Performance Modelling with Stochastic Process Algebra

Jane Hillston
LFCS, Edinburgh

Dynamic Properties of Complex Networks,
31st March 2009

The PEPA project

- The PEPA project started in Edinburgh in 1991.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.

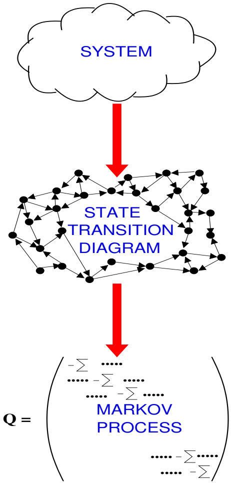
The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC)

Outline

Performance Modelling using CTMC

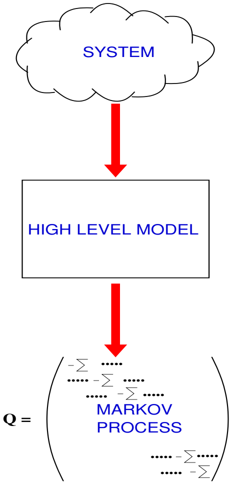
Model Construction



Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC



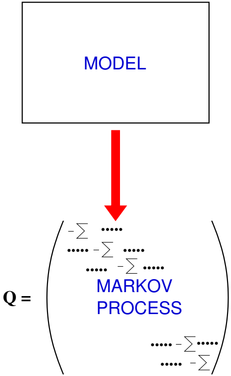
Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation



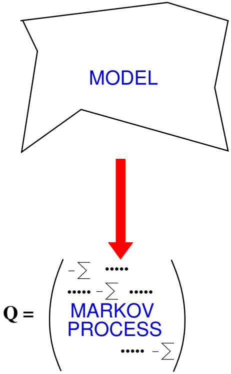
Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation



Performance Modelling using CTMC

Model Construction

- describing the system using a high level modelling formalism
- generating the underlying CTMC

Model Manipulation

- model simplification
- model aggregation

Model Solution

- solve to find steady state or transient probability distribution
- deriving performance measures

Process Algebra

- Models consist of agents which engage in actions.

$$\underbrace{\alpha}_{\text{action type or name}}.\underbrace{P}_{\text{agent/ component}}$$

Process Algebra

- Models consist of agents which engage in actions.

$$\underbrace{\alpha}_{\text{action type or name}}.\underbrace{P}_{\text{agent/ component}}$$

Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type
or name

agent/
component

Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name — agent/ component

- The language is used to generate a labelled transition system for functional verification: reachability analysis, specification matching and model checking.

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

Process Algebra

- Models consist of agents which engage in actions.

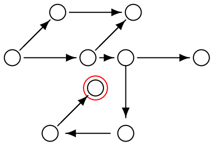
$$\alpha.P$$

action type or name

agent/ component

- The language is used to generate a labelled transition system for functional verification: reachability analysis, specification matching and model checking.

Will the system arrive in a particular state?



Process Algebra

- Models consist of agents which engage in actions.

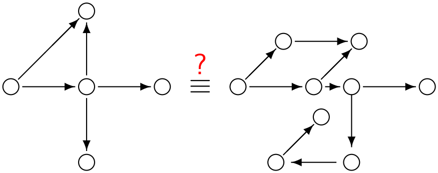
$$\alpha.P$$

action type or name

agent/ component

- The language is used to generate a labelled transition system for functional verification: reachability analysis, specification matching and model checking.

Does system behaviour match its specification?



Process Algebra

- Models consist of agents which engage in actions.

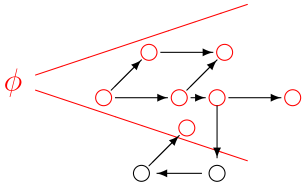
$$\alpha.P$$

action type or name

agent/ component

- The language is used to generate a labelled transition system for functional verification: reachability analysis, specification matching and model checking.

Does a given property ϕ hold within the system?



Performance Evaluation 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

Performance Evaluation 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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, \color{red}{r}).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation 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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

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

PEPA
MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

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

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

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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.

$(\alpha, r).P$	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

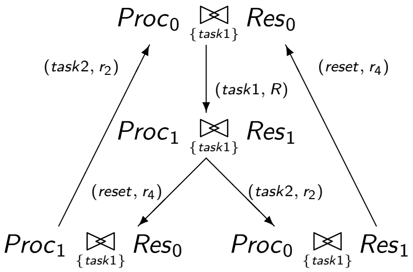
$$
\begin{aligned}
Proc_0 &\stackrel{def}{=} (task1, r_1).Proc_1 \\
Proc_1 &\stackrel{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\stackrel{def}{=} (task1, r_3).Res_1 \\
Res_1 &\stackrel{def}{=} (reset, r_4).Res_0
\end{aligned}
$$

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

A simple example: processors and resources

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

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

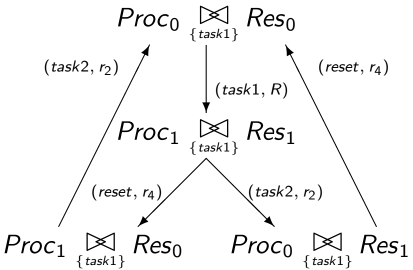


$$R = \min(r_1, r_3)$$

A simple example: processors and resources

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

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



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

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

Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.

