Learning and designing stochastic processes from logical constraints

Luca Bortolussi[1,2] and Guido Sanguinetti[3,4]

[1] Department of Mathematics and Geosciences, University of Trieste
[2] CNR/ISTI, Pisa, Italy
[3] School of Informatics, University of Edinburgh
[4] SynthSys, Centre for Synthetic and Systems Biology, University of Edinburgh

Abstract. Continuous time Markov Chains (CTMCs) are a convenient mathematical model for a broad range of natural and computer systems. As a result, they have received considerable attention in the theoretical computer science community, with many important techniques such as model checking being now mainstream. However, most methodologies start with an assumption of complete specification of the CTMC, in terms of both initial conditions and parameters. While this may be plausible in some cases (e.g. small scale engineered systems) it is certainly not valid nor desirable in many cases (e.g. biological systems), and it does not lead to a constructive approach to rational design of systems based on specific requirements. Here we consider the problems of learning and designing CTMCs from observations/ requirements formulated in terms of satisfaction of temporal logic formulae. We recast the problem in terms of learning and maximising an unknown function (the likelihood of the parameters) which can be numerically estimated at any value of the parameter space (at a non-negligible computational cost). We adapt a recently proposed, provably convergent global optimisation algorithm developed in the machine learning community, and demonstrate its efficacy on a number of non-trivial test cases.

1 Introduction

Stochastic processes are convenient mathematical models of a number of real world problems, ranging from computer systems to biochemical reactions within single cells. Typically, such models are formulated intensionally by specifying the transition kernel of a *continuous time Markov chain* (CTMC, [10]). A classical question in formal modelling is to calculate the probability that a certain temporal logic formula is true, given a certain process (with specified parameters); this is the question addressed by *stochastic model checking*, one of the major success stories of formal modelling in the last thirty years [4, 15].

While probabilistic model checking is indubitably a success story, it is not an unqualified one. Computationally, model checking suffers from limitations, either due to state space explosion or to the difficulty (impossibility) in checking analytically formulae in specific logics [4, 8]. Simulation-based approaches,

such as statistical model checking, can be used to circumvent these problems: these methods are usually asymptotically exact, in the limit when the number of simulations used is large; nevertheless, establishing what is a sufficiently large number of simulations to achieve a certain accuracy is a nontrivial problem. Conceptually, both model checking and statistical model checking start from the premise that a CTMC model of the system is entirely specified, i.e. the underlying parameters of the CTMC are known exactly. This is generally not true: it is certainly never true when employing CTMCs as models of physical systems (such as systems biology models, where parameters are at best known with considerable uncertainty), but it is often not appropriate even when modelling large-scale computer systems, when a coarse grained abstraction may be useful. In these cases, one would wish to use observations of the system or of its properties to determine (approximately) its parameters: this is the *system identification* problem. Moreover, the assumption of complete specification is not productive in an engineering context: rather than checking properties of systems with specific parameters, one is often interested in specifying *a priori* the properties of the system (requirements), and then adjust (few) control parameters in order to best match the requirements (the *system design* problem).

The identification of parameters of CTMCs from observations has recently received considerable interest in both the statistical machine learning and formal modelling communities, where a number of approximate methods have been proposed [3, 17]. All of these methods assume that the *state* of the system, e.g. the counts of particles of each molecular species, is observed at discrete time points. Here we consider the more general case where the observations are represented by truth values of linear time temporal logic formulae representing qualitative properties of the system. This may be more appropriate in a computer systems scenario, as it may represent an easier type of data to store/ observe, or in a systems biology scenario, when one observes a qualitative phenotype in multiple cells as opposed to directly measuring protein counts. It is also a more natural framework in which to address the design problem, as it is easier to formulate requirements in terms of logical constraints than in terms of particle counts. The restriction to linear time properties is justified because we can only observe single realisations (trajectories) of a system. Naturally, the amount of information contained in these qualitative observations is lower, making the problem more challenging.

For both the design and identification problems the outstanding difficulty is the lack of an objective function that can be used in an optimisation routine: the fit of a CTMC with specific parameters to observations (or the match to requirements) cannot in general be estimated analytically. We therefore need to optimise an unknown function with the smallest number of function evaluations. The key observation in this paper is that a similar problem also occurs in the classical AI problem of *reinforcement learning*: there, the goal is to devise a *strategy* (i.e. an update rule) which will lead to the optimisation of an unknown reward function with the smallest number of trials (function evaluations). This observation allows us to leverage powerful, provably convergent algorithms from

the statistical machine learning community: in particular, we adapt to our situation the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm [20], and show on a number of examples that this provides a practical and reliable approach to both the identification and design problem. We further extend the algorithm in order to provide confidence estimates for the obtained parameters, and to detect possible non-identifiability issues. The paper is organised as follows: the problems we tackle and the formal methods tools we use are introduced in Section 2. We then present the machine learning tools we use in Section 3, while in Section 4 we present some computational experiments that give a proof of concept demonstration of our approach. We then briefly discuss our results, highlighting how the coupling of advanced machine learning and formal modelling can open innovative directions in both fields.

