Embedding Machine Learning in Formal Stochastic Models of Biological Processes

Jane Hillston

School of Informatics, University of Edinburgh

28th November 2017

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 chains.
- 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)

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 chains.
- 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)

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 chains.
- 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)

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 chains.
- 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)

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 chains.
- 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

1. Stochastic Process Algebras
2. Modelling Biological Processes
   - Bio-PEPA
   - Approaches to system biology modelling
3. ProPPA
4. Inference
5. Results
6. Conclusions

Outline

1 Stochastic Process Algebras

2 Modelling Biological Processes
- Bio-PEPA
- Approaches to system biology modelling

3 ProPPA

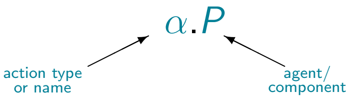
4 Inference

5 Results

6 Conclusions

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

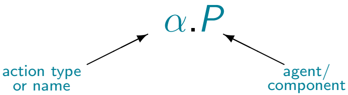
action type or name

agent/ component

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

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

agent/ component

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

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

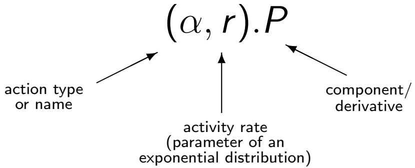
Stochastic process algebras

Stochastic process algebra

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.



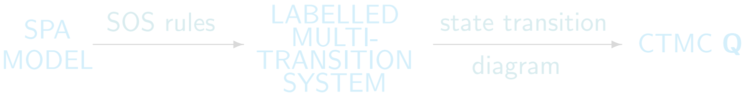
$(\alpha, r).P$

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.



SPA MODEL

SOS rules

LABELLED MULTI-TRANSITION SYSTEM

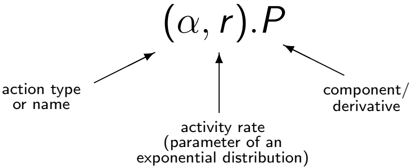
state transition diagram

CTMC **Q**

- The CTMC can be analysed numerically (linear algebra) or by stochastic simulation.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



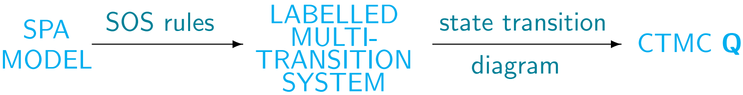
$(\alpha, r).P$

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

SOS rules

LABELLED MULTI-TRANSITION SYSTEM

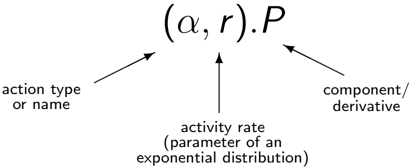
state transition diagram

CTMC **Q**

- The CTMC can be analysed numerically (linear algebra) or by stochastic simulation.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

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

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

- The CTMC can be analysed numerically (linear algebra) or by stochastic simulation.

Integrated analysis

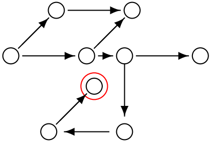
Qualitative verification can now be complemented by quantitative verification.

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?

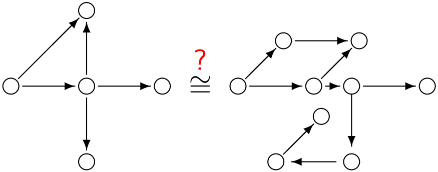


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Specification matching

With what probability does system behaviour match its specification?

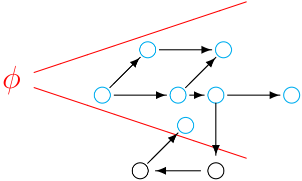


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

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

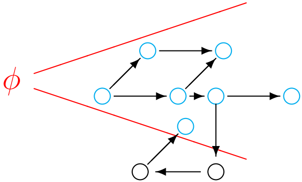


Integrated analysis

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?



Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- For example the congruence Markovian bisimulation, allows exact model reduction to be carried out compositionally.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- For example the congruence Markovian bisimulation, allows exact model reduction to be carried out compositionally.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- For example the congruence Markovian bisimulation, allows exact model reduction to be carried out compositionally.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- For example the congruence Markovian bisimulation, allows exact model reduction to be carried out compositionally.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) allows automatic evaluation of quantified properties of the behaviour of the system.

Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.
- Compositionality can be exploited both for model construction and (in some cases) for model analysis.
- Formal reasoning techniques such as equivalence relations and model checking can be used to manipulate or interrogate models.
- For example the congruence Markovian bisimulation, allows exact model reduction to be carried out compositionally.
- Stochastic model checking based on the Continuous Stochastic Logic (CSL) allows automatic evaluation of quantified properties of the behaviour of the system.

Outline

1 Stochastic Process Algebras

2 Modelling Biological Processes
- Bio-PEPA
- Approaches to system biology modelling

3 ProPPA

4 Inference

5 Results

6 Conclusions

Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems (e.g. HYPE, HyPA), spatial temporal systems (e.g. PALOMA, CARMA) and ecological processes (e.g. PALPS, MELA).

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems (e.g. HYPE, HyPA), spatial temporal systems (e.g. PALOMA, CARMA) and ecological processes (e.g. PALPS, MELA).

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

Stochastic process algebras

Over the last two decades stochastic process algebras (mostly with Markovian semantics) have been applied to a wide range of application domains.

In some case there have been new languages developed to support particular features of the application domain. These have included stochastic process algebras for modelling hybrid systems (e.g. HYPE, HyPA), spatial temporal systems (e.g. PALOMA, CARMA) and ecological processes (e.g. PALPS, MELA).

This is most noticeable in the arena of systems biology, which is often focussed on biomolecular processing systems, for example Bio-PEPA.

Molecular processes as concurrent computations

Concurrency	**Molecular Biology**	**Metabolism**	**Signal Transduction**
Concurrent computational processes	Molecules	Enzymes and metabolites	Interacting proteins
Synchronous communication	Molecular interaction	Binding and catalysis	Binding and catalysis
Transition or mobility	Biochemical modification or relocation	Metabolite synthesis	Protein binding, modification or sequestration

A. Regev and E. Shapiro *Cells as computation*, Nature 419, 2002.

The Bio-PEPA abstraction

- Each “species” (biochemical component) i is described by a species component C_i
- Each reaction j is associated with an action type α_j and its dynamics is described by a specific function f_{α_j}

The species components are then composed together to describe the behaviour of the system.

The Bio-PEPA abstraction

- Each "species" (biochemical component) i is described by a species component C_i
- Each reaction j is associated with an action type α_j and its dynamics is described by a specific function f_{α_j}

The species components are then composed together to describe the behaviour of the system.

The Bio-PEPA abstraction

- Each “species” (biochemical component) i is described by a species component C_i
- Each reaction j is associated with an action type α_j and its dynamics is described by a specific function f_{α_j}

The species components are then composed together to describe the behaviour of the system.

Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

Bio-PEPA modelling

- The state of the system at any time consists of the local states of each of its sequential/species components.
- The local states of components are quantitative rather than functional, i.e. biological changes to species are represented as distinct components.
- A component varying its state corresponds to it varying its amount.
- This is captured by an integer parameter associated with the species and the effect of a reaction is to vary that parameter by a number corresponding to the stoichiometry of this species in the reaction.

