Logical Methods in Computer Science
Vol. 11(2:3)2015, pp. 1–24
www.lmcs-online.org

Submitted Dec. 29, 2013
Published May. 31, 2015

LEARNING AND DESIGNING STOCHASTIC PROCESSES FROM LOGICAL CONSTRAINTS

LUCA BORTOLUSSI [a] AND GUIDO SANGUINETTI [b]

[a] Modelling and Simulation Group, Saarland University, Germany and Department of Mathematics and Geosciences, University of Trieste and CNR/ISTI, Pisa, Italy

[b] School of Informatics, University of Edinburgh and SynthSys, Centre for Synthetic and Systems Biology, University of Edinburgh

Abstract. Stochastic processes offer a flexible mathematical formalism to model and reason about systems. Most analysis tools, however, start from the premises that models are fully specified, so that any parameters controlling the system's dynamics must be known exactly. As this is seldom the case, many methods have been devised over the last decade to infer (learn) such parameters from observations of the state of the system. In this paper, we depart from this approach by assuming that our observations are *qualitative* properties encoded as satisfaction of linear temporal logic formulae, as opposed to quantitative observations of the state of the system. An important feature of this approach is that it unifies naturally the system identification and the system design problems, where the properties, instead of observations, represent requirements to be satisfied. We develop a principled statistical estimation procedure based on maximising the likelihood of the system's parameters, using recent ideas from statistical machine learning. We demonstrate the efficacy and broad applicability of our method on a range of simple but non-trivial examples, including rumour spreading in social networks and hybrid models of gene regulation.

1. Introduction

Stochastic processes are fundamental tools for modelling and reasoning about many physical and engineered systems. Their elegant mathematical formulation allows to capture quantitatively the mechanisms underlying the intrinsically noisy dynamics frequently encountered in many applications, ranging from computer networks to systems biology. At the same time, their importance has motivated intense research in analytical and computational tools to characterize emergent properties of models, and to efficiently simulate system trajectories by

2012 ACM CCS: [**Theory of computation**]: Theory and algorithms for application domains—Machine learning theory; Logic—Modal and temporal logics; [**Software and its engineering**]: Software organization and properties—Software functional properties—Formal methods—Software verification.

Key words and phrases: machine learning, parameter synthesis, stochastic modelling, temporal logics, statistical model checking.

[a] Work partially supported by EU-FET project QUANTICOL (nr. 600708) and by FRA-UniTS..

[b] Work supported by European Research Council under grant MLCS 306999.

DOI:10.2168/LMCS-11(2:3)2015

sampling from stochastic processes. While the predictive power of stochastic models is a key to their success in scientific applications, the development of algorithms and methodologies to reason about stochastic models has been a consistent focus of research in theoretical computer science over the past five decades. Of particular importance in verification, and for this paper, is (stochastic) model checking: given a property (formalised as a formula in a suitable logic), estimate the probability that it is satisfied by a random trajectory of the model [3].

Model checking tools, either numerical or statistical, however, can only be deployed if a model is fully specified (or, at least, if sample trajectories can be computed effectively). This requirement is often conceptually and practically untenable in many situations: modelling is the result of a mathematical formalisation of scientific expertise, and while such expertise is often able to define suitable model structures, it is implausible to expect to be able to pin-point uniquely defined values for the many parameters which are involved in many complex models. The increasing awareness of this limitation has motivated considerable research in statistical machine learning and systems engineering; while parameter synthesis is still an open research question, there are several approaches which estimate the parameters of a stochastic process from observations of the state of the modelled system. These approaches assume that (noisy) observations of the actual state of the system are available, usually in the form of time series [2, 34].

In this paper, we shift the focus from observations of the state of the system to observations of the *emergent properties* of the system: we assume to observe *truth values* or *satisfaction probabilities* of logical formulae over sample trajectories of the system, and use such data to identify the parameters of the stochastic process. The rationale for exploring this problem, that to our knowledge has not been extensively studied (see below for related work), is three-fold: in the first instance, in many applications gathering and storing (multiple) time series data is difficult and expensive, while qualitative global properties (e.g. phenotypes in a biological application) may be more readily available. Secondly, learning a model from logical constraints more closely matches the modelling process itself: generally a suitable model is chosen to capture some qualitative behaviour of the system (e.g. a negative feedback loop for an oscillator); it is therefore natural to also attempt to recover plausible parametrisations from such data. Thirdly, this approach illustrates the close relationship between the system identification and the system design problem: one could equally well imagine the satisfaction probabilities to be not the result of observations, but requirements set out by a user which need to be matched.

Solving these problems presents considerable computational and statistical challenges: in order to define a suitable objective function for parameter optimisation (e.g. a likelihood function), one needs to be able to explicitly determine the functional dependence of satisfaction probabilities on the parameters, which is impossible in all but the simplest cases. One can however obtain an approximate estimate of this likelihood at specific parameter values by using a Statistical Model Checking (SMC) procedure. This enables us to leverage a powerful class of machine learning algorithms for optimising unknown (but computable) functions, Bayesian Optimisation. Within the Bayesian Optimisation family, we select a provably convergent, recently developed global optimisation algorithm, the Gaussian Process Upper Confidence Bound (GP-UCB) optimisation algorithm. We show that this approach is effective and accurate in both the system identification and the system design problems on a range of non-trivial examples.

The rest of the paper is organised as follows: we start by briefly recapitulating the fundamental notions about stochastic processes and temporal logics. We then introduce the main methodological tools underpinning our approach. The approach is evaluated on a number of model examples, including continuous-time Markov chains and hybrid stochastic systems. We conclude by discussing the merits and limitations of this approach.

Related work. This paper grows out of a conference paper of the same title [13]. While the core idea is the same, it is extended in several directions: we now apply the methodology to a broader class of stochastic processes (including SDEs and hybrid models), we improve the algorithm by incorporating a hyper-parameter optimisation routine, we provide approximate estimates of the uncertainty over the optimal parameters, and we devise methodology to handle the non-homogeneous nature of the noise in SMC. Furthermore, the paper is completed by a new experimental section on different examples. Within the recent literature, earlier attempts were made to use model checking methods for parameter estimation in [22]; while the underlying idea of constraining a model with logical properties is shared, the quantitative semantics we employ here and the more powerful algorithmic solutions lead to considerable differences. Also related is the idea of model repair [8], whereby the parametrisation of an original model is locally modified to increase the satisfaction probability of a logical property. However, this approach is based on parametric model checking [26], which heavily suffers from state space explosion.

Optimisation methods can be fruitfully employed in other formal modelling scenarios: [6] uses similar algorithmic procedures to optimise the robustness with which a formula is satisfied, while [7] attacks the converse problem of identifying properties with high satisfaction probability within a parametric family of formulae (given a fixed model).

Within the machine learning literature, [20] has developed novel approximation techniques to solve the problem of *Bayesian inference* from (continuous-time) constraints on trajectories. The considerably harder nature of this problem (involving estimation of a whole posterior process, as opposed to just the parameters) implied however that only a very restricted class of models and constraints could be considered in that paper.

2. Background

In this section, we provide a brief introduction to the fundamental mathematical and logical concepts underpinning our approach. We will start in Section 2.1 by briefly recalling the broad class of systems we consider. We then introduce in Section 2.2 the logical formalism in which system properties will be encoded, namely Metric Interval Temporal Logic (MiTL). We stress that this particular choice of logic is not essential for our approach, which will work for any logic whose predicates are verified on individual, time bounded trajectories. Once these preliminaries are established, we will formally define the system identification problem (Section 3.1) and the system design problem (Section 3.2). In both cases, we limit ourselves to the problem of identifying parameters of a model with fixed structure, leaving structural identification to further work.

2.1. **Stochastic Processes.** Here we provide a quick and informal introduction to the classes of stochastic processes considered in this paper, briefly introducing the simulation algorithms used for drawing samples from them. The reader interested in a more thorough introduction is referred to standard textbooks like [23, 33, 16].

Let the state of the system at any one time $t \in [0, T]$ be defined by a *state variable* $\mathbf{V}$ taking values in a suitable measurable space $\mathcal{D}$. A *stochastic process* is a family of $\mathcal{D}$-valued random variables indexed by t; equivalently, this defines a measure over the space of trajectories of the system $\mathcal{T} = \{f \colon [0, T] \to \mathcal{D}\}$. Selecting a finite subset of indices $t_0, \ldots, t_N$, one obtains finite-dimensional random variables given by the configurations of the system at those times; the distribution of such random variables are the *finite-dimensional marginals* of the process. The process is *Markovian* if, given any finite set of state values $\mathbf{V}(t_1), \ldots, \mathbf{V}(t_N)$, the finite-dimensional joint marginal distribution factorises as

$$p\left(\mathbf{V}(t_1), \ldots, \mathbf{V}(t_N)\right) = p\left(\mathbf{V}(t_1)\right) \prod_{j=2}^{N} p\left(\mathbf{V}(t_j) | \mathbf{V}(t_{j-1})\right). \tag{2.1}$$

The conditional probability $p\left(\mathbf{V}(t + \delta t) | \mathbf{V}(t)\right)$ is usually termed transition probability; its derivative (transition rate) is called the generator of the Markov process. We will assume that the parametric dependence of the system is contained in the generator of the Markov process, and that the generator does not explicitly depend on time (time homogeneous process). The Markov property implies that the transition probabilities satisfy deterministic differential equations which in general are known as *Chapman-Kolmogorov equations*. We will consider the following three types of Markovian stochastic processes:

