Population models from PEPA descriptions

Jane Hillston.
LFCS, University of Edinburgh

19th April 2006

Joint work with Jeremy Bradley, Muffy Calder and Stephen Gilmore

Outline

Introduction
- Stochastic Process Algebra
- State Space Explosion

Continuous State Space Models
- Analysis based on Continuous-time Markov Chains
- Analysis based on Ordinary Differential Equations
- Deriving Differential Equations

Case Study in Internet Worms

Conclusions

Outline

Introduction
- Stochastic Process Algebra
- State Space Explosion

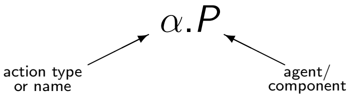
Continuous State Space Models
- Analysis based on Continuous-time Markov Chains
- Analysis based on Ordinary Differential Equations
- Deriving Differential Equations

Case Study in Internet Worms

Conclusions

Process Algebra

- Models consist of agents which engage in actions.



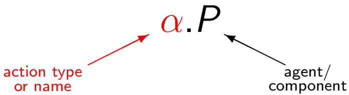
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



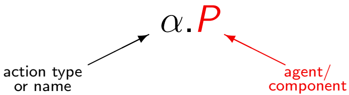
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



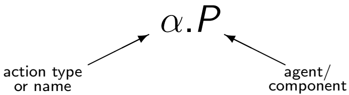
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

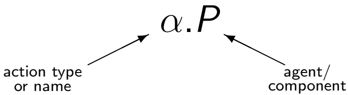
action type
or name

agent/
component

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

Process Algebra

- Models consist of agents which engage in actions.



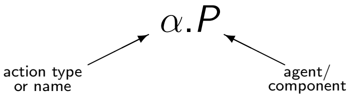
$\alpha.P$

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

Process algebra model

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

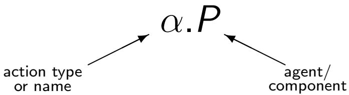
agent/ component

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

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

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type
or name

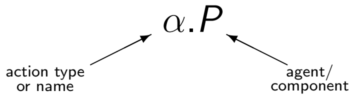
agent/
component

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

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

Process Algebra

- Models consist of agents which engage in actions.



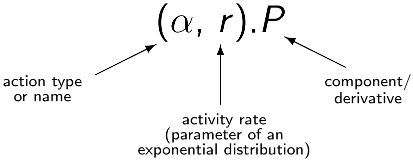
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

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

- Key feature is compositionality

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type or name

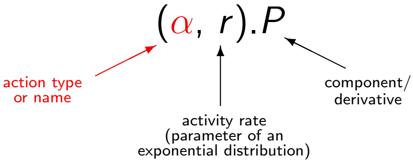
activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type or name

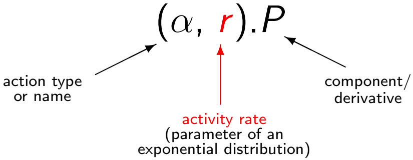
activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

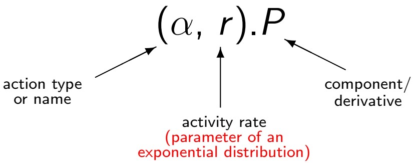
activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

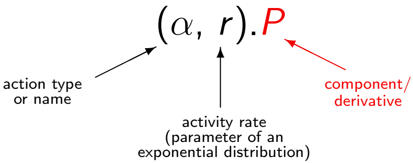
activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

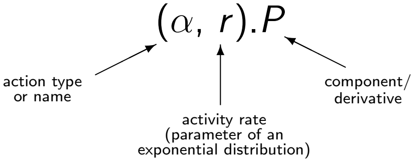
activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



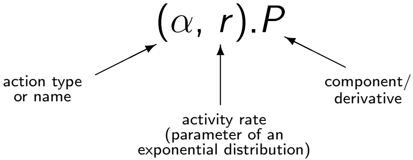
$(\alpha, r).P$

The language may be used to generate a Markov Process (CTMC).

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

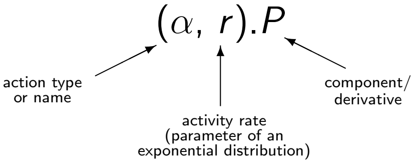
component/ derivative

The language may be used to generate a Markov Process (CTMC).

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



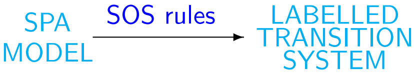
$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).



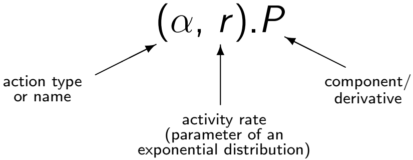
SPA MODEL

SOS rules

LABELLED TRANSITION SYSTEM

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



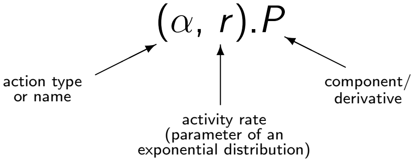
$$(\alpha, r).P$$

The language may be used to generate a Markov Process (CTMC).

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a Markov Process (CTMC).