2 Problem definition

Let the probability distribution on trajectories of stochastic process of interest be denoted as $P(x_{0:T}|\theta)$, where $x_{0:T}$ denotes a trajectory of the system up to time T, θ is a set of parameters, and $P(\cdot)$ denotes the probability distribution/ density. Let $\varphi_1, \ldots, \varphi_d$ be d (temporal) logic formulae whose truth depends on the specific trajectory of the process which is observed. We are interested in the following two problems:

Identification Problem: Given evaluations of each of the d formulae over N independent runs of the process, arranged into a $d \times N$ binary design matrix D, determine the value(s) of the parameters θ that make these observations most probable.

Design Problem: Given a probability table P for the joint occurrence of a number of formulae, determine the parameters of the stochastic process which optimally match these probabilities.

We will see that a very similar approach can be adopted to solve both problems. We introduce now the main logical and algorithmic ingredients of our approach.

2.1 Metric interval Temporal Logic

We will consider properties of stochastic trajectories specified by Metric interval Temporal Logic (MiTL), see [1, 16]. This logic belongs to the family of linear temporal logics, whose truth 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, and its temporal operators are all time-bounded. We decided to focus on MiTL because when we observe a system, e.g. a biological one, we always observe *single time-bounded realisations* (essentially, time-bounded samples from its trajectory space). Hence, MiTL is the natural choice to formalise the qualitative outcome of experiments.

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 $\mathbf{U}^{[T_1,T_2]}$. Atomic propositions μ are defined like in Signal Temporal Logic (STL [16]) as boolean predicate transformers: they take a real valued function $\boldsymbol{x}(t)$, $\boldsymbol{x} : [0,T] \to \mathbb{R}^n$, as input, and produce a boolean signal $s(t) = \mu(\boldsymbol{x}(t))$ 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 defined 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 $\boldsymbol{x}$, and its satisfaction relation is given in a standard way, see e.g. [1, 16]. We report here only the rules for atomic propositions and the temporal operator, as those for boolean connectives are standard:

- $\boldsymbol{x}, t \models \mu$ if and only if $\mu(\boldsymbol{x}(t)) = \mathtt{tt}$;
- $\boldsymbol{x}, t \models \varphi_1 \mathbf{U}^{[T_1,T_2]} \varphi_2$ if and only if $\exists t_1 \in [t+T_1, t+T_2]$ such that $\boldsymbol{x}, t_1 \models \varphi_2$ and $\forall t_0 \in [t, t_1]$, $\boldsymbol{x}, t_0 \models \varphi_1$ (here we follow the treatment of STL [16]).

The temporal logic MiTL can be easily extended to the probabilistic setting, and interpreted over CTMC [12, 8]. Essentially, one is interested in the path probability of a formula φ, defined as $P(\varphi|\theta) = P\left(\{\boldsymbol{x}_{0:T} | \boldsymbol{x}_{0:T}, 0 \models \varphi\}|\theta\right)$, i.e. as the probability of the set of time-bounded CTMC trajectories that satisfy the formula[5].

2.2 Likelihood function

Consider now a CTMC depending on a set of parameters θ, and a set of d MiTL formulae $\varphi_1, \ldots, \varphi_d$ whose truth values have been observed over N independent runs of the process. Let D be the $d \times N$ *design matrix*, whose column vectors correspond to joint observations of the properties. Given a specific value of the parameters θ, the probability of a particular joint truth value for the set of formulae of interest is uniquely determined. Let $P(D_i|\theta)$ be the probability of the joint truth value of formulae of the i_{th} column of the matrix D given the parameters θ. Under the assumption of independent runs, the likelihood of the observations D is then simply

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

Alternatively, if prior knowledge over the parameters is available as a prior distribution $P(\theta)$, we may want to consider the un-normalised posterior distribution $P(\theta, D) = P(\theta) \prod_{i=1}^{N} P(D_i|\theta)$. The identification problem can be carried out by

[5] We assume implicitly that T is sufficiently large so that the truth of φ at time 0 can always be established from $\boldsymbol{x}$. The minimum of such times can be easily deduced from the formula φ, see [12, 16]

maximising the likelihood (1) (maximum likelihood, ML) or the un-normalised posterior (maximum a posteriori, MAP).

