Probabilistic Programming for Stochastic Dynamical Systems (ProPPA)

Jane Hillston

School of Informatics, University of Edinburgh

8th April 2019

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

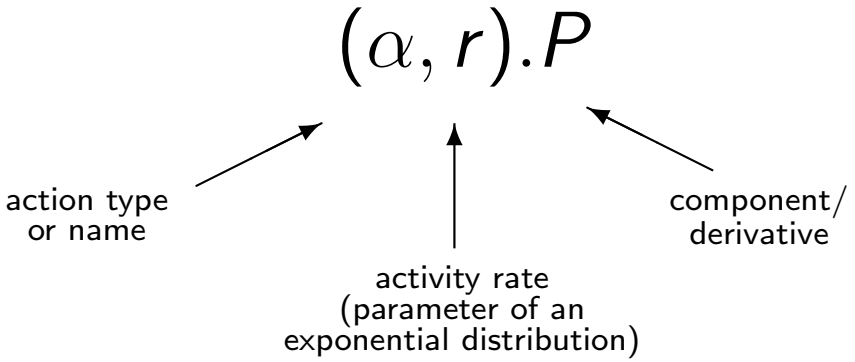
4 Inference

5 Results

6 Conclusions

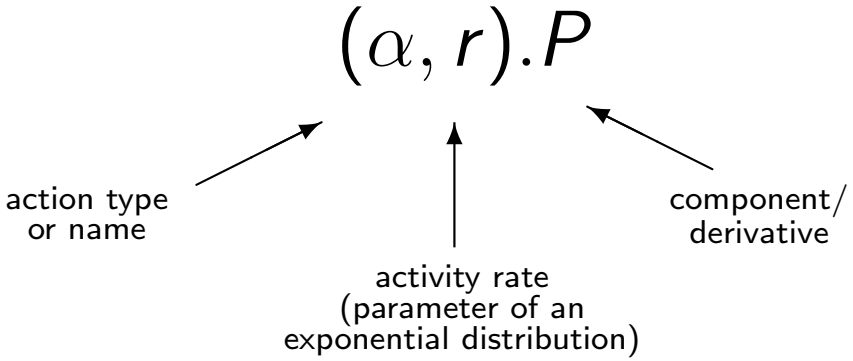
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

Benefits of integration

- Properties of the underlying mathematical structure may be deduced by the construction at the process algebra level.

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.

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.

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.

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.

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 cases 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 collective adaptive systems (e.g. PALOMA, CARMA) and ecological processes (e.g. PALPS, MELA).

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 cases 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 collective adaptive 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

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 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.

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.

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.

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)

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

The syntax

Sequential component (species component)

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

The syntax

Sequential component (species component)

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

The syntax

Sequential component (species component)

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

The syntax

Sequential component (species component)

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

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

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

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

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

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

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

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

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

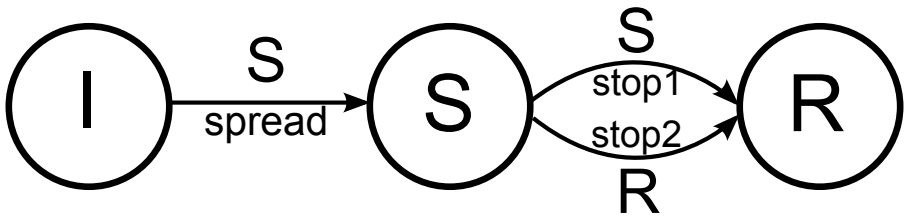
Each action α_j is associated with a rate f_{α_j}

The semantics

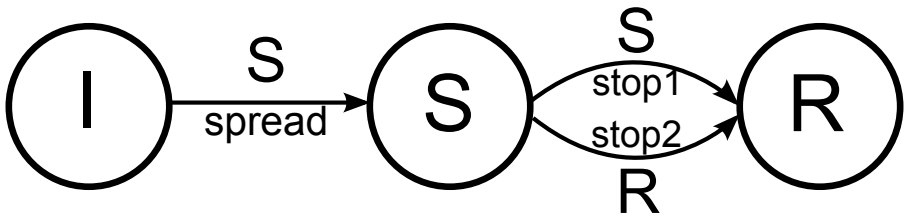
The semantics is defined by two transition relations:

- 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



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] ⋈*  S[5] ⋈*  R[0]
```

Formal modelling in systems biology

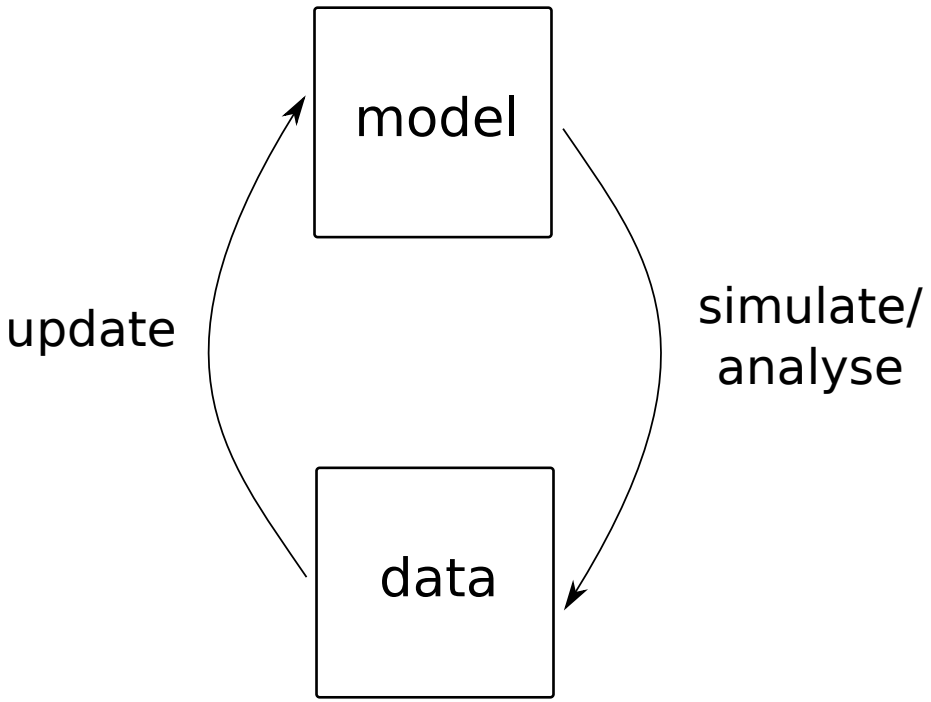
- Formal languages like Bio-PEPA provide a convenient interface for describing complex systems, reflecting what is known about the biological components and their behaviour.
- High-level abstraction eases writing and manipulating models.
- They are compiled into executable models which can be run to 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?

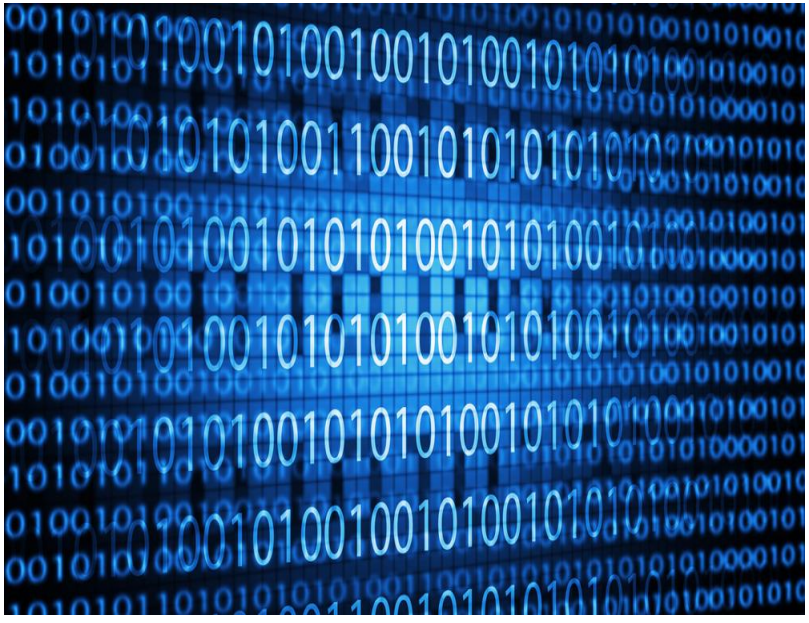
Jasmin Fisher, Thomas A. Henzinger: *Executable cell biology*. Nature Biotechnology 2007

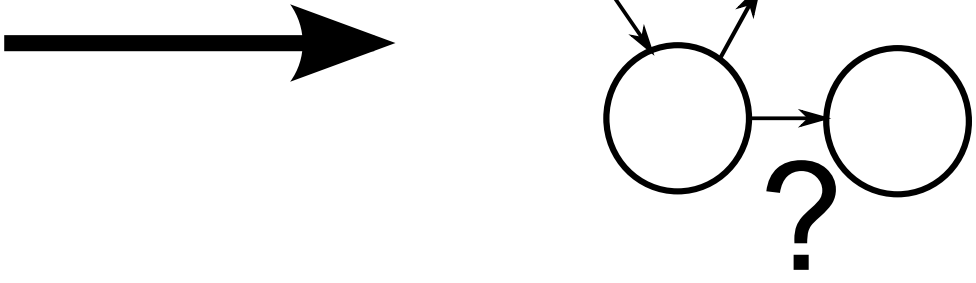
Optimizing models

Usual process of parameterising a model is iterative and manual.



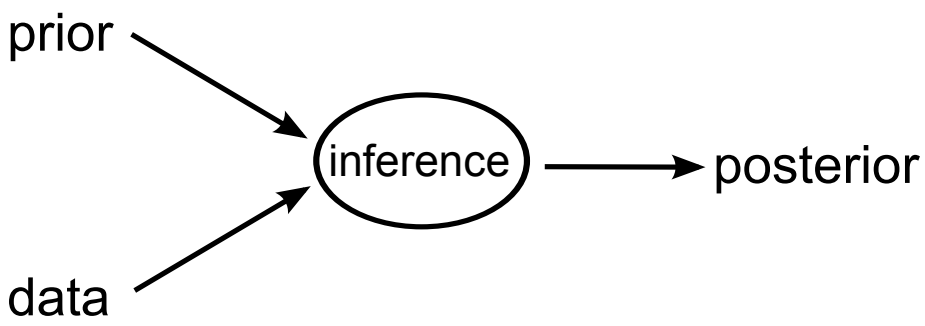
Alternative perspective



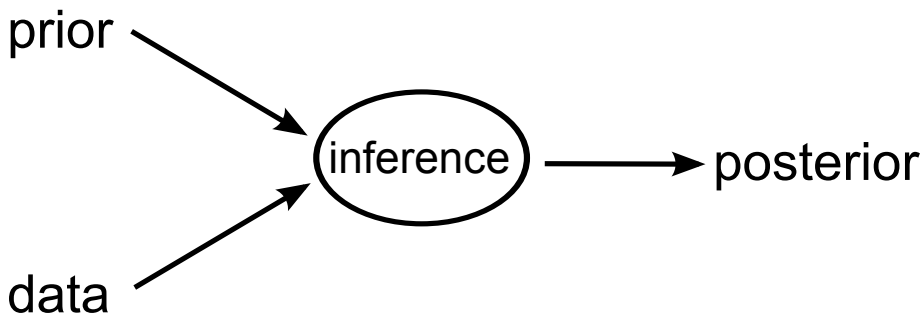