- *Continuous-Time Markov Chains (CTMCs)* CTMCs are a common mathematical model of stochastic dynamical processes in many areas of science; they are Markovian stochastic processes with discrete state space (i.e. $\mathcal{D} \subset \mathbb{Z}^d$). We will adopt the population view of CTMCs [10], in which the state space is described by a collection of n integer-valued variables $\mathbf{V} = (V_1, \ldots, V_n)$, describing the number of entities in each population of the model, and will borrow the notation of chemical reactions [40] to describe CTMCs. The transition probability of a CTMC obeys the *Chemical Master Equation* (CME), a (potentially infinite) set of coupled ordinary differential equations. The CME cannot be solved in all but the simplest cases; however, an exact algorithm, Gillespie's Stochastic Simulation Algorithm, exists to draw samples from a (time homogeneous) CTMC [25].
- *Stochastic Differential Equations (SDEs)* SDEs [33] define stochastic processes with continuous state space (usually $\mathcal{D} = \mathbb{R}^n$) and continuous (but nowhere differentiable) trajectories. We can think of SDEs as ordinary differential equations associated with a vector field which is randomly perturbed at each point by a white noise process. SDEs play an important role in science and engineering; in recent years, they have attracted considerable attention in computer science as fluid approximations to CTMCs. We will confine ourselves to Itô SDEs, which can be written as

  $$d\mathbf{V} = F(\mathbf{V})dt + G(\mathbf{V})d\mathbf{W},$$

  where $\mathbf{W}$ is a d-dimensional Wiener process (whose derivative is known as white noise), F is the n-dimensional *drift* function and G is the $n \times d$ diffusion matrix. SDEs can be simulated using a variety of numerical schemes; here we will use the Euler-Maruyama scheme, which fixes a time step h and iteratively computes $\mathbf{v}(t+h) = \mathbf{v}(t) + F(\mathbf{v}(t))h + G(\mathbf{v}(t))\mathcal{N}(0, hI_d)$, where $\mathcal{N}(\mathbf{0}, hI_d)$ is a d-dimensional Gaussian random variable with mean $\mathbf{0}$ and diagonal

covariance matrix hI_d. It is important to notice that, in contrast to the CTMC case, this simulation procedure is no longer exact, but introduces an error which reduces to zero only in the $h \to 0$ limit.

- *Stochastic Hybrid Systems (SHS)* More generally, one may also consider stochastic processes with hybrid state space, i.e. $\mathcal{D} \subset \mathbb{Z}^d \times \mathbb{R}^D$. These may arise e.g. as approximations to CTMCs where some of the populations have large numbers (which can be well approximated as continuous variables) while others have sufficiently small numbers to require a discrete treatment [14, 12, 32]. The models so obtained are known as stochastic hybrid systems (SHS) [16], and their dynamics can be seen as a sequence of discrete jumps, instantaneously modifying population variables, interleaved by periods of continuous evolution along a trajectory of the SDE. As the rates of discrete transitions can depend on the continuously evolving variables, and vice versa, these systems can exhibit rich dynamics; nonetheless, the Markovian nature of the SHS still implies that effective (approximate) simulation algorithms can be obtained, see for instance [36].

2.2. **Metric interval Temporal Logic.** We will consider properties of stochastic trajectories specified by Metric interval Temporal Logic (MiTL), see [1, 30]. This logic is a linear temporal logic, so that the truth of a formula can be assessed over single trajectories of the system. MiTL, in particular, is used to reason on real-time systems, like those specified by CTMC or SDEs. Here we consider the fragment of MiTL in which all temporal operators are all time-bounded; this choice is natural in our context, as we want to compare a model with properties of experimental observations of *single time-bounded realisations* (essentially, time-bounded samples from its trajectory space).

The syntax of MiTL is given by the following grammar:

$$\varphi ::= \mathtt{tt} \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{U}_{[T_1,T_2]} \varphi_2,$$

where $\mathtt{tt}$ is the true formula, conjunction and negation are the standard boolean connectives, and there is only one temporal modality, the time-bounded until $\mathbf{U}_{[T_1,T_2]}$, where $T_1 < T_2$ are the time bounds. Atomic propositions μ are defined like in Signal Temporal Logic (STL [30]) as boolean predicate transformers: they take a real valued function $\mathbf{v}(t)$, $\mathbf{v} : [0, T] \to \mathbb{R}^n$, as input, and produce a boolean signal $s(t) = \mu(\mathbf{v}(t))$ as output, where $s : [0, T] \to \{\mathtt{tt}, \mathtt{ff}\}$. As customary, boolean predicates μ are (non-linear) inequalities on vectors of n variables, that are extended point-wise to the time domain. Temporal modalities like time-bounded eventually and always can be derived in the usual way from the until operator: $\mathbf{F}_{[T_1,T_2]}\varphi \equiv \mathtt{tt}\mathbf{U}_{[T_1,T_2]}\varphi$ and $\mathbf{G}_{[T_1,T_2]}\varphi \equiv \neg\mathbf{F}_{[T_1,T_2]}\neg\varphi$.

A MiTL formula is interpreted over a real valued function of time $\mathbf{v}$, and its satisfaction relation is given in a standard way, see e.g. [1, 30]. We report here the semantic rules for completeness:

- $\mathbf{v}, t \models \mu$ if and only if $\mu(\mathbf{v}(t)) = \mathtt{tt}$;
- $\mathbf{v}, t \models \neg\varphi$ if and only if $\mathbf{v}, t \not\models \varphi$;
- $\mathbf{v}, t \models \varphi_1 \wedge \varphi_2$ if and only if $\mathbf{v}, t \models \varphi_1$ and $\mathbf{v}, t \models \varphi_2$;
- $\mathbf{v}, t \models \varphi_1 \mathbf{U}_{[T_1,T_2]} \varphi_2$ if and only if $\exists t_1 \in [t + T_1, t + T_2]$ such that $\mathbf{v}, t_1 \models \varphi_2$ and $\forall t_0 \in [t, t_1]$, $\mathbf{v}, t_0 \models \varphi_1$[1].

[1]For the semantics of the until, we require that at time t_1, both φ_1 and φ_2 are true, following the treatment of STL [30].

A MiTL formula φ can be verified [30] over a real valued function $\mathbf{v}$ by first converting $\mathbf{v}$ into a vector of boolean signals $\mu_j(\mathbf{v}(t))$, where μ_j are all the atomic predicates appearing in φ, and then processing these signals bottom up from the parse tree of the formula[2]. A standard assumption is that the so obtained boolean signals change truth value only a finite number of times in $[0, T]$ (finite variability).

The temporal logic MiTL can be easily extended to the probabilistic setting, and interpreted over CTMC or other stochastic models like SDE or SHS [27, 18]. Essentially, the quantity of interest is the path probability of a formula φ, defined as[3]

$$p(\varphi) = p\left(\{\mathbf{v}_{0:T} | \mathbf{v}_{0:T}, 0 \models \varphi\}\right),$$

i.e. as the probability of the set of time-bounded trajectories that satisfy the formula[4]. Here $\mathbf{v}_{0:T}$ denotes a trajectory $\mathbf{v}$ restricted to the time interval $[0, T]$.

Note that trajectories of a CTMC with bounded transition rates, or more generally non-explosive [23], will always enjoy the finite variability property, as they are piecewise constant and their number of jumps is finite in $[0, T]$ with probability one. A more complex argument, based on first passage times for Brownian motion, can be employed to show that trajectories of SDEs and SHS also have finite variability [23].

3. Problem definition

We give here a precise definition of the two related problems we set out to solve in this work.

3.1. System identification.

Consider now a stochastic process $\mathbf{V}$ depending on a set of parameters θ, and a set of d MiTL formulae $\boldsymbol{\varphi} = \{\varphi_1, \ldots, \varphi_d\}$. We assume that the truth values of the d formulae have been observed over N independent runs of the process and gather the observations in the $d \times N$ *design (or data) matrix* D. Given a specific value of the parameters θ, the probability of observing the design matrix, $p(D|\theta)$, is uniquely determined, and can be computed by model checking (possibly using a randomized algorithm such as SMC). The *system identification problem* addresses the inverse problem of finding the value(s) of parameters θ which best explain the observed design matrix.

The key ingredient in the identification of probabilistic systems is the *likelihood*, i.e. the probability of observing the data matrix D for a given set of parameters θ; under the assumptions that observations are independent and identically distributed the likelihood factorises as the product of the probabilities of observing the individual columns of the design matrix

$$\mathcal{L}(D, \theta) = \prod_{i=1}^{N} p(D_i|\theta). \tag{3.1}$$

The *system identification problem* then corresponds to finding the parameter configuration θ^* that maximises the likelihood (3.1) (*maximum likelihood*, ML). Equivalently, we can

[2]Notice that the algorithm for monitoring a logic formula is irrelevant for the methodology presented in this paper, which only relies on the availability of boolean qualitative data.

[3] We use the notation $p(x)$ to denote the probability density of x of x, while $p(x|y)$ is used for the conditional probability density of x given y.

[4]We assume implicitly that T is sufficiently large so that the truth of φ at time 0 can always be established from $\mathbf{v}$. The minimum of such times can be easily deduced from the formula φ, see [27, 30]

maximise the logarithm of $\mathcal{L}(D, \theta)$, the so called log-likelihood, as is common practice in statistics (the result is the same due to monotonicity of the logarithm).

If prior knowledge over the parameters is available as a prior distribution $p(\theta)$ on the space of parameters Θ, we can consider the un-normalised posterior distribution

$$p(\theta, D) \propto p(\theta) \prod_{i=1}^{N} p(D_i|\theta). \tag{3.2}$$

and alternatively seek to maximise this quantity, giving rise to *maximum a posteriori* (MAP) estimation.

3.2. **System Design.** Consider again d MiTL formulae $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_d)$ and a stochastic process $\mathbf{V}(t)$ depending on parameters θ. We fix a *target probability table* P for the joint occurrence of the d formulae. The *system design problem* then consists of determining the parameters of the stochastic process which optimally match these probabilities.

