Università degli Studi di Udine
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Dottorato di Ricerca in Informatica e Scienze Matematiche e Fisiche

Ph.D. Thesis

Combining Machine Learning and Formal Methods for Complex Systems Design

Candidate
Simone Silvetti

Supervisor
Prof. Alberto Policriti

Co-Supervisor
Prof. Luca Bortolussi

Tutor
Dr. Enrico Rigoni

Cycle XXX — A.Y. 2018

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
http://www.dimi.uniud.it/

Author's Contacts
Via della raffineria, 7
34138 Trieste — Italia
http://simonesilvetti.com
simone.silvetti@gmail.com

Abstract

During the last 20 years, *model-based design* has become a standard practice in many fields such as automotive, aerospace engineering, systems and synthetic biology. This approach allows a considerable improvement of the final product quality and reduces the overall prototyping costs. In these contexts, formal methods, such as *temporal logics*, and *model checking* approaches have been successfully applied. They allow a precise description and automatic verification of the prototype's requirements.

In the recent past, the increasing market requests for performing and safer devices shows an unstoppable growth which inevitably brings to the creation of more and more complicated devices. The rise of *cyber-physical systems*, which are on their way to become massively pervasive, brings the complexity level to the next step and open many new challenges. First, the descriptive power of standard temporal logics is no more sufficient to handle all kind of requirements the designers need (consider, for example, non-functional requirements). Second, the standard model checking techniques are unable to manage such a level of complexity (consider the well-known curse of state space explosion). In this thesis, we leverage machine learning techniques, active learning, and optimization approaches to face the challenges mentioned above.

In particular, we define *signal convolution logic*, a novel temporal logic suited to describe non-functional requirements. We also use *evolutionary algorithms* and *signal temporal logic* to tackle a supervised classification problem and a system design problem which involves multiple conflicting requirements (i.e., *multi-objective optimization* problems). Finally, we use an active learning approach, based on *Gaussian processes*, to deal with falsification problems in the automotive field and to solve a so-called *threshold synthesis problem*, discussing an epidemics case study.

Contents

List of Tables

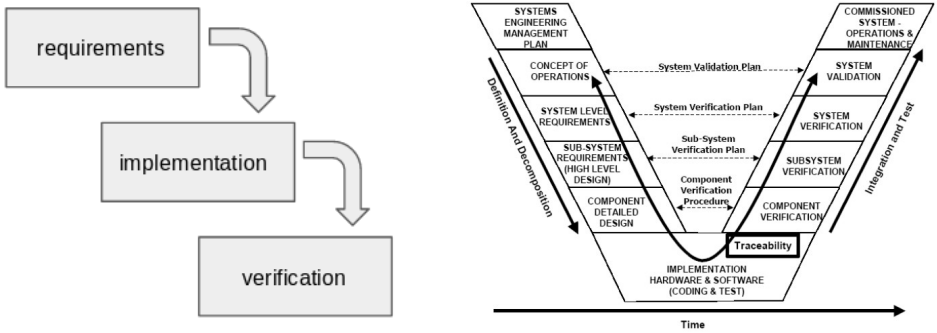
List of Figures

Introduction

Context

From past to present. During the last 20 years the market request of increasingly performing and safe devices shows an unstoppable growth. This trend is evident in many fields, beginning with software and embedded device development, passing through automotive and aerospace fields and arriving at system and synthetic biology. The effect of this pressing demand had a big impact on all levels of industry organization starting from the internal organization and ending with the adoption of new system development methodologies. In software engineering, for example, continuous delivery [HF10], test-driven development [Bec03] and agile approaches [Coc02] became *de facto* standard practices. These approaches allow reducing the time-to-market of the released product, improving productivity, efficiency, and final product quality. Moreover, they ensure the possibility to release software continuously over time, by fulfilling the demands of new functionality and following market's direction.

The system development cycle rapidly changes to face the market requests, as well. It moves from a monolithic waterfall chain (Figure 1a) to a more flexible "V" model (Figure 1b) which essentially replaces the previous chain with a new detailed and standard process.



(a) Waterfall Model

(b) V model

Figure 1: Development Cycle Models

The requirement phase which is represented by the left side of the "V" starts with the definition and decomposition of the requirements and ends with the creation of system

specifications. These specifications are then used to drive the implementation phase which is followed by the verification phase represented by the right side of the "V". The novelty of this approach results in the possibility of concurrent execution of the three steps described before, for example, the implementation and verification processes can start simultaneously, implying that the testing procedure is setting up *before* the implementation. This strategy forces the implemented prototype to satisfy the original requirements from the beginning, which results in the minimization of the project's risk and the improvement and assurance of quality of the final product.

Alongside the high-level vision which considers the product development cycle as a process to be optimized, there is a low-level vision which focuses on the three phases forming this process: the requirements definition, the implementation, and the verification. All these three levels have been improved during the last 20 years.

In many fields, the implementation phase passed from the use of imperative languages to object-oriented languages which more properly assist the development of complex and maintainable software. During 2000 many industries started to use the simulation-driven design processes, so that the *Model-Based Development* (MBD) quickly assumed a central role in system design. It aims to build computational models in place of real prototypes which can be simulated in order to verify their compliance with respect to original requirements. Graphical Programming environments for modeling and simulation, such as Simulink [Mat] and LabVIEW [Nat], quickly become standard tools in such fields as automatic control and signal processing [Cha17, TTS+08]. Their principal contribution was the management, creation, and testing of complex *in silico* models by avoiding the standard implementation of code. The ability to create accurate models and executing them, rapidly increased and the MBD approach was adopted in many engineering industries (mainly automotive and aerospace). These new software programs and approaches bring new descriptive power at the designer's disposal. As a consequence, the complexity of generated models increased and two outcomes arose. First, new challenges to the requirement definition and the verification phase were posed and secondly the adoption of *Multi-disciplinary Optimization* (MDO) (i.e., the numerical optimization approaches used to optimize specific model parameters) was granted.

A set of requirement specifications must adhere to the designer intention and must be coherent with the developing prototype. Too strict or not compatible requirements generate errors during the testing and validation phase, and force the engineers to fix them, manage the implementations, and test/validate again the final prototype. All together, it results in increased time of the product development cycle. In some restricted cases, formal methods have been adopted to overcome these situations. They act at the level of requirement definitions and consist in using formal languages, such as temporal logics and at the level of verification in using model checking techniques. The use of logical languages standardizes the definition of the requirements and reduces the possibility of creating incompatible requirements. Model checking techniques give the opportunity of automatically verifying the satisfiability of these requirements, by freeing the designer from the development of custom (and error-prone) verification algorithms.

Despite the undoubted usefulness of formal methods, which has been achieved and demonstrated in specific high-level industrial fields, these methods are not yet used in many other areas where they might be successfully applied. The reasons are rather complex. First of all there is a lack of knowledge in the field of formal methods (for

example many engineers do not even encounter them during their academic studies) and secondly there are many cases where standard model checking approaches are not directly applicable (e.g., they are not suitable to manage the complexity and size of the model used in industries).

From present to future. Nowadays there is a growing interest in developing connected devices which assist our life. Starting from smart homes and continuing to self-driving vehicles, the presence of intelligent systems which autonomously interact with humans and other devices is increasing. These systems are generally called *Cyber-physical systems* (CPS) [LS16], and their main feature is the integration of computation abilities within the physical world. These devices are able to collect and analyze data generated from physical sensors and to use this information to act and modify the physical world around them, or at least spreading the information to other devices. Consider a modern fly-by-wire aircraft. In this kind of airplane, the pilot's commands are mediated by a computer control which, based on the actual aircraft conditions, could avoid errors. It can prevent the aircraft to go outside the safe operating region and causing, for example, a stall. Cyber-physical systems are generally composed of multiple devices which interact with each other and/or with the physical world even in a non-linear way. The main characteristics which make the CPS complex dynamical systems are their nonlinearity and their hierarchical structure. The first implies that the understanding of the whole CPS *cannot* be derived by separately analyzing agents and combining the collected information. The second characteristic means that, for example, the outputs of some devices are inputs of other devices. In some cases, circular dependencies can even be present, i.e., (closed) feedback loops. These two characteristics and the noisy nature of sensor data are the root causes of CPS complexity. The mathematical counterpart of this is the so-called *state space explosion* which makes the analysis of CPS a challenging task of the present.

The forecast is clear: the use of CPS, which currently has an explosive growth, is going to be soon pervasive. Consider Industry 4.0, self-driving cars, medical devices, and smart grids. Every innovative technology field seems to engage CPS as the leading actor of its storyboard. The improvement process necessary to master the complexity of CPS touches all the level of MBD: requirements definition, implementation, verification and also MDO. Moreover, considering the diffuse presence of CPS devices in our daily life, their verification will assume a more and more central role.

This thesis can be collocated into the established path of research which tries to combine formal methods and machine learning techniques for the verifiability and system designs of complex systems.

Approach

Various mathematical models can be considered to describe CPS, and we can divide them into two categories: *Deterministic* and *Stochastic* models. The former include *Ordinary Differential Equations* or *Finite State Machines* which are widely used in popular modeling software such as MATLAB/Simulink or LabVIEW. The latter include *Continuous Time Markov Chain* (CTMC) and *Stochastic Hybrid Systems* (SHS) mainly used in population dynamics evolution such as biological reaction networks, epidemic mod-

eling or performance evaluation. All these models can be simulated to produce possible realizations of the system, i.e., trajectories. These trajectories define the dynamics of the system induced by specific configurations (inputs and internal states). These models can be considered in the MBD framework which we introduced before. In this thesis, we focused on the above mentioned three aspects: requirements definition, verification, and optimization.

The first ingredient to properly define the requirements are formal languages such as temporal logic. These are modal logics suitable to describe in a rigorous and concise way temporal behaviors. Among the existing temporal logics, we will focus our attention on *Signal Temporal Logic* (STL), which is a linear, continuous-time, temporal logic with future modalities, built on top of atomic predicates which are simple inequalities among variables. The reason for this choice is twofold. First it is sufficiently powerful to describe lots of phenomena, and secondly, it is easily interpretable. Moreover there exist efficient monitoring algorithms which are capable of checking if a trajectory satisfies a given STL formula or not.

The second ingredient is machine learning techniques aimed to assist the verification of CPS. Given a requirement and a model, we could in principle use the standard model checking techniques to verify if the model is compliant with that requirement. If we consider stochastic models, for example, a requirement could be related to the probability that a given temporal logic formula is satisfied. This is a standard task in stochastic modeling, and several probabilistic model checkers have been implemented. Unfortunately, these model checkers are not able to work with complex systems such as the majority of CPS. The mathematical counterpart is the well-known state space explosion, which basically means that the state space of the systems increases so much that the exact model verification requires too much computational effort and memory. *Statistical Model Checking* (SMC) has been introduced to tackle this problem and try to solve it in practice. The underlying idea is to approximate statistically the probability of satisfaction of a given formula by means of simulation. SMC effectively attacks the state space explosion but can be applied only if the model is fully specified, meaning that, for example, all the chemical rates of a modeled chemical reactions have to be known. Obviously, this hypothesis is not reasonable in many situations. For example when system design and system identification problems are consisting precisely in determining those chemical rates! To overcome this situation *smoothed model checking* (smMC) has been proposed. This technique relays on *Gaussian processes*, a regression technique belonging to Bayesian statistics, is the perfect example of combination of formal methods and machine learning.

The third ingredient is the use of optimization techniques aiming at finding the best performing design with respect to a set of objectives. Mathematically this is tackled by several multi-objective optimization approaches primarily used in many industrial fields. These approaches regroup all the numerical algorithms which are able to simultaneously optimize different objectives and to identify the so-called Pareto front (i.e., a set which informally represents the best compromise among different criteria/objectives).

Contribution

The leitmotif of this thesis is "master complexity of CPS". The idea is to use novel machine learning and optimization approaches to perform MBD of CPS in cases where the state of the art does not produce a good result.

As mentioned before, the requirement definition is a preliminary problem in MBD. We work on this subject acting in two opposite directions. In Chapter 6, we move from model/data to requirements by studying the problem of mining signal temporal logic formulae from a dataset composed of two kinds of trajectories (regular and anomalous). In machine learning, this task, known as classification problem, is a well-studied problem usually solved by means of dedicated algorithms. Efficient approaches, such as feedforward neural networks, have been implemented even for big dataset. Such systems produced black box computational models which are able to classify data with high accuracy, but paying a high price: *low interpretability*. It means that we cannot understand the reasons why a path is regular or anomalous. In a broader sense, we cannot *extract knowledge* from our dataset. Using Signal Temporal Logic (STL) is a way to tackle such a problem, because it is a formalism sufficiently powerful to classify observed trajectories, and highly interpretable (thanks to its logical nature: an STL formula is a logical combination of simple linear inequalities). We use a genetic optimization algorithm which automatically combines STL formulae by leveraging on their tree structure. In Chapter 5, we move from requirements to model/data by introducing a temporal logic called Signal Convolution Logic (SCL), a novel specification language which can express non-functional requirements in cyber-physical systems showing noisy and irregular behavior. The STL language is not able to quantify about the percentage of time certain events happen. These kinds of requirements can be usefully considered in modeling many CPS scenarios, e.g., medical devices. Consider, for instance, a medical CPS device measuring glucose level in the blood to release insulin in diabetic patients. In this scenario, we need to check if glucose level is above (or below) a given threshold for a certain amount of time, to detect critical settings. Short periods of hyperglycemia (high level of glucose) are not dangerous for the patients. An unhealthy scenario is when the patient remains in hyperglycemia for more than 3 hours during the day, i.e., for 12.5% of 24 hours. This property cannot be specified by STL.

The verification is another crucial step of MBD which is naturally linked to the requirement definition. We work on this subject by addressing synthesis and falsification problems which can be considered as two sides of the same coin. The first consists in identifying the parameters which induce the model to behave as specified by the requirements. The second problem consists in identifying the parameters which cause malfunctions of the model. The faults are expressed through logical requirements. In [BPS16] we tackle the *parameter synthesis problem* in a multi-objective paradigm. The idea consists in maximizing *simultaneously* the satisfaction probability of two (or more) logical formulae by obtaining a so-called *Pareto front*. The multi-objective approach to model checking verification has been already described in [EKVY07, CMH06, FKP12]. The most common technique consists in transforming the original problem into a linear programming problem. Our approach is different. Similarly to the industrial approach, where the multi-objective paradigm is widely used, we consider the model as a black box and solve the multi-objective problem by customizing the dominance relation of a standard genetic algorithm (NSGAII, [DAPM02]). More specifically, we implemented a

mixed approach consisting in maximizing the average robustness *and* the probability of satisfaction of two STL formulae. We show that this approach produces good results in two test cases. In Chapter 7, we deal with the parametrized verification of temporal proprieties which is an active research field of fundamental importance for the design of stochastic cyber-physical systems. In some domains (such as system and synthetic biology) has become relevant to identify the region of the parameter space where the model satisfies a linear time specification with probability greater (or less) than a given threshold. Solving this problem by employing numerical methods with guaranteed error bounds has been considered in [CDKP14, CDP+17]. This method, however, is severely affected by the state space explosion problem. We proposed an alternative solution to this problem employing machine learning and statistical approaches based on smoothed model checking. Moreover, we introduced the *Bayesian threshold synthesis problem* as the statistical version of the threshold synthesis problem proposed in [CDP+17]. The idea is to identify the region of the parameters space which induces a probability greater (or less) than a given threshold with statistical guarantees.

In [SPB17b] we implemented an active learning approach aimed at falsifying block-diagram models (i.e., Simulink/Stateflow, Scade, LabVIEW, etc.) where several switch blocks, 2/3-D look-up tables, and state transitions coexist. We cast the falsification problem into a *domain estimation problem* and used an active learning optimization approach, similar to the cross-entropy algorithm, which produces more and more counterexamples as the number of iterations increases. The inputs of these systems are mainly continuous, or *càdlàg* functions, and for this reason a parametrization is mandatory. We used an adaptive parametrization which is more expressive than the standard fixed point parametrization. Our approach is based on the active learning paradigm which consists in the simultaneous approximation and minimization of an unknown function (which in this case is the quantitative semantics of the STL requirements). We bring the approximation of the quantitative semantics at the logical level by defining a *probabilistic approximation semantics*. Briefly, we use the Gaussian processes to estimate the probability that specific sub-formulae of the original STL requirement have been falsified. Then, this information is combined by means of a logical bottom-up approach and the probability of falsifying the STL requirement is estimated. Afterward, this probability is used to sample points which eventually falsify the requirements. The combination of the active learning approach and the adaptive parametrization produces good performance regarding the minimum number of simulations to falsify the model. This performance index is rather important in the industrial world where system simulations could be costly.

Thesis structure

This thesis has been divided into two parts. The first part presents the background which is composed of 4 chapters. The second part outlines the contributions in 5 chapters.

Part I (Background). In Chapter 1, we introduce the basic definitions, the stochastic and deterministic models.

Chapter 2 describes the Signal Temporal Logic (STL) which is the main logic formalism used in the rest of the thesis. First, we introduce the Boolean and the quantitative

semantics, and then we summarize the model checking techniques focusing mainly on the statistic model checking.

In Chapter 3, we describe Gaussian processes and how this nonparametric Bayesian technique can be efficiently employed to evaluate the probability of satisfaction of STL specification of stochastic systems. This approach is known as smoothed model checking [BMS16].

In Chapter 4, we briefly introduce the mathematical techniques to solve optimization problems. We focus on Bayesian optimization and describe the evolutionary algorithms able to solve the multi-objective optimization problems.

Part II (Contributions). In Chapter 5, we propose Signal Convolution Logic (SCL), a novel temporal logic suited to express non-functional requirements. This logic uses the mathematical convolution to define a temporal operator which is able to quantify the percentage of time an event occurs. This contribution has been accepted for publication in *the 16th International Symposium on Automated Technology for Verification and Analysis (ATVA 2018)*. A preprint version of this contribution is available, see [SNBB18].

In Chapter 6, we use a genetic algorithm and the average robustness of STL formulas to create a classifier which can discriminate among anomalous and regular trajectories. This contribution has been accepted for publication in *the 15th International Conference on Quantitative Evaluation of SysTems (QEST 2018)*. See [SNBB17] for a preprint version of this work.

In Chapter 7, we combine the smoothed model checking technique with an active learning approach to identify the parameter space of a parametrized model where the satisfaction probability is above a given threshold. We rephrase this problem, known as threshold synthesis problem, in a Bayesian framework and provide an algorithm which efficiently solves it. This contribution has been published, see [BS18].

In Chapter 8, we study the design of stochastic systems in a multi-objective paradigm. The idea is to simultaneously maximize multiple conflicting requirements which are specified employing STL formulae. This contribution has been published, see [BPS16].

Finally in Chapter 9, we present a new approach to tackle the falsification of formal properties of a black box cyber-physical system. The proposed method leverages the Gaussian processes and a new parameterization technique of the input functions to reduce the minimum number of simulation leading to the model falsification. This contribution has been published, see [SPB17b].

I

Background

1

Models

1.1 Basic Concepts

In this section we introduce the basic concepts which are the ground of the most of the results in this thesis.

System. In a broader sense we intend a *system* as a collection of components. We attach the adjective *physical* if these parts are physical objects or the adjective *logical* is these constituents are concepts such as algorithms or software.

State. The state of a system is the set of variables representing quantitative proprieties of the system. Mathematically it is represented as a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}$, $n \in \mathbb{N}$, expressing the quantitative values of the target properties.

Model. A model (of a system) is the description of specific aspects of the system. It has a mathematical form which can be used to simulate the time evolution of specific quantities.

Trajectory (or signal). Informally it is the description of the time evolution of the system. Mathematically it is represented as the time evolution of its state vector, i.e., a function $\mathbf{x}\colon \mathbb{T} \to \mathcal{S}$, where the interval $\mathbb{T} \subset \mathbb{R}_0^+$ represents the time and $\mathcal{S}$ the state space. We denote the trajectory space with $\mathcal{S}^{\mathbb{T}}$.

Cyber-Physical Systems. CPS are systems composed by physical parts (mechanical parts, chemical or biological processes), computational platforms (such as sensors, actuators, embedded devices), and network components which allow communication among the computational platforms and different CPSs. The physical parts are essentially black box systems which react to the environment through physical process. The computational parts are in charge of collecting information from the physical parts by means of sensors and of elaborating this information in order to take actions.

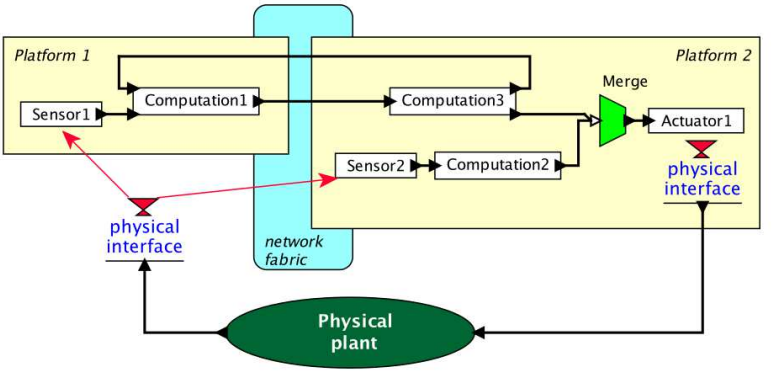


Figure 1.1: A generic CPS plant model

1.2 Continuous-time Stochastic Processes

In this section, we introduce one of the main players of this thesis: *Continuous-time Stochastic Process*. Doing it in a fully rigorous way requires too much machinery, which in the end will be distracting. We focus on the main elements we use and try to maintain a reasonable trade-off between rigorousness and descriptiveness. We refer the reader to standard textbook such as [KSK12] for a completely formal description of the elements of this section.

Let us consider a *probability space* $(\mathcal{S}, \Sigma, P)$ i.e., a tuple where $\mathcal{S}$ is a set, Σ is a σ-algebra constructed on $\mathcal{S}$, $P\colon \Sigma \to [0,1]$ is a countably additive, non-negative measure on $(\mathcal{S}, \Sigma)$ such that $P(\mathcal{S}) = 1$. We also introduce a *random variable* $X\colon \mathcal{S} \to \mathbb{R}$ as a measurable function i.e., $X^{-1}(A) = \{s \in \mathcal{S} : X(s) \in A\} \in \Sigma$ for every Borel set $A \in \mathcal{B}(\mathbb{R})$.

Definition 1 (Stochastic Process)**.** *Let us consider a probability space $(\mathcal{S}, \Sigma, P)$, a set $\mathbb{T} \subseteq \mathbb{R}^+$ and suppose that for each $t \in \mathbb{T}$ exists a random variable $X_t\colon \mathcal{S} \to \mathbb{R}$ defined on $(\mathcal{S}, \Sigma, P)$. The function $x\colon \mathbb{T} \times \mathcal{S} \to \mathbb{R}$, such that $x(t,s) = X_t(s)$, is called stochastic process indexed in $\mathbb{T}$ and usually it is denoted with $\{X_t,\ t \in \mathbb{T}\}$.*

A stochastic process is generally characterized by the structure of $\mathcal{S}$, so we have continuous, discrete, numerable, denumerable stochastic processes. It is convenient but not necessary to consider $\mathbb{T}$ as the time.

We can consider the stochastic process from two different points of view. The first reflects the Definition 1 and considers a stochastic process as a collection of labeled random variables X_t, such that fixed the label $t \in \mathbb{T}$ everything is well defined. The second considers a stochastic process as a random function $t \mapsto x(t,s)$, called *path* or *trajectory*. This point of view poses several challenges. We can easily define the probability space generated by a finite product of probability spaces, for example the probability space containing all the couple $(X_t, X_{t'})$, $t, t' \in \mathbb{T}$ is defined as the $(\mathcal{S}^2, \Sigma \otimes \Sigma, P_2)$ where $\otimes$ is the direct product and $P_2((X_t, X_{t'}) \in A_1 \times A_2) = P(X_t \in A_1) \cdot P(X_{t'} \in A_2 | X_t \in A_2)$. Problem arise when we want to extend this approach to countable or continuous set such as $\mathcal{S}^{\mathbb{T}}$.

This challenge was seriously tackled in the last century. Let us report the fundamental theorem due to Andrey Kolmogorov.

Theorem 1 (Kolmogorov Extension Theorem)**.** *Let $t_1, \dots, t_k \in \mathbb{T}$, $k \in \mathbb{N}$. If the finite dimensional probability measure $\mu_{t_1 \dots t_k}$ defined on $\mathcal{S}^k$ satisfies the following two properties:*

- *for all permutations π of $\{1, \dots, k\}$ and for each Borel set $F_i \in \mathcal{B}(\mathbb{R})$*

$$\mu_{\pi(t_1)\dots\pi(t_k)}(F_{\pi(1)} \times \cdots \times F_{\pi(k)}) = \mu_{t_1 \dots t_k}(F_1 \times \cdots \times F_k)$$

- *for all Borel set $F_i \in \mathcal{B}(\mathbb{R})$ and $m \in \mathbb{N}$*

$$\mu_{t_1 \dots t_k}(F_1 \times \cdots \times F_k) = \mu_{t_1 \dots t_k, t_{k+1}, \dots, t_{k+m}}(F_1 \times \cdots \times F_k \times \mathbb{R}^n \times \cdots \times \mathbb{R}^n)$$

therefore exists a probability space $\Omega = (\mathbb{R}^{\mathbb{T}}, \tilde{\Sigma}, \tilde{P})$ such that Σ is the product σ-algebra of $\mathbb{R}^{\mathbb{T}}$ and $\tilde{P}$ is the unique probability function such that $\tilde{P}(X_{t_1} \in F_1 \wedge \cdots \wedge X_{t_k} \in F_k) = \mu_{t_1 \dots t_k}(F_1 \times \cdots \times F_k)$.

The above theorem defines two conditions that a finite dimensional probability families $\mu_{t_1 \dots t_k}$ must satisfy over the finite dimensional space $\mathbb{R}^k$, $\forall k \leq +\infty$ in order to be lifted up to the infinite dimensional space $\mathbb{R}^{\mathbb{T}}$.

In this thesis, we mostly use a particular kind of continuous-time stochastic process called *Continuous-time Markov Chains* (CTMC). These models are used to describes a broader class of physical phenomena. The reason for this extensive applicability is the so-called *Markov property* or *memoryless property* which we introduce below and that seems to be able to capture the probabilistic nature of many physical processes.

1.2.1 Continuous-time Markov Chains

We first introduce a minimal definition (Definition 2) and express briefly the assumption and the theoretical steps leading to a detailed definition (Definition 3) of Continuous-time Markov Chains, which we use in the rest of this thesis.

Definition 2. *A time-homogeneous Continuous-time Markov Chain (CTMC) is a stochastic process $\{X(t),\, t \in \mathbb{R}_0^+\}$ with discrete or countable state space $\mathcal{S}$, such that $\forall t_0, t_1 \in \mathbb{R}^+$, $\forall s_0, s_1 \in \mathcal{S}$:*

$$\begin{aligned} P(X(t_1) = s_1 \,|\, X(t_0) = s_0, \{X(\tau) : 0 < \tau < t_0\}) \\ = P(X(t_1) = s_1 \,|\, X(t_0) = s_0) = \mathbf{P}_{s_1,s_2}(t_1 - t_0) \end{aligned} \quad (1.1)$$

where $\mathbf{P}_{s_1,s_2}(t)$ is called transition matrix. It represents the probability that the chain moves from s_0 to s_1 within $t_1 - t_0$ time units.

As a consequence of its definition, the transition matrix $\mathbf{P}$ satisfies the following two

properties:

$$\forall t \in \mathbb{R}^+, \forall s, \bar{s} \in \mathcal{S}, \quad 0 \leq \mathbf{P}_{s,\bar{s}}(t) \leq 1 \tag{1.2}$$

$$\forall t \in \mathbb{R}^+, \forall s \in \mathcal{S}, \quad \sum_{\bar{s} \in \mathcal{S}} \mathbf{P}_{s,\bar{s}}(t) = 1 \tag{1.3}$$

Property 1.2 is a consequence of the definition of probability, and Property 1.3 says merely that the chain keeps its state or jumps to another state.

Equation 1.1 (up to the first equality sign) is called *Markov property* (also known as *memoryless property*). It states that the conditional probability distribution of the future chain evolution depends only on the present state. Moreover, we are also stating that the chain's dynamics does not depend on the arrival times (t_0) but only on the sojourn time ($t_1 - t_0$). This is the *time homogeneity assumption*. The Markovian property entails the semi-group property $\mathbf{P}(s+t) = \mathbf{P}(s) \cdot \mathbf{P}(t)$, also called Chapman-Kolmogorov Equations, which brings directly to the well known backward and forward Kolmogorov equations.

Assuming that $\mathbf{P}(t)$ is right-continuous and considering[1] that $\mathbf{P}(0) = \mathbf{I}$, we define the *infinitesimal generator* of a CTMC as

$$\mathbf{Q} = \lim_{h \to 0^+} \frac{\mathbf{P}(h) - \mathbf{I}}{h} \tag{1.4}$$

Considering Properties 1.2 and 1.3, we have that

$$\sum_{s' \in \mathcal{S}} \mathbf{Q}_{s,s'} = 0\,; \quad \mathbf{Q}_{s,s} \leq 0\,; \quad \forall s \neq s',\, \mathbf{Q}_{s,s'} \geq 0\,.$$

When $s \neq s'$ the value $\mathbf{Q}_{s,s'}$ is interpreted as the *rate of moving for s to s'* and $\mathbf{Q}_{s,s} = \sum_{s' \neq s} \mathbf{Q}_{s,s'}$ as the *rate of leaving s*.

Finally, considering Equation 1.4 and $\frac{d\mathbf{P}(t)}{dt} = \lim_{h \to 0^+} \frac{\mathbf{P}(t+h) - \mathbf{P}(t)}{h}$ we obtain the famous *backword and forward Kolmogorov equations*

$$\frac{d\mathbf{P}(t)}{dt} = \mathbf{Q}\mathbf{P}(t), \quad \frac{d\mathbf{P}(t)}{dt} = \mathbf{P}(t)\mathbf{Q}$$

The solution is $\mathbf{P}(t) = e^{t\mathbf{Q}}$ where the exponential of a matrix $\mathbf{A}$ is defined as $e^{\mathbf{A}} = \sum_{k \in \mathbb{N}} \frac{\mathbf{A}^k}{k!}$.

Definition 3. *A Continuous-time Markov Chain is a pair $(\mathcal{S}, \mathbf{R})$ where $\mathcal{S}$ is a finite (or countable) set of states and $\mathbf{R}$ is the rate matrix such that the dynamics is defined by*

$$\mathbf{P}_{s,s'}(t) = \frac{\mathbf{R}_{s,s'}}{E(s)} \left(1 - e^{-E(s)t}\right), \text{ such that } E(s) = \sum_{s' \in \mathcal{S}} \mathbf{R}(s, s') \tag{1.5}$$

The characterization of the transition matrix obtained in Equation 1.5 hints that the CTMCs are separable with respect to time and state transition. The first factor

[1] The semi-group property states that $\mathbf{P}(0)^2 = \mathbf{P}(0)$ which entails that $\mathbf{P}(0)$ is the identity matrix $\mathbf{I}$.

$\mathbf{R}(s, s')/E(s)$ describes the jumping probability from s to s' which is unaffected by the sojourn time t. Whereas the factor $1-e^{-E(s)t}$, which rules the time t within a future outgoing transition happens, is fully agnostic to the ongoing destination s'. This separation is at the basis of many algorithms which simulate CTMC, and it is the reason why these models are usually covered under the umbrella of *jumping and holding stochastic process*.

Throughout this thesis we mainly consider parametric reaction networks. We stress that our approaches can be applied to every population models based on CTMCs, such as *Population Continuous-time Markov Chains* [BHLM13a] or *Markov Population Models* [HJW11].