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

PEPA

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L \end{array}$$

PEPA

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L \end{array}$$

PREFIX: $(\alpha, r).S$ designated first action

PEPA

$$
\begin{array}{rcl}
S & ::= & (\alpha, r).S \mid S + S \mid A \\
P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L
\end{array}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)

PEPA

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \bowtie_{L} P \mid P/L \end{array}$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \overset{def}{=} S$	assigning names

PEPA

$$\begin{array}{lcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L \end{array}$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \overset{\mathit{def}}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*) $\alpha \in L$ cooperative activity (*shared actions*)

PEPA

$$
\begin{array}{rcl}
S & ::= & (\alpha, r).S \mid S + S \mid A \\
P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L
\end{array}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \overset{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*) $\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$

State space explosion

- The CTMC semantics of PEPA and other SPA incurs problems of state space explosion.

State space explosion

- The CTMC semantics of PEPA and other SPA incurs problems of state space explosion. (A problem shared by all discrete state modelling formalisms.)

State space explosion

- The CTMC semantics of PEPA and other SPA incurs problems of state space explosion. (A problem shared by all discrete state modelling formalisms.)
- Numerical solution of the underlying CTMC and model checking become intractable when the state space becomes too large.

State space explosion

- The CTMC semantics of PEPA and other SPA incurs problems of state space explosion. (A problem shared by all discrete state modelling formalisms.)
- Numerical solution of the underlying CTMC and model checking become intractable when the state space becomes too large.
- Symmetries, or repeated copies of the same components, offer ways to tackle this problem.

Individuals vs. Populations

Whenever we have a system consisting of a large number of individuals we can consider treating them instead as a population.

Individuals vs. Populations

Whenever we have a system consisting of a large number of individuals we can consider treating them instead as a population.

These alternative styles of model are already available in the systems biology arena:

Individuals vs. Populations

Whenever we have a system consisting of a large number of individuals we can consider treating them instead as a population.

These alternative styles of model are already available in the systems biology arena:

- Stochastic Simulations (Gillespie *et al.*) are individual-based models in which each molecule is treated separately.

Individuals vs. Populations

Whenever we have a system consisting of a large number of individuals we can consider treating them instead as a population.

These alternative styles of model are already available in the systems biology arena:

- Stochastic Simulations (Gillespie *et al.*) are individual-based models in which each molecule is treated separately.
- Ordinary Differential Equations are population-based models in which populations of molecules are represented abstractly as concentrations.

Individuals vs. Populations

Whenever we have a system consisting of a large number of individuals we can consider treating them instead as a population.

These alternative styles of model are already available in the systems biology arena:

- Stochastic Simulations (Gillespie *et al.*) are individual-based models in which each molecule is treated separately.
- Ordinary Differential Equations are population-based models in which populations of molecules are represented abstractly as concentrations.

These are alternative analysis techniques and we would like access to both and be able to carry out process algebra-based analyses.

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).

SPA
MODEL

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).

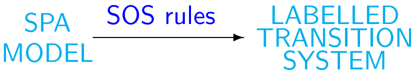
SPA MODEL

SOS rules

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).


SPA MODEL

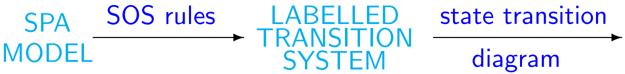
SOS rules

LABELLED TRANSITION SYSTEM

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).



SPA MODEL

SOS rules

LABELLED TRANSITION SYSTEM

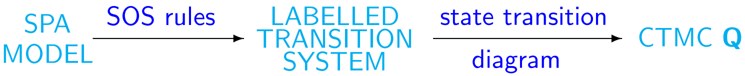
state transition

diagram

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a Markov Process (CTMC).



SPA MODEL —SOS rules→ LABELLED TRANSITION SYSTEM —state transition diagram→ CTMC **Q**

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language also may be used to generate a stochastic simulation.

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

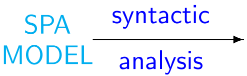
The language also may be used to generate a stochastic simulation.

SPA
MODEL

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language also may be used to generate a stochastic simulation.



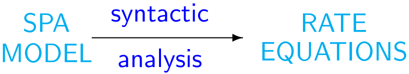
SPA MODEL

syntactic analysis

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language also may be used to generate a stochastic simulation.

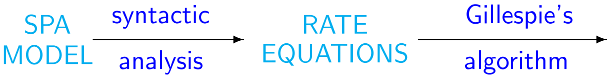


SPA MODEL $\xrightarrow[\text{analysis}]{\text{syntactic}}$ RATE EQUATIONS

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language also may be used to generate a stochastic simulation.


SPA MODEL

syntactic analysis

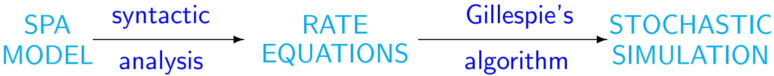
RATE EQUATIONS

Gillespie's algorithm

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language also may be used to generate a stochastic simulation.


SPA MODEL

syntactic analysis

RATE EQUATIONS

Gillespie's algorithm

STOCHASTIC SIMULATION

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a system of ordinary differential equations (ODEs).

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

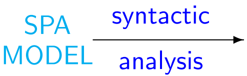
The language may be used to generate a system of ordinary differential equations (ODEs).