Benefits of Quantification

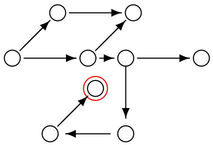
- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Reachability analysis

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

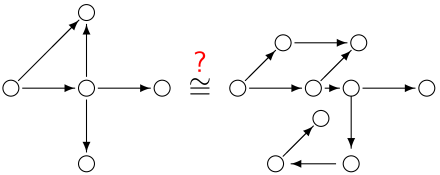


Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Specification matching

With what probability does system behaviour match its specification?

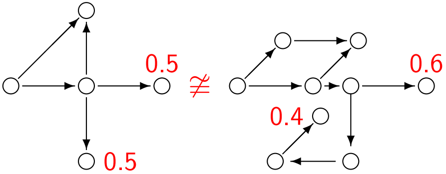


Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Specification matching

Does the "*frequency profile*" of the system match that of the specification?

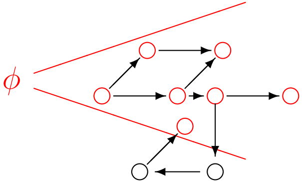


Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Model checking

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

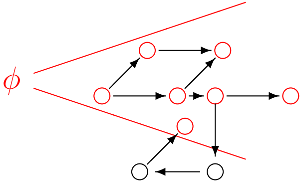


Benefits of Quantification

- Each PEPA expression has an underlying CTMC which can be derived automatically.
- verification can now be complemented by quantitative verification:

Model checking

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



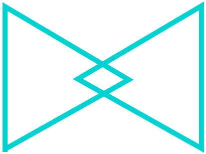
Theory

The theoretical development underpinning PEPA focused on the interaction between the process algebra and the underlying mathematical structure, the Markov process.

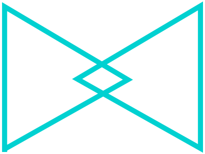
This work can be broadly categorised into three areas:

- Designing the language
- Manipulating models
- Solving models and deriving measures

Designing the language

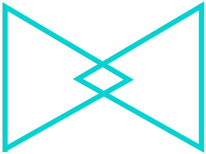


Designing the language



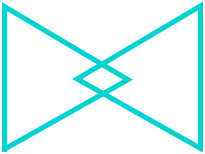
- The issue of what it means for two timed activities to synchronise is a vexed one....

Designing the language



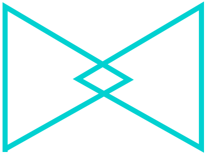
- The issue of what it means for two timed activities to synchronise is a vexed one....
- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.

Designing the language



- The issue of what it means for two timed activities to synchronise is a vexed one....
- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.

Designing the language



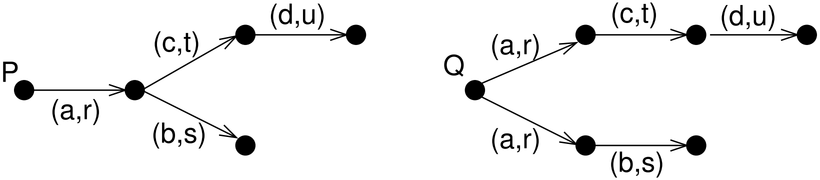
- The issue of what it means for two timed activities to synchronise is a vexed one....
- In PEPA each component has a bounded capacity to carry out activities of any particular type, determined by the apparent rate for that type.
- Synchronisation, or cooperation cannot make a component exceed its bounded capacity.
- Thus the apparent rate of a cooperation is the minimum of the apparent rates of the co-operands.

Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:

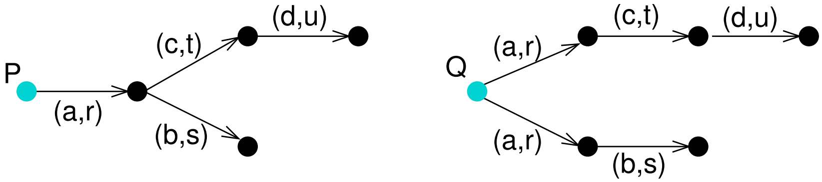
Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:



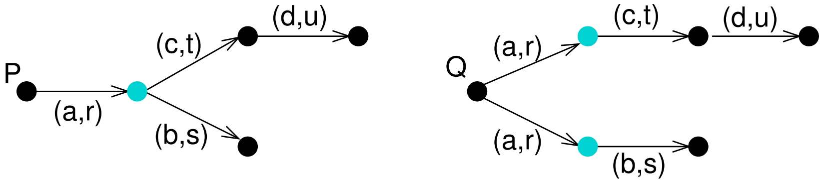
Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:



