Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

BAYESIAN STATISTICAL PARAMETRIC VERIFICATION AND SYNTHESIS BY MACHINE LEARNING

Luca Bortolussi

Department of Mathematics and Geosciences
University of Trieste
Via A. Valerio 12/1
Trieste, 34127, ITALY

Guido Sanguinetti

School of Informatics
University of Edinburgh
10 Crichton St.
Edinburgh, EH8 9AB, ITALY

Simone Silvetti

Esteco SpA
Area Science Park
Padriciano 99
Trieste, 34149, ITALY

ABSTRACT

We consider the problem of parametric verification, presenting a recent statistical method to perform parametric verification of linear time properties of stochastic models, estimating the satisfaction probability as a function of model or property parameters. The approach leverages Bayesian Machine Learning based on Gaussian Processes. Under mild conditions on continuity of parameters of the satisfaction probability, it can be shown that property satisfaction is a smooth function of such parameters. Gaussian Processes can effectively capture this smoothness and obtain more-accurate estimates of satisfaction probabilities by transferring information across the parameter space. We leveraged this approach to efficiently solve several tasks, like parameter synthesis, system design, counterexample generation, and requirement synthesis. In this tutorial, we will introduce the basic ideas of the approach and give an overview of the different applications.

1 INTRODUCTION

The steep curve of technological evolution is facing us with the need of engineering complex systems, ranging from open networks of heterogeneous Cyber Physical Systems to Synthetic Biology. Model-based engineering is a necessity in this context (Lee 2008), and tools for formally specifying and verifying requirements are extremely useful. A witness of this is the success story of CPU design using formal languages for the description of architectures and requirements, the latter based on temporal logic, and effective verification tools scaling up to the complexity of current CPUs (Alur et al. 2015).

Complex systems are typically described relying on stochastic models. Formal methods for specifying and verifying properties are well established also in this setting (Katoen 2016), with leading software tools such a PRISM (Kwiatkowska et al. 2011). A lot of research in the last twenty years has been focused on devising algorithms capable of computing probabilities of complex behavioral properties with a prescribed precision ε, for a given class of models, e.g., Continuous Time Markov Chains (Baier et al. 2003). The issue with these approaches is their scalability: computing probabilities introduces an overhead

that exacerbates the state space explosion. Current numerical algorithms do not scale to models even of a modest complexity compared to real scenarios. A recent trend in dealing with such limitations has been to turn back to simulation and statistical methods, integrating them with the verification routines (Younes and Simmons 2006; Jha et al. 2009).

Alternative approaches are based on exploiting the model structure and approximation properties. In case of population models, which account for a large fraction of complex system modeling, one can leverage stochastic approximation ideas to compute probabilities for large classes of properties, as proposed by Bortolussi and Hillston (2015) and Bortolussi et al. (2017). Other alternatives are based on exploiting the model structure, like in abstraction refinement (Kattenbelt et al. 2009) or quasi-lumpability (Franceschinis and Muntz 1994).

All methods discussed above deal with fully specified systems, not only in terms of interactions but also of model parameters. However, when our interest is in model-based design, several parameters, and possibly inputs, are unspecified and have to be identified in order to achieve the goal (e.g., some behavior or performance of the design). This means that we have to search the space of parameters, introducing an additional large computational overhead. The problem becomes particularly challenging if we have to provide guarantees on the behavior for a set of parameters of the model, typically of uncountable cardinality.

The standard approach in engineering literature is to rely on testing procedures (Dias Neto et al. 2007), where several parameter configurations are explored, trying to cover as much as possible of the parameter space. Providing guarantees for testing procedures, though, is quite hard (Peleska 2013). The verification counterpart of testing is known as parametric verification. Here, the idea is to compute the functional dependency of properties on parameters. In case of stochastic models, this typically means to obtain a function that expresses how the probability of a certain behavior varies with respect to model parameters. This is a much more challenging problem than verification for a single value of parameters. In case of discrete-time Markov Chains (DTMC), there are methods to compute analytic exact representation of the function (whose size explodes with state space), and are at the base of tools like PROPhESY (Dehnert et al. 2015). In case of continuous time, there are numerical methods that compute bounds on the probability in any point of parameter space (Brim et al. 2013; Češka et al. 2014; Češka et al. 2016), which severely suffer from state-space explosion. Statistical parametric verification methods would lie in between testing and exact parametric algorithms, and could provide a reliable and scalable alternative, provided we can infer bounds and accuracy also for points in the parameter space for which no simulation has been performed. This can effectively be done, relying on Bayesian methods for machine learning (Rasmussen and Williams 2006).

In this tutorial paper, we present our journey in this direction, which started five years ago and led us to the development of a broad range of methods integrating verification algorithms for linear time properties and Bayesian machine learning methods based on Gaussian Processes (GPs).

The basic idea is to leverage GPs to learn a model of the satisfaction probability as a function of model parameters from sparse simulations in the parameter space. The Bayesian nature of GPs allows us to compute an analytic approximation of the probability function, together with the uncertainty of our estimate in each point of the parameter space. We called this core approach *smoothed model checking* (Bortolussi et al. 2016a). Uncertainty bounds of the learned reconstruction can then be combined with active learning ideas to provide efficient algorithms for a series of tasks: parameter synthesis (Bortolussi and Silvetti 2018), system design (Bartocci et al. 2015), multi-objective system design (Bortolussi et al. 2016b), counterexample generation and search-based falsification (Silvetti et al. 2017), requirement learning (Bartocci et al. 2014; Bufo et al. 2014), and system identification (Bortolussi and Sanguinetti 2015). We also have developed a Java tool, implementing some of these methods, called U-check (Bortolussi et al. 2015).

In this paper we give a gentle introduction to smoothed model checking and its applications, starting from background material in models and logic in Section 2 and on GP in Section 4. Section 5 presents

smoothed model checking (smmc) and Section 6 discusses several applications combining it with active learning ideas. A final discussion is given in Section 7.

2 MODELS AND PROPERTIES

We now introduce background concepts on models and quantitative formal methods. We will start by introducing the class of models we consider, which are Markovian population models described by chemical reaction networks, where dependency on model parameters is made explicit. However, the techniques we present here have a much broader range of application, requiring only some form of continuity with respect to model parameters. Next, we introduce a simple example, which we will use throughout the paper: a classical epidemic spreading model. In the following subsection, we present a language to describe behavioral properties of population models: Metric Interval Temporal Logic (MITL), showing two alternative semantics that will be exploited further on in the paper. Also in this case, the methods we present have a larger scope of application, and MITL is only one possible choice, though very common and flexible. Alternatives will be discussed in the same section. Finally, we will briefly discuss a statistical approach for the verification of these properties, known as Statistical Model Checking.