SPA
MODEL

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

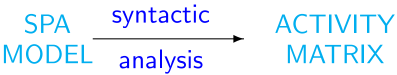
The language may be used to generate a system of ordinary differential equations (ODEs).



Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a system of ordinary differential equations (ODEs).



SPA MODEL

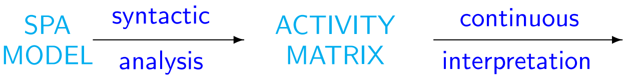
syntactic analysis

ACTIVITY MATRIX

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a system of ordinary differential equations (ODEs).



SPA MODEL

syntactic

analysis

ACTIVITY MATRIX

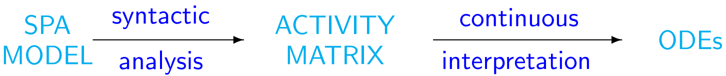
continuous

interpretation

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a system of ordinary differential equations (ODEs).

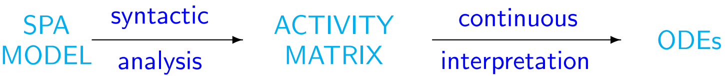


SPA MODEL → syntactic analysis → ACTIVITY MATRIX → continuous interpretation → ODEs

Deriving quantitative data

PEPA models can be analysed for quantified dynamic behaviour in a number of different ways.

The language may be used to generate a system of ordinary differential equations (ODEs).



Each of these has tool support so that the underlying model is derived automatically according to the predefined rules.

Outline

Introduction

Stochastic Process Algebra

State Space Explosion

Continuous State Space Models

Analysis based on Continuous-time Markov Chains

Analysis based on Ordinary Differential Equations

Deriving Differential Equations

Case Study in Internet Worms

Conclusions

Modelling with quantified process algebras

Tiny example

$$P_1 \overset{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \overset{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \overset{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \overset{\mathit{def}}{=} (P_1 \parallel P_1)$$

Modelling with quantified process algebras

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

This example defines a system with nine reachable states:

1. $P_1 \parallel P_1$
2. $P_1 \parallel P_2$
3. $P_1 \parallel P_3$
4. $P_2 \parallel P_1$
5. $P_2 \parallel P_2$
6. $P_2 \parallel P_3$
7. $P_3 \parallel P_1$
8. $P_3 \parallel P_2$
9. $P_3 \parallel P_3$

The transitions between states have quantified duration r which can be evaluated against a CTMC or ODE interpretation.

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 0$:

1. 1.0000
2. 0.0000
3. 0.0000
4. 0.0000
5. 0.0000
6. 0.0000
7. 0.0000
8. 0.0000
9. 0.0000

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 1$:

1. 0.1642
2. 0.1567
3. 0.0842
4. 0.1567
5. 0.1496
6. 0.0804
7. 0.0842
8. 0.0804
9. 0.0432

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \overset{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \overset{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \overset{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 2$:

1. 0.1056
2. 0.1159
3. 0.1034
4. 0.1159
5. 0.1272
6. 0.1135
7. 0.1034
8. 0.1135
9. 0.1012

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 3$:

1. 0.1082
2. 0.1106
3. 0.1100
4. 0.1106
5. 0.1132
6. 0.1125
7. 0.1100
8. 0.1125
9. 0.1119

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 4$:

1. 0.1106
2. 0.1108
3. 0.1111
4. 0.1108
5. 0.1110
6. 0.1113
7. 0.1111
8. 0.1113
9. 0.1116

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 5$:

1. 0.1111
2. 0.1110
3. 0.1111
4. 0.1110
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 6$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 7$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1111
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 0$:	P_1	2.0000
	P_2	0.0000
	P_3	0.0000

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 1$:		
	P_1	0.8121
	P_2	0.7734
	P_3	0.4144

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \overset{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \overset{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \overset{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 2$:

P_1	0.6490
P_2	0.7051
P_3	0.6457

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 3$:		
	P_1	0.6587
	P_2	0.6719
	P_3	0.6692

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 4$:		
	P_1	0.6648
	P_2	0.6665
	P_3	0.6685

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \overset{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \overset{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \overset{\mathit{def}}{=} (stop, r).P_1$$

$$System \overset{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 5$:		
	P_1	0.6666
	P_2	0.6663
	P_3	0.6669

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 6$:		
	P_1	0.6666
	P_2	0.6666
	P_3	0.6666

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 7$:		
	P_1	0.6666
	P_2	0.6666
	P_3	0.6666

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 0$:		
	P_1	3.0000
	P_2	0.0000
	P_3	0.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 1$:		
	P_1	1.1782
	P_2	1.1628
	P_3	0.6590

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 2$:

P_1	0.9766
P_2	1.0754
P_3	0.9479

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 3$:		
	P_1	0.9838
	P_2	1.0142
	P_3	1.0020

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 4$:

P_1	0.9981
P_2	0.9995
P_3	1.0023

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 5$:

P_1	1.0001
P_2	0.9996
P_3	1.0003

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \overset{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \overset{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \overset{\mathit{def}}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 6$:

P_1	1.0001
P_2	0.9999
P_3	1.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \overset{def}{=} (start, r).P_2 \qquad P_2 \overset{def}{=} (run, r).P_3 \qquad P_3 \overset{def}{=} (stop, r).P_1$$

$$System \overset{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 7$:

P_1	1.0000
P_2	0.9999
P_3	0.9999

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 8$:

P_1	1.0000
P_2	1.0000
P_3	1.0000

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

[Stewart, 1994]

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

[Stewart, 1994]

That's not what we're doing. We go directly to ODEs.