This problem is intimately linked to system identification: in fact, one could characterise system design as *inference with the data one would like to have* [4]. In our case, we are given a probability table for the joint occurrence of a number of formulae $\varphi_1, \ldots, \varphi_N$.[5] However, in the design case, we do not aim to use this function to estimate the likelihood of observations, rather to match (or be as near as possible to) some predefined values. We therefore need to define a different objective function that measures the distance between two probability distributions; we choose to use the Jensen-Shannon Divergence (JSD) due to its information theoretic properties and computationally good behaviour (being always finite) [19]. This is defined as

$$JSD(p\|q) = \frac{1}{2} \sum_i \left[p_i \log \frac{2p_i}{p_i + q_i} + q_i \log \frac{2q_i}{p_i + q_i} \right]$$

where p and q are two probability distributions over a finite set. The Jensen-Shannon divergence is symmetric and always non negative, being zero if and only if $q = p$. Hence, system design corresponds to finding the parameter configuration θ^* that minimises the JSD between the target probability distribution P and the joint probability distribution $p(\boldsymbol{\varphi}|\theta)$ of the formulae $\boldsymbol{\varphi}$. Notice that our approach requires the specification of a full joint probability distribution over the truth values of multiple formulae; should such a level of specification not be required, i.e. only some probability values need to be matched, the remaining values can be filled arbitrarily, compatibly with normalisation constraints.

4. Methodology

Solving the system design and system identification problems requires us to optimise a function depending on the joint probability distribution of the satisfaction of d-input formulae $\varphi_1, \ldots, \varphi_d$, which has to be computed for different values of the model parameters θ. As numerical model checking algorithms for MiTL formulae suffer severely from state space explosion [18], we will revert to *statistical model checking* (SMC), which will be introduced in Section 4.1. While SMC provides a feasible way to estimate the joint satisfaction probability, it remains a computationally intensive method, providing only *noisy* estimates. A possible

[5] This problem formulation is different from a recent approach on parameter synthesis for CTMC using SMC, [28], in which the authors look for a subset of parameters in which a single formula φ is satisfied with probability greater than q.

solution, relying on the fact that estimation noise will be approximately Gaussian due to the Central Limit Theorem, is to adopt a Bayesian viewpoint: we can treat the unknown function as a random function (arising from a suitable prior stochastic process) and the numerical estimations based on SMC as (noisy) observations of the function value, which in turn enable a *posterior* prediction of the function values at new input points. This is the idea underlying *statistical emulation* [29], and leads to a very elegant algorithm for optimisation. This framework will be introduced in Sections 4.2 and 4.3. We will conclude discussing a first example based on the Poisson process, for which we can compare the numerical results against analytical formulae.

4.1. **Statistical model checking.** We briefly review the estimation of the probability of MiTL formulae by Statistical Model Checking (SMC [42, 43, 27]). Given a stochastic process with fixed parameters θ, a simulation algorithm is used to sample trajectories of the process. For each sampled trajectory, we run a model checking algorithm for MiTL (for instance, the offline monitoring procedure of [30]), to establish whether φ is true or false, thus generating samples from a Bernoulli random variable Z_φ, equal to 1 if and only if φ is true. SMC uses a statistical treatment of those samples, like Wald sequential testing [43] or Bayesian alternatives [27], to establish if the query $P(\varphi|\theta) > q$ is true, with a chosen confidence level α, given the evidence seen so far. Bayesian SMC, in particular, uses a Beta prior distribution $Beta(q|a, b)$ for the probability of $q = P(\varphi = 1)$; by exploiting the conjugacy of the Beta and Bernoulli distributions [9], applying Bayes' theorem we get

$$P(q|D_\varphi) = \frac{1}{P(D_\varphi)} P(D_\varphi|q)P(q) = Beta(q, a + k_1, b + k_0),$$

where D_φ is the simulated data, k_1 is the number of times $Z_\varphi = 1$ and k_0 the number of observations of 0. The parameters a and b of the Beta prior distribution (usually set to 1) can be seen as pseudo-counts that regularise the estimate when a truth value is rarely observed. Our best guess about the true probability $P(Z_\varphi = \texttt{tt})$ is then given by the predictive distribution [9]: $P(Z_\varphi = \texttt{tt}|D_\varphi) = \mathbb{E}[q|D_\varphi] = \frac{k_1+a}{k_1+a+k_0+b}$.

The Bayesian approach to SMC, especially the use of prior distributions as a form of regularization of sampled truth values of formulae, is particularly relevant for our setting, since we need to estimate probabilities over 2^d joint truth values of d formulae, i.e. we need to sample from a discrete distribution $Z_{\varphi_1,\ldots,\varphi_d}$ with values in $\mathcal{D} = \{\texttt{tt}, \texttt{ff}\}^d$. Some of these truth combinations will be very unlikely, hence regularization is a crucial step to avoid errors caused by keeping reasonably small the number of runs. In order to extend Bayesian SMC to estimate the joint truth probabilities of d formulae, we choose a Dirichlet prior distribution, which is a distribution on the unit simplex in $\mathbb{R}^n$, so that it can be seen as the multidimensional extension of the Beta distribution, and it also enjoys the conjugate prior property. The Dirichlet distribution has density

$$Dirichlet(\mathbf{q}|\alpha_1, \ldots, \alpha_{2^d}) \propto \prod_{i=1}^{2^d} q_i^{\alpha_i - 1}$$

depending on 2^d parameters α_i, which can be seen as pseudo-counts, and which we fix to one.[6] Given observations $D_{\varphi_1,\dots,\varphi_d}$ of the truth values of $Z_{\varphi_1,\dots,\varphi_d}$[7], analogous calculations yield the posterior distribution over multinomial distributions on $\mathcal{D}$ as $p(\mathbf{q}|D_{\varphi_1,\dots,\varphi_d}) = Dirichlet(\mathbf{q}|\alpha_1 + k_1, \dots, \alpha_{2^d} + k_{2^d})$, where k_j is the number of times we observed the jth truth combination, corresponding to a point $\mathbf{d}_j \in \mathcal{D}$. Using the fact that the marginals of the Dirichlet distributions are Beta distributed, the predictive distribution is readily computed as $p(Z_{\varphi_1,\dots,\varphi_d} = \mathbf{d}_j|D_{\varphi_1,\dots,\varphi_d}) = (\alpha_j + k_j)/(\alpha_0 + k)$. This probability is then used to estimate the likelihood $\mathcal{L}(D,\theta)$, as $\mathcal{L}(D,\theta) = \prod_{i=1}^{N} P(D_i|\theta)$ or the JSD. By the law of large numbers, with probability one, this quantity will converge to the true likelihood when the number of samples in the SMC procedure becomes large, and the deviation from the true likelihood will become approximately Gaussian.

4.2. **Gaussian Process Regression.** A *Gaussian Process* (GP) is a probability measure over the space of continuous functions (over a suitable input space) such that all of its finite-dimensional marginals are multivariate normal. A GP is uniquely defined by its *mean* and *covariance* functions, denoted by $\mu(x)$ and $k(x,x')$. By definition, we have that for every finite set of points

$$f \sim \mathcal{GP}(\mu, k) \leftrightarrow \mathbf{f} = (f(x_1), \dots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K) \tag{4.1}$$

where $\boldsymbol{\mu}$ is the vector obtained evaluating the mean function μ at every point, and K is the matrix obtained by evaluating the covariance function k at every pair of points. In the following, we will assume for simplicity that the prior mean function is identically zero (a non-zero mean can be added post-hoc to the predictions w.l.o.g.).

The choice of covariance function determines the type of functions which can be sampled from a GP (more precisely, it can assign prior probability zero to large subsets of the space of continuous functions). A popular choice of covariance function is the *radial basis function* (RBF) covariance

$$k(x,x') = \gamma \exp\left[-\frac{\|x - x'\|^2}{\lambda^2}\right] \tag{4.2}$$

which depends on two hyper-parameters, the *amplitude* γ and the *lengthscale* λ. Sample functions from a GP with RBF covariance are with probability one infinitely differentiable functions. GPs are a very natural framework for carrying out the *regression task*, i.e. estimating a function from observations of input-output pairs. Noisy observations of function values (a *training* set) can be combined with a GP prior to yield Bayesian posterior estimates of the function values at novel query input values. If the observation noise is Gaussian (as is the case we consider in this paper), the required computations can be performed analytically to yield a closed form for the predictive posterior.

Assuming for simplicity a zero prior mean function, we have that the predictive distribution at a new input x^* is Gaussian with mean

$$\mu^* = (k(x^*, x_1), \dots, k(x^*, x_N))\, \hat{K}_N^{-1} \mathbf{y} \tag{4.3}$$

and variance

$$k^* = k(x^*, x^*) - (k(x^*, x_1), \dots, k(x^*, x_N))\, \hat{K}_N^{-1}\, (k(x^*, x_1), \dots, k(x^*, x_N))^T. \tag{4.4}$$

[6]The corresponding Dirichlet distribution boils down to a uniform distribution.

[7]Note that $D_{\varphi_1,\dots,\varphi_d}$ is a matrix, similarly the design matrix discussed in Section 3, but we treat each column/ observation as a single point of $\mathcal{D}$.

Here, $\mathbf{y}$ is the vector of observation values at the training points $x_1, \ldots, x_N$ and

$$\hat{K}_N(i,j) = k(x_i, x_j) + \delta_{ij}\sigma_i^2$$

with σ_i^2 the observation noise variance at point x_i (see below section 5.2 for how this quantity is estimated in our case). Notice that the first term on the r.h.s of equation (4.4) is the prior variance at the new input point; therefore, we see that the observations lead to a *reduction* of the uncertainty over the function value at the new point. The variance however returns to the prior variance when the new point becomes very far from the observation points.

