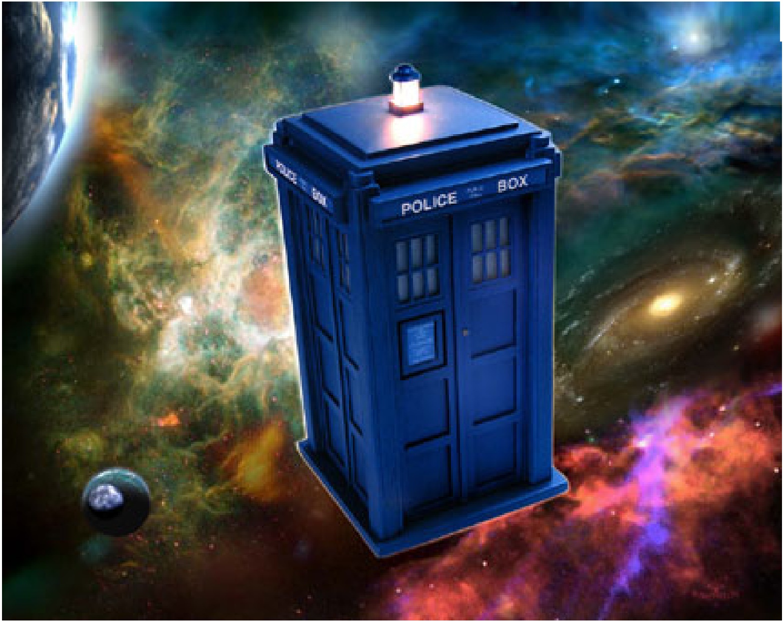
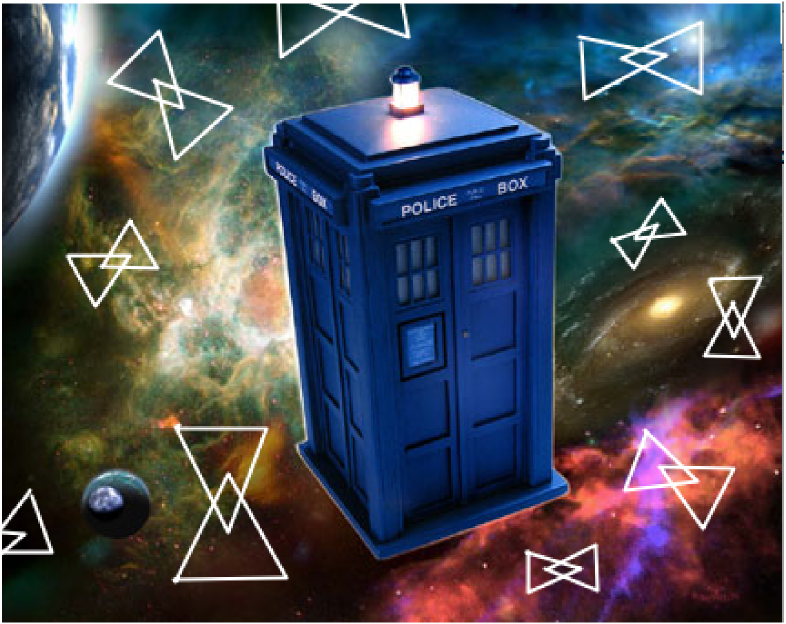
Rigorous and Random Adventures in Time and Space

Jane Hillston

21st November 2012


POLICE BOX


POLICE BOX

Outline

Outline

Performance Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

It has been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed often based on product form results.

But as computer systems have developed these techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems.

Performance Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

It has been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed often based on product form results.

But as computer systems have developed these techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems.

Performance Modelling

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

It has been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed often based on product form results.

But as computer systems have developed these techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems.

Performance Modelling

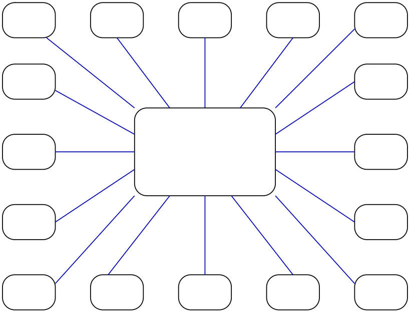
Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

It has been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Originally queueing networks were primarily used to construct models, and sophisticated analysis techniques were developed often based on product form results.

But as computer systems have developed these techniques are no longer widely applicable for expressing the dynamic behaviour observed in distributed systems.

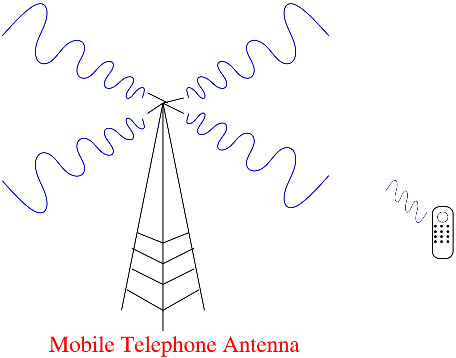
Performance Modelling: Motivation



Capacity planning

- How many clients can the existing server support and maintain reasonable response times?

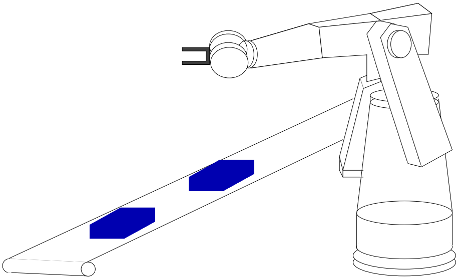
Performance Modelling: Motivation



System Configuration

- How many frequencies do you need to keep blocking probabilities low?

Performance Modelling: Motivation



System Tuning

- What speed of conveyor belt will minimize robot idle time and maximize throughput whilst avoiding lost widgets?

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

Gary Linden, Amazon, quoted on `http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency`

AOL, Bing and Google report similar findings.

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

Gary Linden, Amazon, quoted on http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency

AOL, Bing and Google report similar findings.

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

Gary Linden, Amazon, quoted on `http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency`

AOL, Bing and Google report similar findings.

Does timeliness matter...?

There is sometimes a perception in software development that performance does not matter much, or that it is easily fixed later by buying a faster machine.

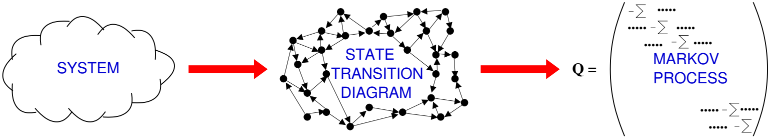
On the contrary — studies have shown that response time is a key feature in user satisfaction and trust in systems.

In a recent study by Amazon they artificially delayed page loading times in increments of 100 milliseconds. Even such very small delays were observed to result in substantial and costly drops in revenue.

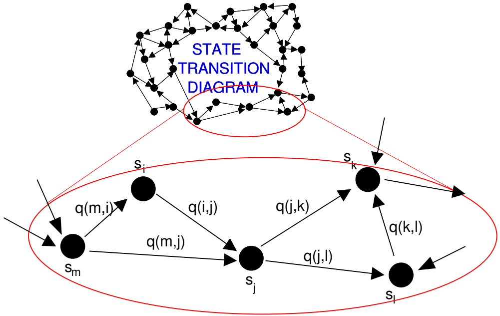
Gary Linden, Amazon, quoted on http://perspectives.mvdirona.com/2009/10/31/TheCostOfLatency

AOL, Bing and Google report similar findings.