Deriving Differential Equations

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

Deriving 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.

Deriving 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.

Deriving 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.

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation (assuming rates are deterministic).

Differential equations from PEPA models

- The PEPA definitions of the component specify the activities which can increase or decrease the number of components exhibited in the current state.
- The cooperations show when the number of instances of another component will have an influence on the evolution of this component.

Differential equations from PEPA models

- The PEPA definitions of the component specify the activities which can increase or decrease the number of components exhibited in the current state.
- The cooperations show when the number of instances of another component will have an influence on the evolution of this component.

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t + \delta t) \; - \; N(\mathcal{C}_{i_j}, t) = \\ - \underbrace{\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{exit activities}} \\ + \underbrace{\sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \; - \; N(\mathcal{C}_{i_j}, t) = \underbrace{- \sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{exit activities}} \underbrace{+ \sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \; - \; N(\mathcal{C}_{i_j}, t) = \\ \underbrace{- \sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{exit activities}} \\ \underbrace{+ \sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$- \underbrace{\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t + \delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$- \underbrace{\sum_{(\alpha,r) \in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t)) \, \delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r) \in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t)) \, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Dividing by δt and taking the limit, $\delta t \longrightarrow 0$:

$$\frac{dN(C_{i_j}, t)}{dt} = -\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t)) \\ + \sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t))$$

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Outline

Introduction

Stochastic Process Algebra

State Space Explosion

Continuous State Space Models

Analysis based on Continuous-time Markov Chains

Analysis based on Ordinary Differential Equations

Deriving Differential Equations

Case Study in Internet Worms

Conclusions

Background

- Internet worms (e.g. Nimbda, Slammer, Code Red, Sasser and Code Red 2) are malicious programs that exploit operating system security weaknesses to propagate themselves.

Background

- Internet worms (e.g. Nimbda, Slammer, Code Red, Sasser and Code Red 2) 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.

Background

- Internet worms (e.g. Nimbda, Slammer, Code Red, Sasser and Code Red 2) 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 [Nicol 2003], due to overload from nonexistent IP lookups.

Background

- Internet worms (e.g. Nimbda, Slammer, Code Red, Sasser and Code Red 2) 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 [Nicol 2003], due to overload from nonexistent IP lookups.
- The estimated cost of computer worms and related activities is about $50 billion a year [Slate 2004].

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.
- Markovian process algebras are founded on an interleaving semantics. Existing 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.
- Markovian process algebras are founded on an interleaving semantics. Existing 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.

Derived forms and additional syntax

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

Derived forms and additional syntax

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

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

Derived forms and additional syntax

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

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

When working with large numbers of hosts which transmit worm infections, we write $P[n]$ to denote an array of n copies of P executing in parallel.

Derived forms and additional syntax

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

Because we are interested in transient behaviour we use the deadlocked process *Stop*.

When working with large numbers of hosts which transmit worm infections, we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

Suceptible-Infective-Removed (SIR) model

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

Suceptible-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.

Suceptible-Infective-Removed (SIR) model

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

Suceptible-Infective-Removed (SIR) model

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

Suceptible-Infective-Removed (SIR) model

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

Suceptible-Infective-Removed over a network

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

Suceptible-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.

Suceptible-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.

Suceptible-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.

Suceptible-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.

Suceptible-Infective-Removed over a network

- The parameter M denotes the number of concurrent, independent connections that the network can sustain.

Suceptible-Infective-Removed over a network

- The parameter M denotes the number of concurrent, independent connections that the network can sustain.
- An attempted network connection can fail or timeout, indicated by the *fail* action.

Suceptible-Infective-Removed over a network

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

Suceptible-Infective-Removed over a network

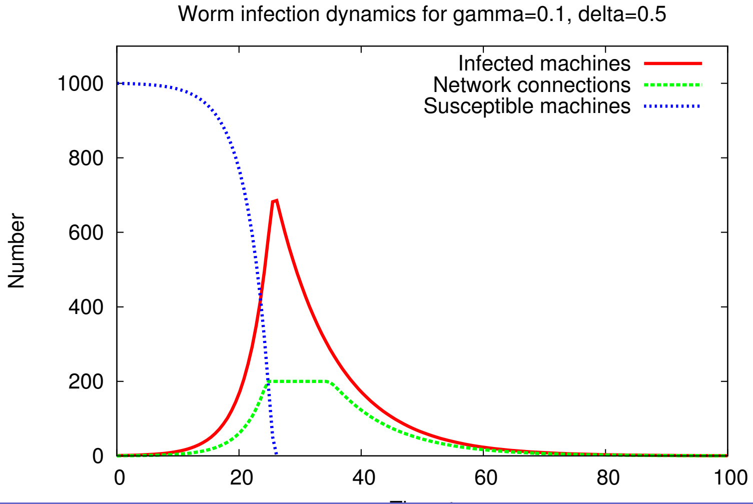
- The parameter M denotes the number of concurrent, independent connections that the network can sustain.
- An attempted network connection can fail or timeout, indicated by the *fail* action.
- This might be due to network contention or the lack of availability of a susceptible machine to infect.
- A certain number of infections will attempt to reinfect hosts; in this instance, the host is unaffected.

