Embedding machine learning in formal stochastic models of biological processes

Jane Hillston

School of Informatics, University of Edinburgh

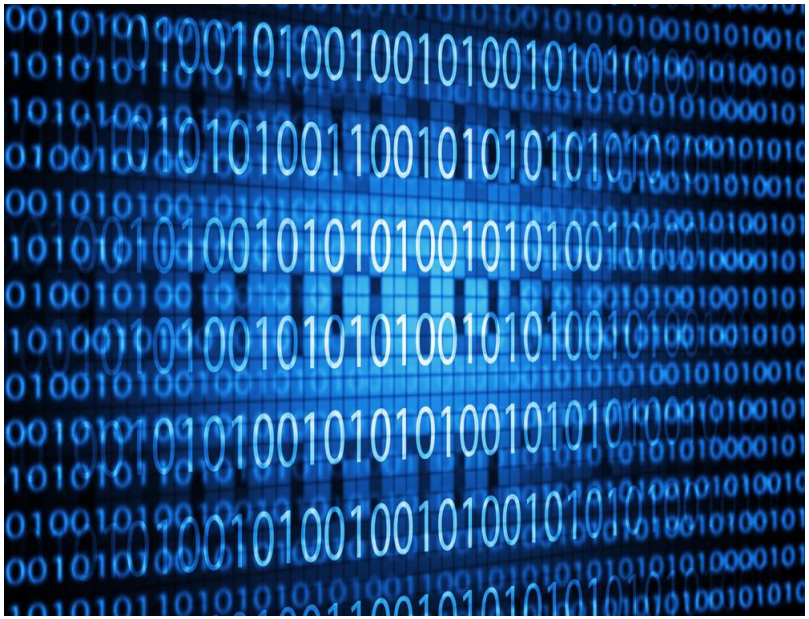
29th October 2014

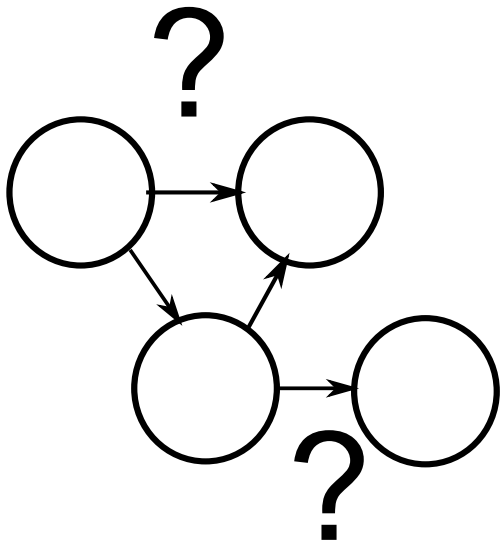
Microsoft Research

quanticol

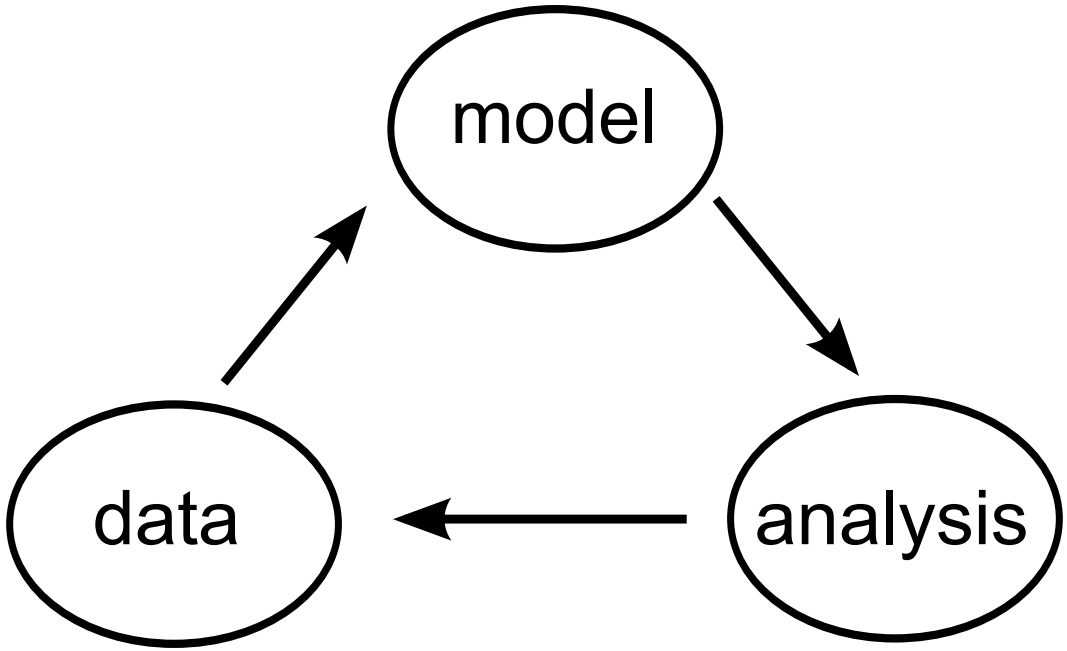
Outline

Outline





Model creation is data-driven



Model creation is data-driven

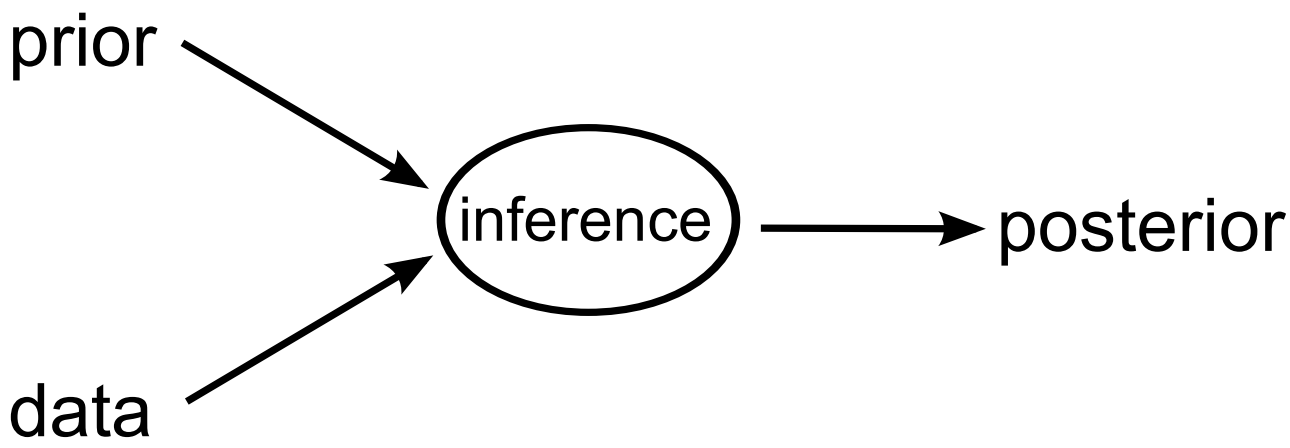
Modelling

There are two approaches to model construction:

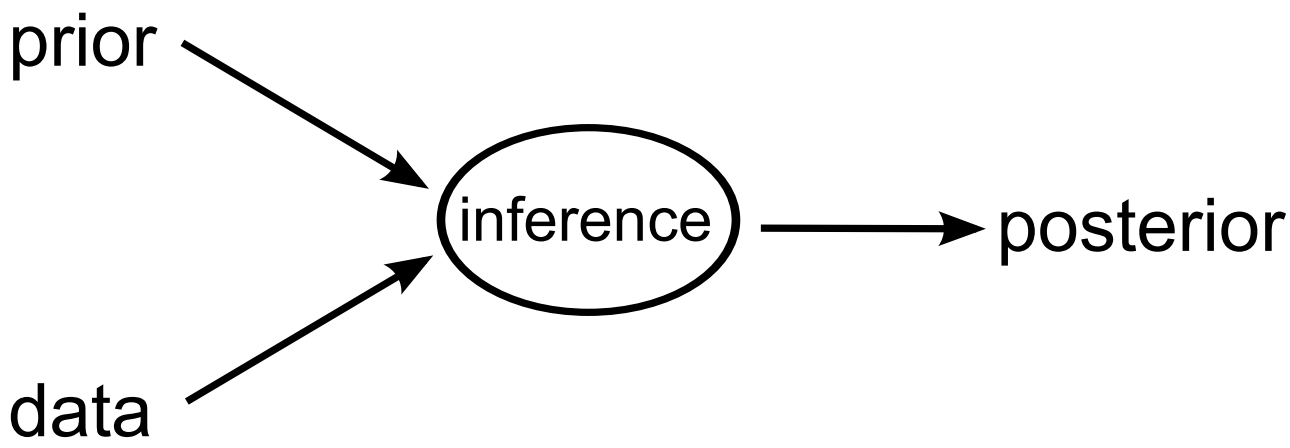
Machine Learning: extracting a model from the data generated by the system, or refining a model based on system behaviour using statistical techniques.

