

Embedding Machine Learning in Stochastic Process Algebra

Jane Hillston

Joint work with Anastasis Georgoulas and Guido Sanguinetti,
School of Informatics, University of Edinburgh

16th August 2017



European Research Council

Microsoft
Research

quanticol

Outline

- 1 Introduction
- 2 Probabilistic Programming
- 3 ProPPA
- 4 Inference
- 5 Results
- 6 Conclusions

Outline

1 Introduction

2 Probabilistic Programming

3 ProPPA

4 Inference

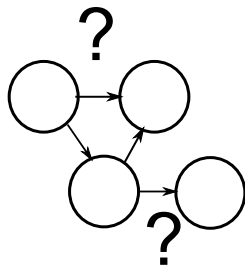
5 Results

6 Conclusions

Formal modelling

- Formal languages (process algebras, Petri nets, rule-based) provide a convenient interface for describing complex systems.
- High-level abstraction makes writing and manipulating models easier.
- They can capture different kinds of behaviour: deterministic, stochastic, ...
- Formal nature lends itself to automatic, rigorous methods for analysis and verification.
- ... but what if parts of the system are unknown?

Alternative perspective



Model creation is data-driven

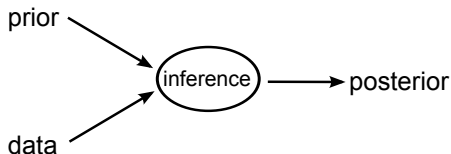
Modelling

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 model that algorithmically mimics the behaviour of the system, validated against data.

Machine Learning: Bayesian statistics

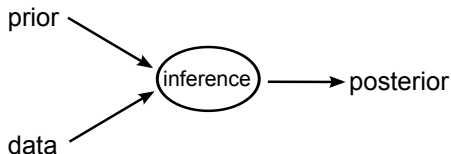


- Represent belief and uncertainty as probability distributions (prior, posterior).
- Treat parameters and unobserved variables similarly.
- Bayes' Theorem:

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

posterior \propto prior \cdot likelihood

Machine Learning: Bayesian statistics

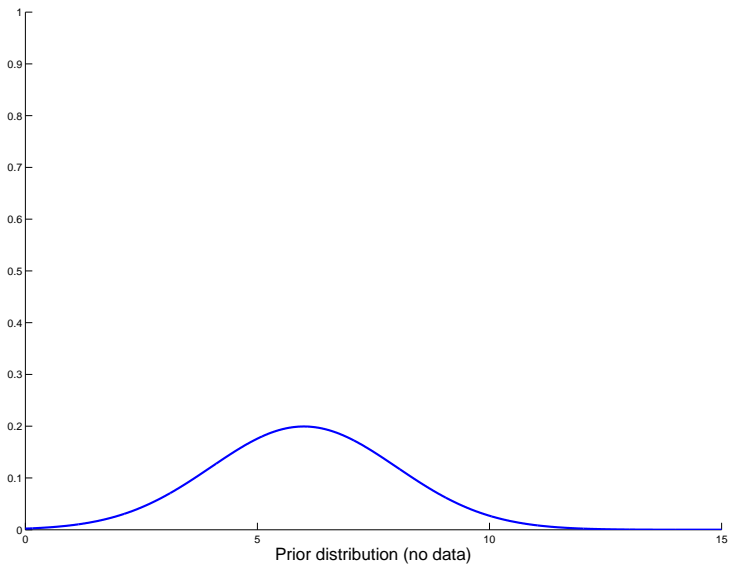


- Represent belief and uncertainty as probability distributions (prior, posterior).
- Treat parameters and unobserved variables similarly.
- Bayes' Theorem:

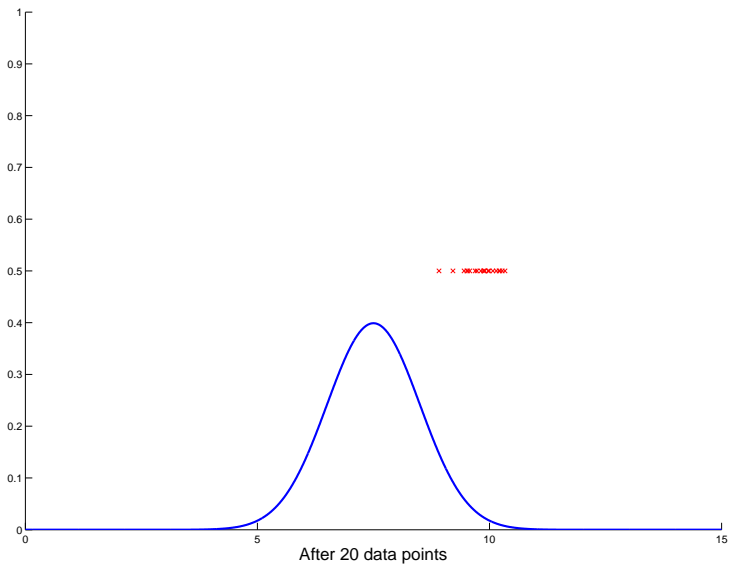
$$P(\theta \mid D) = \frac{P(\theta) \cdot P(D \mid \theta)}{P(D)}$$