2.1 Parametric Chemical Reaction Networks

Population models capture systems of interacting agents under the assumption that individuals behave indistinguishably and can interact with everyone else. This enables a full description of the system from a population perspective. Examples of population models can be found in epidemic spreading (Gardner et al. 2000), in systems biology (Gillespie 1977), in performance evaluation (Bortolussi et al. 2013), just to name some of the application areas. Chemical Reaction Networks (CRN, Gillespie 1977), Population Continuous Time Markov Chains (Bortolussi et al. 2013), and Markov Population Models (Henzinger et al. 2009) are different approaches in literature for modeling population processes. We consider a variant of CRN, making parameters explicit.

Definition 1 A Parametric Chemical Reaction Network (PCRN) $\mathscr{M}$ is a tuple $(S, X, D, x_0, \mathscr{R}, \Theta)$ where

- $S = \{s_1, \ldots, s_n\}$ is the set of species (i.e. the different agent states);
- $X = (X_1, \ldots, X_n)$ is the vector of variables counting the amount of each species, with values $X \in D$, with $D \subseteq \mathbb{N}^n$ being the state space;
- $x_0 \in D$ is the initial state;
- $\mathscr{R} = \{r_1, \ldots, r_m\}$ is the set of chemical reactions, each of the form $r_j = (v_j, f_j)$, with v_j the update vector and $f_j = f_j(X, \theta)$ the rate function. Each reaction can be represented as

$$r_j : u_{j,1}s_1 + \ldots + u_{j,n}s_n \xrightarrow{f_j} w_{j,1}s_1 + \ldots + w_{j,n}s_n,$$

  where $u_{j,i}$ ($w_{j,i}$) is the amount of elements of species s_i consumed (produced) by reaction r_j. Letting $u_j = (u_{j,1}, \ldots, u_{j,n})$ (and similarly w_j), the update vector is $v_j = w_j - u_j$ and describes the net change in population counts due to the happening of reaction j.
- $\theta = (\theta_1, \ldots, \theta_k)$ is the vector of parameters, taking values in a compact hyperrectangle $\Theta \subset \mathbb{R}^k$.

We write $\mathscr{M}_\theta$ to stress the dependency of $\mathscr{M}$ on the parameters $\theta \in \Theta$. A PCRN $\mathscr{M}_\theta$ defines a Continuous Time Markov Chain (Norris 1998; Bortolussi et al. 2013) on D, with infinitesimal generator Q, where $Q_{x,y} = \sum_{r_j \in \mathscr{R}} \{\alpha_j(x,\theta) \mid y = x + v_j\}$, $x \neq y$. We denote by P_θ the probability over the set of trajectories, or paths, $Path^{\mathscr{M}_\theta}$ of $\mathscr{M}_\theta$, equipped with the usual sigma-algebra (Baier et al. 2003).

2.2 Running Example: SIR Epidemic Spreading

In order to illustrate the definition and our methods, we will mostly rely on the popular SIR epidemic model (Gardner et al. 2000), which captures the spreading of a disease among a population of N individuals. There are three different agent classes (species):

- *susceptible S* individuals that are healthy and vulnerable to the infection;
- *infected I* individuals that are actively spreading the disease;
- *recovered R* individuals, which gained immunity to the disease.

The basic version of the SIR model is defined by chemical reactions r_1 and r_2 below:

$$\begin{array}{lll} r_1 : S + I \xrightarrow{f_1} 2I & f_1 = k_i/N \cdot X_s \cdot X_i & r_3 : R \xrightarrow{f_3} S \quad f_3 = k_s \cdot X_r \\ r_2 : I \xrightarrow{f_2} R & f_2 = k_r \cdot X_i & r_4 : S \xrightarrow{f_4} I \quad f_4 = k_e \cdot X_s \end{array}$$

where r_1 describes the infection of a healthy individual due to the contact with an infected one, while r_2 models the recovery of an infected agent. By adding r_3, we obtain the SIRS model, in which recovered individuals may lose immunity to the disease and get infected again. r_4, instead, models an infection from an external source, not explicitly described.

2.3 Behavioral Properties: Metric Interval Temporal Logic

When we model a complex system, we are usually interested in some of its behavioral properties. These may range from safety properties (the system stays in a safe zone), to reachability properties (the system reaches a specific set of configurations), to more quantitative aspects expressible as a cost or reward (e.g., the queue length in a performance model). These are typically *linear time properties*, meaning that they can be verified and monitored on single trajectories of the system. A general form of a linear time property is thus a functional, mapping each trajectory to a real number (possibly a subset of the reals, like integers or Booleans). We are typically interested in the expectations of such functionals. In case of reachability properties, this corresponds to the probability of observing a trajectory entering into the region of interest, while for quantitative properties, this is the average cost or reward.

In quantitative formal methods, it is typical to define a formal language to express such behavioral properties. The typical choice is a temporal logic, possibly enriched by reward operators. In this paper, we consider Metric Interval Temporal Logic (MITL, Alur et al. 1996), also known as Signal Temporal Logic (STL, Maler and Nickovic 2004), which is a linear time temporal logic that allows for reasoning about the future evolution of a trajectory in continuous time. Temporal logics are modal, i.e., they extend propositional logic with operators predicating properties about future configurations. In MITL, temporal operators predicate that a certain property φ will **eventually** hold in some instant $t \in [T_1, T_2]$ in the future, $\mathbf{F}_{[T_1,T_2]}\varphi$, or that it **globally** holds for all instants $t \in [T_1, T_2]$, $\mathbf{G}_{[T_1,T_2]}\varphi$, or that a property φ_1 holds **until** another property φ_2 becomes true, in some future instant $t \in [T_1, T_2]$, $\varphi_1 \mathbf{U}_{[T_1,T_2]} \varphi_2$. Note that these other properties φ are in fact other MITL properties: this nesting allows us to express very complex temporal patterns. The atomic predicates of MITL are inequalities on a set of real-valued variables, i.e., of the form $\mu(X) := [g(X) \geq 0]$, where $g : \mathbb{R}^n \to \mathbb{R}$ is a continuous function; consequently $\mu : \mathbb{R}^n \to \{\texttt{true}, \texttt{false}\}$. Furthermore, temporal operators considered here are all time-bounded, meaning that time-bounded trajectories are sufficient to verify any formula. We present now the syntax of the logic.