GPs are a rich and dynamic field of research in statistical machine learning, and this quick introduction cannot do justice to the field. For more details, we refer the interested reader to the excellent review book of Rasmussen and Williams [35].

4.3. **Bayesian optimisation.** We now return to the problem of finding the maximum of an unknown function with the minimum possible number of function evaluations. The underlying idea of *Bayesian Optimisation* (BO) is to use a probabilistic model (e.g. a GP) to estimate (with uncertainty) a statistical surrogate of the unknown function (this is sometimes called emulation in the statistics literature [29]). This allows us to recast the optimisation problem in terms of trade off between the *exploitation* of promising regions (where the surrogate function takes high values) with the *exploration* of new regions (where the surrogate function is very uncertain, and hence high values may be hidden).

Optimal trade-off of exploration and exploitation is a central problem in reinforcement learning, and has attracted considerable theoretical and applicative research. Here we use the GP Upper Confidence Bound (GP-UCB) algorithm [37], an exploration-exploitation trade-off strategy which provably converges to the global optimum of the function. The idea is intuitively very simple: rather than maximising the posterior mean function, one maximises an upper quantile of the distribution, obtained as mean value plus a constant times the standard deviation (e.g. the 95% quantile, approximately given as $\mu + 2\sigma$). The GP-UCB rule is therefore defined as follows: let $\mu_t(x)$ and $var_t(x)$ be the GP posterior mean and variance at x after t iterations of the algorithm. The next input point is then selected as

$$x_{t+1} = \text{argmax}_x \left[\mu_t(x) + \beta_t \sqrt{var_t(x)}\right] \tag{4.5}$$

where β_t is a constant that depends on the iteration of the algorithm. The importance of the work of [37] lies in the first proof of convergence for such an algorithm: they showed that, with high probability, the algorithm is *no-regret*, i.e.

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} ((x^*) - f(x_t)) = 0.$$

where x^* is the true optimum and x_t is the point selected with the UCB rule at iteration t.

Remark: The use of a GP with RBF covariance implicitly limits the set of possible emulating functions to (a subset of the set of) smooth functions. This is not a problem when optimising parameters of a CTMC: it was recently proved in [11] that the satisfaction probability of a MiTL formula over a CTMC is a smooth function of the model parameters, which immediately implies that the likelihood of truth observations is also smooth. In general, we conjecture that smoothness will hold for purely stochastic processes, i.e. systems where it is impossible to find a (strict) subset of state variables X_J such that the system dynamics conditioned on X_J are deterministic. It is easy to show that smoothness does not hold

for deterministic systems, where the satisfaction probability can jump from zero to one as the parameters are varied. In hybrid deterministic/ stochastic processes, smoothness may therefore not hold; in these cases, the algorithm will still execute, but its convergence guarantees will be lost, so that application of our method to this class of systems should be considered as heuristic.

4.4. **Example: Poisson process.** As a simple example illustrating our approach, we consider observing the truth values of an atomic proposition over realisations of a Poisson process. We briefly recall that a Poisson process with rate μ is an increasing, integer valued process such that

$$P(k = n|\mu, t) = \frac{(\mu t)^n}{n!} \exp[-\mu t]. \tag{4.6}$$

Poisson processes are fundamental in many applications, ranging from molecular biology to queueing theory, where they often form the basic building blocks of more complex models. We consider a very simple scenario where we have observed the truth value of the formula

$$\varphi = \mathbf{F}^{[0,1]}\{k > 3\},$$

i.e. the formula expressing the fact that k has become bigger than 3 within 1 time unit, evaluated on individual trajectories sampled from a process with $\mu = 2$. The probability of φ being true for a trajectory given the value of μ can be calculated analytically as

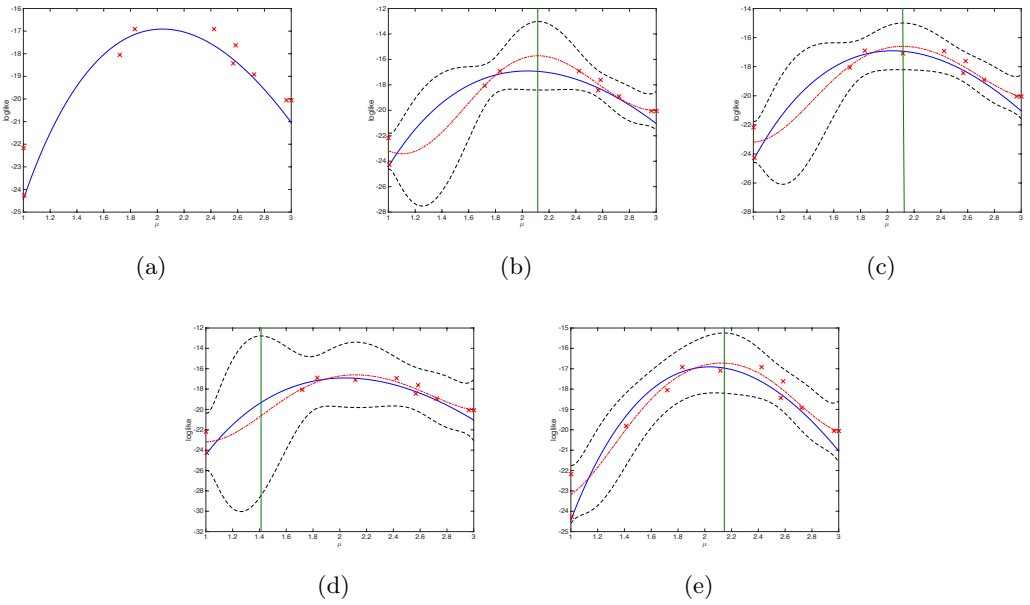
$$p(\mu) = P(\varphi = true) = 1 - P(\varphi = false) = 1 - \sum_{n=0}^{3} \frac{(\mu)^n}{n!} \exp[-\mu]. \tag{4.7}$$

This leads to the following analytical formula for the log-likelihood, given a fixed set of observations D of its truth:

$$\mathcal{L}(\mu, D) = \#_{true}(D) \log(p(\mu)) + \#_{false}(D) \log(1 - p(\mu)), \tag{4.8}$$

where $\#_{true}(D)$ counts the number of times the formula was observed true in D, and $\#_{false}(D)$ counts the occurences of $false$ in D. This gives us an ideal benchmark for our approach.

Figure 1 shows a generic step of the GP-UCB algorithm at work. The starting point is the set D, in this case containing 40 independent observations of process trajectories. The exact log likelihood is computed according to equation (4.8), and shown in Figure 1(a), together with 10 samples of the log-likelihood, computed by SMC (red dots). In Figure 1(b), we show the result of running GP-regression over these 10 observations. The predictive posterior mean is in red, while the dashed black lines represent the upper and lower confidence bounds of the distribution on functions defined by the posterior GP, for $\beta_t \equiv 2$. The vertical line is the maximum identified by the global search procedure needed to optimise the GP upper confidence bound. The log-likelihood is then sampled at this new point, again by SMC, and the GP regression is run again on such an enlarged input set. The result is shown in Figure 1(c). As can be seen, the variance of the prediction, i.e. the width of the upper confidence bound, has been considerably reduced in the region around the new input point for the GP regression task. In this case, however, the maximum of the upper confidence bound is not changed, hence we increase β_t from 2 to 4. The result is shown in Figure 1(d), where we can see that the maximum is now shifted on the right, on a high uncertainty region. The log-likelihood is sampled again at this newly identified point, and the result is a large reduction of variance for small μ, as can be seen in Figure 1(e).



(a) (b) (c)

(d) (e)

Figure 1: Illustration of the learning procedure on the Poisson process example. (**1(a)**) Exact log likelihood (computed from formula (4.8)) and 10 SMC estimation (from 100 simulation runs, red crosses) for $\mu \in [1,3]$. (**1(b)**) Illustration of the GP-UCB algorithm: GP likelihood estimation (red dash-dotted line), true likelihood (solid blue line), and GP-UCB confidence bounds (black dotted lines). The maximum of the upper confidence bound is identified by the vertical line. (**1(c)**) Illustration of the GP-UCB algorithm, after sampling the log-likelihood (by SMC estimation) at the maximum value previously identified. Lines have the same meaning as before. Notice the reduced variance near the new sampled point. The maximum, however is predicted at the same place. (**1(d)**) Illustration of the GP-UCB algorithm after increasing β_t from 2 to 4. Now the maximum of the upper confidence bound is in an area of high uncertainty, on the left. (**1(e)**) Illustration of the GP-UCB algorithm, after sampling the log-likelihood at the maximum identified in 1(d). Again, notice how the uncertainty is reduced.

5. Enhancing the methodology

We briefly discuss here some further improvements to the basic methodology presented in Section 4; an extensive tutorial introduction of the statistical concepts used is beyond the scope of this paper, however full details can be found in the referenced literature. The methodological enhancements are discussed in the context of system identification using the likelihood, but similar considerations apply for design or MAP identification.