posterior \propto prior \cdot likelihood

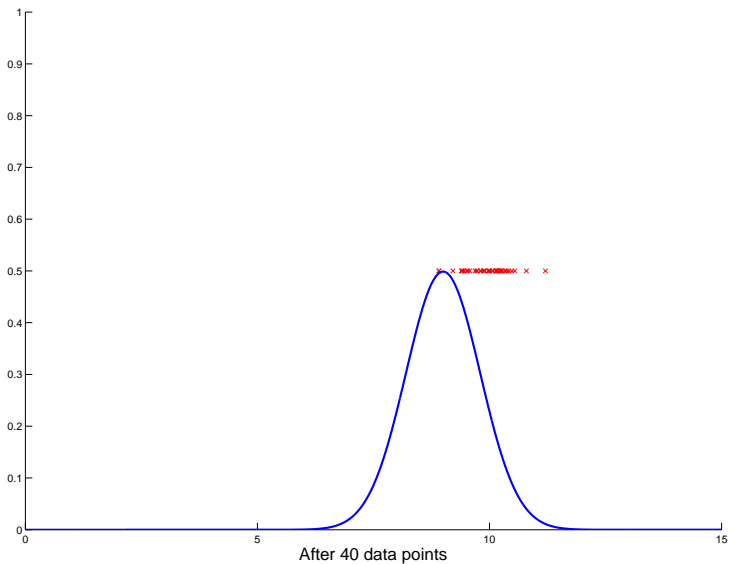
Bayesian statistics



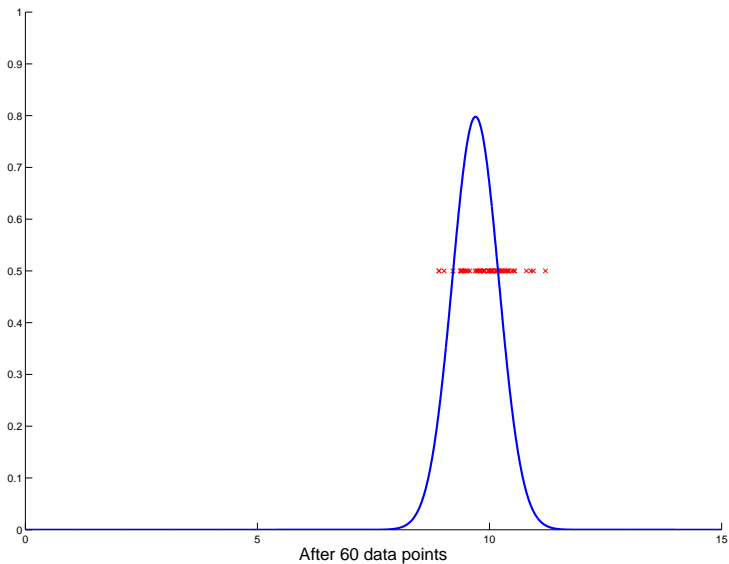
Bayesian statistics



Bayesian statistics



Bayesian statistics



Mechanistic modelling

Models are constructed reflecting what is known about the components of the biological system and their behaviour.

Mechanistic modelling

Models are constructed reflecting what is known about the components of the biological system and their behaviour.

Several approaches originating in theoretical computer science have been proposed to capture the system behaviour in a high-level way.

Mechanistic modelling

Models are constructed reflecting what is known about the components of the biological system and their behaviour.

Several approaches originating in theoretical computer science have been proposed to capture the system behaviour in a high-level way.

These are then **compiled** into **executable models** which can be run to deepen understanding of the model.

Mechanistic modelling

Models are constructed reflecting what is known about the components of the biological system and their behaviour.