Definition 4. *A Parametric Reaction Network*[2] *(PRN) is a tuple* $\mathcal{M}_\vartheta = (\mathcal{A}, \mathcal{S}, \mathbf{x_0}, \mathcal{R}, \vartheta)$*, where:*

- $\mathcal{A} = \{a_1, \dots, a_n\}$ *is the set of species (or agents).*
- $\mathbf{x} = (x_1, \dots, x_n)$ *is the vector of variables counting the amount of each species, with values* $\mathbf{x} \in \mathcal{S}$*, where* $\mathcal{S} \subseteq \mathbb{N}^n$ *is the state space.*
- $\mathbf{x_0} \in \mathcal{S}$ *is the initial state.*
- $\mathcal{R} = \{r_1, \dots, r_m\}$ *is the set of reactions, each of the form* $r_j = (\mathbf{v_j}, \alpha_j)$*, with* $\mathbf{v_j}$ *the stoichiometry or update vector and* $\alpha_j = \alpha_j(\mathbf{x}, \vartheta)$ *the propensity or rate function. Each reaction can be represented as*

$$r_j : u_{j,1}a_1 + \ldots + u_{j,n}a_n \xrightarrow{\alpha_j} w_{j,1}a_1 + \ldots + w_{j,n}a_n,$$

  where $u_{j,i}$ *(*$w_{j,i}$*) is the amount of elements of species* a_i *consumed (produced) by reaction* r_j*. We let* $\mathbf{u_j} = (u_{j,1}, \dots, u_{j,n})$ *(and similarly* $\mathbf{w_j}$*) and define* $\mathbf{v_j} = \mathbf{w_j} - \mathbf{u_j}$*.*
- $\vartheta = (\vartheta_1, \dots, \vartheta_k)$ *is the vector of parameters, taking values in a compact hyperrectangle* $\Theta \subset \mathbb{R}^k$*.*

A PRN $\mathcal{M}_\vartheta$ defines a Continuous-time Markov Chain [Nor98, BHLM13a] on $\mathcal{S}$, with infinitesimal generator Q, where $Q_{\mathbf{x},\mathbf{y}} = \sum_{r_j \in \mathcal{R}} \{\alpha_j(\mathbf{x}, \vartheta) \mid \mathbf{y} = \mathbf{x} + \mathbf{v_j}\}$, $\mathbf{x} \neq \mathbf{y}$.

1.2.2 Topology of trajectories

A useful definition, in the context of stochastic processes, is the following.

Definition 5 (Càdlàg function)**.** *A Càdlàg*[3] *function* $X \colon \mathbb{T} \to \mathcal{S}$ *is a right continuous function with left-hand limit*

(i) $X(\hat{t}) = \lim_{t \to \hat{t}^+} X(t)$

(ii) $\forall \hat{t} \in \mathbb{T}$, $\exists \lim_{t \to \hat{t}^-} X(t)$

Thanks to their ability to describe jumps (which is a standard feature of stochastic trajectories), càdlàg functions are largely used in the field of stochastic modeling.

[2] Also known as Parametric Chemical Reaction Network (PCRN).
[3] French: "continue à droite, limite à gauche".

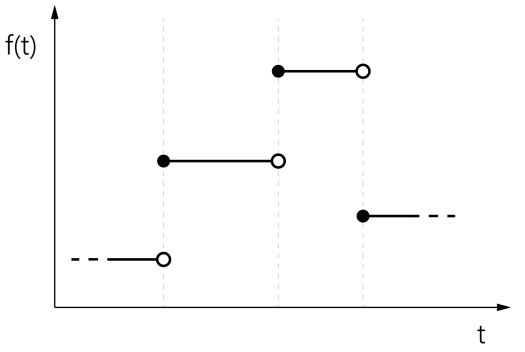


Figure 1.2: A Càdlàg function

In this domain the uniform topology $L_\infty(\mathbb{T},\mathbb{R})$, induced by the distance $d_\infty(X,Y) = \sup_{t\in[0,T]} \|X(t)-Y(t)\|_\infty$, is not satisfactory[4]. Let us consider a stochastic trace modeled by the Heaviside function H:

$$H\colon \mathbb{R} \to \{0,1\}$$
$$H(t) = \begin{cases} 0 & t < 0 \\ 1 & \text{otherwhise} \end{cases}$$

It describes a single jump from state $x = 0$ to $x = 1$ at time 0. If we consider another trace $\tilde{H}(t) \coloneqq H(t-\varepsilon)$ which represents the same jump in almost the same time $t = \varepsilon$, $\varepsilon \ll 1$, we would like to have guarantees that these two trajectories are close. This requirement is natural if we consider that those two trajectories visit the same sequence of locations at almost the same time. Unfortunately the uniform topology is unable to capture that behavior, as implied indeed by $\forall\varepsilon > 0\, \|H(x) - H(x+\varepsilon)\|_\infty = 1 > 0$.

A practical example which evidences the lackingness of the uniform distance d_∞ comes out from the runtime verification area. During monitoring of trajectories could happen that small delays affect the measurement procedure. For this reason, it is necessary to consider a measure which is not too much sensitive to such artifacts. The Skorohod topology was introduced to solve this task.

Definition 6. *(Skorohod Distance) Let $\Lambda = \mathbb{T}^\mathbb{T}$ be the set of the strictly increasing functions from $\mathbb{T}$ to $\mathbb{T}$, and consider two càdlàg functions $X, Y\colon \mathbb{T} \to \mathbb{R}$. The Skorohod distance is*

$$d_{sk}(X,Y) = \inf_{\lambda\in\Lambda} \max(\|\lambda - \mathbb{1}\|_\infty, d_\infty(X, Y\circ\lambda))$$

where $\mathbb{1}$ is the identity function, i.e., $\mathbb{1}(x) \coloneqq x$.

This distance considers simultaneously the d_∞ distance (between X and a deformation of Y, i.e., $Y\circ\lambda$) and a penalizing term which is proportional to the deformation induced by $\lambda \in \Lambda$, i.e., $\|\lambda - \mathbb{1}\|_\infty$. As a consequence of its strictly increasing trend, the λ functions preserve the original sequence of events, allowing, however, dilatations or

[4] If $\mathbf{x} = (x_1, \ldots, x_n)$ then $\|x\|_\infty = \max_{i\le n} |x_i|$.

contractions of time. Informally, the Skorohod distance permits to manage perturbation of both space and time. We always work with trace defined on a closed subinterval $\mathbb{T} \subset \mathbb{R}_0^+$ where the Skorohod distance is well defined. In other cases could be necessary to extend its definition to the whole $\mathbb{R}_0^+$. For a detailed explanation see [Bil99].

1.2.3 Simulating Continuous-time Markov Chains

There are many algorithms suitable for CTMC simulation. We briefly describe two fundamental approaches which were originally introduced by Daniel T. Gillespie.

The *Stochastic Simulation Algorithm* (SSA) [Gil77], is an exact[5] algorithm which allows to numerically simulate the time evolution of well-stirred (chemically) reacting systems. Under this "well-stirred" assumption the state of the system, which is denoted by a vector $\mathbf{X}(t) = \mathbf{x}$ and which represents the number of each reagent, is a stochastic process. Its dynamics is ruled by the *next-reaction density function*, $p(\tau, j|\mathbf{x}, t) = a_j(\mathbf{x}) \exp(-\sum_j a_j(\mathbf{x})\tau)$ which represents the probability that the next reaction R_j will occur in the infinitesimal interval of time $[t + \tau, t + \tau + dt)$. The core of the SSA algorithm is a Monte Carlo approach, called Direct Method, aimed to sample the random pairs (τ, j). The idea consists in decomposing $p(\tau, j|\mathbf{x}, t)$ as the product of two independent terms i.e. $p(\tau, j|\mathbf{x}, t) = p_1(\tau|\mathbf{x}, t)p_2(j|\tau, \mathbf{x}, t)$ such that, $p_1(\tau|\mathbf{x}, t) = \sum_j a_j(\mathbf{x}) \exp(-\sum_j a_j(\mathbf{x}))$ and $p_2(j|\tau, \mathbf{x}, t) = \frac{a_j(\mathbf{x})}{\sum_j a_j(\mathbf{x})}$. In practice the time to next reaction (τ) is sampled by the exponential distribution with decay constant $\sum_j a_j(\mathbf{x})$, whereas the next reaction is sampled by the discrete probability distribution $a_j(\mathbf{x})/\sum_j a_j(\mathbf{x})$, $j = 1, \ldots, n$. Since this approach requires the generation of two random numbers for each reaction, it can be particularly expensive.

The *τ-leaping method* (τ-leap) [Gil01] decreases the computational cost of SSA by reducing the number of simulations. The idea consists in splitting the time interval in a set of contiguous subintervals and estimating how many reactions of each kind occurs in each of them. Mathematically we can define the function $Q(k_1, \ldots, k_M|\tau; \mathbf{x}, t)$ which describes the probability that exactly k_j firing of the reaction R_j occur in $[t, t + \tau)$. We consider the random variables $K_j(\tau, \mathbf{x}, t)$ which counts how many R_j reactions occur. The assumption is that exists a $\tau \geq 0$ sufficiently small, such that the time evolution of the vector state $\mathbf{X}$ in $[t, t + \tau)$ slightly affect the propensity function, i.e., $\forall t' \in [t, t + \tau)$, $a_j(\mathbf{X}(t')) \cong a_j(\mathbf{X}(t))$. This property, known as *τ-leaping condition*, permits to approximate $K_j(\tau, \mathbf{x}, t)$ with a Poisson process variable $\mathcal{P}(a_j(\mathbf{x})\tau)$. Finally the i.i.d condition of the random variables K_j implies that $Q(k_1, \ldots, k_M|\tau; \mathbf{x}, t) = \prod_{j=1}^{M} P_{\mathcal{P}}(k_j, a_j(\mathbf{x})\tau)(t)$ which can be efficiently calculated by means of standard algorithms.

1.3 Deterministic Models

In this section, we discuss the approximation of PRNs' dynamics through the use of *Ordinary Differential Equations* (ODEs). We also introduce the *Block Diagram Models* (BDM) which allow designing interacting physical systems through suitable graphical

[5] In the sense that it is based on the same physical assumption which justifies the Chemical Master Equation

workflows. These models are essentially hybrid systems with several internal states which are associated with specific continuous dynamics described through ODEs systems. The activation of these internal states, which is ruled by a specific guard mechanism, allows the modeling of complex dynamics, which are frequent in the field of CPS. Finally, we introduce the nonautonomous dynamical systems which are practically used to model BDMs in the context of black box modeling.

1.3.1 Ordinary Differential Equations System

Let us consider the following reaction network with $a_1, \ldots, a_n$ species and $r_1, \ldots, r_j$ reactions

$$\begin{aligned} r_1 : u_{1,1}a_1 + \ldots + u_{1,n}a_n &\xrightarrow{\alpha_1} w_{1,1}a_1 + \ldots + w_{1,n}a_n \\ r_2 : u_{2,1}a_1 + \ldots + u_{2,n}a_n &\xrightarrow{\alpha_2} w_{2,1}a_1 + \ldots + w_{2,n}a_n \\ \ldots &\to \ldots \\ r_j : u_{j,1}a_1 + \ldots + u_{j,n}a_n &\xrightarrow{\alpha_j} w_{j,1}a_1 + \ldots + w_{j,n}a_n \end{aligned}$$

This reaction network can be modeled with an ODEs system as follows

$$\begin{aligned} \frac{dx_1}{dt} &= f_1(x_1, \ldots, x_j) \\ \frac{dx_2}{dt} &= f_2(x_1, \ldots, x_j) \\ &\ldots\ldots \\ \frac{dx_j}{dt} &= f_j(x_1, \ldots, x_j) \end{aligned}$$

where $x_1(t), \ldots, x_n(t)$ represent the percentage[6] of the specie $a_1, \ldots, a_n$ which populate the reaction networks at time t and the functions $f_1, \ldots, f_j$ depend on the structure and the rates of the reaction. This approach was criticized by Daniel T. Gillespie in the introduction of SSA original paper [Gil77]. Its primary concern is referred to the oversimplification of the non-deterministic and discrete nature of PRN to a continuous and deterministic process. This approximation, which is reliable if the number of molecules is sufficiently high, results from the following statement

$$\forall i \leq j, \lim_{n\to+\infty} \frac{\sum_n X_i^n(0)}{n} = x(0) \Longrightarrow \forall i \leq j, \lim_{n\to+\infty} P\left\{ \sup_{s\leq t} \left| \frac{\sum_n X_i^n(s)}{n} - x(s) \right| \geq \varepsilon \right\} = 0$$

which asserts that the averages of stochastic variables X_i evolve by following a deterministic process ruled by the above ODEs. A formal proof of this statement can be found in [Kur70] and a detailed presentation of this topic, known as *fluid approximation*, is discussed in [BHLM13a].

Let us introduce an example to whom we refer a lot in the contribution part of this thesis.

[6] It is a continuous approximation of the real percentage which assumes value in a discrete set.

Example 1. *We consider a SIRS epidemic model, introduced back in 1927 by W.O. Kermack and A.G. McKendrick [GCC00], which is still widely used to simulate the spreading of a disease among a population. The population of N individuals is typically divided into three classes:*

susceptible (S) **:** *representing healthy individuals that are vulnerable to the infection.*

infected (I) **:** *describing individuals that have been infected by the disease and are actively spreading it.*

recovered (R) **:** *modeling individuals that are immune to the disease, by having recovered from the disease (R).*

The chemical reactions are the following

$$\begin{aligned} r_1 : S + I &\xrightarrow{\alpha_1} 2I \qquad \alpha_1 = k_{si} \cdot \frac{X_s \cdot X_i}{N} \\ r_2 : I &\xrightarrow{\alpha_2} R \qquad \alpha_2 = k_{ir} \cdot X_i \end{aligned}$$

The corresponding ODE System is the following:

$$\begin{aligned} \frac{ds(t)}{dt} &= -k_{si}s(t)i(t) \\ \frac{di(t)}{dt} &= k_{si}s(t)i(t) - k_{ir}i(t) \\ \frac{dr(t)}{dt} &= k_{ir}i(t) \end{aligned} \tag{1.6}$$

1.3.2 Block Diagram Models

Many physical systems can be described employing multiple interacting ODEs systems. From a modeling point of view, it is convenient to have a tool which permits to graphically design and simulate such systems. Simulink [Mat], Modelica [Mod] and LabVIEW [Nat] are the most used software of this kind. These models, known as *Block Diagram Models* (BDM), are based on the fundamental concept of *Actor Model* [HBS73] originated in 1973. An actor is a computational entity which receives messages from other actors, use this information to change its internal state, and eventually produces outputs which are spread to other connected actors. These models can be used to describe a continuous-time system S which operates on continuous signals. The idea is to draw it as a graphical box with inputs and outputs as following

Mathematically S is a functional that depends on two parameters p and q

$$S(q,p)\colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}} \tag{1.7}$$

Multiple actor models can be collected and considered together as a unique entity, which is considered again as a *single* actor model. This is the purpose of the subsystems block which is shown in Figure 1.4, where also other common blocks are reported. This grouping property allows hierarchically treating BDMs, enhancing the possibility to manage different levels of complexity. In Figure 1.5a a block diagram model of an

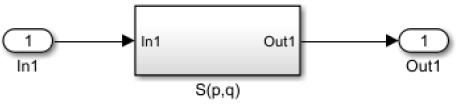


Figure 1.3: An actor model S which has an input (In1), an output (Out1) and depends on two parameters p and q.

ODE is represented. It consists of a simple feed-forward loop with an integrator and a gain block. Whereas, in Figure 1.5b we represent the SIR model (Example 1) as a combination of two feed-forward loops.

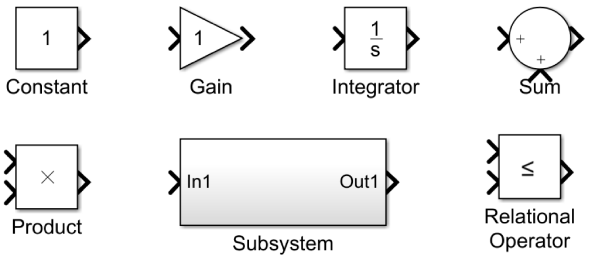


Figure 1.4: The most common operators are shown. We have the constant block which simply produces a constant output function, the gain block which multiplies by a factor the input function. Then we see the integrator, sum and product blocks. The Subsystem block

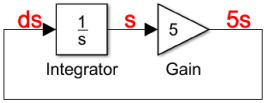
1.3.3 Black box Dynamical System

In the Introduction, we outlined the black box approach to modeling. It is a mandatory approach in the absence of a model's internal specifications and a reasonable one if these specifications are too complicated. These two cases are a standard set up in many industrial fields, which indeed use this kind of approach. The idea is to model a system as a pair $\mathcal{M} = (\mathbf{U} \times \mathbf{X}, \texttt{sim})$ where $\mathbf{U}$ and $\mathbf{X}$ are finite sets representing, respectively, the input values of the system and the internal state (coinciding for us with the output). The input set is characterized as follows:

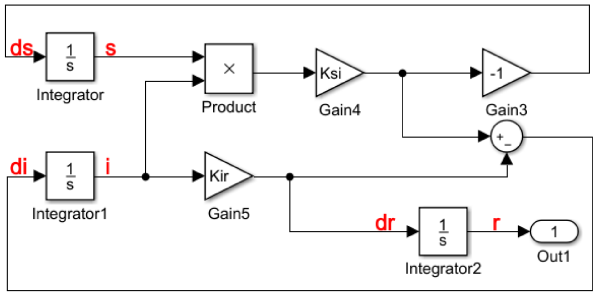
$$\mathbf{U} = \bar{U}_0 \times \cdots \times \bar{U}_n \times U_1 \times \cdots \times U_m \qquad \text{(input set)}$$

where $\bar{U}_i$ are finite (or numerable) sets and U_i are compact sets in $\mathbb{R}$, representing respectively the discrete input events and the continuous input signals. A similar characterization is valid for the internal state

$$\mathbf{X} = \bar{X}_0 \times \cdots \times \bar{X}_{n'} \times X_1 \times \cdots \times X_{m'} \qquad \text{(internal state set)}$$



(a)



(b)

Figure 1.5: **(a)** Block diagram models of $\frac{ds}{dt} = 5s$. **(b)** A block diagram model which represents the SIR model (Equation 1.6)

We call state of the system at time t, the couple $(\mathbf{u}(t), \mathbf{x}(t)) \in \mathbf{U} \times \mathbf{X}$

Example 2. *Let us consider a model of a car which takes as input any driver's actions. There are continuous actions, such as throttle and brake pedal dynamics, and discrete actions, such as gear switching or activation/deactivation of ABS controller. The outputs are also continuous quantities, such as RPM, car speed, and emission or discrete quantities, such as gear dynamics (in case of an automatic gear system), etc.*

$\mathcal{M}$ is equipped with a simulator, `sim`, which we assume to be a black box (i.e., we can only provide any inputs and read the generated outputs). It requires a state at time t (i.e., $(\mathbf{u}(t), \mathbf{x}(t))$ and a rational time step (i.e., h) as input and provides a new state at time $t + h$ as output:

$$\mathbf{x}(t + h) = \mathtt{sim}(\mathbf{u}(t), \mathbf{x}(t), h) \tag{1.8}$$

The above equation describes the time evolution of a nonautonomous discrete-time dynamical system which produces traces of the following kind

$$(\mathbf{u}(t_0), \mathbf{x}(t_0), t_0), ((\mathbf{u}(t_1), \mathbf{x}(t_1), t_1)), \ldots, (\mathbf{u}(t_n), \mathbf{x}(t_n), t_n)$$

It is convenient to describes each trace in a continuous fashion, by means of two functions: the *internal state function* $\mathbf{x} : \mathbb{T} \to \mathbf{X}$ and the *input function* $\mathbf{u} : \mathbb{T} \to \mathbf{U}$ defined as follows

$$\mathbf{x}(t) = \mathbf{x}(t'), \mathbf{u}(t) = \mathbf{u}(t') \quad \text{s.t.} \quad t' = \max_{k \leq n} \{t_k \mid t_k \leq t\}$$

2

Temporal Logic

In the first part of this chapter, we introduce the temporal logics, mainly focusing on *Signal Temporal Logic* (STL), largely used to reason about the future evolution of a path in continuous time. In the second part, we describe the monitoring and the model checking techniques available.

2.1 Temporal Logic

Temporal logic [Pnu77] is a formalism tailored to reason about events in time. It extends the propositional or predictive logic by modalities (temporal operators) that permit to manage time. These modalities are built on the couple $(\mathbb{T}, <)$, where $\mathbb{T}$ is the usual time interval and $<$ is the *accessibility relation* on $\mathbb{T}$. These formalisms permit a rigorous description of events in time, therefore are used to define temporal proprieties of dynamical systems. There are different kinds of temporal logics which are based on different assumptions about the nature of time. It can be *linear*, if there is only a single timeline, or *branched*, if there exist multiple timelines (i.e., branches which can be represented as a tree). Moreover, we can consider time as a discrete, numerable or continuous set. We are mainly interested in a linear continuous-time temporal logic. Linear, because we deal with simulation trajectories and continuous-time because the systems, that we describe, have continuous dynamics.

2.1.1 Signal Temporal Logic

STL [MN04] has been introduced in 2004 with the purpose of defining properties of continuous-time real-valued signals which can also be useful for the runtime verification of dynamical systems' trajectories. In that period many discussions were related to the verification of continuous hybrid systems and how the exhaustive and precise verification of these systems was not decidable (except for trivial cases). STL extends $\text{MITL}_{[a,b]}$, a fragment of the *Metric Interval Temporal Logic* (MITL) [AFH96] which uses bounded temporal operators. Basically, STL redefines the atomic predicates of MITL into atomic propositions interpreted over real-values signals.

We call *primary signal* a trajectory as defined in Section 1.1, it merely describes the quantitative evolution of the state vector. We now recall the definition of Boolean signals, which are abstraction of primary signals, introduced to represent the qualitative time evolution of specific properties. Finally, we present the syntax and semantics of STL.

Definition 7. *Let us consider a primary signal* $\mathbf{x}\colon \mathbb{T} \to \mathcal{S}$ *and the set of atomic predicates* $\Pi = \{f(\mathbf{s}) \geq 0 \,|\, f \in \mathcal{C}(\mathbb{R}^m, \mathbb{R}), \mathbf{s} \in \mathcal{S}\}$ *where* $\mathcal{C}(\mathbb{R}^m, \mathbb{R})$ *is the set of continuous function from* $\mathbb{R}^m$ *to* $\mathbb{R}$*. We call Boolean signal the function* $\gamma\colon \mathbb{T} \to \mathbb{B} = \{\bot, \top\}$ *defined as* $\gamma(t) = \pi(\mathbf{x}(t)) = (f \circ \mathbf{x})(t) > 0$*. Moreover, we call secondary signal the function* $f \circ \mathbf{x}\colon \mathbb{T} \to \mathbb{R}$*.*

The secondary signal represents a quantity that we would like to monitor. It is a natural concept by a modeling and monitoring point of view and can be computed with physical filters (i.e., specific devices which monitor trajectories), or computational algorithms usually called monitors.

Definition 8 (Syntax)**.** *The formulae of STL (denoted with* $\mathcal{L}_{STL}$*) are defined by the following syntax:*

$$\varphi := \bot \,|\, \top \,|\, \mu \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \varphi \mathbf{U}_{[T_1,T_2]}\varphi, \tag{2.1}$$

where $\bot$ *and* $\top$ *denote logical false and true,* $\neg$ *and* $\vee$ *are the usual logical connectives, the atomic predicates* $\mu \in \Pi$ *have been defined before,* $[T_1, T_2] \subset \mathbb{R}_0^+$ *and* $\mathbf{U}_{[T_1,T_2]}$ *is a temporal operator called* Until*. For simplicity, two other temporal operators* Finally *and* Globally *can be defined as customary as* $\mathbf{F}_{[T_1,T_2]}\varphi \equiv \top\mathbf{U}_{[T_1,T_2]}\varphi$ *and* $\mathbf{G}_{[T_1,T_2]}\varphi \equiv \neg\mathbf{F}_{[T_1,T_2]}\neg\varphi$*, respectively.*

The formulae of $\mathcal{L}_{\text{STL}}$ are interpreted over trajectories respect to the following *Boolean semantics* [MN04].

Definition 9 (Boolean Semantics)**.** *Given a trajectory* $\mathbf{x} \in \mathcal{S}^{\mathbb{T}}$*, the Boolean semantics* $\models \subset (\mathcal{S}^{\mathbb{T}} \times \mathbb{R}_0^+) \times \mathcal{L}_{STL}$ *is defined recursively by:*

$$\begin{aligned}
&(\mathbf{x}, t) \models \mu && \iff \mu(\mathbf{x}(t)) = \top \\
&(\mathbf{x}, t) \models \neg\varphi && \iff (\mathbf{x}, t) \not\models \varphi \\
&(\mathbf{x}, t) \models \varphi_1 \vee \varphi_2 && \iff (\mathbf{x}, t) \models \varphi_1 \ \texttt{or}\ (\mathbf{x}, t) \models \varphi_2 \\
&(\mathbf{x}, t) \models \varphi_1 \mathbf{U}_{[T_1,T_2]}\varphi_2 && \iff \texttt{exists}\ t' \in [t + T_1, t + T_2]\ \texttt{such that}\ (\mathbf{x}, t') \models \varphi_2 \\
& && \quad \texttt{and for each}\ t'' \in [t, t')\ (\mathbf{x}, t'') \models \varphi_1
\end{aligned}$$

where for simplicity we write $(\mathbf{x}, t) \models \mu$ *instead of* $((\mathbf{x}, t), \mu) \in \models$*.*

We usually consider $t = 0$ and write $\mathbf{x} \models \varphi$ to mean $(\mathbf{x}, 0) \models \varphi$.

Remark 1 (Temporal horizon)**.** *In Section 1.1 we introduced the time domain of a signal, denoted with* $\mathbb{T}$*, as a closed subset of* $\mathbb{R}^+$*. The motivation is purely practical and reflects the real case where the simulation or monitoring of trajectories is always limited to a given closed temporal interval. The time interval* $\mathbb{T}$ *has to be considered when working with STL formulae. Indeed, if it is not sufficiently large, the satisfiability*

of those formulae cannot be determined. The concept of time horizon T_φ has been introduced to overcome this issue [MN04]. The idea is to estimate, for each formula φ, the smallest interval $[t, t+T_\varphi]$ which is sufficiently large to permit the evaluation of $(\mathbf{x}, t) \models \varphi$. T_φ is defined by the following simple rules: $T_\perp = 0$, $T_\mu = 0$, $T_{\varphi_1 \vee \varphi_2} = \max(T_{\varphi_1}, T_{\varphi_2})$, $T_{\varphi_1 \mathbf{U}_{[T_1,T_2]} \varphi_2} = \max(T_{\varphi_1}, T_{\varphi_2}) + T_2$. For simplicity, we denote $[t, t+T_\varphi]$ with $\mathbb{T}_\varphi$ (consider that t is always fixed in advanced and we remark that in the rest of the thesis it is assumed to be 0).

Remark 2 (Why the Boolean semantics is not enough)**.** *The Boolean semantics as defined above, is simply the semantics of MITL adapted to STL. Considering that STL works on trajectories, which have continuous dynamics, the simple Boolean dynamics seems to be not adequate. As a simple example, let us consider the atomic predicate $X \leq c$ and two trajectories $\mathbf{x}_1$ and $\mathbf{x}_2$, such that $\mathbf{x}_1(0) = c + \varepsilon$ and $\mathbf{x}_2(0) \gg c$. The Boolean semantics assigns a violation of the atomic predicate to both the trajectories, without considering that in the second case this violation is more evident. Quantitative semantics have been introduced to overcome this limitation. Their benefit consists in enriching the expressiveness of Boolean semantics, passing from a Boolean concept of satisfaction (yes/no) to a (continuous) degree of satisfaction. This permits us to quantify "how much", concerning a specific criterion, a trajectory satisfies (or not) a given requirement.*

Let us introduce the *robustness semantics* of STL, a quantitative semantics which appears in [DM10] and consists on a reformulation of the quantitative semantics originally defined by Pappas and Fainekos in [FP09].

Definition 10 (Robustness Semantics)**.** *Given a trajectory $\mathbf{x} \in \mathcal{S}^{\mathbb{T}}$, the robustness semantics is a function $\varrho \colon \mathcal{S}^{\mathbb{T}} \times \mathbb{R}_0^+ \times \mathcal{L}_{STL} \to \mathbb{R} \cup \{-\infty, +\infty\}$ defined recursively as follows:*

$$
\begin{aligned}
&\varrho(\mathbf{x}, t, \top) && = +\infty \\
&\varrho(\mathbf{x}, t, \mu) && = (f \circ \mathbf{x})(t) \\
&\varrho(\mathbf{x}, t, \neg\varphi) && = -\varrho(\mathbf{x}, t, \varphi) \\
&\varrho(\mathbf{x}, t, \varphi_1 \vee \varphi_2) && = \max(\varrho(\mathbf{x}, t, \varphi_1), \varrho(\mathbf{x}, t, \varphi_2)) \\
&\varrho(\mathbf{x}, t, \varphi_1 \mathbf{U}_{[T_1,T_2]} \varphi_2) && = \sup_{t' \in [t+T_1, t+T_2]} (\min(\varrho(\mathbf{x}, t, \varphi_2), \inf_{t'' \in [t,t')} \varrho(\mathbf{x}, t, \varphi_1)))
\end{aligned}
$$

where $f \circ \mathbf{x}$ is a secondary signal as defined in Definition 7.

This quantitative semantics satisfies the following important property

Definition 11 (Soundness Property)**.** *A quantitative semantics (ϱ) is sound with the Boolean semantics ($\models$), if and only if*

$$
\begin{aligned}
\varrho(\mathbf{x}, t, \varphi) > 0 &\implies (\mathbf{x}, t) \models \varphi \\
\varrho(\mathbf{x}, t, \varphi) < 0 &\implies (\mathbf{x}, t) \models \neg\varphi
\end{aligned}
$$

The case $\varrho(\mathbf{s}, t, \varphi) = 0$ is not specified because is not possible to establish if $(\mathbf{s}, t) \models \varphi$ or $(\mathbf{s}, t) \models \neg\varphi$ (consider that $\varrho(\mathbf{s}, t, \varphi) = 0$ and $\varrho(\mathbf{s}, t, \neg\varphi) = 0$). The name robustness is related to the meaning of its absolute value. It is the maximum point perturbation

allowed for the secondary signals, composing a formula, so that the truth-value of the whole formula does not change. This remarkable property is known as *Correctness Property* which is reported below.

Definition 12. *Consider an STL formula φ which is composed by a logical combination of atomic predicates $(\mu_1, \dots, \mu_n)$, $n < +\infty$ such that $\mu_i := [f_i(X) \geq 0]$, and two signals $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}^{\mathbb{T}}$. We write*

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_\varphi := \max_{i \leq n} \sup_{t \in \mathbb{T}_\varphi} |f_i(\mathbf{x}_1(t)) - f_i(\mathbf{x}_2(t))|$$

where $\mathbb{T}_\varphi$ is the horizon interval as defined in Remark 1.

Definition 13 (Correctness Property)**.** *The quantitative semantics ϱ satisfies the correctness property with respect to the formula φ, if:*

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}^{\mathbb{T}},\ \|\mathbf{x}_1 - \mathbf{x}_2\|_\varphi < \varrho(\mathbf{s}_1, t, \varphi) \ \ \textit{and} \ \ (\mathbf{x}_1, t) \models \varphi$$

$$\textit{implies} \ \ (\mathbf{x}_2, t) \models \varphi$$

If the previous property is true for all the formulae of the language we say that the quantitative semantics satisfies the correctness property for the entire language.

2.2 Usefulness of Quantitative Semantics

Quantitative semantics was initially introduced to overcome the limitation of Boolean semantics, which sometimes is not able to fully capture properties of specific systems. This is the case of systems equipped with a specific topology, such as dynamical systems where the $\|\cdot\|_\infty$ distance, for example, is used to define similarity among trajectories. A simple example is discussed in Remark 2, where the inability of Boolean semantics to discriminate between a strong and weak violation of a specific predicate has been highlighted.

Quantitative semantics have been introduced to overcome this limitation by enriching the expressiveness of Boolean semantics, passing from a Boolean concept of satisfaction (yes/no) to a (continuous) degree of satisfaction. This permits to quantify "how much", concerning a specific criterion, a trajectory satisfies (or not) a given requirement. The continuous value provided by quantitative semantics can be efficiently used in formal modeling of complex systems because of two properties this semantics satisfies: soundness (Definition 11) and correctness (Definition 13).