Suceptible-Infective-Removed over a network

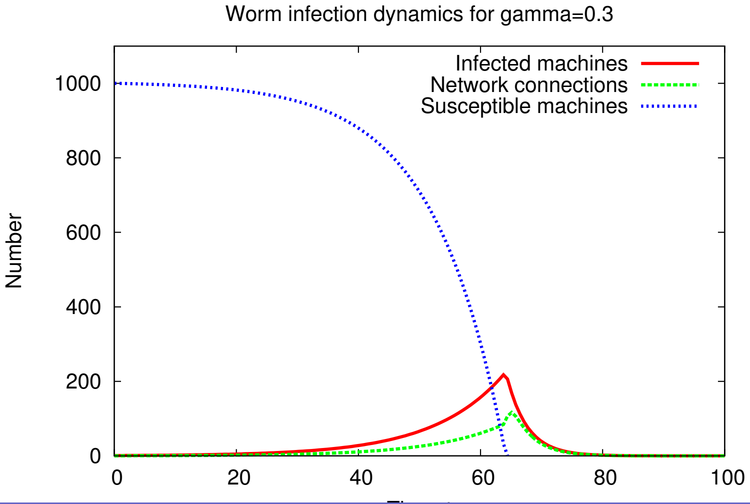
$$
\begin{aligned}
S &\overset{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\overset{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\overset{\mathit{def}}{=} \mathit{Stop} \\
\mathit{Net} &\overset{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\overset{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net} \\
\mathit{Sys} &\overset{\mathit{def}}{=} (S[N] \parallel I) \underset{L}{\bowtie} \mathit{Net}[M]
\end{aligned}
$$

where $L = \{\, \mathit{infectI}, \mathit{infectS} \,\}$

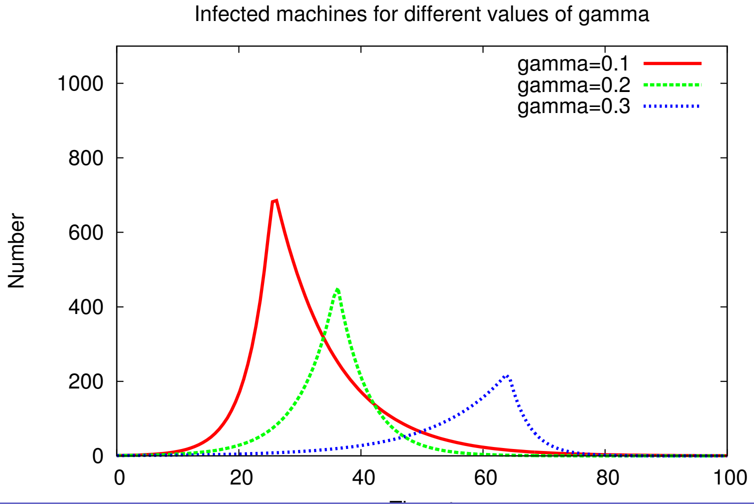
Patch rate $\gamma = 0.1$. Connection failure rate $\delta = 0.5$



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

- Here 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

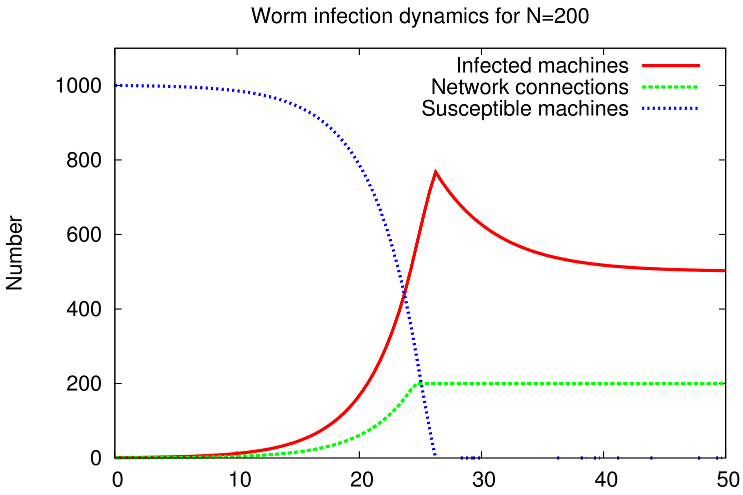
- Here a small modification in the process model of infection allows for removed computers to become susceptible again after a delay.
- We use this to model a faulty or incomplete security upgrade or the mistaken removal of security patches which had previously defended the machine against attack.