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa)\ \mathsf{op}\ S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \ \mathtt{op} \ S \mid S + S \mid C \qquad \text{where } \mathtt{op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(I)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(I)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

Each action α_j is associated with a rate f_{α_j}

The syntax

Sequential component (species component)

$$S ::= (\alpha, \kappa) \text{ op } S \mid S + S \mid C \qquad \text{where op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot$$

Model component

$$P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(l)$$

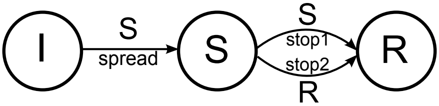
Each action α_j is associated with a rate f_{α_j}

The semantics

The semantics is defined by two transition relations:

- First, a capability relation — is a transition possible?;
- Second, a stochastic relation — gives rate of a transition, derived from the parameters of the model.

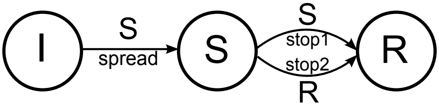
Example



```
k_s = 0.5;
k_r = 0.1;

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$I[10] \bowtie_{*} S[5] \bowtie_{*} R[0]$

Example



```
k_s = 0.5;
k_r = 0.1;

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\texttt{I[10]} \underset{*}{\bowtie} \texttt{S[5]} \underset{*}{\bowtie} \texttt{R[0]}$$

Formal modelling in systems biology

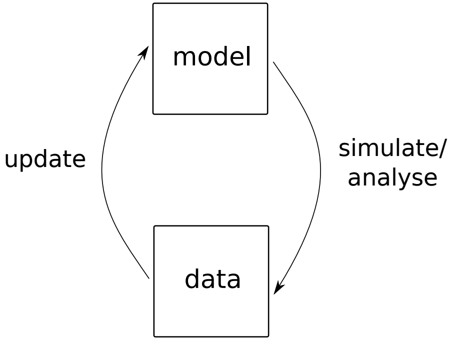
- Formal languages like Bio-PEPA provide a convenient interface for describing complex systems, reflecting what is know about the biological components and their behaviour.
- High-level abstraction eases writing and manipulating models.
- They are compiled into executable models which can be run to deepen understanding of the model.
- Formal nature lends itself to automatic, rigorous methods for analysis and verification.
- Executing the model generates data that can be compared with biological data.

- . . . but what if parts of the system are unknown?

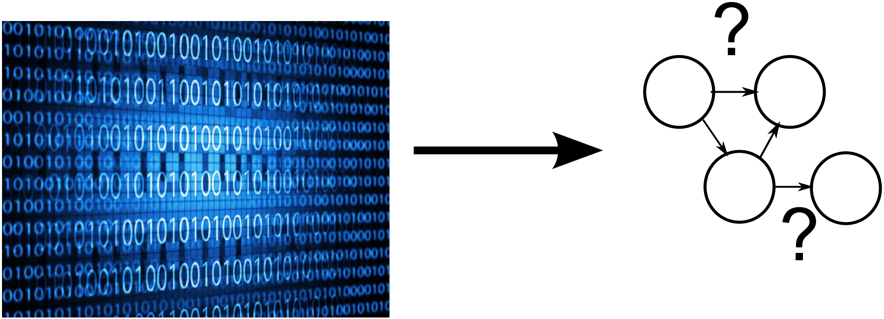
Jasmin Fisher, Thomas A. Henzinger: *Executable cell biology*. Nature Biotechnology 2007

Optimizing models

Usual process of parameterising a model is iterative and manual.

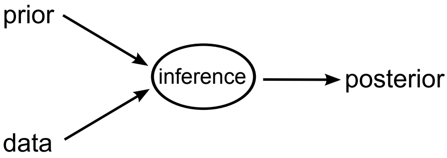


Alternative perspective



Model creation is data-driven

Machine Learning: Bayesian statistics

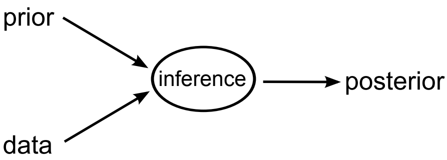


- Represent belief and uncertainty as probability distributions (prior, posterior).
- Treat parameters and unobserved variables similarly.
- Bayes' Theorem:

$$P(\theta \mid D) = \frac{P(\theta) \cdot P(D \mid \theta)}{P(D)}$$

$$\text{posterior} \propto \text{prior} \cdot \text{likelihood}$$

Machine Learning: Bayesian statistics

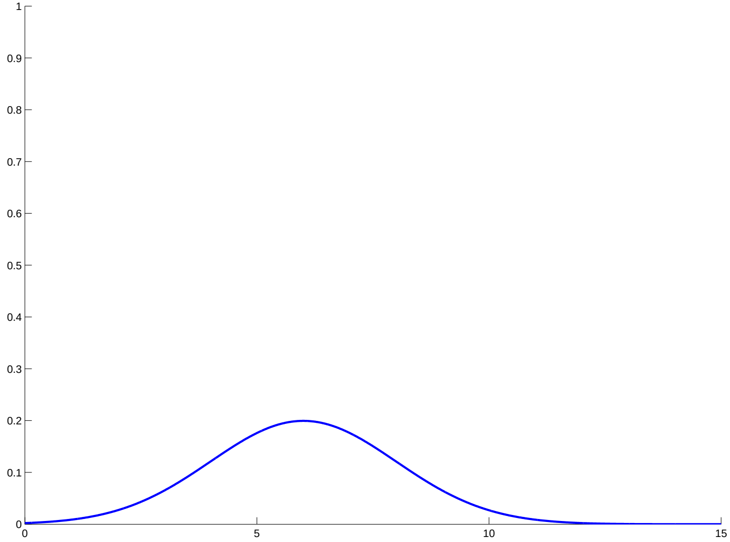


- Represent belief and uncertainty as probability distributions (prior, posterior).
- Treat parameters and unobserved variables similarly.
- Bayes' Theorem:

$$P(\theta \mid D) = \frac{P(\theta) \cdot P(D \mid \theta)}{P(D)}$$

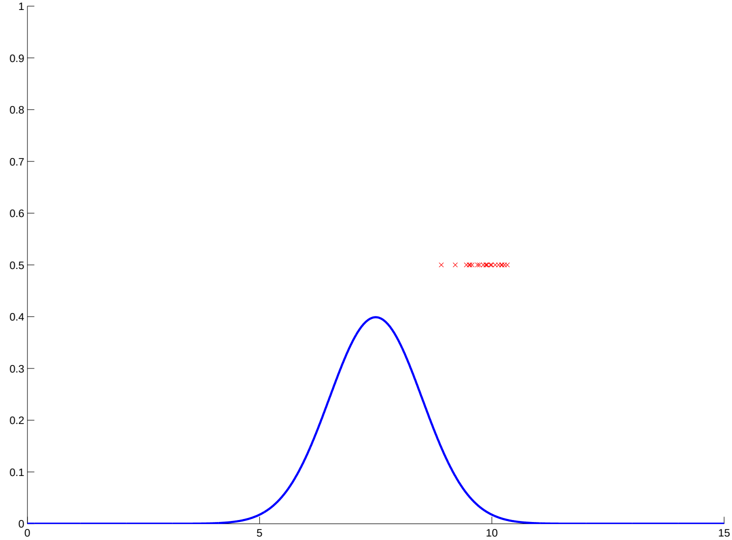
$$\text{posterior} \propto \text{prior} \cdot \text{likelihood}$$