Numerical evaluation of $P(D_i|\theta)$ is a major challenge: computing the path probability of a MiTL formula is an extremely difficult problem, with current algorithms [8] suffering severely from the state space explosion. Furthermore, numerical methods for stochastic model checking have always been developed to compute the path probability of a *single* formula, while computing $P(D_i|\theta)$ requires to know the *joint* probability distribution of the formulae $\varphi_1, \ldots, \varphi_d$. We therefore resort to statistical model checking to approximately evaluate the likelihood $\mathcal{L}(D, \theta)$.

2.3 Statistical model checking

We now briefly review Statistical Model Checking (SMC [12, 22]), a class of methods that try to estimate the probability of a path formula or the truth of a state formula relying on simulation and statistical means. In the context of MiTL, SMC works as follows. Given a CTMC with fixed parameters θ, a simulation algorithm, like SSA [11], is used to sample trajectories of the process. For each sampled trajectory, we run a model checking algorithm for MiTL and establish if φ is true or false. The process therefore generates 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 [22] or Bayesian alternatives [12], 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 [6], applying Bayes' theorem we get

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

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. Given the simulated data D_φ, our best guess about the true probability $P(Z_\varphi = \mathtt{tt})$ is then given by the predictive distribution [6]:

$$P(Z_\varphi = \mathtt{tt}|D_\varphi) = \int_0^1 P(Z_\varphi = \mathtt{tt}|q)P(q|D_\varphi)dq = \mathbb{E}[q|D_\varphi] = \frac{k_1 + a}{k_1 + a + k_0 + b}$$

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 the much larger set of joint truth values of several formulae.

To extend Bayesian SMC to checking the joint truth probabilities of multiple formulae, we choose a Dirichlet prior distribution with parameters $\alpha_1, \ldots, \alpha_{2^d}$ equal to 1 (corresponding to adding one pseudo-count to every possible joint

truth value). Given observations $D_{\varphi_1,\ldots,\varphi_d}$ of the truth values of $Z_{\varphi_1,\ldots,\varphi_d}$[6], analogous calculations yield the predictive distribution

$$P(Z_{\varphi_1,\ldots,\varphi_d} = \boldsymbol{d}_j | D_{\varphi_1,\ldots,\varphi_d}) = (\alpha_j + k_j)/(\alpha_0 + k)$$

where k_j is the number of times we observed the jth truth combination, corresponding to a point $\boldsymbol{d}_j \in \mathcal{D}$ and $\alpha_0 = \sum_j \alpha_j$. This probability is then used to estimate the likelihood $\mathcal{L}(D, \theta)$, as $\mathcal{L}(D, \theta) = \prod_{i=1}^{N} P(D_i|\theta)$. 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.

3 Global optimisation

As we have seen, the identification problem entails the maximisation of an unknown function which can be only estimated (with approximately Gaussian noise) at isolated points at considerable computational cost. One can approach this problem also from a Bayesian angle by treating the unknown function as a random function (arising from a suitable prior stochastic process) and then use the numerical evaluations 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* [14]. This leads to a very elegant algorithm for optimisation; we now briefly review the main concepts and the algorithm we use.

3.1 Gaussian Processes

Gaussian Processes (GPs) are a natural extension of the multivariate normal distribution to infinite dimensional spaces of functions. A GP is a probability measure over the space of continuous functions (over a suitable input space) such that the random vector obtained by evaluating a sample function at a finite set of points $x_1, \ldots, x_N$ follows a multivariate normal distribution. 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

$$f \sim \mathcal{GP}(\mu, k) \leftrightarrow \mathbf{f} = (f(x_1), \ldots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K) \tag{2}$$

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 is an important modelling decision, as it essentially determines the type of functions which can be sampled from a GP

[6] Note that $D_{\varphi_1,\ldots,\varphi_d}$ is a matrix, similarly the design matrix discussed in Section 2, but we treat each column/ observation as a single point of $\mathcal{D}$.

(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

$$k(x, x') = \gamma \exp\left[-\frac{\|x - x'\|^2}{\lambda^2}\right] \tag{3}$$

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. For more details, we refer the interested reader to the excellent review book of Rasmussen and Williams [18].

3.2 GP regression and prediction

Suppose now that we are given a set of noisy observations $\mathbf{y}$ of the function value at input values $\mathbf{x} = x_1, \ldots, x_N$, distributed around an unknown true value $f(\mathbf{x})$ with spherical Gaussian noise of variance σ^2. We are interested in determining how these observations influence our belief over the function value at a further input value x^* where the function value is unobserved.