5.1. **Laplace approximation.** The GP-UCB algorithm enables us to find the maximum of a function (in our case, the likelihood function or the un-normalised posterior); in many cases, however, it is very desirable to be able to provide uncertainty estimates over the parameter values returned. Given the intractable nature of the likelihood, which requires a

computationally expensive statistical model checking procedure at every parameter value, a fully Bayesian treatment (e.g. based on Markov chain Monte Carlo simulations) is ruled out. A cheaper alternative is to compute a local Gaussian approximation: this procedure, known as *Laplace approximation* in statistics and machine learning, approximates the uncertainty as the inverse of the local curvature at the optimum (see e.g. [9], Ch 4.4). This is equivalent to locally approximating the log-likelihood with a quadratic form. In our case, we cannot directly compute derivatives of the unknown likelihood function: instead, we compute a Laplace approximation to the GP mean function at the optimum. In practice, as the GP is only estimated on a discrete subset of points, we perform a local optimisation step around the GP-UCB solution (by using the Newton-Raphson algorithm applied to the posterior GP mean function), and compute the Hessian at the resulting maximum. The inverse of the Hessian matrix then provides a local approximation to the covariance structure of the estimated parameters: in particular, its diagonal entries can be used to provide confidence values over the estimated parameters. Naturally, the local properties of the GP posterior mean are influenced both by the true underlying function, but also on the hyper-parameters of the GP prior; we discuss below how such hyperparameters can be set automatically using an additional optimisation step.

5.2. **Heteroschedastic noise.** One drawback of the method introduced in Section 4 is that it assumes the observation noise to be uniform in the whole parameter space. This is an oversimplification, as the noise in the SMC estimation of the likelihood is heteroschedastic, i.e. it depends on the joint satisfaction probability and on the variability of the estimate. A non-homogeneous treatment of noise will reduce the variability in the estimation of the likelihood function.[8] We propose two approaches to compute the noise in the log-likelihood, the first one computational, based on bootstrapping, and the second one analytic, exploiting the nature of the posterior distribution of Bayesian SMC.

5.2.1. *Bootstrapping.* Bootstrapping is a standard statistical technique to obtain estimates of confidence intervals [9]. In our setting, it works by resampling with repetition from the set of observed joint truth values, and recomputing the log-likelihood from each sampled set. In this way, one obtains an empirical distribution of the log-likelihood, from which statistics and confidence intervals can be extracted. In our case, the bootstrapped statistic is the standard deviation of the empirical bootstrap distribution.

5.2.2. *Posterior estimate.* As an alternative to bootstrapping, we can exploit the fact that Bayesian SMC gives us a posterior distribution on the space of probability distributions over the joint truth value of d MiTL formulae $\varphi_1, \ldots, \varphi_d$. We recall that, assuming a Dirichlet prior with 2^d parameters $\boldsymbol{\alpha} = (\alpha_i)$, and if k_j simulations resulted in the truth value $\mathbf{d}_j$, then the posterior distribution is again Dirichlet with parameters $\boldsymbol{\alpha} + \mathbf{k}$.

A typical Bayesian treatment of noise is to compute the average distribution of the quantity of interest with respect to the posterior distribution, thus taking into account the full noise distribution. Recall that the likelihood can be seen as a function of $\mathbf{q}$, with $\mathbf{q} \sim \mathit{Dirichlet}(\boldsymbol{\alpha} + \mathbf{k})$, and let $\mathbf{h}$ be the vector counting truth value in the observations D, $h_j = \#(\mathbf{d}_j, D)$. Simple computations give $\mathbb{E}[L(\mathbf{q})] = \frac{B(\boldsymbol{\alpha}+\mathbf{k}+\mathbf{h})}{B(\boldsymbol{\alpha}+\mathbf{k})}$ and $VAR[L(\mathbf{q})] =$

[8]Notice that the only difference in GP regression is that now the covariance matrix of observation added to $\hat{K}_N$ is diagonal with non-constant elements on the diagonal.

$\frac{B(\boldsymbol{\alpha}+\mathbf{k}+2\mathbf{h})}{B(\boldsymbol{\alpha}+\mathbf{k})} - \frac{B^2(\boldsymbol{\alpha}+\mathbf{k}+\mathbf{h})}{B^2(\boldsymbol{\alpha}+\mathbf{k})}$, where $B(\mathbf{x}) = \frac{\prod_{i=1}^{2^d}\Gamma(x_i)}{\Gamma(\sum_{i=1}^{2^d} x_i)}$ is the multinomial Beta function and $\Gamma(x) = \int_0^\infty y^{x-1}e^{-y}dy$ is the gamma function.

5.3. **Optimisation of hyperparameters.** A delicate issue about GP-UCB optimisation is that the emulation of the log-likelihood or of the JSD depends on the choice of hyperparameters of the kernel, which in the case of the RBF Gaussian kernel are the amplitude α and the lengthscale λ. The problem of leaving this choice to the user is that the results of the optimisation and its computational complexity (number of log-likelihood evaluations) depend in unpredictable ways on these parameters, particularly on the lengthscale. In fact, the lengthscale governs the Lipschitz constant of the functions sampled from the GP, hence of the posterior prediction, especially at a low number of input points. It would be wise, therefore to try to estimate such hyperparameters from the batch of initial observations. There are two main approaches to do this [35]. One way is to take a Bayesian perspective and put a prior on hyperparameters, estimating their posterior distribution from observed data via Monte Carlo sampling, which is unfeasible in our setting. Alternatively, we can treat the estimation of hyperparameters as a model selection problem, which can be tackled in a maximum-likelihood perspective by optimising the model evidence

$$p(\mathbf{y}|X,\alpha,\lambda) = \int p(\mathbf{y}|\mathbf{f},X)p(\mathbf{f}|X,\alpha,\lambda)d\mathbf{f}.$$

Essentially, $p(\mathbf{y}|X,\alpha,\lambda)$ is the marginal likelihood of the observed data, computed by marginalising the product of the likelihood times the GP prior, and is a function of the hyperparameters for which an analytic expression can be derived [35]. The idea behind is that the larger this value, the better the hyperparameters, and hence the GP, explain the observed data. In this paper, we take this second approach. As the model evidence can potentially have multiple local maxima, we use a simple global optimisation scheme, running several times a Newton-Raphson local optimisation algorithm from random starting points [17]. Experimentally, we found that the model evidence tends to behave quite well, with a global optimum having a large basin of attraction, a phenomenon often observed in practice [35], so that few runs, on the order of five, of the optimisation routine suffice.

5.4. **Grid sampling strategies.** A final improvement of the algorithm we consider in this paper is related to the sampling strategies for the initial set of points at which the likelihood or the JSD is evaluated, and the sampling strategy for the points at which the emulated likelihood is computed to look for a maximum. The goal is to maximise the coverage of the parameter space, keeping the number of sampled points to a minimum. Simple but effective schemes in this respect are based on the *latin hypercube sampling* strategy (LHS) [31], which splits a d-dimensional cube into k^d smaller cubes, and samples k points, at most one for each smaller cube, with the constraint that two sampled points cannot belong to cubes that overlap when projected in any of the d dimensions. For $d = 2$, LHS samples a latin square, from which it derives the name. The sampling approach we use is a variation of LHS, called *orthogonal LHS*, which further subdivides the space into equally probable subspaces. Points sampled still satisfy the LHS property, with the further constraint that each subspace contains the same number of points, thus improving the coverage [38].

6. Experiments

In this section we will discuss two examples in more detail: a simple CTMC model of rumour spreading in a social network, which resembles the diffusion of an epidemics, and a more complex SHS model of the toggle-switch, a simple genetic network composed of two genes repressing each other that shows bistable behaviour.

6.1. **Rumour spreading.** The spreading or rumours or information in a social network is a phenomenon that has received a lot of attention since the sixties. Here we consider a simple model [21], in which agents are divided into three classes: those that have not heard the rumour, the *ignorants* (I), those that have heard the rumour and are actively spreading it, the *spreaders* (S), and those that have stopped spreading it, the *repressors* (R). The dynamics is given by three simple rules: when an ignorant comes into contact with a spreader, the rumour is transmitted at rate k_s, while when a spreader comes into contact with another spreader or with a repressor, it stops spreading the rumour. This happens at rate k_r. We further multiply those rates by the average degree of connectivity $\langle k \rangle$ in the social network, i.e. the average number of people one is in contact with. The use of the average degree corresponds to the hypothesis of a homogeneous social network, see [5]. Summarising, the model is a CTMC on three populations, V_I, V_S, and V_R, subject to three types of events, which in the reaction-rate style [25] are:

- $V_I + V_S \to V_S + V_S$, with rate function $a_s(\mathbf{V}, k_s, \langle k \rangle) = \frac{k_s \langle k \rangle}{N} \cdot V_S \cdot V_I$;
- $V_S + V_S \to V_R + V_S$, with rate function $a_{r1}(\mathbf{V}, k_r, \langle k \rangle) = \frac{k_r \langle k \rangle}{N} \cdot V_S \cdot V_S$;
- $V_S + V_R \to V_R + V_R$, with rate function $a_{r2}(\mathbf{V}, k_r, \langle k \rangle) = \frac{k_r \langle k \rangle}{N} \cdot V_S \cdot V_R$;

Notice the normalisation factor N (the total population), which corresponds to a density dependence assumption, i.e. to a constant rate of contact per person, which is then multiplied by the probability of finding a spreader or a repressor, assuming random neighbours, see [21, 5]. For this system, we considered four temporal logic properties, expressed as MiTL formulae, concerned with the number of spreaders and repressors, fixing the total population to 100. The properties are:

(1) $\mathbf{G}_{[0,200]}(V_S < 45)$: the fraction of spreaders never exceeds 45% in the first 200 time units. This bounds the number of active spreaders from above;
(2) $\mathbf{F}_{[22,40]}(V_S > 35)$: between time 22 and 40, the fraction of spreaders exceeds 35%. This locates the spreading peak between time 22 and 40;
(3) $(\mathbf{F}_{[65,90]}(V_s = 0)) \wedge (\mathbf{G}_{[0,65]}(V_S > 0))$: the spreading process stops between time 65 and 90, and is active before time 65.
(4) $\mathbf{G}_{[90,200]}(82 < V_R < 88)$: the fraction of repressors stabilises from time 90 to 200 at between 82% and 88%. This corresponds to the fraction of population having heard the rumour.