Susceptible-Infective-Removed-Reinfection (SIRR) model

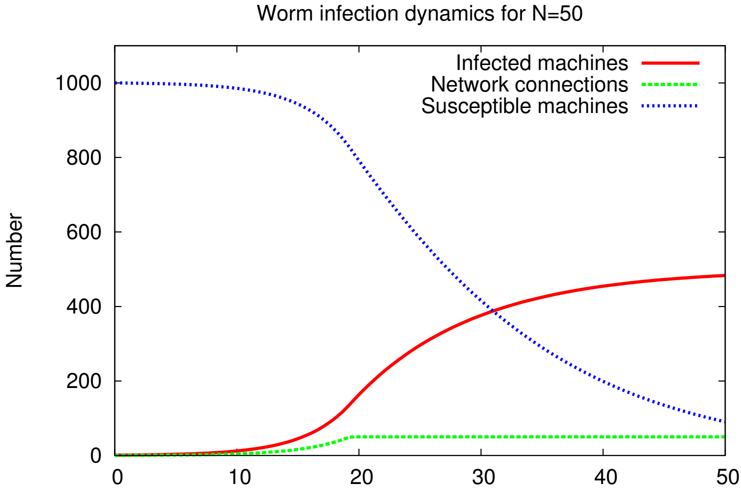
$$
\begin{aligned}
S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\mathit{def}}{=} (\mathit{unsecure}, \mu).S \\
\mathit{Net} &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net} \\
\mathit{Sys} &\stackrel{\mathit{def}}{=} (S[1000] \parallel I) \underset{L}{\bowtie} \mathit{Net}[M]
\end{aligned}
$$

where $L = \{\mathit{infectI}, \mathit{infectS}\}$.

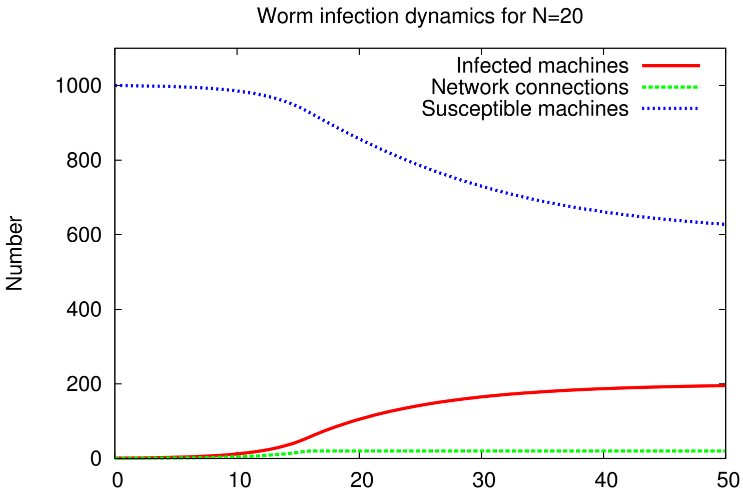
Unsecured SIR model ($M = 200$ network channels)



Unsecured SIR model ($M = 50$ network channels)



Unsecured SIR model ($M = 20$ network channels)



Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- Some worms have bimodal behaviour — either a worm can infect another computer or it can start an attack on a victim computer.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- Some worms have bimodal behaviour — either a worm can infect another computer or it can start an attack on a victim computer.
- The attack may be as simple as requesting a specific web page, or issuing a *ping* request.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

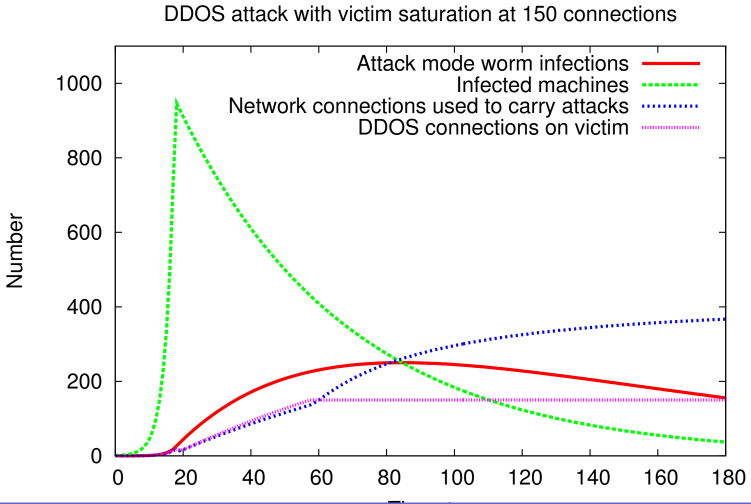
- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- Some worms have bimodal behaviour — either a worm can infect another computer or it can start an attack on a victim computer.
- The attack may be as simple as requesting a specific web page, or issuing a *ping* request.
- The combination of perhaps millions of machines making such requests quickly overwhelms the target computer, which either crashes under the huge load, or becomes unusably slow.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