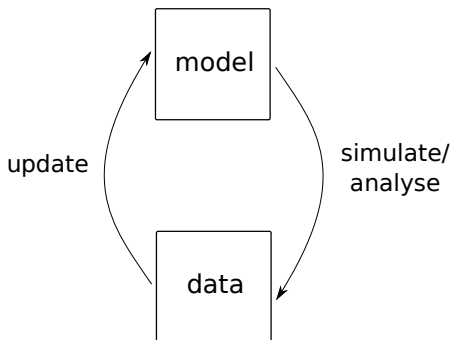
Several approaches originating in theoretical computer science have been proposed to capture the system behaviour in a high-level way.

These are then **compiled** into **executable models** which can be run to deepen understanding of the model.

Executing the model generates data that can be compared with biological data.

Optimizing models

Usual process of parameterising a model is iterative and manual.



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.

Outline

- 1 Introduction
- 2 Probabilistic Programming
- 3 ProPPA
- 4 Inference
- 5 Results
- 6 Conclusions

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.

Outline

- 1 Introduction
- 2 Probabilistic Programming
- 3 ProPPA
- 4 Inference
- 5 Results
- 6 Conclusions

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) have a **name** or type, but also a **stochastic** duration or **rate**.

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) have a **name** or type, but also a **stochastic** duration or **rate**.

In systems biology modelling it is these rates that are often **unknown**.

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) have a **name** or type, but also a **stochastic** duration or **rate**.

In systems biology modelling it is these rates that are often **unknown**.

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



Stochastic Process Algebra

In a stochastic process algebra actions (reactions) have a **name** or type, but also a **stochastic** duration or **rate**.

In systems biology modelling it is these rates that are often **unknown**.

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



Q is the infinitesimal generator matrix characterising the CTMC.

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) have a **name** or type, but also a **stochastic** duration or **rate**.

In systems biology modelling it is these rates that are often **unknown**.

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



Q is the infinitesimal generator matrix characterising the CTMC.

Models are typically executed by **simulation** using Gillespie's Stochastic Simulation Algorithm (SSA) or similar.

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

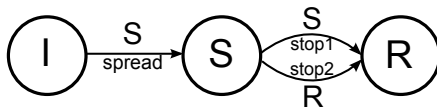
The objective of ProPPA is to retain the features of the stochastic process algebra:

- 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


 $k_s = 0.5;$
 $k_r = 0.1;$
 $\text{kineticLawOf spread} : k_s * I * S;$
 $\text{kineticLawOf stop1} : k_r * S * S;$
 $\text{kineticLawOf stop2} : k_r * S * R;$
 $I = (\text{spread}, 1) \downarrow ;$
 $S = (\text{spread}, 1) \uparrow + (\text{stop1}, 1) \downarrow + (\text{stop2}, 1) \downarrow ;$
 $R = (\text{stop1}, 1) \uparrow + (\text{stop2}, 1) \uparrow ;$
 $I[10] \quad \boxtimes \quad S[5] \quad \boxtimes \quad R[0]$

* *

Additions

Declaring uncertain parameters:

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

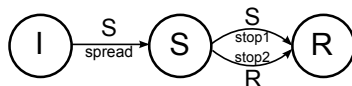
Providing observations:

- `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;
```

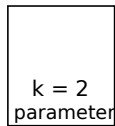
```
I = (spread,1) ↓ ;
S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
R = (stop1,1) ↑ + (stop2,1) ↑ ;
```