Bayesian statistics



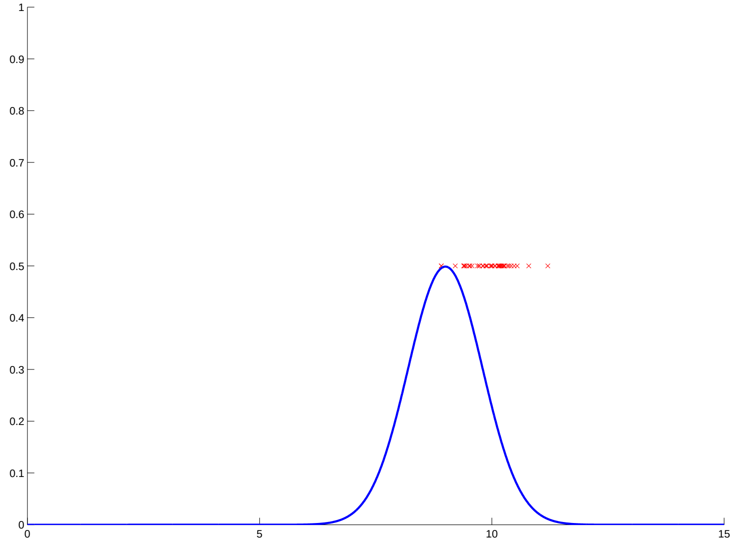
Prior distribution (no data)

Bayesian statistics



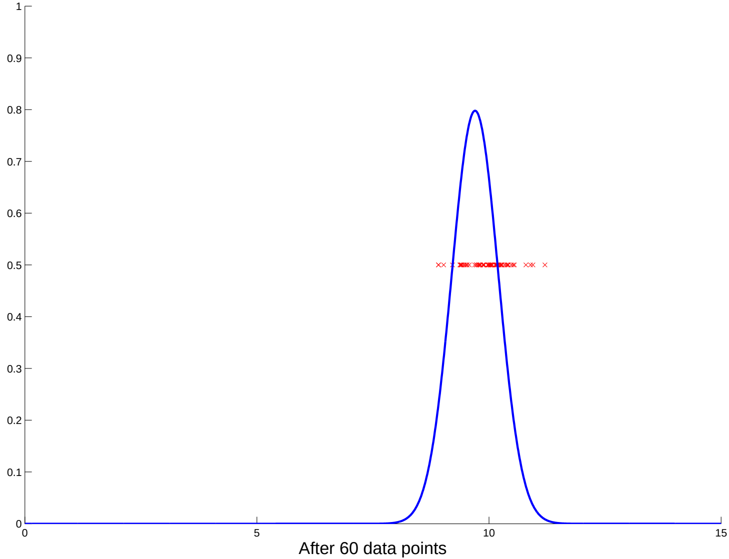
After 20 data points

Bayesian statistics



After 40 data points

Bayesian statistics



Modelling

Thus there are two approaches to model construction:

Machine Learning: extracting a model from the data generated by the system, or refining a model based on system behaviour using statistical techniques.

Mechanistic Modelling: starting from a description or hypothesis, construct a formal model that algorithmically mimics the behaviour of the system, validated against data.

Comparing the techniques

Data-driven modelling:

+ rigorous handling of parameter uncertainty
- limited or no treatment of stochasticity
- in many cases bespoke solutions are required which can limit the size of system which can be handled

Comparing the techniques

Data-driven modelling:

+ rigorous handling of parameter uncertainty
- limited or no treatment of stochasticity
- in many cases bespoke solutions are required which can limit the size of system which can be handled

Mechanistic modelling:

+ general execution ”engine” (deterministic or stochastic) can be reused for many models
+ models can be used speculatively to investigate roles of parameters, or alternative hypotheses
- parameters are assumed to be known and fixed, or costly approaches must be used to seek appropriate parameterisation

Developing a probabilistic programming approach

What if we could...

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

Starting from the 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.

Outline

1 Stochastic Process Algebras

2 Modelling Biological Processes
- Bio-PEPA
- Approaches to system biology modelling

3 ProPPA

4 Inference

5 Results

6 Conclusions

Probabilistic programming

- A programming paradigm for describing incomplete knowledge scenarios, and resolving the uncertainty.
- Programs probabilistic models in a high level language, like software code.
- Offers automated inference without the need to write bespoke solutions.
- Platforms: IBAL, Church, Infer.NET, Fun, Anglican, WebPPL,....
- Key actions: specify a distribution, specify observations, infer posterior distribution.

Probabilistic programming workflow

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

Probabilistic programming workflow

- 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 workflow

- 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 workflow

- 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.

A Probabilistic Programming Process Algebra: ProPPA

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...

A Probabilistic Programming Process Algebra: ProPPA

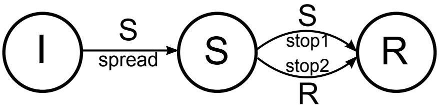
The objective of ProPPA is to retain the features of the stochastic process algebra:

- 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

Example Revisited



```
k_s = 0.5;
k_r = 0.1;

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\texttt{I[10]} \bowtie_{*} \texttt{S[5]} \bowtie_{*} \texttt{R[0]}$$

Additions

Declaring uncertain parameters:

- `k_s = Uniform(0,1);`
- `k_t = Uniform(0,1);`

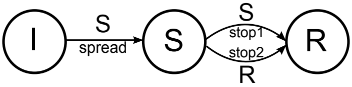
Providing observations:

- `observe('trace')`

Specifying inference approach:

- `infer('ABC')`

Additions



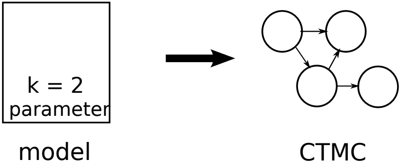
```
k_s = Uniform(0,1);
k_r = Uniform(0,1);

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\mathtt{I[10]} \underset{*}{\bowtie} \mathtt{S[5]} \underset{*}{\bowtie} \mathtt{R[0]}$$

```
observe('trace')
infer('ABC') //Approximate Bayesian Computation
```

Semantics

- A Bio-PEPA model can be interpreted as a CTMC; however, CTMCs cannot capture uncertainty in the rates (every transition must have a concrete rate).
- ProPPA models include uncertainty in the parameters, which translates into uncertainty in the transition rates.
- A ProPPA model should be mapped to something like a distribution over CTMCs.

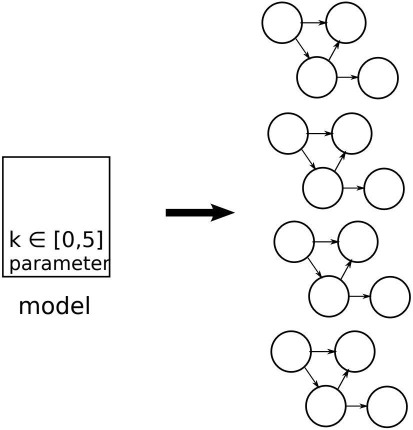


k = 2

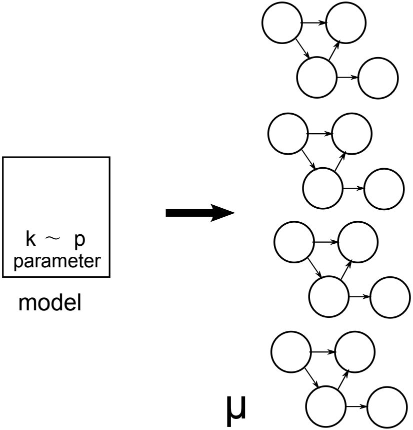
parameter

model

CTMC



set
of CTMCs



k ∼ p
parameter

model

μ

distribution
over CTMCs

Constraint Markov Chains

Constraint Markov Chains (CMCs) are a generalization of DTMCs, in which the transition probabilities are not concrete, but can take any value satisfying some constraints.