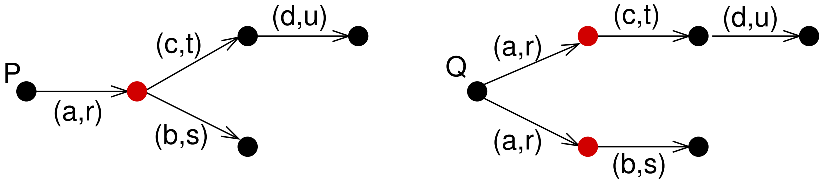
Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:



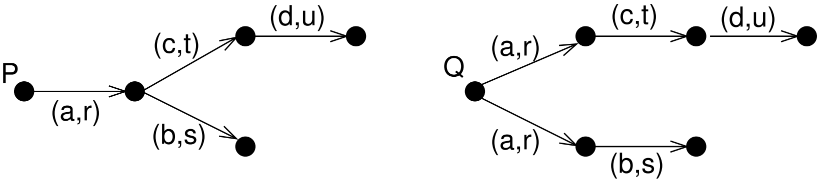
Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:



Semantic Equivalence

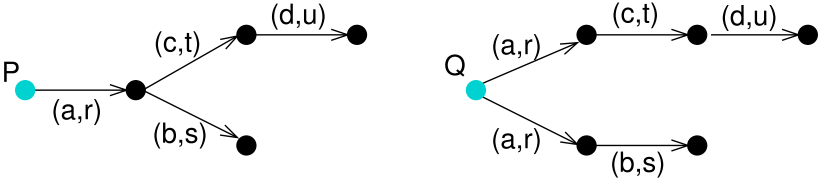
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

Semantic Equivalence

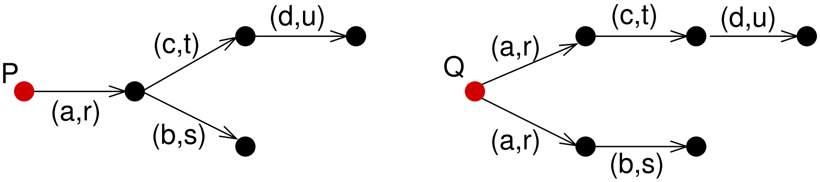
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

Semantic Equivalence

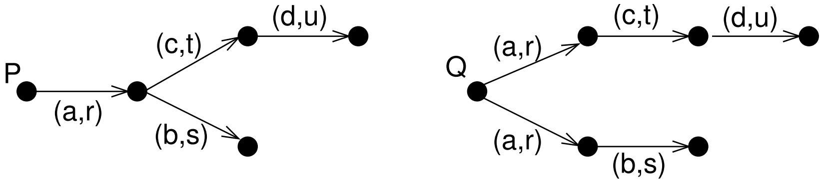
In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

Semantic Equivalence

In process algebra equivalence relations are defined based on the notion of observability:



In PEPA observation is assumed to include the ability to record timing information over a number of runs.

The resulting equivalence relation is a bisimulation in the style of Larsen and Skou, and coincides with the Markov process notion of lumpability.

Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model Manipulation

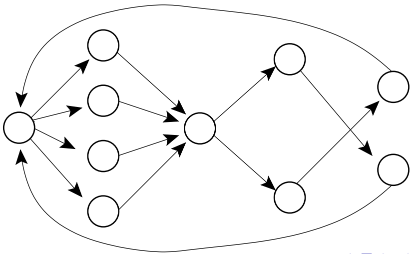
Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.

Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

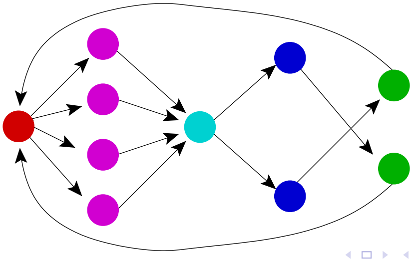
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

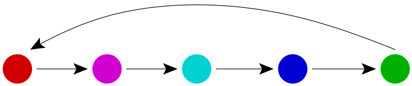
Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



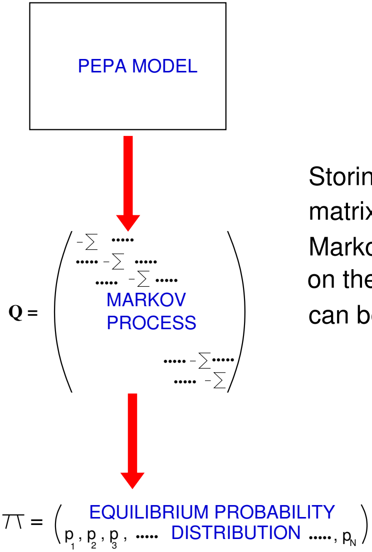
Model Manipulation

Model simplification: use a model-model equivalence to substitute one model by another which is more attractive from a solution point of view, e.g. smaller state space, special class of model, etc.

Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state, i.e. take a different stochastic representation of the same model.



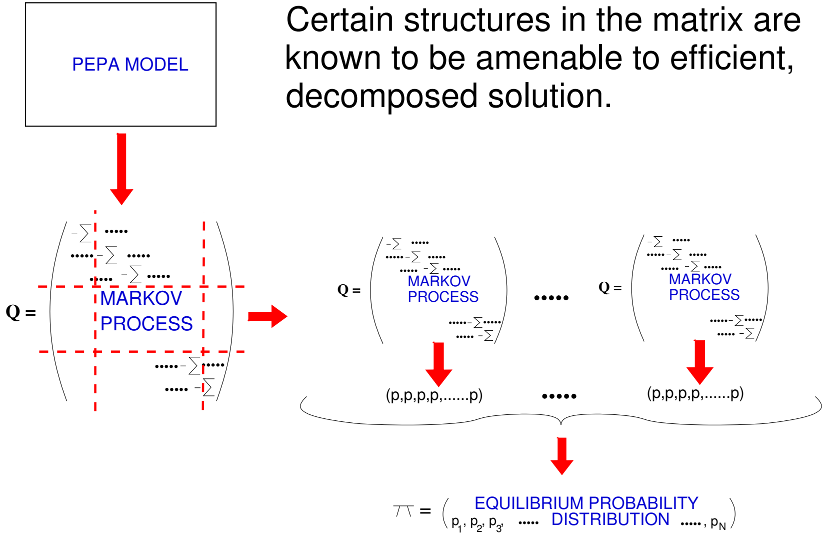
Characterising efficient solution



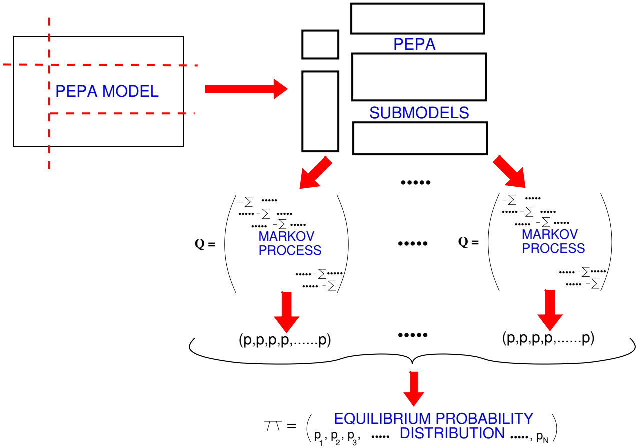
Storing and manipulating the matrix which represents the Markov process places limitations on the size of model which can be analysed.

Characterising efficient solution

Certain structures in the matrix are known to be amenable to efficient, decomposed solution.



Characterising efficient solution



Finding the corresponding structures in the process algebra means that these techniques can be applied automatically, before the monolithic matrix is formed.

Applications

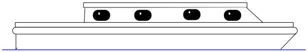
- Developing models of real applications has always been an integral part of the PEPA project.
- This allows us to demonstrate to ourselves and others that the theory we have developed is useful.
- It is also a valuable source of inspiration for new theory and future directions.

PEPA Case Studies (1)

- Protocols for fault-tolerant systems (Clark, Gilmore, Hillston and Ribaudo, Edinburgh and Turin)
- Multimedia traffic characteristics (Bowman et al, Kent)
- Database systems (The STEADY group, Heriot-Watt University)
- Software Architectures (Pooley, Bradley and Thomas, Heriot-Watt and Durham)
- Switch behaviour in active networks (Hillston, Kloul and Mokhtari, Edinburgh and Versailles)
- Mobility and QOS protocols in wireless networks Hillston, Laurenson and Wang, Edinburgh

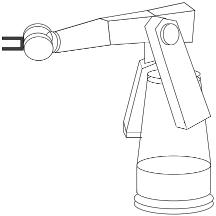
PEPA Case Studies (2)

- Locks and movable bridges in inland shipping in Belgium (Knapen, Hasselt)



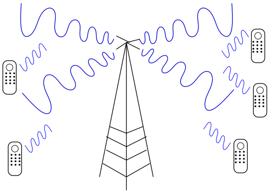
PEPA Case Studies (2)

- Locks and movable bridges in inland shipping in Belgium (Knapen, Hasselt)
- Robotic workcells (Holton, Gilmore and Hillston, Bradford and Edinburgh)



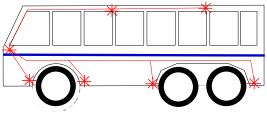
PEPA Case Studies (2)