```
I[10]  S[5]  R[0]
```

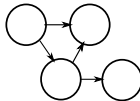
```
observe('trace')
infer('ABC') //Approximate Bayesian Computation
```

Semantics

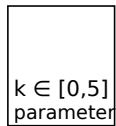
- A Bio-PEPA model can be interpreted as a CTMC; however, CTMCs cannot capture uncertainty in the rates (every transition must have a concrete rate).
- ProPPA models include uncertainty in the parameters, which translates into uncertainty in the transition rates.
- A ProPPA model should be mapped to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



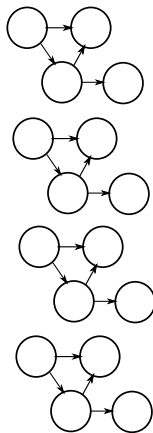
model



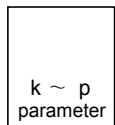
CTMC



model



set
of CTMCs

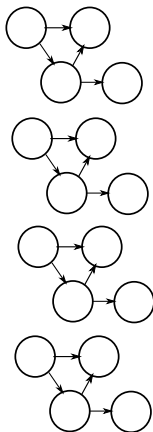


model



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.
- $V : S \rightarrow 2^{2^A}$ gives a set of acceptable labellings for each state.
- $\phi : S \times [0, 1]^k \rightarrow \{0, 1\}$ is the **constraint function**.

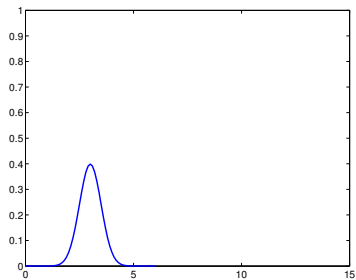
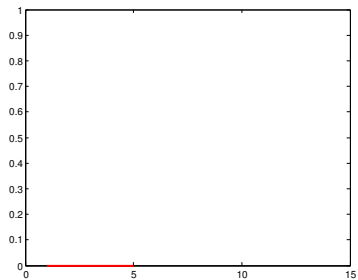
Constraint Markov Chains

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**.



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.
 - $V : S \rightarrow 2^{2^A}$ gives a set of acceptable labellings for each state.
 - $\phi : S \times [0, \infty)^k \rightarrow [0, \infty)$ 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 rate of a transition

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).

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).

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

This gives rise the **underlying PCMC**.

Simulating Probabilistic Constraint Markov Chains

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

- 1 For each trajectory, for each uncertain transition rate, sample once at the start of the run and use that value throughout ([Uncertain Markov Chains](#)) ;
- 2 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 ([Imprecise Markov Chains](#)).

Simulating Probabilistic Constraint Markov Chains

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

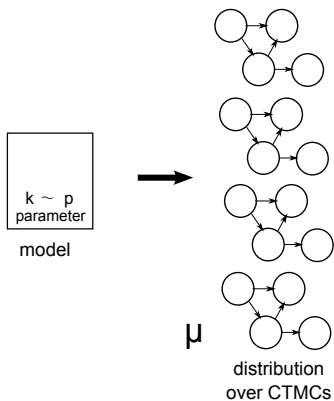
- 1 For each trajectory, for each uncertain transition rate, sample once at the start of the run and use that value throughout ([Uncertain Markov Chains](#)) ;
- 2 During each trajectory, each time a transition with an uncertain rate is encountered, sample a value but then discard it and re-sample whenever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