Consider for example a family of deterministic models $\mathcal{M}_\vartheta$ which depends on parameters $\vartheta \in \Theta$. A typical design problem consists in identifying the best parameters so that a given requirement $\varphi \in \mathcal{L}_{\text{STL}}$ is satisfied. Considering, for example, the robustness semantics of Definition 10 which satisfies soundness and the correctness, this task can be achieved through its maximization. In fact, if the value of the robustness is strictly positive then the property is satisfied. Moreover, maximizing the robustness value is preferable because of the correctness property which equates *high robustness* with *low sensitivity to perturbation*. During the recent years different quantitative semantics were introduced, mainly to describe various properties of trajectories,

[DM10, RBNG17, AH15]. In the related work of Chapter 5, we discuss some of these quantitative semantics.

2.3 Monitoring

A monitor is an algorithm aimed to establish if a trajectory $\mathbf{x}$ satisfies or not a temporal formula φ, i.e., $\mathbf{x} \models \varphi$. *Offline monitors* relate on *a posteriori* analysis performed on complete simulation traces (i.e., traces where time horizon have been reached). In this case, computational efficiency is the most addressed feature as results from the contribution of many researchers [Don10, DFM13]. *Online monitors*, on the contrary, have been developed to work with real-time systems. In this scenario, the responsiveness and the quantity of stored data at each t required to perform the monitoring, are the main feature to take into account. Consider a real-time system in charge of performing actions (such as notification deliveries) if a specific event occurs (such as malicious detections). In this context, the responsiveness is a crucial aspect as well as the buffer memory used by each of these monitors (consider that many of them run on embedded deceives). Another important feature of online monitors is the ability to work with partial trajectories, meaning that in each instant of time t, if the satisfiability of a formula cannot be addressed at least an estimate should be produced.

Donze et al. [DFM13] present an offline algorithm for monitoring STL formulae over piecewise continuous signals, i.e., linear interpolation of traces, which are sequences of couple time values, usually obtained by numerical integration or stochastic simulation. The results of the monitoring is also a piecewise continuous function entailing the possibility of a recursive approach (which is indeed used).

Approach Idea. This algorithm follows a bottom-up approach, evaluating the quantitative or Boolean semantics of atomic proposition and lifting up the results up to the root of the tree structure which represents the formula. A sketch of the algorithm is shown in Algorithm 1. It is a recursive procedure based on four different inputs: true symbol, atomic (line 1 - 4), unary (line 5 - 7) and binary (line 8 - 11) operators or connectives (represented as a generic operator $\circ$). The COMPUTE routine implements a specific strategy for each logical symbol passed as input (similarly the COMPUTEATOMIC routine at line 4). It is evident that Algorithm 1 is a simple implementation of the definition of the Boolean and quantitative semantics.

For the details of the offline Boolean algorithm we refer the reader to [MN04] and for the offline quantitative version to [DFM13].

2.4 Statistical Model Checking

Stochastic processes are largely used to model phenomena showing intrinsically random effects such as biological processes. Model checking algorithms have been developed for verifying probabilistic properties of stochastic system, which are usually described employing specific temporal logics such as PCTL [HJ94] or CSL [BHHK03]. These logics express simple inequalities on probability of target properties (e.g., the model

Algorithm 1 MONITOR$(\varphi, \mathbf{x})$

case $\varphi = \top$
  return $y = \bar{\top}$ (a constant signal true)
case $\varphi = y(\mathbf{x}(t)) \geq 0$
  return COMPUTEATOMIC$(y(\mathbf{x}(t)))$
case $\varphi = \circ\varphi_1$
  $y =$ MONITOR$(\varphi_1, \mathbf{x})$
  return COMPUTE $(\circ, y)$
case $\varphi = \varphi_1 \circ \varphi_2$
  $y_1 =$ MONITOR$(\varphi_1, \mathbf{x})$
  $y_2 =$ MONITOR$(\varphi_2, \mathbf{x})$
  return COMPUTE $(\circ, y_1, y_2)$

$\mathcal{M}_\vartheta$ satisfies the property φ with a probability greater than (or lower than) a given threshold k, i.e., $\mathcal{M}_\vartheta \models (P(\varphi) \sim k)$, $\sim \in \{\geq, \leq\}$. For simplicity[1] we write $P_\varphi(\vartheta)$ to mean the probability that a trajectory generated by $\mathcal{M}_\vartheta$ satisfies φ. The necessary theoretical steps to define $P_\varphi(\vartheta)$ are not trivial.

Provided that the trajectories of the interested model $\mathcal{M}_\vartheta$ is a probability space, e.g. $\Omega_\vartheta = (\mathcal{S}^{\mathbb{T}}, \Sigma, \mathcal{P}_\vartheta)$ it is important to verify that for each logical formula φ, the set $Sat(\varphi) = \{\mathbf{x} \in \mathcal{S}^{\mathbb{T}} | \mathbf{x} \models \varphi\}$ belongs to the sigma algebra Σ of that probability space. Otherwise the following probability is meaningless (or at least not formally definable)

$$P_\varphi(\vartheta) = \mathcal{P}_\vartheta(Sat(\varphi)) \tag{2.2}$$

The Kolmogorov Extension theorem (Theorem 1) implies that the trajectories of a CTMC is a probability space, moreover in this case the above quantity (Equation 2.2) is well defined as discussed in [BHHK03, Var85]. Their argumentation follows from the basic cylindric decomposition of CTMC trajectories.

Many of the algorithm for estimating $P_\varphi(\vartheta)$ rely on analytical techniques which compute this probability using numerical integration of Kolmogorov equations and then compare the result with the reference threshold. Unfortunately, these approaches suffer from the well-known curse of *state space explosion*, which makes these approaches not always feasible. Examples are CTMCs with many states and transitions.

Remark 3. *Let us briefly describe the numerical approach of [CDKM11] in order to understand why sometimes this approach is not feasible. First, we know that STL considers only time bounded formulae, meaning that for each φ exists a finite interval $\mathbb{T}_\varphi$ (see Remark 1) such that it is sufficient to consider $\{\mathbf{x}(t) \,|\, t \in \mathbb{T}_\varphi\}$ to determine if $\mathbf{x} \models \varphi$. The key idea is to notice that $Sat(\varphi) = Sat_{\leq N}(\varphi) \cup Sat_{>N}(\varphi)$, where $Sat_{\leq N}(\varphi)$ collects all the trajectories which make less than N jumps in $\mathbb{T}_\varphi$. Vice-versa, $Sat_{>N}(\varphi)$ collects all the trajectories which make more than N jumps. Considering the structure of CTMC it is easy to shows that for each $T > 0$ and $\varepsilon > 0$ exists an N such that $\mathcal{P}_\vartheta(Sat_{>N}(\varphi)) < \varepsilon$. Therefore, we can approximate $\mathcal{P}_\vartheta(Sat(\varphi))$ by estimating $\mathcal{P}_\vartheta(Sat_{\leq N}(\varphi))$. The idea is to decompose $Sat_{\leq N}$ so to create a partition which we denote with $\overline{Sat}_{\leq N}(\varphi)$. Each class $[x_1 \dots x_m]$ of this partition represents trajectories*

[1] We write $P_\varphi(\mathcal{M})$ if we are referring to a specific model $\mathcal{M}$.

$\{(x_i,t_i)\}_{i\in\mathbb{N}}$ *such that* $\forall i \in \mathbb{N}$, $t_i < t_{i+1}$, $t_m \leq \max(\mathbb{T}_\varphi)$ *and* $t_{m+1} > \max(\mathbb{T}_\varphi)$ *which are different for the order of visited states. The next step is to filter out all the classes which do not contain trajectories which satisfy* φ. *For example, if* $\varphi = \mathbf{F}_{[0,T]} f(s) > 0$ *the algorithm will discard all the classes* $[x_1 \dots x_m]$ *such that* $\forall i \leq N$, $f(x_i) \leq 0$, *because independently on* $t_1 t_2 \dots t_N$ *we have that* $((x_1,t_1),(x_2,t_2),\dots,(x_N,t_N)) \not\models \varphi$. *The remaining classes,* $\overline{Sat}_{\leq N}(\varphi)$, *contain only trajectories which have at least a combination of times* $(\{t_i\}_{i\leq m})$ *such that* $\{(x_i,t_i)\}_{i\leq m} \models \varphi$, *and* $m < N$. *Finally for each class of* $\overline{Sat}_{\leq N}(\varphi)$ *we can estimate* $\bar{S}([x_1 \dots x_m]) \subseteq \mathbb{T}^m$ *which contains all the combination of times* $(\{t_i\}_{i\leq m})$ *such that* $\{(x_i,t_i)\}_{i\leq m} \models \varphi$. *Finally we can approximate* $\mathcal{P}_\vartheta(Sat(\varphi))$ *as follows:*

$$\mathcal{P}_\vartheta(Sat(\varphi)) \approx \sum_{[x_1\dots x_m]\in\overline{Sat}_{\leq N}(\varphi)} \prod_{i=1}^{m-1} \mathbf{R}_{x_i,x_{i+1}} \int\cdots\int_{\tau\in\bar{S}([x_1\dots x_m])} e^{-\mathbf{E}\cdot\tau} \tag{2.3}$$

where $\mathbf{E} = (E(x_1),\dots,E(x_m))$ *and* $\tau = (t_1,\dots,t_m)$.

The evaluation of the above formula 2.3 can be computational expensive. In order to evaluate the multidimensional integral, first a decomposition of $\bar{S}([x_1 \dots x_m])$ *should be provided. In [CDKM11] for example, a polyhedral decomposition is considered and the integral is evaluated in each of these polyhedra. This decomposition and numerical integration might be performed many times, considering that usually* $|\overline{Sat}_{\leq N}(\varphi)|$ *is a high value.*

Statistical Model Checking (SMC) [YS06, ZPC13] have been introduced to solve these issues. It relies on simulations which are faster than the numerical, symbolic algorithm used to achieve a precise evaluation of probability. Furthermore, it is a general approach which does not depend on the stochastic model class. For this reason, it is more suitable for an industrial application.

We now describe the Bayesian Statistical approach appears in [ZPC13]. For a frequentist approach, we refer the reader to [YS06].

Bayesian SMC. There are two approaches: the *Bayesian Interval Estimation* (BIE), which estimates a confidence interval for the target probability p (i.e., the interval that contains p with at least a given confidence probability) and the *Bayesian Hypothesis Testing*, which determines if p satisfies a given inequality within a given confidence probability. We describe the first approach. Let us consider a stochastic model, such as CTMC, and the Bernoulli random variable X_φ (equal to 1 if a simulation trace satisfies φ, 0 otherwise).The goal of BIE is to estimate the probability p that X_φ is equal to 1 (i.e., $p = P(X_\varphi = 1)$). This probability is well defined as already discussed in Section 1.2. The Bayesian approach assumes that p satisfies a prior probability distribution, which reflects our previous experiences and beliefs about the model. In this case a reasonable choice is the distribution $Beta(\alpha,\beta)$ which is defined in $[0,1]$ and depends on two shape parameters $\alpha,\beta > 0$. By varying the shape parameters, it is possible to approximate different unimodal density distributions such as the uniform distribution in $[0,1]$ ($\alpha = \beta = 1$) which is a reasonable choice with lack of prior information.

The cumulative distribution function $F_{(\alpha,\beta)}$ of the beta distribution is defined for

each $u \in [0,1]$ as:

$$F_{(\alpha,\beta)}(u) = \frac{1}{B(\alpha,\beta)} \int_0^u t^{\alpha-1}(1-t)^{\beta-1}dt$$

where the beta function, $B(\alpha,\beta)$, is defined as:

$$B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$$

The beta distribution is the conjugate prior probability distribution for the Bernoulli, binomial, negative binomial and geometric distributions. That property, generally called *conjugate property*, is appreciated in Bayesian statistics because it permits to avoid numerical integration as described below.

Let us consider n simulations and $h \leq n$ successes (i.e., $h = \#(X^i_\varphi = 1)$), if we consider a beta prior distribution $Beta(p \mid \alpha,\beta)$ for the probability of $p = P(X_\varphi = 1)$, we easily obtain that the posterior probability of the Bernoulli i.i.d samples $\{X^i_\varphi\}_{i\in\mathbb{N}}$ is:

$$P(p \mid (h,n)) = \frac{P((h,n) \mid p)Beta(p \mid \alpha,\beta)}{Z(h,n)} = Beta(\alpha+h, \beta+n-h) \tag{2.4}$$

where $P((h,n) \mid p)$ is a Bernoulli distribution and $Z(h,n) = \int_0^1 P((h,n) \mid p)Beta(p \mid \alpha,\beta)dp$ is the normalizing factor. Equation (2.4) is a consequence of the aforementioned conjugate property. The Bayesian approach works as follows. First a given confidence probability c (typically $c = 0.99$ or 0.95) and an admissible error $\delta \in (0, \frac{1}{2})$ are defined. At each simulation the number of successes is stored and the expectation (i.e., $\mu = \frac{h+\alpha}{n+\alpha+\beta}$) of the posterior probability is calculated. At this point the following probability

$$P\big(p \in (\mu-\delta, \mu+\delta)\big) = F_{(\alpha+h,\beta+n-h)}\left(\mu+\frac{\delta}{2}\right) - F_{(\alpha+h,\beta+n-h)}\left(\mu-\frac{\delta}{2}\right)$$

is evaluated and compared against the threshold c. Afterward, if this probability is higher than c then the confidence probability has been reached, and the algorithm stops; otherwise, another simulation is performed, and the algorithm continues.

2.5 Statistics of Robustness

The probability of satisfaction for temporal logic (Equation 2.2) is an essential concept in the field of stochastic system modeling. However, similarly to the robustness, which has been induced to overcome the limitations of Boolean semantics, we can consider the probability distribution of robustness values [BBNS15].

Let us consider a set of models $\mathcal{M}_\vartheta$, $\vartheta \in \Theta$, and the probability space $\Omega_\vartheta = (\mathcal{S}^{\mathbb{T}}, \Sigma, \mathcal{P}_\vartheta)$ generated by their trajectories. We introduce the following set

$$Sat(\varphi \mid a,b) = \{\mathbf{x} \in \mathcal{S}^{\mathbb{T}} | \varrho(\varphi,\mathbf{x},0) \in [a,b]\}$$

and the real-valued random variable $R_\varphi(\vartheta)$ with probability distribution

$$P(R_\varphi(\vartheta) \in [a,b]) = \mathcal{P}_\vartheta(Sat(\varphi \mid a,b)) \tag{2.5}$$

The well-foundedness of 2.5 obtained through the measurability of $Sat(\varphi \mid a,b)$, i.e., $Sat(\varphi; a,b) \in \Sigma$, is demonstrated in [BBNS15].

Useful statistics can be extracted from this probability distribution. A standard statistics is the average robustness which was successfully applied in system design of and biological processes such as Incoherent type 1 feed-forward loops and Schlögl system [BBNS15]. In Chapter 8, [BPS16], we also use the average robustness to solve a multi-objective design problem.

3

Statistical Modeling

3.1 Gaussian Processes

Gaussian Processes (GP) [RW06] are non parametric Bayesian techniques used for classification or regression. Thanks to their Bayesian nature GPs are particularly suitable to manage noise and encompass statistical beliefs about the considered dataset. We now introduce the GP approach to regression problem. For the classification problem we refer the reader to [RW06].

Regression Task. We call *training set* a set of input-outputs pairs $\{(\mathbf{x}_i, y_i)\}_{i \leq d}$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$. We denote with $X = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$ and $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_d)$, respectively, the input and the output variables of the training set.

A regression problem consists in identifying a mathematical function $f \colon \mathbb{R}^n \to \mathbb{R}$ such that the predicted values $f(\mathbf{x}_i)$ are close as much as possible to the real values y_i. The quality of this prediction is evaluated through a cost (or loss) function which quantifies the distance between the predicted and the real values. Many regression techniques, such as linear and nonlinear least-squares approaches, try to minimize the outcome of a proper loss function.

GPs use a different approach which relies on the probability framework. They combine a prior distribution over a set of possible regression functions with the information provided by the training set to choose a proper posterior distribution which is suitable for estimation purposes.

Definition 14. *Formally, a GP is a collection of random variables $f(\mathbf{x}) \in \mathbb{R}$ indexed by an input variable $\mathbf{x} \in \mathbb{R}^n$ such that every finite number of them have a joint Gaussian distribution.*

A GP f is uniquely determined by its mean and covariance function (called also kernel) denoted respectively with $m : \mathbb{R}^n \to \mathbb{R}$ and $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$

such that for every finite set of points $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$:

$$f \sim \mathcal{GP}(m, k) \iff (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)) \sim \mathcal{N}(\mathbf{m}, K) \tag{3.1}$$

where $\mathbf{m} = (m(\mathbf{x}_1), m(\mathbf{x}_2), \ldots, m(\mathbf{x}_d))$ is the vector mean and $K \in \mathbb{R}^{n \times n}$ is the covariance matrix, such that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. A standard assumption is to consider (prior) Gaussian processes with a zero mean function. This is not a limitation because the mean of the posterior distribution can be different from zero (see Equation 3.7).

A popular choice for the covariance function is the *squared exponential covariance function*

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{\lambda^2}\right) \tag{3.2}$$

where *length-scale*, λ, is related to the minimum distance of the input space necessary to significantly change the value of the Gaussian process. The *signal variance* σ^2 represents the overall variance of the process. The reason for its importance is the following property

Definition 15 (Universal Property)**.** *Let f be a continuous function defined in a compact domain of $\mathbb{R}^n$. For every $\varepsilon > 0$ exists a sample function g from a GP with exponential kernel such that*

$$\|f - g\|_2 \leq \varepsilon$$

where $\|\cdot\|_2$ is the L_2 norm.

The previous property is very important because makes the GPs a suitable prior distribution over continuous functions.

Learning Phase. The GP regression consists in modeling the relation between input $\mathbf{x}_i$ and output y_i as noisy observations by setting $y_i = f(\mathbf{x}_i) + \varepsilon_i$ where the latent function f is a Gaussian Process and the noises ε_i are sampled from a normal distribution, i.e., $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. The inference problem consists in identifying the best hyper-parameters (i.e., k_f) composing the covariance function of f, and the parameter of noise distribution σ_n, such that the model $f(\mathbf{x}) + \varepsilon$ approximates the training set at the best. This problem is solved in the probabilistic framework through the maximization of the *marginal likelihood* defined as

$$p(\mathbf{y}|X) = \int \underbrace{p(\mathbf{y}|\mathbf{f}, X)}_{likelihood} \underbrace{p(\mathbf{f}|X)}_{\text{prior}} d\mathbf{f} \tag{3.3}$$

where $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$.

Considering that the prior $p(\mathbf{f}|X)$ and the likelihood $p(\mathbf{y}|\mathbf{f}, X)$ are Gaussian distributed, we obtain that the marginal likelihood $p(\mathbf{y}|X)$ is *also* Gaussian distributed. It is convenient to introduce the log marginal likelihood (which is maximized in place of the marginal likelihood)

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n I)^{-1}\mathbf{y} - \frac{1}{2}|K + \sigma_n I| - \frac{n}{2}\log 2\pi \tag{3.4}$$

where K is the $m \times m$ matrix, such that $K_{ij} = k_f(\mathbf{x}_i, \mathbf{x}_j)$, and $|K + \sigma_n I|$ is the deter-

minant of $K + \sigma_n I$.

Posterior Prediction. Let us consider a GP f trained on a set $(X, \mathbf{y})$. We are naturally interested in computing the distribution of f on new points $X^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*)$. The steps are the following:

1. From the GPs definition (Equation 3.1) we infer the conjugate prior

$$p(\mathbf{f}^*, \mathbf{f} \mid X^*, X) = \mathcal{N}(0, K(X^*, X))$$

where $\mathbf{f}^* = (f(\mathbf{x}_1^*), \ldots, f(\mathbf{x}_m^*)))$ and $K(X, X)$ is a $(m+d) \times (m+d)$ matrix of the covariance evaluated at all the pairs of the test and the training set points.

2. The conjugate prior is combined with the likelihood and the following joint posterior distribution is obtained.

$$p(\mathbf{f}^*, \mathbf{f} \mid \mathbf{y}, X^*, X) = \frac{1}{Z} p(\mathbf{f}^*, \mathbf{f} \mid X^*, X) p(\mathbf{y}|\mathbf{f}, X) \tag{3.5}$$

where $Z = \int p(\mathbf{f}^*, \mathbf{f} \mid X^*, X) p(\mathbf{y}|\mathbf{f}, X) d\mathbf{f}$ is the normalization factor.

3. Finally integrating over $\mathbf{f}$ from Equation 3.5, we obtain

$$p(\mathbf{f}^*|\mathbf{y}, X^*, X) = \frac{1}{Z} \int p(\mathbf{f}^*, \mathbf{f} \mid X^*, X)\, p(\mathbf{y}|\mathbf{f}, X) d\mathbf{f} \tag{3.6}$$

Equation 3.6 can be analytically computed and the result is the following Gaussian distribution

$$\begin{aligned} p(\mathbf{f}^*|\mathbf{y}) &= \mathcal{N}(\bar{\mathbf{f}}^*, cov(\mathbf{f}^*)) \\ \bar{\mathbf{f}}^* &= K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y} \\ cov(\mathbf{f}^*) &= K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X^*, X) \end{aligned} \tag{3.7}$$

3.2 Smoothed Model Checking

Statistical Model Checking (SMC), as described in Section 2.4, can be applied only if the model is fully specified (i.e., the vector of parameters is fixed). In many domains, this limitation is too strict. Consider for example System Biology where in general many model parameters are not directly measurable, and a continuous set of possible values (i.e., an interval) is considered. Applying SMC, in this case, is not possible because an infinite (or very large) number of SMC routine has to be performed (i.e., one for each parameter value of the interval or in a discretized grid).

Smoothed Model Checking (smMC) [BMS16] is a novel technique which leverages the continuity of the satisfaction function (w.r.t the model parameters) and the regression ability of the Gaussian processes to estimate the probability of satisfaction $P_\varphi(\vartheta)$ of a specific property φ. It cast this task into a learning problem taking as input the truth value of φ for few simulations, $(m << +\infty)$, at different parameter vectors $\vartheta_1, \ldots, \vartheta_n$.

As already discuss in Section 2.4, the satisfaction of the formula φ over a trajectory generated by the models $\mathcal{M}_\vartheta$ is a Bernoulli random variable with success probability $P_\varphi(\vartheta)$. Unfortunately, this function has a bounded codomain, and therefore it can not be modeled through Gaussian processes. Smoothed Model Checking solves this issue by considering the *inverse probit transformation* $\Psi\colon [0,1]^\Theta \to \mathbb{R}^\Theta$ defined as follows

$$\Psi(P_\varphi) \equiv f \iff \forall \vartheta \in \Theta,\ P_\varphi(\vartheta) = \int_{-\infty}^{f(\vartheta)} \mathcal{N}(0,1) d\vartheta$$

where $\mathcal{N}$ is the standard Gaussian distribution. The function $g_\varphi(\vartheta) = \Psi(f_\varphi(\vartheta))$ is a smooth real valued function of the model parameters and for this reason it can be modeled with Gaussian processes.

Let us denote with $\mathcal{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_n]$ the matrix whose rows $\mathbf{o}_i$ are the Boolean m-vectors of the evaluations in ϑ_j. Hence, we have that each observation $\mathbf{o}_i$ is an independent draw from a $Binomial(m, P_\varphi(\vartheta_j)))$. Smoothed Model Checking plugs these observations into a Bayesian inference scheme, assuming a prior $p(f)$ for the latent variable f. As f is a random function, one can take as a prior a GP, specifying its mean and kernel function, and then invoke Bayes theorem to compute the joint posterior distribution of f at a prediction point ϑ^* and at the observation points $\vartheta_1, \ldots, \vartheta_n$ as

$$p(f(\vartheta^*), f(\vartheta_1), \ldots, f(\vartheta_n) \mid \mathbf{o}) = \frac{1}{Z} p(f(\vartheta^*), f(\vartheta_1), \ldots, f(\vartheta_n)) \prod_{j=1}^{n} p(\mathbf{o}_j \mid f(\vartheta_j)).$$

In the previous expression, on the right hand side, Z is a normalisation constant, while $p(f(\vartheta^*), f(\vartheta_1), \ldots, f(\vartheta_n))$ is the prior, which is Gaussian distribution with mean and covariance matrix computed according to the GP. $p(\mathbf{o}_j \mid f(\vartheta_j))$, instead, is the noise model, which in our case is given by a Binomial density. By integrating out the value of the latent function at observations points in the previous expression, one gets the predictive distribution

$$p(f(\vartheta^*) \mid \mathcal{O}) = \int \prod_{j=1}^{n} d(f(\vartheta_j)) p(f(\vartheta^*), f(\vartheta_1), \ldots, f(\vartheta_n) \mid \mathcal{O}).$$

The presence of a Binomial observation model makes this integral analytically intractable, and forces us to resort to an efficient variational approximation known as Expectation Propagation [BMS16, RW06]. The result is a Gaussian form for the predictive distribution for $p(f(\vartheta^*) \mid \mathcal{O})$, 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, which makes assumptions on the smoothness and density of the functions that can be sampled by the GP. The squared exponential covariance function (Equation 3.2) is dense in the space of continuous functions over a compact set [Ste01], hence it can approximate arbitrarily well the satisfaction probability function. By setting its lengthscale via marginal likelihood optimization, we are picking the best prior for the observed data.

4

Optimization

In this chapter we introduce the mathematical optimization, giving particular emphasis to black box optimization. In many cases solving optimization problems with few evaluations is a strict requirement. Therefore, we focus on active learning approaches which aim to reduce the number of evaluations needed to solve optimization tasks. In particular, we introduce the *Gaussian process upper confidence bound algorithm* which is often used in black box optimization. Finally, we introduce the multi-objective optimization paradigm consisting in the simultaneous minimization of two or more objective functions. This approach is used in many engineering fields, and it can be successfully applied in the model-based design of cyber-physical systems.

4.1 Bayesian Optimization

Definition 16. *The general single-objective constrained optimization problem is defined as follows*

$$f_{gl} = \min_{\mathbf{x}\in\mathbb{D}} f(\mathbf{x})$$
$$\mathbb{D} = \left(\bigcap_{i=1}^{m'} \{\mathbf{x} \in \mathbb{R}^n \,|\, c_i(\mathbf{x}) = 0\}\right) \bigcap \left(\bigcap_{j=m'+1}^{m} \{\mathbf{x} \in \mathbb{R}^n \,|\, c_j(\mathbf{x}) \geq 0\}\right)$$

where the objective function $f\colon \mathbb{R}^n \to \mathbb{R}$, $n < \infty$ *has to be minimized and the* decision space $\mathbb{D}$*, where* f *takes values, has been defined by* m' equality constraints *(i.e.,* $c_i(\mathbf{x}) = 0$, $i \in \{1, \ldots, m'\}$*) and* $m - m'$ inequality constraints *(i.e.,* $c_i(\mathbf{x}) \geq 0$, $i \in \{m'+1, \ldots, m\}$*) and* $0 \leq m' < m < +\infty$*.*

$f_{gl} = f(\mathbf{x}^*_{gl})$ and $\mathbf{x}^*_{gl}$ are called, respectively, *global minimum* and *global minimum point*.

A point, $\mathbf{x}_{loc}$, is called *local minimum point* if

$$f(\mathbf{x}_{loc}) = \min_{\mathbf{x}\in B_\delta(\mathbf{x}_{loc})} f(\mathbf{x})$$

Properties of $f(x)$	Properties of $c_i(x)$
Function of a single variable	No constraints
Linear function	Simple Bounds
Sum of squares of linear functions	Linear functions
Quadratic function	Sparse linear functions
Sum of squares of nonlinear functions	Smooth non linear functions
Smooth non linear functions	Sparse non linear functions
Sparse nonlinear functions	Non-smooth non linear functions
Non-smooth non linear functions	

Table 4.1: A standard classification schema of the optimization problems. This table originally appears in Section 1.2 of [GMW81].

where $B_\delta(\mathbf{x}_{loc})$ is the neighborhood of $\mathbf{x}_{loc}$ of radius $\delta > 0$. $f(\mathbf{x}_{loc})$ is called local minimum.

Remark 4. *In Definition 16 we equate optimization problems to minimization problems. Maximization problems can also be defined but considering that*

$$\max_{\mathbf{x}\in D} f(\mathbf{x}) = -\min_{\mathbf{x}\in D} -f(\mathbf{x})$$

we can always cast them into minimization problems.

The optimization problems have been classified into several groups with the idea of collecting problems which might be solved with a similar approach. These approaches are tailored to the specific characteristic of the group and for this reason are particularly efficient. The classification schema was so created balancing the improvement in efficiency provided by grouping similar problems against the increase in number of approaches necessary to solve each kind of problem. The characteristics considered in this classification span from the nature of variables involved (i.e., discrete, continuous), the mathematical property of the objective function f (such as continuity, derivability, etc.) and of the constraint function used to characterize the decision space $\mathbb{D}$. A standard classification is reported in Table 4.1.

Clearly other characteristics can be considered and specific algorithms can be implemented. For a detailed and comprehensive description of the optimization problems we refer the reader to [GMW81, Fle13].

In the Introduction, we explain how the increase in models' complexity contributes to change the system developing cycle, and how the demand for new approaches related to machine learning increases as well. The optimization theory and methods have followed a similar path. With the increase in complexity of models, become very difficult to establish which is the right optimization algorithm to use. As a consequence, general approaches that do not require that the optimization problem satisfies specific properties, have been created. These approaches are suited to solve the so-called *black box optimization problems*, where no hypothesis about the target function f is required. This function can be evaluated in any points of the optimization domain $\mathbb{D}$, but other information, such as the mathematical form, gradient, or trends are not provided. This minimal assumption makes this kind of optimization problem applicable across many

domains, where prior information on target functions is not known. In many practical cases, the evaluation of the objective functions is also expensive. Consider, for example, the case of real experiments, physical simulations, or intensive computational models where a single execution could require several hours. For this reason, a significant challenge is to solve the optimization problem with few evaluations. *Bayesian Optimization* (BO) approaches are particularly suited for that task. These methods incorporate a natural trade-off between exploration and exploitation. The former is the ability of sampling points from the optimization domain in a uniform setting, and the latter is the ability to identify the promising area of the domain space where to concentrate the sampling.

A typical Bayesian optimization approach can be divided into the following three phases

Initial phase: an initial set of points is considered for evaluation. The purpose is to collect a starting knowledge of the target function f to drive the further optimization algorithm. If prior knowledge of f is available, it can be considered, and a coherent sampling strategy adopted. On the other hand, if no prior knowledge is available, a uniform sampling strategy[1], such as *Latin Hypercube Sampling* (LHS) [MBC79], *Uniform Random Sampling* or *Full Factorial*, is considered. All the nominated techniques are usually grouped under the umbrella of the *Design of Experiments* (DOE) [Mon17].

Regression phase: a Bayesian regression technique (such as GPs) is used to derive a mathematical representation of the underlined function f. This surrogate model is trained on a training set generated during the initial phase.

Active learning phase: the surrogate model defined in the previous step is used to sample a point $\bar{\mathbf{x}}$ which is the best candidate (up to the actual knowledge) to be the global minimum. Afterwards, the function $\bar{f} = f(\bar{\mathbf{x}})$ is evaluated and the couple $(\bar{\mathbf{x}}, \bar{f})$ is added to the training set of the surrogate model. The regression and active learning phase are then iterated until a convergence criterion is reached.

4.1.1 Gaussian Processes Upper Confidence Bounds

Gaussian Processes Upper Confidence Bound (GP-UCB) is an interesting Bayesian optimization approach particularly useful to minimize noisy function. Originally it was proposed to solve the multi-armed bandit problem, a well-known problem of probability theory, where a gambler has to decide which slot machines, in which order, and how many times to play. The task of the gambler is the maximization of the reward produced randomly by each machine. Mathematically this problem consists in maximizing an unknown reward function, $f\colon \mathbb{R}^n \to \mathbb{R}$, by choosing, at each i-th round, a proper input point $\mathbf{x}_i$. The objective is to maximize $\sum_{i=1}^{m} f(\mathbf{x}_i)$ which is essentially the same of discovering $\mathbf{x}^* = \arg\max_{x \in \mathbb{D}} f(\mathbf{x})$ as soon as possible. Let us consider the minimization version of the GP-UCB Algorithm.