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

Machine Learning



Machine Learning



Bayes' Theorem

For the distribution of a parameter θ and observed data D,

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

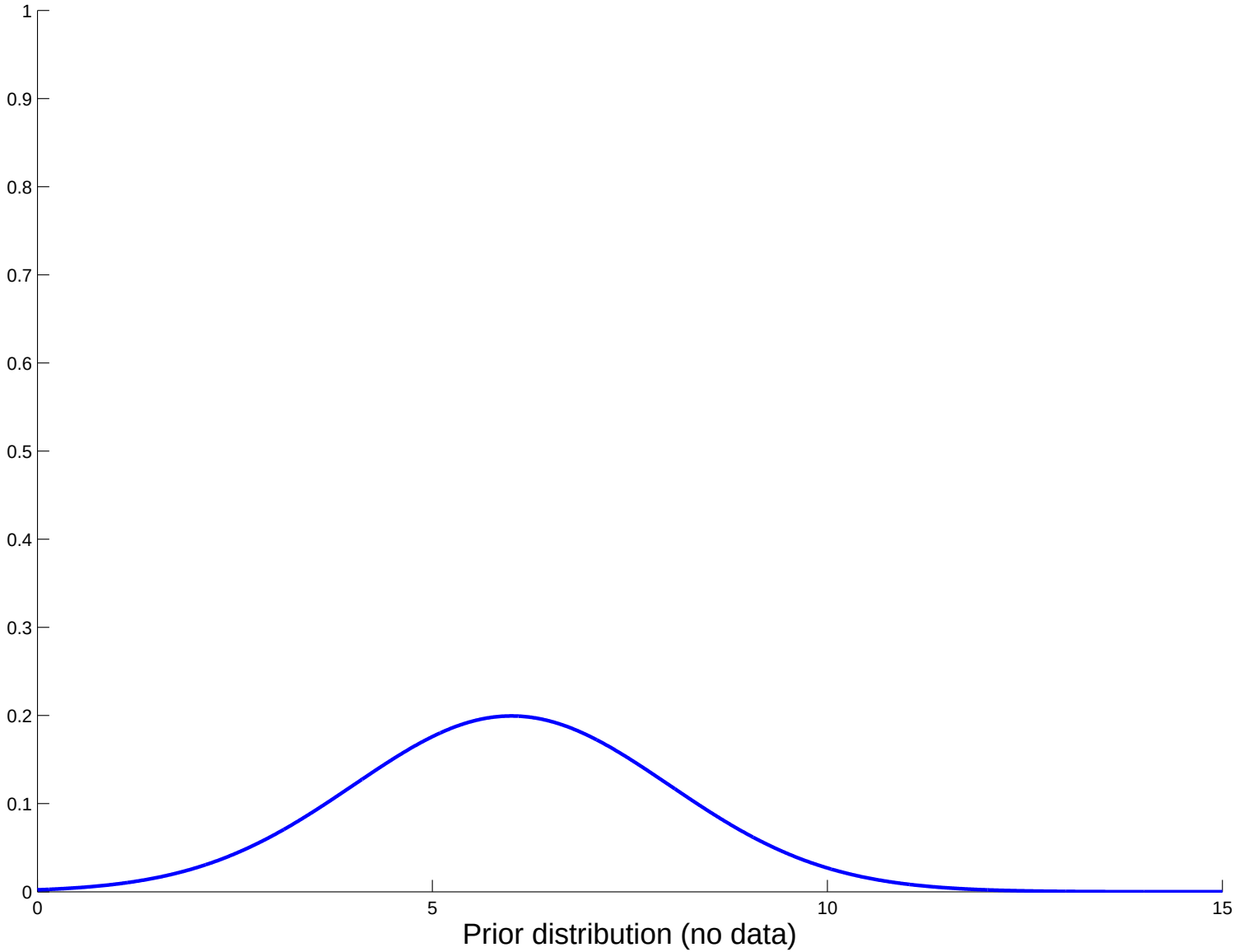
Bayesian statistics

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

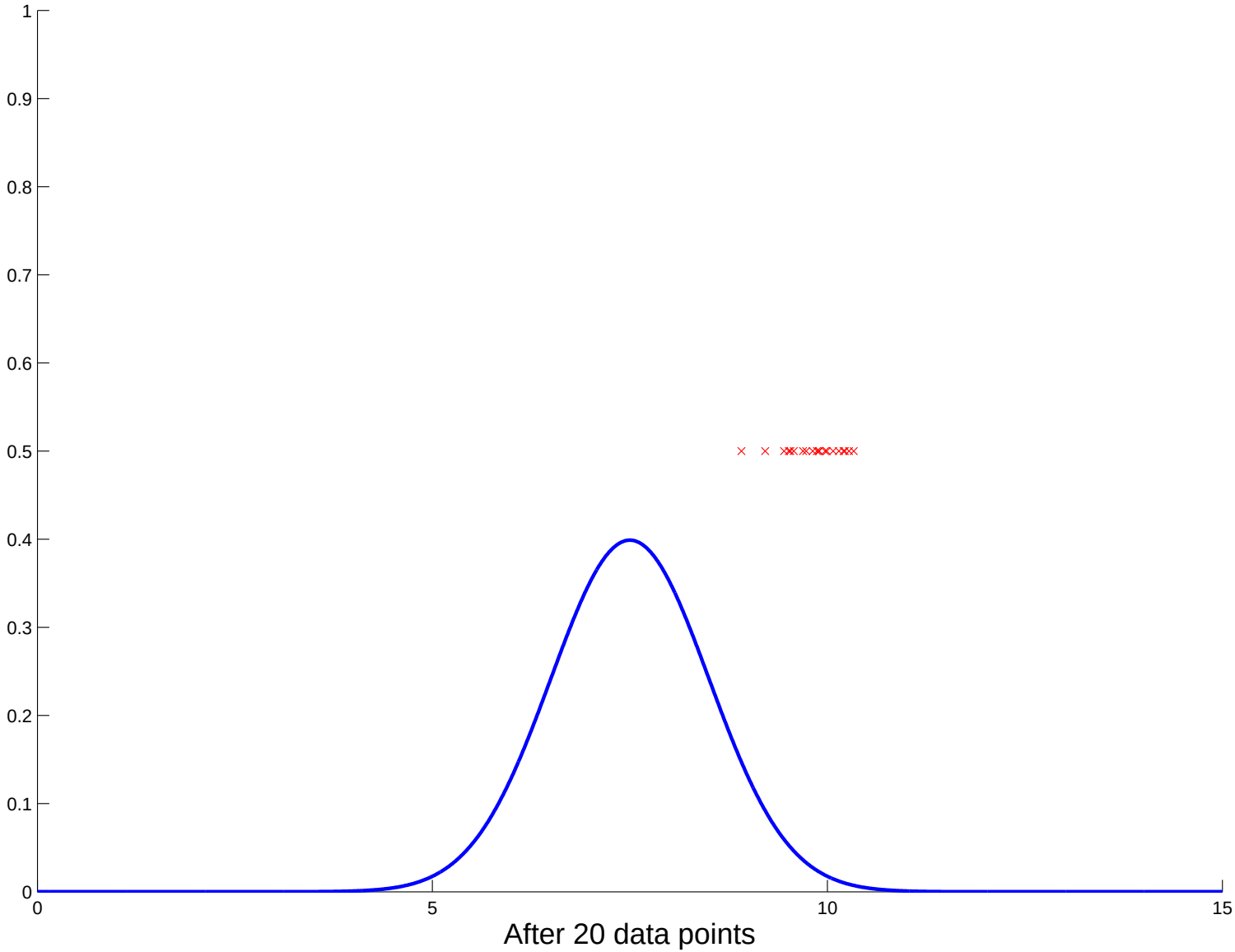
$$P(\theta \mid D) = \frac{P(\theta) \cdot P(D \mid \theta)}{P(D)}$$

