A mathematical approach to defining the semantics of modelling languages

Jane Hillston
LFCS, University of Edinburgh

19th November 2016

A modelling language approach to defining mathematical structures via semantics

Jane Hillston
LFCS, University of Edinburgh

19th November 2016

Outline

Outline

Quantitative Modelling

Quantitative modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

Quantitative Modelling

Quantitative modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- For example, in performance evaluation users typically want to optimise external metrics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);

Quantitative Modelling

Quantitative modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- For example, in performance evaluation users typically want to optimise external metrics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal metrics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

Quantitative Modelling

Quantitative modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

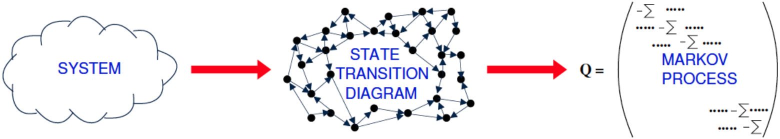
There are often conflicting interests at play:

- For example, in performance evaluation users typically want to optimise external metrics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal metrics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

Mathematical models are needed to represent and analyse the dynamic behaviour to gain understanding and make predictions.

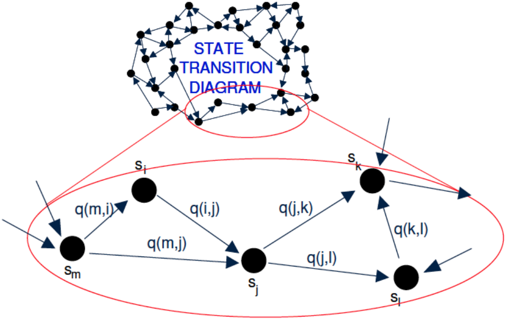
Quantitative Modelling using CTMC

Continuous Time Markov Chains are often the formalism of choice



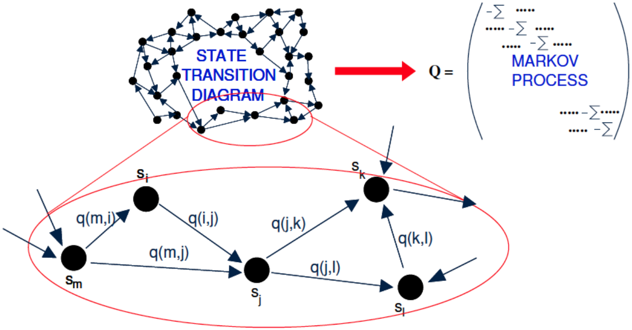
Quantitative Modelling using CTMC

Continuous Time Markov Chains are often the formalism of choice



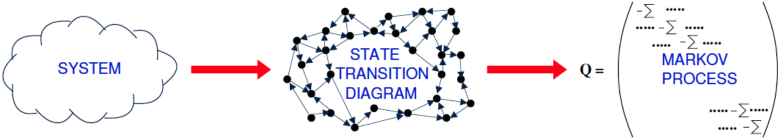
Quantitative Modelling using CTMC

Continuous Time Markov Chains are often the formalism of choice

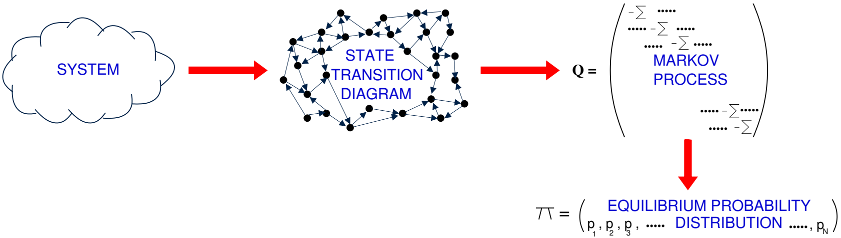


Quantitative Modelling using CTMC

Continuous Time Markov Chains are often the formalism of choice

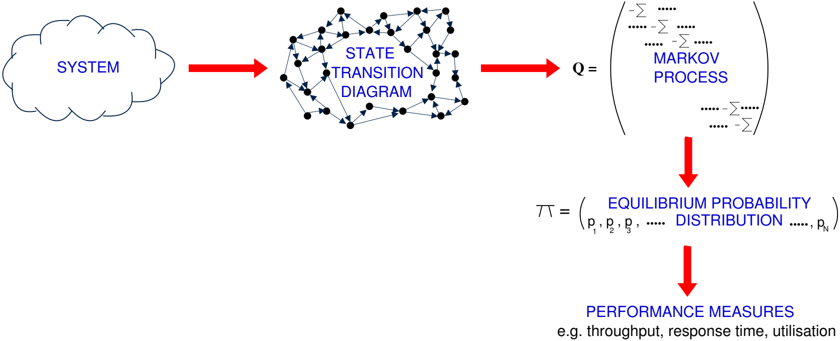


Deriving Performance Measures



Linear algebra is used to derive a transient or steady state probability distribution — the probability that the system is in each particular state at a given time.

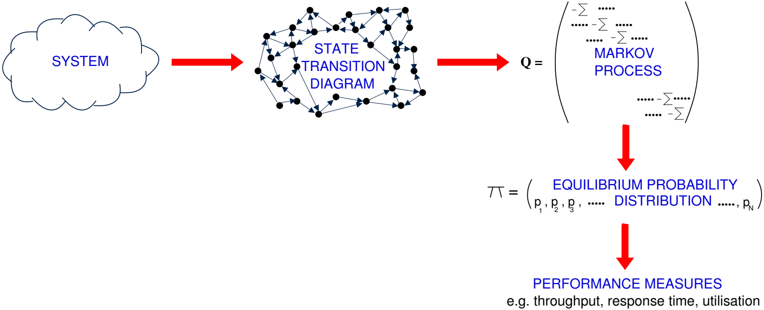