The GP-UCB algorithm (Algorithm 2) takes as input an initial set of already evaluated points (i.e., `trainingSet`) and the Gaussian process (trained on the `trainingSet`)

[1] which is the most rationale option with this premise.

Algorithm 2 GP-UCB algorithm

Require: $\mathbb{D}$ (input space), $\mu_0 = 0$ (mean function), k (covariance function), `trainingSet`

1: **for** $i = 1, 2, \ldots$ **do**
2: Solve $\mathbf{x}_i = \arg\min_{\mathbf{x}\in\mathbb{D}} \mu_{i-1}(\mathbf{x}) - \sqrt{\beta_i}\,\sigma_{i-1}(\mathbf{x})$
3: $y_i = f(\mathbf{x}_i) + \varepsilon_i$
4: `trainingSet` $\leftarrow$ `trainingSet` $\cup\, (\mathbf{x}_i, y_i)$
5: $(\mu_i, \sigma_i) \leftarrow$ GP(`trainingSet`)
6: **end for**

represented by μ_0, k. The active learning phase is then performed (line 2) through the minimization of the objective function $obj(x) = \mu_{i-1}(\mathbf{x}) - \sqrt{\beta_i}\,\sigma_{i-1}(\mathbf{x})$. This function is a natural trade-off between exploitation and exploration. The former is related to $\mu(\mathbf{x})$ which describes the mean of the posterior Gaussian distribution in $\mathbf{x}$ (a lower value entails and higher probability that $\mathbf{x}$ is the global minimum). The latter is ruled by the $\sigma(\mathbf{x})$ which describes the uncertainty of the Gaussian posterior distribution. The balance between exploration and exploitation is ruled by the β_i parameter, which is increased accordingly to the performance of the optimization algorithm at the i-th iteration. So if it gets stuck near a point $\bar{\mathbf{x}}$ the values of β_i can be increased to improve the exploration performance. After a prefix number of β_i adjustments, if no optimal point (different from $\bar{\mathbf{x}}$) is discovered, then $\bar{\mathbf{x}}$ is considered the global minimum.

4.2 Multi-objective Optimization

Until now, we have considered only single optimization problems. However many decision, planning, and design problems consider multiple conflicting objectives or criteria. In engineering design, for example, the cost is usually minimized whereas other quality measures (such as strength, safety, comfort) which typically conflict with the cost, are maximized. Such problems are known as *Multi-objective Optimization Problems* and consist in finding the so-called Pareto Front.

A *Multi-objective Optimization Problem* (MOP) can be formulated as follows:

$$\begin{aligned} &\text{minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ &\text{s.t } \mathbf{x} \in \mathbb{D} \end{aligned} \tag{4.1}$$

where $\mathbf{x} \in \mathbb{D} \subset \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^m$, $n, m < +\infty$ is a vectorial function, and $\mathbb{D}$ is the decision space (Definition 16).

The solution of the MOP 4.1 is simply the minimal set of the following partial order relation,

Definition 17 (Pareto Dominance Relation)**.** *Let us consider two vectors $u, v \in \mathbb{R}^m$. We say that u dominates v (or equivalently that v is dominated by u), $v \preceq u$, if and only if*

$$\forall i \leq m,\ v_i \leq u_i \text{ and exists } j \leq m,\ u_j \neq v_j$$

We can now define the Pareto optimal set as follows

Definition 18. *The Pareto optimal Set (PS) is the subset of the decision space* $\mathbb{D}$ *defined as*

$$PS = \{\mathbf{x} \in \mathbb{D} \mid \nexists \mathbf{y} \in \mathbb{D}, \mathbf{f}(\mathbf{y}) \preceq \mathbf{f}(\mathbf{x})\}$$

and the Pareto Front (PF) is the subset of the codomain of $\mathbf{f}$*, defined as*

$$PF = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in PS\}$$

Informally, the Pareto optimal set is the best compromise that can be obtained by maximizing or minimizing a set of functions on the same domain set. A point belongs to it if an improvement of an objective function causes another objective function to deteriorate (i.e., decreases its value).

4.2.1 Algorithms and Approaches

Many approaches have been developed to solve MOP, we refer the reader to the following surveys [ZQL+11, MA04] and books [Mie99, Deb01]. In this section, we describe two approaches: the *wighted sum* and *non-dominated sorting genetic algorithm* which is one of the most used Evolutionary Algorithm (EA).

Weighted Sum Approach (WSA). We present this approach for three reasons. First, it is still used to solve simple MOP, second, it is the simplest approach that might come to mind, and third its main weakness justify the existence of *more advanced* multi-objective approaches. The idea of WSA is straightforward and consists in considering a unique function by a linear combination of the objective functions (i.e., $F(x) = \sum_{j=1}^{n} \alpha_j f_j(x)$, where $\forall i \leq n$, $\alpha_i \geq 0$, and $\sum_{i=1}^{n} \alpha_i = 1$). Fixed α, the minimization of the referred function means to discover a single point of the Pareto front. As a consequence, multiple single objective problems have to be solved to estimate the whole Pareto front. The main weakness of WSA is its inability to approximate non-convex Pareto front.

Evolutionary Algorithms (EA). Evolutionary Algorithms are population-based meta-heuristic algorithms inspired by the natural evolution of species, where the genetic heritage of populations improves during time evolution. Genetic Algorithms (GA) are particular kind of EA, largely used in engineering fields, where each design $\mathbf{x} = (x_1, \ldots, x_n)$ represents a *chromosome*, formed by *genes* (x_i, $i \leq n$). Chromosomes are modified by two genetic operators: *crossover* and *mutation*. The first is a binary operator which gets two chromosomes and generates a new one by mixing their genes. The second is a unary operator which acts on a single chromosome by randomly muting its genes (see Figure 4.2). Mathematically crossover and mutation are related, respectively, to exploitation and exploration.

Genetic algorithms execute a predefined number of cycles (or iterations), usually called generations, where a fixed number of designs (for example N) are evaluated and sorted accordingly to a specific ranking function. Afterward, the genetic operators have performed on these designs accordingly to their ranking score (i.e., designs with a higher rank are considered with a higher frequency) and a new generation of design is generated. The algorithm then iterates until a threshold number of generations is reached.

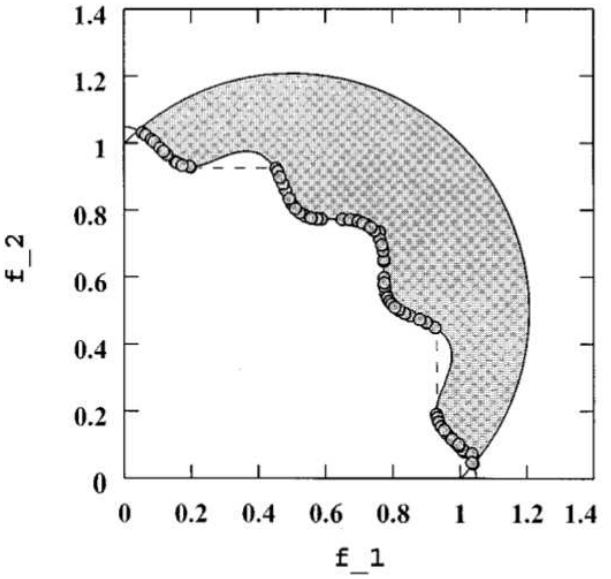


Figure 4.1: This is an approximation of the Pareto front of a standard bi-objective optimization test case called TNK, which has been obtained employing NSGA-II (original Figure in [DAPM02]). In this case, the WSA approach will be unable to identify points in the middle of the Pareto front and will identify only points around $(0, 1)$ and $(1, 0)$.

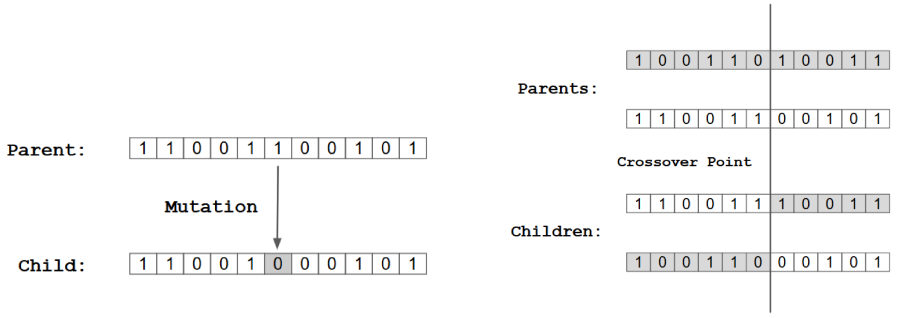
Non-dominated Sorted Genetic Algorithm (NSGA-II) is a well known GA introduced in 2000 by K. Deb [DAPM02]. It is an improvement of NSGA ([SD94]) algorithm obtained by the implements of three techniques: *non-dominated sorting*, *crowding distance* and *elitism* .

Non-dominated sorting is a ranking function based on the Pareto dominance relation (Definition 17). This ranking function splits a finite set of designs in chunks: the first chunk includes the non-dominated points, the second considers the points dominated only by the first chunks, and so on. The points contained in each chunk, which by definition cannot be ordered by the Pareto dominance relation, are sorted by using the crowding distance.

The crowding distance is a metric used to maximize the uniform displacement of designs towards the Pareto front. Consider a set of designs of an n-dimensional MOP (i.e., a MOP with n objective functions). The $2n$ extreme points $\hat{\mathbf{x}}_i^- = \min_{\mathbf{x}\in D} f_i(\mathbf{x})$, $\hat{\mathbf{x}}_i^+ = \max_{\mathbf{x}\in D} f_i(\mathbf{x})$, $i \leq n$ have a fixed crowding distance of $+\infty$ (i.e., $c(\hat{\mathbf{x}}_i^-) = c(\hat{\mathbf{x}}_i^+) = +\infty, \forall i \leq n$). For any other point $\mathbf{x}$, and for each objective f_i, the adjacent points $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ to $\mathbf{x}$ are defined as follows

$$\mathbf{x}_i^+ = \underset{\mathbf{x}'\in D\cap\{x' \mid f_i(\mathbf{x}')>f_i(\mathbf{x})\}}{\arg\min} f_i(\mathbf{x}')$$

$$\mathbf{x}_i^- = \underset{\mathbf{x}'\in D\cap\{x' \mid f_i(\mathbf{x}')<f_i(\mathbf{x})\}}{\arg\max} f_i(\mathbf{x}')$$



(a) Mutation Operator (b) Crossover Operator

Figure 4.2: Genetic operators

the crowding distance is then defined as

$$c(\mathbf{x}) = \sum_{i \leq n} \frac{f_i(\mathbf{x}_i^+) - f_i(\mathbf{x}_i^-)}{f_i(\hat{\mathbf{x}}_i^+) - f_i(\hat{\mathbf{x}}_i^-)}$$

The above distance assigns a higher value to designs which are isolated in the objective space (i.e., points with distant adjacent points). As a consequence, sampling points with high crowding distance means maximizing the displacement of these points toward the Pareto front.

The elitism approach [Rud98] is a technique introduced to enforce the convergence of the algorithm towards the Pareto Front. The genetic operators might generate suboptimal designs (i.e., designs dominated by their parents). If this possibility is not considered, then optimal designs can be discarded in favor of suboptimal designs. The elitism approach avoids this possibility by defining a new generation as the best designs of the union between the parents and the offspring generations.

II

Contributions

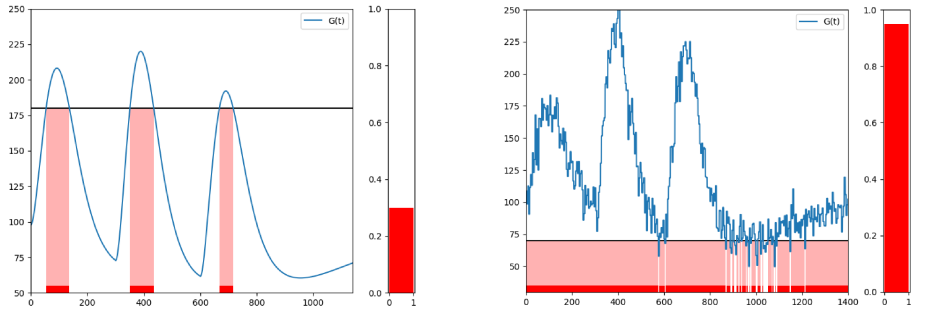
5

Signal Convolution Logic

In this chapter, we introduce *Signal Convolution Logic* (SCL), a logical framework which is able to express non-functional requirements. The fundamental idea of this logic is to consider a modal operator, $\langle k_T, p \rangle \varphi$, that we call *the convolution operator*, which depends on a non-linear kernel function k_T, and asserts that the convolution between the kernel and the signal (i.e., the satisfaction of φ) is above a given threshold p. This operator allows us to specify queries about the fraction of time a specific property is satisfied, possibly weighting unevenly the satisfaction in a given time interval $T = [T_0, T_1]$, $T_0, T_1 \in \mathbb{R}_0^+$, e.g., allowing to distinguish traces that satisfy a property more at the beginning or the end of T. We equip the logic with a Boolean semantics, and then define a quantitative semantics, proving its soundness and correctness concerning the former. Similarly to *Signal Temporal Logic* (STL), our definition of quantitative semantics permits to quantify the maximum allowed uniform translation of the signals, preserving the true value of the formula. We also show that SCL is an extension of the fragment of STL restricted to finally and globally modalities. We then discuss monitoring algorithms for both semantics, in particular presenting an efficient monitoring algorithm for the Boolean semantics and an arbitrary kernel function. We finally show the logic at work to monitor an artificial pancreas device releasing insulin in patients affected by type-I diabetes.

Motivation. Despite STL is a powerful specification language, it does not come without limitations. An essential type of properties that STL cannot express is the non-functional requirements related to the percentage of time-specific events happen. The globally and finally operators of STL can only check if a condition is true for all time instants or in at least one time instant, respectively. There are many real situations where these conditions are too strict, where we could be interesting to describe a property that is in the middle between eventually and globally. We now give three examples where SCL can be useful.

- **Time duration.** Consider for instance a medical cyber-physical system, e.g., a device measuring glucose level in the blood to release insulin in diabetic patients. In this scenario, we need to check if glucose level is above (or below) a given



(a) (b)

Figure 5.1: **(a)** A graphical representation of the property $\varphi : \mathbf{G}(t) \geq 180$ for at least 12.5% in [0,24h], meaning that the concentration of glucose has to be greater than 180 for at least 3h in 24h. **(b)** A graphical representation of the property $\psi : \mathbf{G}(t) > 70$ for at least 95% in [0,24h]. **(both)** The bars on the right represent the percentage of time the property is satisfied. At the bottom of each graph we highlight (dark red) the time where the atomic predicates are true.

threshold for a certain amount of time, to detect critical settings. Short periods of hyperglycemia (high level of glucose) are not dangerous for the patients. An unhealthy scenario is when the patient remains in hyperglycemia for more than 3 hours during the day, i.e., for 12.5% of 24 hours (see Figure 5.1a). STL cannot specify this property.

- **Noisy measurement.** A second issue is that often such measurements are noisy, and measurement errors or tiny random fluctuations due to environmental factors can easily violate (or induce the satisfaction) of a property. One way to approach this problem is to filter the signal to reduce the impact of noise, in a way that is consistent with the property at hand. This requires a signal pre-processing phase, which may, however, alter the signal introducing spurious behaviors. Another possibility instead is to ask that the property is true for at least 95% of operating time, rather than for 100% of the time, this requirement can be seen as a relaxed globally condition (see Figure 5.1b).

- **Inhomogeneous time.** Finally, there are situations when it is difficult to precisely specify the time intervals of temporal operators. As an example, while measuring the glucose level in blood, it is more dangerous if the glucose level is high just before a meal. To capture this requirement, one could give different weight to the satisfiability of the formula at the beginning or the end or in the middle of a time interval, i.e., to consider the inhomogeneous temporal satisfaction of a formula. This is also not possible in STL.

Related Work. In this paragraph, we discuss different extensions of STL, in particular analyzing different notions of robustness. In Section 2.2 we already described the quantitative semantics and its benefits from a general point of view. We also introduced a quantitative semantics called robustness semantics [FP09, DM10]. Other quantitative semantics are discussed below. In [DM10] the authors consider both the spatial robustness and also the displacement of a signal in the time domain (temporal robustness). These semantics, since are related to the L_∞ topology, are very sensitive to glitches (i.e., sporadic peaks in the signals due to measurement errors). This limitation affects the use of this semantics in the model predictive control of noisy systems such as robot planning and autonomous driving. In these contexts Probabilistic Signal Temporal Logic (PrSTL) [SK16] and Chance-Constrained Signal Temporal Logic (C2CTL) [JRSS18] have been proposed.

Rodionova et al. [RBNG17] proposed a quantitative semantics based on filtering. More specifically they provide a quantitative semantics for the *positive normal form* fragment of STL which measures the number of times a formula it is satisfied within an interval associating to it different types of kernels. One limitation of this approach is that, by restricting the quantitative semantics to the positive normal form they needed to give up the duality property between the eventually and the globally operators, which instead we can keep in our approach.

In [AH15], Akazaki et al. have extended the syntax of STL by introducing averaged temporal operators which enable the possibility to consider time not homogeneously. Their quantitative semantics expresses the preference that a specific requirement occurs as earlier as possible in a given range of time. However, this time inhomogeneity can be evaluated only for the quantitative semantics. The new operators, at the Boolean level, are equal to the classic until (since) operators. Furthermore, they can consider only this specific type of time inhomogeneities (early as possible, for as long as possible) and a combination of them with classic temporal STL operators.

Note that defining a new quantitative semantics has an intrinsic limitation. Even if the robustness can help the system design or the falsification process by guiding the underline optimization, it cannot be used at a syntax level. It means that we cannot write logical formulae which predicate about the property. For example, even if we introduce a quantitative semantics which measures how many times a property is satisfied, as in [RBNG17], we cannot write a formula which quantifies it, e.g., *the propriety has to be satisfied in at least the 50% of interval T*, but we can only measure the percentage of time the property has been verified.

5.1 Signal Convolution Logic

In this section we present the syntax and semantics of *Signal Convolution Logic* (SCL), also discussing its soundness, correctness, and finally commenting on expressiveness of the logic.

Let us briefly introduce the notions needed later in this chapter: kernel and convolution.

Definition 19 (Bounded Kernel)**.** *Let $T \subset \mathbb{R}$ be a closed interval. We call bounded*

kernel	expression
constant ($\texttt{flat}(x)$)	$\frac{\texttt{1}(x)}{(T_1-T_0)}$
linear ($\texttt{lin}(x)$)	$\frac{x-T_0}{T_1-T_0}$
exponential ($\texttt{exp}[\alpha](x)$)	$\frac{\exp(\alpha x)}{\int_T \exp(\alpha\tau)d\tau}$
Gaussian ($\texttt{gauss}[\mu,\sigma](x)$)	$\frac{\exp((x-\mu)^2)/\sigma^2)}{\int_T \exp((\tau-\mu)^2)/\sigma^2)d\tau}$

Table 5.1: Different kind of kernels.

kernel a function $k_T\colon \mathbb{R} \to \mathbb{R}$ *such that:*

$$\forall t \in T,\ k_T(t) > 0 \tag{5.1a}$$

$$\int_T k_T(\tau)d\tau = 1 \tag{5.1b}$$

Several examples of kernels are shown in Table 5.1. We call T the time window of the bounded kernel k_T, which will be used as a convolution[1] operator, defined as:

$$(k_T * f)(t) = \int_{t+T_0}^{t+T_1} k_T(\tau - t)f(\tau)d\tau$$

We also write $k_T(t) * f(t)$ in place of $(k_T * f)(t)$.

In the rest of the chapter, we assume that the function f is always a Boolean function: $f\colon \mathbb{R} \to \{0,1\}$. This implies that $\forall t \in \mathbb{R}$, $(k_T * f)(t) \in [0,1]$. Independently of the kernel used, the extreme values 0 or 1 will be assumed by the convolution only when f is always true in $t+T$ (i.e., $\forall t' \in t+T$, $f(t') = 1$), or always false (i.e., $\forall t' \in t+T$, $f(t') = 0$), respectively. On the contrary, if the Boolean function f assumes different values in $t+T$, the convolution kernel will assume a value in $(0,1)$. This value can be interpreted as a sort of *measure* of how long the function f is true in $t+T$. In fact, the kernel induces a measure on the time line, giving different importance of the time instants contained in its time window T. As an example, suppose we are interested in designing a system so to make an output signal f as true as possible in a time window T. Using a non-constant kernel k_T for this purpose will put more effort in making f true in the temporal regions of T where the value of the kernel k_T is higher. More formally, the analytical interpretation of the convolution is simply the expectation value of f in a specific interval $t+T$ w.r.t. the measure $k_T(dx)$ induced by the kernel. In Figure 5.2 (a) we show some examples of different convolution operators on the same signal.

[1] This operation is in fact a cross-correlation, but here we use the same convention of the deep learning community (see [GBCB16]) and call it convolution. By a practical point of view the cross-correlation corresponds to the convolution without flipping the kernel.

5.1.1 Syntax and Semantics

The atomic predicates of SCL are inequalities on a set of real-valued variables, i.e. of the form $\mu(\mathbf{s}) \coloneqq [g(\mathbf{s}) \geq 0]$, where $g \colon \mathcal{S} \to \mathbb{R}$ is a continuous function, $\mathbf{s} \in \mathcal{S}$ and consequently $\mu : \mathcal{S} \to \{\top, \bot\}$. The formulae $\mathcal{L}_{SCL}$ of SCL are defined by the following grammar

$$\varphi \coloneqq \bot \mid \top \mid \mu \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle k_T, p\rangle\varphi, \tag{5.2}$$

where μ are atomic predicates as defined above, k_T is a bounded kernel and $p \in [0, 1]$. SCL introduces a novel modal operator $\langle k_T, p\rangle\varphi$ (in fact, a family of them) defined parametrically w.r.t. a kernel k_T and a threshold p. This operator specifies the probability of φ being true in T, computed w.r.t. the probability measure $k_T(ds)$ of T. The choice of different types of kernel will give rise to different kind of operators: a constant kernel will measure the fraction of time φ is true in T, while an exponentially decreasing kernel will concentrate the focus on the initial part of T. We interpret the SCL formulae over signals.

Before describing the semantics, we give a couple of examples of SCL properties and their natural language expression. Considering again the glucose scenario presented in Section 5. The properties in Figure 5.1 are specified in SCL as

$$\begin{aligned}
\varphi_1 &: \langle \texttt{flat}_{[0,24h]}, 0.125\rangle\,(\mathbf{G}(t) \geq 180)\\
\varphi_2 &: \langle \texttt{flat}_{[0,24h]}, 0.95\rangle\,(\mathbf{G}(t) \geq 50)\\
\varphi_3 &: \langle \texttt{exp}_{[0,8h]}, 0.95\rangle\,(\mathbf{G}(t) \geq 180)
\end{aligned}$$

where φ_1 means the level of glucose ($\mathbf{G}$) is higher than 180 for more than 3 hours and φ_2 mean that the the same level is higher than 50 for more than 95% of the day (i.e., 24h), (see Figure 5.1). φ_3 leverages an exponential increasing kernel to describe the more dangerous situation of high glucose close (in time) to a meal .

Let us introduce the Boolean and quantitative semantics. Consider that temporal operators $\langle k_T, p\rangle$ are time-bounded, and for this reason, time-bounded signals are sufficient to assess the truth of every formula.

Definition 20 (Boolean Semantics)**.** *Given a signal* $\mathbf{s} \in \mathcal{S}^{\mathbb{T}}$*, the Boolean semantics* $\chi \colon \mathcal{S}^{\mathbb{T}} \times \mathbb{T} \times \mathcal{L}_{SCL} \to \{0, 1\}$ *is defined recursively by:*

$$\chi(\mathbf{s}, t, \mu) = 1 \iff \mu(\mathbf{s}(t)) = \top \text{ where } \mu(X) \equiv [g(X) \geq 0] \tag{5.3a}$$
$$\chi(\mathbf{s}, t, \neg\varphi) = 1 \iff \chi(\mathbf{s}, t, \varphi) = 0 \tag{5.3b}$$
$$\chi(\mathbf{s}, t, \varphi_1 \vee \varphi_2) = \max(\chi(\mathbf{s}, t, \varphi_1), \chi(\mathbf{s}, t, \varphi_2)) \tag{5.3c}$$
$$\chi(\mathbf{s}, t, \langle k_T, p\rangle\varphi) = 1 \iff k_T(t) * \chi(\mathbf{s}, t, \varphi) \geq p \tag{5.3d}$$

The atomic propositions μ are inequalities over the signal's variables. The semantics of negation and conjunction are the same as classical temporal logics. The semantics of $\langle k_T, p\rangle\varphi$ requires to compute the convolution of k_T with the truth value $\chi(\mathbf{s}, t, \varphi)$ of the formula φ as a function of time, seen as a Boolean signal, and compare it with the threshold p. An example of the Boolean semantics can be found in Figure 5.2(**a - bottom**) where four horizontal bars visually represent the validity of $\psi = \langle k_{[0,0.5]}, 0.5\rangle(s > 0)$, for

4 different kernels k (one for each bar). We can see that the the only kernel for which $\chi(s, \psi) = 1$ is the exponential increasing one, i.e., $k = \texttt{exp}[3]$.

Definition 21 (Quantitative semantics)**.** *The quantitative semantics* $\varrho\colon \mathcal{S}^{\mathbb{T}} \times \mathbb{T} \times \mathcal{L}_{SCL} \to \mathbb{R} \cup \{-\infty, +\infty\}$ *is defined as follows:*

$$\varrho(\mathbf{s}, t, \top) = +\infty \tag{5.4a}$$
$$\varrho(\mathbf{s}, t, \mu) = g(\mathbf{s}(t)) \text{ where } g \text{ is such that } \mu(\mathbf{s}) \equiv [g(\mathbf{s}) \geq 0] \tag{5.4b}$$
$$\varrho(\mathbf{s}, t, \neg\varphi) = -\varrho(\varphi, \mathbf{s}, t) \tag{5.4c}$$
$$\varrho(\mathbf{s}, t, \varphi_1 \vee \varphi_2) = \max(\varrho(\varphi_1, \mathbf{s}, t), \varrho(\varphi_2, \mathbf{s}, t)) \tag{5.4d}$$
$$\varrho(\mathbf{s}, t, \langle k_T, p\rangle\varphi) = \sup\{r \in \mathbb{R} \mid k_T(t) * [\varrho(\mathbf{s}, t, \varphi) > r] > p\} \tag{5.4e}$$

where in equation (5.4e) we use the Iverson brackets [Knu92], i.e., $[\varrho(\mathbf{s}, t, \varphi) > r]$ *is a function of* t *which assumes* 1 *if* $\varrho(\mathbf{s}, t, \varphi) > r$*, 0 otherwise.*

Intuitively the quantitative semantics of a formula φ w.r.t. a primary signal $\mathbf{s}$ describes the maximum allowed uniform translation of the secondary signals $\mathbf{g}(\mathbf{s}) = (g_1(\mathbf{s}), \ldots, g_{n(\varphi)}(\mathbf{s}))$ in φ preserving the truth value of φ. Stated otherwise, a robustness of r for φ means that all signals $\mathbf{s}'$ such that $\|\mathbf{g}(\mathbf{s}') - \mathbf{g}(\mathbf{s})\|_\infty \leq r$ will result in the same truth value for φ: $\chi(\mathbf{s}, t, \varphi) = \chi(\mathbf{s}', t, \varphi)$. Figure 5.2(b) shows this geometric concept visually. Let us consider the formula $\varphi = \langle k_{[0,3]}, 0.3\rangle(\mathbf{s} > 0)$, k a flat kernel. A signal $s(t)$ satisfies the formula if it is greater than zero for at most the 30% of the time interval $T = [0, 3]$. The robustness value corresponds to how much we can translate $s(t)$ s.t. the formula is still true, i.e., r s.t. $s(t) - r$ still satisfies φ. In the figure, we can see that $r = 0.535$. The formal justification of it is rooted in the correctness theorem (Theorem 3).

5.1.2 Soundness and Correctness

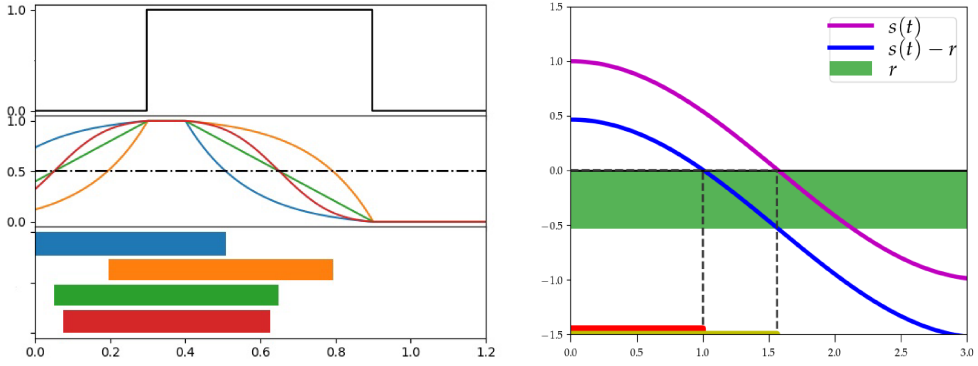
We turn now to discuss soundness and correctness of the quantitative semantics with respect to the Boolean one.

Theorem 2 (Soundness Property)**.** *The quantitative semantics is sound with respect to the Boolean semantics:*

$$\varrho(\mathbf{s}, t, \varphi) > 0 \implies (\mathbf{s}, t) \models \varphi$$
$$\varrho(\mathbf{s}, t, \varphi) < 0 \implies (\mathbf{s}, t) \not\models \varphi$$

Proof. The demonstration is by induction on the structure of the formula. The soundness holds for the atomic predicates and Boolean operators: this is the proof of soundness for classic robustness, see [DM10, FP09]. Consider then the formula $\langle k_T, p\rangle\varphi$. If $\varrho(\mathbf{s}, t, \langle k_T, p\rangle\varphi) \geq 0$, by Definition 21 (Equation 5.4e) we have that $k_T(t) * [\varrho(\mathbf{s}, t, \varphi) > 0] > p$. By the inductive hypothesis, φ satisfies the soundness property, hence it follows that $k_T(t) * \chi(\mathbf{s}, t, \varphi) \geq p$, i.e., $(\mathbf{s}, t) \models \langle k_T, p\rangle\varphi$. □

Theorem 3 (Correctness Property)**.** *The quantitative semantics* ϱ *satisfies the correctness property with respect to the Boolean semantics if and only if, for each formula* φ*,*



(a) (b)

Figure 5.2: (**a - top**) A Boolean signal $s(t)$ true in $[0.3, 0.9]$ and false outside. (**a - middle**) Convolution of the kernel function $(\texttt{exp}[3]_{[0,0.5]} * s)(t)$ (blue), $(\texttt{exp}[-3]_{[0,0.5]} * s)(t)$ (orange), $(\texttt{flat}_{[0,0.5]} * s)(t)$ (green) and $(\texttt{gauss}_{[0,0.5]} * s)(t)$ (red) with the signal above in the time windows. The horizontal threshold is set to 0.5. (**a - bottom**) The 4 horizontal bars show when $\chi(s, \psi, t) = 1$, with $\psi = \langle k_{[0,0.05]}, 0.5\rangle(s > 0)$, i.e when $(k_{[0,0.5]} * s)(t) > 0.5$. (**b**) Example of quantitative semantics of SCL. A signal $s(t)$ satisfies the formula $\varphi = \langle k_{[0,3]}, 0.3\rangle(\mathbf{s} > 0)$, with k a flat kernel, if it is greater than zero for at most the 30% of the time interval $T = [0, 3]$. The robustness value corresponds to how much we can translate $s(t)$ s.t. the formula is still true, i.e., r s.t. $s(t) - r$ still satisfies φ, (red line). In the figure we can see that $r = 0.535$.

it holds:

$$\forall \mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}^{\mathbb{T}},\ \|\mathbf{s}_1 - \mathbf{s}_2\|_\varphi < \varrho(\mathbf{s}_1, t, \varphi) \Rightarrow \chi(\mathbf{s}_1, t, \varphi) = \chi(\mathbf{s}_2, t, \varphi)$$

where the distance $\|\cdot\|_\varphi$ *has been defined in Chapter 2 (Definition 12).*

Proof. We proceed by induction on the structure of the formula. The correctness holds for atomic predicates and Boolean operators by the correctness proof of quantitative semantics for Metric Interval Temporal Logic (MITL) [FP09]. Let us consider the formula $\psi = \langle k_T, p\rangle\varphi$, with φ satisfying correctness by induction. Let $\mathbf{s}_1$ and $\mathbf{s}_2$ be two signals satisfying the hypothesis of the theorem, $\hat{r} = \varrho(\mathbf{s}_1, t, \psi)$ and $\chi(\mathbf{s}_1, t, \psi) = 1$ (the case $\chi(\mathbf{s}_1, t, \psi) = 0$ is similar). We have to prove that $\chi(\mathbf{s}_2, t, \psi) = 1$.