- Locks and movable bridges in inland shipping in Belgium (Knapen, Hasselt)
- Robotic workcells (Holton, Gilmore and Hillston, Bradford and Edinburgh)
- Cellular telephone networks (Kloul, Fourneau and Valois, Versailles)



PEPA Case Studies (2)

- Locks and movable bridges in inland shipping in Belgium (Knapen, Hasselt)
- Robotic workcells (Holton, Gilmore and Hillston, Bradford and Edinburgh)
- Cellular telephone networks (Kloul, Fourneau and Valois, Versailles)
- Automotive diagnostic expert systems (Console, Picardi and Ribaudo, Turin)



Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

PRISM model checker
(Oxford University)

Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

PRISM model checker
(Oxford University)

Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

PRISM model checker
(Oxford University)

Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

PRISM
model checker
(Oxford University)

Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

PRISM model checker
(Oxford University)

Tool Support

PEPA Eclipse Plug-in
(Edinburgh University)

Möbius modelling platform
(University of Illinois)

PEPA

International PEPA Compiler (IPC)
(Imperial/Edinburgh)

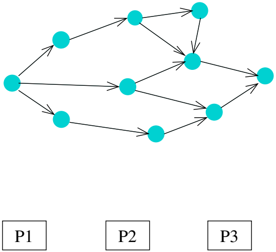
PRISM model checker
(Oxford University)

Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.

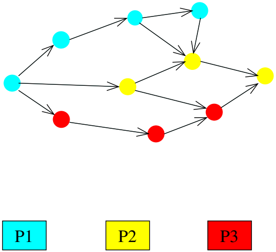
Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



Grid Scheduling

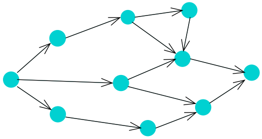
In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.

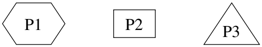


A schedule maps tasks to processors

Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.

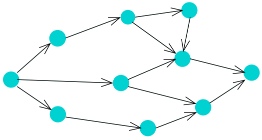


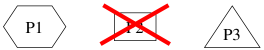


In a grid processors are heterogeneous...

Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.

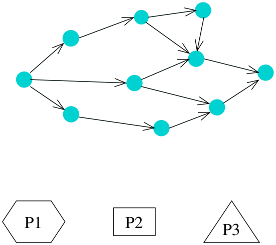




In a grid processors are heterogeneous and dynamic.

Grid Scheduling

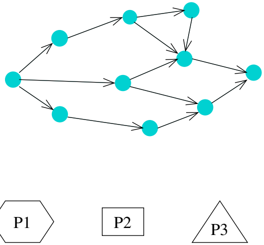
In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



- Current performance parameters obtained from monitoring.

Grid Scheduling

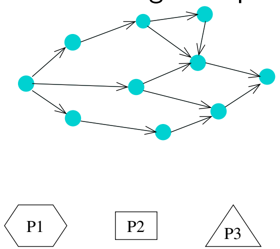
In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



- Current performance parameters obtained from monitoring.
- Highly abstract model components configured to represent different scheduling possibilities.

Grid Scheduling

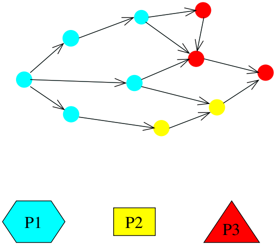
In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



- Current performance parameters obtained from monitoring.
- Highly abstract model components configured to represent different scheduling possibilities.
- Fast evaluation and comparison of alternatives.

Grid Scheduling

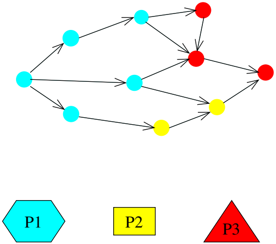
In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



- Current performance parameters obtained from monitoring.
- Highly abstract model components configured to represent different scheduling possibilities.
- Fast evaluation and comparison of alternatives.

Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.

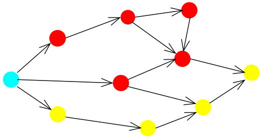


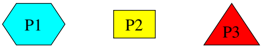
The dynamic nature may mean that tasks have to be re-scheduled during the course of the computation.

- Current performance parameters obtained from monitoring.
- Highly abstract model components configured to represent different scheduling possibilities.
- Fast evaluation and comparison of alternatives.

Grid Scheduling

In the EPSRC-funded ENHANCE research project we investigated ways to use performance predictions to improve scheduling decisions in large computational grids.



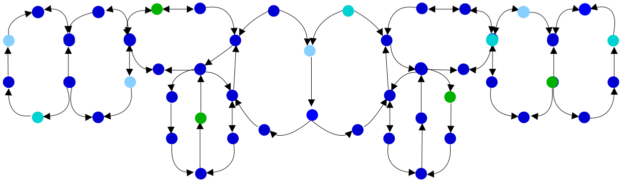


The dynamic nature may mean that tasks have to be re-scheduled during the course of the computation.