Model creation is data-driven

Machine Learning: Bayesian statistics



Machine Learning: Bayesian statistics

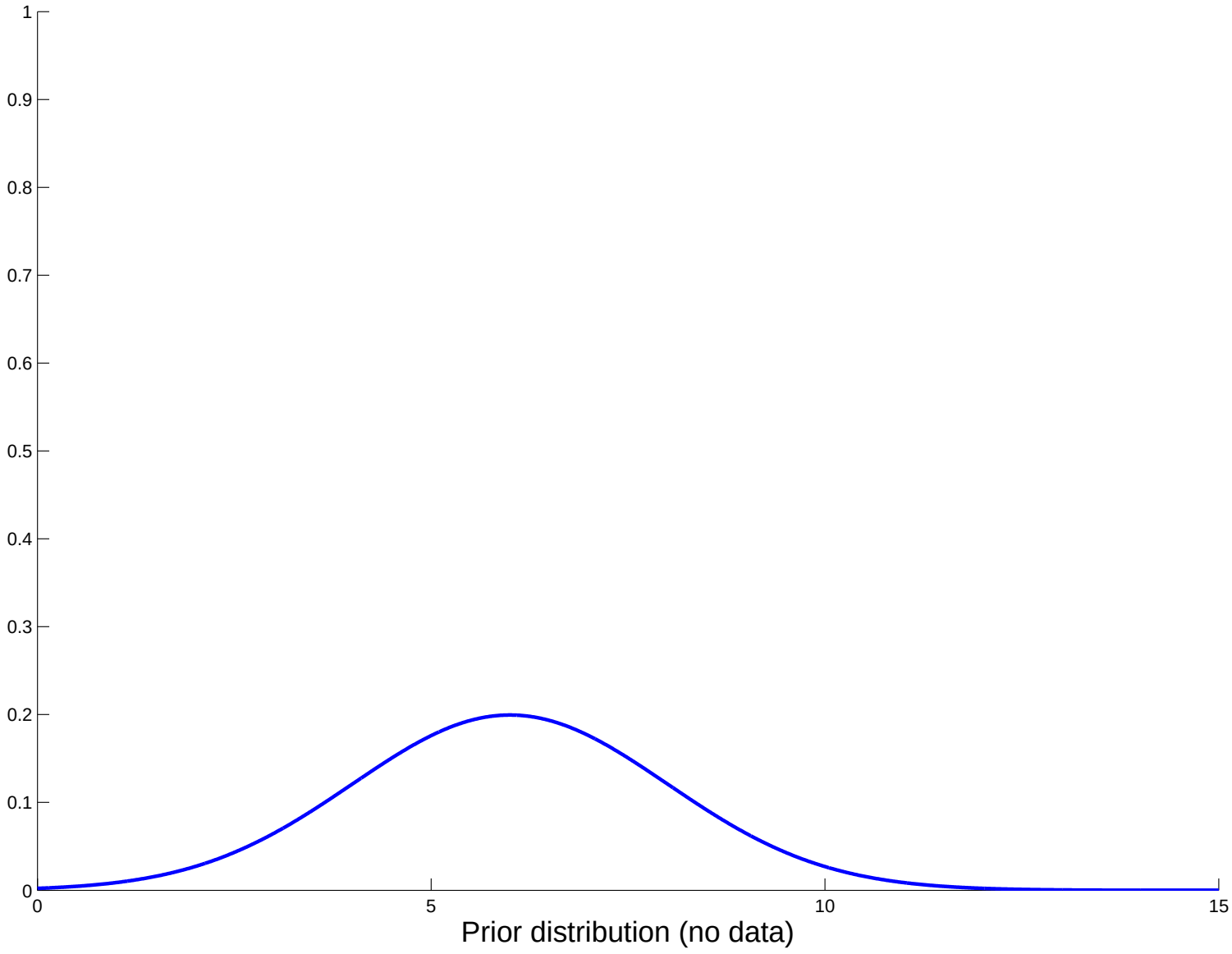


- Represent belief and uncertainty as probability distributions (prior, posterior).
- Treat parameters and unobserved variables similarly.
- Bayes' Theorem:

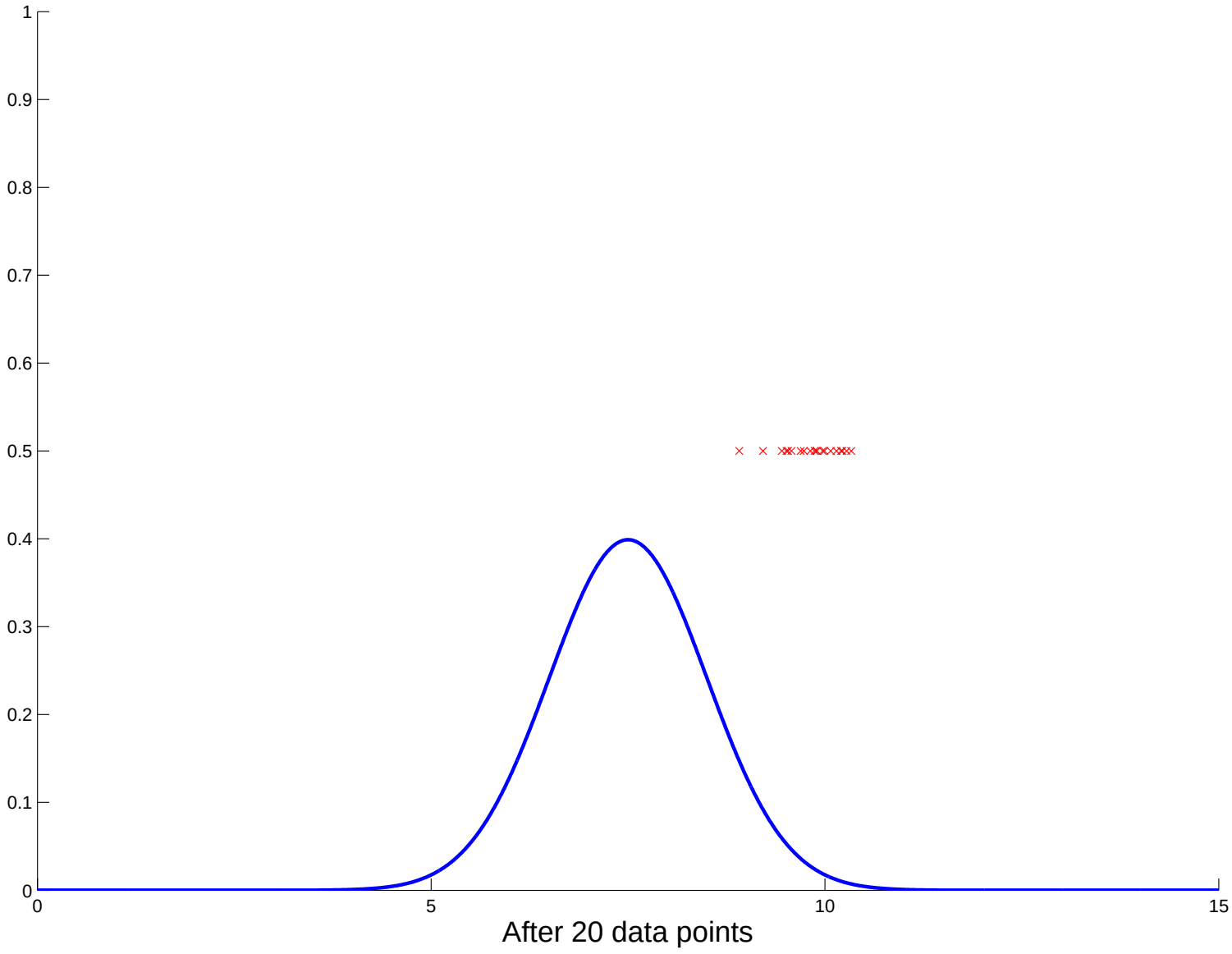
$$P(\theta \mid D) = \frac{P(\theta) \cdot P(D \mid \theta)}{P(D)}$$

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

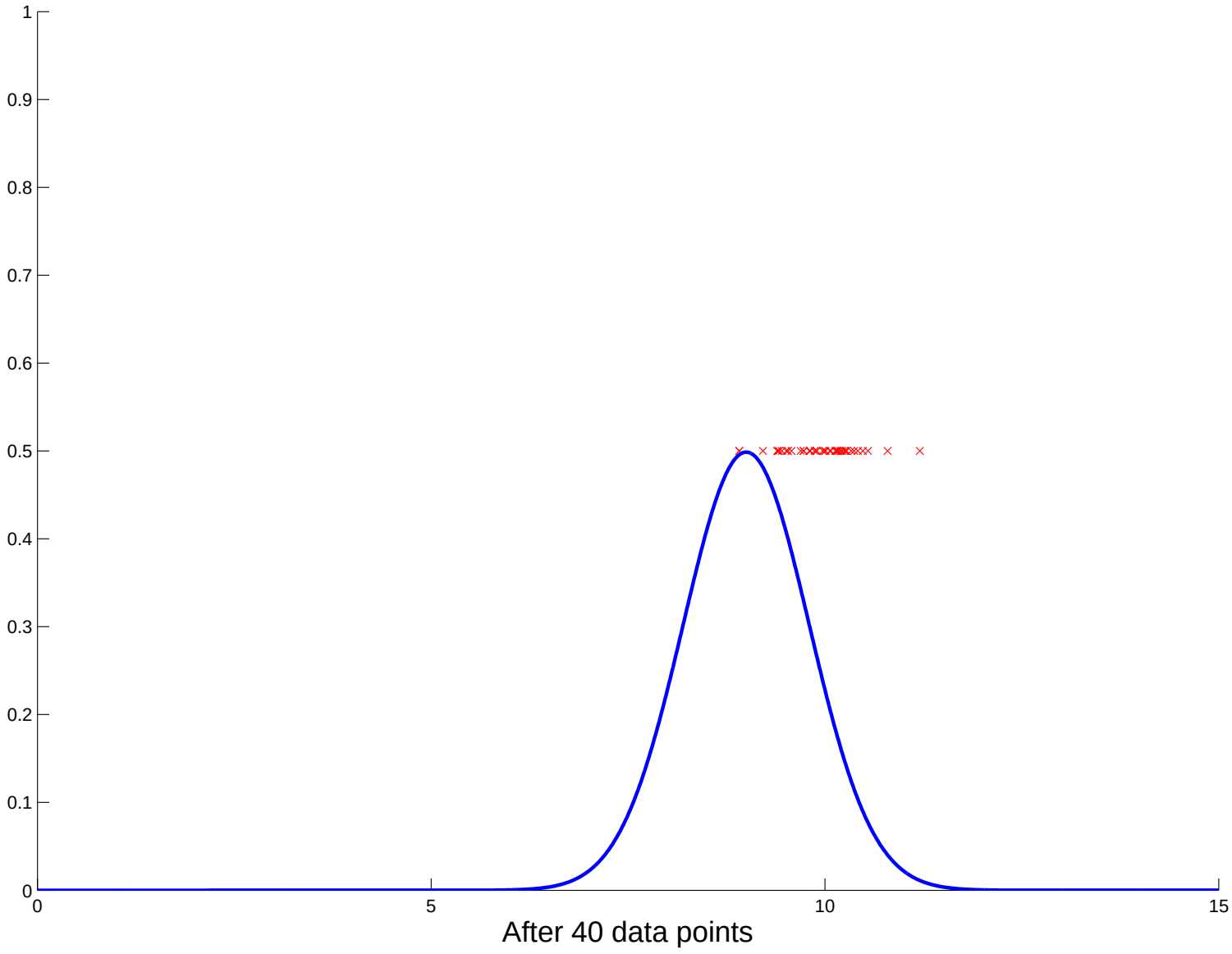
Bayesian statistics



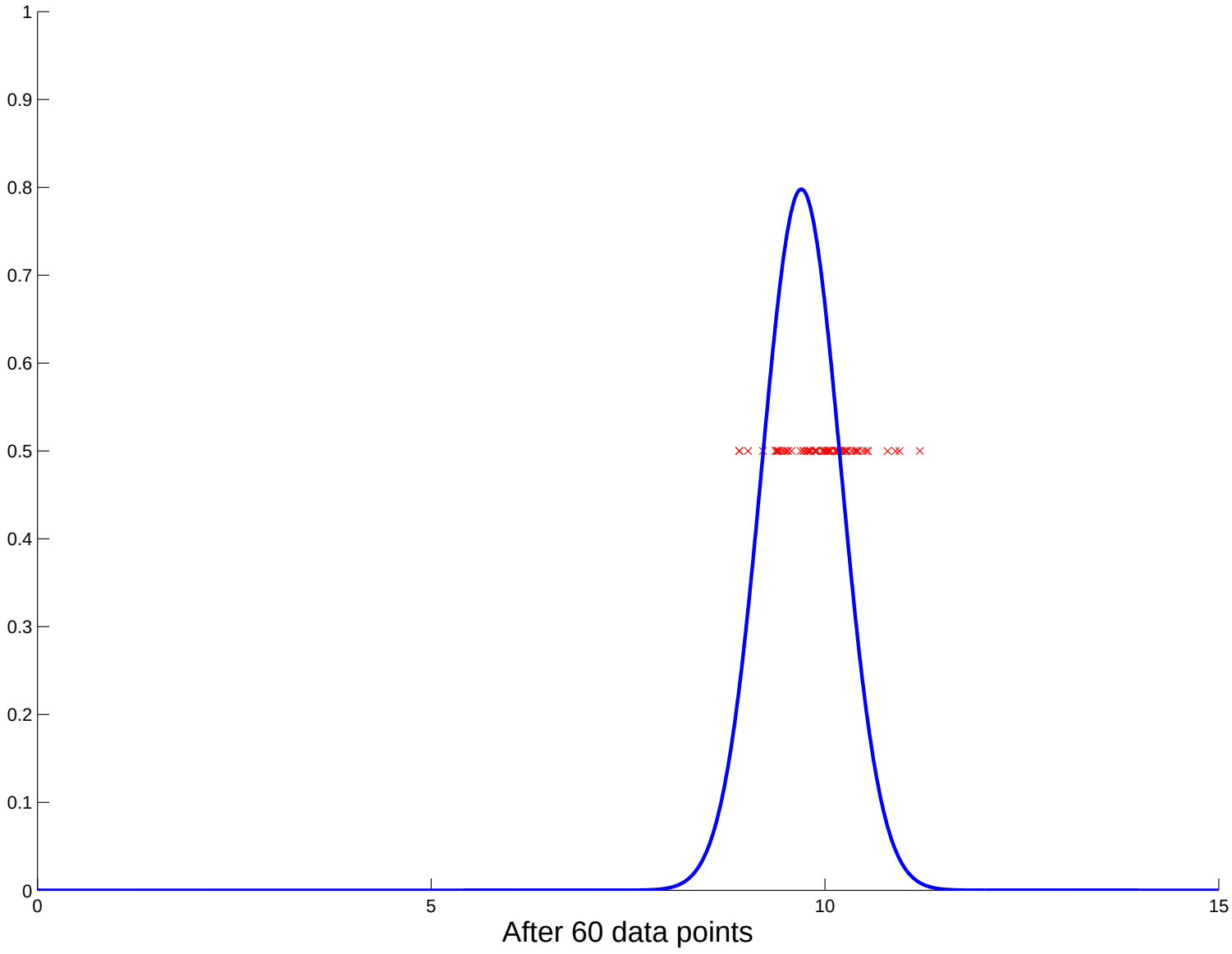
Bayesian statistics



Bayesian statistics



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, Stan, 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

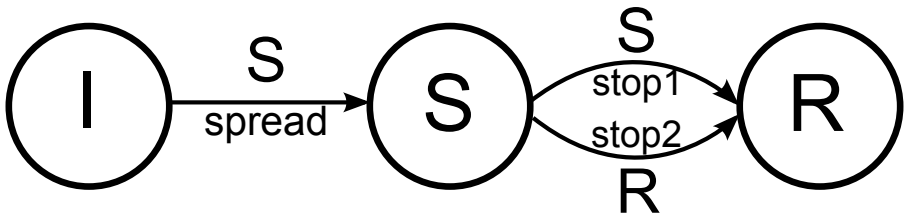
The objective of ProPPA is to retain the features of the stochastic process algebra:

- 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) ↑ ;

I[10] ⋈*  S[5] ⋈*  R[0]
```