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

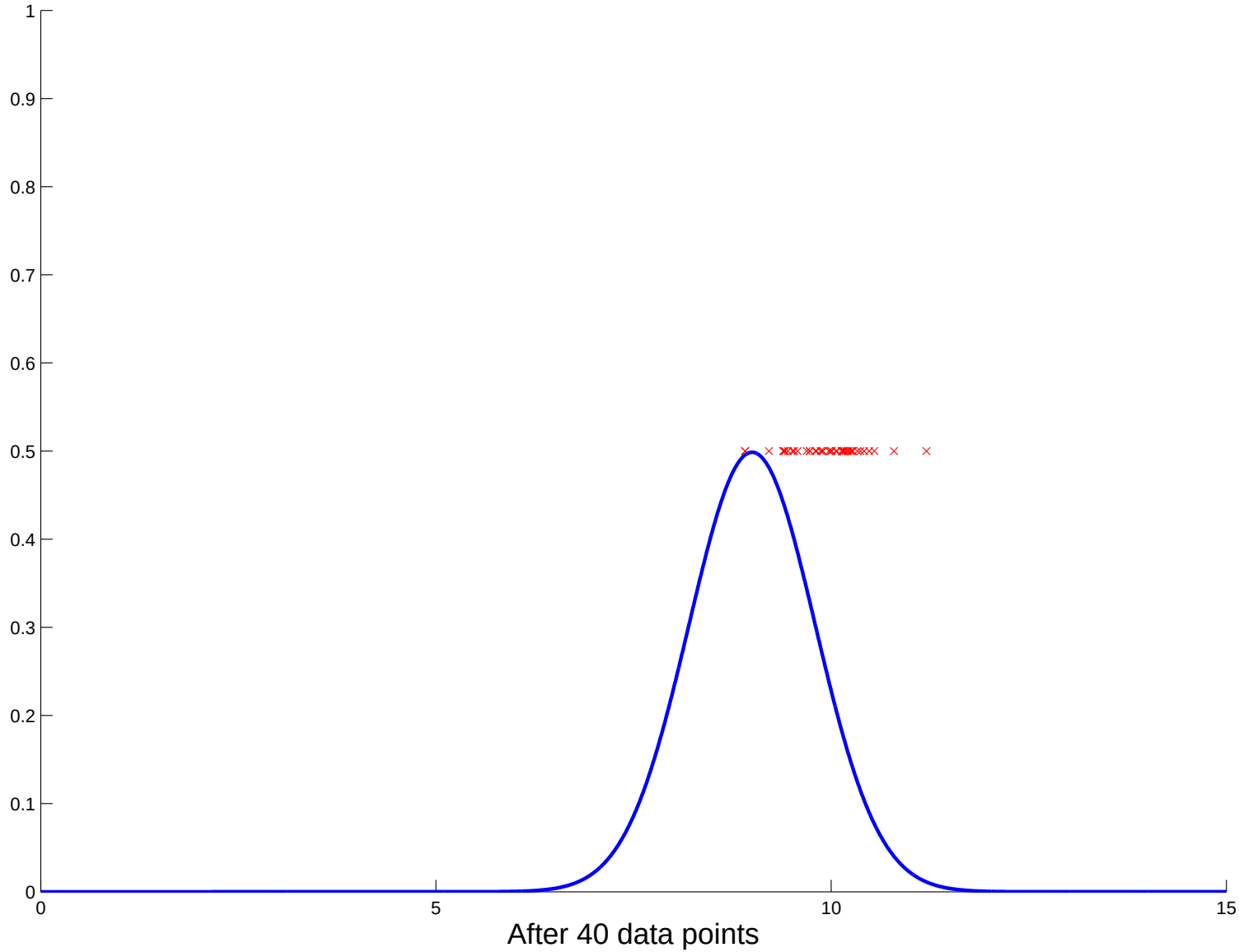
Bayesian statistics



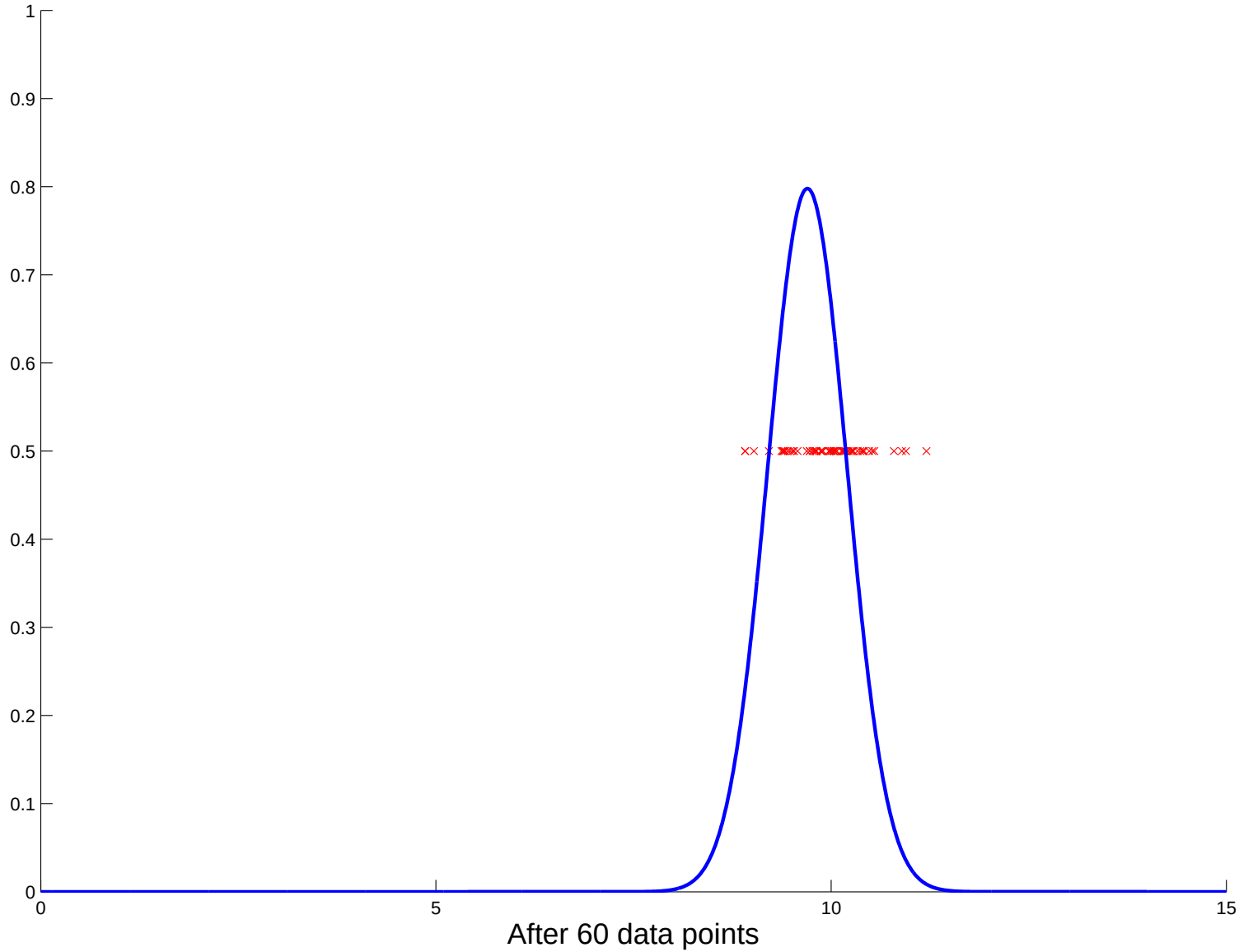
Bayesian statistics



Bayesian statistics



Bayesian statistics



Mechanistic modelling

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

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

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.

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

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[1] which can be run to deepen understanding of the model.

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

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[1] which can be run to deepen understanding of the model.

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

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

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

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

Probabilistic Programming seeks to bring elements of both forms of modelling together.

Outline

Introduction

- Uncertainty is pervasive in biology: noise, parameter uncertainty, stochasticity...
- ...hence the need for probabilistic models and machine learning
- Large models remain intractable and hard to manage
- We need a general way of dealing with them, rather than reimplementing algorithms every time:
- We don't want to tell the compiler what to do, but write in high-level languages; similarly, we want a higher-level framework that does the inference for us

Probabilistic programming

- A way to express 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, ...
- Key actions: specify a distribution, specify observations, infer posterior distribution.

Probabilistic Process Algebra

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?

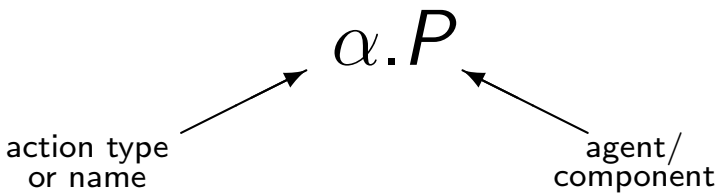
Starting from an existing process algebra (Bio-PEPA), we have developed a new language **ProPPA** that addresses these issues.[2]

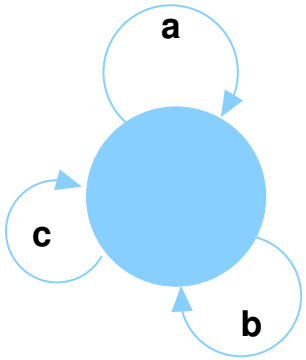
[2] Anastasis Georgoulas, Jane Hillston, Dimitrios Milios, Guido Sanguinetti: *Probabilistic Programming Process Algebra*. QEST 2014: 249-264.

Outline

Process Algebra

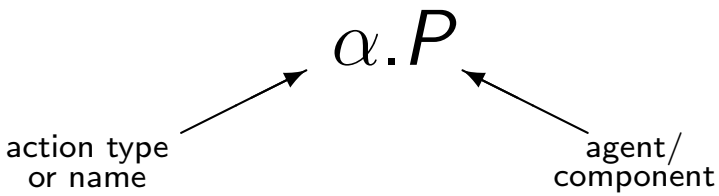
- Models consist of agents which engage in actions.

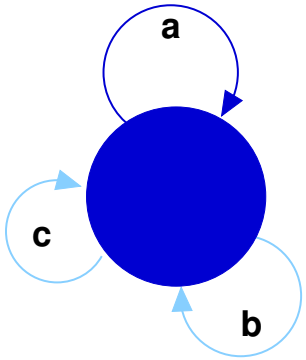




Process Algebra

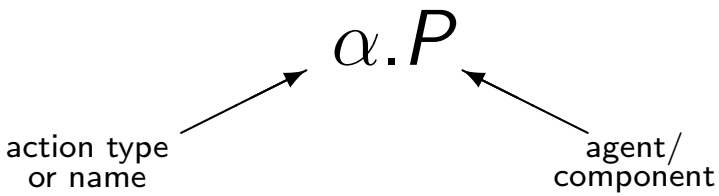
- Models consist of agents which engage in actions.

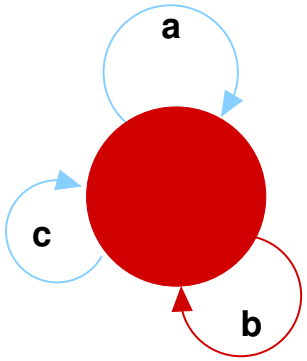




Process Algebra

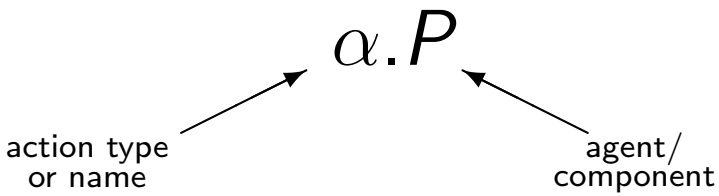
- Models consist of agents which engage in actions.

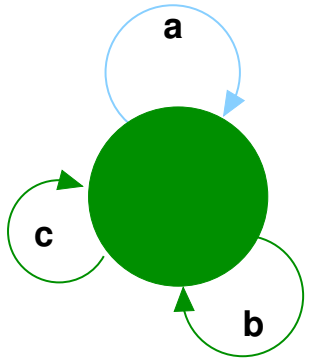




Process Algebra

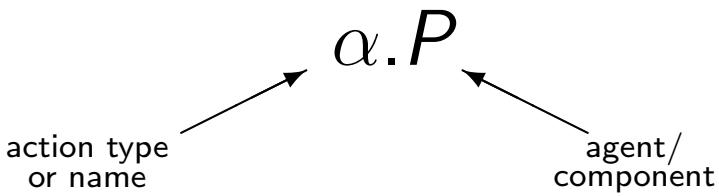
- Models consist of agents which engage in actions.

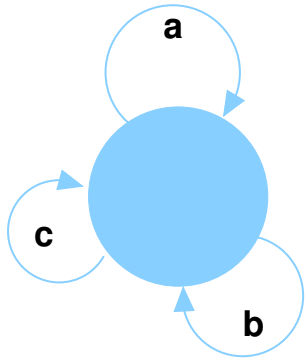




Process Algebra

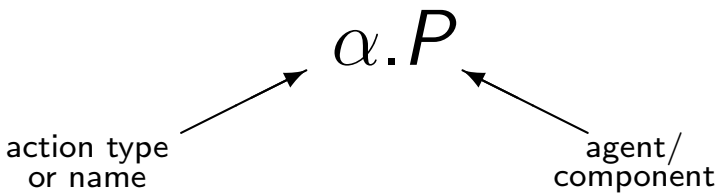
- Models consist of agents which engage in actions.

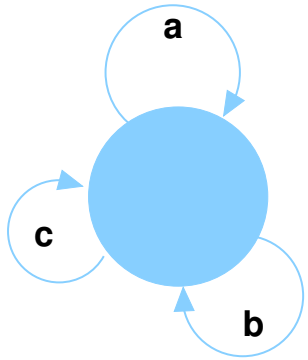




Process Algebra

- Models consist of agents which engage in actions.