- Current performance parameters obtained from monitoring.
- Highly abstract model components configured to represent different scheduling possibilities.
- Fast evaluation and comparison of alternatives.

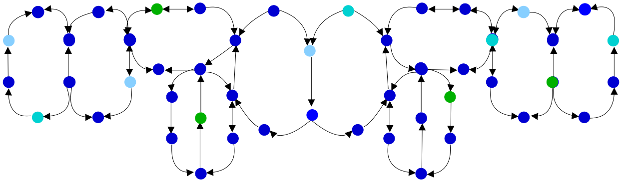
Collective Dynamics

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.

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.

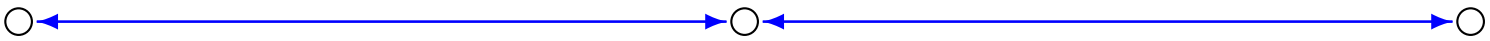
Use continuous state variables to approximate the discrete state space.

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.

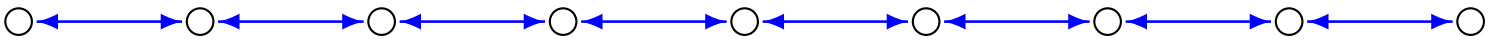
Use continuous state variables to approximate the discrete state space.

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.

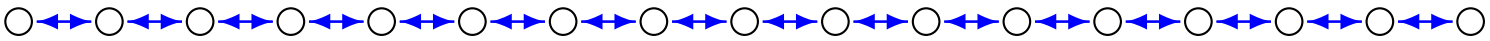
Use continuous state variables to approximate the discrete state space.

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.

Performance Evaluation 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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a set of ODEs.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

- The language is used to generate a set of ODEs.

PEPA
MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a set of ODEs.

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a set of ODEs.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{ABSTRACT MARKOV PROCESS}$$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a set of ODEs.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{ABSTRACT MARKOV PROCESS} \xrightarrow[\text{approximation}]{\text{fluid}}$$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a set of ODEs.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{ABSTRACT MARKOV PROCESS} \xrightarrow[\text{approximation}]{\text{fluid}} \text{VECTOR FIELD } \mathcal{F}(x)$$

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.

Capturing behaviour in the Generator Function

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

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

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

Example: Internet worms

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

Example: Internet worms

- 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.

Example: Internet worms

- 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.

Example: Internet worms

- 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.

Example: Internet worms

- 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 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

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

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

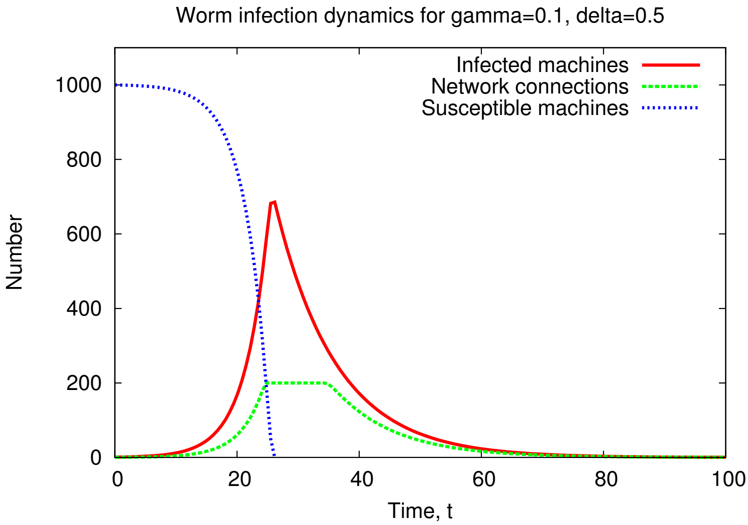
Susceptible-Infective-Removed over a network

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

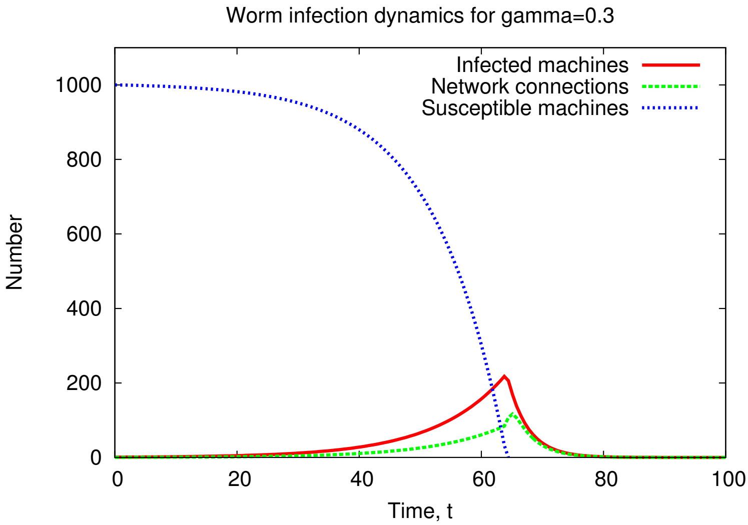
$$
\begin{aligned}
\mathit{Net} &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net}
\end{aligned}
$$