Additions

Declaring uncertain parameters:

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

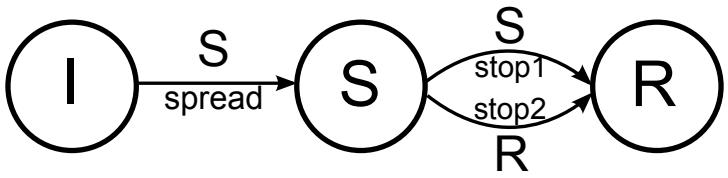
Providing observations:

- `observe('trace')`

Specifying inference approach:

- `infer('ABC')`

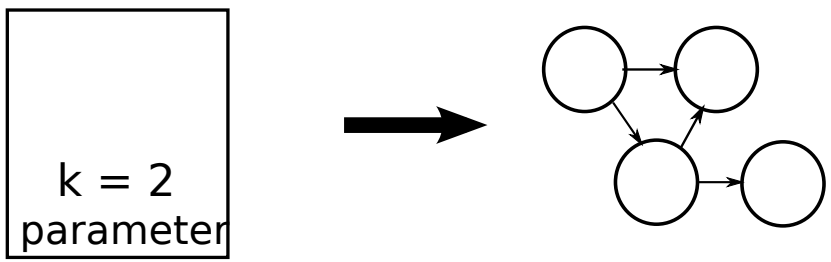
Additions

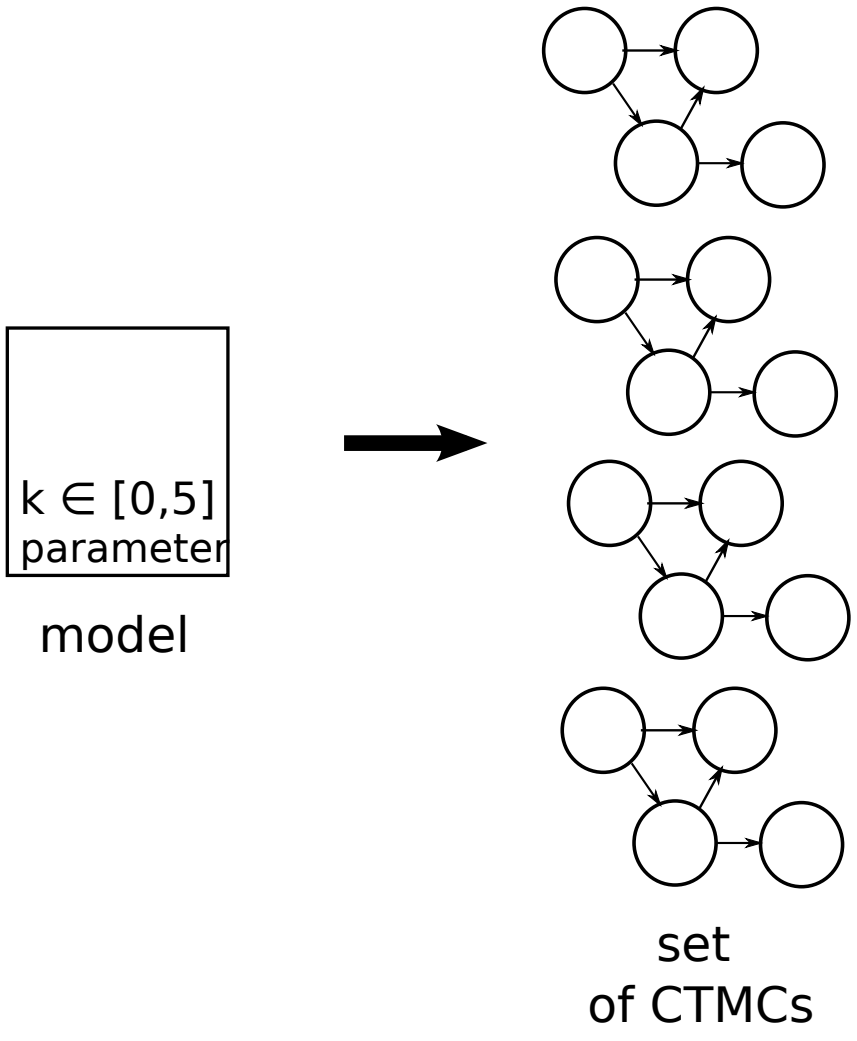


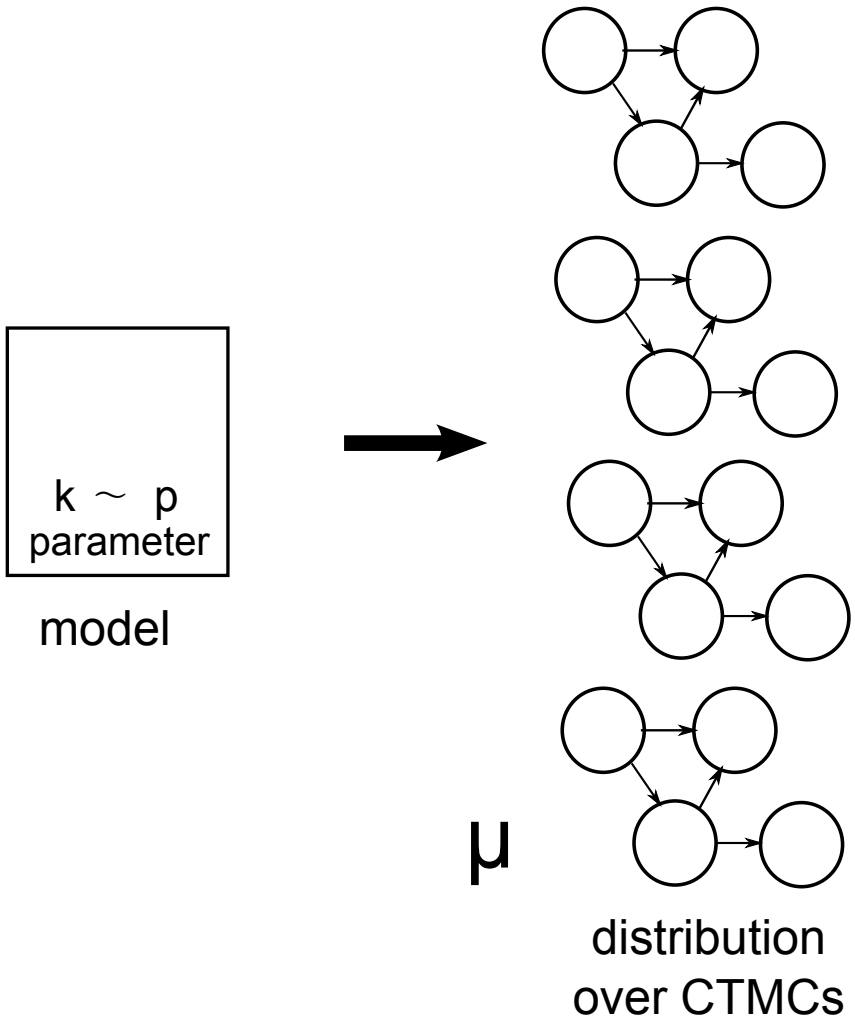

```
k_s = Uniform(0,1);
k_r = Uniform(0,1);

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

- As with PEPA, 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


k ∈ [0,5]
parameter
model
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

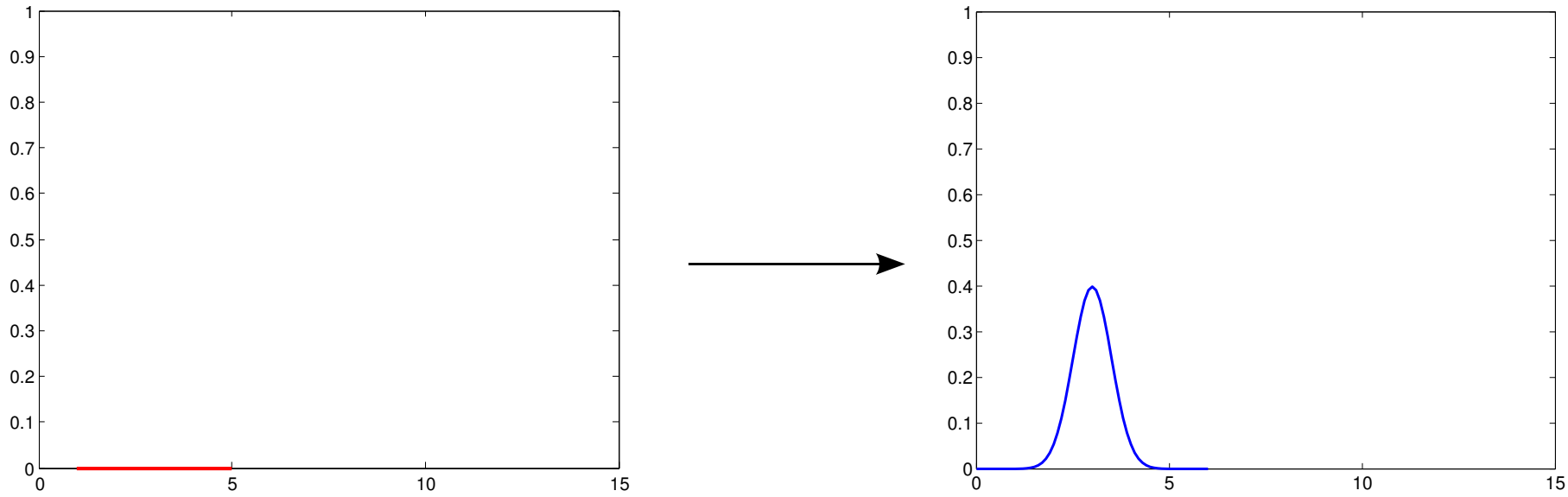
Constraint Markov Chains

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.

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.

This gives rise the underlying PCMC.

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.

Current work on ProPPA is focused on the Uncertain Markov Chain case.

Outline

1 Stochastic Process Algebras

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

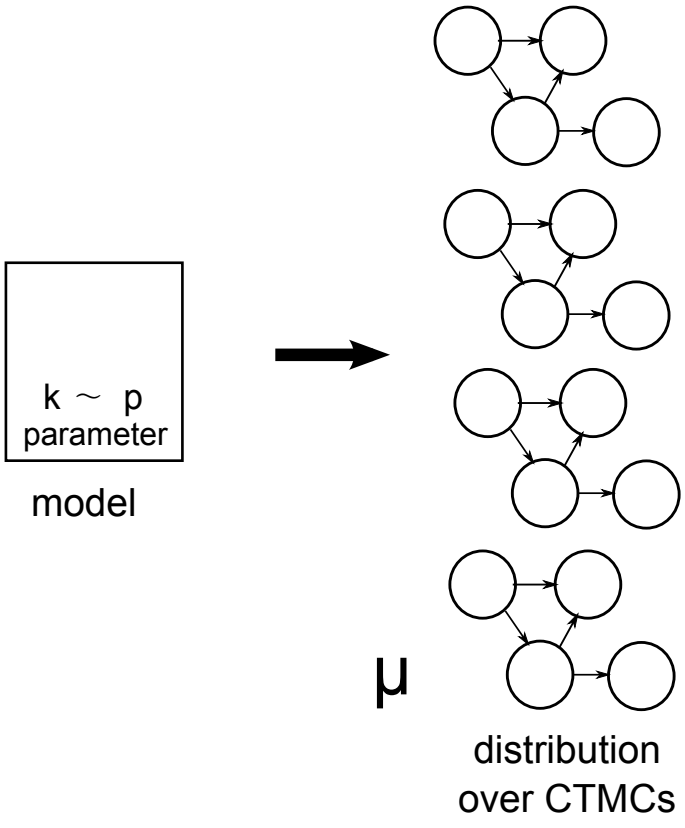
3 ProPPA

4 Inference

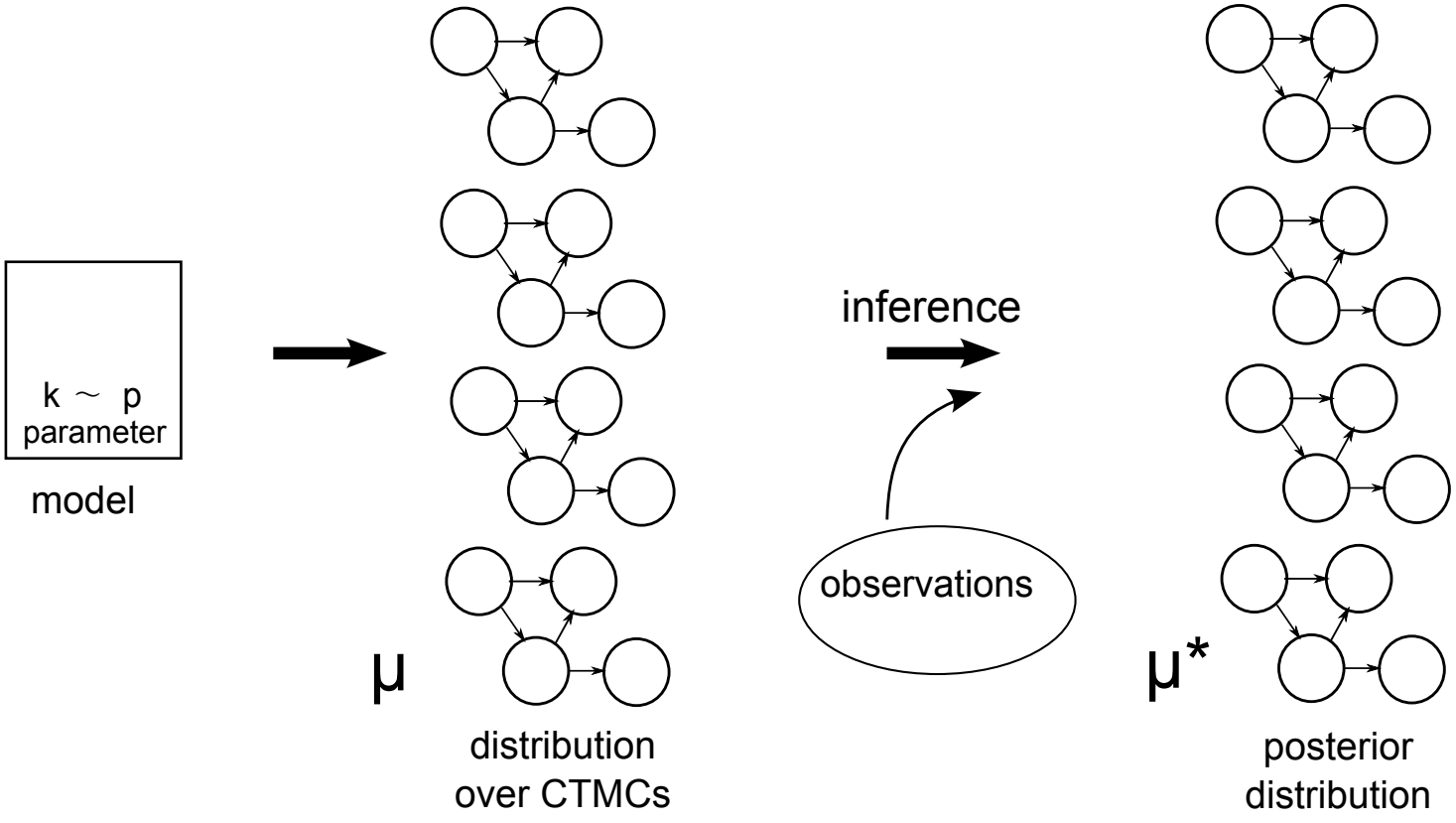
5 Results

6 Conclusions

Inference



Inference


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

Inference

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

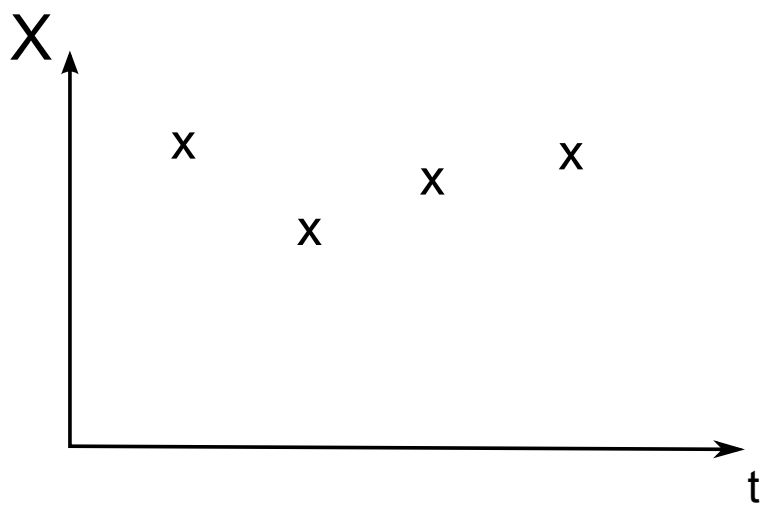
- Exact inference is impossible, as we cannot calculate the likelihood.