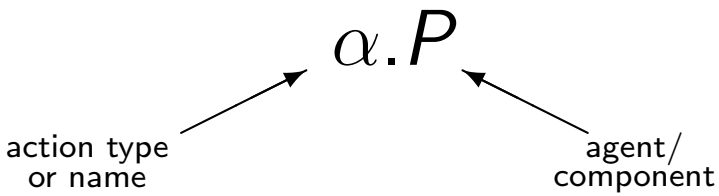


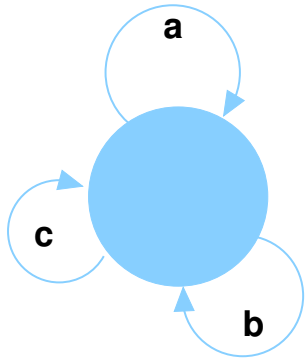


- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process Algebra

- Models consist of agents which engage in actions.



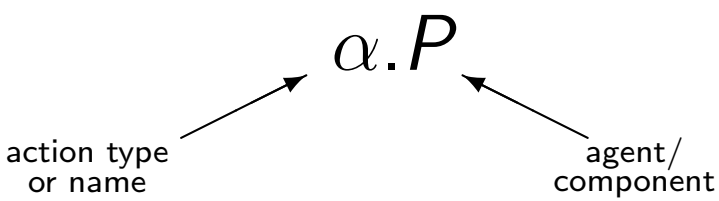


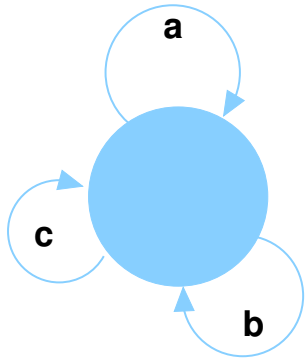
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra
model

Process Algebra

- Models consist of agents which engage in actions.



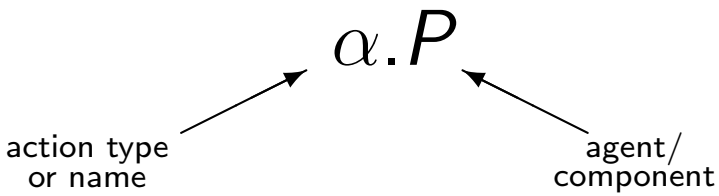


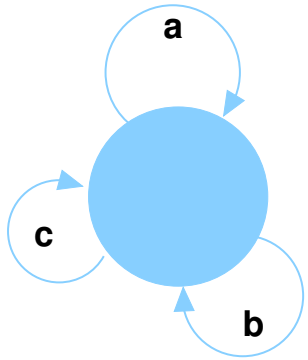
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model $\xrightarrow{\text{SOS rules}}$

Process Algebra

- Models consist of agents which engage in actions.



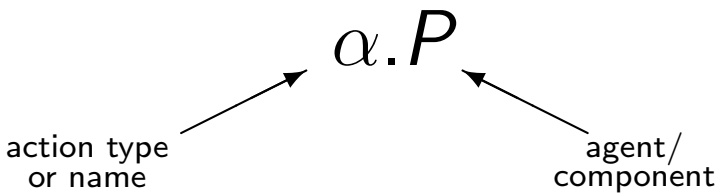


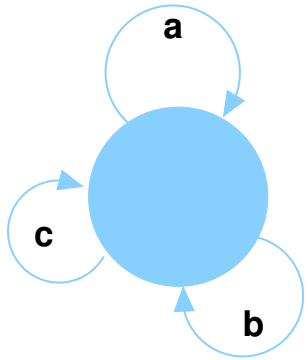
- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model → (SOS rules) → Labelled transition system

Process Algebra

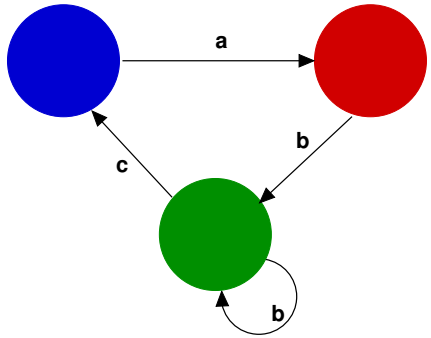
- Models consist of agents which engage in actions.





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

Process algebra model → SOS rules → Labelled transition system



Using Process Algebras in Biology

Process algebras have several attractive features which could be useful for modelling and understanding biological systems:

Using Process Algebras in Biology

Process algebras have several attractive features which could be useful for modelling and understanding biological systems:

- Process algebraic formulations are compositional and make interactions/constraints explicit.

Using Process Algebras in Biology

Process algebras have several attractive features which could be useful for modelling and understanding biological systems:

- Process algebraic formulations are compositional and make interactions/constraints explicit.
- Structure can also be apparent.

Using Process Algebras in Biology

Process algebras have several attractive features which could be useful for modelling and understanding biological systems:

- Process algebraic formulations are compositional and make interactions/constraints explicit.
- Structure can also be apparent.
- Equivalence relations allow formal comparison of high-level descriptions.

Using Process Algebras in Biology

Process algebras have several attractive features which could be useful for modelling and understanding biological systems:

- Process algebraic formulations are compositional and make interactions/constraints explicit.
- Structure can also be apparent.
- Equivalence relations allow formal comparison of high-level descriptions.
- There are well-established techniques for reasoning about the behaviours and properties of models, supported by software. These include qualitative and quantitative analysis, and model checking.

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.

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.

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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

SPA
MODEL

Stochastic Process Algebra

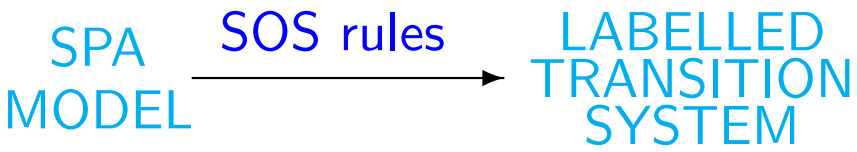
In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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



Stochastic Process Algebra

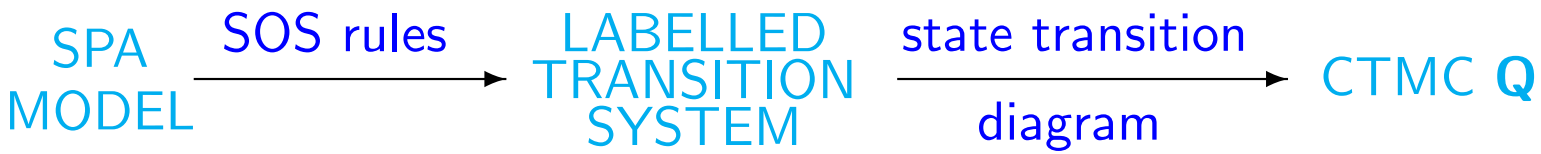
In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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

Stochastic Process Algebra

In a stochastic process algebra actions (reactions) not only have a name or type, but also a stochastic duration or rate.

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) not only have a name or type, but also a stochastic duration or rate.

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.

State representation

- The state of the system at any time consists of the local states of each of its sequential/species components.

State representation

- 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.

State representation

- 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.

State representation

- 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)

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

The syntax

Sequential component (species component)

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

The syntax

Sequential component (species component)

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

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)

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

The syntax

Sequential component (species component)

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

Model component

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

The syntax

Sequential component (species component)

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

Model component

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

The syntax

Sequential component (species component)

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

Model component

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

The syntax

Sequential component (species component)

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

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)

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

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 list $\mathcal{N}$ contains the numbers of levels/maximum concentrations

The abstraction

- Each species i is described by a species component C_i

The abstraction

- Each species 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 abstraction

- Each species 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}
- Compartments or locations are considered static and not represented explicitly.

The abstraction

- Each species 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}
- Compartments or locations are considered static and not represented explicitly.

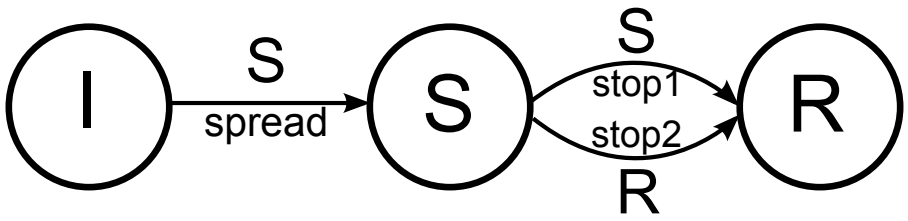
The species components are then composed together to describe the behaviour of the system.

The semantics

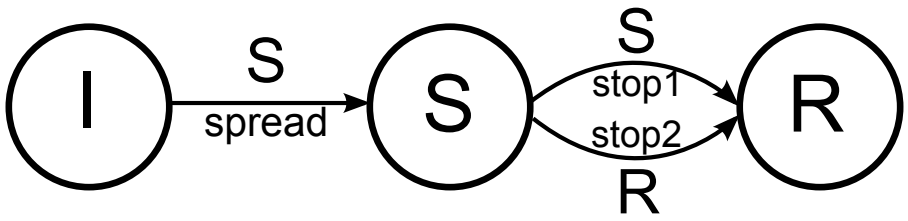
The semantics is defined by two transition relations:

- 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]
```

Outline

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

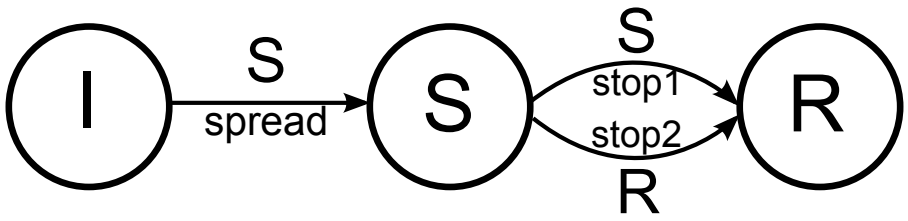
The objective of ProPPA is to retain the features of the stochastic process algebra:

- 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
- accss 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 = Gaussian(0,1);`

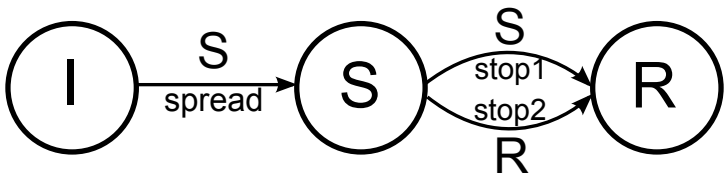
Providing observations:

- `observe('trace')`

Specifying inference approach:

- `infer('ABC')`

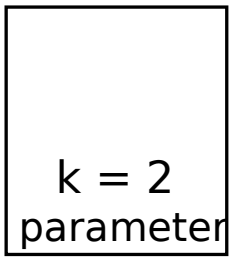
Additions



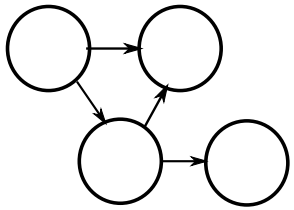