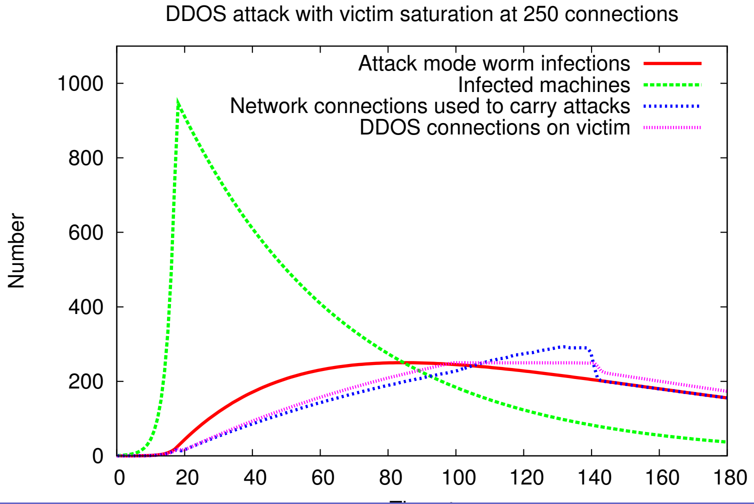
$$\begin{aligned}
S &\overset{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\overset{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R + (\mathit{attack}, \chi).A \\
A &\overset{\mathit{def}}{=} (\mathit{attackA}, \lambda).A + (\mathit{patch}, \gamma).R \\
R &\overset{\mathit{def}}{=} \mathit{Stop} \\
\mathit{Net} &\overset{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' + (\mathit{attackA}, \top).\mathit{Net}'' \\
\mathit{Net}' &\overset{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net} \\
\mathit{Net}'' &\overset{\mathit{def}}{=} (\mathit{attackV}, \rho).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net} \\
V &\overset{\mathit{def}}{=} (\mathit{attackV}, \top).V' \\
V' &\overset{\mathit{def}}{=} (\mathit{release}, \sigma).V \\
\mathit{Sys} &\overset{\mathit{def}}{=} (S[100] \parallel I \parallel V) \underset{L}{\bowtie} \mathit{Net}[M]
\end{aligned}$$

where $L = \{\mathit{infectI}, \mathit{infectS}, \mathit{attackA}, \mathit{attackV}\}$.

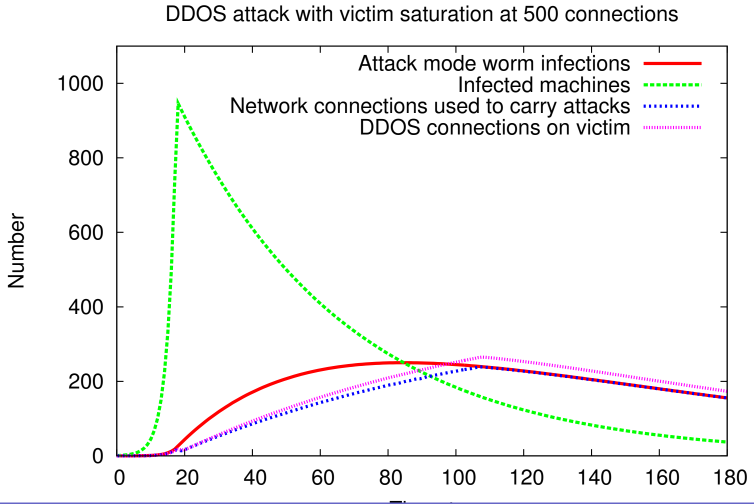
DDOS attack that overwhelms a victim machine



DDOS attack that briefly incapacitates a victim machine



DDOS attack that does not saturate the victim's capacity



Case Study 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.

Case Study 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.
- Large-scale modelling can be effective here, because it abstracts away from modelling of individual behaviour and considers population-based representations.

Case Study 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.
- Large-scale modelling can be effective here, because it abstracts away from modelling of individual behaviour and considers population-based representations.
- The scale of problems which can be modelled in this way vastly exceeds those which are founded on explicit state representations.

Outline

Introduction
- Stochastic Process Algebra
- State Space Explosion

Continuous State Space Models
- Analysis based on Continuous-time Markov Chains
- Analysis based on Ordinary Differential Equations
- Deriving Differential Equations

Case Study in Internet Worms

Conclusions

Conclusions

The derivation of differential equations appears to offer an interesting alternative to existing modelling approaches to performance evaluation of large scale models.

Alternative Representations

ODEs
(Continuous Approximation)

Stochastic Simulation

Alternative Representations

SPA models with discrete levels of population

ODEs
(Continuous Approximation)

SPA models of individuals

Stochastic Simulation

Alternative Representations

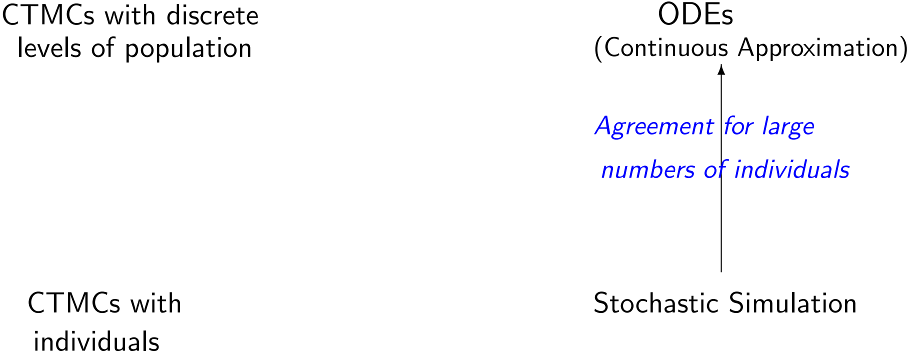
CTMCs with discrete
levels of population

ODEs
(Continuous Approximation)