- We must use approximate algorithms or approximations of the system.
- The ProPPA semantics does not define a single inference algorithm, allowing for a modular approach.
- Different algorithms can act on different input (time-series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

Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that with a notion of distance.

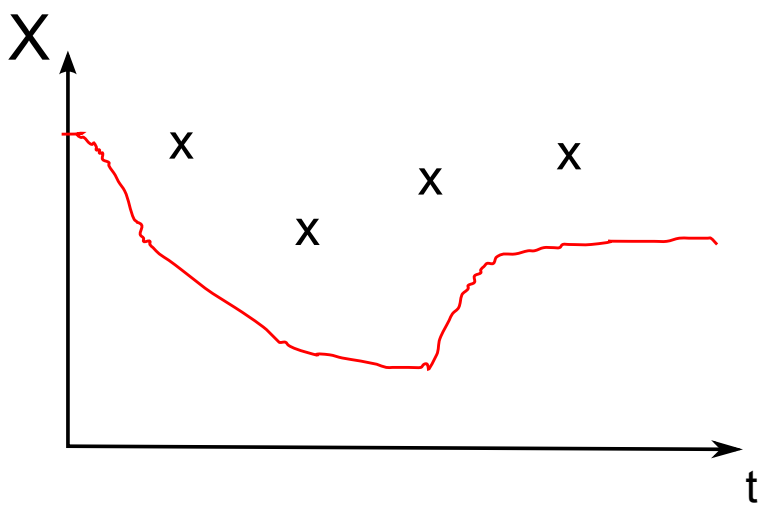
Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that with a notion of distance.



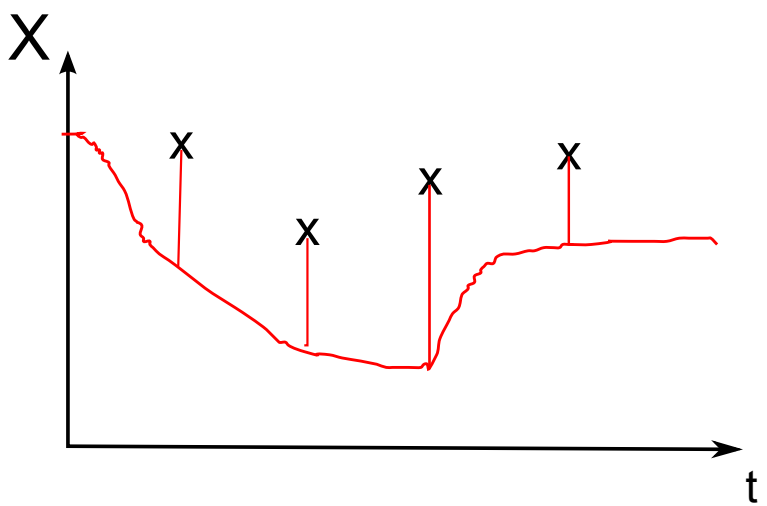
Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that with a notion of distance.



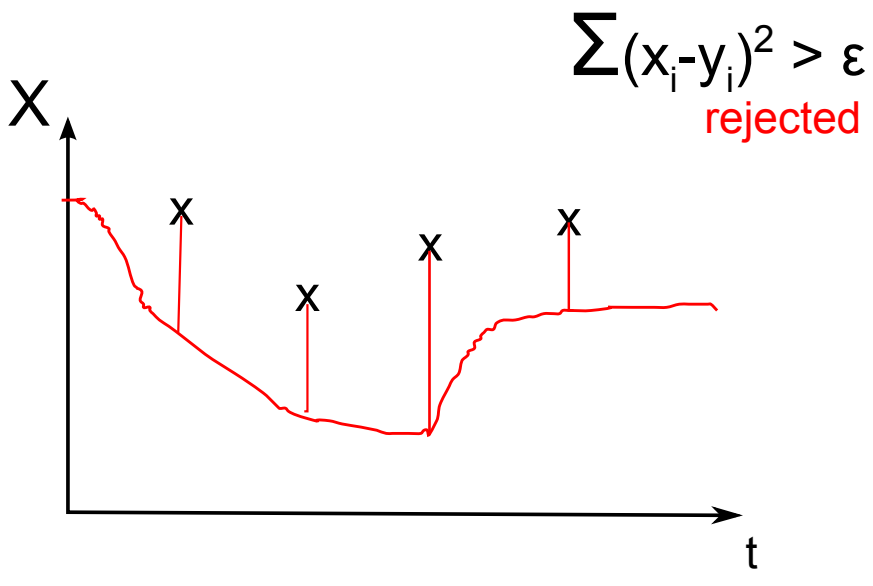
Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that with a notion of distance.



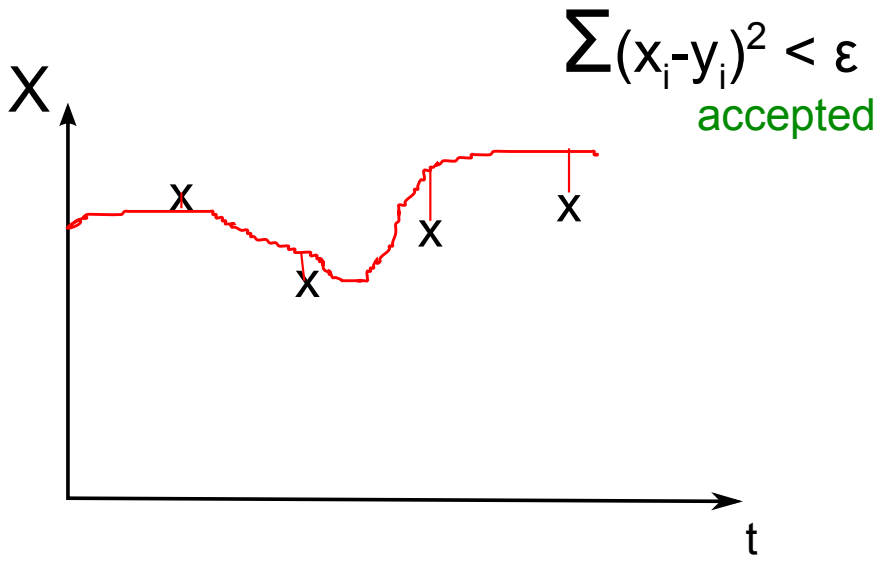
Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that with a notion of distance.



Inference

- Approximate Bayesian Computation is a simple simulation-based solution:
  - Approximates posterior distribution over parameters as a set of samples
  - Likelihood of parameters is hard to compute in CTMCs, approximates that 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 approximate 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.

Outline

1 Stochastic Process Algebras

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

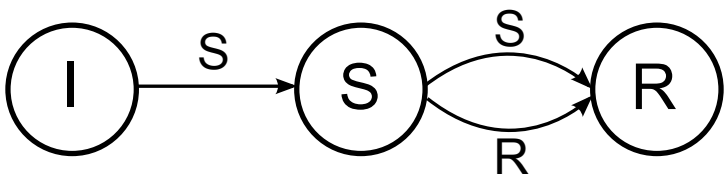
3 ProPPA

4 Inference

5 Results

6 Conclusions

Example model



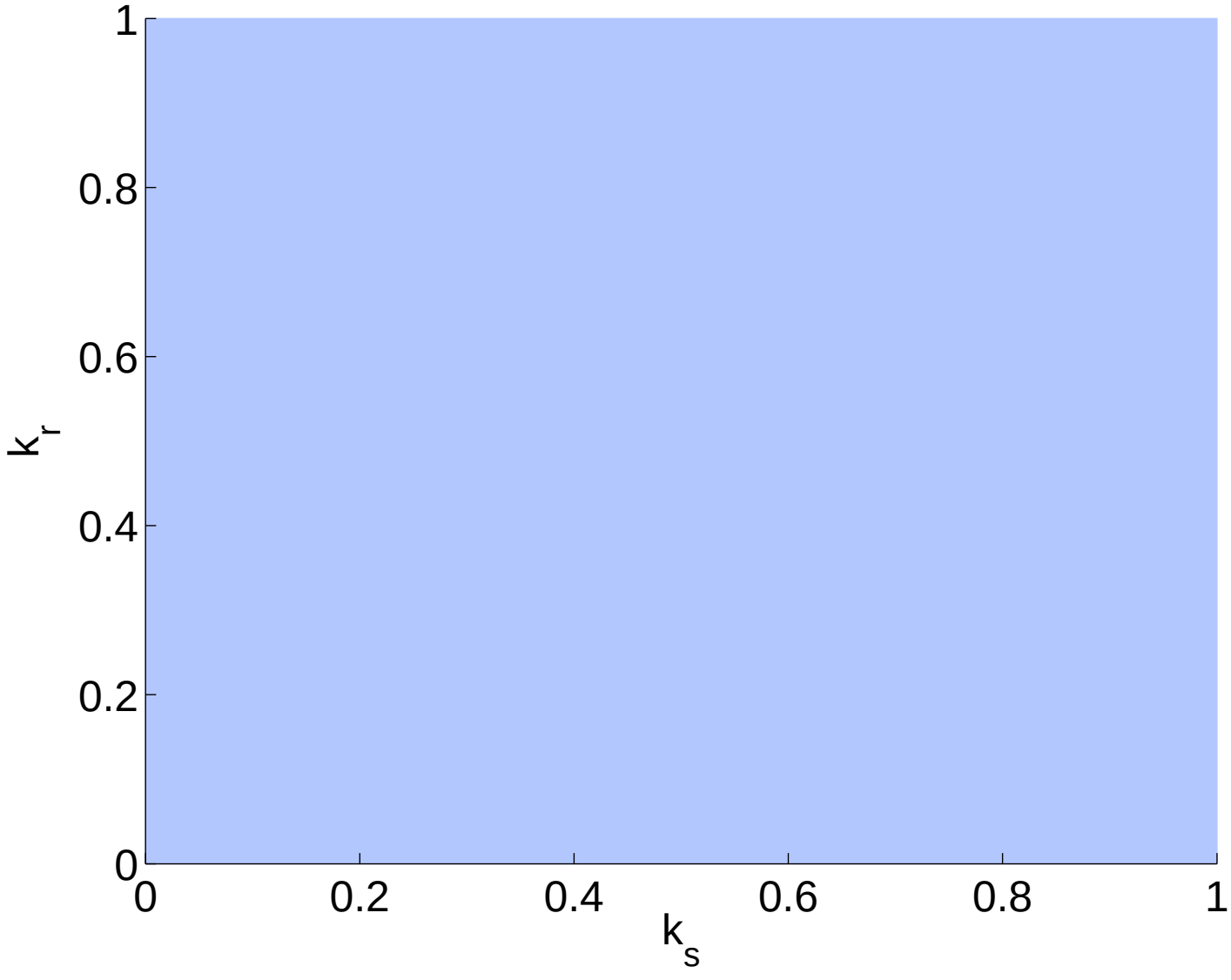

