Probabilistic Programming Process Algebra

Anastasis Georgoulas[1], Jane Hillston[1], Dimitrios Milios[1], and Guido Sanguinetti[1,2]

[1] School of Informatics, University of Edinburgh
[2] SynthSys — Synthetic and Systems Biology, University of Edinburgh

Abstract. Formal modelling languages such as process algebras are widespread and effective tools in computational modelling. However, handling data and uncertainty in a statistically meaningful way is an open problem in formal modelling, severely hampering the usefulness of these elegant tools in many real world applications. Here we introduce ProPPA, a process algebra which incorporates uncertainty in the model description, allowing the use of Machine Learning techniques to incorporate observational information in the modelling. We define the semantics of the language by introducing a quantitative generalisation of Constraint Markov Chains. We present results from a prototype implementation of the language, demonstrating its usefulness in performing inference in a non-trivial example.

1 Introduction

Stochastic process algebras are an established tool for modelling and analysing the behaviour of dynamical systems, combining theoretical elegance with a range of attractive and practically useful features — compositionality, formal interpretation of models and the ability to verify their behaviour using model-checking, to list a few. The starting point for process algebras, as for many other formal modelling methods, is a full specification of the system being modelled, both in terms of interaction structure and of parameters quantifying the (infinitesimal) dynamics of the system. It is increasingly clear, however, that such complete knowledge is seldom achieved, and unattainable in a number of important application domains. This problem is particularly acute in the biomedical field, where parameters such as reaction rates are estimated from noisy measurements which are often highly sensitive to the experimental conditions. How to quantify and propagate this uncertainty in formal models is an open problem of fundamental importance in any scientific application.

Constraining models from empirical observations is a very large and active research field in machine learning and signal processing; however, these fields usually work directly with low-level mathematical descriptions of the system, which negate some of the major advantages of formal languages. Within the process algebra community, attempts to address this problem have mostly used greedy optimisation methods — the Evolving Process Algebra framework ([17, 18]), for instance, uses evolutionary computation algorithms to fit the parameters and structure of a process algebra model. While this and similar approaches

(further discussed in Section 7) are valuable contributions, they do present practical and conceptual limitations. First and foremost, the optimization methods used lack a statistical framework, and hence cannot quantify the uncertainty associated with their predictions. In general, optimisation methods return a single optimal value of the parameters: this implies an assumption that measurements are sufficient to *completely remove* uncertainty, clearly an untenable assumption. Secondly, the learning and the modelling take place on separate levels, since the modelling language itself does not really include the uncertainty about the system. Thus, the model does not reflect our understanding, and inference is done in an *ad hoc* manner, independently from the modelling. While this orthogonality may appear attractive, it introduces a degree of conceptual dissonance between what we try to capture and the way we represent it.

In this paper we aim to address these problems by introducing a formal modelling language which directly incorporates observations (and the associated uncertainty) and can leverage cutting-edge statistical machine learning tools to perform inference and quantify uncertainty. We are inspired by recent progress in probabilistic programming languages, which aim to perform *inference by programming*; however, current probabilistic programming languages are all relatively low-level. We introduce ProPPA, a Probabilistic Programming Process Algebra; to our knowledge, this is the first time the probabilistic programming paradigm is extended to a higher-level, formal system description language like a process algebra. Note that the ability to perform inference from data qualitatively distinguishes our approach from general stochastic modelling methodologies such as stochastic process algebras, which simply incorporate uncertainty in model evolution through the use of random variables to determine rates.

ProPPA is based on the stochastic process algebra Bio-PEPA [8], and inherits many of its qualities. We show how to include uncertainty in the definition of the language (Section 3), and propose an appropriate semantic model for uncertain models (Sections 4 and 5). We adopt a modular approach to construct our language, so that the core language is capable of adopting different machine learning methodologies to perform inference from possibly very different types of data. We demonstrate the power of this approach by performing inference in a nontrivial example in Section 6.

2 Background

This section gives some information on the language on which ProPPA is based, the frameworks and mathematical objects used for the definition of its semantics, and the field of probabilistic programming from which we draw inspiration.

2.1 Process algebras and Bio-PEPA

Process algebras are a family of languages first used to model concurrent systems, by specifying the system's components and the actions that these may perform. The formal nature of the languages allow one to reason about the behaviour of

the modelled system, such as verifying that undesirable states (configurations of the system) are avoided or that simple properties hold. The sub-family of stochastic process algebras (e.g. PEPA [15], IMC [14], EMPA [1]) extend this framework by introducing time into the system and assuming that the time for a transition to occur is an exponentially distributed random variable. The parameter of the distribution is called the rate of the transition, and, when multiple transitions are possible, the probability of choosing a particular transition is proportional to its rate. This is formalized through the concept of a Continuous Time Markov Chain (CTMC), a mathematical description of the possible states of the system and the transitions between them.

The description of a system in a process algebra can be used to implicitly generate its state space, in the form of a labelled transition system (LTS). A LTS is a graph whose nodes are the system's states and whose edges are the possible transitions between states, labelled with some information (e.g. what reaction causes the transition or at what rate the transition occurs). Analysing the LTS can give important insights into the behaviour of the system, such as whether a state is reachable under certain conditions or within a specified time frame. Verifying whether a system description satisfies such properties is the subject of model checking algorithms and tools, with the properties to be checked often being expressed in a temporal logic, such as CSL [2] or CTL [9].