6.1.1. *Experimental Setup.* We fixed the average degree $\langle k \rangle$ to 20,[9] while the remaining parameters are explored. To test the method under different conditions, we sampled uniformly $k_s \in [0.8, 1.2]$ and $k_r \in [0.6, 1.0]$, and use the sampled configuration to generate 40 observations D of the value of the logical formulae. Then, we ran 20 times the GP-UCB optimisation algorithm in the following search space: $k_s \in [0.1, 10]$, $k_r \in [0.08, 8]$, so that

[9]Note that the average degree multiplies all rates, hence fixing it corresponds to fixing the time scale.

true k_s	mean k_s	median k_s	std dev k_s	true k_r	mean k_r	median k_r	std dev k_r
1.0313	1.048	1.0433	0.0714	0.6284	0.6308	0.626	0.0215
1.1674	1.2544	1.2536	0.0497	0.7481	0.7535	0.7565	0.0261
1.0806	1.0794	1.1052	0.1203	0.7775	0.813	0.834	0.0622
0.8112	0.7817	0.8071	0.1049	0.8332	1.0202	0.9075	0.3017
1.1231	1.0344	1.0357	0.041	0.9894	1.0086	1.0083	0.0385
0.8888	0.8265	0.8551	0.0794	0.9818	1.1193	1.0057	0.2257
1.0125	1.0382	1.0304	0.0459	0.6957	0.7363	0.7343	0.0271
1.0338	1.0422	1.0511	0.0469	0.6109	0.6052	0.6044	0.0273
0.9312	0.9096	0.9053	0.0296	0.7596	0.791	0.7831	0.0334
0.8606	0.7079	0.7083	0.081	0.8692	1.2647	1.1302	0.3749

Table 1: Results for the maximum likelihood learning problem for the rumours spreading model. We report mean, median, and standard deviation on 20 runs, for 10 different true parameter combinations.

each parameter domain spans over two orders of magnitude. To treat equally each order of magnitude, as customary we transformed logarithmically the search space, and rescaled each coordinate into $[-1, 1]$ (log-normalisation). The algorithm first computes the likelihood, using statistical model checking, for 48 points sampled randomly according to the orthogonal LHS strategy from the log-normalized space, and then uses the GP-UCB algorithm to estimate the position of a potential maximum of the upper bound function in a grid of 500 points, again sampled using orthogonal LHS. Noise is treated heteroschedastically, using bootstrapping, and the other hyperparameters are optimised after the computation of the likelihood for the initial points. If in the larger grid a point is found with value greater than those of the observation points, we run a local optimisation algorithm (a Newton-Raphson scheme) to find the exact local maximum nearby, and then compute the likelihood for this point and add it to the observations (thus changing the GP approximation). Termination happens when no improvement can be made after three grid resamplings. The algorithm terminated after only 10-15 additional likelihood evaluations on average.

Results are reported for ML and MAP, in this case using independent, vaguely informative Gamma priors, with mean 1 for k_s and 0.8 for k_r, and shape equal to 10. We also compare the effect of different enhancements, fixing the "true" parameter values to $k_s = 1.0$ and $k_r = 0.8$ and the 40 observations, and running 100 times the algorithm for each combination of features.

6.1.2. *Results.* Results for maximum likelihood are shown in Table 1, where we compare the true value of parameters, against the predicted mean, median, and standard deviation of the prediction in a batch of 20 runs. As we can see, the algorithm is able to reconstruct the true parameterisation with a good accuracy. As a metric to assess the quality, we consider the average observed error (euclidean distance from the true configuration), which is 0.131 for the data shown in Table 1, the average normalised error, obtained by dividing the absolute error by the diameter of the search space, which is 1.03%, and the mean relative error, i.e. the absolute error for each parameter divided by the true parameter and averaged over all parameters and runs, which equals 9.23%. In Table 2, instead, we similarly report the results for the maximum a posteriori estimate. In this case the average observed error is

true k_s	mean k_s	median k_s	std dev k_s	true k_r	mean k_r	median k_r	std dev k_r
0.8321	0.7897	0.7963	0.0497	0.9747	1.1186	1.0952	0.0946
1.1046	1.125	1.1325	0.0407	0.6566	0.6771	0.6758	0.0234
1.1541	1.1698	1.167	0.0412	0.6934	0.6944	0.699	0.0265
0.938	1.0129	1.0182	0.0409	0.7254	0.7618	0.7616	0.0255
0.9504	1.0646	1.0624	0.0357	0.719	0.7228	0.7191	0.0242
0.818	0.8414	0.8465	0.0342	0.6316	0.6599	0.6672	0.0296
1.1626	1.1685	1.1614	0.0628	0.6596	0.6332	0.6376	0.0224
1.0475	1.0098	1.0128	0.038	0.9136	0.8742	0.8665	0.0323
1.0174	0.9933	1.0023	0.0381	0.9257	0.9683	0.9734	0.0271
1.0666	1.039	1.0537	0.0463	0.8853	0.8837	0.893	0.045

Table 2: Results for the maximum a posteriori learning problem for the rumours spreading model. We report mean, median, and standard deviation on 20 runs, for 10 different true parameter combinations.

0.074, the average normalised error is 0.58%, and the average relative error is 5.12%. These results show that the use of (good) prior information can improve the performances of the algorithm, as expected, as its effect is to increase the likelihood near the optimal point and decrease it in other areas of the parameter space.

We also run some tests to check if and to what extent the enhancements of Section 5 are improving the search algorithm. To this end, we fixed the true parameter value to $k_s = 1.0$ and $k_r = 0.8$, we sampled 40 observations, and run the optimisation routine for 100 times, for each possible combination of the following three features: heteroschedastic noise estimation (with bootstrapping), hyperparameter optimisation, and orthogonal LHS. We then compared the distribution of the predicted parameters for each pair of combination of features, by running a Kolmogorov-Smirnoff 2 sample test, at 95% confidence level. The p-values of the tests are reported in Table 3. Data show that hyperparameter optimisation consistently and significantly improves the quality of the results. Heteroschedastic noise estimation has a milder effect (it is significant in 2 cases out of four), but it relieves the user from guessing the intensity of noise. Orthogonal sampling, instead, produces a significant improvement only in one case out of four. We can check the quality of results from the standard deviations of the predictions, shown in Table 4: the smallest standard deviation is obtained when both heteroschedastic noise and hyperparameter optimisation were turned on.

As a final test, we consider the effect of changing the set of observable formulae, restricting to two formulae only: the one constraining the extinction time of the gossiping process, and the one concerned with the final number of people knowing the rumour. The effect of this removal is dramatic, as can be seen from Figure 2, where we compare the emulated log-likelihood for the full case with the emulated log-likelihood for the two formulae case. While in the first case we have a clear peak standing out, in the second setting we have an U-shaped ridge of points of almost equivalent likelihood. Not surprisingly, in this case the algorithm can return one point from the ridge without much preference, increasing the variability of the outcome. This suggests that the choice of the logical observables is a crucial step of the method, and they should somehow capture and constrain the key features of the dynamics of the process. Further investigation on this relationship between logic

	001	010	011	100	101	110	111
000	$< \mathbf{10^{-5}}$	0.3439	$< \mathbf{10^{-5}}$	**0.0314**	$< \mathbf{10^{-5}}$	0.1400	$< \mathbf{10^{-5}}$
001		$< \mathbf{10^{-5}}$	**0.0082**	$< \mathbf{10^{-5}}$	0.3439	$< \mathbf{10^{-5}}$	0.6766
010			$< \mathbf{10^{-5}}$	0.2606	$< \mathbf{10^{-5}}$	0.8938	$< \mathbf{10^{-5}}$
011				$< \mathbf{10^{-5}}$	0.1930	$< \mathbf{10^{-5}}$	**0.0314**
100					$< \mathbf{10^{-5}}$	0.4431	$< \mathbf{10^{-5}}$
101						$< \mathbf{10^{-5}}$	0.4431
110							$< \mathbf{10^{-5}}$

Table 3: P-values for the two sample Kolmogorov-Smirnoff test for the comparison of the different enhancements discussed in Section 5. We compared the predicted values of parameter k_s. Results for k_r are similar. The three-digit labels of rows and columns refer to the presence (1) or absence (0) of a specific feature in the optimisation. The first digit from the left is the estimation of heteroschedastic noise, the second is the orthogonal grid sampling, the third is the hyperparameter optimisation. Significant values at 95% confidence are in bold.

	000	001	010	011	100	101	110	111
$std(k_s)$	0.0319	0.0203	0.0333	0.0197	0.0315	0.0200	0.0312	0.0196
$std(k_r)$	0.0338	0.0203	0.0352	0.0196	0.0320	0.0202	0.0326	0.0196

Table 4: Standard deviations of predicted parameter values for the comparison of the different enhancements discussed in Section 5. The three-digit labels of columns are as in the caption of Table 3.

and identifiability, in the light of identifying a minimal set of properties that can describe a model, is a promising future research direction.