```
k_s = Uniform(0,1);
k_r = Uniform(0,1);

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) ↑ ;

I[10] ⋈*  S[5] ⋈*  R[0]

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

Results

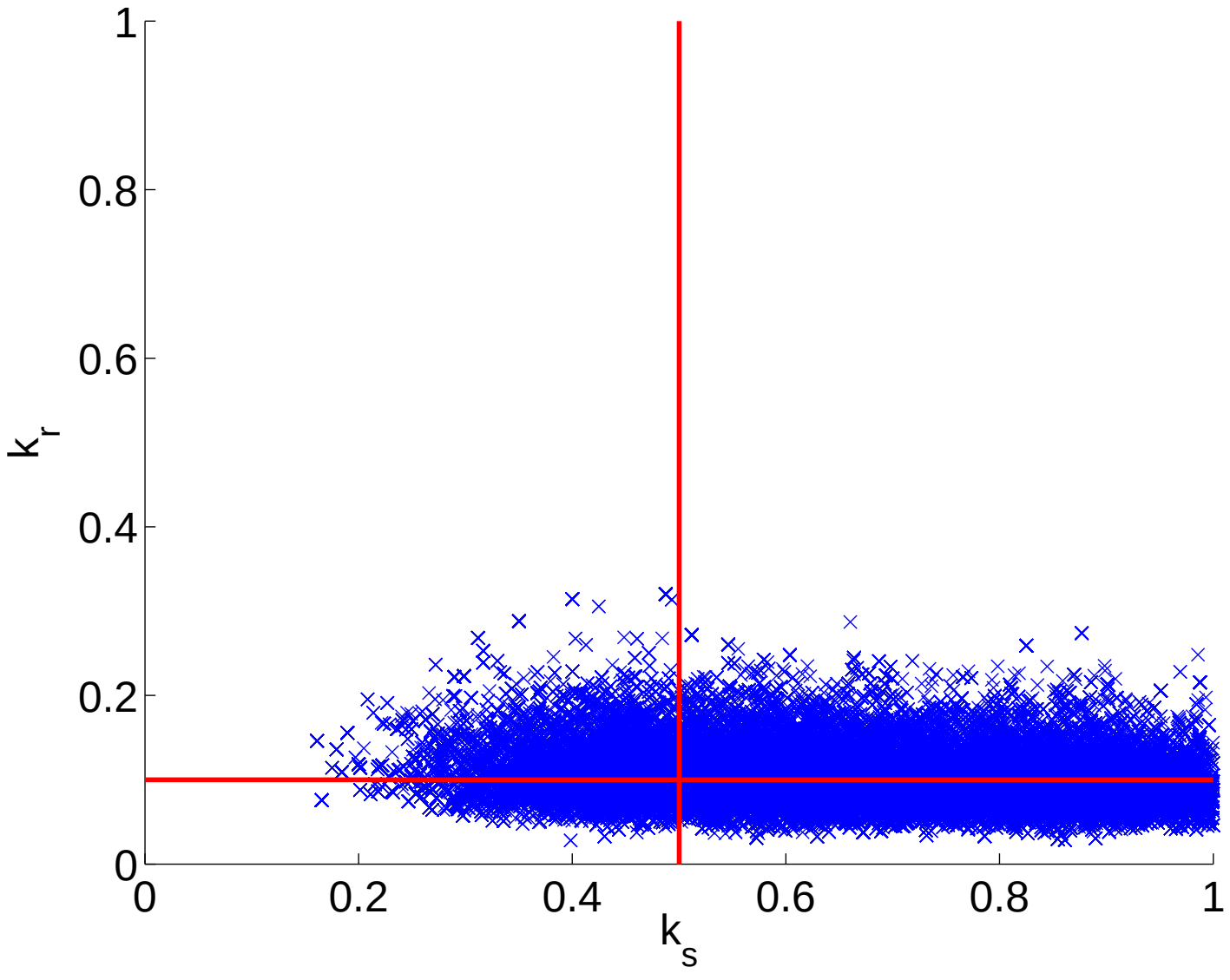
Tested on the rumour-spreading example, with uniform priors for the two parameters k_s and k_r.