Deriving Performance Measures



Linear algebra is used to derive a transient or steady state probability distribution — the probability that the system is in each particular state at a given time.

From the probability distribution the measures such a throughput, response time and utilisation can be straightforwardly derived

Difficulties of working with Markov processes



Whilst Markov process-based modelling has many advantages, working directly in terms of the state transition diagram or infinitesimal generator matrix is at best time-consuming and error prone, and often simply infeasible.

The PEPA project

- The PEPA project started in Edinburgh in 1991.

The PEPA project

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

The PEPA project

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

The PEPA project

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

The PEPA project

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

Outline

Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

action type or name

agent/ component

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

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

Process Algebra

- Models consist of agents which engage in actions.

$$\alpha.P$$

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

- For quantitative modelling we need to incorporate quantitative information — stochastic process algebra (SPA).

SPA Languages

SPA

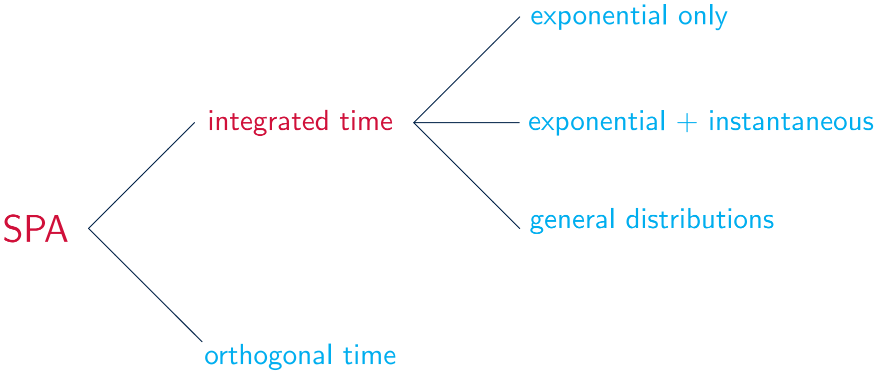
SPA Languages

integrated time

SPA

orthogonal time

SPA Languages



SPA Languages

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

SPA Languages

- SPA
  - integrated time
    - exponential only
      - PEPA, Sπ-calculus,SCCP
    - exponential $+$ instantaneous
      - EMPA, Markovian TIPP
    - general distributions
      - TIPP, SPADES, GSMPA
  - orthogonal time
    - exponential only
    - general distributions

SPA Languages

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

SPA Languages

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

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

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

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

- The language is used to generate a CTMC.