The hypothesis $\hat{r} = \varrho(\mathbf{s}_1, t, \psi)$ implies that $k_T(t) * [\varrho(\mathbf{s}_1, t, \varphi) > \hat{r}] > p$. This means that $\exists P \subseteq t + T$ s.t.

$$\forall \tau \in P,\ \varrho(\mathbf{s}_1, \tau, \varphi) > \hat{r} = \varrho(\mathbf{s}_1, t, \psi) > \|\mathbf{s}_1 - \mathbf{s}_2\|_\psi = \|\mathbf{s}_1 - \mathbf{s}_2\|_\varphi$$

(φ and ψ have the same atomic propositions). Then, for the inductive hypothesis, $\forall \tau \in P$, $\chi(\mathbf{s}_1, \tau, \varphi) = \chi(\mathbf{s}_2, \tau, \varphi) = 1$ which clearly implies that $k_T(t) * \chi(\mathbf{s}_2, t, \varphi) > p$, i.e., $\chi(\mathbf{s}_2, t, \psi) = 1$.

□

5.1.3 Expressiveness

We show that SCL is more expressive than the fragment of STL composed of the logical connectivities and the eventually $\mathbf{F}$ and globally $\mathbf{G}$ temporal operators, i.e., $STL(\mathbf{F}, \mathbf{G})$.

First of all, globally is easily definable in SCL. Take any kernel k_T, and observe that $\mathbf{G}_T\varphi \equiv \langle k_T, 1\rangle\varphi$ which results considering that

$$(\mathbf{s}, t) \models \langle k_T, 1\rangle\varphi \iff k_T(t) * \chi(\mathbf{s}, t, \varphi) = 1$$

which holds only if φ is true in the whole interval T. This statement is valid only if we restrict ourselves to Boolean signals of finite variation, as for [MN04], which are changing truth value a finite amount of times and are never true or false in isolated points: in this way we do not have to care what happens in sets of zero measure. With a similar restriction in mind, we can define the eventually, provided we can check that

$$k_T(t) * \chi(\mathbf{s}, t, \varphi) > 0$$

To see how this is possible, start from the fundamental equation $k_T(t) * \chi(\mathbf{s}, t, \neg\varphi) = 1 - k_T(t) * \chi(\mathbf{s}, t, \varphi)$. By applying 5.3d and 5.3b we easily get $\chi(\mathbf{s}, t, \neg\langle k_T, 1-p\rangle\neg\varphi) = 1 \iff k_T(t) * \chi(\mathbf{s}, t, \neg\varphi) < 1 - p \iff k_T(t) * \chi(\mathbf{s}, t, \varphi) > p$. For compactness we write $\langle k_T, p\rangle^* = \neg\langle k_T, 1-p\rangle\neg$, and thus define the eventually modality as $\mathbf{F}_T\varphi \equiv \langle k_T, 0\rangle^*\varphi$. Furthermore, consider the uniform kernel $\mathtt{flat}_T$: a property of the form $\langle \mathtt{flat}_T, 0.5\rangle\varphi$, requesting φ to hold at least half of the time interval T, cannot be expressed in STL, showing that SCL is more expressive than STL$(\mathbf{F}, \mathbf{G})$.

5.2 Monitoring Algorithm

We now provide two offline monitor algorithms for the Boolean semantics of SCL formulae (approach I and approach II) and a monitor algorithm for the quantitative semantics.

Boolean monitoring: approach I. The first method we present works for a subclass of kernels (mostly used in practice), satisfying a semigroup property. The second method, which is an extension of the first one, instead, works for generic kernels.

Consider the following SCL formula $\langle k_{[T_0,T_1]}, p\rangle\varphi$ and a Boolean signal $\mathbf{s}$, we are interested in computing $\chi(\mathbf{s}, t, \langle k_{[T_0,T_1]}, p\rangle\varphi) = (H(t) - p \geq 0)$, as a function of t, where H is the following convolution function

$$H(t) = k_T(t) * \chi(\mathbf{s}, t, \varphi) = \int_{t+T_0}^{t+T_1} k_{[T_0,T_1]}(\tau - t)\chi(\mathbf{s}, \tau, \varphi)d\tau \tag{5.5}$$

It follows that the efficient monitoring of the Boolean semantics of SCL is linked to the efficient evaluation of $H(t) - p$, which is possible if exists h sufficiently big such that $H(t+h)$ can be computed by reusing the value of $H(t)$ previously stored. If the kernel is a constant function (i.e., $\forall t \in T,\ k(t) = k_0$), linear (i.e., $\forall t_0, t_1 \in T,\ k(t_0 + t_1) = k(t_0) + k(t_1)$), or exponential (i.e., it satisfies the semigroup property: $\forall t_0, t_1 \in T,\ k(t_0 + t_1) = k(t_0) \cdot k(t_1)$), this can be efficiently done:

Constant: $H_{const}(t + h) - p = H(t) - p + I_{k_T}(t + T_1, t + T_1 + h) - I_{k_T}(t + T_0, t + T_0 + h)$.

Linear: $H_{lin}(t+h) - p = H(t) - p + I_{k_T}(t+T_1, t+T_1+h) - I_{k_T}(t+T_0, t+T_0+h) + k_T(-h)H_{const}(t+h)$.

Exponential: $H_{exp}(t+h) - p = H(t) - p + I_{k_T}(t+T_1, t+T_1+h) - I_{k_T}(t+T_0, t+T_0+h) + p(1-e^{-\alpha h})$.

where $I_{k_T}(a, a+h) = \int_a^{a+h} k_T(\tau - a)\chi(\mathbf{s}, \tau, \varphi)d\tau$.

These relationships can be easily exploited to obtain an algorithm that computes the values of $H(t) - p$. The computation can be optimized by suitably tracking the times in which the signal $\chi(\mathbf{s}, \tau, \varphi)$ switches values. The algorithm is fully reported in the following subparagraph.

The Algorithm. We now describe the monitoring algorithm for the convolution formula $\chi(\mathbf{s}, t, \langle k_{[T_0,T_1]}, p\rangle\varphi)$. The monitors for the others formulae are described in [DFM13]. The Boolean signal can be decomposed (see [MN04]) in a sequence of disjoint unitary signals $U = U^1 + \cdots + U^n$ s.t. each $U^i(t)$ is true only in a single interval of time $[U_0^i, U_1^i]$, false outside. We denote with subscript 0 the lower bounds and with subscript 1 the upper bound, whereas the superscript i indicates a specific unitary signal.

The key idea consists in monitoring H by choosing h such that between t and $t+h$ the Boolean signal $\chi(\mathbf{s}, t, \varphi)$ assumes a constant value (i.e., 1 or 0) in $[t+T_0, t+T_0+h]$ and $[t+T_1, t+T_1+h]$, respectively. Therefore, I_{k_T} can be easily computed as the difference of the primitive of k_T evaluated in $0, h$ and $T_1 - T_0 - h, T_1 - T_0$ multiplied by the constant value assumes by the signal. For example if $\chi(\mathbf{s}, \tau, \varphi) = 1$ if $\tau \in [t+T_0, t+T_0+h]$ than $I_{k_T}(t+T_0, t+T_0+h) = K_T(h) - K_T(0)$ where K_T is the primitive of the kernel k_T. Let us describes this procedure (Algorithm 3) in details. The algorithm takes as inputs the interval $[t, t_e]$ where we want to monitor $\chi(\mathbf{s}, t, \langle k_{[T_0,T_1]}, p\rangle\varphi)$, the kernel $k_{[T_0,T_1]}$, the Boolean signal $\chi(\mathbf{s}, t, \varphi)$, and the threshold $p \in [0, 1]$. At line 1 the Boolean signal $\chi(\mathbf{s}, t, \varphi)$ is decomposed into a sequence of unitary signals as explain above, afterward (line 2) the algorithm initializes $H(t)$ to $(k_{[T_0,T_1]} * U)(t)$, which is evaluated by a standard integration technique, and the value of *init* and *end*, which are two integers pointing to the first unitary signal (i.e, U^{init}) such that $U_1^{init} > t + T_0$, and the last unitary signal (i.e., U^{end}) such that $U_0^{end} \leq t + T_1$. A while cycle (line $4-32$) performs the integration step until the convolution is evaluated on the entire interval $[t, t_e]$. Depending on the cases, if the bounds of the convolution window (i.e., $t + T_0$ and $t + T_1$) intersect an unitary signal composing U, four different integration strategies are performed (blocks at lines: $5-8, 9-12, 13-16, 17-20$), and the proper value of the integration step (i.e., h) which avoids discontinuity is also evaluated (lines: $6, 10, 14, 18$). The value of h, indeed, implies that between t and $t+h$ no new intersection of unitary signals U^i with the convolution windows occurs. Afterward, the finite difference $H(t+h) - H(t)$ is stored in ΔH and the integration step ($H(t+h) = H(t) + \Delta H$) is performed (line 21). The value of this finite difference depends on the particular kernel used. It involves the evaluation of the primitive of k_T which we denote by capital letter K_T plus other terms (i.e, zero for the flat kernel, $p(1-e^{-\alpha h})$ for the exponential kernel, etc.).

If during this integration step $H(t) - p$ changes sign (line 22), then the exact moment is evaluated by means of a root finding routine (line 23). These time values define the Boolean semantics of $\chi(\mathbf{s}, t, \langle k_{[T_0,T_1]}, \delta, p\rangle\varphi)$. Finally, in lines $25-30$, the values of t, *init* and *end* are updated and the cycle continues.

Algorithm 3 (Approach I) Monitoring Algorithm: $\chi(\mathbf{s}, t, \langle k_{[T_0,T_1]}, p\rangle\varphi)$

Require: $t, t_e, k_{[T_0,T_1]}, \chi(\mathbf{s}, t, \varphi), p$
1: $U \leftarrow$ DECOMPOSE$(\chi(\mathbf{s}, t, \varphi))$
2: $H(t) = (k_{[T_0,T_1]} * U)(t)$
3: INITIALIZE$(init, end)$

4: **while** $(t < t_e)$ **do**
5: **if** $(t + T_0 \in U^{init}$ **and** $t + T_1 \in U^{end})$ **then**
6: $h \leftarrow \min(t + T_0 - U_1^{init}, U_1^{end} - t - T_1)$
7: $\Delta H \leftarrow K_T(h) - K_T(0) + K_T(T_1 - T_0) - K_T(T_1 - T_0 - h) +$ other terms.
8: **end if**
9: **if** $(t + T_0 \in U^{init}$ **and** $t + T_1 \notin U^{end})$ **then**
10: $h \leftarrow \min(t + T_0 - U_1^{init}, U_0^{end+1} - t - T_1)$
11: $\Delta H \leftarrow K_T(h) - K_T(0) +$ other terms.
12: **end if**
13: **if** $(t + T_0 \notin U^{init}$ **and** $t + T_1 \in U^{end})$ **then**
14: $h \leftarrow \min(U_1^{end} - t - T_1, U_0^{init} - t - T_0)$
15: $\Delta H \leftarrow K_T(T_1 - T_0) - K_T(T_1 - T_0 - h) +$ other terms.
16: **end if**
17: **if** $(t + T_0 \notin U^{init}$ **and** $t + T_1 \notin U^{end})$ **then**
18: $h \leftarrow \min(U_0^{init} - t - T_0, U_0^{end+1} - t - T_1)$
19: $\Delta H \leftarrow$ other terms.
20: **end if**

21: $H(t + h) \leftarrow H(t) + \Delta H$
22: **if** $(H(t + h) - p) \cdot (H(t) - p) \leq 0)$ **then**
23: FIND$(\{x \in [t, t + h] \mid H(t + x) - p = 0\})$
24: **end if**
25: $t \leftarrow t + h$
26: **if** $(t + T_0 \geq U_1^{init})$ **then**
27: $init \leftarrow init + 1$
28: **end if**
29: **if** $(t + T_1 \leq U_1^{end})$ **then**
30: $end \leftarrow end + 1$
31: **end if**
32: **end while**

Boolean monitoring: approach II. The previous algorithm can be easily modified to work with generic kernels. To see how to proceed, assume the signal $\chi(\mathbf{s}, \tau, \varphi)$ to be *unitary*, namely that it is true in a single interval of time, say from time u_0 to time u_1, and false elsewhere. In this case, it readily follows that the convolution with the kernel will be non-zero only if the interval $[u_0, u_1]$ intersects the convolution window $t + T$. Inspecting Figure 5.3, we can see that sliding the convolution window forward of a small time h corresponds to sliding the positive interval of the signal $[u_0, u_1]$ of h time units backward.

In case $[u_0, u_1]$ is fully contained into $t+T$, by making h infinitesimal and invoking the fundamental theorem of calculus, we can compute the derivative of $H(t)$ with respect to time as $\frac{d}{dt}H(t) = k_T(u_0 - t) - k_T(u_1 - t)$. By taking care of cases in which the overlap is only partial, we can derive a general formula for the derivative:

$$\frac{d}{dt}H(t) = k_T(u_0 - (t+T_0))I\{u_0 \in t+T\} - k_T(u_1 - (t+T_1))I\{u_1 \in t+T\}, \quad (5.6)$$

where I is the indicator function, i.e., $I\{u_i \in t+T\} = 1$ if $u_i \in t+T$ and zero otherwise, and $T = [T_0, T_1]$. This equation can be seen as a differential equation that can be integrated with respect to time by standard ODE solvers (taking care of discontinuities, e.g., by stopping and restarting the integration at boundary times when the signal changes truth value), returning the value of the convolution for each time t. The initial value is $H(t)$, that has to be computed integrating explicitly the kernel (or setting it to zero if $u_0 \geq T_1$). If the signal $\chi(\mathbf{s}, t, \varphi)$ is not unitary, we have to add a term like the right-hand side of 5.6 in the ODE of $H(t)$ for each unitary component (positive interval) in the signal. We also use a root finding algorithm integrated with the ODE solver to detect when the property will be true or false, i.e., when $H(t)$ will be above or below the threshold p.

The Algorithm. We now describe this algorithm (Algorithm 4), which efficiently evaluates the convolution between a generic kernel, k_T, and a Boolean signal $\chi(\mathbf{s}, t, \varphi)$. It is simple customization of the Algorithm 3. In this case, indeed, in order to integrate a general ODE defined in Equation (5.6) we have to bound the translation from $H(t)$ to $H(t+h)$ with a suitable integration step (i.e., $h \leq \delta$, see lines $6, 10, 14, 18$). Also the finite differences ΔH (lines: $7, 11, 15, 19$) have a different update rule considering that a generic kernel has no special properties w.r.t translation such as the constant, linear or exponential kernels. It means that the effect of each unitary signal contained in the convolution window during the translation has to be estimated separately (summation at lines $7, 11, 15, 19$). The other steps are the same as Algorithm 3.

Time Complexity. The time complexity of both the algorithms depends on the initialization phase (line $1-3$) which is linear w.r.t the unitary signals composing $\chi(\mathbf{s}, t, \varphi)$ from t to $t+T_1$ plus the number of h translations needed to reach t_e multiplied by the computational costs to evaluate the finite difference ΔH and eventually to solve the equation at line 23. In the first approach, Algorithm 3, the computational cost depends on the evaluation of the primitive function of the kernel k_T which we denote by capital letter K_T and which is performed for a maximum of the quadruple of the unitary signals composing $\chi(\mathbf{s}, t, \varphi)$. For this reason the Algorithm 3 has a time complexity bounded by $O(|U|_{[t,t_e]})$ where $|U|_{[t,t_e]}$ is the number of unitary signals composing U between t and t_e.

In the second approach Algorithm 4 that computational cost is proportional to the cost of numerically integrating the differential equation 5.6 which is performed in lines $7, 11, 15, 19$. Each integrations consists in the evaluation of the kernel k_t on number of points corresponding to the unitary signals composing $\chi(\mathbf{s}, t, \varphi)$ in the convolution windows $t+T$. Using a solver with constant step size δ, the complexity is proportional to the number of integration steps, times the number $|U|_{[t,t_e]}$ of unitary components

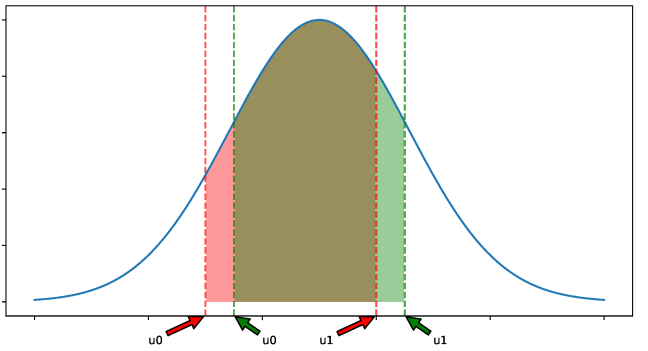


Figure 5.3: Sketch of the general monitoring algorithm. The green arrows represents $[u_0, u_1]$ in the constitutional window at time t, the red arrows instead represents the same interval at time $t + h$ (backwards translation).

in the signal: $O((T_s/\delta)|U|_{[t,t_e]})$. By fixing the precision of the integrator to δ, the number $|U|_{[t,t_e]}$ of unitary components is itself bounded by (T_s/δ). This is because the integral, being sampled at δ units of time, cannot cross the threshold p more than once in each δ-time interval. It follows that $O(T_s/\delta)$ is an upper bound on $|U|_{[t,t_e]}$ for any signal $\chi(\mathbf{s}, t, \varphi)$ encountered in the monitoring procedure. This gives a complexity upper bound of the order or $O((T_s/\delta)^2)$ for each convolution operator of a formula. Considering that the time-complexity of the other logical operators is bounded by $O(|U|_{[t,t_e]}) = O((T_s/\delta))$ (see [DFM13]), we conclude that for a formula of size M (i.e., having M logical operators) and integration step δ, the complexity of monitoring is $O(M(T_s/\delta)^2)$.

Quantitative Monitoring. In this chapter, we follow a simple approach to monitor it: we run the Boolean monitor for different values of r and t in a grid, using a coarse grid for r, and compute at each point of such grid the value $H(t, r) = k_T(t) * [\varrho(\mathbf{s}, t, \varphi) > r] - p$. Relying on the fact that $H(t, r)$ is monotonically decreasing in r, we can find the correct value of r, for each fixed t, by running a bisection search starting from the unique values r_k and r_{k+1} in the grid such that $H(t, r)$ changes sign, i.e., such that $H(t, r_k) < 0 < H(t, r_{k+1})$. The bounds of the r grid are set depending on the bounds of the signal, and may be expanded (or contracted) during the computation if needed. A more efficient procedure for quantitative monitoring is in the top list of our future work, and it can be obtained by exploring only a portion of such a grid, combining the method with the Boolean monitor based on ODEs, and alternating steps in which we advance time from t to $t+h$ (fixing r_t to its exact value at time t), by integrating ODEs and computing $H(t + h, r_t)$, and steps in which we adjust the value of r_t at time $t + h$ by locally increasing or decreasing its value (depending if $H(t + h, r_t)$ is negative or positive), finding r_{t+h} such that $H(t + h, r_{t+h}) = 0$.

Algorithm 4 (Approach II) Monitoring Algorithm: $M_{\langle k_{[T_0,T_1]},p\rangle\varphi}$

Require: $t_e, k_{[T_0,T_1]}, \chi(\mathbf{s},t,\varphi), p, \delta$
$U \leftarrow \text{DECOMPOSE}(\chi(\mathbf{s},t,\varphi))$
$H(t) = (K_{[T_0,T_1]} * U)(t)$
INITIALIZE$(init, end)$

while $(t < t_e)$ **do**
  if $(t+T_0 \in U^{init}$ **and** $t+T_1 \in U^{end})$ **then**
    $h \leftarrow \min(\delta, t+T_0-U_1^{init}, U_1^{end}-t-T_1)$
    $\Delta H \leftarrow K_T(T_0) - K_T(T_1) + \sum_{i=init+1}^{end-1}(k_T(U_0^i - t) - k_T(U_1^i - t))$
  end if
  if $(t+T_0 \in U^{init}$ **and** $t+T_1 \notin U^{end})$ **then**
    $h \leftarrow \min(\delta, t+T_0-U_1^{init}, U_0^{end+1}-t-T_1)$
    $\Delta H \leftarrow K_T(T_0) + \sum_{i=init+1}^{end}(k_T(U_0^i - t) - k_T(U_1^i - t))$
  end if
  if $(t+T_0 \notin U^{init}$ **and** $t+T_1 \in U^{end})$ **then**
    $h \leftarrow \min(\delta, U_1^{end}-t-T_1, U_0^{init}-t-T_0)$
    $\Delta H \leftarrow -K_T(T_1) + \sum_{i=init}^{end-1}(k_T(U_0^i - t) - k_T(U_1^i - t))$
  end if
  if $(t+T_0 \notin U^{init}$ **and** $t+T_1 \notin U^{end})$ **then**
    $h \leftarrow \min(\delta, U_0^{init}-t-T_0, U_0^{end+1}-t-T_1)$
    $\Delta H \leftarrow \sum_{i=init}^{end}(k_T(U_0^i - t) - k_T(U_1^i - t))$
  end if

  $H(t+h) \leftarrow H(t) + h \cdot \Delta H$
  if $(H(t+h)-p)\cdot(H(t)-p) \leq 0)$ **then**
    FIND$(\{x \in [t,t+h] \mid H(t+x)-p=0\})$
  end if
  $t \leftarrow t+h$
  if $(t+T_0 \geq U_1^{init})$ **then**
    $init \leftarrow init+1$
  end if
  if $(t+T_1 \leq U_1^{end})$ **then**
    $end \leftarrow end+1$
  end if
end while

5.3 Case Study: Artificial Pancreas

In this example, we show how SCL can be useful in the specification and monitoring of the Artificial Pancreas (AP) systems. The AP is a closed-loop system of insulin-glucose for the treatment of Type 1 diabetes (T1D). The T1D is a chronic disease caused by the inability of the pancreas to secrete insulin, a hormone essential to regulate the blood glucose level. In the AP system, a Continuous Glucose Monitor (CGM) detects the blood glucose levels, and a pump delivers insulin through injection managed by a

software-based controller.

The efficient design of control systems to automate the delivery of insulin is still an open challenge for many reasons. Many activities are still under control of the patient, e.g., increasing insulin delivery at meal times (meal bolus), and decreasing it during physical activity. A complete automatic control includes many risks for the patient. High level of glucose (hyperglycemia) implies ketoacidosis and low level (hypoglycemia) can be fatal leading to death. The AP controller must tolerate many unpredictable events such as pump failures, sensor noise, meals and physical activity.

Formal methods can be exploited to specify and control the key properties that the system has to satisfy. However, standard verification procedures are not feasible, first because we need efficient and fast monitoring processes able to detect events *in vivo* and second because, even considering *in silico* models, the system has very large state-spaces and is mostly affected by noise and disturbances. In this setting, simulation-based verification techniques are a valid alternative. Furthermore, the quantitative semantics can be used to detect potential failures. AP Controller Falsification via SMT solver [SPB+17a] and robustness of STL [CFMS15] has been recently proposed. In particular, [CFMS15] formulates a series of STL properties testing insulin-glucose regulatory system. Here we show the advantages of using SCL for this task.

PID Controller. Consider a system/process which takes as input a function $u(t)$ and produces as output a function $y(t)$. A PID controller is a simple closed-loop system aimed at maintaining the output value $y(t)$ as close as possible to a set point sp. It continuously monitors the error function, i.e., $e(t) = sp - y(t)$ and defines the input of the systems according to $u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(s)ds + K_d \cdot \frac{d}{dt}e(t)$. The *proportional* ($K_p$), *integral* ($K_i$) and *derivative* ($K_d$) parameters uniquely define the PID controller and have to be calibrated in order to achieve a proper behavior.

System. PID controllers have been successfully used to control the automatic infusion of insulin in AP. In [SPB+17a], for example, different PID have been synthesized to control the glucose level for the well studied Hovorka model [HCC+04]:

$$\frac{d}{dt}\mathbf{G}(t) = \mathbf{F}(\mathbf{G}(t), u(t), \mathbf{\Theta}), \tag{5.7}$$

where the output $\mathbf{G}(t)$ represents the glucose concentration in blood and the input $u(t)$ is the infusion rate of bolus insulin which has to be controlled. The vector $\mathbf{\Theta} = (dg1, dg2, dg3, T_1, T_2)$ are the control parameters which define the quantity of carbohydrates $(dg1, dg2, dg3)$ assumed during the three daily meals and the inter-times between each of them T_1 and T_2. Clearly a PID controller for Equation (5.7) has to guarantee that under different values of the control parameters $\mathbf{\Theta}$ the glucose level remains in the *safe region* $\mathbf{G}(t) \in [70, 180]$. In [SPB+17a] four different PID that satisfy the safe requirement, have been discovered by leveraging SMT solver under the assumption that the inter-times T_1 and T_2 are both fixed to 300 minutes (5 hrs) and that $(dg1, dg2, dg3) \in (\mathcal{N}(40, 10), \mathcal{N}(90, 10), \mathcal{N}(60, 10))$, which correspond to the average quantity of carbohydrates contained in breakfast, lunch and dinner. ($\mathcal{N}(\mu, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2). Here, we consider the PID controller C_1 which has been synthesized in [SPB+17a] by fixing the glucose setting

point sp to $110mg/dl$ and maximizing the probability to remain in the safe region, provided a distribution of the control parameter $\mathbf{\Theta}$ as explained before. We consider now some properties which can be useful to check expected or anomalous behaviors of an AP controller.

Hypoglycemia and Hyperglycemia. Consider the following informal specifications: *never during the day the level of glucose goes under* $70\,mg/dl$, and *never during the day the level of glucose goes above* $180\,mg/dl$, which technically mean that the patient is never under Hypoglycemia or hyperglycemia, respectively. These behaviours can be formalized with the two **STL** formulas $\varphi^{HO}_{STL} = \mathbf{G}_{[0,24h]}\mathbf{G}(t) \geq 70$ and $\psi^{HR}_{STL} = \mathbf{G}_{[0,24h]}\mathbf{G}(t) \leq 180$. The problem of STL is that it does not distinguish if these two conditions is violated for a second, few minutes or even hours. It only says those events happen. Here we propose stricter requirements described by the two following **SCL** formulas $\varphi^{HO}_{SCL} = \langle \mathtt{flat}_{[0,24h]}, 0.95\rangle\, \mathbf{G}(t) \geq 70$ for the Hypoglycemia regime, and $\varphi^{HR}_{SCL} = \langle \mathtt{flat}_{[0,24h]}, 0.95\rangle\, \mathbf{G}(t) \leq 180$ for the hyperglycemia regime. We are imposing not that the globally in a day the hypoglycemia and the hyperglycemia event never occur, but that these conditions persist for at least 95% of the day (i.e., 110 minutes). We will show above in a small test case how this requirement can be useful.

Prolongated Conditions. As already mentioned in the motivating example, the most dangerous conditions arise when hypoglycemia or hyperglycemia last for a prolongated period of the day. In this context a typical condition is the **Prolongated Hyperglycemia** which happens if the total time under hyperglycemia (i.e., $\mathbf{G}(t) \geq 180$) exceed the 70% of the day, or the **Prolongated Severe Hyperglycemia** when the level of glucose is above $300\,mg/dl$ for at least 3 hrs in a day. The importance of these two conditions has been explained in [SKC+17], however the authors cannot formalized them in STL. On the contrary, SCL is perfectly suited to describe these conditions as shown by the following two formulas: $\varphi^{PHR}_{SCL} = \langle \mathtt{flat}_{[0,24h]}, 0.7\rangle\, \mathbf{G}(t) \geq 180$ and $\varphi^{PSHR}_{SCL} = \langle \mathtt{flat}_{[0,24h]}, 0.125\rangle\, \mathbf{G}(t) \geq 300$. Here we use flat kernels to mean that the period of a day where the patient is under Hyperglycemia or Severe Hyperglycemia does not count to the evaluation of the Boolean semantics. Clearly, a hyperglycemia regime in different times of the day can count differently. In order to capture this "preference" we can use non-constant kernels.

Correctness of the insulin delivery. During the hypoglycemia regime the insulin should not be provided. The SCL formula: $\mathbf{G}_{[0,24h]}(\langle \mathtt{flat}_{[0,10min]}, 0.95\rangle\, \mathbf{G}(t) \leq 70 \rightarrow \langle \mathtt{flat}_{[0,10min]}, 0.90\rangle\, I \leq 0)$ states that if during the next 10 minutes the patient is in hypoglycemia for at least the 95% of the time then the delivering insulin pump is shut off (i.e., $I \leq 0$) for at least the 90% of the time. This is the "cumulative" version of the STL property $\mathbf{G}_{[0,24h]}(\mathbf{G}(t) \leq 70 \rightarrow I \leq 0)$ which says that in hypoglycemia regime no insulin should be delivered. During the hyperglycemia regime the insulin should be provided as soon as possible. The property SCL formula: $\mathbf{G}_{[0,24h]}(\mathbf{G}(t) \geq 300 \rightarrow \langle \mathtt{exp}[-1]_{[0,10min]}, 0.9\rangle\, I \geq k)$ says that if we are in severe hyperglycemia regime (i.e., $\mathbf{G}(t) \geq 300$) the delivered insulin should be higher than k for at least the 90% of the following 10 minutes. We use a negative exponential kernel to express (at the robustness level) the preference of having a higher value of delivered insulin as soon as possible.

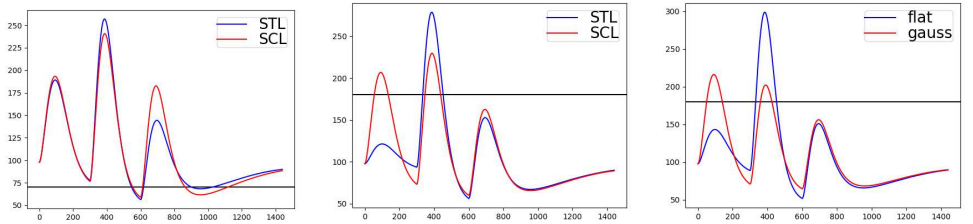


Figure 5.4: **(left),(middle)** The solution of the SCL formula falsification (red line) maximize the time under hypoglycemia (left) and hyperglycemia (right), whereas the solution of the STL formula falsification (blue line) maximizes the displacement w.r.t the predicate thresholds. **(right)** Solution of the falsification for the SCL properties φ^{PHR}_{SCL} (blue line) and $\varphi^{PHR}_{SCL(Gauss)}$ (red line) which implement `flat` and `gauss` kernel, respectively.