Definition 2 A formula $\varphi \in \mathscr{F}$ of MITL is defined by the following syntax:

$$\varphi := \texttt{false} \,|\, \texttt{true} \,|\, \mu \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \varphi \mathbf{U}_{[T_1,T_2]} \varphi,$$

where μ are atomic predicates as defined above, and $T_1 < T_2 < +\infty$. Eventually and globally modal operators can be defined as customary from the until, as $\mathbf{F}_{[T_1,T_2]}\varphi \equiv \texttt{true}\mathbf{U}_{[T_1,T_2]}\varphi$ and $\mathbf{G}_{[T_1,T_2]}\varphi \equiv \neg\mathbf{F}_{[T_1,T_2]}\neg\varphi$.

Example: Consider the SIR model introduced in the section above. Some typical properties of interest are naturally expressed in MITL:

- Between days 20 and 30, the number of infected peaks over 40% of the population: $\mathbf{F}_{[20,30]}[X_I \geq 0.4N]$;
- The epidemic terminates between days 70 and 100: $[X_I \geq 1]\mathbf{U}_{[70,100]}[X_I \leq 0]$;
- The final size of the epidemic is between 50% and 60% of the population (assuming the epidemics is over after 200 days): $\mathbf{G}_{[200,500]}[0.5N \leq X_R \leq 0.6N]$.

MITL formulae are interpreted over the paths $x(t)$ of a PCRN $\mathscr{M}_\theta$. The Boolean semantics of Maler and Nickovic (2004) assigns to each trajectory $x(t)$ and initial time t_0 a truth value. We write $(x,t_0) \models \varphi$ to indicate that $x(t)$, starting from time t_0, satisfies φ. The definition of the semantics proceeds recursively on the formula structure. The cases corresponding to atomic predicates and Boolean operators are standard, and are evaluated pointwise at a given time t. The until temporal modality $\varphi_1\mathbf{U}_{[T_1,T_2]}\varphi_2$ holds for path $x(t)$ at time t_0, $(x,t_0) \models \varphi_1\mathbf{U}_{[T_1,T_2]}\varphi_2$, if and only if there is a $t \in [t_0+T_1, t_0+T_2]$ such that $(x,t) \models \varphi_2$, while for all $\tau \in [t_0,t]$, $(x,\tau) \models \varphi_1$. More details about the Boolean semantics and its monitoring algorithms are given by Maler and Nickovic (2004).

The Boolean semantics can be combined with the probability distribution P_θ over trajectories induced by a PCRN model $\mathscr{M}_\theta$, to obtain the satisfaction probability of a formula φ:

$$P_\varphi(\theta) \equiv P(\varphi \mid \mathscr{M}_\theta) := P_\theta(\{x(t) \in Path^{\mathscr{M}_\theta} \mid (x,0) \models \varphi\}).$$

In Bortolussi et al. (2016a), $P_\varphi(\theta)$ was shown to be a *smooth function* of parameters θ, for PCRN models with polynomial rate functions. This covers most models used in practice, though we conjecture that smoothness holds also for the more general class of smooth rate functions.

An alternative semantics for MITL formulae emerged in recent years, assigning a real value to each formula, rather than a Boolean (Donzé and Maler 2010; Fainekos and Pappas 2009). The sign of such a value tells us the truth of the formula (with positive being `true`), while the absolute number is a measure of the robustness of the satisfaction, i.e., of how much we need to perturb a trajectory at any point to change the truth status of the formula.

Definition 3 (MITL Quantitative Semantics) The *quantitative satisfaction function* ρ returns a value $\rho(\varphi,x,t) \in \mathbb{R} \cup \{-\infty,+\infty\}$, quantifying the robustness degree (or satisfaction degree) of the property φ by the trajectory x at time t. It is defined recursively as follows:

$$\begin{aligned}
\rho(\texttt{true},x,t) &= +\infty \\
\rho(\mu,x,t) &= g(x(t)) \text{ where } \mu \equiv g(x(t)) \geq 0 \\
\rho(\neg\varphi,x,t) &= -\rho(\varphi,x,t) \\
\rho(\varphi_1 \wedge \varphi_2,x,t) &= \min(\rho(\varphi_1,x,t),\rho(\varphi_2,x,t)) \\
\rho(\varphi_1\,\mathbf{U}_{[T_1,T_2]}\varphi_2,x,t) &= \sup_{t' \in t+[a,b]} (\min(\rho(\varphi_2,x,t'), \inf_{t'' \in [t,t')} (\rho(\varphi_1,x,t''))))
\end{aligned}$$

Similarly to the Boolean case, where we combine the concept of quantitative semantics with the probability over paths, we obtain a real-valued random variable, of which we are mostly interested in its expectation:

$$R_\varphi(\theta) \equiv \mathbb{E}_{x \sim \mathscr{M}_\theta}[\rho(\varphi,x,0)].$$

MITL is not the only choice to express behavioral properties. Alternatives rely on timed regular expressions (Asarin et al. 2002) or Deterministic Timed Automata (Bortolussi and Hillston 2015). Furthermore, we can also enrich the PCRN model with rewards or costs (Kwiatkowska et al. 2011), which are functions of the current state, $r = r(x)$. In this case, we are typically interested in the expected instantaneous reward at time T or the expected cumulative reward up to time T. All our methods work also for this extended class of properties, the only requirement being *continuity* with respect to θ of probabilities $P_\varphi(\theta)$ and expectations $R_\varphi(\theta)$.

2.4 Statistical Verification of MITL Properties

Verification of MITL properties for a given PCRN $\mathscr{M}_\theta$, for fixed θ, amounts to compute $P_\varphi(\theta)$ (or $R_\varphi(\theta)$). Doing this with numerical algorithms is extremely costly from a computational point of view (Chen et al. 2011), hence the typical approach is to rely on statistical estimates, which simulate trajectories of the PCRN and then check on each trajectory if the formula φ holds (resp. compute the quantitative semantics), thus generating observations of Bernoulli (resp. real-valued) random variables. Standard frequentist (Younes and Simmons 2006) or Bayesian (Jha et al. 2009) statistics can be then be used to estimate $P_\varphi(\theta)$ (resp. $R_\varphi(\theta)$) or to test if $P_\varphi(\theta) > p$. A recent survey is given by Agha and Palmskog (2018).