Constraint Markov Chain

A CMC is a tuple $\langle S, o, A, V, \phi \rangle$, where:

- S is the set of states, of cardinality k.
- $o \in S$ is the initial state.
- A is a set of atomic propositions.
- $V : S \to 2^{2^A}$ gives a set of acceptable labellings for each state.
- $\phi : S \times [0,1]^k \to \{0,1\}$ is the constraint function.

Caillaud *et al.*, *Constraint Markov Chains*, Theoretical Computer Science, 2011

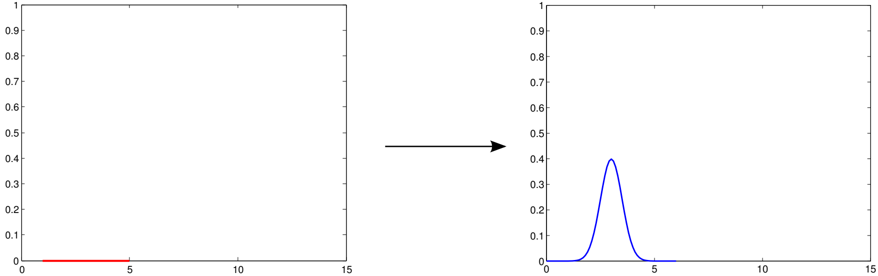
Constraint Markov Chains

In a CMC, arbitrary constraints are permitted, expressed through the function ϕ: $\phi(s, \vec{p}) = 1$ iff $\vec{p}$ is an acceptable vector of transition probabilities from state s.

However,

- CMCs are defined only for the discrete-time case, and
- this does not say anything about how likely a value is to be chosen, only about whether it is acceptable.

To address these shortcomings, we define **Probabilistic Constraint Markov Chains**.


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15

Probabilistic CMCs

A Probabilistic Constraint Markov Chain is a tuple $\langle S, o, A, V, \phi \rangle$, where:

- S is the set of states, of cardinality k.
- $o \in S$ is the initial state.
- A is a set of atomic propositions.
- $V : S \to 2^{2^A}$ gives a set of acceptable labellings for each state.
- $\phi : S \times [0, \infty)^k \to [0, \infty)$ is the constraint function.

- This is applicable to continuous-time systems.
- $\phi(s, \cdot)$ is now a probability density function on the transition rates from state s.

Semantics of ProPPA

The semantics definition follows that of Bio-PEPA, which is defined using two transition relations:

- Capability relation — is a transition possible?
- Stochastic relation — gives distribution of the rate of a transition

The distribution over the parameter values induces a distribution over transition rates.

Rules are expressed as state-to-function transition systems (FuTS[1]).

[1] De Nicola *et al.*, *A Uniform Definition of Stochastic Process Calculi*, ACM Computing Surveys, 2013

Semantics of ProPPA

The semantics definition follows that of Bio-PEPA, which is defined using two transition relations:

- Capability relation — is a transition possible?
- Stochastic relation — gives distribution of the rate of a transition

The distribution over the parameter values induces a distribution over transition rates.

Rules are expressed as state-to-function transition systems (FuTS[1]).

This gives rise the underlying PCMC.

[1] De Nicola *et al.*, *A Uniform Definition of Stochastic Process Calculi*, ACM Computing Surveys, 2013

Simulating Probabilistic Constraint Markov Chains

Probabilistic Constraint Markov Chains are open to two alternative dynamic interpretations:

1. Uncertain Markov Chains: For each trajectory, for each uncertain transition rate, sample once at the start of the run and use that value throughout;
2. Imprecise Markov Chains: During each trajectory, each time a transition with an uncertain rate is encountered, sample a value but then discard it and re-sample whenever this transition is visited again.

Simulating Probabilistic Constraint Markov Chains

Probabilistic Constraint Markov Chains are open to two alternative dynamic interpretations:

1. Uncertain Markov Chains: For each trajectory, for each uncertain transition rate, sample once at the start of the run and use that value throughout;
2. Imprecise Markov Chains: During each trajectory, each time a transition with an uncertain rate is encountered, sample a value but then discard it and re-sample whenever this transition is visited again.

Our current work is focused on the Uncertain Markov Chain case.

Outline

1 Stochastic Process Algebras

2 Modelling Biological Processes
- Bio-PEPA
- Approaches to system biology modelling

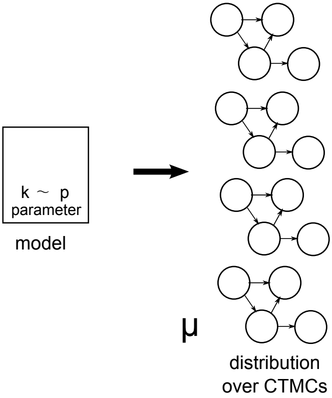
3 ProPPA

4 Inference

5 Results

6 Conclusions

Inference



k ∼ p

parameter

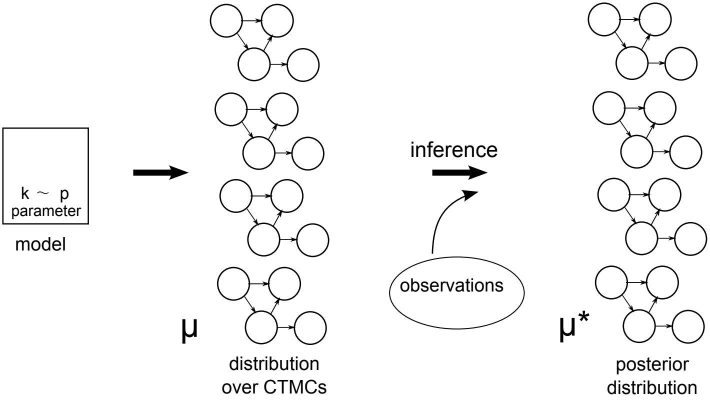
model

μ

distribution

over CTMCs

Inference


k ∼ p
parameter
model
μ
distribution
over CTMCs
inference
observations
μ*
posterior
distribution

Inference

$$P(\theta \mid D) \propto P(\theta)P(D \mid \theta)$$

- Exact inference is impossible, as we cannot calculate the likelihood.
- We must use approximate algorithms or approximations of the system.
- The ProPPA semantics does not define a single inference algorithm, allowing for a modular approach.
- Different algorithms can act on different input (time-series *vs* properties), return different results or in different forms.

Inferring likelihood in uncertain CTMCs

Transient state probabilities can be expressed as:

$$\frac{dp_i(t)}{dt} = \sum_{j \neq i} p_j(t) \cdot q_{ji} - p_i(t) \sum_{j \neq i} q_{ij}$$

Inferring likelihood in uncertain CTMCs

Transient state probabilities can be expressed as:

$$\frac{dp_i(t)}{dt} = \sum_{j \neq i} p_j(t) \cdot q_{ji} - p_i(t) \sum_{j \neq i} q_{ij}$$

The probability of a single observation (y, t) can then be expressed as

$$p(y, t) = \sum_{i \in \mathcal{S}} p_i(t) \pi(y \mid i)$$

where $\pi(y \mid i)$ is the probability of observing y when in state i.

Inferring likelihood in uncertain CTMCs

Transient state probabilities can be expressed as:

$$\frac{dp_i(t)}{dt} = \sum_{j \neq i} p_j(t) \cdot q_{ji} - p_i(t) \sum_{j \neq i} q_{ij}$$

The probability of a single observation (y, t) can then be expressed as

$$p(y, t) = \sum_{i \in \mathcal{S}} p_i(t) \pi(y \mid i)$$

where $\pi(y \mid i)$ is the probability of observing y when in state i.