Our current work is focused on the [Uncertain Markov Chain](#) case.

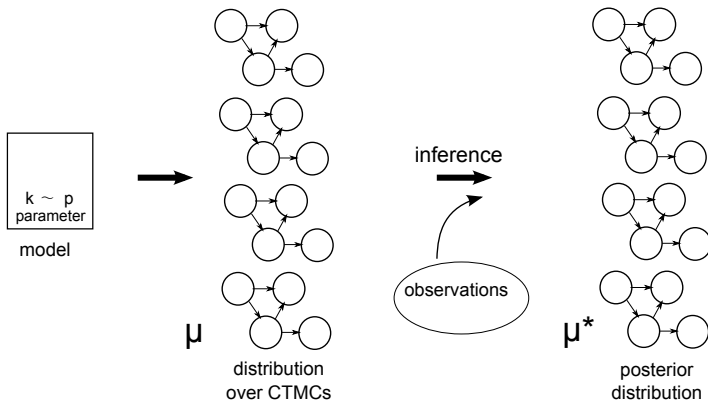
Outline

- 1 Introduction
- 2 Probabilistic Programming
- 3 ProPPA
- 4 Inference
- 5 Results
- 6 Conclusions

Inference



Inference



Inference

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

- 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.
- Exact inference is often impossible, as we cannot calculate the likelihood.
- We must use approximate algorithms or approximations of the system.

Inferring likelihood in uncertain CTMCs

Transient 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 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 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

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

Likelihood is hard to compute:

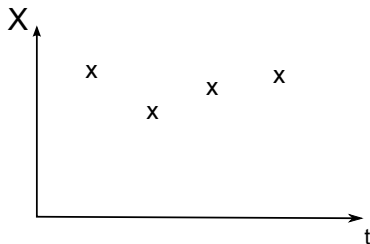
- Computing $e^{\mathbf{Q}t}$ is expensive if the state space is large
- Impossible directly in infinite state-spaces

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.

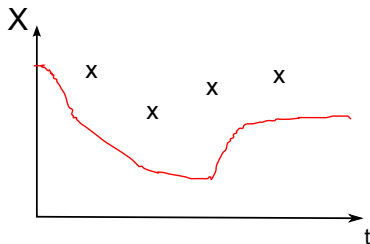
Basic Inference

- 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.



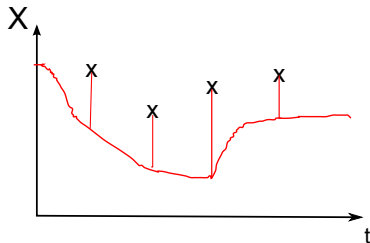
Basic Inference

- 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.



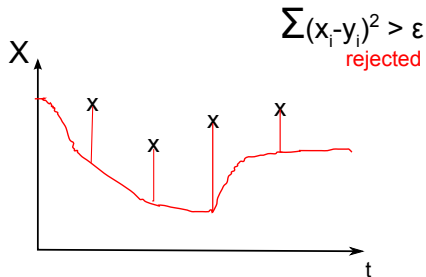
Basic Inference

- 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.



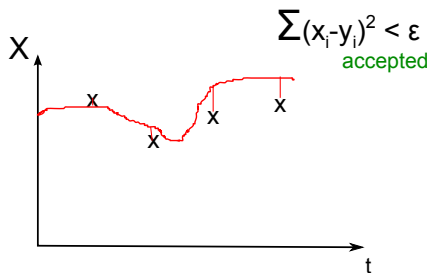
Basic Inference

- 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.

J.M.Marin, P.Pudlo, C.P.Robert, R.J.Ryder. *Approximate Bayesian computational methods*. Statistics and Computing 22(6) (2012)

This results in an approximation to the posterior distribution.

As $\epsilon \rightarrow 0$, 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**

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

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

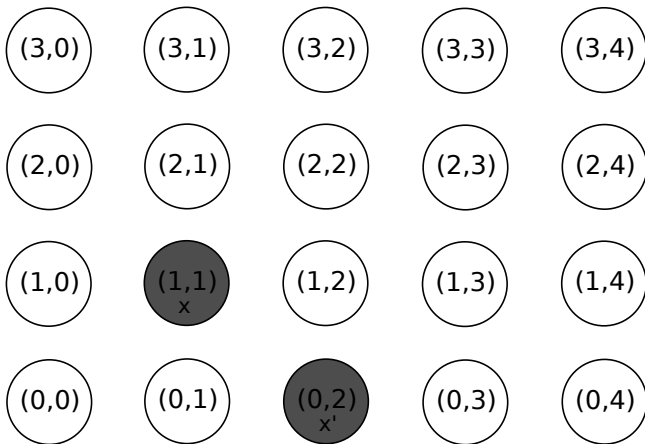
where the f_n 's are computable.

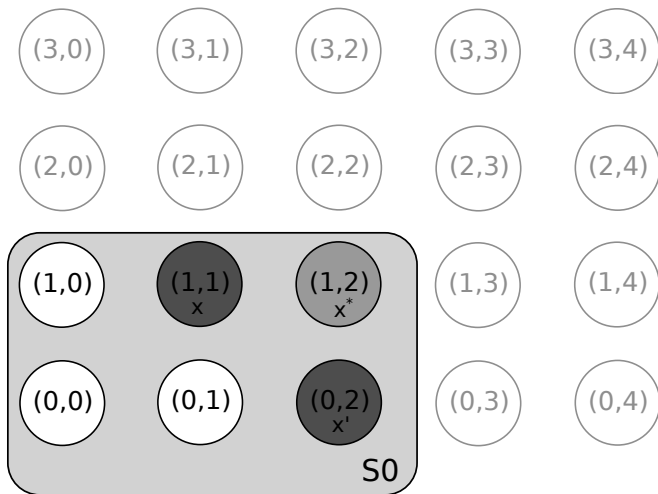
- Truncate the sum randomly: stop at term k with probability q_k .
- Form \hat{f} as a partial sum of the f_n , $n = 1, \dots, k$, rescaled appropriately.

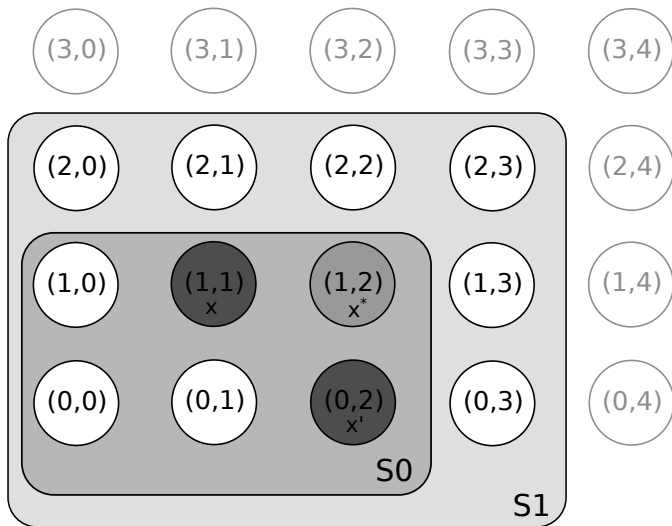
$$\hat{f} = \sum_{n=0}^k \frac{f_n}{\prod_{j=0}^{k-1} (1 - q_j)}$$

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
```