Performance Modelling using CTMC

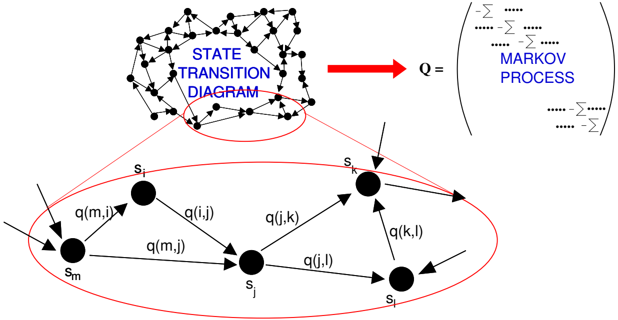


Performance Modelling using CTMC



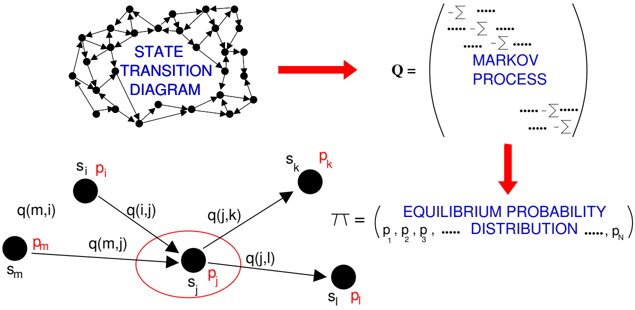
A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



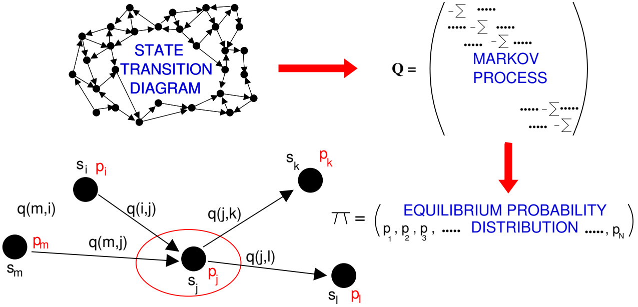
these parameters form the entries of the infinitesimal generator matrix Q

Performance Modelling using CTMC



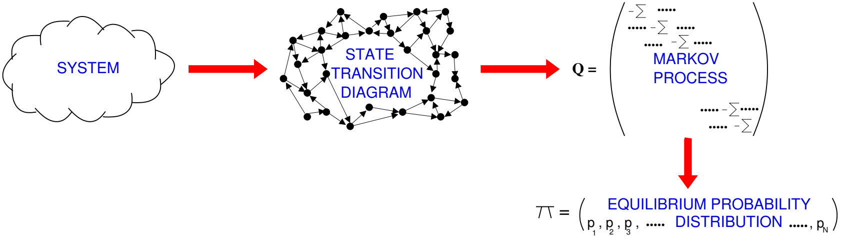
In steady state the probability flux out of a state is balanced by the flux in.

Performance Modelling using CTMC

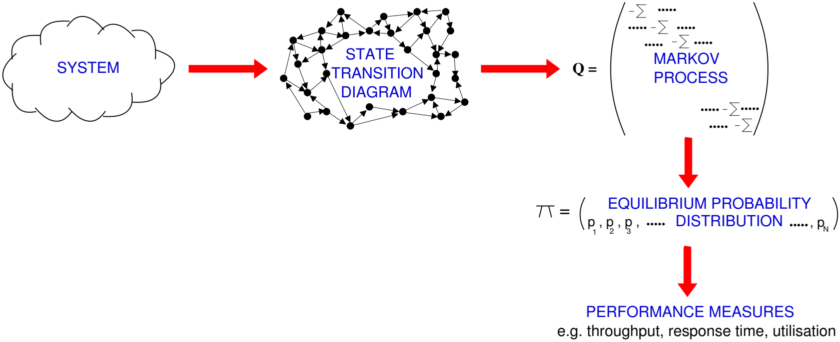


"Global balance equations" captured by $\pi Q = 0$ solved by linear algebra

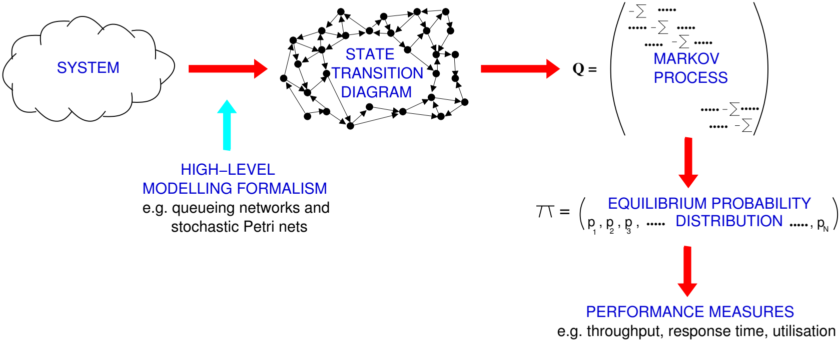
Performance Modelling using CTMC



Performance Modelling using CTMC



Performance Modelling using CTMC



High-level modelling languages are used to automatically generate the state transition diagram/infinitesimal generator matrix **Q**, lifting the description to a level closer to the system behaviour.

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 $\xrightarrow{\text{SOS rules}}$ Labelled transition system

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 $\xrightarrow{\text{SOS rules}}$ Labelled transition system

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 $\xrightarrow{\text{SOS rules}}$ Labelled transition system

Stochastic process algebras

Process algebras where models are decorated with quantitative information used to generate a stochastic process are stochastic process algebras (SPA).

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 is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.

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

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 is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.

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

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 is used to generate a Continuous Time Markov Chain (CTMC) for performance modelling.

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

Integrated analysis

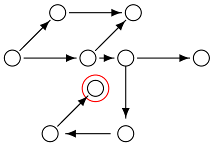
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

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

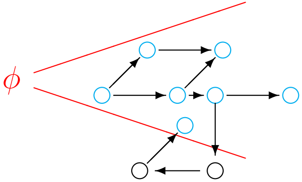


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

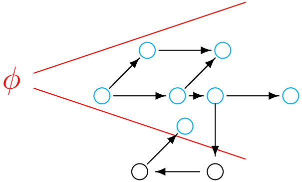


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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



Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

Performance Evaluation Process Algebra (PEPA)

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

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

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

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

A simple example: processors and resources

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

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

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

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

A simple example: processors and resources

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

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

$$
\begin{tikzcd}
 & Proc_0 \underset{\{task1\}}{\bowtie} Res_0 \arrow[d, "{(task1, R)}"] & \\
 & Proc_1 \underset{\{task1\}}{\bowtie} Res_1 \arrow[dl, "{(reset, r_4)}"'] \arrow[dr, "{(task2, r_2)}"] & \\
Proc_1 \underset{\{task1\}}{\bowtie} Res_0 \arrow[uur, "{(task2, r_2)}"] & & Proc_0 \underset{\{task1\}}{\bowtie} Res_1 \arrow[uul, "{(reset, r_4)}"']
\end{tikzcd}
$$