Bio-PEPA [8] is a stochastic process algebra based on PEPA but designed for the modelling of biological processes. In Bio-PEPA, system components (termed *species*) are defined through their behaviour, that is, how they interact with each other, reflecting a reagent-centric modelling style. The definition of a species takes the form

$$A = (\alpha_1, k_1)op_1 + \cdots + (\alpha_n, k_n)op_n \text{ where } op_i = \downarrow, \uparrow, \oplus, \ominus \text{ or } \odot$$

which means that species A takes part in reaction α_i with stoichiometry k_i. The different options for op_i correspond to different roles of A in α_i: reactant, product, catalyst, inhibitor or generic modifier, respectively. These definitions are composed using the choice operator ($+$) to describe species that can take part in multiple reactions.

Each reaction has an associated rate law, which can be specified either as a formula or using a predefined law (such as mass-action or Hill kinetics). Parameters can be defined and used, for example, in kinetic laws or as initial concentrations, but their values must be specified and are considered fixed. The language results in a modular or compositional approach, wherein the behaviour of the system emerges as a direct consequence of the behaviour of the species (without the need to, for instance, explicitly write out ODEs or chemical equations for reactions, as they can be automatically computed). This means modifications to the model can be performed by changes to the "local" species definitions.

Formally, a Bio-PEPA system is defined as a tuple $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}_R, Comp, P \rangle$, where $\mathcal{V}$ is the set of compartments (locations) in the system; $\mathcal{N}$ is a set of quantities associated with each species, such as its maximum concentration; $\mathcal{K}$ is the set of parameters; $\mathcal{F}_R$ is the set of rate laws; $Comp$ is the set of sequential components (species definitions); and P is the model component, which describes

how the various species cooperate with each other as well as their initial concentrations. An example of a model component comprising three species A_i with initial quantities l_i is

$$A_1[l_1] \underset{*}{\bowtie} A_2[l_2] \underset{*}{\bowtie} A_3[l_3]$$

2.2 FuTS

The FuTS (state-to-function transition system) framework [20] is a way of describing semantics of process algebras. In a FuTS, the possible transitions from a state s can be represented collectively as $s \overset{\alpha}{\rightarrowtail} f$, where f is called the *continuation* and is a function over states. The value of $f(s')$ then gives some information (such as the rate) about the transition $s \xrightarrow{\alpha} s'$. Depending on the codomain of the continuation functions, FuTS can represent different kinds of behaviour and their associated information, such as non-determinism (continuations take boolean values to denote possible next states), discrete time systems (values in $[0, 1]$ give transition probabilities) or continuous time systems (values in $\mathbb{R}$ to denote transition rates). The notation $[s_1 \mapsto v_1, s_2 \mapsto v_2, \ldots, s_n \mapsto v_n]$ is shorthand for a function f such that $f(s_i) = v_i$, $i = 1 \ldots n$ and $f(s)$ takes the zero value of its codomain (0 for real values, *false* for boolean etc.) for all states besides the specified s_i. As ProPPA represents uncertainty using probability distributions, the FuTS style, which already makes use of functions, seems a natural fit for expressing its semantics.

2.3 Constraint Markov Chains

Constraint Markov Chains (CMCs, [6]) are a generalisation of Discrete Time Markov Chains in which the probability of transitioning from a given state to another does not have a fixed value. Instead, the CMC specifies a constraint that the values of the various transition probabilities must obey or, equivalently, a set of acceptable values for them. Formally, a 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.
- $\phi : S \times [0, 1]^k \to \{0, 1\}$ is the *constraint function*.

The constraint function indicates whether a given set of transition probabilities satisfies the constraints: $\phi(i, \boldsymbol{r}) = 1$ if and only if $\boldsymbol{r} = (r_1, r_2, \ldots, r_k)$ is an acceptable vector of transition rates from state i.

As explained in [24], there are two ways of interpreting the uncertainty in the transition probabilities, which give rise to different behaviours when simulating a CMC. One way is to assume that the transition probabilities can change during the run of the system, so that each time we visit a state we must choose new values for them. This is referred to as Markov Decision Process (MDP) semantics.

Alternatively, under the Uncertain Markov Chain (UMC) semantics, we assume that each probability has a constant (but unknown) value. In this case, the values are fixed before the simulation and maintained throughout.

2.4 Probabilistic Programming

Probabilistic programming is a framework for reasoning about uncertain processes in a statistically consistent manner. In a probabilistic program, uncertain aspects of the system, such as unknown parameters, are treated as random variables and can be assigned probability distributions that express this uncertainty. Additionally, one can specify observations of the system, from which information about the unobserved aspects can be gleaned. In other words, the program specifies a probability distribution, which can be viewed in two ways: one can sample from it, essentially simulating the system; or, if one has additional information about the system, one can condition the distribution on this data, inferring an updated distribution over the unknown variables that takes into account this new knowledge. Probabilistic programming offers an elegant approach for treating uncertain systems in these two ways, automating the process to a degree and eliminating the need for bespoke inference solutions, as the inference algorithm can be configured and executed automatically based on the structure of the program.