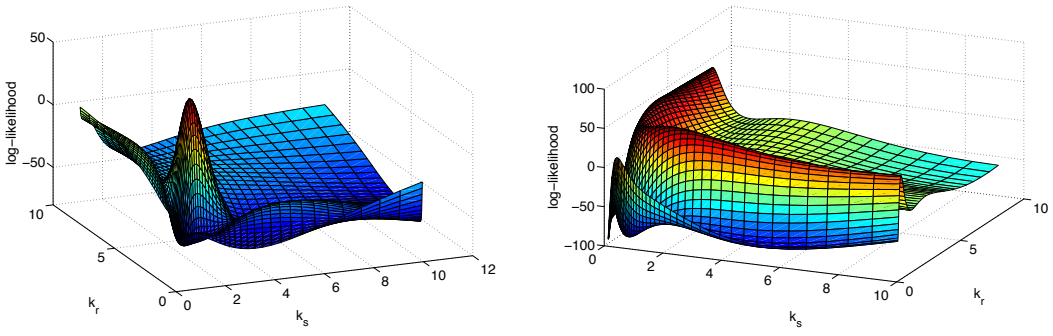


Figure 2: Comparison of estimated log-likelihood surfaces for two scenarios with true parameters fixed to $k_s = 1.0$, $k_r = 0.8$. Left: we observe all four formulae. Right: we observe only two logical properties, the one related to extinction and the one on the stability of repressors.

6.2. **Genetic Toggle-Switch.** We consider now a model of a simple genetic network implementing a toggle-switch, i.e. a form of local memory [24, 39]. The gene circuit is composed of two genes G_1 and G_2, expressing proteins X_1 and X_2 that act as mutual repressors: X_1 represses G_2 and X_2 represses G_1. For certain values of the parameter space, this circuit has two stable states, one in which the first protein is expressed and the second is not, and the symmetric one. Internal noisy fluctuations or external stimuli, like an increase in temperature, can force the system to jump from one stable state to the other. Hence, a proper treatment of stochastic behaviour is fundamental to properly understand (and design) this kind of circuit. The model we consider here follows the approach of [32], and describes the genetic network as a stochastic hybrid system, where genes are modelled as a two-state telegraph processes, while proteins are represented as continuous species, subject to a noisy continuous evolution given by an SDE with drift modulated by the state of the gene. More specifically, protein X_i evolves according to the SDE

$$dX_i = (\lambda_i G_i - \mu_i x_i)dt + \sigma_i dW_i,$$

while gene G_i changes from state $G_i = 1$ to $G_i = 0$ with rate $f_i^- = k_i \exp(\alpha_i X_j)$, $j \neq i$, and jumps back to the active state with constant rate c_i. The toggle-switch genetic network is known to be bistable. The two stable equilibria correspond to one protein expressed and the other not expressed. Hence, we consider four MiTL formulae, two per protein, expressing the active and inactive status. Furthermore, we require the protein to remain active or inactive for some time. Specifically, the formulae we consider are

(1) $\mathbf{F}_{[0,T]}\mathbf{G}_{[0,T_1]}\mathbf{F}_{[0,T_2]}X_i \geq th_{high}$, expressing the fact that between time $[0, T]$ the system stabilises for T_1 time units in a state in which $\mathbf{F}_{[0,T_2]}X_i \geq th_{high}$ holds, i.e. a state such that protein X_i is always found above th_{high} within additional T_2 time units.[10]
(2) $\mathbf{F}_{[0,T]}\mathbf{G}_{[0,T_1]}X_i \leq th_{low}$, expressing the fact that X_i remains inactive (below th_{low}) for T_1 time units.

If all formulae are true for both proteins, we are in a situation in which the process jumps from one stable state to the other during its observed life span of T time units.

6.2.1. *Experimental Setup.* We consider a scenario in which genes are symmetric, having a total of six parameters: two for the protein dynamics (λ and μ), three for the telegraph process (k, c, and α), and one for the noise (σ). In the experiments, we decided to fix the production rate $\lambda = 2$ and the degradation rate $\mu = 0.01$, exploring the other four parameters. As for the rumour spreading model, we consider 40 observations, generated from random parameters values sampled uniformly according to $k \in [0.08, 0.12]$, $c \in [0.03, 0.07]$, $\alpha \in [0.08, 0.12]$, and $\sigma \in [0.8, 1.2]$. For each of the 5 parameter combinations generated, we run 6 times the optimisation of the log-likelihood. Parameters were searched in the space $k \in [0.01, 1.0]$, $c \in [0.005, 0.5]$, $\alpha \in [0.01, 1.0]$, and $\sigma \in [0.1, 10]$, after log-standardising parameter ranges. Parameters of the formula where set as follows: $th_{low} = 20$, $th_{high} = 80$ (the average concentration of a protein in absence of regulation is 200), $T = 7000$, $T_1 = 1000$, and $T_2 = 200$. In particular, notice that we look at a very long temporal window, and require properties to hold for a long time. We run the GP-UCB optimisation starting from 96 points sampled using the orthogonal LHS scheme, and we evaluated the emulated function

[10]Notice that we do not require the protein to remain constantly above th_{high}, because we will choose a large value for the threshold, and the noisy evolution will easily make the protein fall below th_{high} in T_1 time units.

on a random grid of 1024 points, again sampled according to LHS. We used bootstrapping estimation of (heteroschedastic) noise, and we optimised hyperparameters at each run.

6.2.2. *Results.* The results for the exploration of 4 parameters are reported in Table 5. We obtained an average distance from the true parameter configuration of 0.8537 and an average normalised distance of 0.0853. Inspecting the data in more detail, we can see that parameters are captured less accurately than in the rumour spreading case and that there is a remarkable variability between the simulation runs. To better validate the accuracy of the data, in Table 5 we also show the mean value of the standard deviation estimated by the Laplace method. If we consider the fraction of optimisation runs in which the true parameter value falls within the 95% confidence interval constructed using the Laplace approximation (data not shown, but derivable from Table 5), we can observe that the parameters captured more accurately are k and c, as they almost always fall within the 95% confidence interval. The estimate of α and σ, instead, are subject to a much larger variability. This is a sign of a rugged log-likelihood landscape. To understand the origin of such a behaviour and check if it is caused by an intrinsic lack of identifiability of some of the parameters, given the observed data, we rerun the optimisation on different subsets of two parameters, fixing the other two to a nominal value of $k = 0.1$, $c = 0.05$, $\alpha = 0.1$, and $\sigma = 1$. What we observed is reported in Figure 3, in which the left and middle charts show the estimated log-likelihood surface for k and α and for k and σ. In both cases, we see a ridge or multiple maxima aligned on a line parallel to the α or σ axes, of approximatively constant height. This basically shows that the system is largely insensitive to the precise value of α and σ, provided they remain within a reasonable range. On the other hand, k is predicted reasonably accurately. This is in agreement with [32], where insensitivity with respect to α has also been observed. In the right chart of Figure 3, instead, we show the estimated log-likelihood as a function of k and c, varying them in the region $k \in [0.01, 0.2]$, $c \in [0.005, 0.1]$. As we can see, there is a flat region in the upper left corner, corresponding to small values of k and c. This shows that k and c, for the current formulae, cannot be identified precisely, explaining the results in Table 5. Note that this region is nonetheless relatively small, and the Laplace approximation manages to capture, at least partially, the variability in the prediction. We note here that looking at the 2 dimensional landscape of the log-likelihood, for pairs of parameters, can be a potentially interesting direction to investigate, in order to get insights on parameter identifiability, and to infer possible relationships between parameters, also to reduce the search space. This can be also combined with sensitivity analysis, to identify the most relevant parameters to explore.

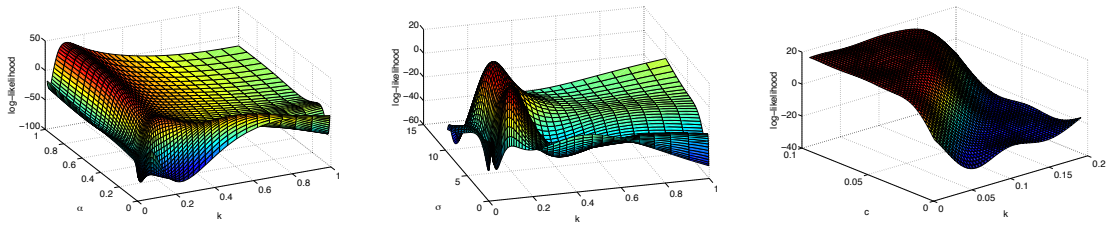


Figure 3: Emulation of log-likelihood as a function of k and α (left), k and σ (middle), and k and c (right) in the toggle switch example.

true k	mean $k \pm$ std	median k	Lap. sdt k	true α	mean $\alpha \pm$ std	median α	Lap. std α
0.09	0.035 ± 0.017	0.034	0.054	0.112	0.201 ± 0.167	0.12	0.058
0.103	0.034 ± 0.023	0.028	0.058	0.094	0.309 ± 0.233	0.247	0.067
0.099	0.067 ± 0.05	0.053	0.052	0.12	0.384 ± 0.321	0.257	0.071
0.107	0.045 ± 0.044	0.023	0.054	0.098	0.218 ± 0.262	0.082	0.044
0.085	0.018 ± 0.004	0.017	0.053	0.091	0.336 ± 0.176	0.308	0.066
true c	mean $c \pm$ std	median c	Lap. std c	true σ	mean $\sigma \pm$ std	median σ	Lap std σ
0.062	0.073 ± 0.034	0.066	0.054	1.2	1.168 ± 1.123	0.978	0.06
0.062	0.09 ± 0.107	0.056	0.074	0.834	1.512 ± 1.048	1.79	0.074
0.041	0.018 ± 0.015	0.012	0.056	0.837	1.134 ± 1.042	0.877	0.078
0.07	0.1 ± 0.048	0.088	0.066	1.161	0.817 ± 0.446	0.862	0.061
0.058	0.025 ± 0.01	0.022	0.056	1.044	0.795 ± 0.765	0.525	0.072

Table 5: Results for the maximum likelihood learning problem for the toggle switch model, for the joint optimisation of k, c, α and σ. We report mean plus/minus standard deviation, median, and mean standard deviation estimated by the Laplace method. We consider 6 runs, for 5 different true parameter combinations.