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

PEPA

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

PEPA

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

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

PEPA

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

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

PEPA

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

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

PEPA

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

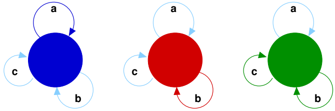
PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \stackrel{\mathit{def}}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions $\alpha \in L$ shared actions

PEPA

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

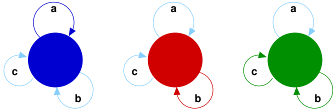
PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \stackrel{\mathit{def}}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions $\alpha \in L$ shared actions
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow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

Solving discrete state models

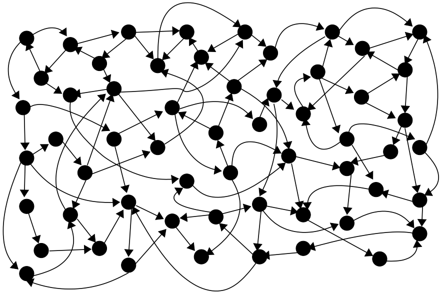


Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

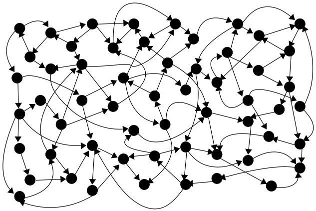
Solving discrete state models



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.

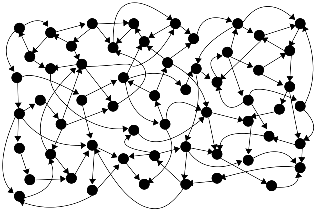


Solving discrete state models



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.

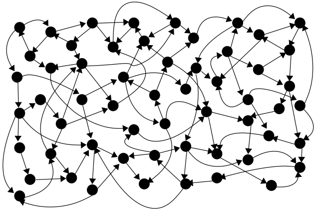
Solving discrete state models



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.

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

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.

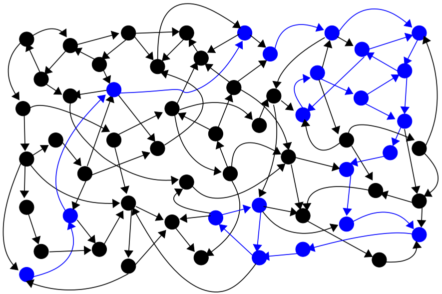
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

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

$$\pi(\infty)Q = 0$$

Solving discrete state models

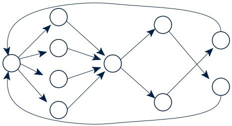
Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.



Benefits of using a language

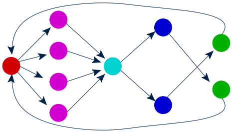
- There are clear benefits for model construction in using a modelling language and its semantics to build the required CTMC.
- But the language also allows you to characterise properties of the CTMC, previously described as properties of the infinitesimal generator matrix, as easily checked syntactic conditions.
- This supports automatic model reductions and model manipulations to improve the efficiency of solution.

Aggregation and lumpability



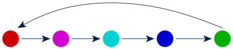
- Model aggregation: partition the state space of a model, and replace each set of states by one macro-state

Aggregation and lumpability



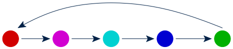
- Model aggregation: partition the state space of a model, and replace each set of states by one macro-state

Aggregation and lumpability



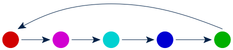
- Model aggregation: partition the state space of a model, and replace each set of states by one macro-state

Aggregation and lumpability



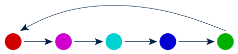
- Model aggregation: partition the state space of a model, and replace each set of states by one macro-state
- This is not as straightforward as it may seem if we wish the aggregated process to still be a Markov process — an arbitrary partition will not in general preserve the Markov property.