Test Case: falsification. As a first example we show how SCL logic can be effectively used for falsification. The AP control system has to guarantee that the level of glucose remains in a safe region, as explained before. The falsification approach consists in identifying the control parameters ($\boldsymbol{\Theta}^*$) which force the system to violate the requirements, i.e., to escape from the safe region. The standard approach consists in minimizing the robustness of suited temporal logic formulas which express the aforementioned requirements, e.g. $\varphi^{HR}_{SCL}, \varphi^{HO}_{SCL}$. In this case the minimization of the STL robustness forces the identification of the control parameters which causes the generation of trajectories with a maximum displacement under the threshold 70 or above 180. To show differences among the STL and SCL logics, we consider the PID C_1 + Hovorka model and perform a random sampling exploration among its input parameters. At each sampling we calculate the robustness of the STL formulas φ^{HO}_{STL} and the SCL formula φ^{HO}_{SCL} and separately store the minimum robustness value. For this minimum value, we estimate the maximum displacement with respect to the hypoglycemia and hyperglycemia thresholds and the maximum time spent violating the hypoglycemia and hyperglycemia thresholds. Fig. 5.4(left, middle) shows the trajectory with minimum robustness. We can see that the trajectory which minimizes the robustness of the STL formula has a higher value of the displacement from the hypoglycemia (13) and hyperglycemia (98) thresholds than SCL trajectory (which are 11 and 49 respectively). On the contrary, the trajectory which minimizes the robustness of the SCL formula remains under hypoglycemia (for 309 min) and hyperglycemia (for 171 min) longer than the STL trajectory (189 min and 118 min, respectively). These results show how the convolutional operator and its quantitative semantics can be useful in a falsification procedure. This is particularly evident in the Hyperglycemia case (Fig. 5.4 (middle)) where the falsification of the SCL hyperglycemia formula φ^{HR}_{SCL} shows two subintervals where the level of glucose is above the threshold. In order to show the effect of non-homogeneous kernel, we perform the previous experiment, with the same setting, for properties φ^{PHR}_{SCL} and $\varphi^{PHR}_{SCL(Gauss)}$. From the results (Fig. 5.4 (right)) is evident how the Gaussian kernel of property $\varphi^{PHR}_{SCL(Gauss)}$ forces the glucose to be higher of the hyperglycemia threshold

	noise free				with noise			
	h_{55}	h_{60}	h_{65}	h_{70}	h_{55}	h_{60}	h_{65}	h_{70}
$\mathbf{F}[0, 24h]$	0.00	0.19	0.81	1.00	0.98	1.00	1.00	1.00
$\langle \texttt{flat}_{[0,24]}, 0.03 \rangle$	0.00	0.00	0.20	0.91	0.00	0.02	0.77	1.00

Table 5.2: Results of the falsification test case. The performance of STL and SCL formulas verified on the PID C_1 + Hovorka model with noise and noise free are compared. The STL formula on the noisy model is uninformative.

just before the first meal ($t \in [0, 200]$) and ignores for example the last meal ($t \geq 600$).

Test Case: noise robustness. Now we compare the sensitivity to noise of SCL and STL formulae. We consider three degrees of hypoglycemia $h_k(t) = \{G \leq k\}$, where $k \in \{55, 60, 65, 70\}$ and estimate the probability that the Hovorka model controlled by the usual PID C_1 (i.e., PID C_1 + Hovorka Model) satisfies the STL formulas $\varphi^k_{STL} = \mathbf{F}[0, 24h]\, h_k$ and the SCL formulas $\varphi^k_{\text{SCL}} = \langle \texttt{flat}_{[0,24h]}, 0.03 \rangle\, h_k$ under the usual distribution assumption for the control parameters $\boldsymbol{\Theta}$. The results are reported in column "noise free" of Table 5.2. Afterwards, we consider a noisy outcome of the same model by adding a Gaussian noise, i.e., $\varepsilon \in \mathcal{N}(0, 5)$, to the generated glucose trajectory. We estimate the probability that this noisy system satisfies the STL and SCL formulas above, see column "with noise" of Table 5.2. The noise corresponds to the disturbance of the original signals which can occur, for example, during the measurement process.

As shown in Table 5.2, the probability estimation of the STL formulas changes drastically with the addition of noise (the addition of noise forces all the trajectory to satisfy the STL formula). On the contrary, the SCL formulas φ^k_{SCL} are more stable under noise and can be even used to approximate the probability of the STL formulas on the noise-free model. To better asses this, we checked how much the STL formula φ^k_{STL} and the SCL formula φ^k_{SCL}, evaluated in the noisy model, agree with the STL formula φ^k_{STL} evaluated in the noise-free model, by computing their truth value on 2000 samples, each time choosing a random threshold $k \in [50, 80]$. The score for STL is 56%, while SCL agrees on 78% of the cases.

5.4 Conclusion and Future Works

We have introduced SCL, a novel specification language that employs signal processing operations to reason about temporal behavioural patterns. The key idea is the definition of a family of modal operators which compute the convolution of a kernel with the signal and check the obtained value against a threshold. Our case study on monitoring glucose level in artificial pancreas demonstrates how SCL empowers the classical temporal logic operators (i.e., such as *finally* and *globally*) with noise filtering capabilities, and enable us to express temporal properties with soft time bounds and with nonsymmetric treatment of time instants in a unified way. The convolution operator of SCL can be seen as a syntactic bridge between temporal logic and digital signal processing, trying to combine the advantages of both these two worlds. This point of view can be explored further, bringing into the monitoring algorithms of SCL tools from frequency analysis

of signals. Future work includes the release of a Python library and the design of efficient monitoring algorithms also for the quantitative semantics. Finally, we also plan to develop online monitoring algorithms for real-time systems using hardware dedicated architecture such as field-programmable gate array (FPGA) and digital signal processor (DSP).

6

Classification of Trajectories

Learning temporal logic requirements from temporal data is an emergent research field gaining momentum in the rigorous engineering of cyber-physical systems. Classical machine learning methods typically generate very powerful black box (statistical) models. However, these models often do not help in the comprehension of the phenomenon they capture. Temporal logic provides a precise formal specification language that can be easily interpreted by humans. The possibility to describe data sets in a concise way using temporal logic formulae can thus help to better clarify and comprehend which are the emergent patterns for the system at hand. A clearcut example is the problem of anomaly detection, where the input is a set of trajectories describing regular or anomalous behaviors, and the goal is to learn a classifier that not only can be used to detect anomalous behaviors at runtime, but also gives insights on what characterizes an anomalous behavior. Learning temporal properties is also relevant in combination with state of the art techniques for search-based falsification of complex closed-loop systems [ZSD+15, SKC+17, BBNS15, SPB17b], as it can provide an automatic way to describe unwanted (or desired) behaviors that the system needs to satisfy.

In this contribution, we consider the problem of learning a temporal logic specification from a set of trajectories which are labeled either as *good* for the normally expected behaviors or as *bad* for the anomalous ones. The goal is to automatically synthesize both the structure of the formula and its parameters providing a temporal logic classifier that can discriminate as much as possible the bad and the good behaviors. This specification can be turned into a monitor that outputs a positive verdict for good behaviors and a negative verdict for bad ones.

Related Work. Mining temporal logic requirements is an emerging field of research in the analysis of cyber-physical systems [ADMN11, YHF12, BBS14a, JDDS15, NKJ+17, ZRWT17, KJB17, BVP+16, BBS+14b].

This approach is orthogonal to Active Automata Learning (AAL) such as $L*$ Angluin's algorithm [Ang87] and its recent variants [IHS14, SHI12]. AAL is suitable to capture the behaviours of black box reactive systems and it has been successfully employed in the field of Cyber-physical Systems (CPS) to learn how to interact with the

surrounding environments [CDKM13, FTHC14]. Mining temporal logic requirements has the following advantages with respect to AAL. The first is that it does not require to interact with a reactive system. AAL needs to query the system in order to learn a Mealy machine representing the relation between the input provided and the output observed. Mining temporal logic requirements can be applied directly to a set of observed signals without the necessity to provide any input. The second is the possibility to use temporal logic requirements within popular tools (such as Breach [Don10] and S-TaLiRo [ALFS11a]) for monitoring and falsification analysis of CPS models.

Most of the literature relates to *Parametric Signal Temporal Logic* (PSTL), which extends *Signal Temporal Logic* (STL) by allowing parametrization of formulae. In [ADMN11], where PSTL was introduced, the authors provided a geometric approach to identify a subset of formula's parameters which makes a signal to satisfy that formula. Xu et al. have recently proposed in [XJ18] a temporal logic framework called *CensusSTL* for multi-agent systems that consists of *an outer logic STL formula* with a variable in the predicate representing the number of agents whose satisfying *an inner logic STL formula*. In the same paper the authors also propose a new inference algorithm similar to [ADMN11] that given the templates for both the *inner* and *outer* formulas, searches for the optimal parameter values that make the two formulas capturing the trajectory data of a group of agents. In [HDF18] the authors use the same parametric STL extension in combination with the quantitative semantics of STL to perform a counterexample guided inductive parameter synthesis. This approach consists in iteratively generating a counterexample by executing a falsification tool for a template formula. The counterexample found at each step is then used to further refine the parameter set and the procedure terminates when no other counterexamples are found. In general, all these methods, when working directly with raw data are potentially vulnerable to the noise of the measurements and are limited by the amount of data available.

Learning both the structure and the parameters of a formula from a data set poses even more challenges [KJB17, BVP+16, BBS14a, BBS+14b].This problem is usually addressed in two steps, learning the structure of the formula and synthesizing its parameters. In particular, in [KJB17] the structure of the formula is learned by exploring a directed acyclic graph and the method exploits *Support Vector Machine* (SVM) for the parameter optimization. In [BVP+16] the authors use instead a *decision tree* based approach for learning the formula, while the optimality is evaluated using heuristic impurity measures. In [BBS14a] the structure of the formula is learned using a heuristics algorithm, while in [BBS+14b] using a genetic algorithm. The synthesis of the parameters is instead performed in both cases using the *Gaussian Process Upper Confidence Bound* (GP-UCB) algorithm. GP-UCB provides a statistical emulation of the satisfaction probability (and not of the average robustness as in this chapter) of a formula for a given set of parameters. Both these methods require learning first a statistical model from the training set of trajectories.

Contribution. The approach presented in this chapter instead is based on the ideas of [BBS+14b, BBS14a], but does not require to learn a statistical model of the training set of trajectories. Furthermore, we introduce some techniques to improve the performance and to deal with the noise of the data, an even more important issue in the absence of an underlying model.

First, instead of using the probability satisfaction as an evaluator for the best formula, we consider the average robustness (Section 2.5) which enables us to differentiate among STL classifiers that have similar classification performance with respect to the data by choosing the most robust one. This gives us more information than just having the probability of satisfaction: for each trajectory, we can evaluate how much is it close to the "boundary" of the classifier, instead of only checking whether it satisfies or not a formula. We then modify the discrimination function and the GP-UCB algorithm (Subsection 4.1.1) used in [BBS14a] and [BBS+14b] to better deal with the noise of the data and to use the quantitative semantics to emulate the average robustness distribution with respect to the parameter space of the formula.

Second, we reduce the computational cost of the *Evolutionary Algorithm* (EA) (implemented in [BBS+14b]), by using a lightweight configuration (i.e., a low threshold of the max number of iterations) of the GP-UCB optimization algorithm to estimate the parameters of the formulae at each generation. The EA algorithm generates, as a final result, an STL formula tailored for classification purpose.

Finally, we compare our framework with the methodology of [BBS+14b] and with the decision-tree based approach presented in [BVP+16] on an anomalous trajectory detection problem of naval surveillance.

6.1 Problem Formulation

Let us start with the definition of the parametric signal temporal logic which we use in the rest of this chapter.

PSTL *Parametric Signal Temporal Logic* [ADMN11] is an extension of STL that parametrizes the formulae. There are two types of formula parameters: temporal parameters, corresponding to the time bounds in the time intervals associated with temporal operators (e.g., $a, b \in \mathbb{R}_0^+$, with $a < b$, s.t. $\mathbf{F}_{[a,b]}\mu$) and the threshold parameters corresponding to the constants in the inequality predicates (e.g., $k \in \mathbb{R}, \mu = x_i > k$, where x_i is a variable of the trajectory). In this work, we allow only atomic propositions of the form $\mu = x_i \bowtie k$ with $\bowtie \in \{>, \leq\}$. Given an STL formula φ, let $\mathbb{K} = (\mathcal{T} \times \mathcal{C})$ be the *parameter space*, where $\mathcal{T} \subseteq (\mathbb{R}_0^+)^{n_t}$ is the temporal parameter space (n_t being the number of temporal parameters), and $\mathcal{C} \subseteq \mathbb{R}^{n_k}$ is the threshold parameter space (n_k being the number of threshold parameters). A $\boldsymbol{\vartheta} \in \mathbb{K}$ is a parameter configuration that induces a corresponding STL formula; e.g., $\varphi = \mathbf{F}_{[a,b]}(x_i > k), \boldsymbol{\vartheta} = (0, 2, 3.5)$ then $\varphi_{\boldsymbol{\vartheta}} = \mathbf{F}_{[0,2]}(x_i > 3.5)$.

We focus our attention on learning the best property (or set of properties) that discriminates trajectories belonging to two different classes, say good and bad, starting from a labeled data set of observed trajectories. Essentially, we want to tackle a supervised two-class classification problem over paths, by learning a temporal logic discriminant, describing the temporal patterns that better separate two sets of observed paths.

The idea behind this approach is that there exists an unknown procedure that, given a temporal trajectory, is able to decide if the signal is good or bad. This procedure can correspond to many different things, e.g., to the reason of failure of a sensor that breaks

when it receives specific inputs. Our task is to approximate this unknown procedure with an STL monitoring algorithm. In general, as there may not be an STL formula that perfectly explains/mimics the unknown process, our task is to find the most effective one.

The approach we present here works directly with observed data and avoids the reconstruction of an intermediate generative model $p(\mathbf{x}|c)$ of trajectories $\mathbf{x}$ conditioned on their class c, as in [BBS+14b, BBS14a]. The reason is that such models can be hard to construct, even if they provide a powerful regularization, as they enable the generation of an arbitrary number of samples to train the logic classifier.

In a purely data-driven setting, to build an effective classifier, we need to deal with the fact that training data is available in limited amounts and it is typically noisy. This reflects the necessity of finding methods that guarantee good generalization performance and avoid overfitting. In our context, overfitting can result in overly complex formulae, de facto encoding the training set itself rather than guessing the underlying patterns that separate the trajectories. This can be faced by using a score function based on the robustness of temporal properties, combined with suitably designed regularizing terms.

We want to stress that the approach we present here, due to the use of the average robustness of STL properties, can be easily tailored to different problems, like apprenticeship learning, i.e., finding the property that best characterizes a single set of observations.

6.2 Methodology

Learning an STL formula can be separated into two optimization problems: the learning of the formula structure and the synthesis of the formula parameters. The structural learning is treated as a discrete optimization problem using *Genetic Algorithm* (GA); the parameter synthesis, instead, considers a continuous parameter space and exploits an active learning algorithm called *Gaussian Process Upper Confidence Bound (GP-UCB)* algorithm. The two techniques are not used separately but combined together in a bi-level optimization. The GA act externally by defining a set of template formulae. Then, the GP-UCB algorithm, which acts at the inner level, finds the best parameter configuration such that each template formula better discriminates between the two data sets. To apply this second optimization we need to define a score function to optimize, encoding the criterion to discriminate between the two data sets.

Our implementation, called *RObustness GEnetic* (ROGE) algorithm is described in Algorithm 5. First, we give an overview of it and then we described each specific function in the next subsections. The algorithm requires as input the data set $\mathcal{D}_p(\text{good})$ and $\mathcal{D}_n(\text{bad})$, the parameter space $\mathbb{K}$, with the bounds of each formula parameters, the size of the initial set of formulae Ne, the number of iterations Ng, the mutation probability α and the initial formula size s.

line 1) It starts generating an initial set of PSTL formulae, called *gen*.

line 2) For each of these formulae, the algorithm learns the best parameters accordingly to a discrimination function G. We call gen_Θ the generation for which we know the best parameters of each formula.

line 3) The algorithm starts the iterations.

line 4) The algorithm, guided by a fitness function F, extracts a subset $subg_\Theta$ composed by the best $Ne/2$ formulae, from the initial set gen_Θ. The formulae are extracted proportionally to their fitness values (e.g., a formula with a double F-value w.r.t to another one, has a doubled probability to be sampled).

line 5) From this subset, a new set $newg$ composed of N_e formulae is created by employing the EVOLVE routine, which implements two *genetic* operators.

line 6) As in line 2, the algorithm identifies the best parameters of each formula belonging to $newg$.

line 7) The new generation $newg_\Theta$ and the old generation gen_Θ are then merged together. From this set of $2Ne$ formulae the algorithm extracts, with respect to the fitness function F, the new generation gen_Θ of Ne formulae.

line 9) At the end of the iterations, the algorithm returns the last generated formulae. The best formula is the one with the highest value of the discrimination function, i.e., $\varphi_{best} = \text{argmax}_{\varphi_\vartheta \in gen_\Theta}(G(\varphi_\vartheta))$.

We describe below in order: the discrimination function, the learning of the formula parameters and the learning of the formula structure.

6.2.1 Discrimination Function

We have two data sets $\mathcal{D}_p$ and $\mathcal{D}_n$ and we search for the formula φ that better separates these two classes. We define a function able to discriminate between this two data sets, such that maximizing this *discrimination function* corresponds to finding the best formula classifier. We decide to use, as evaluation of satisfaction of each formula, the quantitative semantics of STL. As described in Subsection 2.1.1, this semantics returns a real-value of satisfaction instead of only a yes/no answer.

Given a data set $\mathcal{D}_i$, we assume that the data comes from an unknown stochastic process $\mathcal{M}$. The process, in this case, is like a black box for which we observe only a subset of trajectories, the data set $\mathcal{D}_i$. We can then evaluate the averages robustness $\mathbb{E}(R_\varphi|\mathcal{M})$ and its variance $\sigma^2(R_\varphi|\mathcal{M})$, averaging over $\mathcal{D}_i$, (see Section 2.5).

To discriminate between the two data sets $\mathcal{D}_p$ and $\mathcal{D}_n$, we search the formula φ that maximizes the difference between the average robustness of $\mathcal{M}_p$, $\mathbb{E}(R_\varphi|\mathcal{M}_p)$, and the average robustness of $\mathcal{M}_n$, $\mathbb{E}(R_\varphi|\mathcal{M}_n)$ divided by the sum of the respective standard deviation.

$$G(\varphi) = \frac{\mathbb{E}(R_\varphi|\mathcal{M}_p) - \mathbb{E}(R_\varphi|\mathcal{M}_n)}{\sigma(R_\varphi|\mathcal{M}_p) + \sigma(R_\varphi|\mathcal{M}_n)} \tag{6.1}$$

This formula is proportional to the probability that a new point sampled from the same distribution of points generating $\mathcal{M}_p$ or $\mathcal{M}_n$, will belong to one set and not to the other. In fact, an higher value of $G(\varphi)$ implies that the two average robustness scores will be sufficiently far away, compared to their length-scale, given by the standard deviation σ.

As said above, we can evaluate just a statistical approximation of $G(\varphi)$ because we are working with just evaluations of the unknown functions $\mathbb{E}(R_\varphi|\mathcal{M})$. We will see in the next paragraph how we tackle this problem.

6.2.2 GP-UCB: learning the parameters of the formula

Given a formula φ and a parameter space $\mathbb{K}$, we want to find the parameter configuration $\boldsymbol{\vartheta} \in \mathbb{K}$ that maximises the score function $g(\boldsymbol{\vartheta}) := G(\varphi_{\boldsymbol{\vartheta}})$, considering that we have only noise and costly evaluation of it. The question is therefore how to best estimate (and optimize) an unknown function from observations of its value at a finite set of input points. This is a classic non-linear non-convex optimization problem that we tackle by means of the GP-UCB algorithm (Subsection 4.1.1). This algorithm interpolates the noisy observations using a stochastic process (a procedure called emulation in statistics) and optimize the emulation function using the uncertainty of the fit to determine regions where the true maximum can lie. More specifically, the GP-UCB bases its emulation phase on Gaussian Processes (GP), a Bayesian non-parametric regression approach, adding candidate maximum points to the training set of the GP in an iterative fashion, and terminating when no improvement is possible (see Section 4.1 and [SKKS12] for more details). After this optimization, we have found a formula that separates the two data sets, not from the point of view of the satisfaction (yes/no) of the formula but from the point of view of the robustness value. In other words, there is a threshold value α such that $\mathbb{E}(R_\varphi|\mathcal{M}_p) > \alpha$ and $\mathbb{E}(R_\varphi|\mathcal{M}_n) \leq \alpha$. Now, we consider the new STL formula φ' obtained translating the atomic predicates of φ by α (e.g., $y(x) > 0$ becomes $y(x) - \alpha > 0$). Taking into account that the quantitative robustness is achieved by combination of min, max, inf and sup, which are linear algebraic operators w.r.t translations (e.g, $\min(x, y) \pm c = \min(x \pm c, y \pm c)$), we easily obtain that $\mathbb{E}(R_{\varphi'}|\mathcal{M}_p) > 0$ and $\mathbb{E}(R_{\varphi'}|\mathcal{M}_n) < 0$. Therefore, considering the soundness of the robustness semantics of STL, we have that φ' divides the two data set also from the point of view of the satisfaction.

We remark that the use of the robustness degree of STL hampers to use the minimal representation of the formula. Indeed, even if the minimal representation has the same Boolean satisfaction it can have different robustness values.

6.2.3 Genetic Algorithm: learning the structure of the formula

To learn the formula structure, we exploit a modified version of the Genetic Algorithm (GA) presented in [BBS+14b]. In Section 4.2 we provide an overview of this approach. Genetic Algorithms belong to the larger class of evolutionary algorithms (EA). They are used for search and optimization problems. The strategy takes inspiration from the genetic area, in particular in the selection strategy of species. Let us see now in detail the genetic routines of the ROGE algorithm.

$gen = \textsc{generateInitialFormulae}(Ne, s)$**.** This routine generates the initial set of Ne formulae. A fraction $n_l < Ne$ of this initial set is constructed by a subset of the temporal properties: $\mathbf{F}_I\mu$, $\mathbf{G}_I\mu$, $\mu_1\mathbf{U}_I\mu_2$, where the atomic predicates are of the form $\mu = (x_i > k)$ or $\mu = (x_i \leq k)$ or simple Boolean combinations of them (e.g., $\mu = (x_i > k_i) \wedge (x_j > k_j)$). The size of this initial set is exponential w.r.t the number of considered variables x_i. If we have few variables we can keep all the generated formulae, whereas if the number of variables is large we consider only a random subset. The remain $n_r = Ne - n_l$ formulae are chosen randomly from the set of formulae with a maximum size defined by the input parameter s. This size can be adapted to have a

Algorithm 5 ROGE – RObustness GEnetic Algorithm

Require: $\mathcal{D}_p, \mathcal{D}_n, \mathbb{K}, Ne, Ng, \alpha, s$
1: $gen \leftarrow$ GENERATEINITIALFORMULAE(Ne, s)
2: $gen_\Theta \leftarrow$ LEARNINGPARAMETERS$(gen, G, \mathbb{K})$
3: **for** $i = 1 \dots Ng$ **do**
4: $subg_\Theta \leftarrow$ SAMPLE(gen_Θ, F)
5: $newg \leftarrow$ EVOLVE$(subg_\Theta, \alpha)$
6: $newg_\Theta \leftarrow$ LEARNINGPARAMETERS$(newg, G, \mathbb{K})$
7: $gen_\Theta \leftarrow$ SAMPLE$(newg_\Theta \cup gen_\Theta, F)$
8: **end for**
9: **return** gen_Θ

wider range of initial formulae.

$subg_\Theta = \textbf{SAMPLE}(gen_\Theta, F)$**.** This procedure extracts from gen_Θ a subset $subg_\Theta$ of $Ne/2$ formulae, accordingly to a fitness function $F(\varphi) = G(\varphi) - S(\varphi)$. The first factor $G(\varphi)$ is the dicrimination function previously defined and $S(\varphi)$ is a size penalty, i.e., a function penalizing formulae with large size. In this contribution, we consider $S(\varphi) = g \cdot p^{size(\varphi)}$, where p is heuristically set such that $p^5 = 0.5$, i.e., formulae of size 5 get a 50% penalty, and g is adaptively computed as the average discrimination in the current generation. An alternative choice of p can be done by cross-validation.

$newg = \textbf{EVOLVE}(subg_\Theta, \alpha)$**.** This routine defines a new set of formulae ($newg$) starting from $subg_\Theta$ by exploiting two *genetic* operators: the *recombination* and the *mutation* operator. The recombination operator takes as input the tree structure of two formulae φ_1, φ_2 (the parents), it randomly chooses a subtree from each formula and it swaps them, i.e., it assigns the subtree of φ_1 to φ_2 and viceversa. As a result, the two generated formulae (the children) share different subtrees of the parents' formulae. The mutation operator takes as input a formula and induces small randomized changes (such as inequality flips, temporal operator substitution, etc.) on a randomly selected node of its tree-structure. The purpose of the genetic operators is to introduce innovation in the offspring population which leads to the optimization of a target function (G in this case). In particular, the recombination is related to the exploitation, whereas the mutation to the exploration.

The EVOLVE routine implements an iterative loop that at each iteration selects which genetic operators to perform. A number $p \in [0, 1]$ is randomly sampled. If its value is lower than the mutation probability, i.e., $p \leq \alpha$, then the mutation is selected, otherwise the recombination. At this point the input formulae of the selected genetic operator are chosen randomly between the formulas composing $subg_\Theta$ and the genetic operations is performed. The iteration loops will stop when the number of generated formula is the double of the cardinality of $subg_\Theta$.

In our implementation, we give more importance to the exploitation; therefore the mutation acts less frequently than the recombination (i.e., $\alpha \leq 0.1$).

6.2.4 Regularization

In the evolutionary algorithm, we use two strategies to penalize complex formulae and bias the algorithm towards simple ones. The first strategy is to use a size penalty $S(\varphi)$ in the fitness function: $F(\varphi) = G(\varphi) - S(\varphi)$. In this work, we consider $S(\varphi) = g \cdot p^{size(\varphi)}$, where p is heuristically set such that $p^5 = 0.5$, i.e., formulae of size 5 get a 50% penalty, and g is adaptively computed as the average discrimination in the current generation. An alternative choice of p can be done by cross-validation. The second strategy, instead, consists in selecting the initial population biasing it towards simple formulae. In particular, this set is constructed by (a) a subset of random formulae and (b) a subset of logically non-equivalent temporal properties of size 1, of the form $\mathbf{F}_I\mu$, $\mathbf{G}_I\mu$, $\mu_1\mathbf{U}_I\mu_2$, where the atomic predicates μ are sampled from the subset of all atomic predicates (with threshold zero) or simple Boolean combinations of them.

6.3 Case Study: Maritime Surveillance

We consider the maritime surveillance case study presented in [BVP+16] to compare our framework with their *Decision Tree* (DTL4STL) approach. The experiments with the DTL4STL approach were implemented in Matlab with the code available at [DTL16]. We also test the implementation presented in [BBS+14b] with the same benchmark. Both the new an the previous learning procedure were implemented in JAVA (JDK 1.8_0) and run on a Dell XPS, Windows 10 Pro, Intel Core i7-7700HQ 2.8 GHz, 8GB 1600 MHz memory.

The synthetic data set of naval surveillance reported in [BVP+16] consists of 2-dimensional coordinates traces of vessels behaviors. It considers two kinds of anomalous trajectories and regular trajectories, as illustrated in Figure 6.1. The data set contains 2000 total trajectories (1000 normal and 1000 anomalous) with 61 sample points per trace. We fixed the training set as the 80% of the entire data set and the validation set as the rest 20% of the traces.

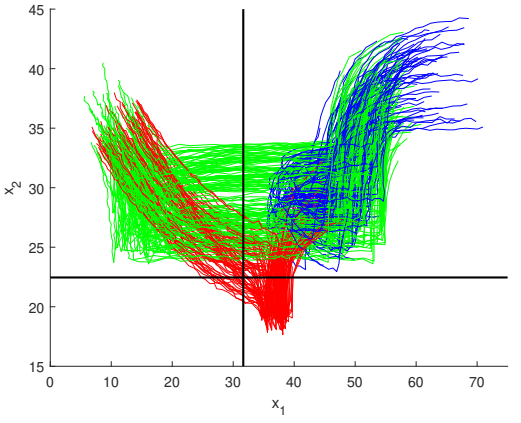


Figure 6.1: The regular trajectories are represented in green. The anomalous trajectories which are of two kinds, are represented respectively in blue and red.

	ROGE	DTL4STL	DTL4STL$_p$
Mis. Rate	0	0.01 ± 0.01	0.007 ± 0.008
Comp. Time	73 ± 18	144 ± 24	-

Table 6.1: The comparison of the computational time (in seconds), the mean miss-classification rate and its standard deviation between the learning procedure using the RObust GEnetic algorithm, the Decision-Tree (DTL4STL) Matlab code provided by the authors and the results reported in [BVP+16] (DTL4STL$_p$).

We run the experiments using a 10-fold cross-validation to collect the mean and variance of the misclassified trajectories of the validation set. The implementation presented in [BBS+14b] performs so poorly on the chosen benchmark that is not meaningful to report it: the misclassification rate on the validation set is around 50%.

The other obtained performances are presented in Table 6.1 as well as the average and variance of the execution time. We also report the DTL4STL performance (DTL4STL$_p$ in Table 6.1) declared in [BVP+16], but we were not be able to reproduce them in our setting.

An example of the formula that we learn with ROGE is

$$\varphi = (x_2 > 22.46)\, \mathbf{U}_{[49,287]}\, (x_1 \leq 31.65) \tag{6.2}$$

DTL4STL instead does not consider the until operator in the set of primitives (see [BVP+16] for more details) and the formula found by the tool using the same data set is the following:

$$\begin{aligned}\psi = (((&\mathbf{G}_{[187,196)} x_1 < 19.8) \wedge (\mathbf{F}_{[55.3,298)} x_1 > 40.8)) \vee \\ ((&\mathbf{F}_{[187,196)} x_1 > 19.8) \wedge ((\mathbf{G}_{[94.9,296)} x_2 < 32.2) \vee \\ ((&\mathbf{F}_{[94.9,296)} x_2 > 32.2) \wedge (((\mathbf{G}_{[50.2,274)} x_2 > 29.6) \wedge \\ &(\mathbf{G}_{[125,222)} x_1 < 47)) \vee ((\mathbf{F}_{[50.2,274)} x_2 < 29.6) \wedge \\ &(\mathbf{G}_{[206,233)} x_1 < 16.7))))))\end{aligned}$$

The specific formula generated using ROGE is simpler than the formula generated using DTL4STL and indeed it is easier to understand it. More specifically, the formula 6.2 identifies all the regular trajectories, remarkably resulting in a misclassification error equal to zero, as reported in Table 6.1. The red anomalous trajectories falsify the predicate $x_2 > 22.46$ before verifying $x_1 \leq 31.65$, on the contrary the blue anomalous trajectories globally satisfy $x_2 > 22.46$ but never verify $x_1 \leq 31.65$ (consider that all the vessels start from the top right part of the graphic in Figure 6.1). Both these conditions result in the falsification of the formula 6.2, which on the contrary is satisfied by all the regular trajectories. In Figure 6.1, we have reported the threshold $x_2 = 22.46$ and $x_1 = 31.65$. In terms of accuracy, our approach is comparable w.r.t. the performance of the DTL4STL approach shown in [BVP+16]. Similarly to other EA based approach, ROGE needs to be configured in some of its parameters (such as the mutation, the recombination probability, the fitness function and the initial population) meaning that

a specific setting can influence its performance.

6.4 Conclusion and Future Works

We present a framework to learn a Signal Temporal Logic (STL) specification from a labeled data set of regular and anomalous trajectories that better discriminates them in two subsets. In particular, we design a *RObust GEnetic* algorithm (ROGE) that combines an evolutionary algorithm for learning of the structure of the formula and the Gaussian process upper confidence bound algorithm for the synthesis of the formula parameters. We compare ROGE with [BBS+14b] and the decision tree approach presented in [BVP+16] on an anomalous trajectory detection problem of a maritime surveillance system.