Previous work has focused mainly on integrating the paradigm into traditional programming languages, giving rise to frameworks like Church [13], IBAL [21] and Infer.NET [19]. These languages, however, describe systems at a low level: one must explicitly specify all the statistical dependences between the different variables, yielding potentially large descriptions which are difficult to manage. This limits the range of systems that can be modelled, with continuous-time dynamical systems being particularly hard or even impossible to deal with. We therefore advocate combining the principle of probabilistic programming with a formal language like a process algebra, for a flexible, high-level framework in which to model and analyse complex systems with uncertain aspects.

3 A Probabilistic Programming Process Algebra

The ProPPA syntax is based on Bio-PEPA, with the addition of two key features that introduce aspects of probabilistic programming. The first concerns the representation of uncertainty in the system. We should note that we are only considering uncertainty in the kinetics, and assume that we fully know what reactions each species can take part in. In Bio-PEPA, parameters can be used in the definition of kinetic rate functions, but their values must be fixed. With this in mind, we allow uncertain parameters, whose values are given as probability distributions rather than concrete numbers. The second feature is a way of incorporating information about the behaviour of the system into the model. These will be the *observations*, which may take the form of actual partial observations of the state of the system (as a time series) or, more generally, could be any

observed function of the specific trajectory of the system (specified through a temporal logic formula, for instance).

On a formal level, we will need to modify the definition of a Bio-PEPA system (given previously in Section 2.1), mainly by reconsidering the role of the set of parameters $\mathcal{K}$. We extend the system definition in two ways, corresponding to the two features described above. Firstly, since the uncertain quantities are represented as parameters in the model, we extend the definition of a parameter to include a distribution rather than a concrete value. These are the *prior distributions* or *priors* over parameters, which express our belief about a parameter's values before seeing any data. The set of parameters $\mathcal{K}$ is then partitioned into two subsets: $\mathcal{K}_c$ comprises the concrete parameters, while $\mathcal{K}_u$ contains the uncertain ones, along with the priors associated with them. We write $(k \sim \mu) \in \mathcal{K}_u$ if the parameter k is drawn *a priori* from the distribution μ. Importantly, the functional rates $\mathcal{F}_R$ can refer to any parameter in $\mathcal{K}_u$ as well as those in $\mathcal{K}_c$; in this sense, a functional rate can represent a family of functions.

Secondly, we add a new component $\mathcal{O}$ representing the observations. These impose restrictions on the acceptable parameter values and modify our belief about their distribution, as described in more detail in Section 5.4. Extending the syntax to accommodate these features is straightforward. A ProPPA system is therefore a tuple $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}_R, \mathcal{O}, Comp, P \rangle$, with the other components retaining the meaning they have in Bio-PEPA. Following the terminology of [11], we will also write a system as $\langle \mathcal{T}, P \rangle$ where $\mathcal{T} = \langle \mathcal{V}, \mathcal{N}, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}_R, \mathcal{O}, Comp \rangle$ is called the context.

3.1 A rumour spreading example

As an example of a ProPPA model, we will consider a population CTMC model of rumour spreading over a network [10]; this consists of three types of agents (Figure 1). A spreader (S) is someone who has already encountered the rumour and is actively trying to spread it. When an ignorant (I) meets a spreader, the ignorant also becomes a spreader. When two spreaders meet, one of them stops spreading and becomes a blocker (R), reflecting the idea that only new rumours are worth spreading. A blocker can then convert spreaders into other blockers. The dynamics of the system can exhibit qualitatively different behaviours depending on the parameter values: in particular, two possible steady state regimes exist, where all agents are in blocker state, or where a blocker and an ignorant population coexist.

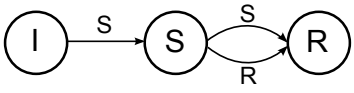


Fig. 1: State transitions of a rumour-spreading agent. The arrow labels indicate the type of agent that must be encountered for the corresponding transition to take place.

```
1  k_s = Uniform(0,1);
2  k_r = Uniform(0,1);
3
4  kineticLawOf spread : k_s * I * S;
5  kineticLawOf stop1 : k_r * S * S;
6  kineticLawOf stop2 : k_r * S * R;
7
8  I = (spread,1) ↓ ;
9  S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
10 R = (stop1,1) ↑ + (stop2,1) ↑ ;
11
12 I[10] ⋈*  S[5] ⋈*  R[0]
13
14 observe('trace')
15 infer('ABC')
```

Fig. 2: ProPPA model of the rumour spreading example.