Aggregation and lumpability



- Model aggregation: partition 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.
- In order to preserve the Markov property we must ensure that the partition satisfies a condition called lumpability.

Aggregation and lumpability



- Model aggregation: partition 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.
- In order to preserve the Markov property we must ensure that the partition satisfies a condition called lumpability.
- Use a behavioural equivalence in the process algebra to form the partitions; moreover this is a congruence allowing the reduction to be carried out compositionally.

State space explosion

Unfortunately, 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.

Even with sophisticated model reduction and aggregation techniques discrete approaches are defeated by the scale of many dynamic systems.

Outline

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

For a large class of models, just as the size of the state space becomes unmanageable, the models become amenable to an efficient, scale-free approximation.

The Fluid Approximation Alternative

Fortunately there is an alternative: fluid approximation.

For a large class of models, just as the size of the state space becomes unmanageable, the models become amenable to an efficient, scale-free approximation.

These are models which consist of populations.

Population models

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.

Population models

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.

Population models

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 calculate the proportion of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Population models

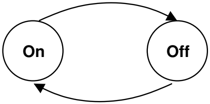
A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

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 calculate the proportion of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

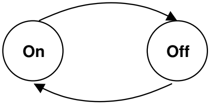
Furthermore we make a continuous approximation of how the proportions vary over time.

Population models — intuition



$Y(t)$

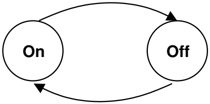
Population models — intuition

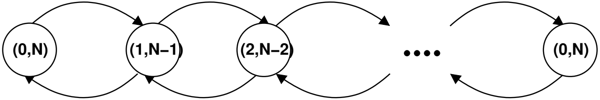


$Y(t)$

N copies: $Y_i^{(N)}$

Population models — intuition



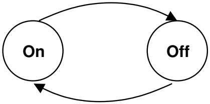


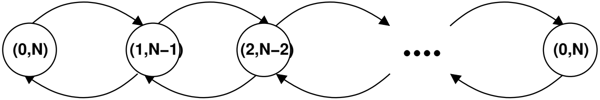
$Y(t)$

N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

Population models — intuition





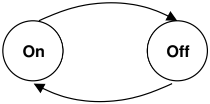
$Y(t)$

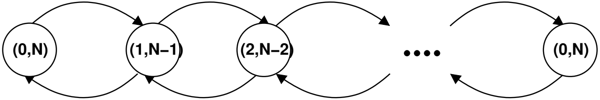
N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

Population models — intuition





$Y(t)$

N copies: $Y_i^{(N)}$

$\mathbf{X}^{(N)}(t)$

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}$$

- $Y(t)$, $Y_i^{(N)}(t)$ and $\mathbf{X}^{(N)}(t)$ are all CTMCs;
- As N increases we get a sequence of CTMCs, $\mathbf{X}^{(N)}(t)$

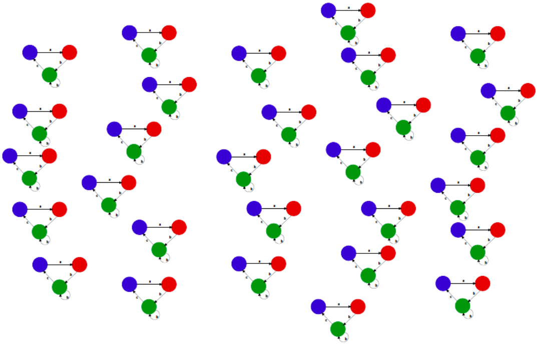
Normalised process — intuition

We consider the sequence of CTMCs, $\mathbf{X}^{(N)}(t)$ as $N \longrightarrow \infty$.

Normalised process — intuition

We consider the sequence of CTMCs, $\mathbf{X}^{(N)}(t)$ as $N \longrightarrow \infty$.