By using the basic rules of probability and matrix algebra, we have that the predictive distribution at x^* is again Gaussian with mean

$$\mu^* = (k(x^*, x_1), \ldots, k(x^*, x_N))\, \hat{K}_N^{-1} \mathbf{y} \tag{4}$$

and variance

$$k^* = k(x^*, x^*) - (k(x^*, x_1), \ldots, k(x^*, x_N))\, \hat{K}_N^{-1}\, (k(x^*, x_1), \ldots, k(x^*, x_N))^T . \tag{5}$$

where $\hat{K}_N$ is obtained by evaluating the covariance function at each pair of training points and adding σ^2 times the identity. Notice that the first term on the r.h.s of equation (5) 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.

Equation (4) warrants two important observations: first, as a function of the new point x^*, μ^* is a linear combination of a finite number of *basis* functions $k(x^*, x)$ centred at the observation points. Secondly, the posterior mean at a fixed x^* is a linear combination of the observed values, with weights determined by the specific covariance function used. For the RBF covariance, input points further from the new point x^* are penalised exponentially, hence contribute less to the predicted value.

3.3 Upper Confidence Bound optimisation

We now return to the problem of finding the maximum of an unknown function with the minimum possible number of function evaluations. This is related to the problem of performing sensitivity analysis w.r.t. the parameters of complex

computer models, e.g. climate models, where a quantification of uncertainty on the model outputs is essential. An elegant approach to solving this problem has been proposed by Kennedy and O'Hagan [14] by recasting the problem in a Bayesian formalism: the true function linking the parameters to the model outputs is assumed unknown and is assigned a GP prior. A (limited) number of function evaluation are then used as (noiseless) observations to obtain a GP posterior mean function which *emulates* the true unknown function, and is used for subsequent analyses.

In the optimisation case, the situation is slightly different: given an initial set of function evaluations, we are interested in determining a sequence of input values that converges to the optimal value of the function. A naive approach would be to use GP regression to emulate the unknown function, and to explore the region near the maximum of the posterior mean. It is easy to see, though, that this approach is vulnerable to remaining trapped in local optima. On the other hand, one could sample uniformly across the input domain of interest; this is guaranteed to eventually find the global optimum but is unlikely to do so in a reasonable time. It is therefore clear that one needs to trade off the *exploitation* of promising regions (high posterior mean) with the *exploration* of new regions (high posterior variance).

The GP Upper Confidence Bound (GP-UCB) algorithm [20] prescribes an exploration-exploitation trade-off 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

$$x_{t+1} = \text{argmax}_x \left[\mu_t(x) + \beta_t \sqrt{var_t(x)} \right] \tag{6}$$

where β_t is a constant that depends on the iteration of the algorithm.

To specify in which sense the algorithm converges, we need a definition.

Definition 1. *Let x^* be the value at which a function f attains its maximum. The instantaneous regret of selecting a point x_t is defined as $r_t = f(x^*) - f(x_t)$ and the cumulative regret at time T is defined as $\sum_{t=1}^{T} r_t$. An iterative optimisation algorithm is no-regret if*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t = 0.$$

Srinivas et al [20] then proved the following theorem

Theorem 1. *Let $\beta_t = k + \alpha \log t$, where k and α are positive constants. Then the GP-UCB algorithm in equation (6) is no-regret. More specifically, with high probability, the cumulative regret is bounded by $O(\sqrt{T})$.*

This theorem indicates that, as the algorithm proceeds, exploration needs to become gradually more important than exploitation (β_t is monotonically increasing), as one would intuitively expect. The algorithm has been successfully employed in a number of difficult optimisation problems, from determining optimal structure of synthetic proteins [19] to computer vision [21].

3.4 Estimating uncertainty

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 [6]) is ruled out.

We therefore resort to a simple deterministic approximation which estimates the variance/ covariance in the parameter estimates by inverting the Hessian of the likelihood at its maximum. This approach, known as *Laplace approximation* in statistics/ machine learning, is equivalent to approximate the posterior around the maximum with a Gaussian which locally optimally matches the posterior. In order to estimate the Hessian, there are at least two strategies available: one can estimate the likelihood (numerically) on a fine (small) grid around the maximum and then use standard numerical estimation methods, or one can use the GP emulation as a surrogate of the function and directly differentiate the GP mean. This second option has the advantages of handling the noise in the estimation of the likelihood arising from statistical model checking (which is smoothed out in GP regression), and of being analytically tractable. Recalling that the GP mean at a point is a linear combination of basis functions, one can just differentiate twice equation (4) to obtain the result.