CTMCs with
individuals

Stochastic Simulation

Alternative Representations



CTMCs with discrete levels of population

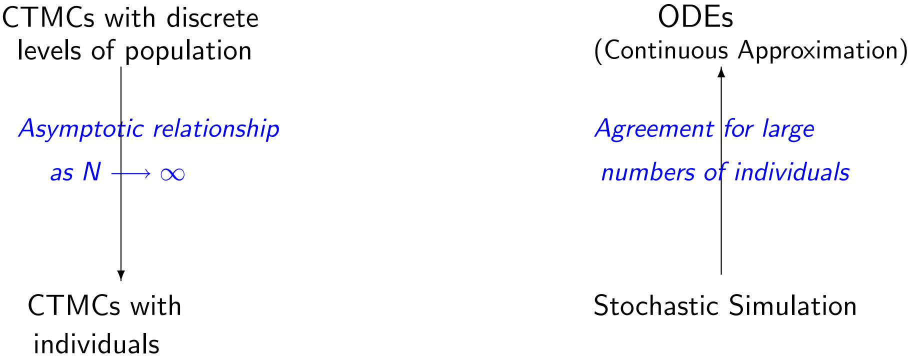
ODEs (Continuous Approximation)

Agreement for large numbers of individuals

CTMCs with individuals

Stochastic Simulation

Alternative Representations



CTMCs with discrete levels of population

Asymptotic relationship as $N \longrightarrow \infty$

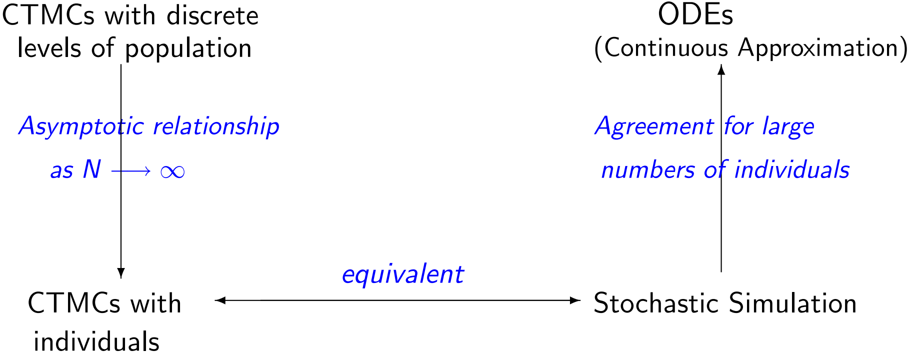
CTMCs with individuals

ODEs (Continuous Approximation)

Agreement for large numbers of individuals

Stochastic Simulation

Alternative Representations



CTMCs with discrete levels of population

Asymptotic relationship as $N \longrightarrow \infty$

CTMCs with individuals

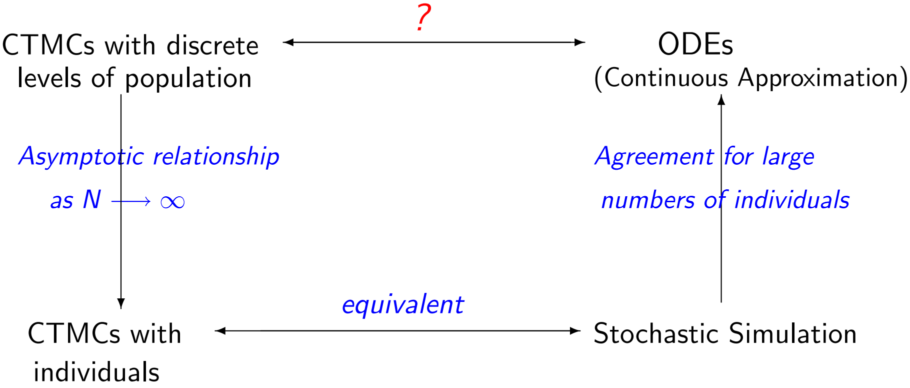
equivalent

Stochastic Simulation

Agreement for large numbers of individuals

ODEs (Continuous Approximation)

Alternative Representations



?

CTMCs with discrete levels of population

ODEs (Continuous Approximation)

Asymptotic relationship

as $N \longrightarrow \infty$

Agreement for large

numbers of individuals

equivalent

CTMCs with individuals

Stochastic Simulation

Future Work

- Establish the relationship between the ODE model and the population model CTMC

Future Work

- Establish the relationship between the ODE model and the population model CTMC
  - When can we use verification (ie model checking) on a CTMC population model and expect the results to hold with respect to the ODE population model?

Future Work

- Establish the relationship between the ODE model and the population model CTMC
    - When can we use verification (ie model checking) on a CTMC population model and expect the results to hold with respect to the ODE population model?
- Reintroduce a stochastic element:

Future Work

- Establish the relationship between the ODE model and the population model CTMC
    - When can we use verification (ie model checking) on a CTMC population model and expect the results to hold with respect to the ODE population model?
- Reintroduce a stochastic element:
    - Use of random or stochastic differential equations;

Future Work

- Establish the relationship between the ODE model and the population model CTMC
  - When can we use verification (ie model checking) on a CTMC population model and expect the results to hold with respect to the ODE population model?
- Reintroduce a stochastic element:
  - Use of random or stochastic differential equations;
  - Limit the continuous element to a few continuous component types, with others having usual CTMC semantics (c.f. fluid stochastic Petri models)

Thank You!