Expanding the likelihood

The likelihood can be written as an infinite series:

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

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)

Russian Roulette

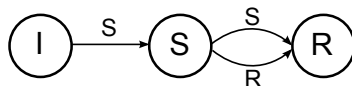
$$\hat{f} = \sum_{n=0}^k \frac{f_n}{\prod_{j=0}^{k-1} (1 - q_j)}$$

- In our case, f is a probability that we wish to approximate, $p(x', t' \mid x, t)$.
- 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.

Outline

- 1 Introduction
- 2 Probabilistic Programming
- 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) ↑ ;
```

```
I[10]  S[5]  R[0]
```

```
observe('trace')
```

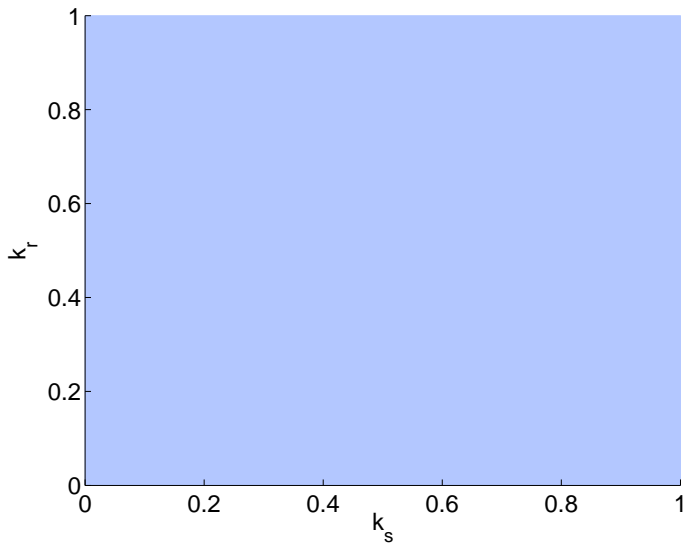
```
infer('ABC') //Approximate Bayesian Computation
```

Results

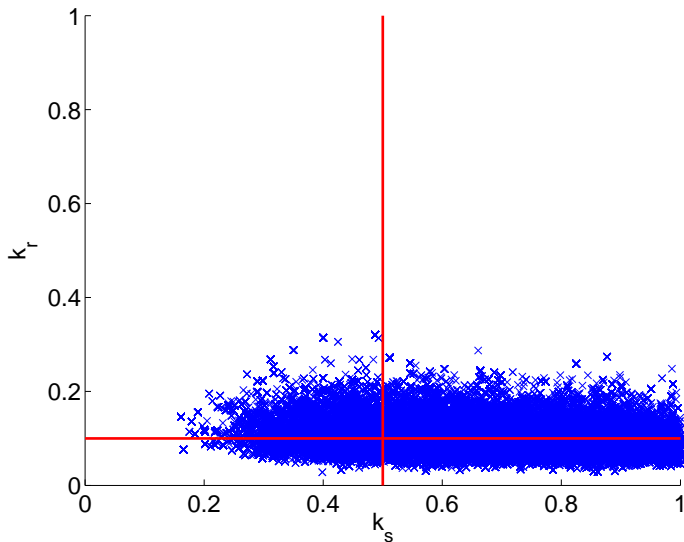
Tested on the rumour-spreading example, giving the two parameters uniform priors.

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

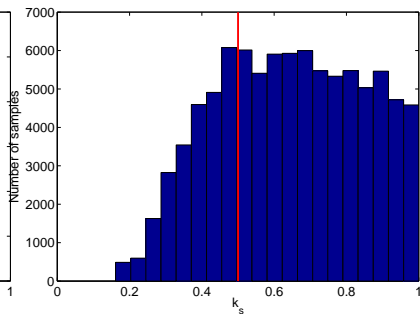
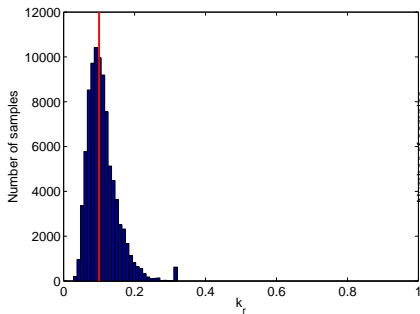
Results: ABC



Results: ABC



Results: ABC



Genetic Toggle Switch

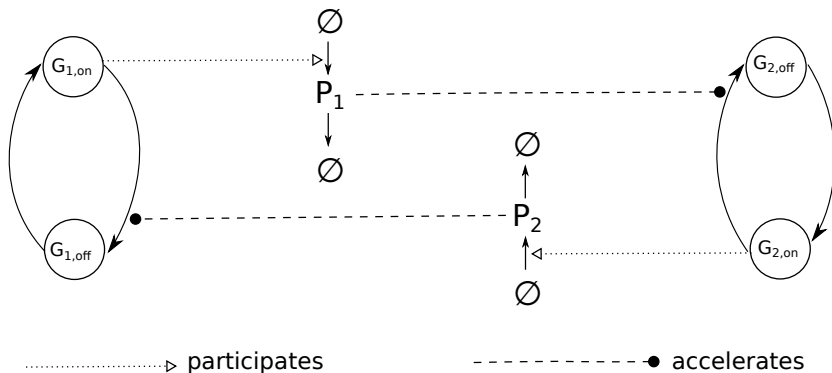
- Two mutually-repressing genes: promoters (unobserved) and their protein products
- Bistable behaviour: switching induced by environmental changes
- Synthesised in *E. coli*

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

- Stochastic variant where switching is induced by noise

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

Genetic Toggle Switch



Toggle switch model: species

$G1 = \text{activ1} \uparrow + \text{deact1} \downarrow + \text{expr1} \oplus;$

$G2 = \text{activ2} \uparrow + \text{deact2} \downarrow + \text{expr2} \oplus;$

$P1 = \text{expr1} \uparrow + \text{degr1} \downarrow + \text{deact2} \oplus ;$

$P2 = \text{expr2} \uparrow + \text{degr2} \downarrow + \text{deact1} \oplus$

$G1[1] \lt * \gt G2[0] \lt * \gt P1[20] \lt * \gt P2[0]$

`observe(toggle_obs);`

`infer(rouletteGibbs);`