3 PROBLEM STATEMENT

Given an MITL formula φ and a PCRN model $\mathscr{M}_\theta$, we consider the following verification task:

compute or estimate the satisfaction probability $P_\varphi(\theta)$
as a function of $\theta \in \Theta$ (Parametric verification)

Our approach to this problem is statistic: we will simulate the model and check the MITL property at several points of the parameter space, possibly conducting few runs per point, and use these observations to reconstruct the satisfaction probability function $P_\varphi(\theta)$. To this end, we will rely on Bayesian methods based on Gaussian Processes. Technically, the key ingredient underpinning our approach is the regularity result proved by Bortolussi et al. (2016a), which says that, under very mild conditions, the satisfaction probability of a temporal logic property is a smooth (infinitely differentiable) function of the model parameters. This enables us to recast the parametric verification problem as a function approximation problem, leveraging decades of machine learning literature on global approximations of input-output relationships (in our case, the inputs are parameter values, and the outputs are simulation results). In particular, Gaussian Processes provide a rigorous way to constrain the space of functions to the smooth family. As a side effect, we will not only have an estimate of $P_\varphi(\theta)$ for each θ, but also statistical confidence bounds that allow us to quantify the accuracy of the reconstruction, and pave the way to active learning methods, which are exploited for parameter synthesis and system design purposes.

It is worth remarking that our method is not restricted to PCRN or to MITL properties: we just need that the satisfaction probability function (or whatever reward we want to estimate) depends smoothly (though continuously is enough) on the parameters.

4 GAUSSIAN PROCESSES

Bayesian statistical model checking provides an effective way to verify temporal logic properties given a complete specification of the problem. The parametric verification task, however, requires us to estimate a satisfaction function. While in principle this could be achieved by repeated applications of Bayesian SMC (e.g., over a grid of parameter values), this is likely to be highly inefficient. A more elegant solution is to directly frame the parametric verification problem as a Bayesian model checking problem in a functional space; to do so, we need to consider distributions over functions. The natural choice falls on the non-parametric class of regression models known as Gaussian Processes (Rasmussen and Williams 2006).

To understand GP, consider a random function defined by a linear combination of a fixed set of basis functions $b_i(x)$, $i = 1,\ldots,m$, with $x \in \mathbb{R}^k$, where coefficients are drawn randomly from a Gaussian distribution. Specifically, let

$$f(x) = \sum_{i=1}^{m} w_i b_i(x), \quad \text{where } w_i \sim \mathscr{N}(0,\sigma^2).$$

With this definition, it is easy to see that, for any $x_1,\ldots,x_n$, $f(x_1),\ldots,f(x_n)$ is a multivariate Gaussian distribution, with mean $\mathbb{E}[f(x_j)] = 0$ for each x_j, and with covariance $COV[f(x_i), f(x_j)] = b(x_i)^T \cdot b(x_j)$

given by the dot product of the vectors of basis functions evaluated at points x_i and x_j. By defining the covariance or kernel function $k(x_i,x_j) = b(x_i)^T \cdot b(x_j)$, we can see that we do not need to refer explicitly to the basis functions to describe the joint distribution of $f(x_1),\ldots,f(x_n)$: knowledge of the kernel k is enough. Using more general kernels than dot product of basis function, we can thus consider a much larger class of random functions, possibly corresponding to function spaces of infinite dimensionality. What we obtain in this way is a Gaussian Process.

Technically, a GP is a stochastic process, i.e., a collection of random variables $\{f(x)\}_{x\in E}$, indexed by $x \in E$, a compact subset of $\mathbb{R}^k$, a taking values $f(x) \in \mathbb{R}$. The characterizing property of such collections is that any finite subset of them, i.e., $f(x_1),\ldots,f(x_n)$, jointly follows a multivariate normal distribution. In this way, a GP can be viewed as an infinite-dimensional generalization of a multivariate normal distribution: a function can be viewed as an infinite-dimensional vector whose entries are indexed by the function's input value. A GP is uniquely determined by its mean and covariance functions (called also kernels) denoted respectively with $m : E \to \mathbb{R}$ and $k : \mathbb{E}\times\mathbb{E} \to \mathbb{R}$: for every finite set of points $(x_1,x_2,\ldots,x_n)$:

$$f \sim \mathscr{GP}(m,k) \iff (f(x_1),f(x_2),\ldots,f(x_n)) \sim \mathscr{N}(m,K)$$

where $m = (m(x_1),m(x_2),\ldots,m(x_n))$ is the vector mean, $K \in \mathbb{R}^{n\times n}$ is the covariance matrix, such that $K_{ij} = k(f(x_i),f(x_j))$, and $\mathscr{N}$ is the normal distribution. From a functional point of view, GP is a probability distribution on the set of functions $f : \mathscr{E} \to R$.

The choice of the covariance function determines which functions will be sampled with higher probability from a GP (see Chapter 4 of Rasmussen and Williams 2006), typically by imposing smoothness constraints. A common choice, which we will also use in this paper, is to work with the Gaussian Radial Basis Function (GRBF) kernel (Rasmussen and Williams 2006). The kernel is defined as

$$k(x_1,x_2) = \exp(-\|x_1 - x_2\|^2/l^2),$$

where $\|\cdot\|$ is the Euclidean norm and l is the lengthscale hyperparameter, which governs how far away observations are contributing to predictions in a point (as if x^* and x_i are much more distant than l, then $k(x^*,x_i)$ is approximately zero). Note that l determines the Lipschitz constant of the GRBF kernel, which is $l^{-1}\sqrt{2e^{-1}}$. Sampled functions from a GP equipped with this kernel are smooth (infinitely differentiable) with probability one. Furthermore, the GRBF kernel enjoys the universality property: samples from a GP with GRBF can approximate arbitrarily well any continuous function on a compact set E.