With respect to the previous work [BBS14a, BBS+14b], we avoid the reconstruction of a generative statistical model for the data set and present a procedure that works well also directly with the data set. Furthermore, to improve the quality of the learning procedure, we modify both the structure and the parameters of the optimization algorithms. Concerning the learning of the structure, we modify the evolutionary algorithm drastically improving its performance, in particular, we change the policy to choose the initial formula generation and we simplify the genetic operators of mutation and recombination. Concerning the learning of the parameters, we leverage the average robustness (using the STL quantitative semantics) to discriminate also between two trajectories satisfying the same STL specification, but with two distinct robustness values that can be separated by a threshold. This provides more information that can be exploited in the optimization process.

We observe that the application of the approach [BBS+14b] directly on the data set of the naval surveillance system considered here, performs very poorly. We compare our method also with the Decision Tree (DTL4STL) approach of [BVP+16] showing that we have a comparable accuracy producing smaller and easier to understand STL specifications. Furthermore, we do not restrict the class of the temporal formula to only *eventually* and *globally* and we do not impose only one possible temporal nesting. On the other hand, the genetic algorithm can get completely wrong results if the initial formula generation is chosen completely randomly. Note that our choice of initial formulae is a way to bias the search towards simple properties, i.e., it is a form of regularization and resembles the choice of the set of primitive in the DTL4STL approach.

As future work, we aim to improve the genetic algorithm avoiding the optimization of similar formulae. We plan to exploit a Bootstrap aggregating (i.e., Bagging) technique to face the overfitting problem and guarantee good generalization performance. The idea is to generate new subsets from the initial one, through a uniform sampling with replacement, and apply the genetic algorithm to each subset. This will decrease the effect of possible biases in the initial distribution of the data. Finally, we plan also to test our procedure in more interesting case studies, particularly in the presence of strong noise in the data sets, where we believe our method can exhibit good results.

7

Parameter Synthesis

Stochastic models are used to describe and simulate a wide range of dynamical systems, such as biological processes, chemical reactions, and networks dynamics. The most successful representations is this scenarios are Chemical Reaction Networks (CRN) (Definition 4, Section 1.2), stochastic Petri nets [Haa04] and population models [BHLM13b]. All these representations underline a CTMC which under reasonable hypothesis well approximate the modeled dynamical systems. For example, a chemical reaction can always be represented as a CRN, but the underlying CTMC is a reliable approximation only if the well-stirred hypothesis is verified.

A typical model checking problem consists in estimating the following probability

$$P_{\varphi}(\mathcal{M}) = \mathcal{P}_{\mathcal{M}}(Sat(\varphi))$$

where φ is a formula in a suitable temporal logic and $Sat(\varphi) = \{\mathbf{x} \in \mathcal{S}^{\mathbb{T}} \mid (\mathbf{x}, 0) \models \varphi\}$, as already defined in Section 2.4 . This is a typical *a posteriori* problem, where a model is given, and some qualitative and/or quantitative information (described by means of logics) are extracted. As already discuss in Section 2.4, there are two main approaches to solve this issue: analytical methods which compute multidimensional integrals and statistical methods which simulate a model of a system many times. The former method relies on the assumption that the parameter values of the model are known, whereas the latter on the availability of a computation model able to draw independent and identically distributed trajectories. These are strong assumptions which are not satisfied in specific contexts such as model-based design and system biology. In the early designing process, for example, not all the parameters are fixed, but only intervals (defined by means of expertise) are considered. Moreover, in system biology, many parameter values are not even known because estimating them is highly costly or cannot be measured. These two problems have different goals. The first problem, called *Parameters Synthesis*, tries to identify a set of model parameters such that a given property is verified or a given probability maximized. On the contrary, the second problem, called *System Identification* follows a reverse direction. It considers the satisfiability of a set of different formulae against a target systems with the aim of identifying a neighborhood

of the parameter model (which well approximate that target system). In this chapter, we describe a statistical approach which relies on Smoothed Model Checking (smMC) (Section 3.2) to solve the parameter synthesis problem that we define in a particular setting.

Related work. Parameter synthesis of CTMC is an active field of research. In [CDKP14] and [CDP+17] the authors use Continuous Stochastic Logic (CSL) and uniformization methods for computing exact probability bounds for parametric models of CTMCs obtained from chemical reaction networks. In [CPP+16] the same authors extend their algorithm to GPU architecture to improve the scalability. Authors in these two papers solve two problems: one is the threshold synthesis, the other is the identification of a parameter configuration maximizing the satisfaction probability. In this chapter, we focus on the former, as we already presented a statistical approach to deal the latter problem in [BBNS15] for the single objective case and in [BPS16] for the multi-objective case (see also Chapter 8). An alternative statistical approach for multi-objective optimization is that of [DDL+13], where authors use ANOVA test to estimate the dominance relation. Another approach to parameter synthesis for CTMC is [HKM08], where the authors rely on a combination of discretization of parameters with a refinement technique.

In this work we use a statistical approach to approximate the satisfaction probability function, building on smoothed model checking [BMS16]. This approach applies to CTMC with rate functions that are smooth with respect to parameters and leverages statistical tools based on Gaussian process regression [RW06] to learn an approximation of the satisfaction function from few observations. Moreover, this approach allows us to deal with a richer class of linear time properties than reachability, like those described by metric temporal logic [AFH96, MN04], for which numerical verification routines are heavily suffering from state space explosion [BCH+11]. Another statistical approach is that of [JL11], which combines sensitivity analysis, statistical model checking and uniform continuity to approximate the satisfaction probability function, but it is restricted to cases when the satisfaction probability is monotonic in the parameters. In contrast, Gaussian process-based methods have no restriction (as Gaussian processes are universal approximators), and also have the advantage of requiring much fewer simulations than pointwise statistical model checking, as information is shared between neighboring points (see [BMS16] for a discussion in this sense). Parametric verification and synthesis approaches are more consolidated for discrete time Markov chains [Kat16], where mature tools like PROPhESY exist [DJJ+15], which rely on a symbolic representation of the reachability probability, which does not generalize to the continuous time setting.

Contribution. We propose a Bayesian statistical approach for parameter synthesis, which leverages a statistical parameterised verification method known as smoothed model checking [BMS16] (Section 3.2) and the nice theoretical approximation properties of Gaussian process [RW06] (Section 3.1). Being based on a Bayesian inference engine, this naturally gives statistical error bounds for the estimated probabilities. Our algorithm uses active learning strategies to steer the exploration of the parameter space only where the satisfaction probability is close to the threshold. We also provide a prototype implementation of the approach in Python.

Despite being implemented in Python, our approach turns to be remarkably efficient, being slightly faster than [CDP+17] for small models, and outperforming it for more complex and large models or when the number of parameters is increased, at the price of a weaker form of correctness. Compared to [CDP+17], we also have an additional advantage: the method treats the simulation engine and the routine to verify the linear time specification on individual trajectories as black boxes. This means that, not only, we can handle arbitrary MTL properties (while in [CDP+17] there is an essential restriction to non-nested CSL properties, i.e., reachability), but also other more complicated linear time specifications (e.g., using hybrid automata, providing the satisfaction probability is a smooth function of model parameters). Moreover, we can also apply the same approach to more complex stochastic models for which efficient simulation routines exist, like stochastic differential equations.

7.1 Problem Formulation

Statistical Threshold Synthesis Problem. We consider a family $\mathcal{M}_\vartheta$ of CTMC indexed by continuous parameters ϑ which assume values in a compact hyperrectangle $\Theta \subset \mathbb{R}^k$. We also assume that $\mathcal{M}_\vartheta$ depend smoothly on ϑ through their rate functions and consider a linear time specifications φ described by Metric Interval Temporal Logic [MN04], with bounded time operators. Our goal is to find the subset of parameters $\Gamma_\alpha \subset \Theta$ such that $P_\varphi(\vartheta) \geq \alpha$ for a threshold α, where $P_\varphi(\vartheta)$ represents the probability that $\mathcal{M}_\vartheta$ satisfies φ, see Section 2.4. We introduce *the positive class* $\mathcal{P}_\alpha$, which is composed of parameter values for which the probability of satisfying φ is higher than a threshold value α, *the negative class* $\mathcal{N}_\alpha$, composed of parameters values for which this probability is lower than α, and *the undefined class* $\mathcal{U}_\alpha$, which collects all the other parameter values. This class represents the region where it is not possible to decide if $P_\varphi(\vartheta) \geq \alpha$ or $P_\varphi(\vartheta) \leq \alpha$. Following [CDP+17], we will look for a partition where the volume of the undefined class is lower a fraction of the volume of Θ. This problem is called *threshold synthesis problem* and it was originally introduced and solved by analytical technique in [CDP+17]. This contribution is based on the min max approximation where the idea is to consider a subset $H \subseteq \Theta$ and estimating, with a variable degree of accuracy, the minimum and the maximum of $P_\varphi(\vartheta)$. For example if we estimate that

$$[\min_{\vartheta \in H} P_\varphi(\vartheta), \max_{\vartheta \in H} P_\varphi(\vartheta)] \subseteq [\lambda^H_{\min}, \lambda^H_{\max}]$$

we can easily decide if H belongs to $\mathcal{P}_\alpha$ or $\mathcal{N}_\alpha$ just by considering if $\lambda^H_{\min} > \alpha$ or $\lambda^H_{\max} < \alpha$ respectively. Otherwise H is divided into two subsets H_0 and H_1 and the algorithm is iterated over them. This procedure stops when the volume of $\mathcal{U}_\alpha$ is below a given threshold. The converge is assured by the sub-linearity of the min and the max.

In Section 2.4 we already discussed the drawbacks of the analytical methods. In this case, the same considerations apply, yet the possibility of parallelizing the algorithm mitigates them.

Our approach, on the other hand, will be statistic: we assume that models are too complex to numerically compute bounds on the reachability probability, and we only rely on the possibility of simulating the model. As a consequence, our solution to the parameter synthesis problem will have only statistical guarantees of being correct. For

example, if a parameter belongs to $\mathcal{P}_\alpha$, the confidence of this point satisfying $P_\varphi(\vartheta) \geq \alpha$ will be larger than a prescribed probability (typically 95% or 99%), though for most points this probability will be essentially one, and similarly for $\mathcal{N}_\alpha$. The challenge of such an approach is that estimating the satisfaction probability at many different points in the parameter space by simulation is very expensive and inefficient, unless we are able to share the information carried by simulation runs at neighboring points in the parameter space.

Bayesian Approach. We start by rephrasing the parameter synthesis problem defined in [CDP+17] in the context of Bayesian statistics, where truths are quantified probabilistically. The basic idea is that we will exhibit a set of parameters that satisfy the specification with high confidence, which in the Bayesian world means with high posterior probability. To recall and fix the notation, let $\mathcal{M}_\vartheta$ be a PCRN (Definition 4) defined over a parameter space Θ, φ a MITL formula and $\tilde{P}_\varphi(\vartheta)$ a statistical approximate model of the satisfaction probability of φ at each point ϑ. In the Bayesian setting, $\tilde{P}_\varphi(\vartheta)$ is in fact a posterior probability distribution over $[0,1]$, hence we can compute for each measurable set $B \subseteq [0,1]$ the probability $p(\tilde{P}_\varphi(\vartheta) \in B)$.

Definition 22 (Bayesian Threshold Synthesis Problem)**.** *Let $\mathcal{M}_\vartheta$, Θ, φ, and $\tilde{P}_\varphi(\vartheta)$ be defined as before. Fix a threshold α and consider the threshold inequality $P_\varphi(\vartheta) > \alpha$, for the true satisfaction probability $P_\varphi(\vartheta)$. Fix $\varepsilon > 0$ a volume tolerance, and $\delta \in (0.5, 1]$ a confidence threshold. The* Bayesian threshold synthesis problem *consists in partitioning the parameter space Θ in three classes $\mathcal{P}_\alpha$ (positive), $\mathcal{N}_\alpha$ (negative) and $\mathcal{U}_\alpha$ (undefined) as follows:*

- *for each $\vartheta \in \mathcal{P}_\alpha$, $p(\tilde{P}_\varphi(\vartheta) > \alpha) > \delta$*
- *for each $\vartheta \in \mathcal{N}_\alpha$, $p(\tilde{P}_\varphi(\vartheta) < \alpha) > \delta$*
- *$\mathcal{U}_\alpha = \Theta \setminus (\mathcal{P}_\alpha \cup \mathcal{N}_\alpha)$, and $\frac{vol(U)}{vol(\Theta)} < \varepsilon$, where vol is the volume of the set.*

Note that the set $\mathcal{P}_\alpha$ solves the threshold synthesis problem defined above, while $\mathcal{N}_\alpha$ solves the threshold synthesis problem $P_\varphi(\vartheta) < \alpha$.

7.2 Methodology

7.2.1 Bayesian Parameter Synthesis: the Algorithm

Our Bayesian synthesis algorithm essentially combines smoothed Model Checking (smMC) with an active learning step to adaptively refine the sets $\mathcal{P}_\alpha, \mathcal{N}_\alpha, \mathcal{U}_\alpha$, trying to keep the number of simulations of the Parametric Chemical Reaction Network (PCRN, Definition 4) $\mathcal{M}_\vartheta$ to a minimum. The smMC produces a Bayesian estimate of the satisfaction probability for each $\vartheta \in \Theta$. More specifically it produces a the posterior distribution $p(\tilde{P}_\varphi(\vartheta))$ which can be efficiently used to estimate the probability that $P_\varphi(\vartheta) \in [a, b]$. This valuable information is then used to compute the following two functions of ϑ:

- $\lambda^+(\vartheta, \delta)$ is such that $p\left(\tilde{P}_\varphi(\vartheta) < \lambda^+(\vartheta, \delta)\right) > \delta$

Algorithm 6 Bayesian Parameter Synthesis.

Require: Θ (parameter space), $\mathcal{M}$ (PCRN), φ (MTL formula), α (threshold), ε (volume precision), δ (confidence).

$\mathcal{S} \leftarrow$ initialSamples$(\Theta, \mathcal{M}, \varphi)$
$\mathcal{P}_\alpha \leftarrow \emptyset$, $\mathcal{N}_\alpha \leftarrow \emptyset$, $\mathcal{U}_\alpha \leftarrow \Theta$
while true **do**
  $\lambda^+, \lambda^- \leftarrow$ smoothedMC$(\Theta, \mathcal{S})$
  $\mathcal{P}_\alpha, \mathcal{N}_\alpha, \mathcal{U}_\alpha \leftarrow$ updateRegions$(\lambda^+, \lambda^-, \mathcal{P}_\alpha, \mathcal{N}_\alpha, \mathcal{U}_\alpha)$
  if $vol(\mathcal{U}_\alpha)/vol(\Theta) < \varepsilon$ **then**
    return $\mathcal{P}_\alpha, \mathcal{N}_\alpha, \mathcal{U}_\alpha$
  else
    $\mathcal{S} \leftarrow$ refineSamples$(\mathcal{S}, \mathcal{U}_\alpha)$
  end if
end while

- $\lambda^-(\vartheta, \delta)$ is such that $p\left(\tilde{P}_\varphi(\vartheta) > \lambda^-(\vartheta, \delta)\right) > \delta$

Essentially, at each point ϑ, $\lambda^+(\vartheta, \delta)$ is the upper bound for the estimate $\tilde{P}_\varphi(\vartheta)$ at confidence δ (i.e., with probability at least δ, the true value $P_\varphi(\vartheta)$ is less than λ^+), while $\lambda^-(\vartheta, \delta)$ is the lower bound. These two functions will be used to split the parameter space into the three regions $\mathcal{P}_\alpha, \mathcal{N}_\alpha, \mathcal{U}_\alpha$ as follows:

- $\vartheta \in \mathcal{P}_\alpha$ iff $\lambda^-(\vartheta, \delta) > \alpha$
- $\vartheta \in \mathcal{N}_\alpha$ iff $\lambda^+(\vartheta, \delta) < \alpha$
- $\mathcal{U}_\alpha = \Theta \setminus (\mathcal{P}_\alpha \cup \mathcal{N}_\alpha)$, $\frac{vol(U)}{vol(\Theta)} < \varepsilon$

To dig into how λ^+ and λ^- are computed, recall that smMC computes a real-valued Gaussian process $f_\varphi(\vartheta)$, with mean function μ and covariance function k, from which the pointwise standard deviation can be obtained as $\sigma(\vartheta) = \sqrt{k(\vartheta, \vartheta)}$. At each ϑ, the function $f_\varphi(\vartheta)$ is Gaussian distributed, hence we can compute the upper and lower confidence bounds for the Gaussian distribution, and then squeeze them into $[0, 1]$ by the probit transform Ψ. Letting $\beta_\delta = \Psi^{-1}(\frac{\delta+1}{2})$, as customary while working with normal distribution, we get:

- $\lambda^+(\vartheta, \delta) = \Psi\left(\mu(\tilde{f}_\varphi(\vartheta)) + \beta_\delta \sigma(\tilde{f}_\varphi(\vartheta))\right)$
- $\lambda^-(\vartheta, \delta) = \Psi\left(\mu(\tilde{f}_\varphi(\vartheta)) - \beta_\delta \sigma(\tilde{f}_\varphi(\vartheta))\right)$

Before describing the Algorithm 6, we introduce some notation for regular grids, as they are used in the current implementation of the method. Let us consider the hyper-rectangular parameter space $\Theta = \bigtimes_{i=1}^n [w_i^-, w_i^+] \subset \mathbb{R}^n$, where w_i^- and w_i^+ are respectively the lower and the upper bound of the domain of the parameter ϑ_i. An $\mathbf{h}$-grid of Θ is the set $\mathbf{h}\text{-grid} = \cup_{\mathbf{m} \in M}\{\mathbf{w}^- + \mathbf{m} * \mathbf{h}\}$ where $\mathbf{h} = \{h_1, \ldots, h_n\}$, $M = \bigtimes_{i=1}^n \{0, \ldots, \frac{w_i^+ - w_i^-}{h_i}\}$, $\mathbf{w}^- = (w_1^-, \ldots, w_n^-)$ and $*$ is the elementwise multiplication.

Given a grid, we define as *basic cell* a small hyperrectangle of size $\mathbf{h}$ whose vertices are consecutive points of the grid. Our idea is simply to discretize the continuous parameter space Θ by means of this grid and further use a subset of it to train a specific surrogate model, such as Gaussian processes. This surrogate model is then used to decide whether each basic cell of the grid to which class of the partition it belongs to.

Initialization. The initialization phase (line 1 and 2) consists in running simulations of the PCRN on all parameters of a coarse grid $\mathbf{h_0}$-grid, with $\mathbf{h_0}$ chosen such that the total number of parameters ϑ explored is reasonably small for smMC to be fast. More specifically we simulate N runs of the model per each point of this grid and pass them to a monitoring algorithm for MITL, obtaining N observations of the truth value of the property φ at each point of $\mathbf{h_0}$-grid, collected in the set $\mathcal{S}$. We also initialize the sets $\mathcal{P}_\alpha$, $\mathcal{N}_\alpha$, and $\mathcal{U}_\alpha$.

Computation of $\mathcal{P}_\alpha$, $\mathcal{N}_\alpha$, and $\mathcal{U}_\alpha$. The algorithm then enters the main loop, first running smMC with the current set of sample points $\mathcal{S}$ to compute the two functions λ^+ and λ^- (line 4). These are then used to update the regions $\mathcal{P}_\alpha$, $\mathcal{N}_\alpha$, and $\mathcal{U}_\alpha$. The main issue is to guarantee that each basic cell is assigned to a class and that the exit condition is satisfied (line 6). We present two possible approaches. The first approach considers directly the basic cell an tries to assign each of them to a class of the partition. The second approach is more flexible. It adapts the dimension of each cell in order to reduce the computational cost of the $\lambda^{+/-}$ functions.

Approach 1: fixed grid. The simplest approach is to partition the parameter space in small cells, i.e., using a $\mathbf{h}$-grid with $\mathbf{h}$ small, and then assign each cell to one of the sets. The assignment will be discussed later, but it involves evaluating the functions λ^+ and λ^- in each point of the grid. If the grid size is small, so that each basic cell contains only a fraction of the volume much smaller than ε, and if the dimension of the parameter space is not large (say up to 3 or 4), so that the grid does not contain too many points, then this method is fast. One advantage of this approach is that it is generally easier to decide if a small region belongs to set with respect to bigger one, the motivation is related to the continuity of $\lambda^{+/-}$ which was discussed in [BHLM13a].

Approach 2: adaptive grid. To scale with the dimension of the parameter space, we can start evaluating the $\lambda^{+/-}$ functions on a coarse grid, and refine the grid iteratively only for cells that are assigned to the uncertain set, until a minimum grid size is reached.

Central in both approaches is how to guarantee that all points of a basic cell are all belonging to one set, inspecting only a finite number of them. In particular, we will limit the evaluation of the $\lambda^{+/-}$ functions to the vertices of each cell, i.e., to the points in the grid $\mathbf{h}$-grid. Intuitively, this will work if the cell has a small edge size compared to the rate of growth of the satisfaction function, and the values of the satisfaction function in its vertices are all (sufficiently) above or below the threshold. However, we need to precisely quantify the word "sufficient". We sketch here one exact methods and a heuristic one, which performs well in practice. We discuss here how to check that a cell belongs to the positive set, the negative one being symmetric.

Lipschitz bound. This approach relies on computing the Lipschitz constant L of $\lambda^{+/-} = \Psi(\mu(\tilde{f}_\varphi(\vartheta)) \pm \beta_\delta \sigma(\tilde{f}_\varphi(\vartheta)))$. Considering its definition we noticed that $\lambda^{+/-}$ is derivable and therefore we can calculate the Lipschitz constant through the maximization of its derivative. Let $d(\mathbf{h})$ the length of the largest diagonal of a basic cell c in a $\mathbf{h}$-grid. Consider the smallest value of the satisfaction function in one of the vertices of c, and call it $\hat{p}$. Then the value of the satisfaction function in the cell is surely greater than $\hat{p} - Ld(\mathbf{h})/2$ (after decreasing for half the diagonal, we need to increase again to reach the value of another vertex). The test then is $\hat{p} - Ld(\mathbf{h})/2 \geq \alpha$.

Heuristic Method. In order to speed up computation and avoid computing Lipschitz constants, we can make the function λ^- more strict. Specifically, we can use a larger β_δ than the one required by our confidence level δ. For instance for a 95% confidence, $\beta_\delta = 1.96$, while we can use instead $\beta_\delta = 3$, corresponding roughly to a confidence of 99%. Coupling this with a choice of the grid step $\mathbf{h}$ at least one order of magnitude smaller than the lengthscale of the kernel learned from the data (which is proportional to the Lipschitz constant of the kernel and of the satisfaction function), which guarantees that the satisfaction function will vary very little in each cell, we can be confident that if the strict λ^- is above the threshold in all vertices of the cell, then the same will hold for all points inside c for the less strict λ^-.

Refinement step. After having built the sets $\mathcal{P}_\alpha$, $\mathcal{N}_\alpha$, and $\mathcal{U}_\alpha$, we check if the volume of $\mathcal{U}_\alpha$ is below the tolerance threshold. If so, we stop and return these sets. Otherwise, we need to increase the precision of the satisfaction function near the uncertain region. This means essentially reducing the variance inside $\mathcal{U}_\alpha$, which can be obtained by increasing the number of observations in this region. Hence, the refinement step samples points from the undefined regions $\mathcal{U}$, simulates the model few times in each of these points, computes the truth of φ for each trace, and adds these points to the training set $\mathcal{S}$ of the smoothed model checking process. This refinement will reduce the uncertainty bound in the undefined regions which leads some part of this region to be classified as Positive $\mathcal{P}$ or Negative $\mathcal{N}$. We iterate this process until the exit condition $\frac{vol(U)}{vol(\Theta)} < \varepsilon$ is satisfied. The convergence of the algorithm is rooted in the properties of smoothed model checking, which is guaranteed to converge to the true function with vanishing variance as the number of observation points goes to infinity. In practice, the method converges quite fastly, unless the problem is very hard (the true satisfaction function is close to the threshold for a large fraction of the parameter space).

7.3 Case Studies

We consider the popular SIR model previously introduced in Example 1, that we report here for simplicity.

$$r_1 : S + I \xrightarrow{\alpha_1} 2I \qquad \alpha_1 = k_{si} \cdot \frac{X_s \cdot X_i}{N}$$
$$r_2 : I \xrightarrow{\alpha_2} R \qquad \alpha_2 = k_{ir} \cdot X_i$$

The reaction r_1 represents the possibility that a healthy individual becomes infected,

Case	$k_i \times k_r$	**h**-grid	Time (sec)
1	$[0.005, 0.3] \times 0.05$	0.0007	17.92 ± 2.61
2	$0.12 \times [0.005, 0.2]$	0.0005	4.87 ± 0.01
3	$[0.005, 0.3] \times [0.005, 0.2]$	(0.003,0.002)	116.4 ± 4.06

Table 7.1: Results of the Statistical Parameter Synthesis for the SIR model with $N = 100$ individuals and formula $\varphi = (I > 0)\, \mathbf{U}_{[100,120]}\, (I = 0)$. We report the mean and standard deviation of the execution time of the algorithm. The volume tolerance is set to 10% and the threshold α is set to 0.1. The h-grid column shows the size $\mathbf{h}$ of the grid used to compute the positive, negative, and uncertain sets.

whereas the reactions r_2 models the recovery of an infected individual. We describe the models as a PCRN where $k_i \in [0.005, 0.3]$, $k_r \in [0.005, 0.2]$ and consider an initial population $(S, I, R) = (95, 5, 0)$. We are interested in estimating the parameters k_i, k_r such that the infection gets extinct between 100 and 120 units of time. This can be modeled with the following STL formula:

$$\varphi = (I > 0)\, \mathcal{U}_{[100,120]}\, (I = 0) \tag{7.1}$$

The parameters $\vartheta = (k_i, k_r)$ rules the timing of disease extinction which in this SIR formulation happens with probability 1. In the following, we report experiments to synthesize the parameter region such that $P_\varphi(\vartheta) > \alpha$, with $\alpha = 0.1$, volume tolerance $\varepsilon = 0.1$, and confidence $\delta = 95\%$. We achieve it in three different parameters settings, see the first two columns of Table 7.1.

The initial training set of the Algorithm initialization has been obtained by sampling point on a grid as described in Section 7.2.1, of size 40 points for 1D case and 400 points for the 2D case. The satisfaction probability of each parameter has been estimated by simulating and testing the STL formula with 1000 repetitions per parameter points. The tessellation for each case are shown in Figure 7.1.

We perform 3 kinds of analysis in order to measure Efficiency, Accuracy and Scalability.

Efficiency. The execution time of our approach which is reported in Table 7.1 shows good performance despite the algorithm have been implemented in Python rather than more efficient programming language such as C. More specifically, our implementation if compare with the exact method [CDP+17] shows an improvement of 42%, 18% and 7% for Case 1, Case 2 and Case 3.

Accuracy. We perform a statistical accuracy test which consists in computing a more precise approximation of the satisfaction probability by means of 10000 repetitions in each parameter close to the undefined region. The number of misclassified points is further evaluated. In Case 1 we consider 300 equally-spaced points between 0.01 and 0.07 (consider that a portion of the undefined region is located in a neighborhood of 0.05, see Figure 7.1a). All points turned to be classified correctly, pointing out to the accuracy of the smMC prediction.

Pop. Size	α	δ	Time (sec)
200	0.13	10%	$13,05 \pm 3,22$
400	0.08	10%	$13,86 \pm 5,99$
800	0.2	4%	$15,02 \pm 0,05$
1000	0.23	4%	$17,44 \pm 0,23$
2000	0.3	4%	$28,81 \pm 0,07$

Table 7.2: Scalability of the method w.r.t. the size of the state space of the SIR model, increasing initial population N. α and δ are the threshold and volume tolerance used in the experiments.

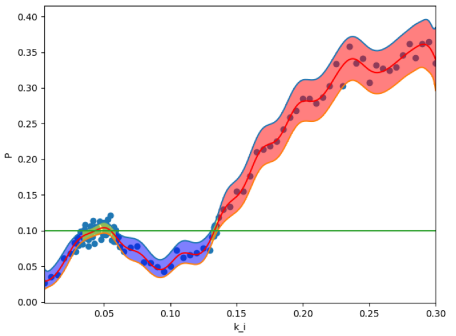
Scalability. As a scalability test we increase the size of the state space of the PCRN model and the initial population size N of the SIR model (case 1) by maintaining the original proportion $\frac{I}{S} = \frac{1}{19}$. Moreover we consider different thresholds α and volume tolerance ε in order to force the algorithm to execute at least one refinement step, as the shape of the satisfaction function changes with N. The execution time increases moderately, following a linear trend as shown in Table 7.2.

7.4 Conclusion and Future Works

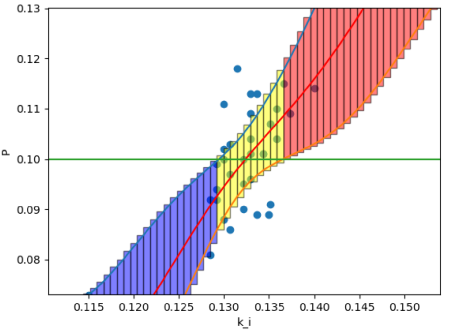
We present a statistical algorithm which is able to identify the region of the parameter space where the target STL formula is satisfied. This algorithm leverages the Smoothed Model Checking technique and an active learning strategy which pushes the algorithm effort toward the undefined region (i.e., the region where the satisfaction probability is near the threshold α). The results are good in term of the execution time and outperform the exact algorithm developed in [CDP+17], retaining good accuracy at the price of having only statistical guarantees of correctness.

Note that we compared with the performance of [CDP+17] and not of their Graphics Processing Unit (GPU) implementation [CPP+16], as our method uses only CPU computing power at the moment. However, it can be implemented on a GPU where we expect a substantial increase in the performance.

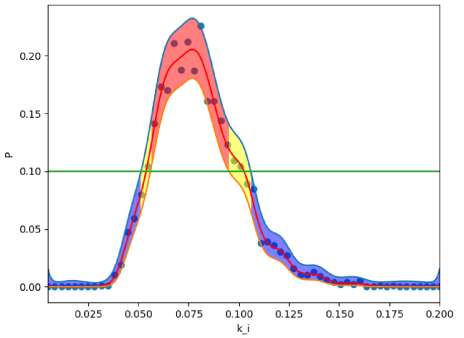
As future work, we can proceed in two main directions. The first consists in increasing the performance of our approach by improving the tessellation strategy. An adaptive strategy which divides and conquer to split the parameter space will reduce the cost of smMC. A similar outcome can be obtained by an improvement of the GP reconstruction method used by smMC. Here we can use classic sparsity methods [RW06] and more recent methods for GPs tailored to work on grids [WN15, WHSX16]. These last approaches have a computational cost of $O(n)$ instead of standard implementation which costs $O(n^3)$. The second direction consists in defining a hybrid approach which uses the beneficial aspects of both the exact and the statistical methods. The idea is to use our algorithm to produce a rough tessellation by cutting out the region of the parameters with higher statistical confidence to be not an undefined region. Further, the exact algorithm can be used in the remaining region to refine it. This combined approach will reduce the computation effort of the exact method, and at the same time will have an exact accuracy where it is needed.



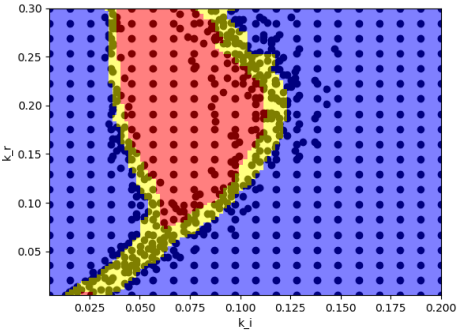
(a) Case 1



(b) Zoom on Case 1



(c) Case 2



(d) Case 3

Figure 7.1: **(a)**,**(c)** and **(d)** show the partition of the parameter space for Cases 1, 2, and 3 respectively. The positive area $\mathcal{P}_\alpha$ is depicted in red, the negative area $\mathcal{N}_\alpha$ is in blue and the undefined region $\mathcal{U}_\alpha$ is in yellow. **(a)** and **(c)** are one dimensional case: in the x-axis we report the parameter explored (respectively k_i and k_r), on the y-axis we show the value of the satisfaction function and the confidence bounds (for $\beta_\delta = 3$). The green horizontal line is the threshold $\alpha = 0.1$ **(d)** shows a two dimensional parameter space, hence no confidence bound has been represented. The circle dot represent the training set. In **(b)** we have zoomed a portion of the parameter space of **(a)** to visualize the cells with base length equals to h and height equal to the span of the confidence bounds.