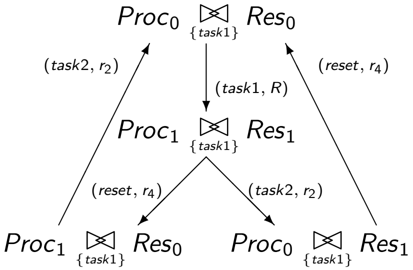
$$R = \min(r_1, r_3)$$

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

A simple example: processors and resources

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

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



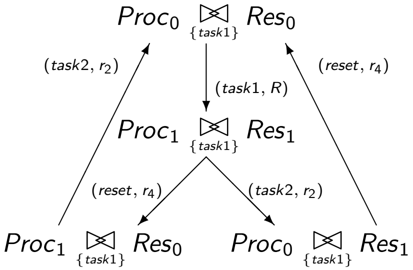
$$
R = \min(r_1, r_3)
$$

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

A simple example: processors and resources

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

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



$R = \min(r_1, r_3)$

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

SPA Languages

SPA

SPA Languages

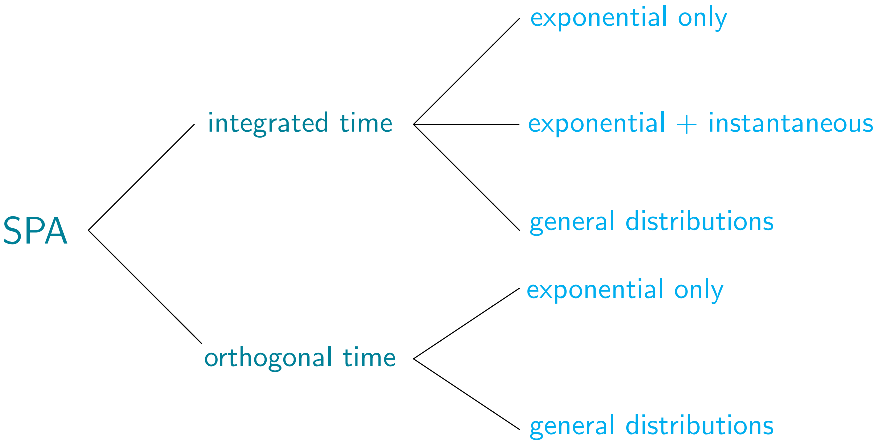
SPA

- integrated time
- orthogonal time

SPA Languages

- SPA
  - integrated time
    - exponential only
    - exponential + instantaneous
    - general distributions
  - orthogonal time

SPA Languages



SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic π-calculus
    - exponential $+$ instantaneous
    - general distributions
  - orthogonal time
    - exponential only
    - general distributions

SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic π-calculus
    - exponential + instantaneous
      - EMPA, Markovian TIPP
    - general distributions
  - orthogonal time
    - exponential only
    - general distributions

SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic π-calculus
    - exponential + instantaneous
      - EMPA, Markovian TIPP
    - general distributions
      - TIPP, SPADES, GSMPA
  - orthogonal time
    - exponential only
    - general distributions

SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic π-calculus
    - exponential + instantaneous
      - EMPA, Markovian TIPP
    - general distributions
      - TIPP, SPADES, GSMPA
  - orthogonal time
    - exponential only
      - IMC
    - general distributions

SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Stochastic π-calculus
    - exponential + instantaneous
      - EMPA, Markovian TIPP
    - general distributions
      - TIPP, SPADES, GSMPA
  - orthogonal time
    - exponential only
      - IMC
    - general distributions
      - IGSMP, Modest

Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

Why use a process algebra?

- High level description of the system eases the task of model construction.
- Formal language allows for unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.

What has been the impact?

The interplay between performance modelling/quantitative modelling and formal methods has stimulated a lot of exciting work.

Major conferences in the field now regularly include probabilistic and stochastic model checking, stochastic logics, bisimulations and other equivalence relations.

Most benefit is gained when properties of models can be established in terms of the model language and its semantics, allowing techniques to be applied without recourse to having to prove applicability on a model-by-model basis.

What has been the impact?

The interplay between performance modelling/quantitative modelling and formal methods has stimulated a lot of exciting work.

Major conferences in the field now regularly include probabilistic and stochastic model checking, stochastic logics, bisimulations and other equivalence relations.

Most benefit is gained when properties of models can be established in terms of the model language and its semantics, allowing techniques to be applied without recourse to having to prove applicability on a model-by-model basis.

What has been the impact?

The interplay between performance modelling/quantitative modelling and formal methods has stimulated a lot of exciting work.

Major conferences in the field now regularly include probabilistic and stochastic model checking, stochastic logics, bisimulations and other equivalence relations.

Most benefit is gained when properties of models can be established in terms of the model language and its semantics, allowing techniques to be applied without recourse to having to prove applicability on a model-by-model basis.

Benefits of process algebra

For example,

- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the CSL family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of process algebra

For example,

- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the CSL family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of process algebra

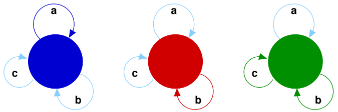
For example,

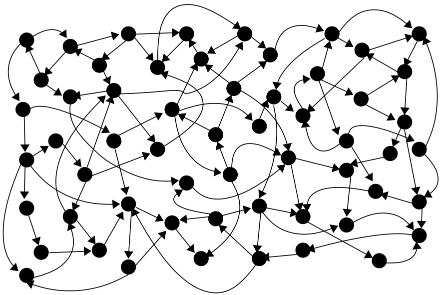
- The correspondence between the congruence, Markovian bisimulation, in the process algebra and the lumpability condition in the CTMC, allows exact model reduction to be carried out compositionally.
- Characterisation of product form structure at the process algebra level allows decomposed model solution based on the process algebra structure of the model.
- Stochastic model checking based on the CSL family of temporal logics allows automatic evaluation of quantified properties of the behaviour of the system.

Outline

Solving discrete state models

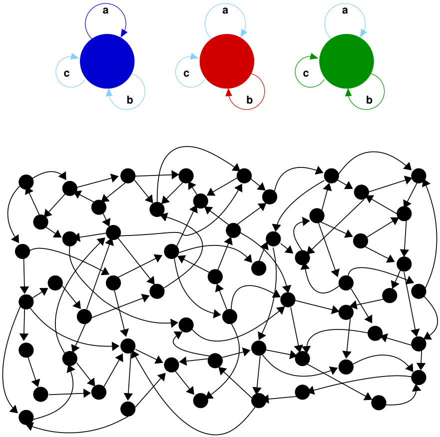
Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.



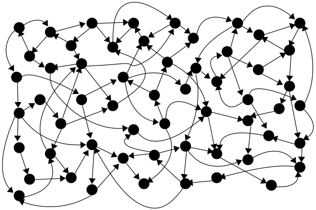


Solving discrete state models

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



Solving discrete state models

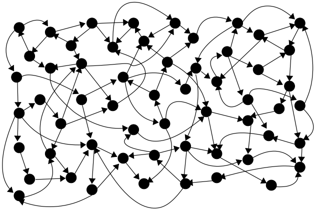


When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models

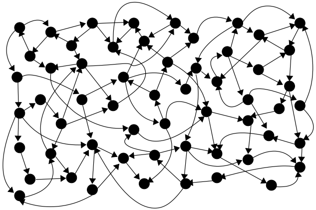


When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models



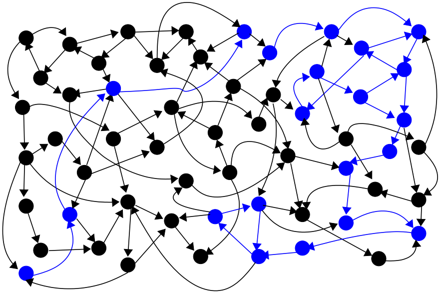
When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

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

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

Solving discrete state models

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.



State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

Aggregation and lumpability

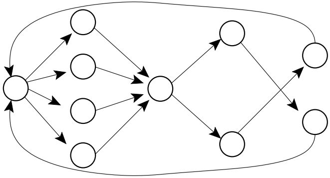
- Model aggregation: use a state-state equivalence to establish a partition of the state space of a model, and replace each set of states by one macro-state
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.

Aggregation and lumpability

- 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
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.

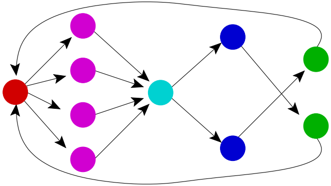
Aggregation and lumpability

- 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
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.



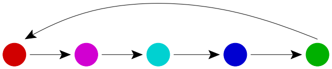
Aggregation and lumpability

- 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
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.



Aggregation and lumpability

- 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
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.
- A lumpable partition is the only partition of a Markov process which preserves the Markov property.



Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.
- This is something of a catch-22 situation when the problem is that the state space of the Markov chain is too large to handle.