Figure 2 shows the description of the system in ProPPA for an initial population of 15 agents. The behaviour of the different agents is described in lines 8-10: line 8 says that the count of ignorants is decreased by 1 when the `spread` interaction occurs, and the other types of agents are similarly defined by the changes to their count through the various interactions. Line 12 shows the initial population of each kind of agent; the cooperation $\bowtie^*$ means that the agents synchronise on all shared reactions. We assume that these interactions happen at rates that obey mass-action kinetics, i.e. they are proportional to the count of the agents involved. We also assume that the rate constants for the spreader-spreader and spreader-blocker interactions are the same (k_r), while the spreader-ignorant interaction has a rate constant k_s; this is shown in lines 4-6. The definition of k_s and k_r as uniformly distributed (lines 1-2) reflects our prior belief that, without seeing any data, they are not biased towards any value in their domain, which in this case we chose to be $[0, 1]$. We will use this model as a running example for the rest of this paper; lines 14-15 are discussed in Section 5.4.

4 Probabilistic Constraint Markov Chains

As mentioned earlier, Bio-PEPA models can be mapped to CTMCs. In a CTMC, every transition between states has a concrete rate, making this interpretation unsuitable for a language with uncertainty, such as ProPPA. We must therefore use a different object to define the semantics of our language, one that is more suited to describing uncertain models, such as a CMC.

The way CMCs were first proposed (shown in Section 2.3) presents two limitations. Firstly, they have been defined only for discrete-time systems, whereas we are interested in modelling in continuous time. Their definition can be adapted to the continuous-time domain through simple alterations which are presented

below. Secondly, while a CMC defines the set of possible values for a rate, it gives no information on the relative likelihood of those values. We would like to move from a purely non-deterministic to a probabilistic setting, where, instead of a binary decision, we have quantitative information about our belief in the plausibility of a value.

Based on the original definition, we can define a Probabilistic Constraint Markov Chain as 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*.

The changes concern the constraint function and address the limitations described. The constraints are now on rates rather than transition probabilities, reflecting the shift to continuous time. Additionally, ϕ now describes a probability density function, therefore it takes values in $\mathbb{R}_+$ instead of $\{0, 1\}$, with the additional restriction that $\int_0^\infty \cdots \int_0^\infty \phi(i, \boldsymbol{r}) d\boldsymbol{r} = 1$ for every $i \in \{1, k\}$. The resulting object is richer, and the additional information it can capture means we can use Probabilistic CMCs to define the semantics of ProPPA models, as explained in the next section.

5 ProPPA Semantics

We now describe the semantics of a ProPPA model, eventually mapping to a Probabilistic CMC. The semantics of Bio-PEPA are given in terms of two relations. The *capability relation* describes what transitions may occur between states, without giving any quantitative information about the rates — in other words, it gives the structure of the transition system. The *stochastic relation* uses that information, as well as the definition of the kinetic functions, to provide the rates of the transitions, thus completing the labelling of the transition system.

We have kept this two-step approach, and have in fact found the separation of the two steps to be useful for our extension. Since there is no uncertainty in the qualitative behaviour of the species, the capability relation remains unchanged by our additions. We present both the existing capability and the new stochastic relation in a uniform way using the FuTS framework.

5.1 Capability relation

To consider the capability relation as a FuTS, we use boolean-valued continuations: $P \overset{l}{\rightarrowtail}_c f$ means that P can transition to those states Q for which $f(Q)$ is *true*. For simple terms, there is at most one reachable state for any reaction, as described by the rules below, where *op* is one of the modifier operators $\oplus$, $\ominus$ and $\odot$, and N is the maximum count for the species S:

$$\texttt{PrefixReac}\ (a,k)\downarrow S(l) \xrightarrow{(a,[S:\downarrow(l,k)])}_c [S(l-k)\mapsto \mathit{true}]\quad k\le l\le N$$

$$\texttt{PrefixProd}\ (a,k)\uparrow S(l) \xrightarrow{(a,[S:\uparrow(l,k)])}_c [S(l+k)\mapsto \mathit{true}]\quad 0\le l\le N-k$$

$$\texttt{PrefixMod}\ \ (a,k)\ op\ S(l) \xrightarrow{(a,[S:op(l,k)])}_c [S(l)\mapsto \mathit{true}]\qquad \begin{cases} k<l\le N \text{ if } op=\oplus \\ k\le l\le N \text{ otherwise}\end{cases}$$

The label (α, w) records two kinds of information: α is the reaction name, while w is a list of *roles*, indicating how a species takes part in a reaction, with what stoichiometry and at what initial count. The rules for the choice and cooperation operators are simple:

$$\texttt{Choice1}\ \frac{P_1 \xrightarrow{(a,w)}_c f}{P_1+P_2 \xrightarrow{(a,w)}_c f} \qquad \texttt{Choice2}\ \frac{P_2 \xrightarrow{(a,w)}_c f}{P_1+P_2 \xrightarrow{(a,w)}_c f}$$

$$\texttt{Coop1}\ \frac{P_1 \xrightarrow{(a,w)}_c f_1 \qquad a\notin\mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(a,w)}_c g_1} \qquad \texttt{Coop2}\ \frac{P_2 \xrightarrow{(a,w)}_c f_2 \qquad a\notin\mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(a,w)}_c g_2}$$

$$\texttt{Coop3}\ \frac{P_1 \xrightarrow{(a,w_1)}_c f_1 \qquad P_2 \xrightarrow{(a,w_2)}_c f_2 \qquad a\in\mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(a,w_1::w_2)}_c g}$$