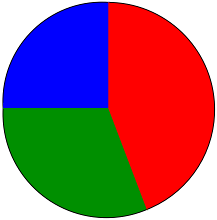
We focus on the occupancy measure — the proportion of the population that is in each possible state.



Normalised process — intuition

We consider the sequence of CTMCs, $\mathbf{X}^{(N)}(t)$ as $N \longrightarrow \infty$.

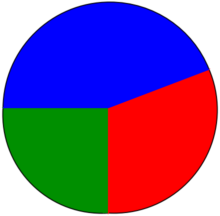
We focus on the occupancy measure — the proportion of the population that is in each possible state.



Normalised process — intuition

We consider the sequence of CTMCs, $\mathbf{X}^{(N)}(t)$ as $N \longrightarrow \infty$.

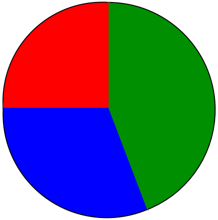
We focus on the occupancy measure — the proportion of the population that is in each possible state.



Normalised process — intuition

We consider the sequence of CTMCs, $\mathbf{X}^{(N)}(t)$ as $N \longrightarrow \infty$.

We focus on the occupancy measure — the proportion of the population that is in each possible state.



In the normalised CTMC $\hat{\mathbf{X}}^{(N)}(t)$ we are concerned with only the proportion of agents that exhibit the different possible states.

Kurtz's Deterministic Approximation Theorem

Kurtz established in the 1970s that for suitable sequences of CTMCs, in the limit, the behaviour becomes indistinguishable from a continuous evolution of the occupancy measures, governed by an appropriate set of ordinary differential equations.

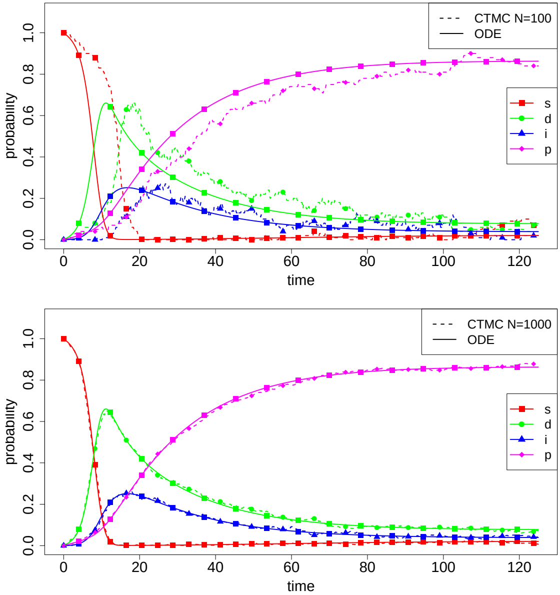
Deterministic Approximation Theorem (Kurtz)

Assume that $\exists\, \mathbf{x_0} \in S$ such that $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ in probability. Then, for any finite time horizon $T < \infty$, it holds that as $N \longrightarrow \infty$:

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

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

Illustrative trajectories



Comparison of the limit fluid ODE and a single stochastic trajectory of a network epidemic example, for total populations $N = 100$ and $N = 1000$.

Fluid semantics for Stochastic Process Algebras

- To apply these results in a stochastic process algebra we need to derive the right set of ODEs, from the model expression.

Fluid semantics for Stochastic Process Algebras

- To apply these results in a stochastic process algebra we need to derive the right set of ODEs, from the model expression.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,

Fluid semantics for Stochastic Process Algebras

- To apply these results in a stochastic process algebra we need to derive the right set of ODEs, from the model expression.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.

Fluid semantics for Stochastic Process Algebras

- To apply these results in a stochastic process algebra we need to derive the right set of ODEs, from the model expression.
- Embedding the approach in a formal language offers the possibility to establish the conditions for convergence at the language level via the semantics,
- This removes the requirement to fulfil the proof obligation on a model-by-model basis.
- Moreover the derivation of the ODEs can be automated in the implementation of the language.