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.
- This is something of a catch-22 situation when the problem is that the state space of the Markov chain is too large to handle.

Lumpability

- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.
- This is something of a catch-22 situation when the problem is that the state space of the Markov chain is too large to handle.

Lumpability

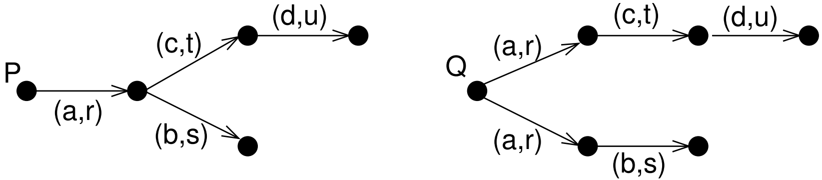
- In the early 1960's Kemeny and Snell established the conditions under which it was possible to lump a Markov chain and still have a Markov chain afterwards.
- In particular these conditions were characterised by conditions on the rates which are straightforward to check.
- However checking the conditions did involve constructing the complete Markov chain first.
- This is something of a catch-22 situation when the problem is that the state space of the Markov chain is too large to handle.

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.

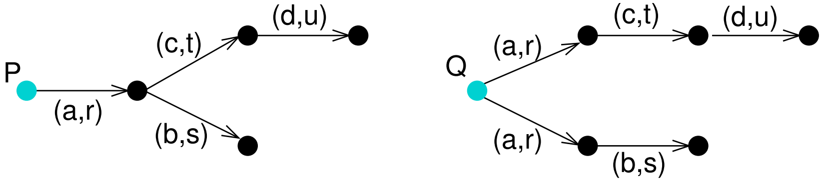
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



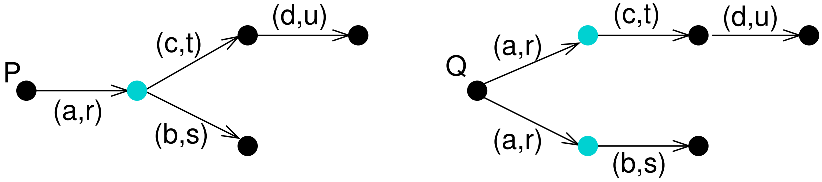
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



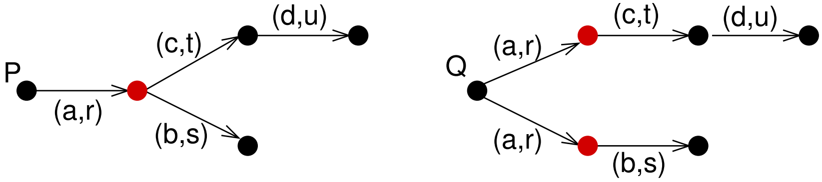
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



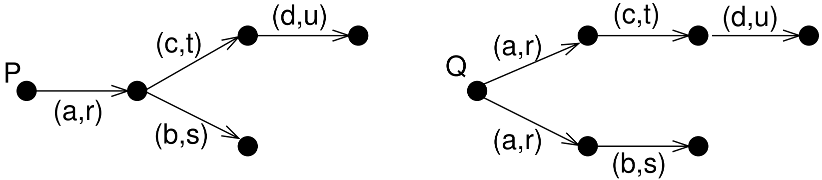
Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Strong Equivalence in PEPA

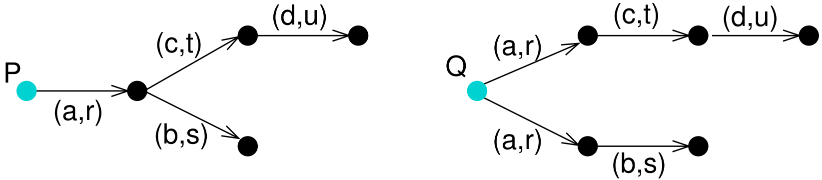
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

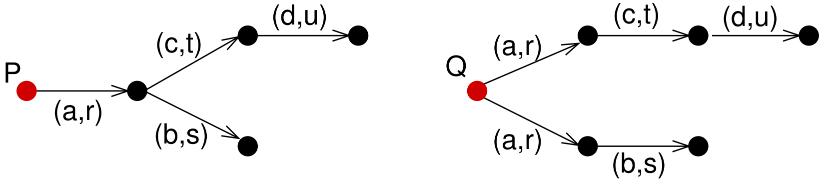
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

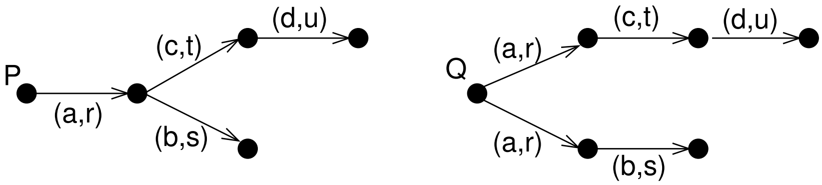
Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.

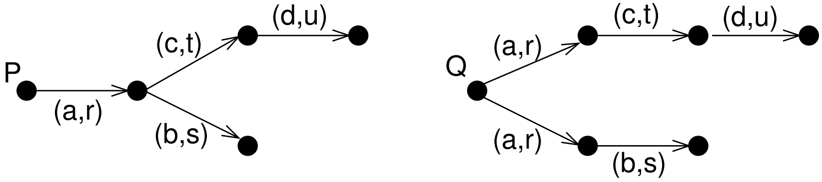


Observability is assumed to include the ability to record timing information over a number of runs.

Two processes are equivalent if they can undertake the same actions, at the same rate, and arrive at processes that are equivalent.

Strong Equivalence in PEPA

Strong equivalence in PEPA is a bisimulation in the style of Larsen of Skou.



Observability is assumed to include the ability to record timing information over a number of runs.

Two processes are equivalent if they can undertake the same actions, at the same rate, and arrive at processes that are equivalent.

Expressed as rates to equivalence classes of processes.

Strong Equivalence and Lumpability

- We can establish that if we consider strong equivalence of states within a single model, it induces a lumpable partition on the state space of the underlying Markov chain.
- Moreover it can be shown that strong equivalence is a congruence.
- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

Strong Equivalence and Lumpability

- We can establish that if we consider strong equivalence of states within a single model, it induces a lumpable partition on the state space of the underlying Markov chain.
- Moreover it can be shown that strong equivalence is a congruence.
- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

Strong Equivalence and Lumpability

- We can establish that if we consider strong equivalence of states within a single model, it induces a lumpable partition on the state space of the underlying Markov chain.
- Moreover it can be shown that strong equivalence is a congruence.
- This means that aggregation based on lumpability can be applied component by component, avoiding the previous problem of having to construct the complete state space in order to find the lumpable partitions.

Modelling at the level of individuals

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

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

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

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

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

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

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

Modelling at the level of individuals

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

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

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

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

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

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

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

Counting abstraction to generate the *Lumped* CTMC

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

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

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

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

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

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

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

Counting abstraction to generate the *Lumped* CTMC

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

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

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

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

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

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

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

Counting abstraction to generate the *Lumped* CTMC

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

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

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

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

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

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

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

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

Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states.

More recent approaches shift to a counting abstraction and a numerical representation of states and transitions.

Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states.

More recent approaches shift to a counting abstraction and a numerical representation of states and transitions.

Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states.

More recent approaches shift to a counting abstraction and a numerical representation of states and transitions.

Using this result in practice

There are well-known algorithms such as Paige and Tarjan for finding the maximal partition of a graph according to some equivalence.

However in practice we would much rather construct the aggregated state space directly.

The first approach to this used canonical forms but still worked syntactically to identify states.

More recent approaches shift to a counting abstraction and a numerical representation of states and transitions.

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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.

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

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

Aggregation is not enough!

Unfortunately, many models still suffer from state space explosion even after aggregation.