The likelihood can then be expressed as

$$P(D \mid \theta) = \prod_{j=1}^{N} \sum_{i \in \mathcal{S}} p_{(i|\theta)}(t_j) \pi(y_j \mid i)$$

Calculating the transient probabilities

For finite state-spaces, the transient probabilities can, in principle, be computed as

$$\mathbf{p}(t) = \mathbf{p}(0)e^{Qt}.$$

Likelihood is hard to compute:

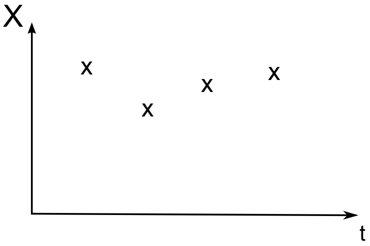
- Computing $e^{\mathbf{Q}t}$ is expensive if the state space is large
- Impossible directly in infinite state-spaces

Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is approximated with a notion of distance.

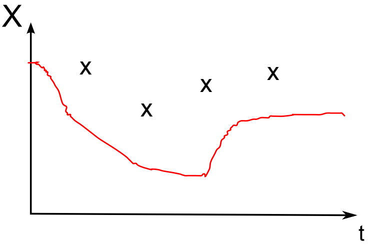
Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is approximated with a notion of distance.



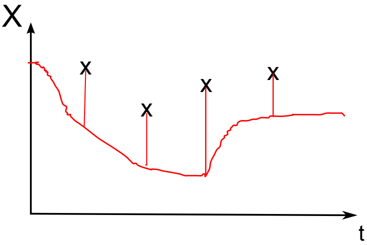
Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is approximated with a notion of distance.



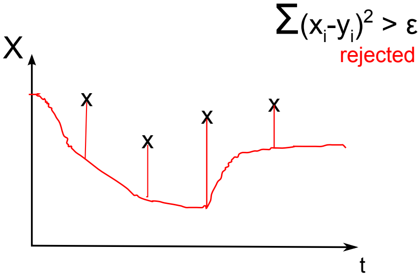
Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
    - Approximates posterior distribution over parameters as a set of samples
    - Likelihood of parameters is approximated with a notion of distance.



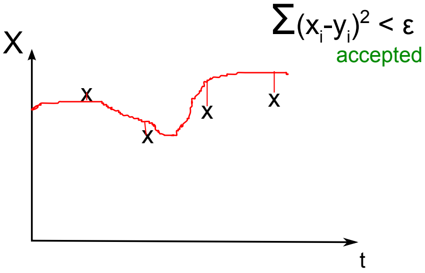
Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
    - Approximates posterior distribution over parameters as a set of samples
    - Likelihood of parameters is approximated with a notion of distance.



Basic Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is approximated with a notion of distance.



Approximate Bayesian Computation

ABC algorithm

1. Sample a parameter set from the prior distribution.
2. Simulate the system using these parameters.
3. Compare the simulation trace obtained with the observations.
4. If distance $< \epsilon$, accept, otherwise reject.

This results in an approximation to the posterior distribution.
As $\epsilon \to 0$, set of samples converges to true posterior.
We use a more elaborate version based on Markov Chain Monte Carlo sampling.

Inference for infinite state spaces

Various methods become inefficient or inapplicable as the state-space grows.

How to deal with unbounded systems?

- Multiple simulation runs
- Large population approximations (diffusion, Linear Noise,. . .)
- Systematic truncation
- **Random truncations**

Expanding the likelihood

The likelihood can be written as an infinite series:

$$p(x', t' \mid x, t) = \sum_{N=0}^{\infty} p^{(N)}(x', t' \mid x, t)$$
$$= \sum_{N=0}^{\infty} \left[f^{(N)}(x', t' \mid x, t) - f^{(N-1)}(x', t' \mid x, t) \right]$$

where

- $x^* = \max\{x, x'\}$
- $p^{(N)}(x', t' \mid x, t)$ is the probability of going from state x at time t to state x' at time t' through a path with maximum state $x^* + N$
- $f^{(N)}$ is the same, except the maximum state cannot exceed $x^* + N$ (but does not have to reach it)

Expanding the likelihood

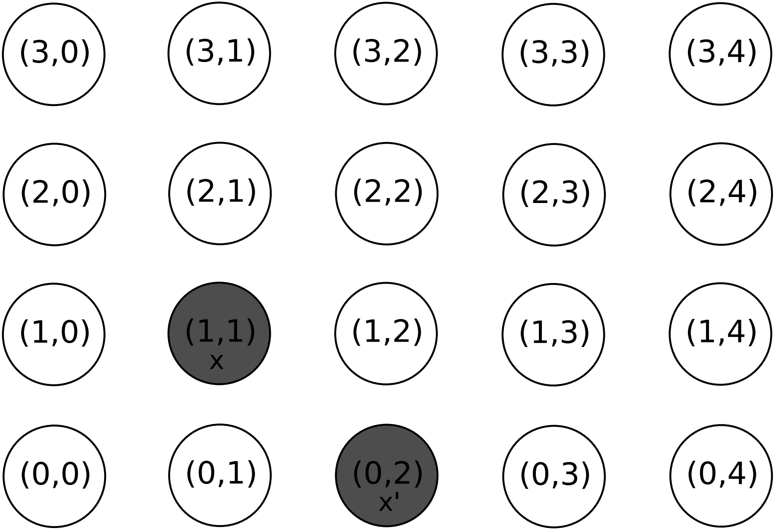
The likelihood can be written as an infinite series:

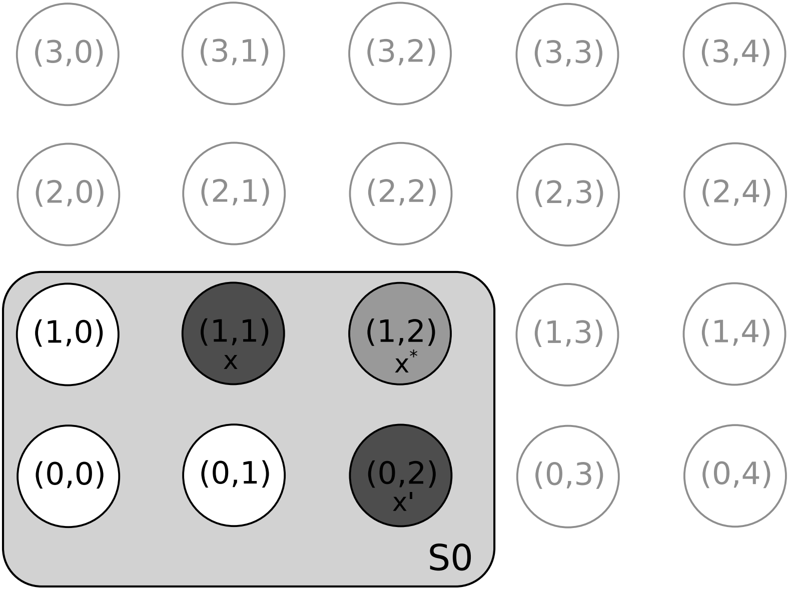
$$p(x', t' \mid x, t) = \sum_{N=0}^{\infty} p^{(N)}(x', t' \mid x, t)$$
$$= \sum_{N=0}^{\infty} \left[f^{(N)}(x', t' \mid x, t) - f^{(N-1)}(x', t' \mid x, t) \right]$$