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

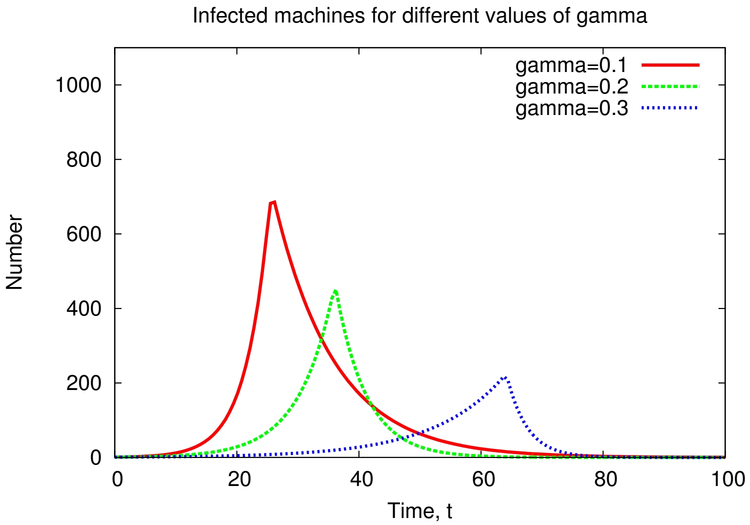
Patch rate $\gamma = 0.1$. Connection failure rate $\delta = 0.5$



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

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

Susceptible-Infective-Removed-Reinfection (SIRR) model

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

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

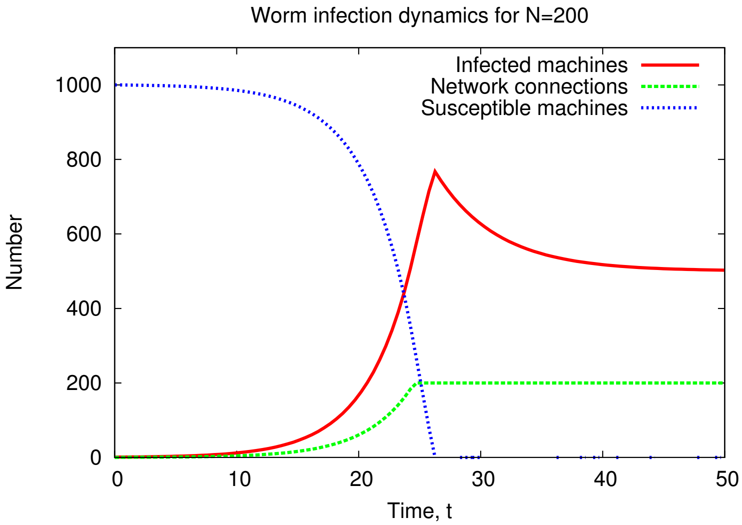
Susceptible-Infective-Removed-Reinfection (SIRR) model

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

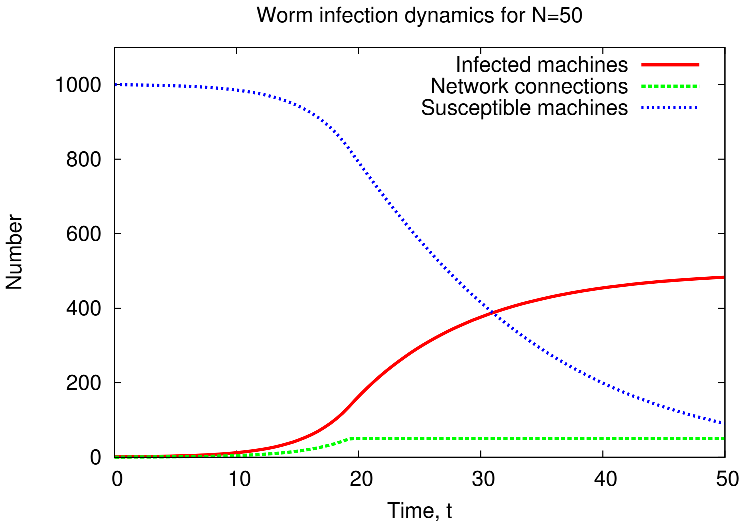
$$
\begin{aligned}
Net &\stackrel{def}{=} (infectI, \top).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