This is particularly a problem for population models — systems where we are interested in interacting populations of entities:

Crowd dynamics

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

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

A conundrum

Process algebras are well-suited to constructing such models:

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

But solution techniques which rely on explicitly building the state space, such as numerical solution, are hampered by space complexity...

...whilst those that use the implicit state space, such as simulation, run into problems of time complexity.

Population statistics: emergent behaviour

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

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Furthermore we make a continuous approximation of how the counts vary over time.

Population statistics: emergent behaviour

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

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Furthermore we make a continuous approximation of how the counts vary over time.

Population statistics: emergent behaviour

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

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Furthermore we make a continuous approximation of how the counts vary over time.

Population statistics: emergent behaviour

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

This allows us to model much larger systems than previously possible but in making the shift we are no longer able to collect any information about individuals in the system.

To characterise the behaviour of a population we count the number of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Furthermore we make a continuous approximation of how the counts vary over time.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

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

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

.

New mathematical structures: differential equations

1. Use a counting abstraction rather than the CTMC complete state space.
2. Assume that these state variables are subject to continuous rather than discrete change.
3. No longer aim to calculate the probability distribution over the entire state space of the model.
4. Instead the trajectory of the ODEs estimates the expected behaviour of the CTMC.

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.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Models suitable for counting abstraction

- In the PEPA language multiple instances of components are represented explicitly — we write $P[n]$ to denote an array of n copies of P executing in parallel.

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

- The impact of an action of a counting variable is
  - decrease by 1 if the component participates in the action
  - increase by 1 if the component is the result of the action
  - zero if the component is not involved in the action.

Simple example revisited

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

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

Simple example revisited

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

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

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

Simple example revisited

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

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

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

$$x_1 = \text{no. of } Proc_1$$

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

Simple example revisited

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

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

ODE interpretation

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_1\,x_1, r_3\,x_3) + r_2\,x_2 \\
& x_1 = \text{no. of } Proc_1 \\
\frac{dx_2}{dt} &= \min(r_1\,x_1, r_3\,x_3) - r_2\,x_2 \\
& x_2 = \text{no. of } Proc_2 \\
\frac{dx_3}{dt} &= -\min(r_1\,x_1, r_3\,x_3) + r_4\,x_4 \\
& x_3 = \text{no. of } Res_0 \\
\frac{dx_4}{dt} &= \min(r_1\,x_1, r_3\,x_3) - r_4\,x_4 \\
& x_4 = \text{no. of } Res_1
\end{aligned}
$$

Scalable Differential Semantics

Whilst fluid approximation has been used for many years in large scale performance models, e.g. fluid queues as an abstraction of routers in communication networks, in general the validity of the abstraction and convergence result must be proved on a case-by-case basis.

In his recent thesis, Mirco Tribastone developed a novel operational semantics for PEPA which can be used to prove the convergence result for all suitably scaled PEPA models.

Moreover the set of ODEs are automatically derived from the semantics.

Scalable Differential Semantics

Whilst fluid approximation has been used for many years in large scale performance models, e.g. fluid queues as an abstraction of routers in communication networks, in general the validity of the abstraction and convergence result must be proved on a case-by-case basis.

In his recent thesis, Mirco Tribastone developed a novel operational semantics for PEPA which can be used to prove the convergence result for all suitably scaled PEPA models.

Moreover the set of ODEs are automatically derived from the semantics.

Scalable Differential Semantics

Whilst fluid approximation has been used for many years in large scale performance models, e.g. fluid queues as an abstraction of routers in communication networks, in general the validity of the abstraction and convergence result must be proved on a case-by-case basis.

In his recent thesis, Mirco Tribastone developed a novel operational semantics for PEPA which can be used to prove the convergence result for all suitably scaled PEPA models.

Moreover the set of ODEs are automatically derived from the semantics.

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

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

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n.
- 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).
- Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n.
- 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).
- Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n.
- 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).
- Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

Consistency results

- The vector field $\mathcal{F}(x)$ is Lipschitz continuous i.e. all the rate functions governing transitions in the process algebra satisfy local continuity conditions.
- The generated ODEs are the fluid limit of the family of CTMCs generated by $f(\xi, l, \alpha)$: this family forms a sequence as the initial populations are scaled by a variable n.
- 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).
- Moreover Lipschitz continuity of the vector field guarantees existence and uniqueness of the solution to the initial value problem.

Outline

Spatial Challenge: capturing logical space

Whilst stochastic process algebras are well-suited to model concurrent systems, there is an implicit assumption that all components are co-located.

10-15 years ago we started modelling systems which broke this assumption and demanded more careful thought about the location of components, and how location influences the dynamic evolution of the system.

Spatial Challenge: capturing logical space

Whilst stochastic process algebras are well-suited to model concurrent systems, there is an implicit assumption that all components are co-located.

10-15 years ago we started modelling systems which broke this assumption and demanded more careful thought about the location of components, and how location influences the dynamic evolution of the system.

Spatial Challenge: capturing logical space

Whilst stochastic process algebras are well-suited to model concurrent systems, there is an implicit assumption that all components are co-located.

10-15 years ago we started modelling systems which broke this assumption and demanded more careful thought about the location of components, and how location influences the dynamic evolution of the system.

Mobile devices and mobile computation

The location of components of a software system can have dramatic effect on the performance, particularly as communication is often slow compared with computation. Thus capturing whether components are co-located or communicating over a distance became important.

Spatial Challenge: capturing logical space

Whilst stochastic process algebras are well-suited to model concurrent systems, there is an implicit assumption that all components are co-located.

10-15 years ago we started modelling systems which broke this assumption and demanded more careful thought about the location of components, and how location influences the dynamic evolution of the system.

Biochemical signalling pathways

Far from being a well-mixed soup, the inside of a cell is highly structured and divided into distinct compartments. This physical organisation can have a strong impact on the dynamic behaviour.

Mobile computation: PEPA nets

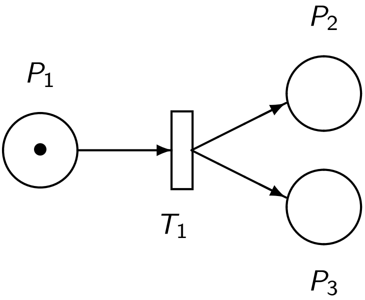
- The PEPA nets formalism uses the stochastic process algebra PEPA as the inscription language for coloured Petri nets.
- The combination naturally represents applications with two classes of change of state (global and local).
- For example, in a mobile code system PEPA terms are used to model the program code which moves between network hosts (the places in the net).

Petri nets

- Petri nets provide a graphical presentation of a model which has an easily accessible interpretation and like process algebras they are supported by an unambiguous formal interpretation.

Petri nets

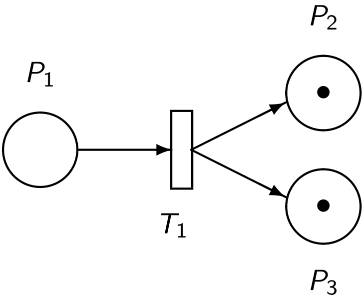
- Petri nets provide a graphical presentation of a model which has an easily accessible interpretation and like process algebras they are supported by an unambiguous formal interpretation.



When a transition fires tokens from input places are absorbed and tokens are created on each of the output places.

Petri nets

- Petri nets provide a graphical presentation of a model which has an easily accessible interpretation and like process algebras they are supported by an unambiguous formal interpretation.



When a transition fires tokens from input places are absorbed and tokens are created on each of the output places.

Petri nets

- Petri nets provide a graphical presentation of a model which has an easily accessible interpretation and like process algebras they are supported by an unambiguous formal interpretation.
- Coloured Petri nets are a high-level form of classical Petri nets. The plain (indistinguishable) tokens of a classical Petri net are replaced by arbitrary terms which are distinguishable.
- In stochastic Petri nets the transitions from one marking to another are associated with a random variable drawn from an exponential distribution.
- PEPA nets are coloured stochastic Petri nets where the colours used as the tokens of the net are PEPA components.