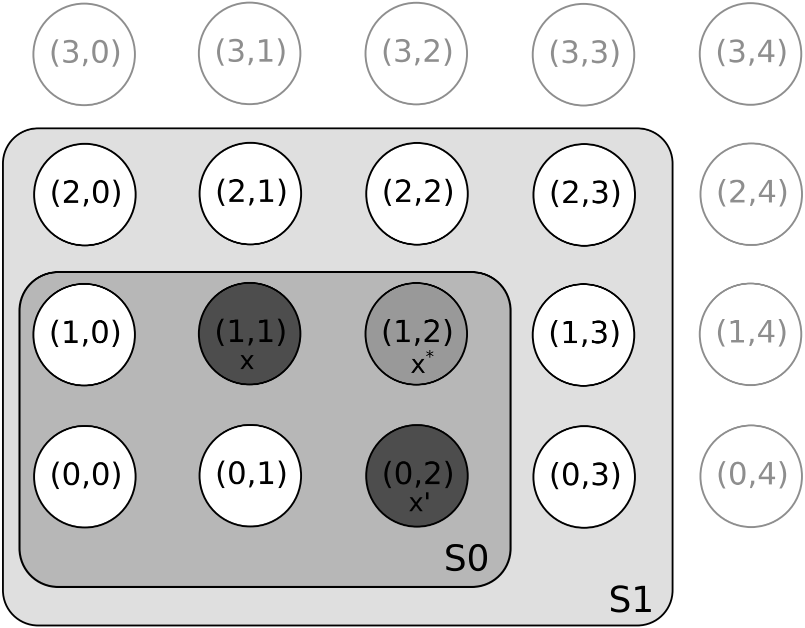
where

- $x^* = \max\{x, x'\}$
- $p^{(N)}(x', t' \mid x, t)$ is the probability of going from state x at time t to state x' at time t' through a path with maximum state $x^* + N$
- $f^{(N)}$ is the same, except the maximum state cannot exceed $x^* + N$ (but does not have to reach it)

Any finite number of terms can be computed — Can the infinite sum be computed or estimated?


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(1,0)
(1,1)
x
(1,2)
(1,3)
(1,4)
(0,0)
(0,1)
(0,2)
x'
(0,3)
(0,4)


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(1,0)
(1,1)
x
(1,2)
x*
(1,3)
(1,4)
(0,0)
(0,1)
(0,2)
x'
(0,3)
(0,4)
S0


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(1,0)
(1,1)
x
(1,2)
x*
(1,3)
(1,4)
(0,0)
(0,1)
(0,2)
x'
(0,3)
(0,4)
S0
S1

Russian Roulette Truncation

- We want to estimate the value of

$$f = \sum_{n=0}^{\infty} f_n$$

where the f_n's are computable.

Russian Roulette Truncation

- We want to estimate the value of

$$f = \sum_{n=0}^{\infty} f_n$$

  where the f_n's are computable.
- Choose a single term f_k with probability p_k; estimate $\hat{f} = \frac{f_k}{p_k}$
- $\hat{f}$ is unbiased... but its variance can be high.

Russian Roulette Truncation

- We want to estimate the value of

$$f = \sum_{n=0}^{\infty} f_n$$

  where the f_n's are computable.
- Truncate the sum randomly: stop at term k with probability q_k.
- Form $\hat{f}$ as a partial sum of the $f_n, n = 1, \ldots, k$, rescaled appropriately.

$$\hat{f} = \sum_{n=0}^{k} \frac{f_n}{\prod_{j=0}^{k-1} (1 - q_j)}$$

Russian Roulette

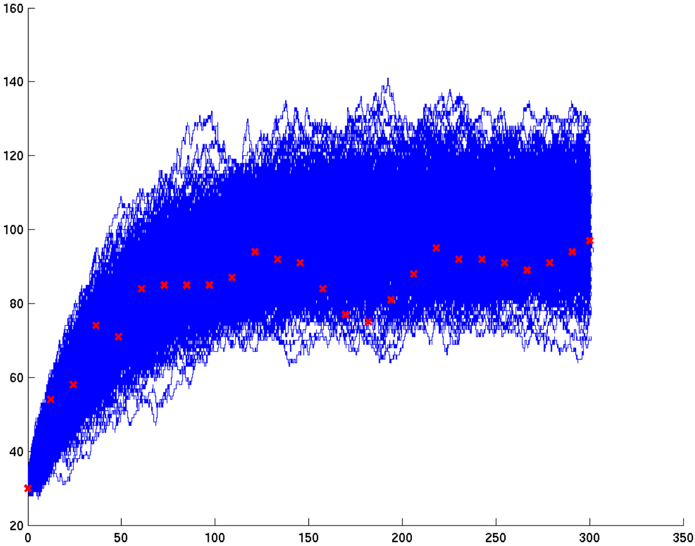
```
$\hat{f} \leftarrow f_0$
$i \leftarrow 1$
$p \leftarrow 1$
loop
    Choose to stop with probability $q_i$
    if stopping then
        return $\hat{f}$
    else
        $p \leftarrow p \cdot (1 - q_i)$
        $\hat{f} \leftarrow \hat{f} + \frac{f_i}{p}$
        $i \leftarrow i + 1$
    end if
end loop
```

Russian Roulette

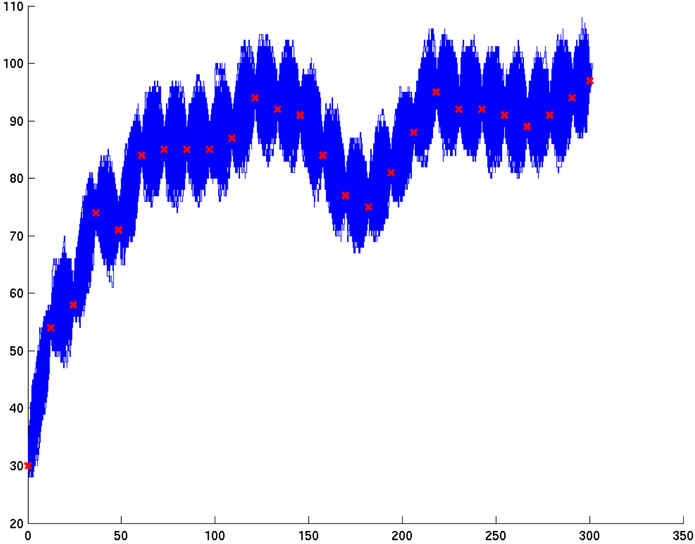
$$\hat{f} = \sum_{n=0}^{k} \frac{f_n}{\prod_{j=0}^{k-1}(1-q_j)}$$

- In our case, f is a probability that we wish to approximate.
- Using $\hat{f}$ instead of f leads to an error; however $\hat{f}$ is unbiased: $E[\hat{f}] = f$.
- $\hat{f}$ is also guaranteed to be positive.
- Pseudo-marginal algorithms can use this and still draw samples from the correct distribution.
- We have developed both Metropolis-Hastings and Gibbs-like sampling algorithms based on this approach[2].