```
 $\theta_1$  = Gamma(3,5); //etc...
```

```
kineticLawOf expr1 :  $\theta_1$  * G1;
```

```
kineticLawOf expr2 :  $\theta_2$  * G2;
```

```
kineticLawOf degr1 :  $\theta_3$  * P1;
```

```
kineticLawOf degr2 :  $\theta_4$  * P2;
```

```
kineticLawOf activ1 :  $\theta_5$  * (1 - G1);
```

```
kineticLawOf activ2 :  $\theta_6$  * (1 - G2);
```

```
kineticLawOf deact1 :  $\theta_7$  *  $\exp(r * P2)$  * G1;
```

```
kineticLawOf deact2 :  $\theta_8$  *  $\exp(r * P1)$  * G2;
```

```
G1 = activ1  $\uparrow$  + deact1  $\downarrow$  + expr1  $\oplus$ ;
```

```
G2 = activ2  $\uparrow$  + deact2  $\downarrow$  + expr2  $\oplus$ ;
```

```
P1 = expr1  $\uparrow$  + degr1  $\downarrow$  + deact2  $\oplus$  ;
```

```
P2 = expr2  $\uparrow$  + degr2  $\downarrow$  + deact1  $\oplus$ 
```

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

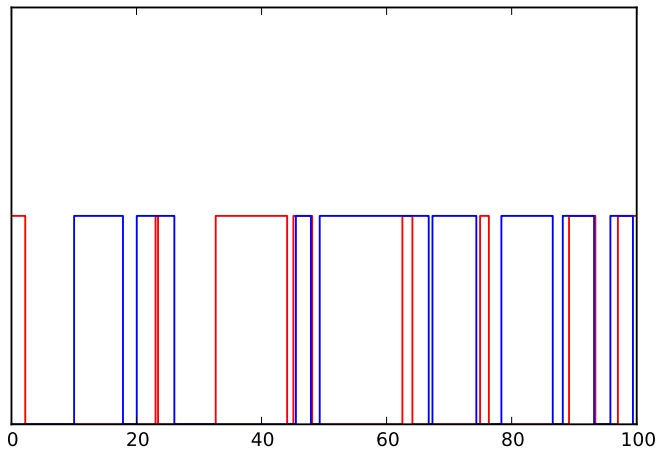
```
observe(toggle_obs);
```