Global and local state changes

- Firings in a PEPA net (at the Petri net level) model macro-step changes of state such as a mobile software agent moving from one network host to another.
- A token/PEPA component will move from one place/context to another.
- Firings have global effect because they involve components at more than one place in the net.
- A transition occurs whenever an action (individual or shared) of a PEPA component can occur.
- Since only co-located components can cooperate all transitions have local effect because they involve only components at one place in the net.

Global and local state changes

- Firings in a PEPA net (at the Petri net level) model macro-step changes of state such as a mobile software agent moving from one network host to another.
- A token/PEPA component will move from one place/context to another.
- Firings have global effect because they involve components at more than one place in the net.
- A transition occurs whenever an action (individual or shared) of a PEPA component can occur.
- Since only co-located components can cooperate all transitions have local effect because they involve only components at one place in the net.

Global and local state changes

- Firings in a PEPA net (at the Petri net level) model macro-step changes of state such as a mobile software agent moving from one network host to another.
- A token/PEPA component will move from one place/context to another.
- Firings have global effect because they involve components at more than one place in the net.
- A transition occurs whenever an action (individual or shared) of a PEPA component can occur.
- Since only co-located components can cooperate all transitions have local effect because they involve only components at one place in the net.

Global and local state changes

- Firings in a PEPA net (at the Petri net level) model macro-step changes of state such as a mobile software agent moving from one network host to another.
- A token/PEPA component will move from one place/context to another.
- Firings have global effect because they involve components at more than one place in the net.
- A transition occurs whenever an action (individual or shared) of a PEPA component can occur.
- Since only co-located components can cooperate all transitions have local effect because they involve only components at one place in the net.

Global and local state changes

- Firings in a PEPA net (at the Petri net level) model macro-step changes of state such as a mobile software agent moving from one network host to another.
- A token/PEPA component will move from one place/context to another.
- Firings have global effect because they involve components at more than one place in the net.
- A transition occurs whenever an action (individual or shared) of a PEPA component can occur.
- Since only co-located components can cooperate all transitions have local effect because they involve only components at one place in the net.

Example: a mobile agent system

- A roving agent visits three sites. It interacts with static software components at these sites and has two kinds of interactions.
- When visiting a site where a network probe is present it interrogates the probe for the data which it has gathered on recent patterns of network traffic.
- When it returns to the central co-ordinating site it dumps the data which it has harvested to the master probe. The master probe performs a computationally expensive statistical analysis of the data.
- The structure of the system allows this computation to be overlapped with the agent's communication and data gathering.

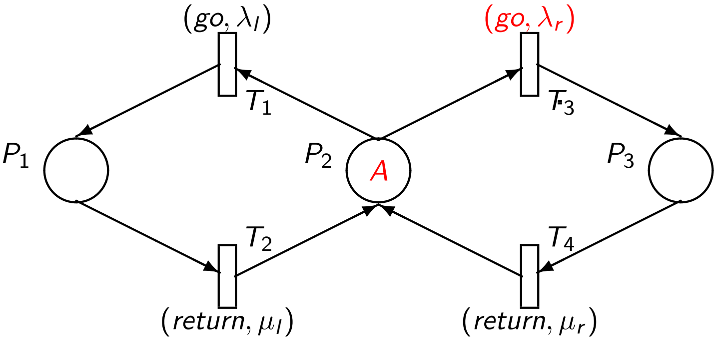
PEPA components

$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

$$
\begin{aligned}
Master &\stackrel{def}{=} (dump, \top).Master' \\
Master' &\stackrel{def}{=} (analyse, r_a).Master
\end{aligned}
$$

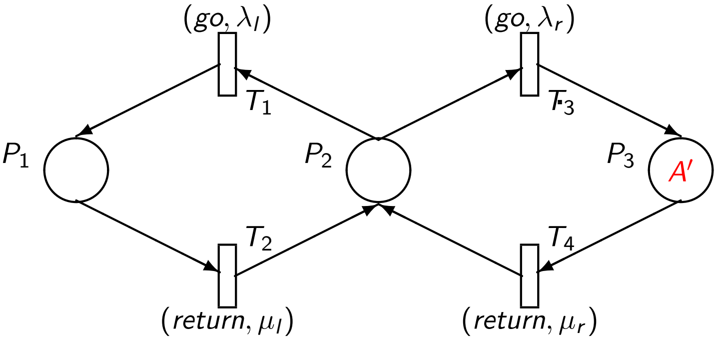
$$
\begin{aligned}
Probe &\stackrel{def}{=} (monitor, r_m).Probe + \\
&\qquad (interrogate, \top).Probe
\end{aligned}
$$

PEPA net example



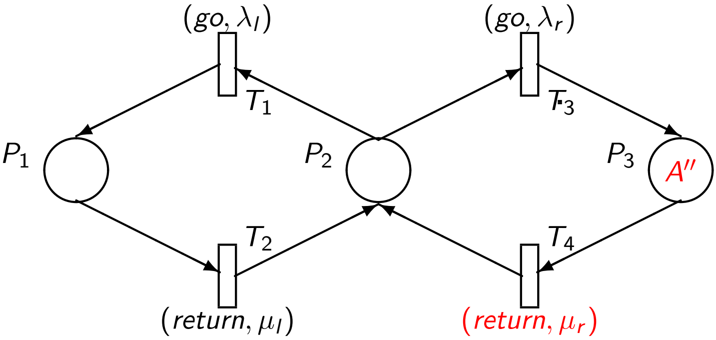
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



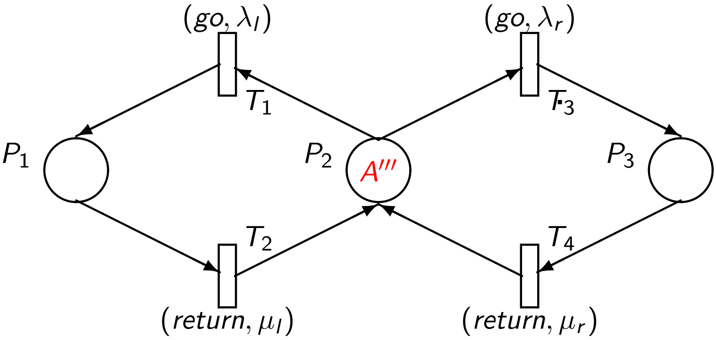
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



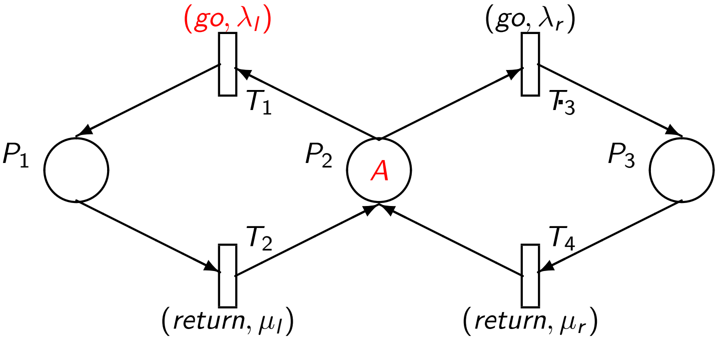
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



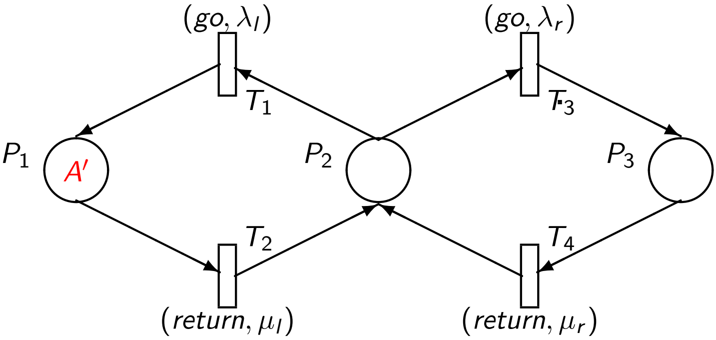
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