[2] *Unbiased Bayesian Inference for Population Markov Jump Processes via Random Truncations.* A.Georgoulas, J.Hillston and D.Sanguinetti, in Stats & Comp. 2017



SSA samples



Posterior samples

Outline

1 Stochastic Process Algebras

2 Modelling Biological Processes
- Bio-PEPA
- Approaches to system biology modelling

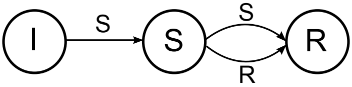
3 ProPPA

4 Inference

5 Results

6 Conclusions

Example model



```
k_s = Uniform(0,1);
k_r = Uniform(0,1);

kineticLawOf spread : k_s * I * S;
kineticLawOf stop1 : k_r * S * S;
kineticLawOf stop2 : k_r * S * R;

I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

$$\mathtt{I[10]} \underset{*}{\bowtie} \mathtt{S[5]} \underset{*}{\bowtie} \mathtt{R[0]}$$

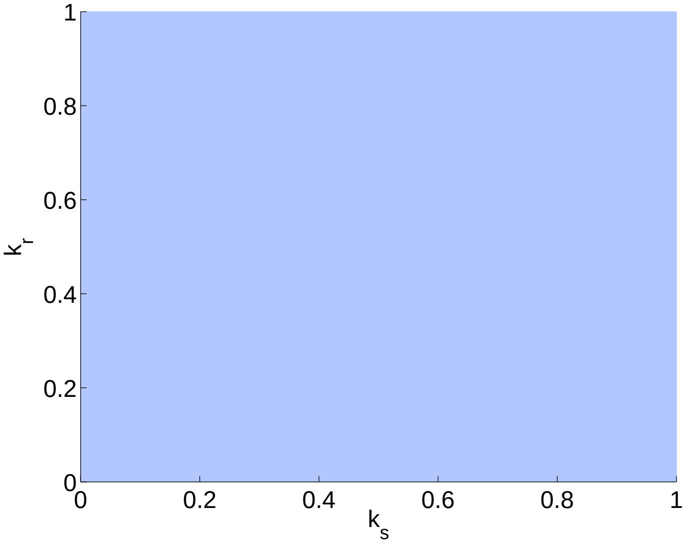
```
observe('trace')
infer('ABC') //Approximate Bayesian Computation
```

Results

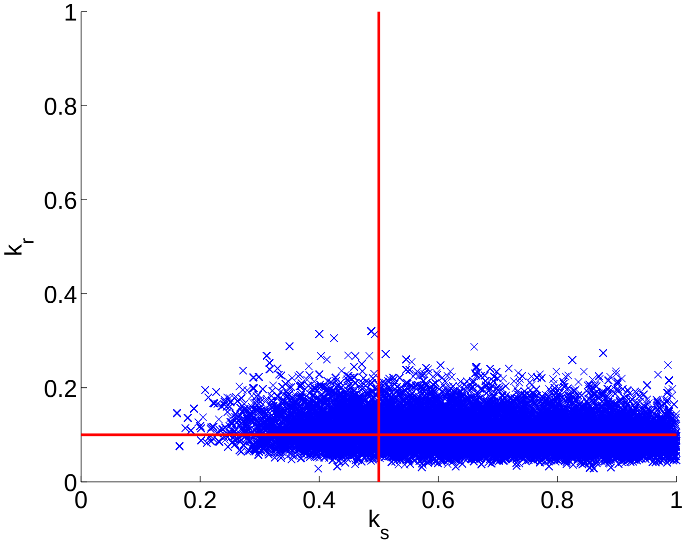
Tested on the rumour-spreading example, giving the two parameters uniform priors.

- Approximate Bayesian Computation
- Returns posterior as a set of points (samples)
- Observations: time-series (single simulation)

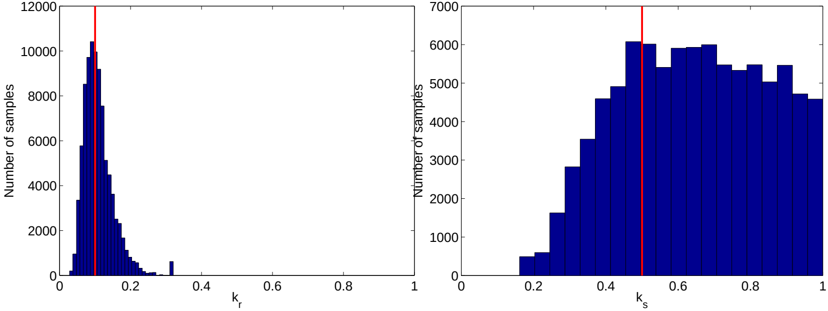
Results: ABC



Results: ABC



Results: ABC


12000
10000
8000
6000
4000
2000
0
Number of samples
0
0.2
0.4
0.6
0.8
1
k_r
7000
6000
5000
4000
3000
2000
1000
0
Number of samples
0
0.2
0.4
0.6
0.8
1
k_s

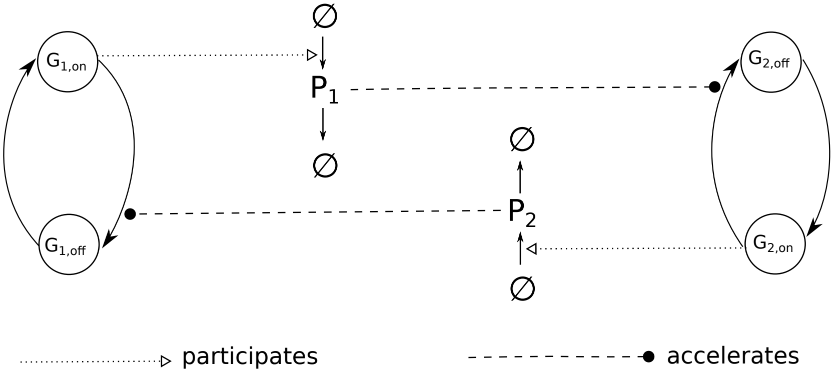
Genetic Toggle Switch

- Two mutually-repressing genes: promoters (unobserved) and their protein products
- Bistable behaviour: switching induced by environmental changes
- Synthesised in *E. coli*[3]
- Stochastic variant[4] where switching is induced by noise

[3] Gardner, Cantor & Collins, *Construction of a genetic toggle switch in Escherichia coli*, Nature, 2000

[4] Tian & Burrage, *Stochastic models for regulatory networks of the genetic toggle switch*, PNAS, 2006

Genetic Toggle Switch



Toggle switch model: species

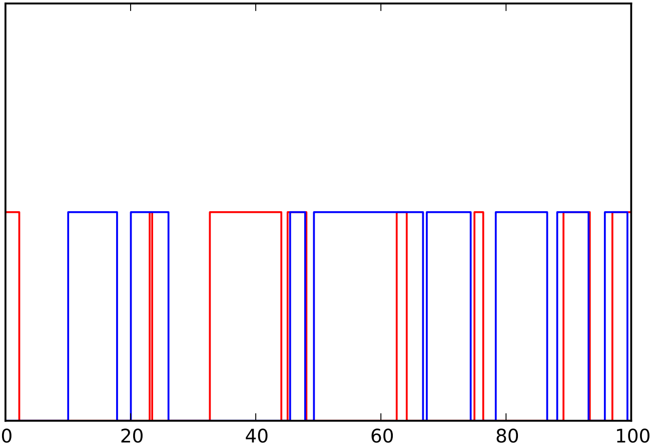
```
G1 = activ1 ↑ + deact1 ↓ + expr1 ⊕;
G2 = activ2 ↑ + deact2 ↓ + expr2 ⊕;

P1 = expr1 ↑ + degr1 ↓ + deact2 ⊕ ;
P2 = expr2 ↑ + degr2 ↓ + deact1 ⊕

G1[1] <*> G2[0] <*> P1[20] <*> P2[0]

observe(toggle_obs);
infer(rouletteGibbs);
```

```
θ_1 = Gamma(3,5); //etc...

kineticLawOf expr1 : θ_1 * G1;
kineticLawOf expr2 : θ_2 * G2;
kineticLawOf degr1 : θ_3 * P1;
kineticLawOf degr2 : θ_4 * P2;

kineticLawOf activ1 : θ_5 * (1 - G1);
kineticLawOf activ2 : θ_6 * (1 - G2);
kineticLawOf deact1 : θ_7 * exp(r ∗ P2) * G1;
kineticLawOf deact2 : θ_8 * exp(r ∗ P1) * G2;