```
infer(rouletteGibbs);
```

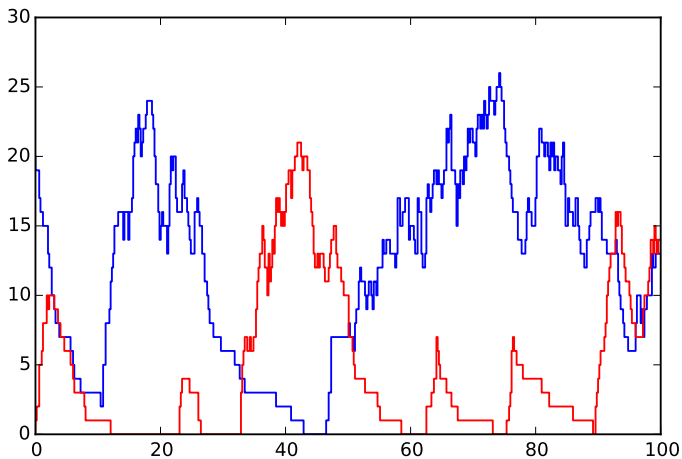
Experiment

- 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

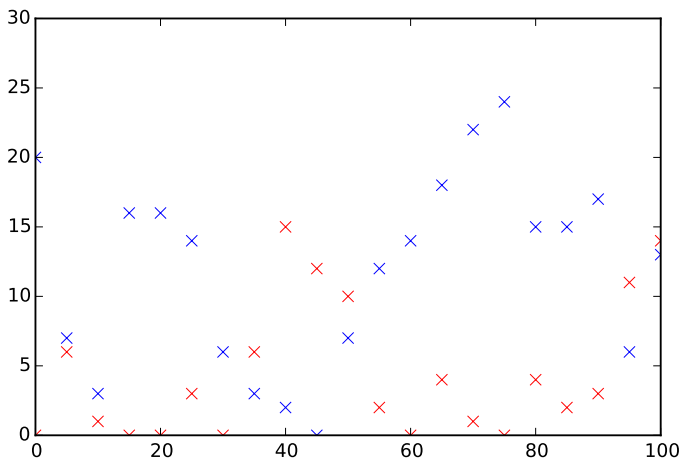
Promoters



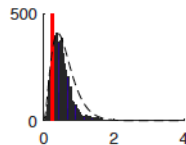
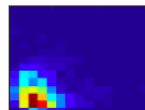
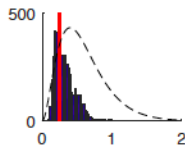
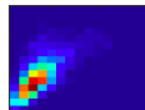
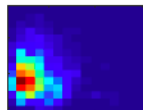
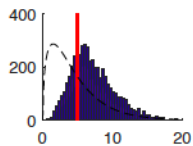
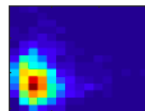
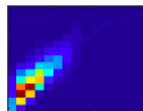
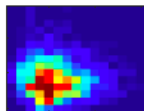
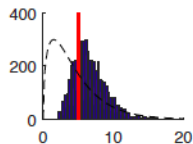
Proteins



Observations used



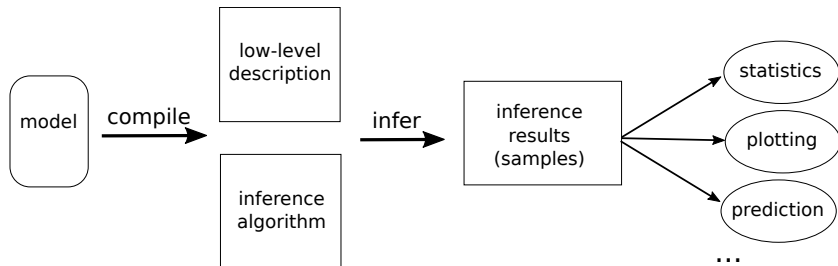
Results



Outline

- 1 Introduction
- 2 Probabilistic Programming
- 3 ProPPA
- 4 Inference
- 5 Results
- 6 Conclusions

ProPPA Workflow



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.

Thanks

- Anastasis Georgoulas



- Guido Sanguinetti