```
k_s = Uniform(0,1);
k_r = Uniform(0,1);

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

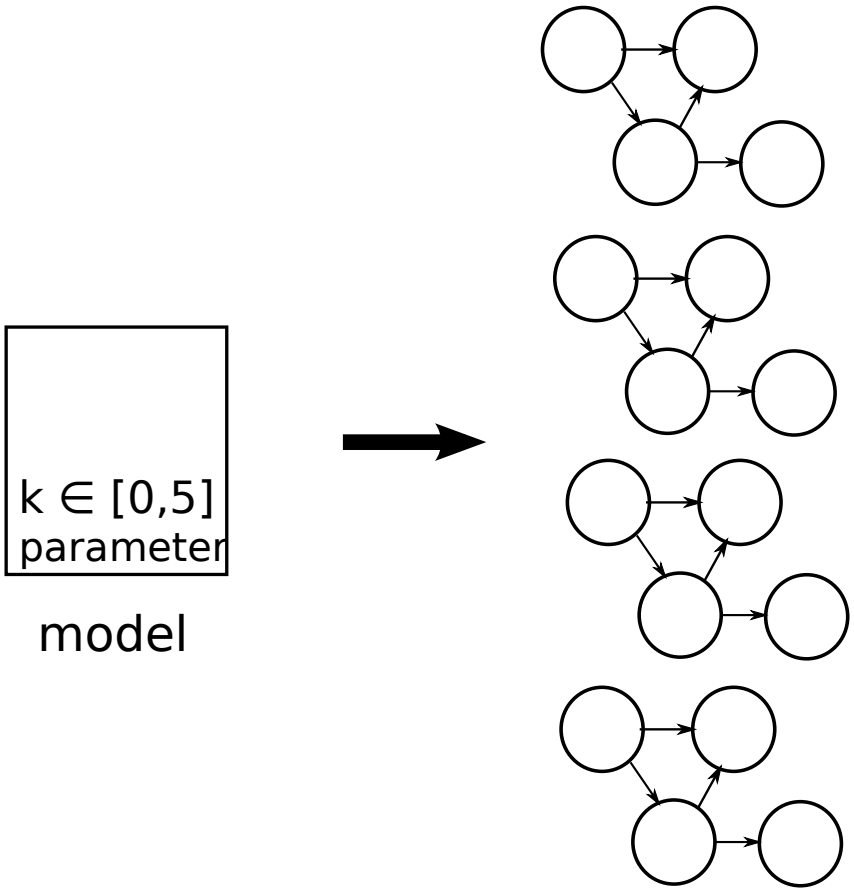
- A Bio-PEPA model can be interpreted as a CTMC; however, CTMCs cannot capture uncertainty in the rates (every transition must have a concrete rate).
- ProPPA models include uncertainty in the parameters, which translates into uncertainty in the transition rates.
- A ProPPA model should be mapped to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

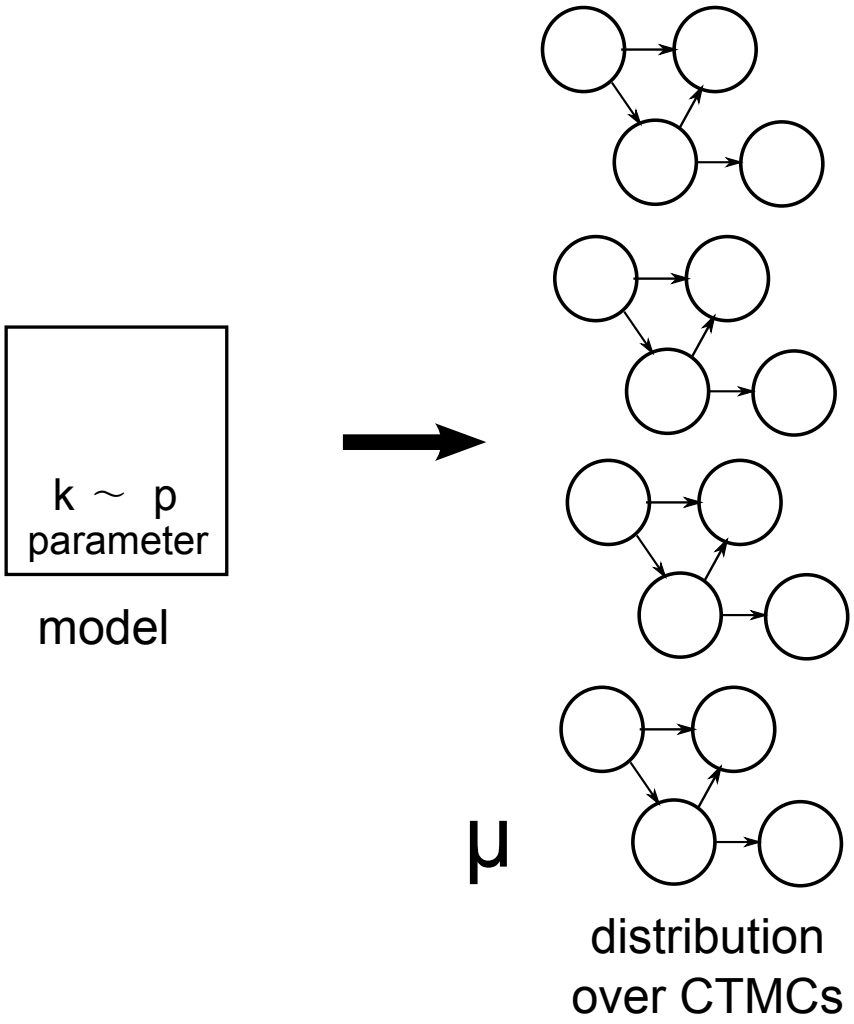


model



CTMC


k ∈ [0,5]
parameter
model
set
of CTMCs


k ~ p
parameter
model
μ
distribution
over CTMCs

Constraint Markov Chains

Constraint Markov Chains[3] (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.

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

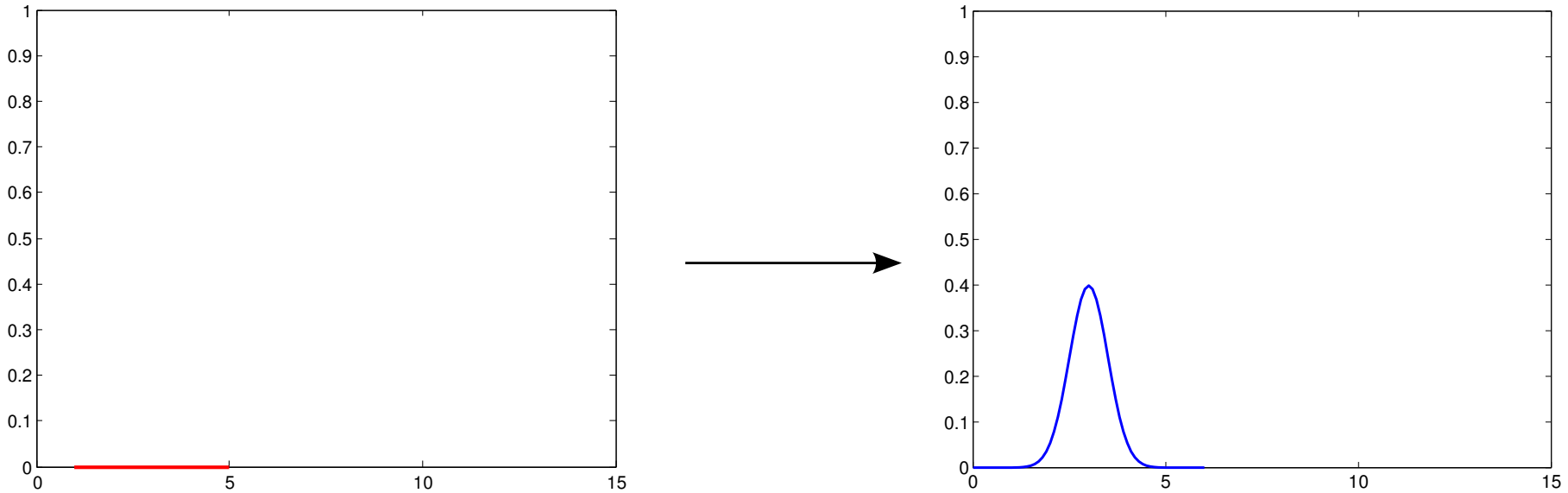
Constraint Markov Chains

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 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

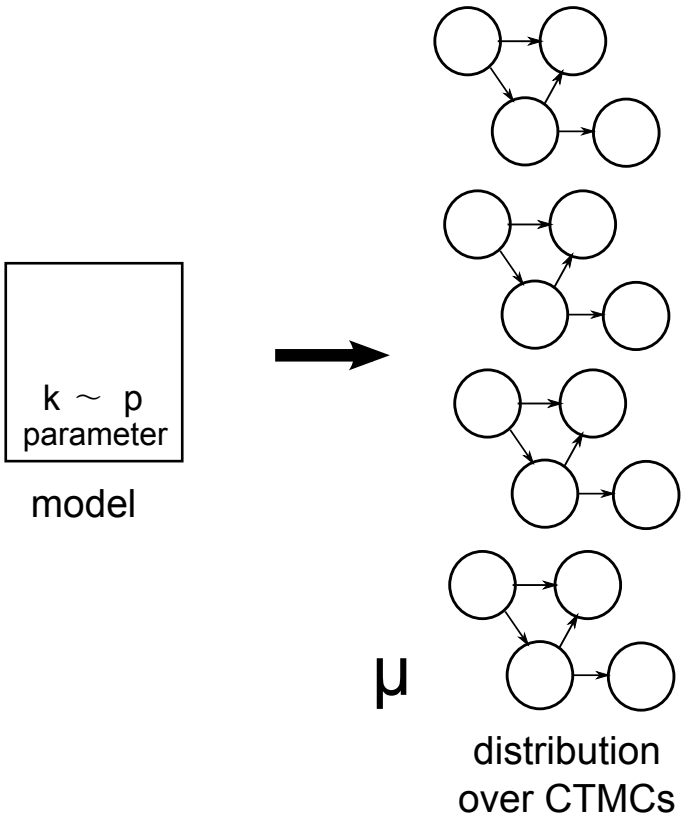
The distribution over the parameter values induces a distribution over transition rates.

