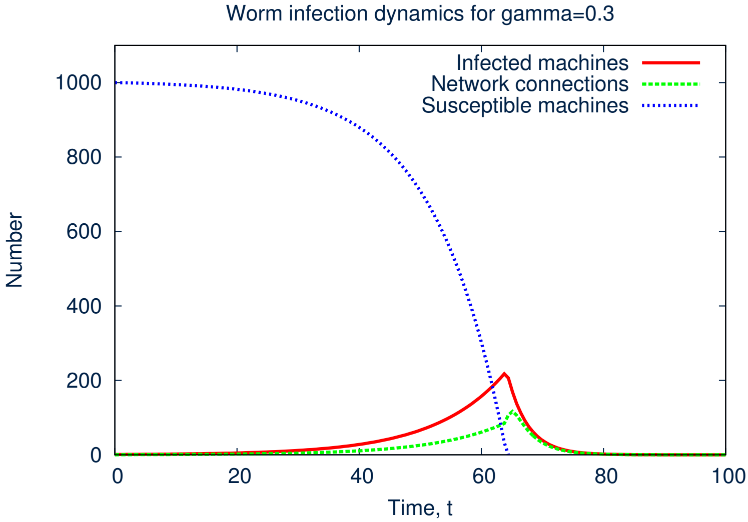
$$
\begin{aligned}
Net &\stackrel{def}{=} (infectI, \beta).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

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

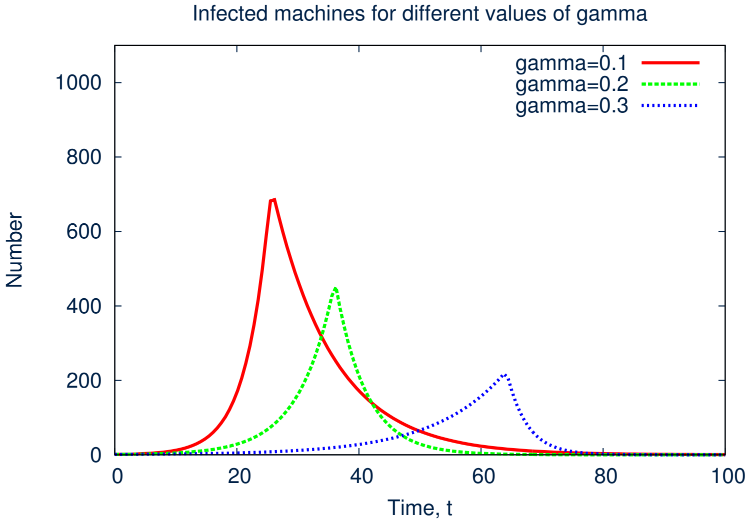
Patch rate $\gamma = 0.1$. Connection failure rate $\delta = 0.5$



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

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

Susceptible-Infective-Removed-Reinfection (SIRR) model

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

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

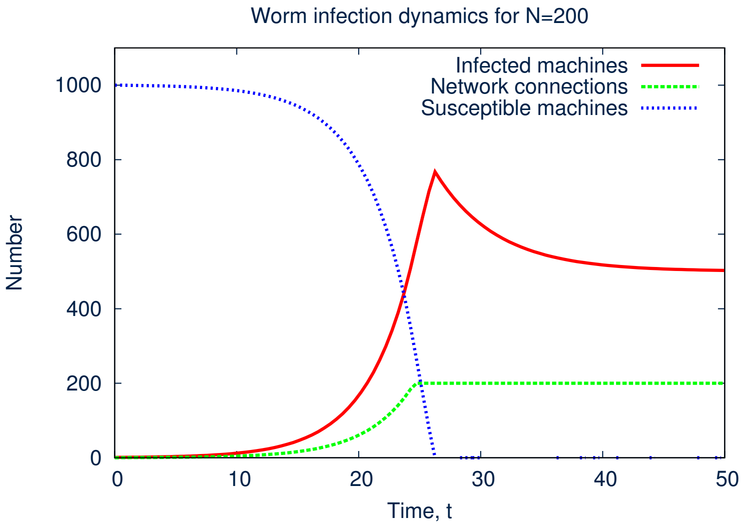
Susceptible-Infective-Removed-Reinfection (SIRR) model

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

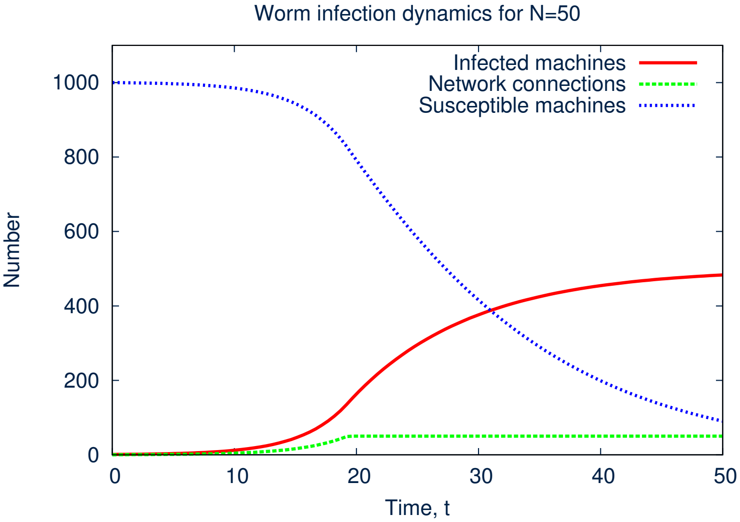
$$
\begin{aligned}
Net &\stackrel{def}{=} (infectI, \beta).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

$$
Sys \stackrel{def}{=} (S[1000] \parallel I) \underset{L}{\bowtie} Net[M] \quad \text{where } L = \{infectI, infectS\}
$$

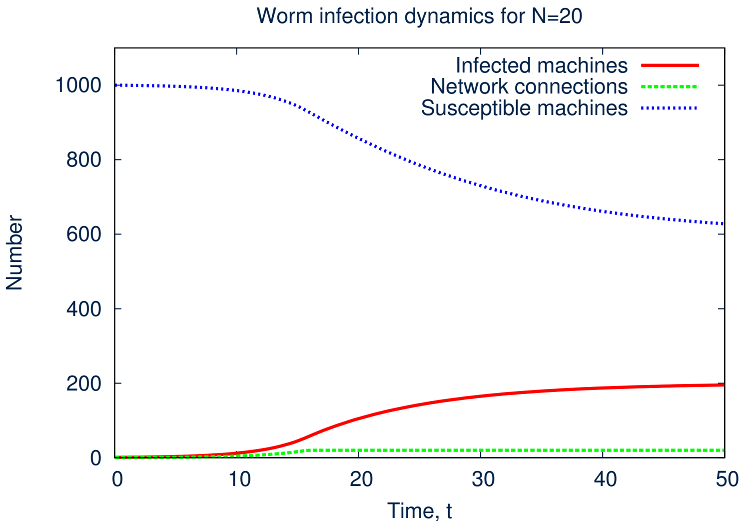
Unsecured SIR model (200 network channels)



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.

Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail.
- 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.
- 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.