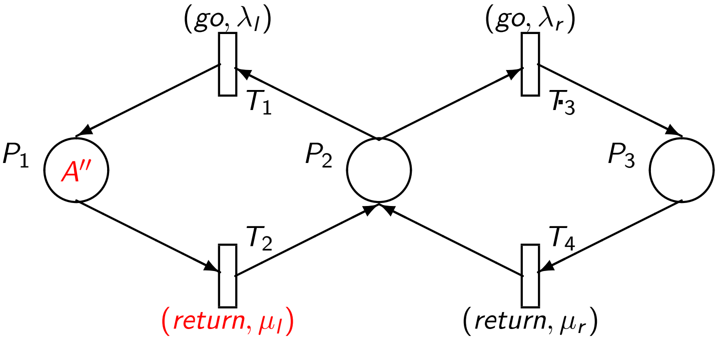
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



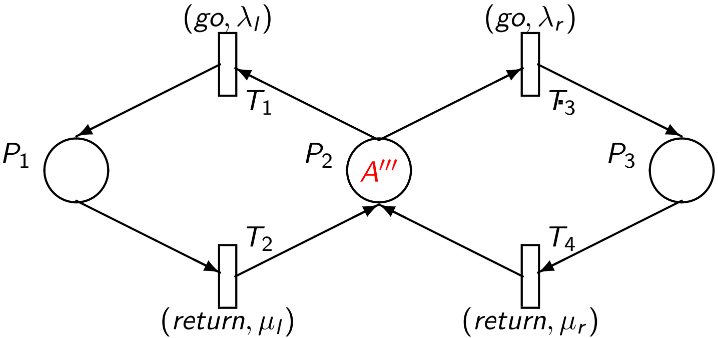
$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



$$
\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}
$$

PEPA net example



$$\begin{aligned}
Agent &\stackrel{def}{=} (go, \lambda).Agent' \\
Agent' &\stackrel{def}{=} (interrogate, r_i).Agent'' \\
Agent'' &\stackrel{def}{=} (return, \mu).Agent''' \\
Agent''' &\stackrel{def}{=} (dump, r_d).Agent
\end{aligned}$$

Bio-PEPA

Bio-PEPA is a stochastic process algebra closely related to PEPA, but specifically designed for capturing biochemical network and systems with large interacting populations.

The language contains some constructs to model locations, and particularly pathways involving multiple compartments.

Bio-PEPA

Bio-PEPA is a stochastic process algebra closely related to PEPA, but specifically designed for capturing biochemical network and systems with large interacting populations.

The language contains some constructs to model locations, and particularly pathways involving multiple compartments.

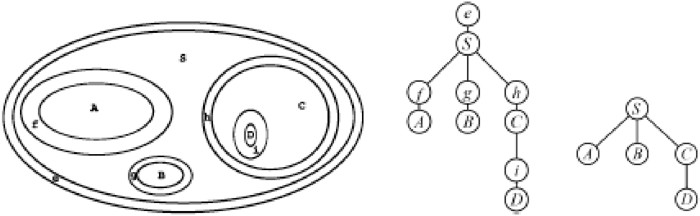
Modelling biological locations

Bio-PEPA considers locations which can be either compartments or membranes.

Reactions can then be considered to be

- internal to one compartment or membrane
- involving elements in one compartment and one membrane
- transport between compartments.

A location tree is used to represent the hierarchy of locations.



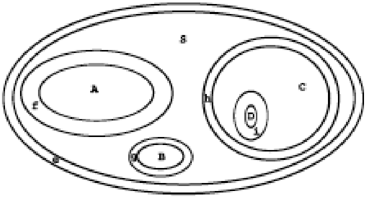
Modelling biological locations

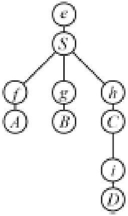
Bio-PEPA considers locations which can be either compartments or membranes.

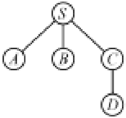
Reactions can then be considered to be

- internal to one compartment or membrane
- involving elements in one compartment and one membrane
- transport between compartments.

A location tree is used to represent the hierarchy of locations.







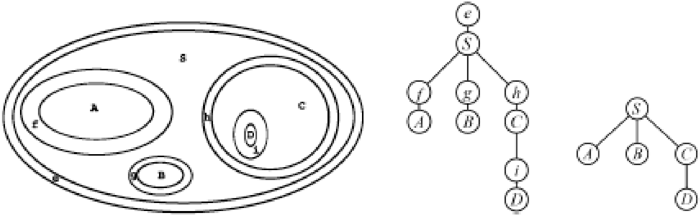
Modelling biological locations

Bio-PEPA considers locations which can be either compartments or membranes.

Reactions can then be considered to be

- internal to one compartment or membrane
- involving elements in one compartment and one membrane
- transport between compartments.

A location tree is used to represent the hierarchy of locations.



Locations in Bio-PEPA

Components in Bio-PEPA are known as species, and in essence, a species in a different location is treated as a distinct species.

However to ease the representation of models, high-level syntax allows some compact representations e.g.

$S \stackrel{def}{=} (\gamma[L_1 \rightarrow L_2], \kappa) \odot S$ for transport from location L_1 to location

$S \stackrel{def}{=} (\alpha, \kappa) op S@L_1$ for reaction α at location L_1

Analysing models with logical locations

Both PEPA Nets and Bio-PEPA allow logical locations to be captured within a process algebra model.

However, for analysis they currently rely on an expansion that treats each component, at each location, as distinct.

This exacerbates the problem of state space explosion and can limit the size of models that can be analysed.

In particular, fluid approximation techniques can only be used when the population at each location is sufficiently large to justify the continuous approximation.

Analysing models with logical locations

Both PEPA Nets and Bio-PEPA allow logical locations to be captured within a process algebra model.

However, for analysis they currently rely on an expansion that treats each component, at each location, as distinct.

This exacerbates the problem of state space explosion and can limit the size of models that can be analysed.

In particular, fluid approximation techniques can only be used when the population at each location is sufficiently large to justify the continuous approximation.

Analysing models with logical locations

Both PEPA Nets and Bio-PEPA allow logical locations to be captured within a process algebra model.

However, for analysis they currently rely on an expansion that treats each component, at each location, as distinct.

This exacerbates the problem of state space explosion and can limit the size of models that can be analysed.

In particular, fluid approximation techniques can only be used when the population at each location is sufficiently large to justify the continuous approximation.

Analysing models with logical locations

Both PEPA Nets and Bio-PEPA allow logical locations to be captured within a process algebra model.

However, for analysis they currently rely on an expansion that treats each component, at each location, as distinct.

This exacerbates the problem of state space explosion and can limit the size of models that can be analysed.

In particular, fluid approximation techniques can only be used when the population at each location is sufficiently large to justify the continuous approximation.

Moving on to physical space

As we begin to witness informatic environments as Robin Milner defined them, with computational capacity embedded in our physical environment, it is going to become increasingly important to be able to model them and predict their behaviour.

In many of these cases, logical location will not be enough and real physical location will need to be incorporated into our modelling techniques.

This poses significant challenges both of model expression and model solution.

Moving on to physical space

As we begin to witness informatic environments as Robin Milner defined them, with computational capacity embedded in our physical environment, it is going to become increasingly important to be able to model them and predict their behaviour.

In many of these cases, logical location will not be enough and real physical location will need to be incorporated into our modelling techniques.

This poses significant challenges both of model expression and model solution.

Moving on to physical space

As we begin to witness informatic environments as Robin Milner defined them, with computational capacity embedded in our physical environment, it is going to become increasingly important to be able to model them and predict their behaviour.