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

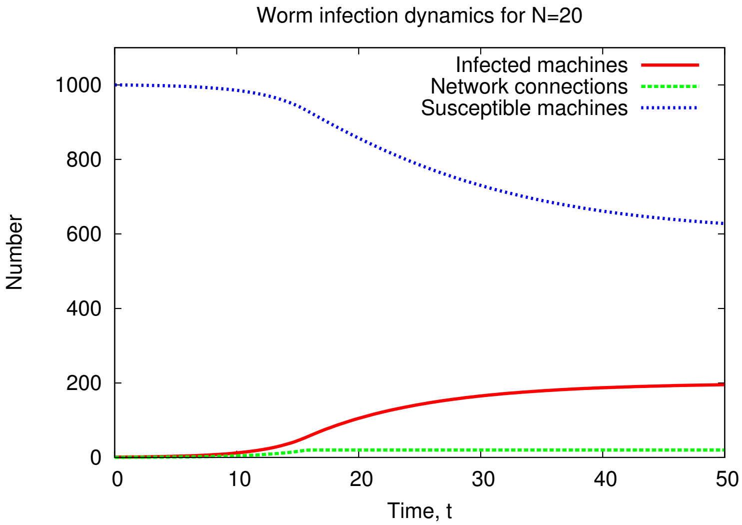
Unsecured SIR model (200 network channels)



Unsecured SIR model (50 network channels)



Unsecured SIR model (20 network channels)



Example 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.

Example 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.

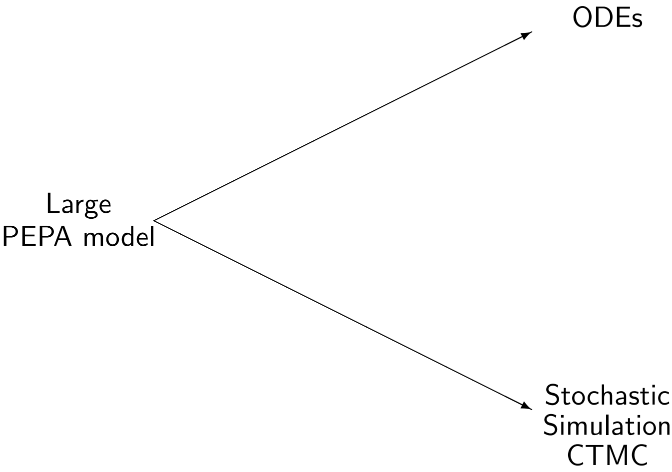
Example Conclusions

- 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.

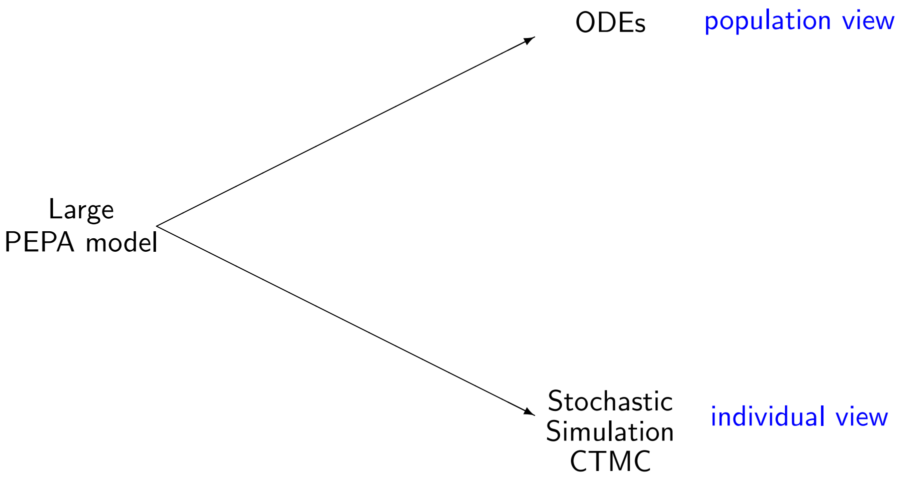
Example Conclusions

- 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.

Alternative Representations



Alternative Representations



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

- PEPA has been successfully used for a number of years as a means of constructing structured Markov chains for performance modelling.

Conclusions

- PEPA has been successfully used for a number of years as a means of constructing structured Markov chains for performance modelling.
- However the state space explosion problem makes this approach infeasible for systems with collective dynamics and emergent behaviour.

Conclusions

- PEPA has been successfully used for a number of years as a means of constructing structured Markov chains for performance modelling.
- However the state space explosion problem makes this approach infeasible for systems with 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

- PEPA has been successfully used for a number of years as a means of constructing structured Markov chains for performance modelling.
- However the state space explosion problem makes this approach infeasible for systems with 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

- PEPA has been successfully used for a number of years as a means of constructing structured Markov chains for performance modelling.
- However the state space explosion problem makes this approach infeasible for systems with 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: funding
Thanks to EPRSC, BBSRC, DTI and CEC IST-FET programme who currently support aspects of work on PEPA.

Thanks!

Acknowledgements: funding
Thanks to EPRSC, BBSRC, DTI and CEC IST-FET programme who currently support aspects of work on PEPA.

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