where :: indicates concatenation of lists and we make the usual distinction according to whether the reaction in question is shared between the cooperating components. The functions g_1, g_2 and g are as follows:

$$g_1(Q)=\begin{cases} f_1(Q_1) & \text{if } Q=Q_1 \underset{\mathcal{L}}{\bowtie} P_2 \\ \mathit{false} & \text{otherwise}\end{cases} \qquad g_2(Q)=\begin{cases} f_2(Q_2) & \text{if } Q=P_1 \underset{\mathcal{L}}{\bowtie} Q_2 \\ \mathit{false} & \text{otherwise}\end{cases}$$

$$g_1(Q)=\begin{cases} f_1(Q_1)\wedge f_2(Q_2) & \text{if } Q=Q_1 \underset{\mathcal{L}}{\bowtie} Q_2 \\ \mathit{false} & \text{otherwise}\end{cases}$$

Finally, for completeness, we have a rule for named species definition:

$$\texttt{Constant}\ \frac{P \xrightarrow{(a,w)}_c f \qquad Q \overset{def}{=} P}{Q \xrightarrow{(a,w[P\to Q])}_c f}$$

where $w[P \to Q]$ means renaming all instances of P in w to Q.

For example, in the rumour-spreading network (Figure 2), we have that $S(2) \xrightarrow{(stop2,[S:\downarrow(2,1)])}_c [S(1)\mapsto \mathit{true}]$ and $R(1) \xrightarrow{(stop2,[R:\uparrow(1,1)])}_c [R(2)\mapsto \mathit{true}]$ from the prefix rules. By applying the cooperation rules, we can infer that

$$I(0) \underset{*}{\bowtie} S(2) \underset{*}{\bowtie} R(1) \xrightarrow{(stop2,w)}_c \Big[I(0) \underset{*}{\bowtie} S(1) \underset{*}{\bowtie} R(2)\mapsto \mathit{true}\Big]$$

where $w = [S :\downarrow (2,1), R :\uparrow (1,1)]$.

5.2 Stochastic relation

The capability relation is defined, as above, between species. The stochastic relation, in contrast, is between whole systems and gives information on the rate of transitioning from one system to another. In Bio-PEPA, the reaction rates depend on the model's parameters, and this dependence is also true in ProPPA. This means that any uncertainty about the values of the parameters must lead to uncertainty about the rates. The ProPPA stochastic relation, therefore, gives a distribution over possible rates instead of a single value. If all the parameters on which a particular rate depends are concrete, this will simply be a Dirac δ distribution, i.e. one where all the probability mass is assigned to a single value.

When building the stochastic relation, we use both the capability relation, which indicates if a state is reachable, as well as the system's context $\mathcal{T}$ (see Section 3), which holds information related to the reaction rates. The context provides two pieces of information we require: the kinetic law for the reaction, and the distribution of the uncertain parameters involved in that law. As the rate normally depends on the concentrations of the species involved, we also need this information, which is contained in the roles w, in the label of the capability relation. The stochastic relation is defined by the rule:

$$\frac{P \overset{(a,w)}{\rightarrowtail}_c g}{\langle \mathcal{T}, P \rangle \overset{a}{\rightarrowtail}_s h_{g,w,\mathcal{T}}}$$

As described above, the function h maps systems to distributions of rates:

$$h_{g,w,\mathcal{T}}(\langle \mathcal{T}', s' \rangle) = \begin{cases} \delta(0) & \text{if } \mathcal{T}' \neq \mathcal{T} \\ \delta(0) & \text{if } g(s') = \mathit{false} \\ \mu & \text{otherwise} \end{cases}$$

According to this, transitions to systems with a different context or where the species is not reachable can only occur at zero rate, i.e. never. We now show how to derive the distribution over rates (μ above) in the non-trivial cases.

Assume the rate Y of a reaction depends on a parameter Θ and let $Y = T(\Theta)$ express this dependence. We know, from the context, that Θ is distributed according to a probability density function (pdf) f_Θ. Y, being a transformation of Θ, will also follow a distribution, whose pdf we denote f_Y. If the function T is strictly monotonic, f_Y can be obtained through a simple change of variable:

$$f_Y(y) = \left| \frac{dT^{-1}(Y)}{dY} \right|_{Y=y} f_\Theta(T^{-1}(y)) \tag{1}$$

where $T^{-1}(y)$ is the (unique) value of the parameter Θ for which the rate is y.

To see why this is valid, let us first consider the case where T is strictly increasing, which implies that the inverse function T^{-1} is well-defined and is also increasing. The cumulative distribution function of Y is then:

$$F_Y(y) = P\left(Y \leq y\right) = P\left(T(\Theta) \leq y\right) = P\left(\Theta \leq T^{-1}(y)\right) = F_\Theta\left(T^{-1}(y)\right)$$

and the corresponding pdf is:

$$f_Y(y) = \left.\frac{dF_Y(y)}{dY}\right|_{Y=y} = \left.\frac{dF_\Theta\left(T^{-1}(Y)\right)}{dY}\right|_{Y=y} = f_\Theta\left(T^{-1}(y)\right)\left.\frac{dT^{-1}(Y)}{dY}\right|_{Y=y}$$

Considering the case where T is decreasing leads to the general result (1).