G1 = activ1 ↑ + deact1 ↓ + expr1 ⊕;
G2 = activ2 ↑ + deact2 ↓ + expr2 ⊕;
P1 = expr1 ↑ + degr1 ↓ + deact2 ⊕ ;
P2 = expr2 ↑ + degr2 ↓ + deact1 ⊕

G1[1] <*> G2[0] <*> P1[20] <*> P2[0]

observe(toggle_obs);
infer(rouletteGibbs);
```

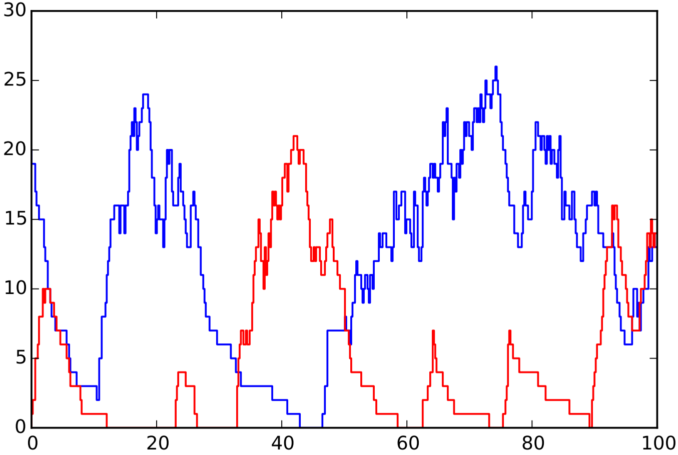
Experiment

- Simulated observations
- Gamma priors on all parameters (required by algorithm)
- Goal: learn posterior of 8 parameters
- 5000 samples taken using the Gibbs-like random truncation algorithm

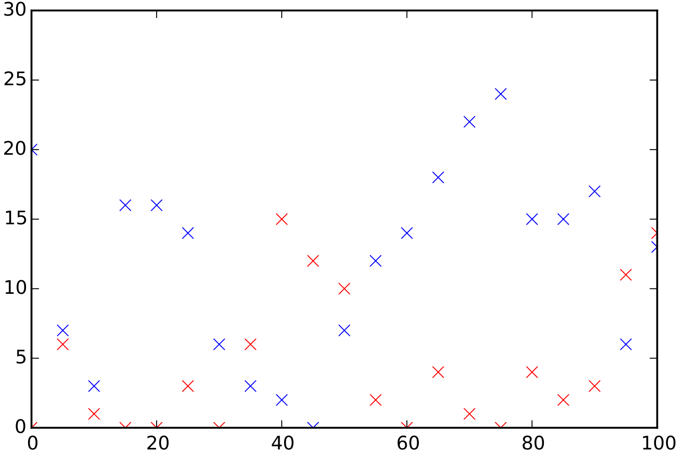
Genes



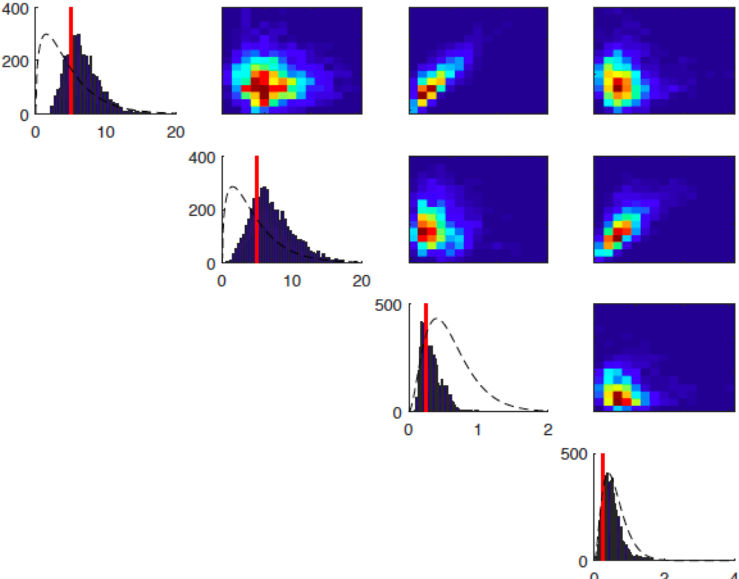
Proteins



Observations used



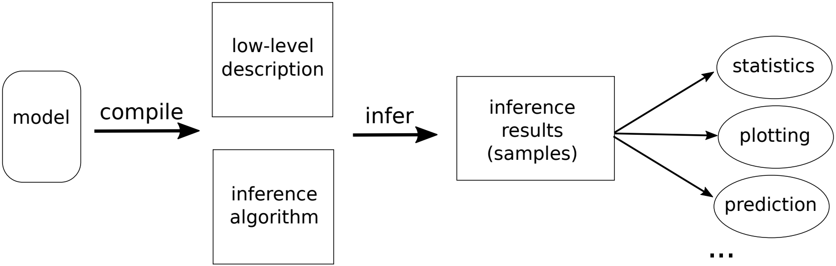
Results



Outline

1. Stochastic Process Algebras
2. Modelling Biological Processes
   - Bio-PEPA
   - Approaches to system biology modelling
3. ProPPA
4. Inference
5. Results
6. Conclusions

Workflow

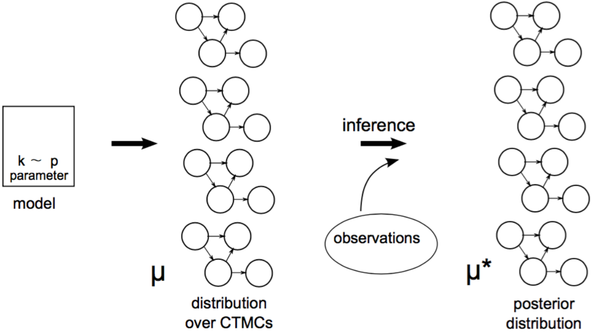

model
compile
low-level description
inference algorithm
infer
inference results (samples)
statistics
plotting
prediction
...

Summary

- ProPPA is a process algebra that incorporates uncertainty and observations directly in the model, influenced by probabilistic programming.
- Syntax remains similar to Bio-PEPA.
- Semantics defined in terms of an extension of Constraint Markov Chains.
- Observations can be either time-series or logical properties.
- Parameter inference based on random truncations (Russian Roulette) offers new possibilities for inference.

Challenges and Future Directions

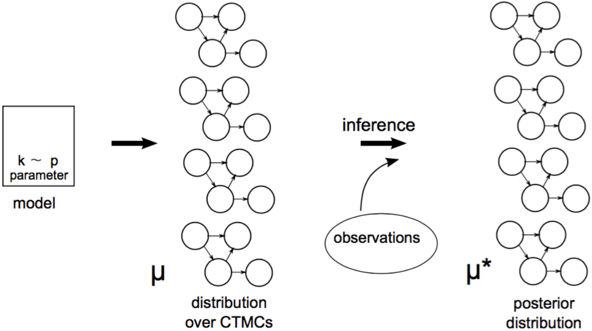
- The value of observations



Can we reason about the “distance” between μ and μ^*?

Challenges and Future Directions

- Heterogeneous populations



What if we are seeking the "optimal mix" rather than the best individual representative?

Thanks

Bio-PEPA

- Federica Ciocchetta

ProPPA

- Anastasis Georgoulas

- Guido Sanguinetti

Thanks

Bio-PEPA

- Federica Ciocchetta

ProPPA

- Anastasis Georgoulas

- Guido Sanguinetti

Funding from EPSRC, ERC, Microsoft Research and CEC FET-Proactive Programme FoCAS.