GP are used in Bayesian non-parametric regression and classification. Starting from a training set $(x_i,\mathscr{O}_i)$ of input x_i and output $\mathscr{O}_i$ pairs, an observation noise model $p(\mathscr{O}_i|f(x_i))$ and a prior GP, typically with zero mean and a given covariance function, GP regression computes a posterior process. Our interest, in particular, is to compute the distribution over f at a *new* input point x^* *given* the observed values $\mathscr{O} = \{\mathscr{O}_1,\ldots,\mathscr{O}_N\}$, $p(f(x^*)|\mathscr{O})$. To solve this inference task, we can use the fact that the function values at any finite collection of input points are Gaussian distributed:

$$p(f(x^*),f(x_1),\ldots,f(x_N)) = \mathscr{N}(m,K)$$

with m and K obtained from the mean and covariance function as explained before. This prior distribution can be combined with likelihood models for the observations, $p(\mathscr{O}|f)$, with $f = (f(x_1),\ldots,f(x_N))$, using Bayes theorem to get a joint posterior

$$p(f(x^*),f(x_1),\ldots,f(x_N)|\mathscr{O}) = \frac{1}{Z}p(f(x^*),f(x_1),\ldots,f(x_N))\prod_i p(\mathscr{O}_i|f(x_i)) \tag{1}$$

where Z is a normalization constant. As typical in Bayesian inference, we look for the posterior predictive distribution $p(f(x^*|\mathscr{O})$, which can then be obtained by *marginalizing* (integrating out) the true function values $f(x_1),\ldots,f(x_N)$ from the previous equation:

$$p(f(x^*|\mathscr{O}) = \int\prod_{i=1}^{N} df(x_i)p(f(x^*),f(x_1),\ldots,f(x_N)|\mathscr{O}). \tag{2}$$

The previous equation plays a central role in non-parametric function estimation. The inference procedure outlined above is typically known as *GP regression*. It is important to note that, in the case of Gaussian observation noise, the integral in the equation above can be computed in closed form, obtaining a linear combination of kernel function evaluations at a test point x^* and observation points x_i with coefficients depending on the observations y_i. The prior kernel hyperparameters, such as the lengthscale in the GRBF kernel, can be set automatically by optimizing the marginal likelihood of the observations, as customary in Bayesian inference. Further details are given, e.g., in Chapters 2 and 3 of Rasmussen and Williams (2006).

5 SMOOTHED MODEL CHECKING

Smoothed Model Checking (smmc) uses GPs to solve the parametric verification task. It estimates globally the function $P_\varphi(\theta)$ by regressing the truth value of φ, evaluated using few simulation runs, against the underlying parameter values $\theta_1, \ldots, \theta_n$. Hence, observations $\mathscr{O}_j$ at θ_j are samples of a Binomial random variable with probability $P_\varphi(\theta_j)$. The resulting GP posterior mean is the estimated satisfaction function, and posterior GP confidence intervals provide statistical bounds on the function at individual points.

As customary in the GP setting, the method tries to learn a real-valued latent function $f(\theta)$, which induces the desired function with values in $[0,1]$ by combining it with the Probit transform:

$$P_\varphi(\theta) = \Psi(f(\theta)).$$

which is the cumulative distribution function at x of a standard normal distribution, i.e., the probability of $(-\infty, x]$.

Smoothed Model Checking plugs observations $\mathscr{O}$ into a Bayesian inference scheme based on GP, assuming a prior GP $p(f)$ for the latent function f, described by specifying its mean and kernel function.

Following the steps of the previous section, by an application of Bayes theorem we obtain equation (1), in which the noise model $p(\mathscr{O}_j \mid f(\theta_j))$ is given by a Binomial density. The predictive distribution is given by equation (2). The integral, though, is analytically intractable, due to the presence of a Binomial observation model. This forces us to resort to an efficient variational approximation known as Expectation Propagation (Bortolussi et al. 2016a; Rasmussen and Williams 2006), resulting in a Gaussian predictive distribution for $p(f(\theta^*) \mid \mathscr{O})$, whose mean and δ-confidence region are then Probit transformed into $[0,1]$.

It is important to stress that the prediction of Smoothed Model Checking, being a Bayesian method, depends on the choice of the prior. In case of Gaussian Processes, choosing the prior means fixing a covariance function that makes assumptions on the smoothness and density of the functions that can be sampled by the GP. The set of functions having positive probability under a GP with Gaussian Radial Basis Function is dense in the space of continuous functions over a compact set (Steinwart 2002), hence it can approximate arbitrarily well the satisfaction probability function, which is a smooth function of model parameters (for PCRN with polynomial rate functions). This approximation, however, holds in the limit of an infinite number of observations. When working with a finite number of samples, accuracy is typically encoded in the uncertainty bounds: the larger the bounds, the less accurate the reconstruction. The number of observations needed to obtain a tight prediction depends on the one hand on the function to learn, and on the other hand on the choice of hyperparameters: a bad choice will considerably slow down the convergence. In our context, by setting the lengthscale of the GRBF kernel via marginal likelihood optimization, we are making the best guess for prior given the observed data.

More generally, Smoothed Model Checking will in principle work whenever the satisfaction probability function is continuous, and even only measurable, owing to the fact that measurable functions over a compact set can be approximated by a sequence of smooth functions. However, this may work only in the limit of a very large number of observations, as the quality of GP approximation for a finite number of observations for non-Lipschitz continuous or discontinuous functions is typically poor.

It is also worth stressing that the statistical asymptotic guarantees of the method hold up to the use of Expectation Propagation (EP) in the inference procedure, as EP is an approximate technique.

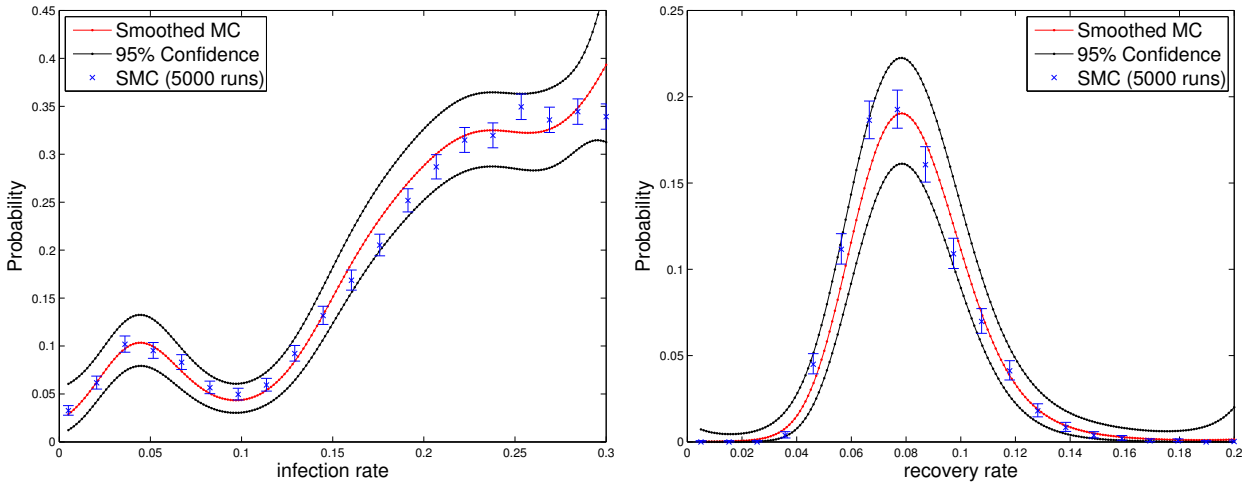

Figure 1: Example of smmc varying k_i (left) and k_r (right). Satisfaction probability functions are reconstructed from 2,000 simulation runs overall. Compare the precision with the estimates for fixed parameters (vertical bars), each requiring 5,000 simulation runs.