Rules are expressed as state-to-function transition systems (FuTS[4]).

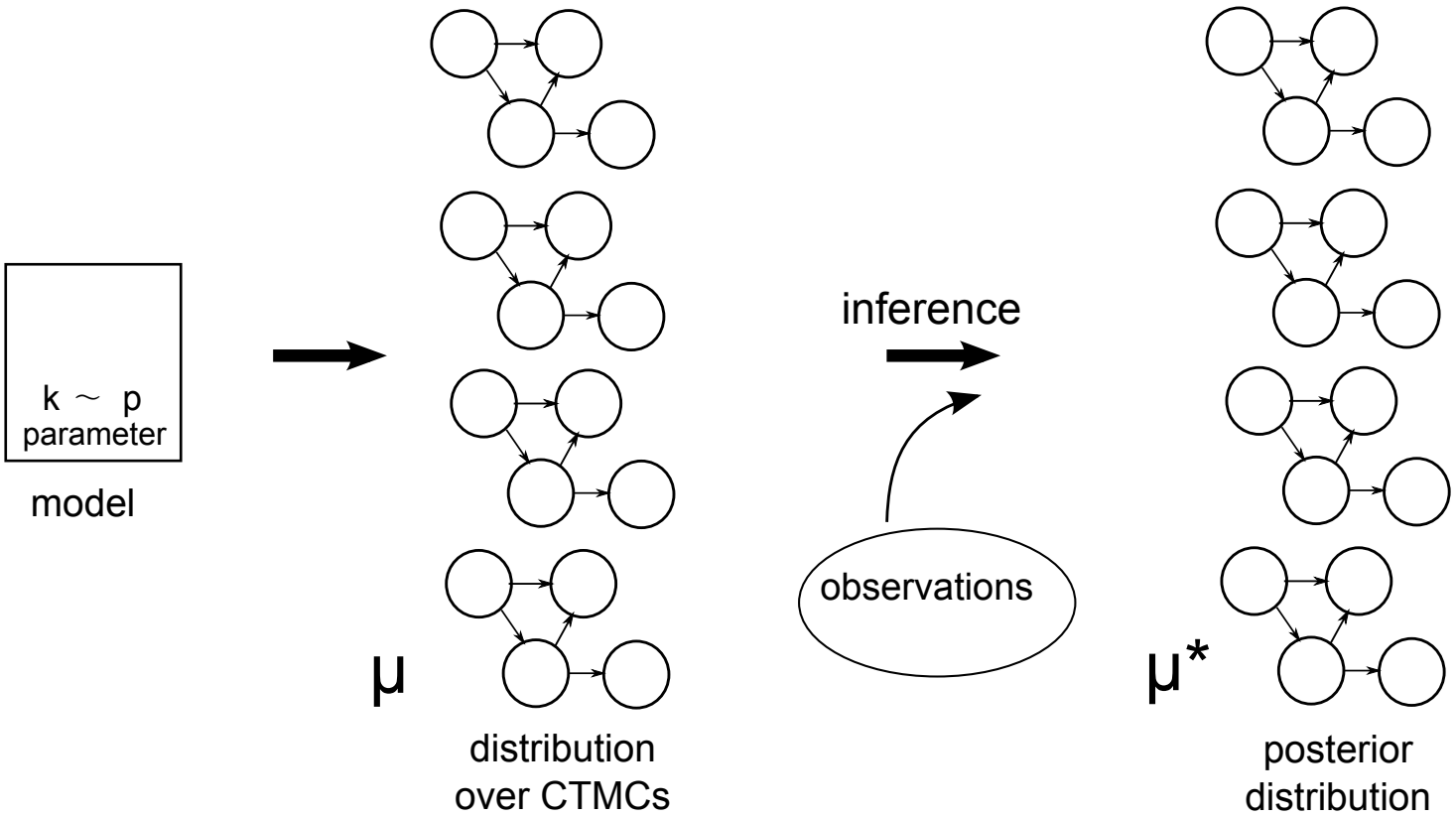
[4] De Nicola *et al.*, *A Uniform Definition of Stochastic Process Calculi*, ACM Computing Surveys, 2013

Outline

Inference



Inference


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

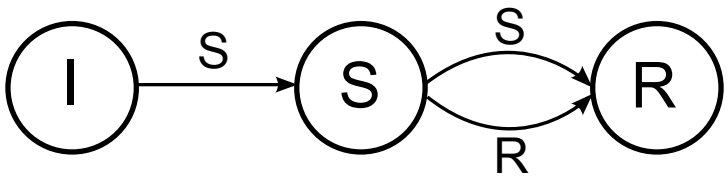
Inference

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

- 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.

Outline

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.

Method 1:

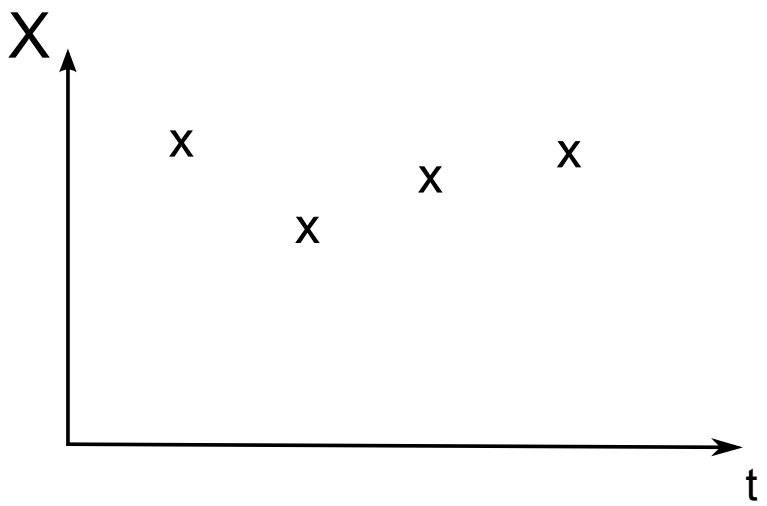
- Approximate Bayesian Computation
- Returns posterior as a set of points (samples)
- Observations: time-series (single simulation)

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.

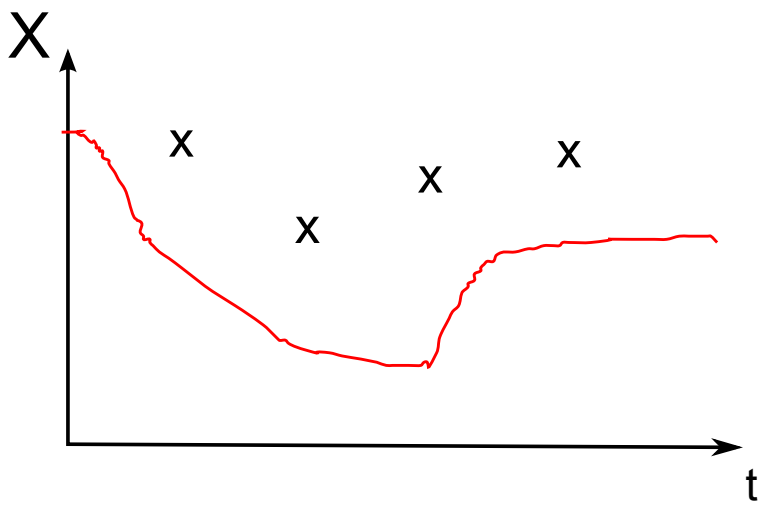
Inference

- 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.



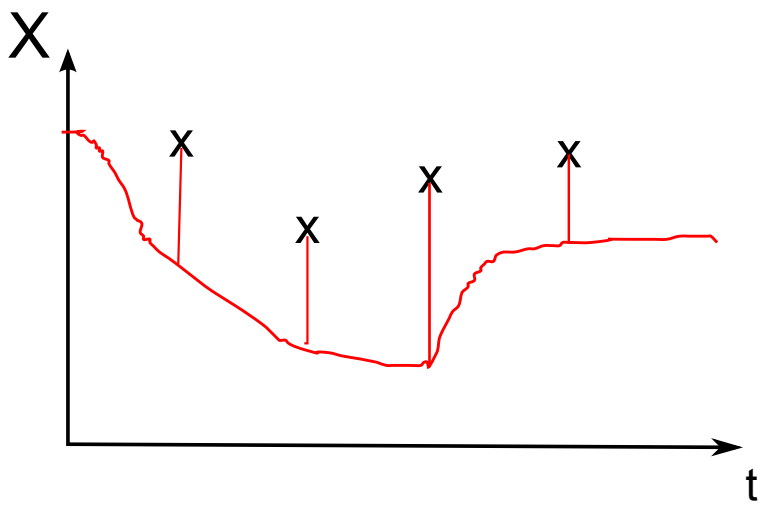
Inference

- 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.



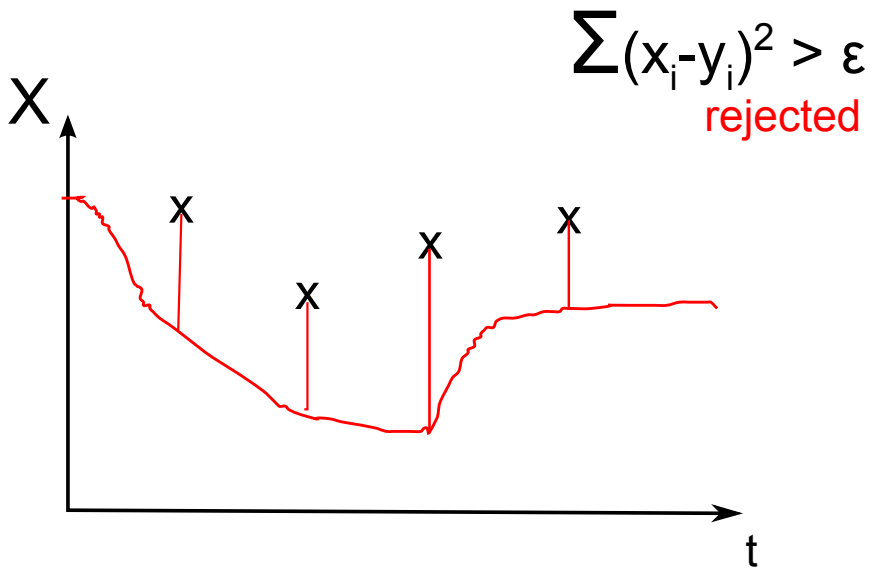
Inference

- 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.