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

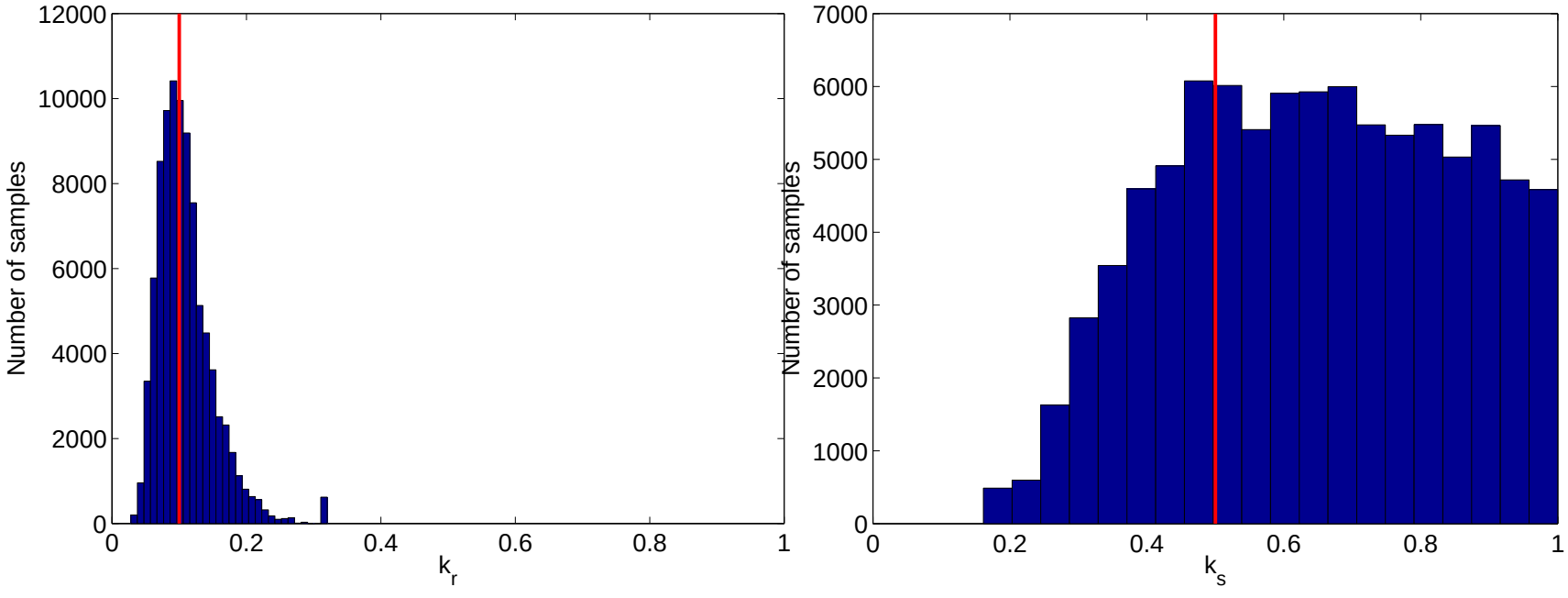
Results: ABC



Results: ABC



Results: ABC


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

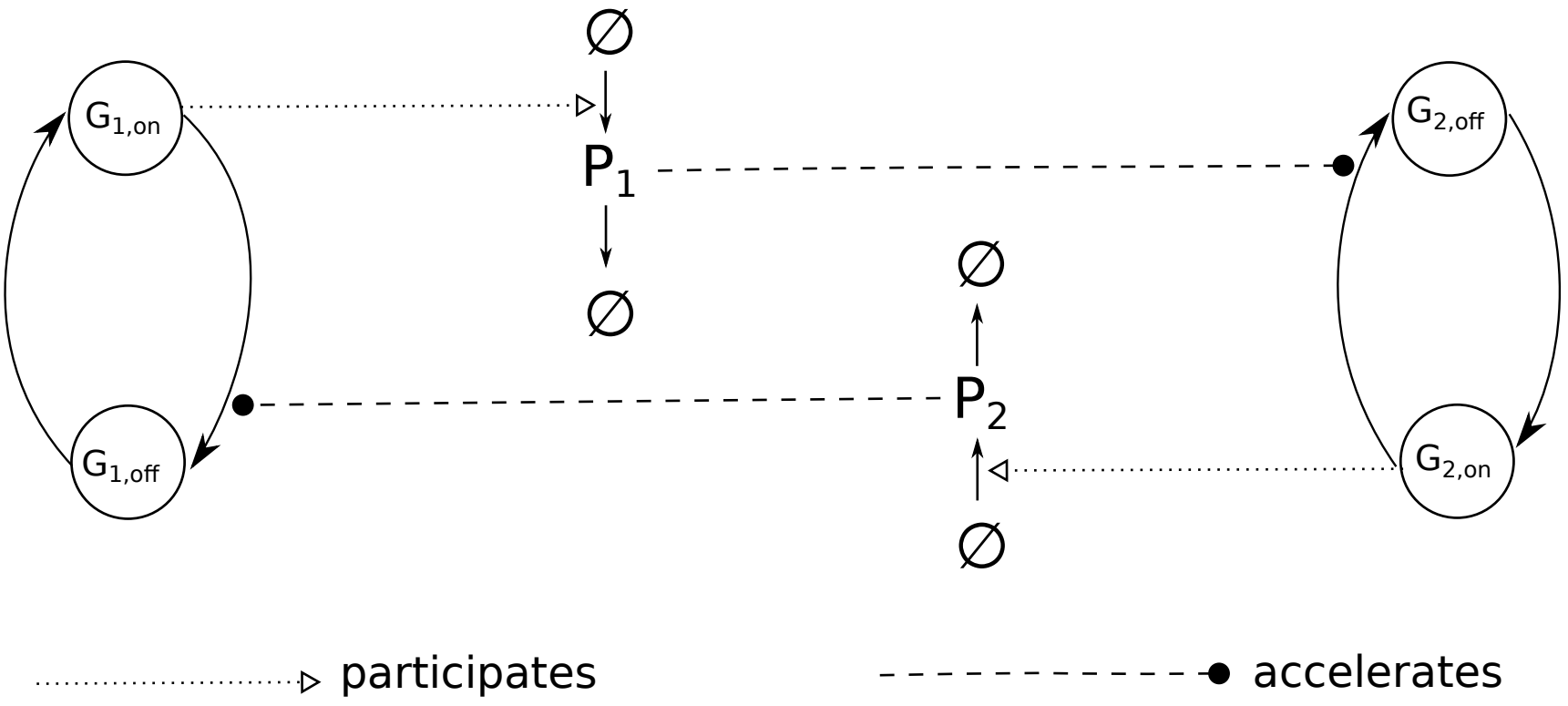
Genetic Toggle Switch

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

  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

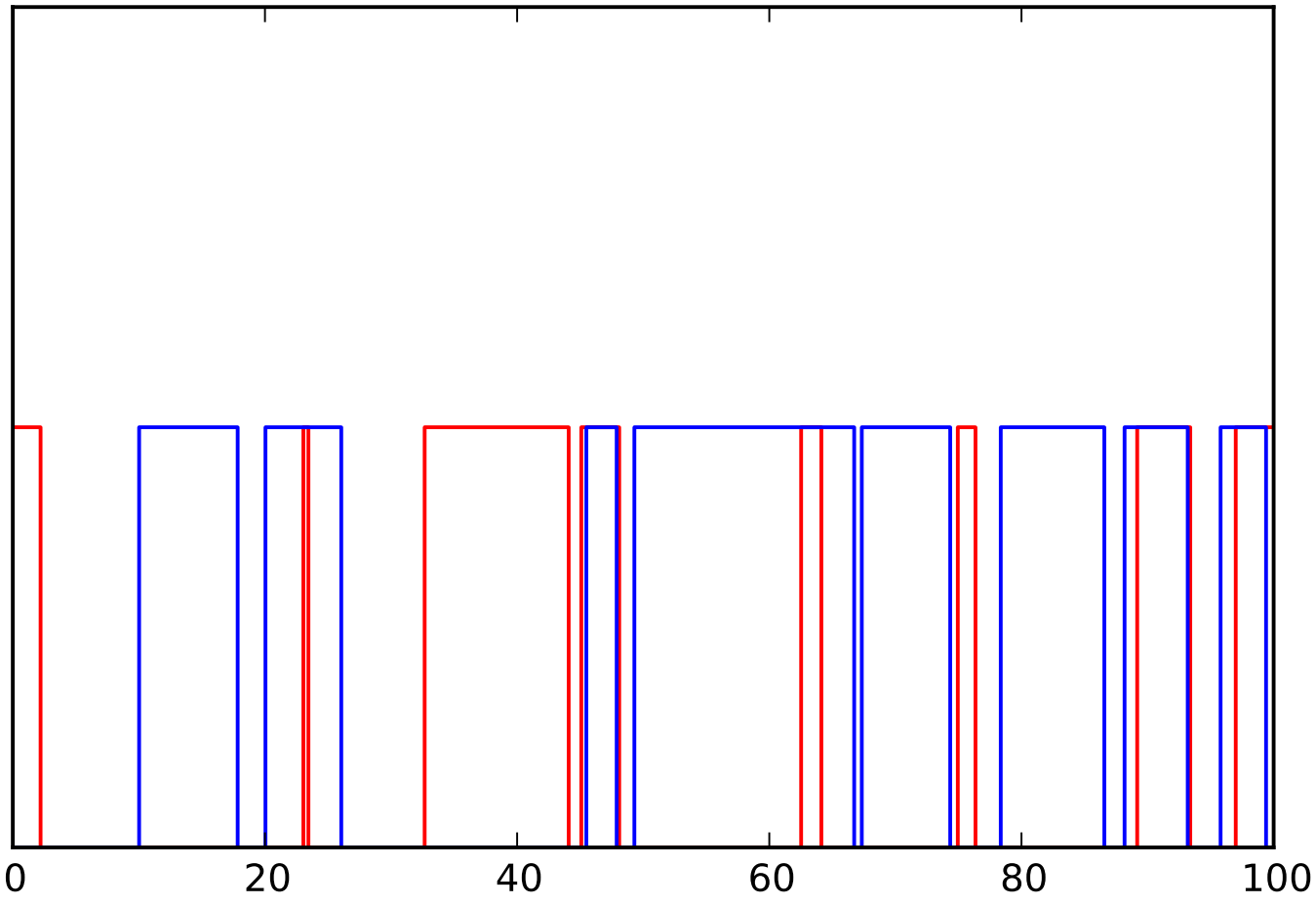
```
G1 = activ1 ↑ + deact1 ↓ + expr1 ⊕;
G2 = activ2 ↑ + deact2 ↓ + expr2 ⊕;

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);
```