3.5 Model design

The problem of model design is intimately linked to the inference problem: in fact, one could characterise model design as *inference with the data one would like to have* [5]. In our case, we are given a probability table for the joint occurrence of a number of formulae $\varphi_1, \ldots, \varphi_N$.[7] As explained earlier, the probability of a specific truth configuration of a number of formulae is an intractable function of the parameters, which in many cases can only be approximately computed by statistical model checking. 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

[7] This problem formulation is different from a recent approach on parameter synthesis for CTMC using SMC, [13], in which the authors look for a subset of parameters in which a single formula φ is satisfied with probability greater than q.

probability distributions; we choose to use the Jensen-Shannon divergence due to its information theoretic properties and computational good behaviour (being always finite) [9]. 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 $q = p$. The GP-UCB minimisation of the Jensen-Shannon divergence between an empirical q and the prescribed p can then be carried out as described above.

4 Experiments

We now illustrate our approach on a number of test cases. We benchmark the approach on a simple example where the calculations can be performed analytically: a Poisson process where the truth values of a single logical formula are observed. We then show how our approach can solve both the identification and the design problems on a non-trivial computer infection model.

4.1 Poisson process

Poisson processes are random processes with values in $\mathbb{N} \cup \{0\}$; they play a fundamental role in physics (e.g. as models of radioactive decay), biology (e.g. as models of RNA production) and computer science (e.g. as models of arrivals of packets at servers). They can be defined equivalently in several different ways; here, we take the operational definition that a Poisson process with rate μ is an increasing, integer valued process such that

$$P(k = n|\mu, t) = \frac{(\mu t)^n}{n!}\exp[-\mu t]. \tag{7}$$

We consider a very simple scenario where we have observed five times independently the truth value of the formula $\varphi(k) = \mathbf{F}^{[0,1]}\{k > 3\}$, i.e. the formula expressing the fact that k has become bigger than 3 within 1 time units, 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

$$p = P(\varphi = true) = 1 - P(\varphi = false) = 1 - \sum_{n=0}^{3} \frac{(\mu)^n}{n!}\exp[-\mu]. \tag{8}$$

and hence we have an analytical expression for the log-likelihood (or un-normalised posterior given a prior).

Figure 1 left panel shows the log-likelihood for 40 independent observations of process trajectories, overlayed with the estimation obtained by SMC over a grid using 12 samples. As we can see, SMC provides a noisy (but overall accurate)

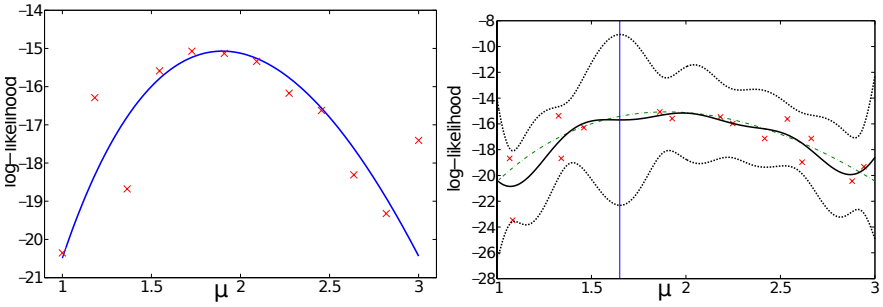

Fig. 1. Simulation on a Poisson process: *left* exact log likelihood and SMC estimation (red crosses) for $\mu \in [1, 3]$; *right* Illustration of the GP-UCB algorithm: GP likelihood estimation (black line), true likelihood (green dashed-dotted line), and GP-UCB upper bound (dotted line).

measurement of the log-likelihood function. Figure 1 right panel instead shows the working of the GP-UCB algorithm (with constant $\beta_t \equiv 2$): here, we have observed only 15 SMC evaluations of the log-likelihood (red crosses); the GP mean is given by the solid black line, and the mean $\pm$ 2 standard deviations by the dashed line. The vertical line represents the next point chosen by the GP-UCB algorithm. The dashed-dotted line is the analytical log-likelihood.

4.2 Network epidemics

We consider now a more structured example of the spread of a worm epidemics in a computer network with a fixed number of nodes [7]. We consider a simple variation of the classical SIR infection model [2], in which an initial population of susceptible nodes can be infected either from outside the network (e.g. by receiving an infected email message) or by the active spread of the virus by infected computers in the network. Infected nodes can be patched, and become immune to the worm for some time, after which they are susceptible again (for instance, to a new version of the worm).

This system is modelled as a population CTMC, in which the state space is described by a vector $\boldsymbol{X}$ of three variables, counting how many nodes are in the susceptible (X_S), infected (X_I), and patched state (X_R). The dynamics of the CTMC is described by a list of transitions, or reactions, together with their rate functions. We represent them in the biochemical notation style (see e.g. [11]). All rates of this model follow the law of mass action.