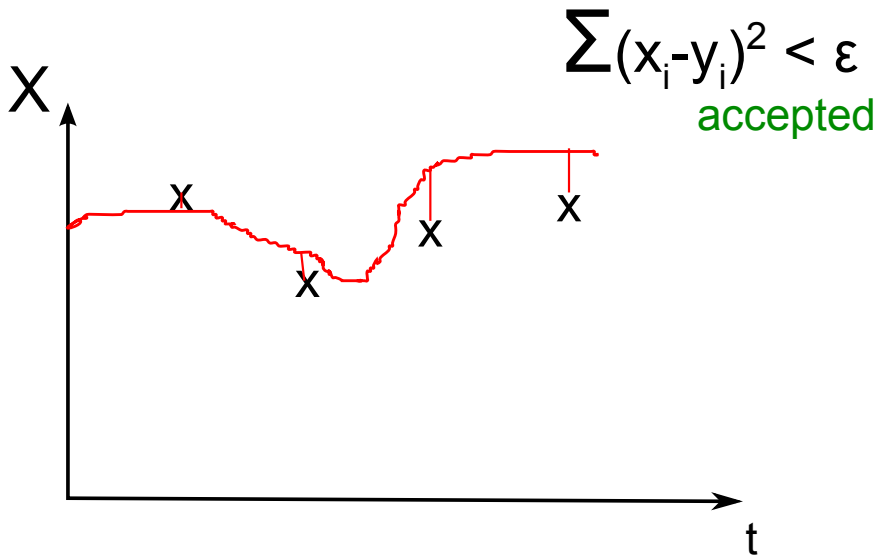
Inference

- 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.



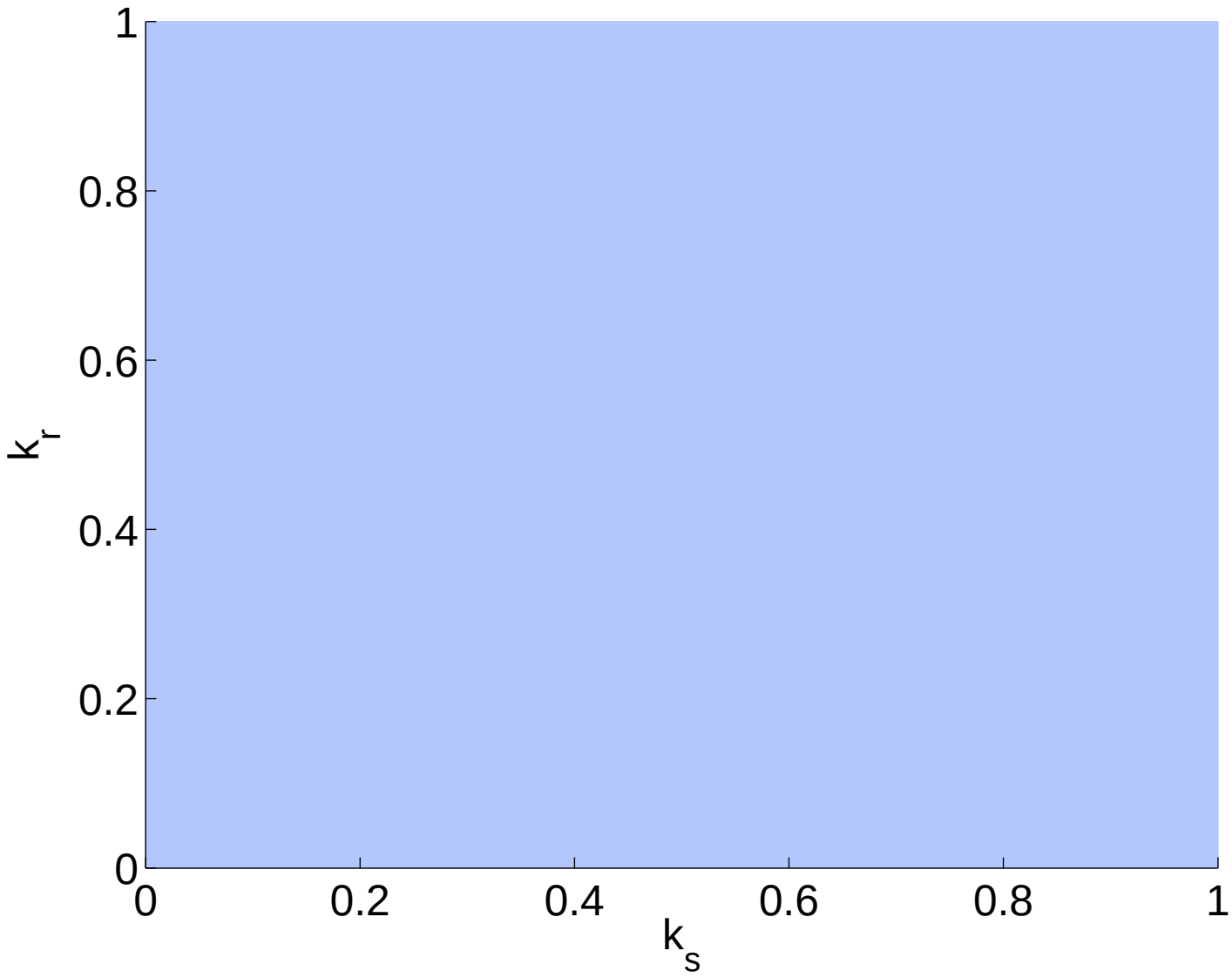
Approximate Bayesian Computation

ABC algorithm

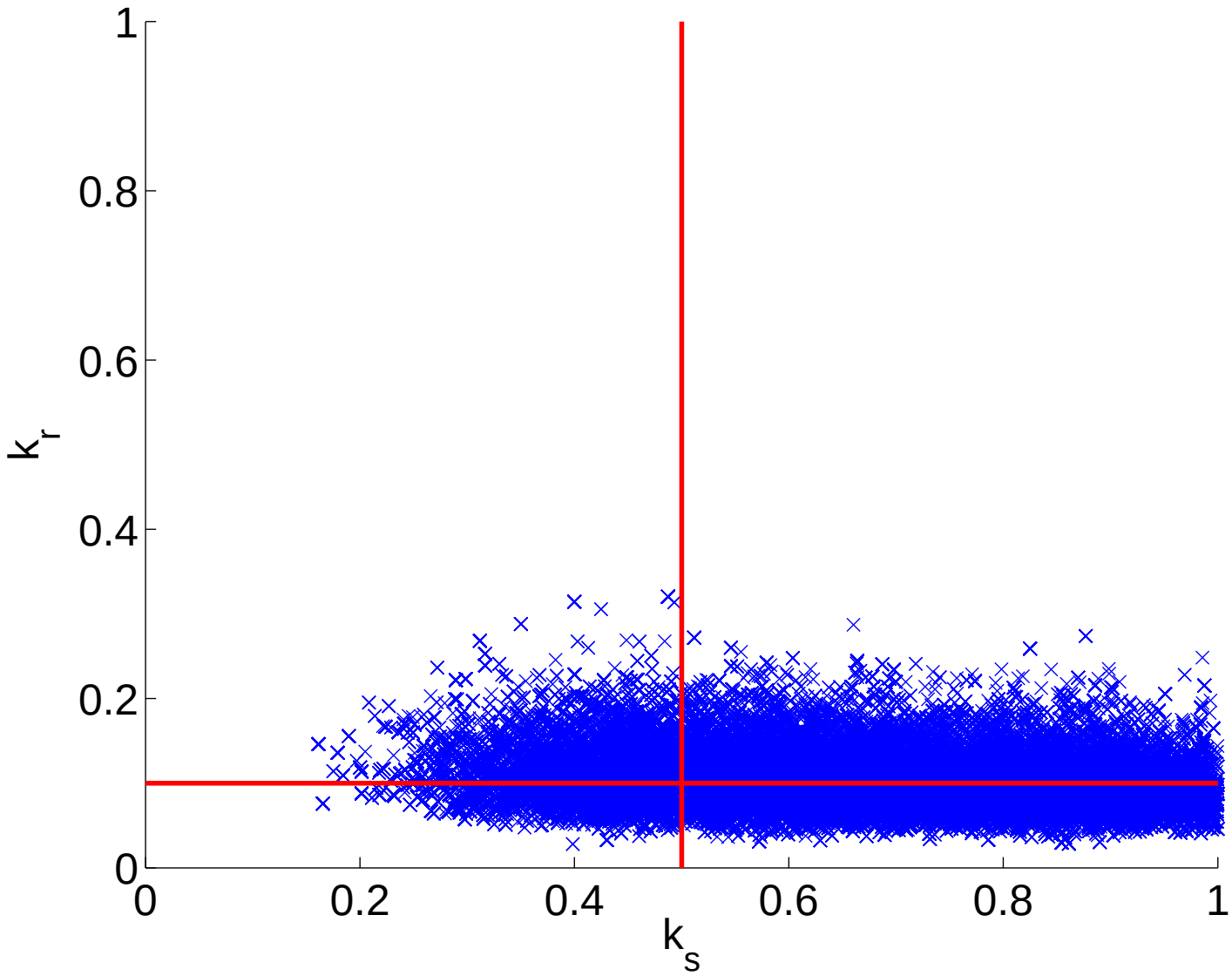
1. Sample a parameter set from the prior distribution.
2. Simulate the system using these parameters.
3. Compare the simulation trace obtained with the observations.
4. If distance $< \epsilon$, accept, otherwise reject.