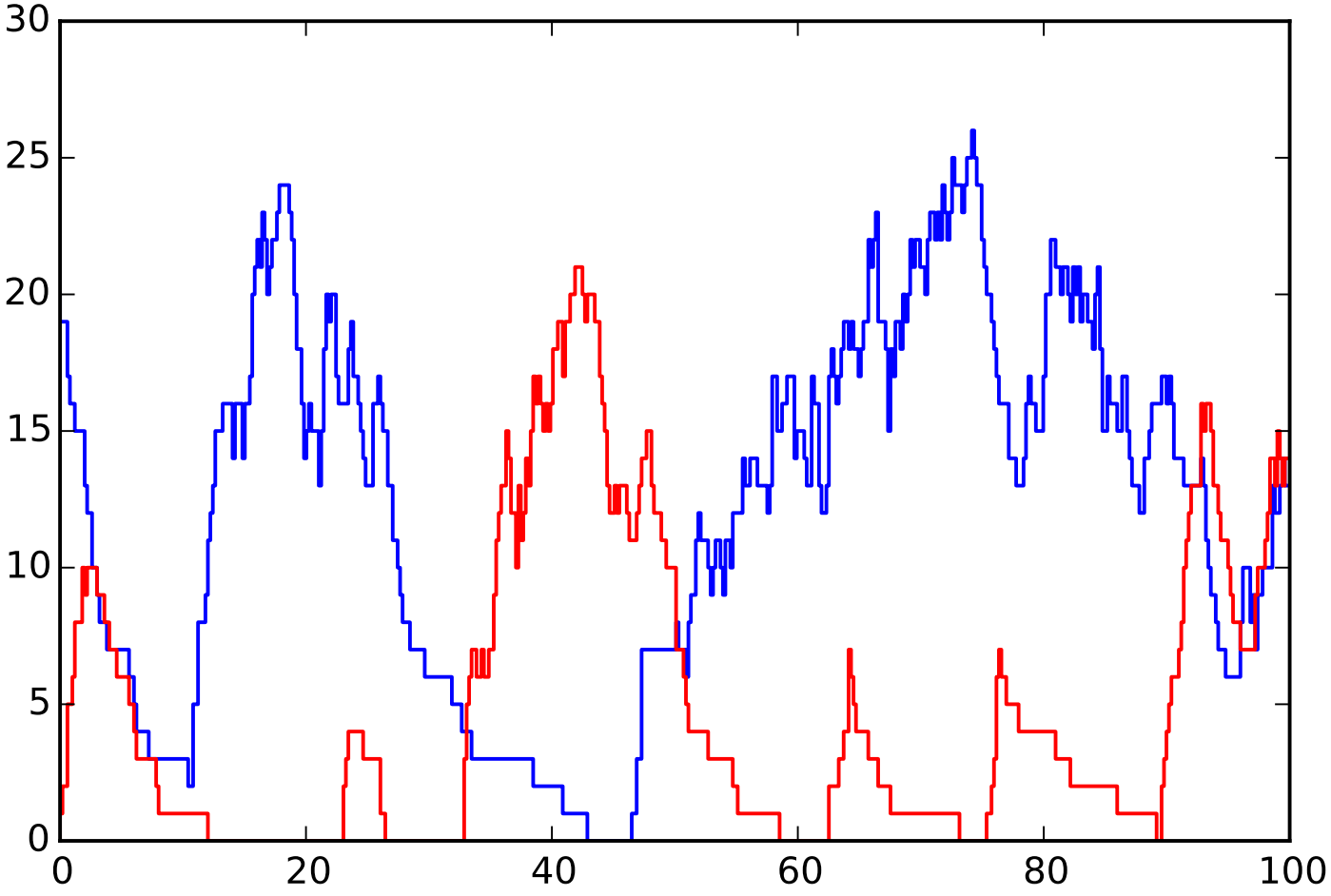
Experiment

- 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

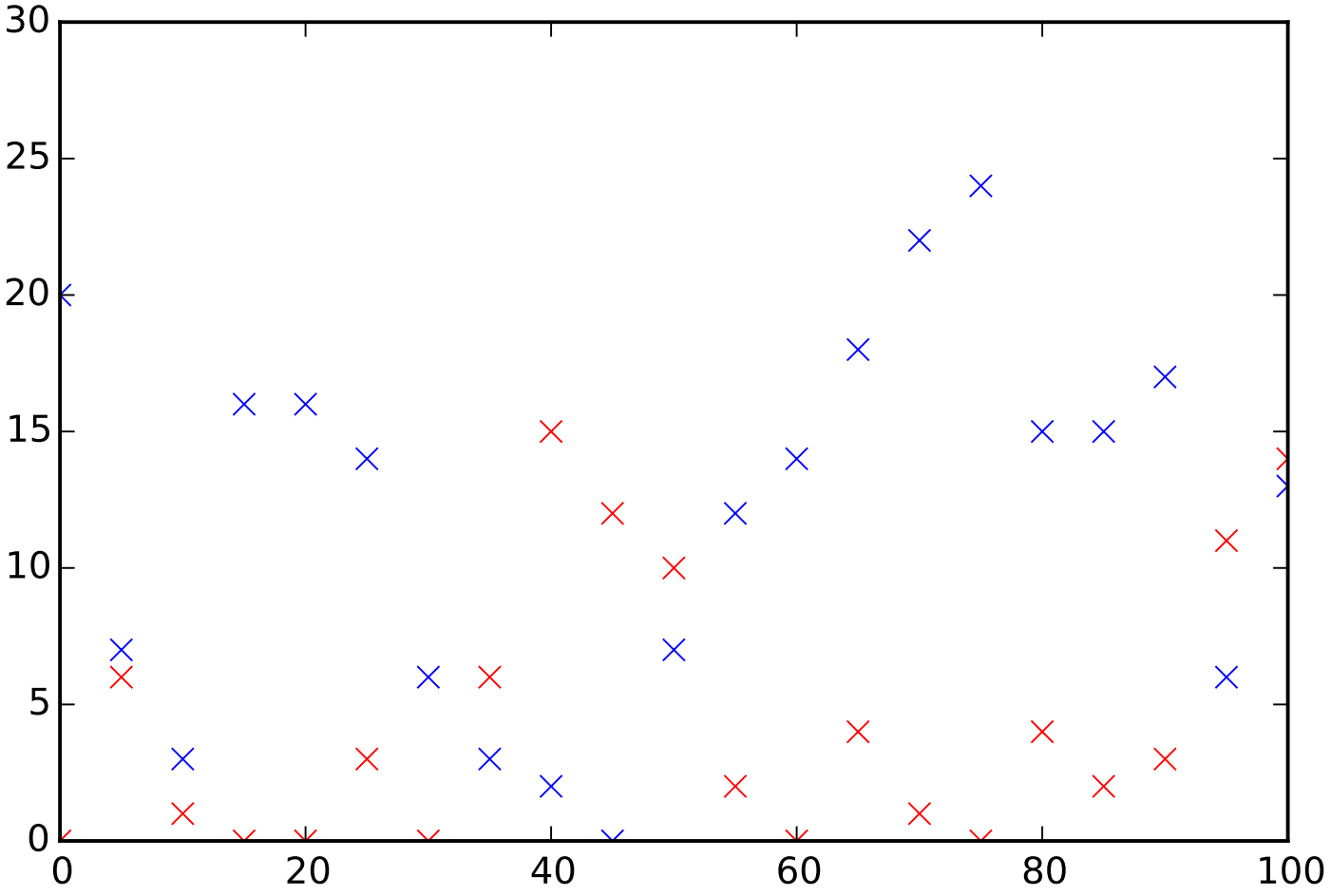
Promoters



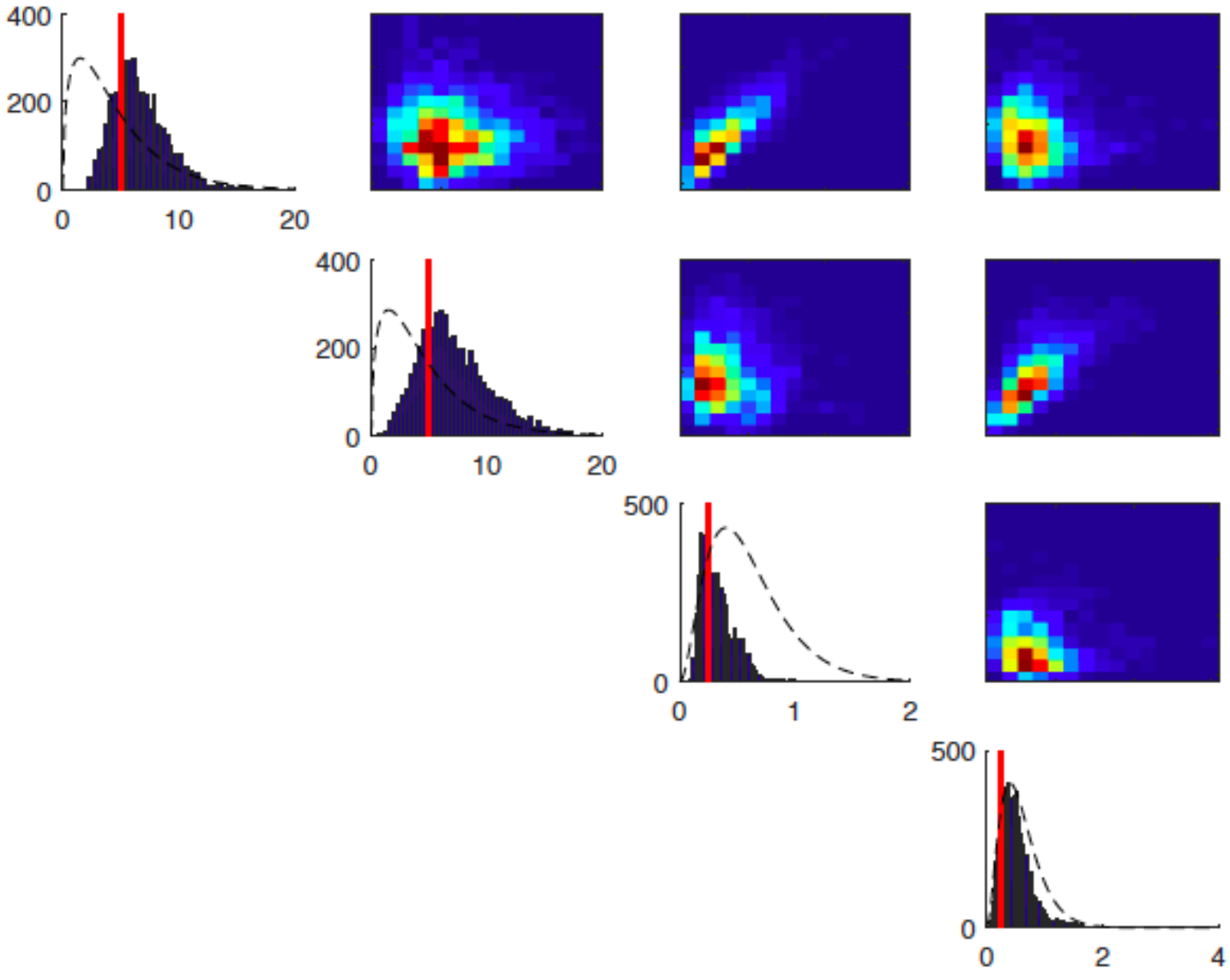
Proteins



Observations used



Results



Outline

1 Stochastic Process Algebras

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

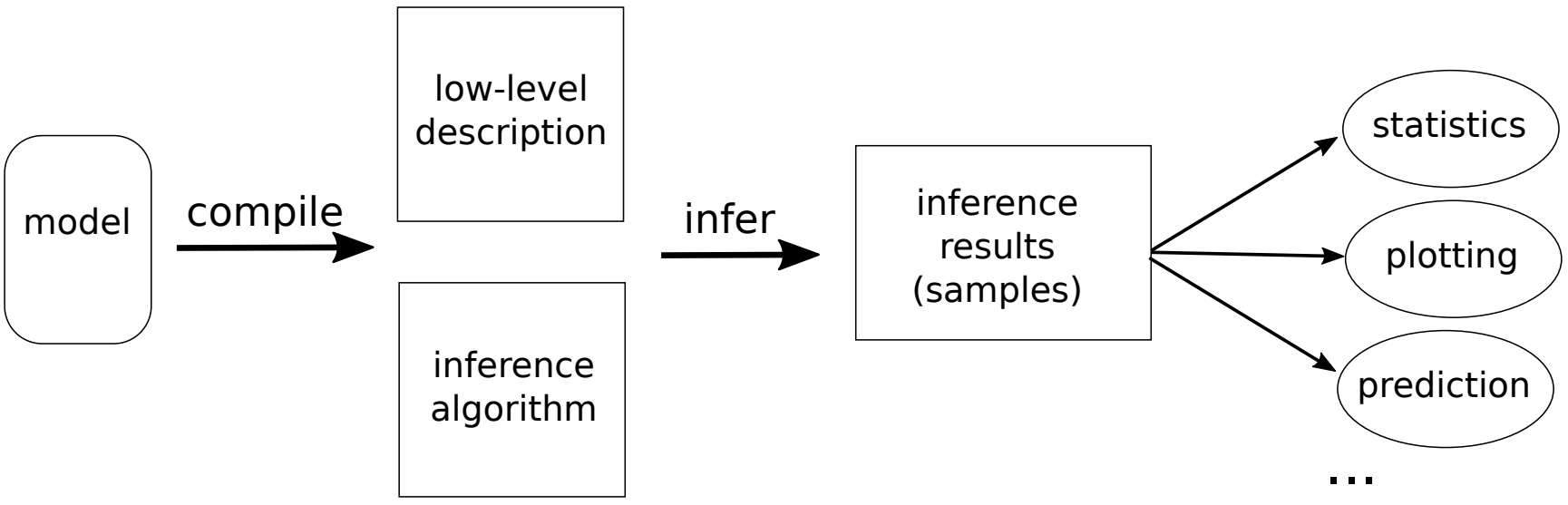
3 ProPPA

4 Inference

5 Results

6 Conclusions

ProPPA Workflow



The ProPPA software framework contains a number of different inference engines, suitable for models with different characteristics, including infinite state space.

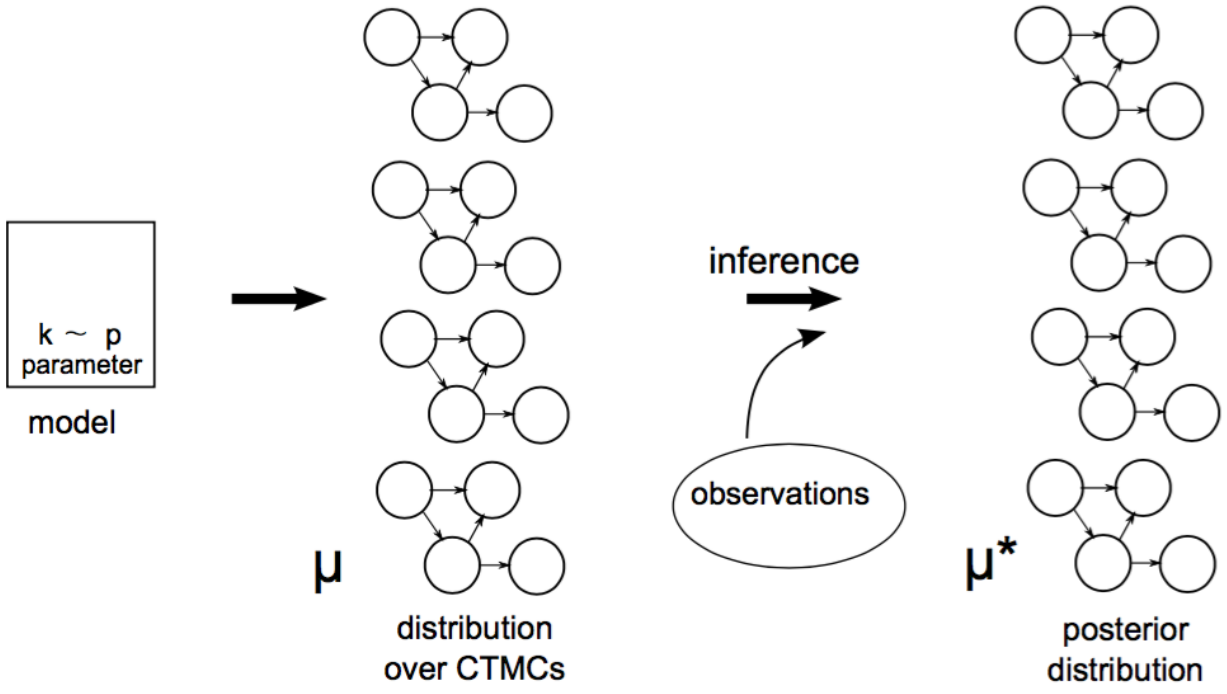
A.Georgoulas, J.Hillston and G.Sanguinetti, *ProPPA: Probabilistic Programming for Stochastic Dynamical Systems*, in TOMACS 2018

Summary

- Integrating CTMCs with process algebras has brought a wide-range of fruitful application domains for stochastic process algebras.
- ProPPA is a process algebra that incorporates uncertainty and observations directly in the model, influenced by probabilistic programming.
- Syntax remains similar to Bio-PEPA whilst the semantics defined in terms of an extension of Constraint Markov Chains.
- Observations can be either time-series or logical properties.
- Embedding inference within a formal language, in the style of probabilistic programming brings many possibilities.

Challenges and Future Directions

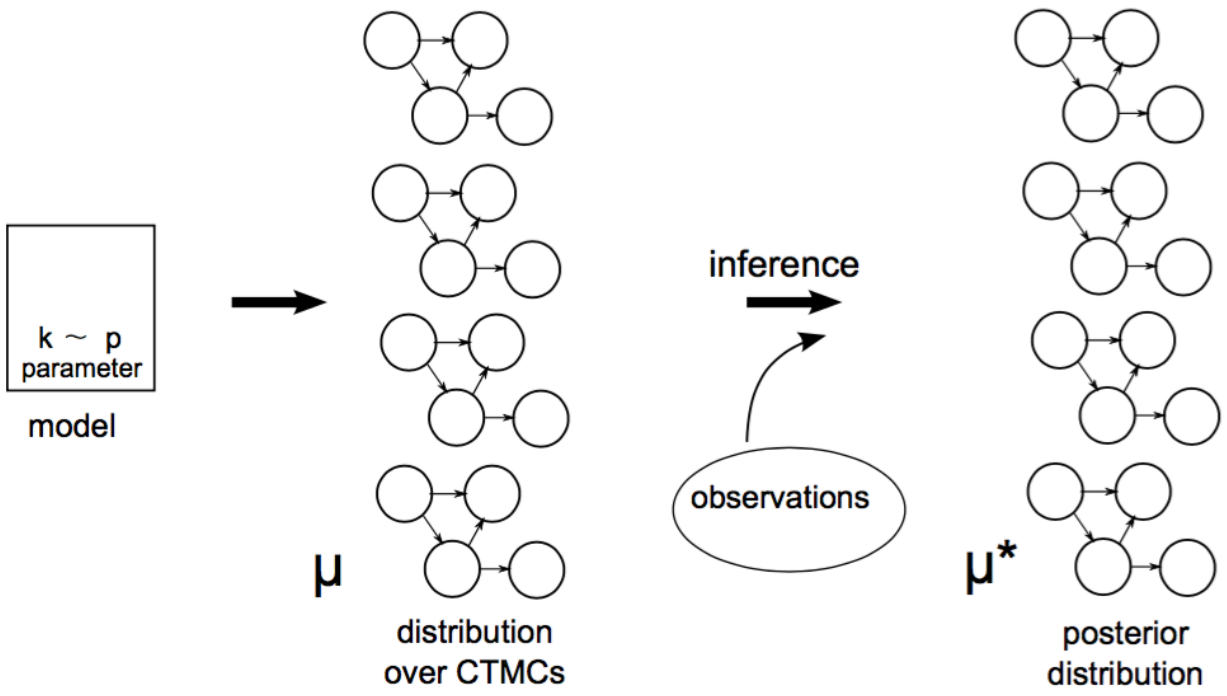
- The value of observations



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 was joint work with Federica Ciocchetta, funded by EPSRC.
- ProPPA was joint work with Anastasis Georgoulas and Guido Sanguinetti, funded by Microsoft Research, ERC and CEC FP7 FET Proactive programme FoCAS.