External infection: $S \xrightarrow{k_e} I$, with rate function $k_e X_S$;
Internal infection: $S + I \xrightarrow{k_i} I + I$, with rate function $k_i X_S X_I$;
Patching: $I \xrightarrow{k_r} R$, with rate function $k_r X_I$;
Immunity loss: $R \xrightarrow{k_s} S$, with rate function $k_s X_R$;

For this system, we considered three temporal logical properties, expressed as MiTL formulae, all concerned with the number of infected nodes (total number of nodes is 100). The properties are:

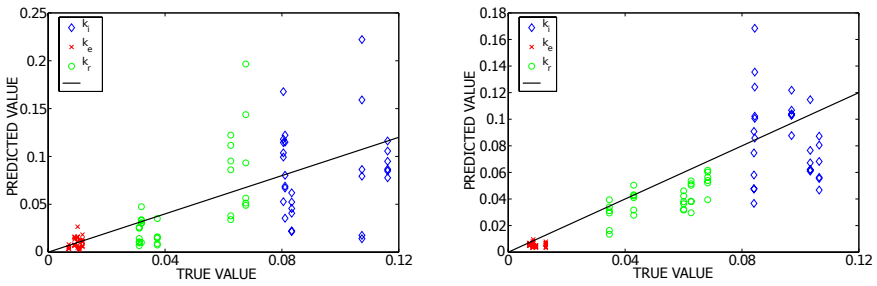

Fig. 2. True versus predicted value of epidemics model parameters k_i, k_e, k_r. The black line represents the identity true=predicted. Left: ML; right: MAP.

1. $\mathbf{G}^{[0,100]}(X_I < 40)$: the fraction of infected nodes never exceeds 40% in the first 100 time units;
2. $\mathbf{F}^{[0,60]}\mathbf{G}^{[0,40]}(5 \leq X_I \leq 20)$: within time 60, the fraction of infected nodes is between 5 and 20 and remains so for 40 time units.
3. $\mathbf{G}^{[30,50]}(X_I > 30)$: the fraction of infected nodes is above 30% between time 30 and time 50.

The first property puts a bound on the peak of infection, while the third constrains it to happen around time 40. The second property, instead, is intended to control the number of infected nodes after the infection peak.

Given the model and the properties, we set up the experiment as follows. We fixed the rate of immunity loss to 0.01; the remaining parameters are those we explored. First we fixed these parameters to a value sampled uniformly in $k_i \in [0.08, 0.12]$, $k_e \in [0.007, 0.013]$, $k_r \in [0.03, 0.07]$, and use the sampled configuration to generate 40 observations D of the value of the logical formulae. Then, we ran the GP-UCB optimisation algorithm with the following search space: $k_i \in [0.01, 1]$, $k_e \in [0.001, 0.1]$, $k_r \in [0.005, 0.5]$, so that 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 60 points sampled randomly and uniformly 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 800 points, sampled uniformly at each iteration. If in this grid a point is found with a larger value than those of the observation points, we compute the likelihood also 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 doing only 12 additional likelihood evaluations on average.

We consider both the maximum likelihood (ML) and maximum a posteriori (MAP) identification problems; in the MAP case, we use independent, vaguely informative Gamma priors, with mean 0.1 for k_i, 0.01 for k_e and 0.05 for k_r, and shape equal to 10. To assess statistically our results, we repeated the experiments (both ML and MAP) on 5 different parameter configurations, doing 6

Max Likelihood

Param	true value	pred1	sd	pred2	sd	pred3	sd
k_i	0.0811	0.0803	0.0142	0.0670	0.0084	0.1100	0.0132
k_e	0.0118	0.0114	0.0029	0.0106	0.0014	0.0065	0.0011
k_r	0.0319	0.0304	0.0032	0.0330	0.0020	0.0293	0.0034

MAP

Param	true value	pred1	sd	pred2	sd	pred3	sd
k_i	0.1034	0.0927	0.0048	0.0946	0.0079	0.0744	0.0062
k_e	0.0084	0.0081	0.0005	0.0106	0.0004	0.0076	0.0011
k_r	0.0683	0.0719	0.0044	0.0683	0.0039	0.0643	0.0113

Table 1. True parameter and three predictions randomly chosen, both for ML and MAP, after running a gradient ascent optimisation on the GP mean. We show the predicted value and the uncertainty estimate, obtained from the estimated hessian of the likelihood function.