Deriving a Fluid Approximation of a SPA model

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

Deriving a Fluid Approximation of a SPA model

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

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

Deriving a Fluid Approximation of a SPA model

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

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

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

Deriving a Fluid Approximation of a SPA model

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

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

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

Deriving a Fluid Approximation of a SPA model

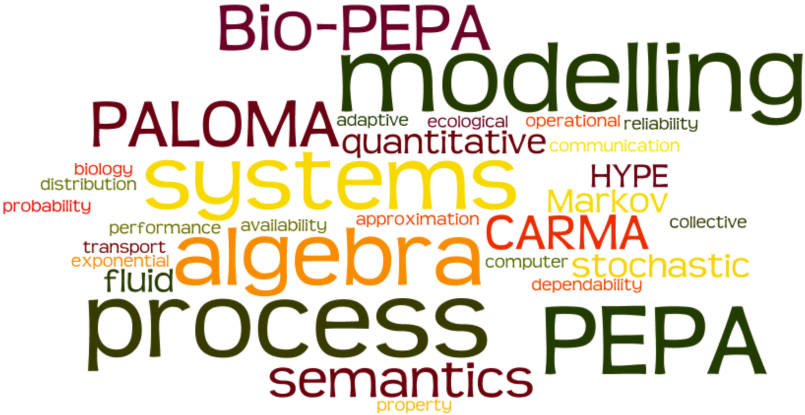
The aim is to represent the CTMC implicitly (avoiding state space explosion), and to generate the set of ODEs which are the fluid limit of that CTMC.

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

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

SPA MODEL $\xrightarrow{\text{SOS rules}}$ SYMBOLIC LABELLED TRANSITION SYSTEM $\xrightarrow[\text{functions}]{\text{generator}}$ ABSTRACT CTMC $\mathbf{Q}$ or ODEs $F_{\mathcal{M}}(x)$

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


Bio-PEPA
modelling
PALOMA
adaptive
ecological
operational
reliability
quantitative
communication
biology
distribution
systems
HYPE
probability
Markov
performance
availability
approximation
CARMA
collective
transport
exponential
algebra
computer
stochastic
fluid
dependability
process
PEPA
semantics
property

Outline

Developing a probabilistic programming approach

SPA represent systems in which there is variability in behaviour but still with the assumption that all parameters (rates) in the model are known.

Developing a probabilistic programming approach

SPA represent systems in which there is variability in behaviour but still with the assumption that all parameters (rates) in the model are known.

What if we could...

- include information about uncertainty about the model?
- automatically use observations to refine this uncertainty?
- do all this in a formal context?

Developing a probabilistic programming approach

SPA represent systems in which there is variability in behaviour but still with the assumption that all parameters (rates) in the model are known.

What if we could...

- include information about uncertainty about the model?
- automatically use observations to refine this uncertainty?
- do all this in a formal context?

Starting from an existing process algebra (Bio-PEPA), we have developed a new language ProPPA that addresses these issues

A.Georgoulas, J.Hillston, D.Milios, G.Sanguinetti: *Probabilistic Programming Process Algebra*. QEST 2014.

Probabilistic programming

A programming paradigm for describing incomplete knowledge scenarios, and resolving the uncertainty.

- Describe how the data is generated in syntax like a conventional programming language, but leaving some variables uncertain.

Probabilistic programming

A programming paradigm for describing incomplete knowledge scenarios, and resolving the uncertainty.

- Describe how the data is generated in syntax like a conventional programming language, but leaving some variables uncertain.
- Specify observations, which impose constraints on acceptable outputs of the program.

Probabilistic programming

A programming paradigm for describing incomplete knowledge scenarios, and resolving the uncertainty.

- Describe how the data is generated in syntax like a conventional programming language, but leaving some variables uncertain.
- Specify observations, which impose constraints on acceptable outputs of the program.
- Run program forwards: Generate data consistent with observations.

Probabilistic programming

A programming paradigm for describing incomplete knowledge scenarios, and resolving the uncertainty.