Example: We return to the SIR model described in Section 2.2, and use smmc to estimate the probability of extinction of the epidemics between 100 and 120 days from the outbreak:

$$\varphi = [X_I \geq 1]\mathbf{U}_{[100,120]}[X_I \leq 0]$$

The results of smmc, varying either the infection rate k_i or the recovery rate k_r, are shown in Figure 1. Reconstruction of the satisfaction functions took only 2000 simulation runs in total. A comparison with standard statistical model checking, which can be found in (Bortolussi et al. 2016a), restricting on a fixed grid of points, shows the advantage of smmc: one needs one tenth of simulation runs to obtain the same level of accuracy in such grid points. GP inference propagates information to neighboring points, hence reducing the cost of analysis, and providing estimates in points which are not explored. We refer the reader to Bortolussi et al. (2016a) for a detailed presentation of the method. There is also a Java tool, U-check (Bortolussi et al. 2015), implementing smmc, that supports PCRN and models written in the PRISM language (Kwiatkowska et al. 2011).

6 APPLICATIONS

The Bayesian approach of smmc opens up the way to further applications. There are two key ingredients: the availability of an analytical surrogate of the true satisfaction function, which can be optimized and evaluated efficiently, and the availability of uncertainty estimates, which tell us how accurate the reconstruction is in a certain point. In fact, at each point of the parameter space, we have a distribution that provides much more information than just bounds. This last aspect, in particular, can be integrated into active learning and Bayesian optimization schemes, to efficiently solve design and synthesis tasks.

In the following, we discuss applications to system design, parameter synthesis, search-based falsification, and requirement learning. An application of this framework to system identification from Boolean observations of temporal properties can be found in Bortolussi and Sanguinetti (2015).

6.1 System Design

An obvious application of smmc is to identify parameter values that satisfy a property with high probability. We tackled this problem in Bartocci et al. (2015), where we noticed that satisfaction probability has many flat regions (typically equal to one or zero), making the answer undetermined. A better solution is to use

the average robustness score, described in Section 2.3, which highly correlates with satisfaction probability. In this case, smmc reduces to do a regression using noisy observations of average robustness. The system design problem is then that of finding the value of parameters θ that optimize the average robustness.

A naive idea to find such optimum would be to optimize the surrogate function learned by GP regression. This would, however, produce bad solutions, as we may either miss information from some important regions of the state space, or need too many simulations to have an accurate model to identify the optimum. A better strategy is to rely on active learning and Bayesian optimization, in particular to the GP-Upper Confidence Bound algorithm (Srinivas et al. 2012). The idea of this approach is to optimize not the average, but an upper confidence bound of the GP, which is large when either the average or the variance is large, obtaining a candidate maximum θ^*. The model is then simulated again at such candidate θ^*, and smmc run again, to obtain a more accurate reconstruction around θ^*. This optimization-sampling-regression scheme is then repeated until convergence, i.e., until the algorithm cannot find a better solution. Depending whether θ^* was coming from a region of high mean or of high variance, the algorithm focuses on exploitation or exploration, resulting in a natural trade-off between these two aspects. More details on system design can be found in Bartocci et al. (2015).

Similar ideas have also been applied to multi-objective design (Bortolussi et al. 2016b), where the identification of the Pareto front was performed by relying on a hybrid genetic algorithm.

This GP-UCB optimization scheme has been applied in Bartocci et al. (2014) for the requirement synthesis problem. The goal is to optimize the parameters of a fixed MITL template formula φ, in order to learn a formula that effectively discriminates between a good and a bad set of trajectories. This method was combined in Bufo et al. (2014) with an evolutionary algorithm to search also for the best formula template, and applied to a medical Cyber-Physical System.

6.2 Parameter Synthesis

Another application of smmc is to the parameter synthesis of stochastic models. We tackle this problem in Bortolussi and Silvetti (2018), where smmc has been applied to explore and identify the regions of the parameter space where the satisfaction probability of a target MITL property φ is above or below a given threshold. The idea is to leverage the GP surrogate model learned by smmc, in order to compute the GP probability that the satisfaction probability of φ is above or below a specific threshold. This confidence probability is then used to drive an active learning strategy which samples parameters θ where this value is as close as possible to 0.5. This value corresponds to the largest possible uncertainty on whether the probability of satisfaction of φ is above or below the predefined threshold. Then, the algorithm runs few simulations at each such θ, calling smmc again to refine the GP estimation of the satisfaction probability of φ near such areas of uncertainty. In Figure 2, we show the parameter synthesis algorithm at work for the SIR model. Notice how the density of sampled points (black dots) is larger in the areas where the satisfaction probability is close to the threshold.

6.3 Search-Based Falsification

Search-based falsification consists of using optimization techniques to identify a configuration of the model (i.e., of some of its parameters) that causes unexpected behaviors. These parameters are generally called counterexamples. We addressed this problem in a deterministic model of the automotive field (Silvetti et al. 2017), looking for a specific driving style (described by the throttle and brake angle dynamics) causing the car engine to violate emission requirements, expressed as MITL formulae. Leveraging the soundness property of the robustness semantics of MITL, it is sufficient to identify parameters that cause robustness to be below zero.

Our strategy leveraged the GP approximation of the robustness semantics of the target MITL requirement, computing the probability that the quantitative semantics assume a negative value at each point of the parameter space, and using this information to sample points that have a high probability of violating φ.