runs per configuration. In the test, we fixed the length-scale hyperparameter of the Gaussian kernel to 0.1, and the amplitude to 60% of the difference between the maximum and the mean value of the likelihood for the 60 initial observations. Results are reported in Figure 2, where we plot the values of the true parameters that generated the samples against the estimated values. As can be seen, the predicted values are quite close to the original ones, both in the ML and in the MAP cases. Indeed, the average observed error (euclidean distance from the true configuration) is 0.0492 for ML and 0.0346 for MAP. These results show that the use of prior information can improve the performances of the algorithm. Furthermore, it tends to reduce the sensitivity of the algorithm to the hyperparameters, especially the length-scale. We report also the relative errors, obtained dividing the absolute ones by the diameter of the search space: 4.43% for ML and 3.11% for MAP. In Table 1, we report for a random subset of runs both the true/ inferred parameter values, and the uncertainty estimates obtained using the Laplace approximation described in Section 3.4. Empirically, we observed that in some instances the Hessian became not negative definite: this may indicate identifiability problems given a specific set of observations. To circumvent this problem, we ran a gradient ascent optimisation on the GP mean before computing the Hessian, to ensure that the point is a local maximum. Also empirically, we observed that the estimation of the uncertainty is affected by the values of the hyperparameters of the GP; we discuss further this issue in the conclusions.

4.3 System Design

We consider now an example of system design, in which we try to minimise the Jensen-Shannon divergence (JSD) between the estimated joint probability distribution and a target one (for numerical stability, we consider $\text{atanh}(1 - 2JSD)$). We consider again the network epidemics model, and look at the following two properties:

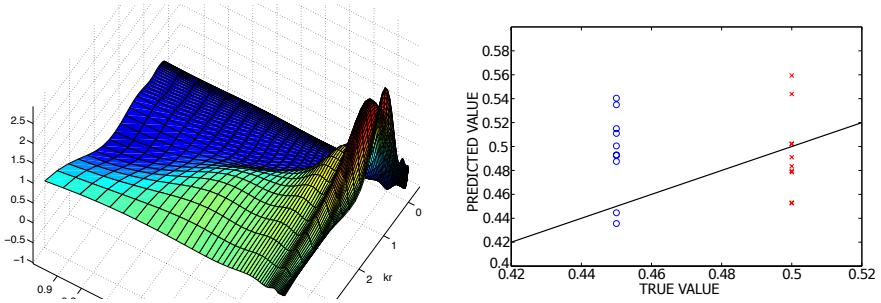

Fig. 3. Left: atanh$(1 - 2JSD)$ GP-estimated landscape for the network epidemics model and target probability $p(1,1) = 0.5$, $p(1,0) = 0.45$, $p(0,1) = 0.04$, $p(0,0) = 0.01$. Right: true versus predicted value of target probability for the network epidemics design problem. The black line represents the identity true=predicted.

1. $\mathbf{G}^{[0,100]}(X_I < 20)$: the fraction of infected nodes never exceeds 20% in the first 100 time units;
2. $\mathbf{F}^{[0,60]}\mathbf{G}^{[0,40]}X_I \leq 5$: within time 60, the fraction of infected nodes is less than or equal to 5 and remains so for 40 time units.

In our experimental setting, we fixed the internal infection rate to $k_i = 1$ and the immunity loss rate to $k_s = 0.01$. Hence, the search space is made of two parameters, k_e, the external infection rate, and k_r, the patch rate. Intuitively, those two parameters are somehow controllable, by filtering suspected messages from outside the network or by active patching. In the first experiment, we set the target probability p to the following: $p(1,1) = 0.5$, $p(1,0) = 0.45$, $p(0,1) = 0.04$, $p(0,0) = 0.01$. The idea is that we really want the first property to hold, but we are somehow less restrictive on the second one (conditional on the first being true). Having a 2 dimension parameter space to explore, allows us to visualise the GP estimate of the the JSD function, by sampling a 12x12 grid of equispaced points in $[0.01, 1] \times [0.01, 5]$ (after log-normalisation, we use length-scale 0.5 and amplitude 1 as hyperparameters). The result can be seen in Figure 3 left. As we can see, there is a relatively small region in the parameter space that seems to collect the larger score. Running ten experiments, we obtained an average JSD of 0.0155, while the probability values estimated for $p(1,1)$ and $p(1,0)$ are visually reported in Figure 3 right.

We also run an experiment varying the target probability distribution, sampling it from a Dirichelet distribution with parameters $10, 0.8, 9, 0.2$, thus giving higher probability to $(1,1)$ and $(0,1)$, differently from the previous test. We sampled 5 different target distributions, and run 5 experiments for each combination, obtaining an average JSD of 0.0168. Probabilities obtained, plotted against target probabilities, are reported in Figure 4 left, while in Figure 4 right, we plot k_r versus k_e for parameter combinations found by the algorithm. While the overall results are good, there is a strong linear dependency between the two parameters, raising an issue of identifiability for this input specification of model design.