In many of these cases, logical location will not be enough and real physical location will need to be incorporated into our modelling techniques.

This poses significant challenges both of model expression and model solution.

QUANTICOL Examples

The most expensive aspect of the Paris bike sharing system is relocating bikes to where they are needed.

In smart grids and sustainable energy production with limited storage capacity the location of production and demand become important.

Hybrid process algebra HYPE

The hybrid process algebra HYPE captures both continuously varying values and discrete changes in behaviour.

Cartesian coordinates can be represented as continuous variables with appropriate functions to capture movement.

Hybrid process algebra HYPE

The hybrid process algebra HYPE captures both continuously varying values and discrete changes in behaviour.

Cartesian coordinates can be represented as continuous variables with appropriate functions to capture movement.

MSc student Cheng Feng used this approach in HYPE to model ZebraNET, a sensor network in which RFID tags are attached to zebras.

Unfortunately based on hybrid simulation only six zebras could be simulated on a standard machine and fluid techniques are not applicable.

Hybrid process algebra HYPE

The hybrid process algebra HYPE captures both continuously varying values and discrete changes in behaviour.

Cartesian coordinates can be represented as continuous variables with appropriate functions to capture movement.

MSc student Cheng Feng used this approach in HYPE to model ZebraNET, a sensor network in which RFID tags are attached to zebras.

Unfortunately based on hybrid simulation only six zebras could be simulated on a standard machine and fluid techniques are not applicable.

Outline

Success of stochastic process algebra

- A high level description of the system eases the task of model construction and use of a formal language allows unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- As a side effect, the performance modelling community has been introduced to (and adopted) some additional rigour.

Success of stochastic process algebra

- A high level description of the system eases the task of model construction and use of a formal language allows unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- As a side effect, the performance modelling community has been introduced to (and adopted) some additional rigour.

Success of stochastic process algebra

- A high level description of the system eases the task of model construction and use of a formal language allows unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- As a side effect, the performance modelling community has been introduced to (and adopted) some additional rigour.

Success of stochastic process algebra

- A high level description of the system eases the task of model construction and use of a formal language allows unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- As a side effect, the performance modelling community has been introduced to (and adopted) some additional rigour.

Success of stochastic process algebra

- A high level description of the system eases the task of model construction and use of a formal language allows unambiguous interpretation and automatic translation into the underlying mathematical structure.
- Moreover properties of that mathematical structure may be deduced by the construction at the process algebra level.
- Furthermore formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- As a side effect, the performance modelling community has been introduced to (and adopted) some additional rigour.

Major challenges

- Future work will also consider exploiting more of the formal structure of the process algebras to assist in the manipulation and analysis of the ODEs.
- Whilst fluid approximation has proved useful for deriving performance measures from models, its full power has yet to be realised for formal interrogation of models using model checking.
- Both logical and physical space pose significant challenges for scalable analysis techniques. Initial work is exploring the use of time scale decompositions, partial differential equations and diffusion models but much more work is needed.

Major challenges

- Future work will also consider exploiting more of the formal structure of the process algebras to assist in the manipulation and analysis of the ODEs.
- Whilst fluid approximation has proved useful for deriving performance measures from models, its full power has yet to be realised for formal interrogation of models using model checking.
- Both logical and physical space pose significant challenges for scalable analysis techniques. Initial work is exploring the use of time scale decompositions, partial differential equations and diffusion models but much more work is needed.

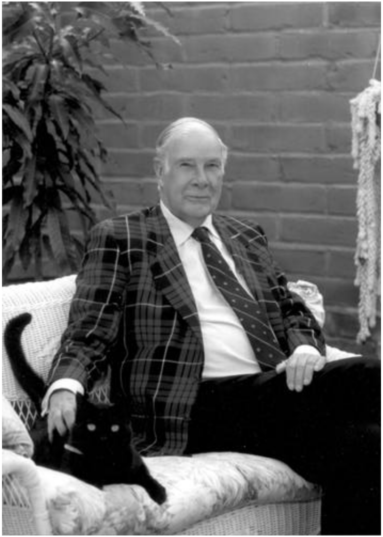
Major challenges

- Future work will also consider exploiting more of the formal structure of the process algebras to assist in the manipulation and analysis of the ODEs.
- Whilst fluid approximation has proved useful for deriving performance measures from models, its full power has yet to be realised for formal interrogation of models using model checking.
- Both logical and physical space pose significant challenges for scalable analysis techniques. Initial work is exploring the use of time scale decompositions, partial differential equations and diffusion models but much more work is needed.

Outline

Some thoughts about Hamming and his advice

Some thoughts about Hamming and his advice



Some reasonable advice....

- “Say to yourself, *“Yes, I would like to do first-class work”*.”
- “One of the characteristics of successful scientists is having courage.”
- “It’s not the consequence that makes a problem important, it is that you have a reasonable attack.”
- Don’t let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- “...it is not sufficient to do a job, you have to sell it too.”

Some reasonable advice....

- “Say to yourself, *“Yes, I would like to do first-class work”*.”
- “One of the characteristics of successful scientists is having courage.”
- “It’s not the consequence that makes a problem important, it is that you have a reasonable attack.”
- Don’t let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- “...it is not sufficient to do a job, you have to sell it too.”

Some reasonable advice....

- "Say to yourself, *"Yes, I would like to do first-class work"*."
- "One of the characteristics of successful scientists is having courage."
- "It's not the consequence that makes a problem important, it is that you have a reasonable attack."
- Don't let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- "...it is not sufficient to do a job, you have to sell it too."

Some reasonable advice....

- "Say to yourself, *"Yes, I would like to do first-class work"*."
- "One of the characteristics of successful scientists is having courage."
- "It's not the consequence that makes a problem important, it is that you have a reasonable attack."
- Don't let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- "...it is not sufficient to do a job, you have to sell it too."

Some reasonable advice....

- "Say to yourself, *"Yes, I would like to do first-class work"*."
- "One of the characteristics of successful scientists is having courage."
- "It's not the consequence that makes a problem important, it is that you have a reasonable attack."
- Don't let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- "...it is not sufficient to do a job, you have to sell it too."

Some reasonable advice....

- "Say to yourself, *"Yes, I would like to do first-class work"*."
- "One of the characteristics of successful scientists is having courage."
- "It's not the consequence that makes a problem important, it is that you have a reasonable attack."
- Don't let success make you feel you can only work on great problems.
- Look on the positive side and not on the negative side.
- "...it is not sufficient to do a job, you have to sell it too."

...but, reflecting on Hamming from an Athena perspective

He is very much a product of his generation, and says some things that I find quite uncomfortable:

He says "...most great scientists have tremendous drive", and sees it as an inevitable consequence that he sometimes neglected his wife.

Even more strongly, he says it is not sufficient to "dabble" which he interprets as "...work during the day and go home and do other things and come back and work the next day..."

Unfortunately this seems to condemn most of us with family responsibilities to being mere "dabblers"!

...but, reflecting on Hamming from an Athena perspective

He is very much a product of his generation, and says some things that I find quite uncomfortable:

He says "...most great scientists have tremendous drive", and sees it as an inevitable consequence that he sometimes neglected his wife.

Even more strongly, he says it is not sufficient to "dabble" which he interprets as "...work during the day and go home and do other things and come back and work the next day..."

Unfortunately this seems to condemn most of us with family responsibilities to being mere "dabblers"!

...but, reflecting on Hamming from an Athena perspective

He is very much a product of his generation, and says some things that I find quite uncomfortable:

He says "...most great scientists have tremendous drive", and sees it as an inevitable consequence that he sometimes neglected his wife.

Even more strongly, he says it is not sufficient to "dabble" which he interprets as "...work during the day and go home and do other things and come back and work the next day..."

Unfortunately this seems to condemn most of us with family responsibilities to being mere "dabblers"!