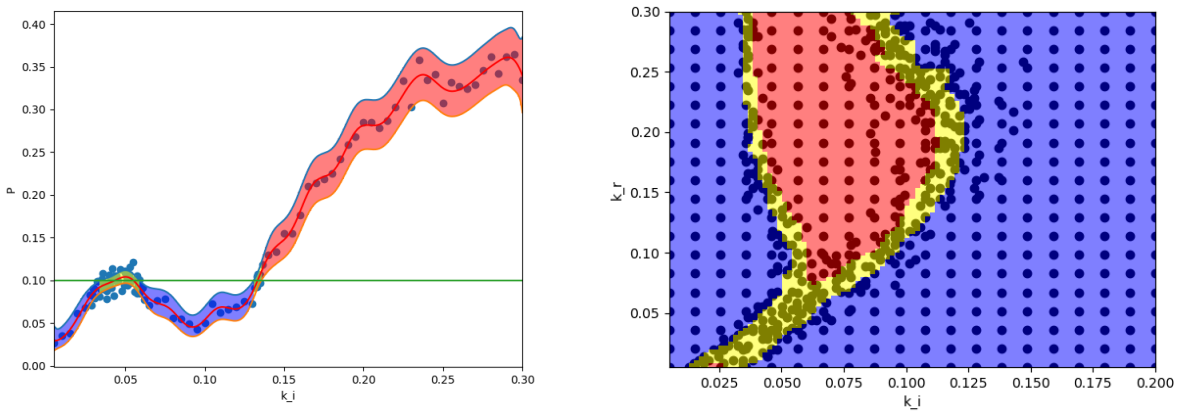

Figure 2: Example of parameter synthesis for the SIR model, looking for infection rate k_i (left) and both infection k_i and recovery rate k_r (right). In blue regions, probability is below the threshold and in red regions it is above with a confidence probability of at least 0.95, while yellow regions are the uncertain ones meaning that the probability is above or below the threshold with not enough confidence.

Plugging this into an active learning strategy, we then simulate the model and evaluate the true robustness score at these points. If their value is negative, we have found counterexamples; otherwise we refine the GP model and iterate the scheme. Iterating this scheme, the probability to sample a counterexample rapidly increases, allowing us to keep the number of model simulations, a typically costly operation, low.

7 DISCUSSION

Modern engineering and scientific applications require effective modeling techniques that can deliver accurate analyses at scale. Formal verification methods have been invaluable in the development of trustable software systems, yet the challenges presented by modern applications going beyond what is possible within the boundaries of traditional model checking. Machine learning methodologies are well suited to take advantage of the often very large amounts of data generated when tackling verification problems (e.g., by running multiple simulations).

In this paper, we describe a family of such methods based on Gaussian Processes. These methods use a flexible non-parametric Bayesian regression approach to effectively use internal simulation runs for global parametric verification. In its basic form, the Smoothed Model Checking approach proves significantly faster than methods based on brute-force exploration of the parameter space. Moreover, the availability of posterior confidence intervals enables the application of the approach to a number of related problems, such as system design and parameter synthesis. The approach is complemented by a freely available software suite, U-check (Bortolussi et al. 2015), which is compatible with major model checking suites such as PRISM.

REFERENCES

Agha, G., and K. Palmskog. 2018. "A Survey of Statistical Model Checking". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28(1), article 6.

Alur, R., T. Feder, and T. A. Henzinger. 1996. "The Benefits of Relaxing Punctuality". *J. ACM* 43(1):116–146.

Alur, R., T. A. Henzinger, and M. Y. Vardi. 2015. "Theory in Practice for System Design and Verification". *ACM SIGLOG News* 2(1):46–51.

Asarin, E., P. Caspi, and O. Maler. 2002. "Timed Regular Expressions". *J. ACM* 49(2):172–206.

Baier, C., B. Haverkort, H. Hermanns, and J.-P. Katoen. 2003. "Model-checking Algorithms for Continuous-time Markov Chains". *IEEE Transactions on Software Engineering* 29(6):524–541.

Bartocci, E., L. Bortolussi, and G. Sanguinetti. 2014. "Data-driven Statistical Learning of Temporal Logic Properties". In *Formal Modeling and Analysis of Timed Systems*, edited by A. Legay and M. Bozga, 23–37. Cham: Springer International Publishing.

Bartocci, E., L. Bortolussi, L. Nenzi, and G. Sanguinetti. 2015. "System Design of Stochastic Models Using Robustness of Temporal Properties". *Theoretical Computer Science* 587:3–25.

Bortolussi, L., and J. Hillston. 2015. "Model Checking Single Agent Behaviours by Fluid Approximation". *Information and Computation* 242:183–226.

Bortolussi, L., and G. Sanguinetti. 2015. "Learning and Designing Stochastic Processes from Logical Constraints". *Logical Methods in Computer Science* 11(2).

Bortolussi, L., and S. Silvetti. 2018. "Bayesian Statistical Parameter Synthesis for Linear Temporal Properties of Stochastic Models". In *Tools and Algorithms for the Construction and Analysis of Systems*, edited by D. Beyer and M. Huisman, 396–413. Cham: Springer International Publishing.

Bortolussi, L., J. Hillston, D. Latella, and M. Massink. 2013. "Continuous Approximation of Collective Systems Behaviour: A Tutorial". *Performance Evaluation* 70(5):317–349.

Bortolussi, L., D. Milios, and G. Sanguinetti. 2015. "U-Check: Model Checking and Parameter Synthesis Under Uncertainty". In *Quantitative Evaluation of Systems*, edited by J. Campos and B. R. Haverkort, 89–104. Cham: Springer International Publishing.

Bortolussi, L., D. Milios, and G. Sanguinetti. 2016a. "Smoothed Model Checking for Uncertain Continuous-time Markov Chains". *Information and Computation* 247:235–253.

Bortolussi, L., A. Policriti, and S. Silvetti. 2016b. "Logic-based Multi-objective Design of Chemical Reaction Networks". In *Hybrid Systems Biology*, edited by E. Cinquemani and A. Donzé, 164–178. Cham: Springer International Publishing.

Bortolussi, L., R. Lanciani, and L. Nenzi. 2017. "Model Checking Markov Population Models by Stochastic Approximations". *arXiv:1711.03826 [cs]*.

Brim, L., M. Češka, S. Dražan, and D. Šafránek. 2013. "Exploring Parameter Space of Stochastic Biochemical Systems Using Quantitative Model Checking". In *Computer Aided Verification*, edited by N. Sharygina and H. Veith, 107–123. Berlin, Heidelberg: Springer.

Bufo, S., E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi. 2014. "Temporal Logic Based Monitoring of Assisted Ventilation in Intensive Care Patients". In *Leveraging Applications of Formal Methods, Verification and Validation*, Part II, edited by T. Margaria and B. Steffen, 391–403. Berlin Heidelberg: Springer.