- Describe how the data is generated in syntax like a conventional programming language, but leaving some variables uncertain.
- Specify observations, which impose constraints on acceptable outputs of the program.
- Run program forwards: Generate data consistent with observations.
- Run program backwards: Find values for the uncertain variables which make the output match the observations.

ProPPA: Probabilistic Programming Process Algebra

The objective of ProPPA is to retain the features of the stochastic process algebra:

- simple model description in terms of components
- rigorous semantics giving an executable version of the model...

ProPPA: Probabilistic Programming Process Algebra

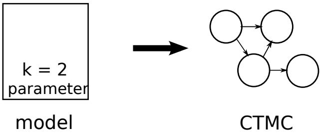
The objective of ProPPA is to retain the features of the stochastic process algebra:

- simple model description in terms of components
- rigorous semantics giving an executable version of the model...

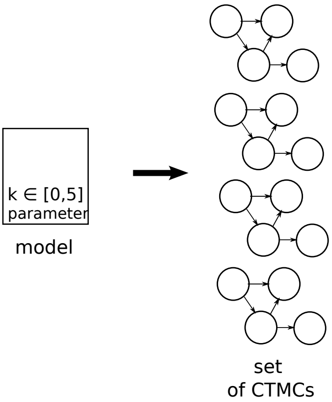
... whilst also incorporating features of a probabilistic programming language:

- recording uncertainty in the parameters
- ability to incorporate observations into models
- access to inference to update uncertainty based on observations

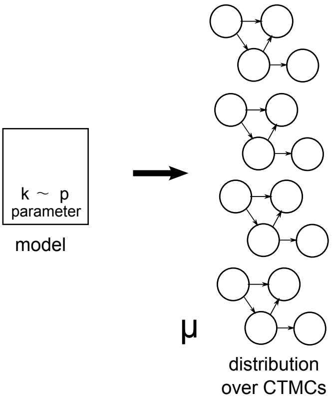
Semantics



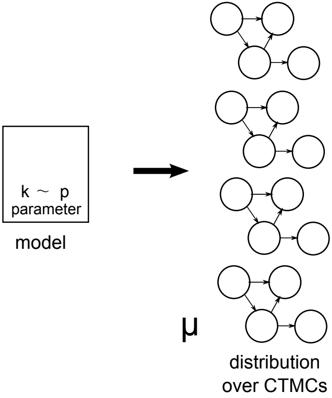
Semantics



Semantics

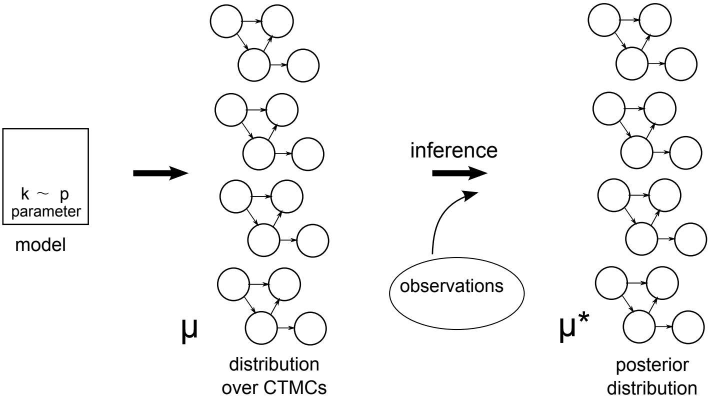


Semantics



ProPPA models are given semantics in terms of Probabilistic Constraint Markov Chains, and a variety of inference algorithms are available to refine the prior distribution into the posterior.

Semantics



ProPPA models are given semantics in terms of Probabilistic Constraint Markov Chains, and a variety of inference algorithms are available to refine the prior distribution into the posterior.

The future?

The area for quantitative analysis and verification is a good example of Strachey's ideal of theory and practice intertwined.

New applications pose new challenges for both representation and analysis and we seek to design languages to support them.

Current challenges include

- Spatially constrained behaviour
- Heterogeneous populations of agents
- Collective adaptive systems where global behaviour is defined by but also influences the behaviour of individual agents.

Thank you!