Once the stochastic relation has been constructed, it is then easy to build a Probabilistic CMC that captures its behaviour by appropriately defining the constraint function. To do so, we need the probability density of a vector of rates, which can be obtained as the product of the densities for each individual rate. The latter are given directly by the stochastic relation. Note that there can be more than one reaction that leads to the same transition in the state space; the total transition rate is then the sum of the rates of the individual reactions, and the pdf of the sum of random variables is the convolution of their individual pdfs. For a system with k states, then, the constraint function ϕ over the rates r_{ij} of transitioning from state i to state j is given by:

$$\phi(i, (r_{i1}, r_{i2}, \ldots, r_{ik})) = \prod_{j=1}^{k} f_j(r_{ij})$$

where $\langle \mathcal{T}, i\rangle \overset{\alpha}{\rightarrowtail}_s f_\alpha$ and $f_j = \bigotimes_\alpha f_\alpha(\langle \mathcal{T}, j\rangle)$ is the convolution described above.

5.3 Concretization

We now describe how the choice of UMC or MDP interpretation (discussed in Section 2.3) affects the behaviour of the system. Essentially, in the UMC setting we replace the prior distribution over each parameter with a Dirac δ distribution, to reflect the fact that a value is chosen only once and remains fixed thereafter. First we note that the concretization procedure only affects the context of a system, so the corresponding transition relation will have the form $\mathcal{T} \rightarrowtail g_{\mathrm{X}}(\mathcal{T})$, where $\mathcal{T}$ is a context (as before) and X can be UMC or MDP. The corresponding continuation functions for each scenario are defined as:

$$g_{\mathrm{UMC}}(\langle \mathcal{V}, \mathcal{N}, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}_R, \mathcal{O}, Comp\rangle)(\mathcal{T}') = \begin{cases} \prod_i f_i(v_i) & \text{if } \mathcal{T}' = \langle \mathcal{V}, \mathcal{N}, \mathcal{K}_c, \{k_i \sim \delta(v_i)\}, \mathcal{F}_R, \mathcal{O}, Comp\rangle \\ 0 & \text{otherwise} \end{cases}$$

and

$$g_{\mathrm{MDP}}(\mathcal{T})(\mathcal{T}') = \begin{cases} 1 & \text{if } \mathcal{T}' = \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

5.4 Observations and inference

We have shown (Section 5.2) how the prior beliefs about the parameters induce a distribution over rates, but we have so far assumed that no observations are present. Observations represent new information which can affect our belief about how likely different values are. Inference can then be thought of as a transformation of the context, which takes the observations into account and updates the distributions of the uncertain parameters accordingly. From a probabilistic point of view, this corresponds to conditioning the prior distribution $P(\theta)$ on the observed data D, obtaining the *posterior distribution* $P(\theta \mid D)$. The relation between these quantities is given by Bayes' Theorem:

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

where $P(D \mid \theta)$ is the likelihood of the data, a measure of how likely we are to see these observations for a particular assignment of values to the parameters.

This view does not give any information on *how* to perform inference, however. Indeed, it is generally not possible to calculate the likelihood or the posterior analytically, so approximations must be used. The specification of the language allows for some freedom in the inference implementation; this approach creates a modular framework that can employ different algorithms, some examples of which are given in the next section. The choice may depend in part on the type of observations available, and here we focus on two possible ways of specifying those: a time-series of measurements of the species in the system, or a set of temporal logic formulae, which describe properties of the system's behaviour. A statement of the form `infer(`*algorithm*`)` in the model sets the desired inference algorithm.

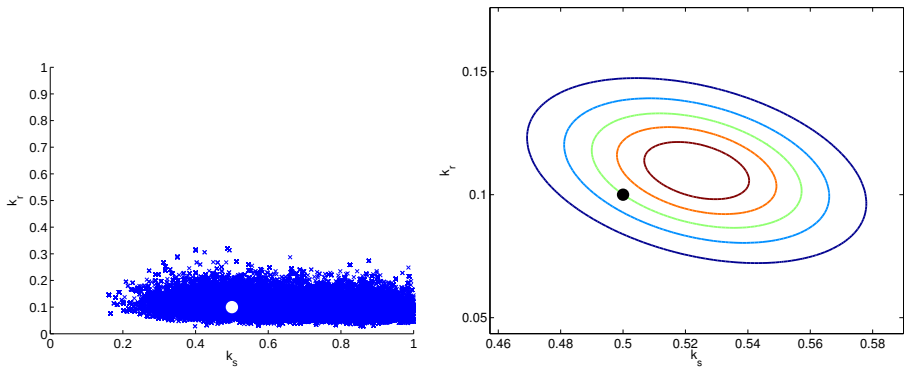
In line 14 of Figure 2, the `observe` statement refers to an external file which holds our observations of the system. Line 15 specifies what inference algorithm will be used.

6 Inference

In this section, we illustrate the capabilities of the language by showing some ways we can perform inference on the model of Figure 2.