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

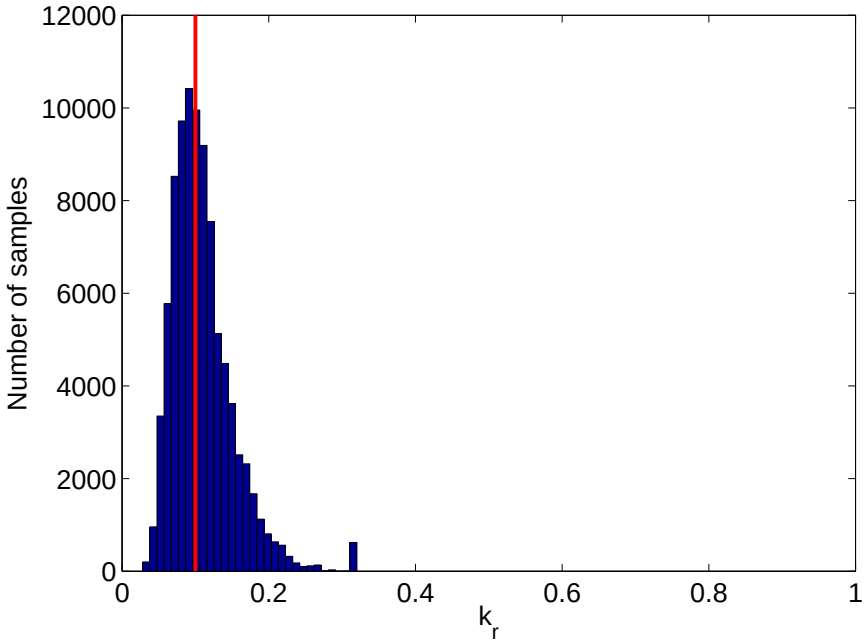
Results: ABC

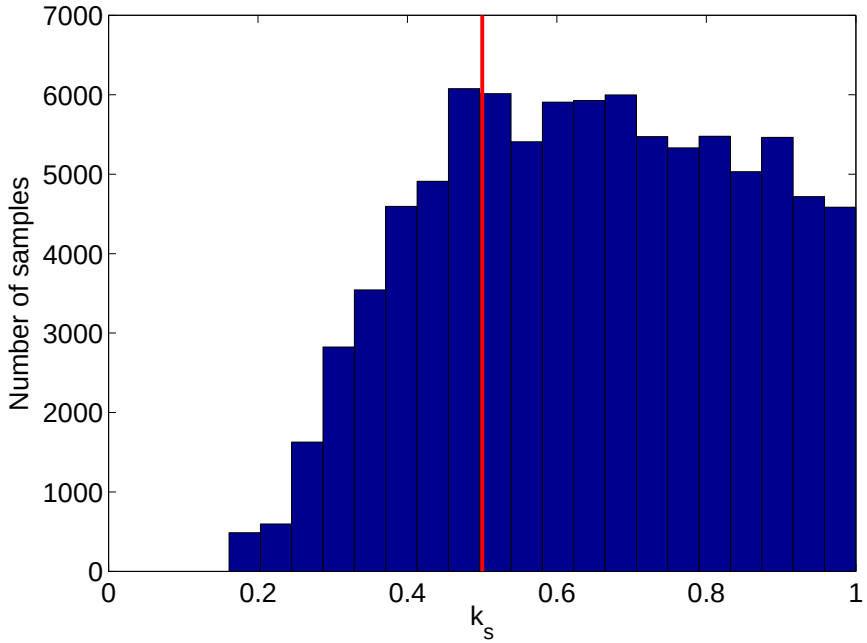


Results: ABC



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

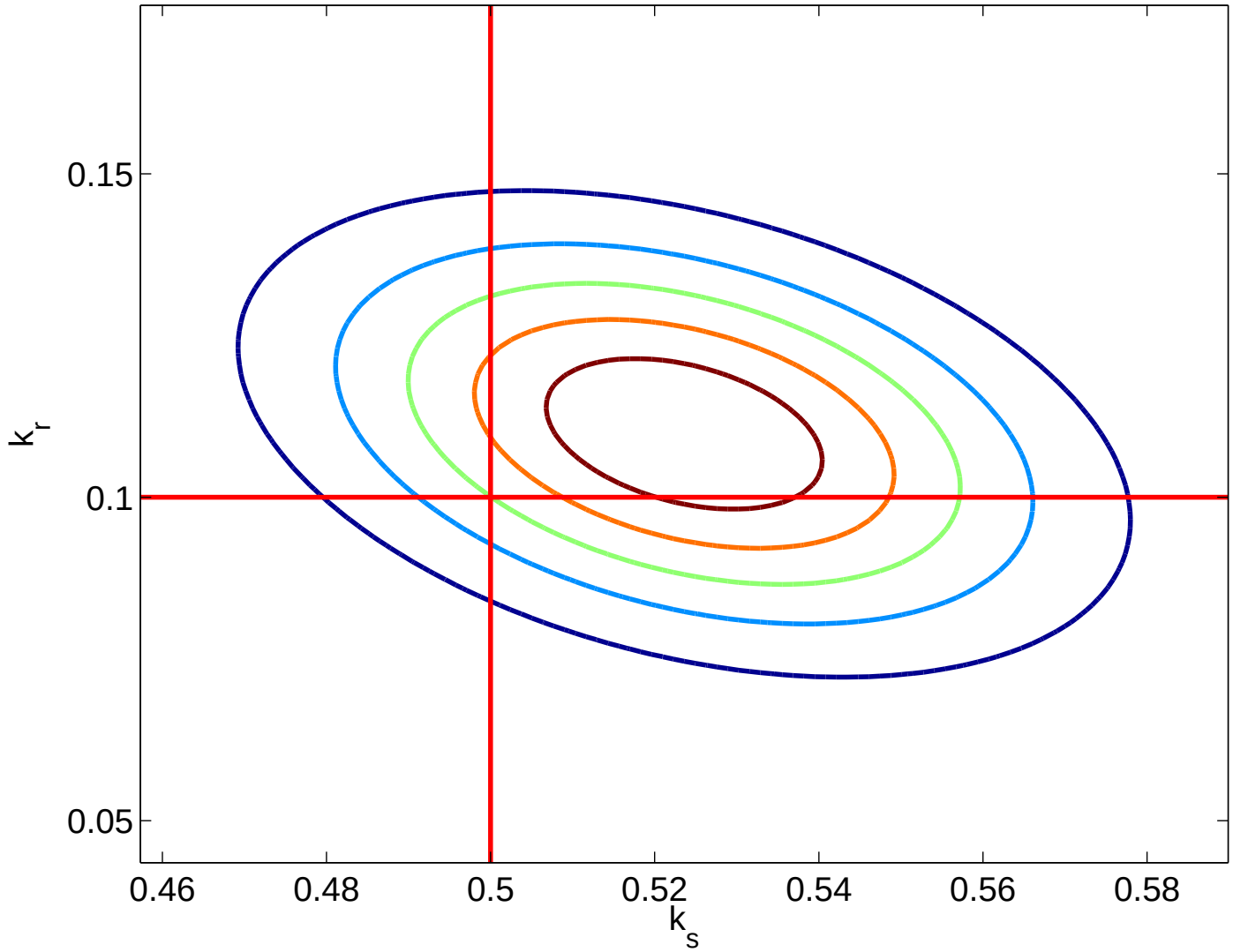
Results: Gaussian Process-based Optimization

Method 2:

- Gaussian Process-based optimization[5]
- Approximate the posterior as a Gaussian distribution (Laplace approximation)
- Observations: MiTL properties observed over a number of simulation runs

[5] Bortolussi and Sanguinetti, *Learning and Designing Stochastic Processes from Logical Constraints*, QEST 2013.

Results: Gaussian Process-based Optimization



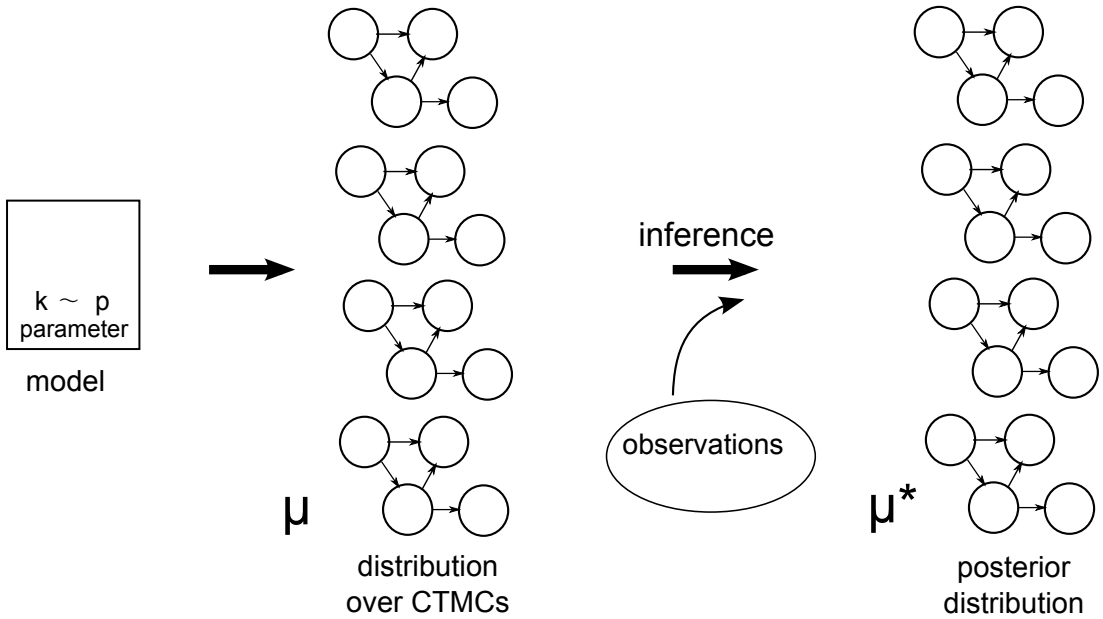
Outline

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.
- Tested the language with small model, employing two different inference algorithms.
- Parameter inference results consistent with expectations.

Future work

- Approximations that will allow for more efficient inference: fluid, Linear Noise Approximation, System Size Expansion
- Other inference algorithms
- Equivalence relations
- Larger examples



Thanks

Bio-PEPA

- Federica Ciocchetta

ProPPA

- Anastasis Georgoulas

- Guido Sanguinetti

GP Inference

- Dimitrios Milios