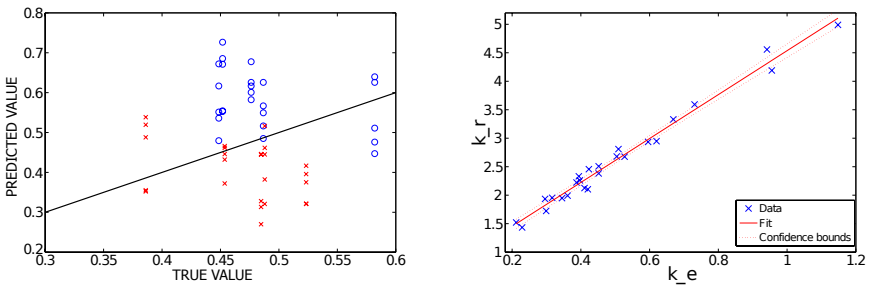

Fig. 4. Left: true versus predicted value of target probability for the network epidemics design problem. The black line represents the identity true=predicted. Right: k_r versus k_e of the predicted parameters for the 5x5 grid.

5 Conclusions

In this paper, we considered the problem of identifying and designing stochastic processes from logical constraints, given by the (probability of) satisfaction of specific formulae in MiTL. This complements approaches to learning parameters of stochastic processes from observations of the state of the system [17, 3], and can be arguably more appropriate in a number of situations; however, the information loss resulting from having only access to qualitative observations makes the problem considerably more challenging. Another benefit of our approach is that it provides a conceptually unified framework that can also be used to address the system design problem, where logical constraints are a much more natural form of specifying requirements. A significant strength of our approach is its computational efficiency and scalability: in our experiments, the global maximum was usually found with few tens of function evaluations, hence the bottleneck is essentially the SMC step. Moreover, the GP regression approach naturally allows to incorporate and smooth the inaccuracies resulting from SMC (modelled as Gaussian observation noise), which means that relatively short runs of SMC are needed at each function evaluation. While we believe the results we have shown are a promising first step in addressing these challenging problems, there is considerable scope for further extension and improvements. Setting the hyperparameters of the GP covariance (3) is currently done heuristically; they could also be optimised, but at a non-negligible computational cost [18]. For the system design problem, one may incur in identifiability problems when the requirements cannot be satisfied (e.g. because of logical contradictions), when they are redundant, or when they under-constrain the system. Tools to address these issues would clearly be beneficial.

Finally, we would want to remark on how these results could only be obtained by the cross-fertilisation of ideas from advanced formal modelling (e.g. Bayesian SMC) and advanced machine learning (Gaussian processes, the GP-UCB algorithm). It is our opinion that increased interaction between these two areas of computer science will be highly beneficial to both, in particular towards the practical deployment of advanced algorithmic tools.

References

1. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
2. H. Andersson and T. Britton. *Stochastic Epidemic Models and Their Statistical Analysis*. Springer-Verlag, 2000.
3. 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.
4. C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model checking continuous-time Markov chains by transient analysis. *IEEE TSE*, 29(6):524–541, 2003.
5. C. P. Barnes, D. Silk, X. Sheng, and M. P. Stumpf. Bayesian design of synthetic biological systems. *PNAS USA*, 108(37):15190–5, 2011.
6. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
7. J. T. Bradley, S. T. Gilmore, and J. Hillston. Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. *J. Comput. Syst. Sci.*, 74(6):1013–1032, 2008.
8. 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.
9. T. Cover and J. Thomas. *Elements of Information Theory, 2nd ed.* Wiley, 2006.
10. Richard Durrett. *Essentials of stochastic processes*. Springer, 2012.
11. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.
12. 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.
13. 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.
14. M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Stat. Soc. Ser. B*, 63(3):425–464, 2001.
15. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *Int. Jour. on Softw. Tools for Tech. Transf.*, 6(2):128–142, 2004.
16. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proc. of FORMATS*, pages 152–166, 2004.
17. M. Opper and G. Sanguinetti. Variational inference for Markov jump processes. In *Proc. of NIPS*, 2007.
18. C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
19. P. A. Romero, A. Krause, and F. H. Arnold. Navigating the protein fitness landscape with Gaussian processes. *PNAS USA*, 110(3):E193–E201, 2013.
20. 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.
21. A. Vezhnevets, V. Ferrari, and J. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *Comp Vision and Pattern Recog*, 2012.
22. 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.