6.1 Inference from time-series observations

We first assume that our observations are measurements of the species at different times during a run of the system. We can use an Approximate Bayesian Computation (ABC) [25] algorithm that takes into account both the prior beliefs and the data to obtain the posterior distribution over parameter values. ABC provides an efficient way of exploring the parameter space and keeping those values which better fit the data, and is particularly useful in cases where the likelihood is unknown or intractable to calculate. The algorithm returns an approximation to the posterior distribution as a (multi)set of parameter samples.



(a) (b)

Fig. 3: Results from the two inference methods: (a) accepted samples after running ABC; (b) contours of the Laplace approximation of the posterior when using formulae as inputs (the outer ring corresponds to the 98th quantile). The dots show the true values.

For this experiment, we simulated the system with $k_s = 0.5$, $k_r = 0.1$, and used ten points from the resulting trajectory as the input time-series. We gathered 100,000 parameter samples using ABC in approximately 20 seconds, plotted in Figure 3a. We can see the distribution of k_r is centred on its true value. It is also narrower than that of k_s, indicating that the behaviour of the system is not as sensitive to the latter. Simulating the system for values of k_s in $[0.3, 1.0]$ shows that its behaviour does not change much in this range, validating our results. Even with this wide variance, however, the posterior differs from the prior in that it assigns little or no probability to values of k_s under 0.3, which produce significantly different behaviour to the one in the input observations.

6.2 Inference from specification

We now examine the possibility of inference from specification in the form of logical constraints, rather than quantitative observations in the form of time-series. In many cases detailed quantitative information is not readily available, whereas we have an idea of how the model should behave with respect to certain properties. We follow the framework of Bortolussi & Sanguinetti [4], in which observations are satisfaction values of formulae specified in a suitable temporal logic (e.g. MiTL). This method yields an estimate for the parameter value that optimises the posterior distribution (Section 5.4); we can then use this to approximate the whole posterior with a Gaussian distribution by employing the *Laplace approximation*.

We have considered the following three properties, expressed in MiTL:

– $\mathbf{G}^{[3,5]}(I > 0)$: There are still ignorants at all time points between time 3 and 5, i.e. the rumour has not reached everyone in the population.

– $\mathbf{G}^{[0.5,1]}(R \geq S)$: Between time 0.5 and 1, the blockers are always more than the spreaders.
– $(\mathbf{F}^{[0,1]}(S \geq 7.5)) \wedge (\mathbb{G}^{[1,2]}(S \leq 3.75))$: The proportion of spreaders reaches or exceeds 50% of the population before time 1, and between time 1 and 2 the spreaders are always less than or equal to 25% of the population.

In order to produce the input specification data, we simulated the system 100 times after fixing $k_s = 0.5$, $k_r = 0.1$ as previously. After running the optimisation algorithm using this input and the priors from the model, we obtained estimates of 0.5236 and 0.1098 for k_s and k_r respectively. Figure 3b depicts the contours of the posterior Gaussian distribution over the parameters, given via Laplace approximation. Similarly to the previous section, the model appears to be less sensitive to k_s compared to k_r. This is reflected in their standard deviations, as given by the Laplace approximation; these are 0.0288 and 0.0199 respectively, meaning that the approximate distribution captures the increased uncertainty regarding k_s.

7 Related work

Some previous work has explored parameter estimation methods for formal models. The Evolving Process Algebra [17] framework uses genetic algorithms to find parametrizations of models written in the PEPA language, such that the behaviour of the model matches an observed time-series. It has been further applied to optimising the structure of Bio-PEPA models, going beyond parameter search [18]. The work in [23] deals with parameter estimation for the Calculus of Wrapped Components, employing a different search algorithm. The problem has also been considered in the case of BIOCHAM in [7], where the space of possible parameters is searched exhaustively to find values giving a good fit.

The somewhat related problem of reasoning about the behaviour of incompletely specified process algebra models, rather than inferring their parameters, is dealt with in [3]. Brim *et al.* [5] propose a method of approximating quantitative model checking results over an entire parameter space as an alternative to parameter estimation. Although not directly relevant to inference, the work in [16] is another example of combining machine learning with formal modelling, presenting a Bayesian approach to statistical model checking.

With regards to probabilistic programming, as mentioned previously, languages like Church [13], IBAL [21] and Infer.NET [19] are not particularly suited to modelling complex dynamical systems. A language for describing continuous-time systems has been proposed in [22], but still lacks the formal features of a process algebra. An initial attempt to apply probabilistic programming to continuous-time models of biological systems also used a lower-level description language [12].

8 Conclusions

We have presented a process algebra that incorporates elements of probabilistic programming, the first such attempt at combining the two fields. This approach integrates uncertainty and observations into the system description, allowing us to model systems for which we have incomplete knowledge and giving us access to techniques from machine learning for inferring the unknown parameters.

The new features, while affecting the syntax of the language only minimally, significantly extend its expressivity. Additionally, our system is modular and flexible in the choice of inference algorithm. The application of two such algorithms on an example gives promising results for the effectiveness of our approach.