6.2.3. *System Design.* To test the performances of the method for the design problem, we consider again the toggle switch scenario, and the formulae described in the previous section, only for one protein. Notice that one formula describes a state in which the protein is expressed, while the other a state in which the protein is repressed. They can be both true in the same trajectory only if the system jumps from one stable state to the other within time $T = 7000$. In this experiment, we will try to force this phenomenon not to happen, at least for within time $[0, T]$. This can be obtained by a distribution of truth values putting mass 0.5 on the situation in which the first formula is true and the second is false, and 0.5 on the symmetric case. The choice of this distribution is due to the symmetry of the system, and the fact that we start simulations from a symmetric state, hence we expect a symmetry in the probability distribution of the truth of the two formulae. With this target probability in mind, we run the optimisation of the JSD exploring a space of three parameters: the production rate $\lambda \in [0.2, 20]$, the binding strength $k \in [0.01, 1]$, and the unbinding rate $c \in [0.01, 1]$. We run the optimisation 25 times, obtaining an average JSD of 0.0011. The probability distribution obtained match the targeted probabilities quite well: the mean difference between probability values is of 0.011, while the max difference is 0.026. The mean, median and the standard deviation of the parameters returned by the optimisation are shown in Table 6, where we can see that there is a large variability on λ, meaning that the parameter is not very important for the design task, provided it is large enough, and a much small variability in k and c. In particular, k here is consistently smaller than c, and this seems a crucial aspect in matching the design specification.

7. Conclusions

The role of uncertainty in formal modelling has historically been a controversial one: while stochastic processes are now common modelling tools in computer science [3], much less work has been dedicated to stochastic models with parametric uncertainty. In this paper we argue that considering parametric classes of stochastic models can be a natural scenario in many real applications, and explore how advanced verification tools can be coupled with

parameter	*mean*	*median*	*std*	*Laplace std*
λ	2.9651	1.9561	2.0725	0.2432
k	0.0946	0.0736	0.0704	0.2856
c	0.4307	0.3606	0.2364	0.3096

Table 6: Results for the design problem for the toggle switch model, for the joint optimisation of k, c, and λ. We report mean, median, and standard deviation of 25 optimisation runs plus the mean standard deviation estimated by Laplace method.

ideas from machine learning to yield effective tools for integrating logical constraints in modelling and design tasks.

Our paper is part of a growing family of works which attempt to bring tools from continuous mathematics and machine learning into formal modelling. The main reference for this paper is the earlier conference paper [13]: this is considerably expanded in this work, in particular by providing an automated procedure to set all the parameters involved in the framework. Related ideas which embed a stochastic model in a local family have been explored in the context of analysing the robustness of logical properties [6] and in the context of model repair [8]. GP optimisation in a formal modelling context has also been employed in the converse problem of learning temporal logic specifications from data [7], and is the basis of a recent novel approach to reachability computations [15].

The methodology presented in this paper offers both a promising avenue to tackle practically relevant modelling problems, and intriguing further challenges. A natural extension of our work would be to consider the identification of model structures, as opposed to model parameters only. While in principle straightforward, algorithmic adjustments will be needed to enforce sparsity constraints in the optimisation. From the modelling point of view, the idea of performing system identification from logical constraints immediately begs the question of a minimal set of logical properties that enable the identification of a system within a parametric family. We don't have an answer to this question, but it is likely that ideas from continuous mathematics will be useful in further exploring this fascinating question. From the computational point of view, the methods we use are limited to exploring systems with a handful of parameters. Scaling of Bayesian optimisation algorithms is a current topic of research in machine learning, and innovative novel ideas on randomisation [41] may hold the key to applying these methodologies to large scale formal models.

References

[1] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[2] A. Andreychenko, L. Mikeev, D. Spieler, and V. Wolf. Approximate maximum likelihood estimation for stochastic chemical kinetics. *EURASIP Journal on Bioinf. and Sys. Bio.*, 9, 2012.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, Cambridge, Mass., 2008. 01316.

[4] C. P. Barnes, D. Silk, X. Sheng, and M. P. Stumpf. Bayesian design of synthetic biological systems. *PNAS USA*, 108(37):15190–5, 2011.

[5] Alain Barrat, Marc Barthlemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, Leiden, 2008. 00763.

[6] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. On the robustness of temporal properties for stochastic models. *Electronic Proceedings in Theoretical Computer Science*, 125:3–19, August 2013.
[7] Ezio Bartocci, Luca Bortolussi, and Guido Sanguinetti. Data-driven statistical learning of temporal logic properties. In Axel Legay and Marius Bozga, editors, *Formal Modeling and Analysis of Timed Systems - 12th International Conference, FORMATS 2014, Florence, Italy, September 8-10, 2014. Proceedings*, volume 8711 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2014.
[8] Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, C. R. Ramakrishnan, and Scott A. Smolka. Model repair for probabilistic systems. In Parosh Aziz Abdulla and K. Rustan M. Leino, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, number 6605 in Lecture Notes in Computer Science, pages 326–340. Springer Berlin Heidelberg, January 2011.
[9] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
[10] L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective systems behaviour: a tutorial. *Performance Evaluation*, 2013.
[11] L. Bortolussi, D. Milios, and G. Sanguinetti. Smoothed Model Checking for Uncertain Continuous Time Markov Chains. *arXiv preprint arXiv:1402.1450*, 2014.
[12] L. Bortolussi and A. Policriti. (hybrid) automata and (stochastic) programs. the hybrid automata lattice of a stochastic program. *Journal of Logic and Computation*, 23(4):761–798, 2013. 00000.
[13] L. Bortolussi and G. Sanguinetti. Learning and designing stochastic processes from logical constraints. In *Proceedings of QEST 2013*, 2013. 00001.
[14] Luca Bortolussi and Alberto Policriti. Hybrid dynamics of stochastic programs. *Theoretical Computer Science*, 411(20):2052–2077, April 2010. 00016.
[15] Luca Bortolussi and Guido Sanguinetti. A statistical approach for computing reachability of non-linear and stochastic dynamical systems. In Gethin Norman and William H. Sanders, editors, *Quantitative Evaluation of Systems - 11th International Conference, QEST 2014, Florence, Italy, September 8-10, 2014. Proceedings*, volume 8657 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2014.
[16] Luminita Manuela Bujorianu. *Analysis of Hybrid Systems*. Number 61 in Communications and Control Engineering. Springer Verlag London, 2012. 00008.
[17] Richard L Burden and J. Douglas Faires. *Numerical analysis*. Brooks/Cole, Cengage Learning, Boston, MA, 2011.
[18] T. Chen, M. Diciolla, M.Z. Kwiatkowska, and A. Mereacre. Time-bounded verification of ctmcs against real-time specifications. In *Proc. of FORMATS*, pages 26–42, 2011.
[19] T. Cover and J. Thomas. *Elements of Information Theory, 2nd ed.* Wiley, 2006.
[20] B. Cseke, M. Opper, and G. Sanguinetti. Approximate inference in latent gaussian markov models from continuous time observations. In M. Welling, Z. Ghahramani, C.J.C.Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 21*. MIT, 2013.
[21] D. J. Daley and D. G. Kendall. Epidemics and Rumours. *Nature*, 204(4963):1118–1118, dec 1964. 00099.
[22] Robin Donaldson and David Gilbert. A model checking approach to the parameter estimation of biochemical pathways. In *Computational Methods in Systems Biology*, page 269–287, 2008. 00049.
[23] C. W. Gardiner. *Handbook of stochastic methods: for physics, chemistry and the natural sciences*. Springer series in synergetics, 13. Springer, 2002.
[24] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, 2000.
[25] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.
[26] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric markov models. *International Journal on Software Tools for Technology Transfer*, 13(1):3–19, 2011. 00049.
[27] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian approach to model checking biological systems. In *Proc. of CMSB*, pages 218–234, 2009.
[28] S. K. Jha and C. J. Langmead. Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. *Theor. Comp. Sc.*, 412(21):2162 – 2187, 2011.
[29] M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Stat. Soc. Ser. B*, 63(3):425–464, 2001.

[30] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proc. of FORMATS*, pages 152–166, 2004.
[31] Michael D. McKay. Latin hypercube sampling as a tool in uncertainty analysis of computer models. In *Proceedings of the 24th conference on Winter simulation*, pages 557–564, 1992.
[32] A. Ocone, A. J. Millar, and G. Sanguinetti. Hybrid regulatory models: a statistically tractable approach to model regulatory network dynamics. *Bioinformatics*, 29(7):910–916, February 2013.
[33] B. K. Øksendal. *Stochastic Differential Equations: An Introduction with Applications.* Berlin: Springer., 2003.
[34] M. Opper and G. Sanguinetti. Variational inference for Markov jump processes. In *Proc. of NIPS*, 2007.
[35] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[36] Derek Riley, Xenofon Koutsoukos, and Kasandra Riley. Simulation of stochastic hybrid systems with switching and reflecting boundaries. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, pages 804–812, Miami, Florida, 2008. Winter Simulation Conference. 00009.
[37] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Information-theoretic regret bounds for Gaussian process optimisation in the bandit setting. *IEEE Trans. Inf. Th.*, 58(5):3250–3265, 2012.
[38] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.
[39] Tianhai Tian and Kevin Burrage. Stochastic models for regulatory networks of the genetic toggle switch. *Proceedings of the National Academy of Sciences*, 103(22):8372–8377, May 2006. PMID: 16714385.
[40] N. G. van Kampen. *Stochastic processes in physics and chemistry*. Elsevier, Amsterdam; Boston; London, 2007.
[41] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2013.
[42] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.
[43] H. L. S. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409, 2006.