Češka, M., F. Dannenberg, M. Kwiatkowska, and N. Paoletti. 2014. "Precise Parameter Synthesis for Stochastic Biochemical Systems". In *Computational Methods in Systems Biology*, edited by P. Mendes et al., 86–98. Cham: Springer International Publishing.

Češka, M., P. Pilař, N. Paoletti, L. Brim, and M. Kwiatkowska. 2016. "PRISM-PSY: Precise GPU-Accelerated Parameter Synthesis for Stochastic Systems". In *Tools and Algorithms for the Construction and Analysis of Systems*, edited by M. Chechik and J.-F. Raskin, 367–384. Berlin, Heidelberg: Springer.

Chen, T., M. Diciolla, M. Kwiatkowska, and A. Mereacre. 2011. "Time-bounded Verification of CTMCs against Real-Time Specifications". In *Formal Modeling and Analysis of Timed Systems*, edited by U. Fahrenberg and S. Tripakis, 26–42. Berlin, Heidelberg: Springer.

Dehnert, C., S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, and E. Ábrahám. 2015. "PROPhESY: A PRObabilistic ParamEter SYnthesis Tool". In *Computer Aided Verification*, edited by D. Kroening and C. S. Păsăreanu, 214–231. Cham: Springer International Publishing.

Dias Neto, A. C., R. Subramanyan, M. Vieira, and G. H. Travassos. 2007. "A Survey on Model-based Testing Approaches: A Systematic Review". In *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies*, WEASELTech '07, 31–36. New York, NY, USA: ACM.

Donzé, A., and O. Maler. 2010. "Robust Satisfaction of Temporal Logic over Real-valued Signals". In *Formal Modeling and Analysis of Timed Systems*, edited by K. Chatterjee and T. A. Henzinger, 92–106. Berlin, Heidelberg: Springer.

Fainekos, G. E., and G. J. Pappas. 2009. "Robustness of Temporal Logic Specifications for Continuous-time Signals". *Theoretical Computer Science* 410:4262–4291.

Franceschinis, G., and R. R. Muntz. 1994. "Bounds for Quasi-lumpable Markov Chains". *Performance Evaluation* 20(1):223–243.

Gardner, T. S., C. R. Cantor, and J. J. Collins. 2000. "Construction of a Genetic Toggle Switch in Escherichia Coli". *Nature* 403(6767):339–342.

Gillespie, D. T. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions". *The Journal of Physical Chemistry* 81(25):2340–2361.

Henzinger, T. A., B. Jobstmann, and V. Wolf. 2009. "Formalisms for Specifying Markovian Population Models". In *Reachability Problems*, edited by O. Bournez and I. Potapov, 3–23. Berlin, Heidelberg: Springer.

Jha, S. K., E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. 2009. "A Bayesian Approach to Model Checking Biological Systems". In *Computational Methods in Systems Biology*, edited by P. Degano and R. Gorrieri, 218–234. Berlin, Heidelberg: Springer.

Katoen, J.-P. 2016. "The Probabilistic Model Checking Landscape". In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, 31–45. New York, NY, USA: ACM.

Kattenbelt, M., M. Kwiatkowska, G. Norman, and D. Parker. 2009. "Abstraction Refinement for Probabilistic Software". In *Verification, Model Checking, and Abstract Interpretation*, edited by N. D. Jones and M. Müller-Olm, 182–197. Berlin, Heidelberg: Springer.

Kwiatkowska, M., G. Norman, and D. Parker. 2011. "PRISM 4.0: Verification of Probabilistic Real-Time Systems". In *Computer Aided Verification*, edited by G. Gopalakrishnan and S. Qadeer, 585–591. Berlin, Heidelberg: Springer.

Lee, E. A. 2008, May. "Cyber Physical Systems: Design Challenges". In *11th IEEE International Symposium on Object and Component-oriented Real-Time Distributed Computing (ISORC)*, May 5th–7th, Orlando, FL, USA, 363–369.

Maler, O., and D. Nickovic. 2004. "Monitoring Temporal Properties of Continuous Signals". In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, edited by Y. Lakhnech and S. Yovine, 152–166. Berlin, Heidelberg: Springer.

Norris, J. R. 1998. *Markov Chains*. Cambridge, UK: Cambridge University Press.

Peleska, J. 2013. "Industrial-Strength Model-based Testing – State of the Art and Current Challenges". *Proceedings Eighth Workshop on Model-Based Testing*, edited by A. K. Petrenko and H. Schlingloff, March 17th, Rome, Italy, 3–28. Electronic Proceedings in Theoretical Computer Science 111.

Rasmussen, C. E., and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Adaptive computation and machine learning. MIT Press.

Silvetti, S., A. Policriti, and L. Bortolussi. 2017. "An Active Learning Approach to the Falsification of Black Box Cyber-Physical Systems". In *Integrated Formal Methods*, edited by N. Polikarpova and S. Schneider, 3–17. Cham: Springer International Publishing.

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger. 2012. "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting". *IEEE Transactions on Information Theory* 58(5):3250–3265.

Steinwart, I. 2002. "On the Influence of the Kernel on the Consistency of Support Vector Machines". *The Journal of Machine Learning Research* 2(Nov):67–93.

Younes, H. L., and R. G. Simmons. 2006. "Statistical Probabilistic Model Checking with a Focus on Time-bounded Properties". *Information and Computation* 204(9):1368–1409.

AUTHOR BIOGRAPHIES

LUCA BORTOLUSSI is associate Professor of Computer Science at the Department of Mathematics and Geosciences of the University of Trieste, Italy. His research is aimed at improving scalability and usability of quantitative formal methods, using machine learning and stochastic approximation, and to apply these methods to concrete scenarios. His e-mail address is lbortolussi@units.it.

GUIDO SANGUINETTI is Professor of Computational Bioinformatics in the School of Informatics, University of Edinburgh. His main interests are in the use and development of new machine learning tools within models of biological systems, particularly dynamical systems. He has published over 80 peer-reviewed papers in international journals and conferences including Science, Nature Methods, Nature Communications, and PNAS. He was the recipient of an ERC Starting Grant, the 2012 PNAS Cozzarelli Prize in Applied Sciences, and the 2016 ECCB best paper award. His e-mail address is gsanguin@inf.ed.ac.uk.

SIMONE SILVETTI is a researcher and developer at the Numerical Methods Group of Esteco SpA. His research focuses on the application of machine learning to quantitative formal methods and optimization. His e-mail address is silvetti@esteco.com.