An interesting question for future work is whether the observations can be further integrated into the semantics of the language — for instance, whether the specification of a system can let us reject some transitions when building the underlying transition system. We are also interested in examining other benefits afforded by the use of a formal language, such as the description of equivalences and how they can be adapted to this probabilistic programming-like setting. Furthermore, we plan to explore more inference algorithms and test our framework on larger systems.

Acknowledgements

This work was supported by Microsoft Research through its PhD Scholarship Programme, the EU FET-Proactive programme through QUANTICOL grant 600708, the SysMIC project through BBSRC grant BB/I014713/1 and the European Research Council through grant MLCS306999. The authors thank Luca Cardelli and Vashti Galpin for their comments and suggestions.

References

1. Aldini, A., Bernardo, M., Corradini, F.: A process algebraic approach to software architecture design. Springer (2010)
2. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Verifying continuous time Markov chains. In Alur, R., Henzinger, T., eds.: Computer Aided Verification. Volume 1102 of LNCS. (1996) 269–276
3. Baldan, P., Bracciali, A., Brodo, L., Bruni, R.: Deducing Interactions in Partially Unspecified Biological Systems. In Anai, H., Horimoto, K., Kutsia, T., eds.: Algebraic Biology. Volume 4545 of LNCS. (2007) 262–276
4. Bortolussi, L., Sanguinetti, G.: Learning and designing stochastic processes from logical constraints. In Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P., eds.: Quantitative Evaluation of Systems. Volume 8054 of LNCS. (2013) 89–105
5. Brim, L., Češka, M., Dražan, S., Šafránek, D.: Exploring parameter space of stochastic biochemical systems using quantitative model checking. In: Computer Aided Verification. (2013) 107–123
6. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wsowski, A.: Constraint Markov Chains. Theor. Comp. Science **412**(34) (2011) 4373 – 4404

7. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine Learning Biochemical Networks from Temporal Logic Properties. In Priami, C., Plotkin, G., eds.: Transactions on Computational Systems Biology VI. Volume 4220 of LNCS. (2006) 68–94
8. Ciocchetta, F., Hillston, J.: Bio-PEPA: A framework for the modelling and analysis of biological systems. Theor. Comp. Science **410**(33-34) (2009) 3065–3084
9. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8**(2) (April 1986) 244–263
10. Daley, D.J., Kendall, D.G.: Epidemics and Rumours. Nature **204**(4963) (1964)
11. Galpin, V.: Equivalences for a biological process algebra. Theor. Comp. Science **412**(43) (2011) 6058 – 6082
12. Georgoulas, A., Hillston, J., Sanguinetti, G.: ABC-Fun: A Probabilistic Programming Language for Biology. In Gupta, A., Henzinger, T., eds.: Computational Methods in Systems Biology. Volume 8130 of LNCS. (2013) 150–163
13. Goodman, N.D., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In McAllester, D.A., Myllymäki, P., eds.: UAI, AUAI Press (2008) 220–229
14. Hermanns, H.: Interactive Markov Chains: and the quest for quantified quality. Springer-Verlag (2002)
15. Hillston, J.: A Compositional Approach to Performance Modelling. CUP (1996)
16. Jha, S., Clarke, E., Langmead, C., Legay, A., Platzer, A., Zuliani, P.: A Bayesian Approach to Model Checking Biological Systems. In Degano, P., Gorrieri, R., eds.: Computational Methods in Systems Biology. Volume 5688 of LNCS. (2009) 218–234
17. Marco, D., Cairns, D., Shankland, C.: Optimisation of process algebra models using evolutionary computation. In: 2011 IEEE Congress on Evolutionary Computation (CEC). (2011) 1296–1301
18. Marco, D., Shankland, C., Cairns, D.: Evolving Bio-PEPA process algebra models using genetic programming. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference. GECCO '12, New York, NY, USA (2012) 177–184
19. Minka, T., Winn, J., Guiver, J., Knowles, D.: Infer.NET 2.5 (2012) Microsoft Research Cambridge. http://research.microsoft.com/infernet.
20. Nicola, R.d., Latella, D., Loreti, M., Massink, M.: A Uniform Definition of Stochastic Process Calculi. ACM Comput. Surv. **46**(1) (October 2013) 5:1–5:35
21. Pfeffer, A.: The Design and Implementation of IBAL: A General-Purpose Probabilistic Language. In Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. The MIT Press (2007)
22. Pfeffer, A.: CTPPL: A Continuous Time Probabilistic Programming Language. In: IJCAI. (2009) 1943–1950
23. Sciacca, E., Spinella, S., Calcagno, C., Damiani, F., Coppo, M.: Parameter Identification and Assessment of Nutrient Transporters in AM Symbiosis through Stochastic Simulations. ENTCS **293**(0) (2013) 83 – 96 Proceedings of CS2Bio'12.
24. Sen, K., Viswanathan, M., Agha, G.: Model-Checking Markov Chains in the Presence of Uncertainties. In Hermanns, H., Palsberg, J., eds.: Tools and Algorithms for the Construction and Analysis of Systems. Volume 3920 of LNCS. (2006) 394–410
25. Toni, T., Welch, D., Strelkowa, N., Ipsen, A., Stumpf, M.P.: Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. Journal of The Royal Society Interface **6**(31) (2009) 187–202