8

System Design with logical objectives

In the Introduction, we have already discussed the importance of system design. In synthetic biology, where the experiments are costly and the analyzed processes are usually complex, a system design approach is widely applied. As typical in computer-aided engineering, this approach consists in building a model of the system of interest, in formalizing objectives to be achieved, in tuning controllable parameters to satisfy the desired goals as robustly as possible. In this chapter[1] we use the logical methods in a multi-objective fashion to perform system design of Chemical Reaction Networks (CRN) (Definition 4). The idea is to explore and discover parameters of the CRN which corresponds to the best trade-off between multiple requirements. We will analyze a specific case where the maximization of the satisfaction probability of two different formulae can be effectively solved in a multi-objective paradigm (Section 4.2). Moreover, we discuss how the average robustness of this formula can be effectively used to reach that goal but we also discuss a possible pitfall in the use of that robustness score, related to a difference in the scale of atomic propositions.

Related work. Multi-objective optimization techniques have been widely studied over the last fifteen years due to their increasing importance in many academic research fields (economics, operational research, stochastic control theory, etc.) as well in industrial research (see [CL04]). Several solution methods exist (e.g., evolutionary algorithms [Deb01, ZQL+11], gradient optimizer with restart, single weighted sum), mostly based on heuristic search. The multi-objective approaches to model checking verification have been described in [EKVY07, CMH06, FKP12]. The most common technique consists in transforming the original problem in a linear programming problem. The system design of stochastic models through the maximization of the robustness of temporal properties is a rather new field of research [BBNS15] and in particular the multi-objective approaches in this case have not been yet deeply explored.

[1] This contribution has been published in [BPS16].

8.1 Problem Formulation and Methodology

We consider the following system design problem:

> given a Parametric Reaction Network (PRN) $\mathcal{M}_\vartheta$ (Definition 4), with h tunable parameters $\vartheta = (\vartheta_1, \ldots, \vartheta_h) \in \Theta \subset \mathbb{R}^h$ and k STL formulae $\Phi = \{\varphi_1, \ldots, \varphi_k\}$, find a value $\vartheta^* \in \Theta$ of the parameters such that all formulae are satisfied "as much as possible" (i.e., their satisfaction probability is maximised).

There is obviously a problem here, as we have k formulae that can represent conflicting objectives. Hence it may be impossible to maximise the satisfaction probability of all formulae at the same time. Moreover, different formulae can play a different role: we need to solve a *multi-objective* optimization problem, that is we need to estimate the Pareto front (Section 4.2). Additionally, we have two semantics for our logic, so we can decide to optimize either the satisfaction probability $P_{\varphi_j}(\vartheta)$ or the expectation value of the robustness stochastic variable $R_{\varphi_j}(\vartheta)$, i.e., $\varrho_{\varphi_j}(\vartheta) = \mathbb{E}(R_{\varphi_j}(\vartheta))$. Note that both these quantities are *functions* of the model parameters $\vartheta \in \Theta$. We consider and compare three different approaches:

Direct probability Approach (DpA): solve directly the multi-objective problem associated to the maximization of the probability

$$P_\Phi(\vartheta) = (P_{\varphi_1}(\vartheta), P_{\varphi_2}(\vartheta), \ldots, P_{\varphi_k}(\vartheta))$$

Direct robustness Approach (DrA): solve the multi-objective problem associated to the maximization of the average robustness

$$\varrho_\Phi(\vartheta) = (\varrho_{\varphi_1}(\vartheta), \varrho_{\varphi_2}(\vartheta), \ldots, \varrho_{\varphi_k}(\vartheta))$$

Mixed Approach(MA): combine the two approaches automatically switching among them

Considering our goal, the DpA seems the natural choice. However, the quantitative semantics carries information about the satisfiability of a formula for each trace, plus additional information about robustness of satisfaction. As discussed in [BBNS15], when we average the robustness score over all trajectories, according to the distribution P_ϑ on trajectories, induced by a PRN $\mathcal{M}_\vartheta$, we typically obtain a score which is positively correlated with the satisfaction probability. This means that, as a function of ϑ, typically satisfaction probability increases when average robustness does (see for instance Figure 8.1).

Furthermore, the robustness score typically carries more information in regions of the parameter spaces in which the satisfaction probability is flat, e.g., equal to zero or to one. In some cases, in fact, it happens that the probability vector is higher than zero only in a very tiny zone of the parameters space (this will be the case, for instance, in the genetic toggle switch). The flatness of the objective vector function in a large area of the search space is a serious challenge for the optimization process, which will be forced to explore the objective space randomly and it is likely to remain stuck in such

region. In these regions, however, the robustness score is typically non-constant, hence robustness is used to guide the optimization in these cases.

However, one can easily check that $\varrho_{\varphi_i}(\vartheta) > \varrho_{\varphi_i}(\vartheta') \not\Rightarrow P_{\varphi_i}(\vartheta) > P_{\varphi_i}(\vartheta')$. This implies that maximizing directly the average robustness (DrA) could produce under optimal results. In fact, even if typically an increase in the average robustness corresponds to an increase in the satisfaction probability (Figure 8.1), this may fail for some parameters ϑ and ϑ' such that $\varrho_{\varphi_i}(\vartheta) > \varrho_{\varphi_i}(\vartheta')$ and $P_{\varphi_i}(\vartheta) < P_{\varphi_i}(\vartheta')$.

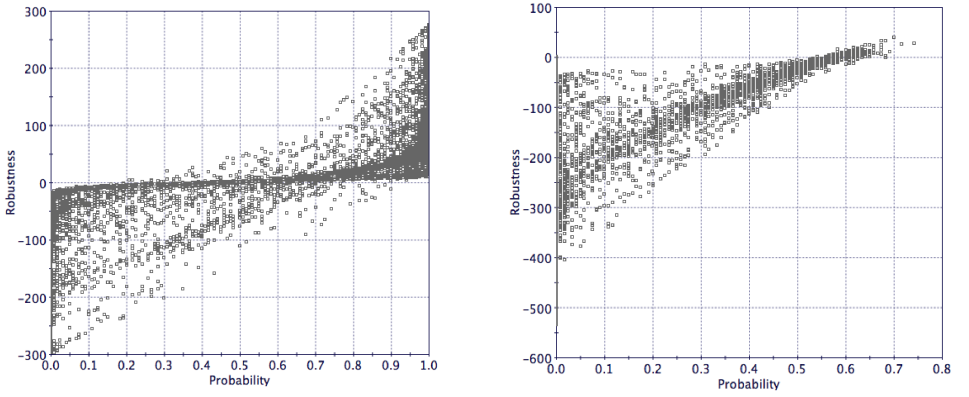
This discussion leads us naturally to consider mixed strategies. In the MA approach, we have modified a genetic algorithm allowing the possibility to compare two designs based on the probability or on the robustness degree automatically. When two designs have the same probability of the STL formulae, the algorithm will automatically switch to the robustness degree. Basically, this algorithm has the possibility to switch to robustness whenever it is stuck in a plateau of the satisfaction probability vector, which typically happens not only when all formulae are satisfied with probability zero, but also with probability one (which is part of the Pareto front). In this region, the robustness score leads us to pick the most robust ϑ^* satisfying the design problem.

Computing the satisfaction probability and the robustness scores. The exact computation of the satisfaction probability or of the expected robustness, up to a fixed numerical precision ε is unfeasible, hence we will exploit statistical methods. More specifically, for each fixed ϑ, we sample N trajectories $\texttt{Traj} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of the PRN using Gillespie's algorithm [Gil77]. Then, for each STL formula φ_i of interest, we estimate $P_{\varphi_i}(\vartheta)$ and $\varrho_{\varphi_i}(\vartheta)$ as

$$\hat{P}_{\varphi_i}(\vartheta) = \frac{|\mathbf{x} \in \texttt{Traj} \mid (\mathbf{x}, 0) \models \varphi_i|}{N}; \qquad \hat{\varrho}_{\varphi_i}(\vartheta) = \frac{\sum_{j=1}^{k} \varrho(\varphi_i, \mathbf{x}_j, 0)}{N}$$

Estimating statistically the functions to optimize is computationally feasible, but has the side effect of making their evaluation noisy, which must be taken into account in the optimization phase. Here we try to reduce this effect by using a large number of simulations per point of the parameter space explored (here we used 500 runs per point, as a good trade-off between noise and computational cost). An appealing alternative would be to rely on statistical regularization methods like Gaussian processes-based emulation (Section 3.1), which are typically an ingredient of active learning algorithms, like Pareto Active Learning (PAL, [ZSKP13]).

The multi-objective optimization algorithm. The optimizer we have used is the NSGA-II [DAPM02], a well known genetic algorithm largely used to solve multi-objective optimization problems, which we also introduce in Subsection 4.2.1. It combines mutation and crossover operators to allow the creation of new points. A comparison procedure based on the Pareto dominance will give preference to points which dominate the largest number of other points, pushing effectively the optimization algorithm towards the Pareto front. We have slightly modified the comparison process: if two points have the same probability with respect to an STL formula, the comparison between them will be based on the robustness degree. Otherwise, the comparison is based on the probability. This strategy permits the optimization process to easily



(a) φ_1 (Probability vs Avg. Robustness)

(b) φ_2 (Probability vs Avg. Robustness)

Figure 8.1: Comparison of average robustness and probability for the toggle switch for **(a)** φ_1 and **(b)** φ_2. We can see how probability and average robustness are positively correlated.

escape from plateaux of the objective function, as discussed previously.

8.2 Case Studies

We consider two test cases that will be used to illustrate our method and discuss the potential of multi-objective optimization in this logic-based setting. The first case is a genetic toggle switch and the second is a variant of the SIR model.

8.2.1 Genetic Toggle Switch

The Genetic Toggle Switch [GCC00] is a regulatory genetic network composed by two genes that mutually repress each other. This is known to be a simple genetic circuit manifesting bistability and memory. It can be described by the following reactions:

$$
\begin{aligned}
r_1 &: \emptyset \xrightarrow{\alpha_1} X_1 && \alpha_1 = 1 \\
r_2 &: \emptyset \xrightarrow{\alpha_2} X_2 && \alpha_2 = 1 \\
r_3 &: X_1 \xrightarrow{\alpha_3} \emptyset && \alpha_3 = \frac{a_1 N^{b_1+1}}{N^{b_1} + X_2^{b_1}} \\
r_4 &: X_2 \xrightarrow{\alpha_4} \emptyset && \alpha_4 = \frac{a_2 N^{b_2+1}}{N^{b_2} + X_1^{b_2}}
\end{aligned}
$$

where the reaction $\emptyset \to X_1$ means that the protein X_1 is created and $X_1 \to \emptyset$ means it is degraded. The model describes genes implicitly, by using Hill's kinetic rate functions, and lumps transcription and translation into a single step. Depending on the parameters, the system can show two different behaviors: either a stable bistability or

rapid switching between equilibria. In the first case the system will tend to stabilize for a long time in one of the two stable equilibria, (either $X_1 \geq X_2$ or $X_2 \geq X_1$). In the second case, the system frequently switches from the state $X_1 \geq X_2$ to $X_2 \geq X_1$, and viceversa.

8.2.2 Epidemic model: SIRS

The SIR model [GCC00] is still widely used to simulate the spreading of a disease among a population. The basic SIR model already introduced in Example 1 can be modified in many ways: considering vaccinations, introducing natural birth and death process, describing a latent phase of the disease, see [Bra08] for an overview. We consider a version of the model described by the following set of reactions:

$$\begin{aligned} r_1 : S + I \xrightarrow{\alpha_1} 2I \qquad & \alpha_1 = k_{si} \cdot \frac{X_s \cdot X_i}{N} \\ r_2 : I \xrightarrow{\alpha_2} R \qquad & \alpha_2 = k_{ir} \cdot X_i \\ r_3 : S \xrightarrow{\alpha_3} R + V \qquad & \alpha_3 = k_v \cdot X_s \\ r_4 : I \xrightarrow{\alpha_4} S \qquad & \alpha_4 = k_{is} \cdot X_i \\ r_5 : R \xrightarrow{\alpha_5} S \qquad & \alpha_5 = k_{rs} \cdot X_r \end{aligned}$$

The population of N individuals is typically divided in three classes (though we will consider an extra one for vaccination):

- *susceptible* S, representing healthy individuals that are vulnerable to the infection.
- *infected* I, describing individuals that have been infected by the disease and are actively spreading it.
- *recovered* R and *vaccinated* V, modelling individuals that are immune to the disease, by having recovered from the disease (R), or by vaccination (V).

Here, r_1 describes the possibility that a susceptible gets the disease and becomes infected. The reaction r_2 models the recovery of an infected agent. The reaction r_3 describes the possibility that susceptible becomes directly a recovered by vaccination (k_v is the vaccination rate). Note that here the population V counts the total number of vaccinated people. Finally, r_4 describes the possibility that an infected individual recovers without gaining immunity, while r_5 models the loss of immunity of a recovered individual. Depending on the parameters $\vartheta = (k_{si}, k_{ir}, k_v, k_{is}, k_{rs})$, different behaviors of the disease could occur, such as the disease rapidly stops, or it becomes endemic, or there are periodic cycles of infections, and so on.

8.3 Results

We discuss now two bi-objective optimization problems, one for each case study. The stochastic systems were simulated and the STL formulae verified with the U-check tool (see. [BMS15]).

8.3.1 Genetic Toggle Switch

In the genetic toggle switch model described in Subsection 8.2.1 there are 4 parameters, (a_1, a_2, b_1, b_2), which we assume that can take values in $[10^{-4}, 5]$, fixing the initial conditions to $(X_1(0), X_2(0)) = (1, 1)$. Depending on the parameters, the system can have two stable equilibria, typically one for which $X_1 > X_2$ and one for which $X_2 > X_1$. In particular, when the difference between X_1 and X_2 is above a certain threshold, then the system tends to stabilize in one equilibrium, and the switching probability (i.e., spontaneously jumping to the other equilibria) will be low. Thus, we consider the two STL formulae:

$$\varphi_1 := \mathbf{F}_{[0,1000]} \quad |X_1 - X_2| > 300 \tag{8.1}$$

$$\varphi_2 := \mathbf{F}_{[0,300]} \quad \mathbf{G}_{[0,50]}(X_1 > X_2) \wedge \mathbf{F}_{[300,550]}\mathbf{G}_{[0,50]}(X_1 < X_2). \tag{8.2}$$

The first formula describes a situation in which equilibria are clearly separated. This has as the side effect the tendency to stabilise the system into one equilibrium. The second formula, instead, describes a switch between equilibria, and hence it is in conflict with the previous goals. In particular, the higher is the difference between X_1 and X_2, the lower is the probability of φ_2. However, one may be interested in maximising the probability of both goals, to obtain a system with well separated configurations and the ability to explore them by the effect of noise. This is a typical use of randomness in cellular decision making, see for instance [BvOC11].

The bi-objective optimization problem we have considered is therefore:

$$\max_{a_1,a_2,b_1,b_2 \in [10^{-4},5]} \{P_{\varphi_1}(\{a_1, a_2, b_1, b_2\}), P_{\varphi_2}(\{a_1, a_2, b_1, b_2\})\} \tag{8.3}$$

The three approaches described in Section 8.1 are compared in Figure 8.3. It can be seen that the mixed (MA) approach performs best, while using only satisfaction probability (DpA) produces very bad results. This is caused by the fact that both STL formulae have a non-zero probability only in a small fraction of the parameter space, and the DpA approach is not able to reach this region. The direct robustness approach (DrA) also produce bad results. In this case the optimization concentrates itself on optimizing the robustness in the zone of the robustness degree space where the probability (at least for φ_2) is zero (see Figure 8.3b).

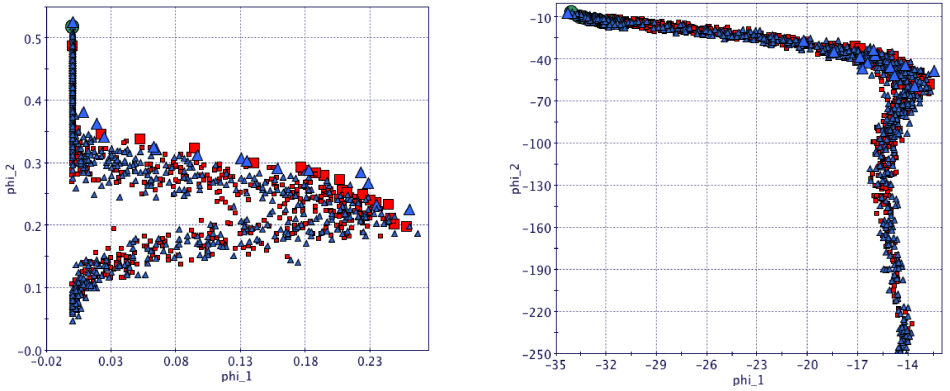
8.3.2 Epidemic model: SIRS

We consider now the SIRS model[2] of Section 8.2.2 and look for the best vaccination rate k_v to obtain the following behaviour for susceptible individuals

$$\varphi_1 := \mathbf{G}_{[20,40]}\,(S < 30 \wedge S > 10) \wedge \mathbf{F}_{[40,60]}\,(S > 50) \tag{8.4}$$

forcing susceptibles to be between 10 and 30 during time 20 to 40 and to get above 50 between time 40 and 60. We additionally want to constraint the number of vaccinations to be below 180 (say the amount of vaccines in storage, so that an higher consumption

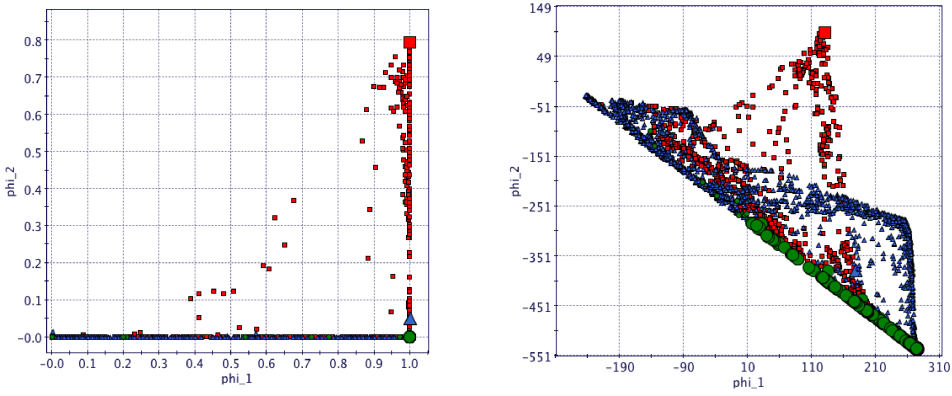
[2] Rate coefficients are as follows: $k_{si} = 0.5$, $k_{ir} = 0.05$, $k_{is} = 0.1$, $k_{rs} = 0.05$, $k_v \in [0.08, 10]$. The vaccination rate is the only free parameter that is optimized.



(a) Probability Space

(b) Robustness Space

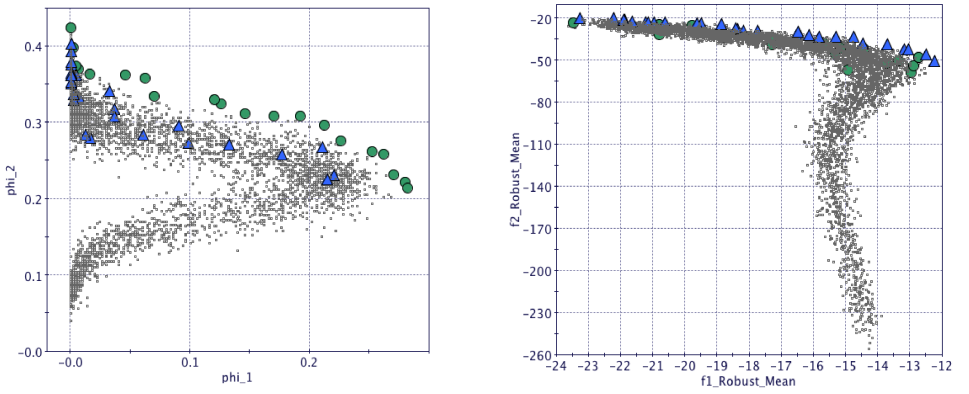
Figure 8.2: Comparison of the three optimization approaches for the SIRS model. The red squared dots represent the MA approach, the blue triangular dots represent the DrA approach and the green circular dots are referred to the DpA. Chart **(a)** shows all designs, with larger dots representing the Pareto fronts of the referred approaches. In **(b)** we have represented the same dots respect to the robustness degree space. Unlike the Toggle Switch example, the three approaches produce comparable results.



(a) Satisfaction Probability

(b) Robustness Degree

Figure 8.3: Comparison of the three optimization approaches for the toggle switch. The red squared dots represent the MA approach, the blue triangular dots represent the DrA approach and the green circular dots are referred to the DpA. Chart **(a)** shows all designs, with larger dots representing the Pareto fronts of the referred approaches. In **(b)** we have represented the same dots respect to the robustness degree space.



(a) Probability Space

(b) Robustness Space

Figure 8.4: Comparison of the Pareto front of the SIRS model, obtained by maximizing the robustness and the probability. The Pareto front obtained by maximizing the probability (green circular designs) dominates the Pareto front obtained by maximizing the robustness (blue triangular designs).

forces to buy new vaccines), which we enforce with the following goal

$$\varphi_2 := \mathbf{G}_{[0,250]}\,(V < 180) \tag{8.5}$$

Differently from the Toggle Switch scenario, the three optimization approaches have similar performances, as shown in Figure 8.2. The reason is that the satisfaction probability of both φ_1 and φ_2 is non-zero in a large portion of the state space, and the correlation between probability and robustness is larger than for the toggle switch.

8.4 A deeper look at robustness

In this section we focus our attention on the robustness score associated with a formula, starting from a potential criticism to multi-objective optimization. Logic formulae can be combined, for instance by Boolean connectives. Hence, if we have two formulae, φ_1 and φ_2, representing two potentially conflicting objectives, we can take their conjunction and solve the single objective optimization for $\varphi_1 \wedge \varphi_2$, e.g., with the method of [BBNS15] which uses the robustness score. Even if the two approaches are not directly comparable considering that they solve different tasks (see. [Deb14]), the second approach could be appealing for computational reasons.

However, consider the following scenario: let φ_1, φ_2 and $\psi := \varphi_1 \wedge \varphi_2$ such that $\forall \vartheta \in \Theta$, $\varrho_{\varphi_1}(\vartheta) \in [a, b]$ and $\varrho_{\varphi_2}(\vartheta) \in [c, d]$ where $a, b, c, d \in \mathbb{R}$ and $b \ll c$. Clearly $\forall \vartheta \in \Theta, \varrho_{\psi}(\vartheta) = \min(\varrho_{\varphi_1}(\vartheta), \varrho_{\varphi_2}(\vartheta)) = \varrho_{\varphi_1}(\vartheta)$, hence optimizing the robustness, we would completely ignore $\varrho_{\varphi_2}(\vartheta)$. As a less artificial example, we have compared these multi-objective and single-objective approaches applied to the maximization of the average robustness of the SIRS model (see Subsection 8.3.2 and Subsection 8.2.2) with respect

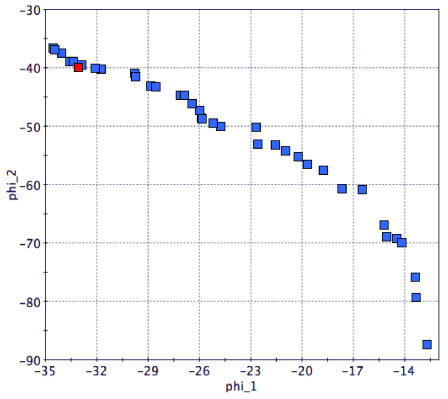


Figure 8.5: Comparison of multi-objective optimization of formulae (8.4) and (8.5) for the SIRS model. The blue boxes represent the Pareto front relative to the maximization of the average robustness. The red box instead represents the result of the maximisation of the average robustness of the conjunction of the two formulae (reporting individual robustness scores).

to the formulae (8.4) and (8.5). The results (Figure 8.5) show that the single objective problem tends to maximize the average robustness of the second formula, largely ignoring the first one.

Indeed, these examples are paradigmatic of a more structural problem with the robustness score, namely its sensitivity to the scale of each atomic predicate. Essentially, a formula with more than one atomic predicate will combine their values in ways that may not be fully meaningful if the scale (domain of variability) of the two predicates is very different, as one may dominate the other. Sorting out this problem by a standardization of the predicates may be difficult, as their range of variability is unknown a priori. To circumvent this issue, at least from a system design perspective, we can de facto resort to multi-objective optimization. Intuitively, at least for a suitable subclass of formulae, we can try to rewrite them so to obtain a Boolean combination of temporal formulae with a single atomic predicate, and then define a suitable multi-objective problem on such subformulae. We plan to investigate the feasibility of this approach in a future work.

8.5 Conclusion and Future Works

In this work we investigated the problem of designing stochastic models of CRN with respect to behavioural goals expressed in temporal logic, with a multi-objective approach. We discussed several score functions, using satisfaction probability, robustness, or a combination of the two, and compared them for two different case studies: a toggle switch and a SIRS epidemic model. We also discussed the advantages of multi-objective approach to deal with robustness more consistently.

The present work used a genetic algorithm to solve the multi-objective optimization

problem. We have modified its comparison criterion in order to use a combination of satisfaction probability and robustness degree. We plan to investigate the use of statistically more refined optimization methods to deal with noise in the estimation of the objective function due to the use of simulation, like Pareto Active Learning [ZSKP13]. This will allow us to reduce the number of simulation runs per point of the parameter space, thus improving complexity.

9

Falsification of Cyber-Physical Systems

Model-based Development (MBD) is a well known design framework of complex engineered systems, concerned with reducing cost and time of the prototyping process. Most prominently, this framework has been adopted in the industrial fields such as automotive and aerospace where the conformity of the end product is extremely important. The majority of systems in these areas are Cyber-Physical Systems (CPS) [LS16] (Section 1.1), where physical and software components interact producing complex behaviors. These systems can be described by appropriate mathematical models which are able to capture all the system behaviors. Moreover, it is necessary to have a suitable specification framework capable of analyzing the output of such models.

Hybrid systems [MMP91] are the mathematical framework usually adopted, while temporal logic (Chapter 2), due to its ability to describe temporal events, is generally used as specification framework. The high level of expressivity of hybrid systems, which is the main reason for their success, is also the cause of their undecidability, even for simple logic formulae. Subclasses of hybrid systems which are decidable for specific temporal logic formulae exist and have been widely studied during the last 15 years, as well as model checking techniques capable of verifying them [ACH+95]. Unfortunately, the majority of CPS used nowadays in the industrial field are much more complex than decidable hybrid systems. They are mainly described by using block diagram tools (i.e., Simulink/Stateflow, Scade, LabVIEW, and so on) where several switch blocks, 2/3-D look-up tables and state transitions coexist. These CPS are generally not decidable and standard model checking techniques are not feasible, leading to the proposal of different techniques [ALFS11b].

Testing procedures with the purpose of verifying the model on specific behaviors have been adopted for several years. These are feasible approaches whenever it is possible to write in advance collections of test cases which extensively cover all the possible events leading to system failure [Vin98]. With the increase of complexity, such an *a priori* assumption is not viable in most of the real cases and for this reason different techniques, such as random testing and search-based testing, have been introduced [ZKH03]. The

general idea consists in expressing the falsification procedure as an optimization process aiming at minimizing a target quantity which describes "how much a" given property is verified. For example, achieving a negative value of the robustness semantics of a given Signal Temporal Logic (STL) [DM10] formula means falsifying the system with respect to that formula.

Related Work. Different approaches have been proposed to achieve the falsification of black box models, starting from test based approaches until recently, when search-based test approaches have become more popular. Stochastic local search [DJKM15], probabilistic Monte Carlo [AFS+13] and mixed coverage/guided strategy [DDD+15] approaches have been proposed and benchmark problems created [JDK+14, HAF14]. Two software packages [Don10, ALFS11b] implement the aforementioned techniques. Both these software tools assume a fix parameterization of the input function, differently from us. Similarly to our approach, in [DDD+15] and [DJKM15] the fixed parameterization is avoided. More specifically in [DDD+15] no parameterization has been used at all and the input signals are modified on the fly based on the robustness of the partial system trajectories. In [DJKM15] a uniform discretization of the input domains (both time and values) is dynamically applied to discretize the search space. The use of Gaussian processes for falsification has been adopted in [Aka16] but it is restricted to conditional safety properties.

Contribution. In this chapter[1] we study the falsification problem of black box systems (i.e., block diagram models such as Simulink/Stateflow model or sets of ordinary differential equations generally used in automotive or aerospace industrial fields) which takes as input and produce as output continuous or piecewise continuous (PWC) signals. The requirements are expressed by using STL. Solving such falsification problems in a search-based framework poses two main challenges. Generally, the simulation of block diagram models is time consuming, hence it is necessary to falsify the model with as few simulations as possible. Moreover, the models accept continuous and/or PWC signals as inputs and an efficient finite dimensional parametrization is necessary to perform an optimization procedure. The contribution we propose in this work is to tackle these challenges by a novel strategy leveraging machine learning techniques (Gaussian processes and active learning) and by using a new adaptive version of the control points parameterization approach.

9.1 Domain Estimation with Gaussian Processes

We now define the domain estimation problem and solve it by using Gaussian processes. In this section we describe how to cast the falsification problem into a domain estimation problem.

Definition 23. *Consider a function $f : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D} \subseteq \mathbb{R}^k$ is a general domain space, and an interval $I = [a, b] \subseteq \mathbb{R}$. We define the* domain estimation problem *as the*

[1] This contribution has been published in [SPB17b].

task of identifying the set $\mathcal{B}$ of points $x \in \mathcal{D}$ such that $f(x) \in I$:

$$\mathcal{B} = \{x \in \mathcal{D} \mid f(x) \in I\} \subseteq \mathcal{D}, \tag{9.1}$$

In practice, if $\mathcal{B} \neq \emptyset$, we will limit us to identify a subset $B \subseteq \mathcal{B}$ of size n.

Gaussian Processes (GP) can be efficiently used to solve this task. Similarly to the cross-entropy methods for optimization [RK13], the idea is to implement an iterative sample strategy in order to increase the probability to sample a point in $\mathcal{B}$, as the number of iterations increases. Consider the set $K(f) = \{(x_i, f(x_i))\}_{i \leq n}$ representing the partial knowledge we have collected after n iterations and the Gaussian process $f_K(x) \sim GP(m_K(x), \sigma_K(x))$ trained on $K(f)$. We can easily estimate $P(x \in \mathcal{B}) = P(f_K(x) \in I)$, where $I = [a, b]$, by computing the probability of a Gaussian distribution with mean $m_K(x)$ and variance $\sigma^2_K(x)$ as follows

$$P(x \in \mathcal{B}) = \Phi\left(\frac{b - m_K(x)}{\sigma_K(x)}\right) - \Phi\left(\frac{a - m_K(x)}{\sigma_K(x)}\right) \tag{9.2}$$

This[2] corresponds to our uncertainty on the value of $f(x)$ belonging to I, as captured by the GP reconstruction of f. The previous probability can be effectively used to solve the domain estimation problem described in Definition 23. Our approach is described in Algorithm 7. The idea is to use $P(f_K(x) \in I)$ to guide the sampling.

- During initialization (line 1), we set the iteration counter (i) and the minimum distance (d) from the interval I. The set (B) containing the elements of ($\mathcal{B}$) is set to empty ensuring that the algorithm is run at least once. The knowledge set $K(f)$ is initialized with some randomized points sampled from $\mathcal{D}$ (line 2).
- In the iterative loop (line 3), the algorithm first checks if the number of counterexamples (ce) or maximum number of iterations ($maxIter$) has been reached. In this case, the method stops returning the estimated set (B) and the minimum distance from I that has been registered until that point. Otherwise new GPs are trained by using $K(f)$ (line 4) and a set composed by m points (D_{grid}) is generated by a Latin Hypercube sampling [MBC79], so to have a homogeneous distribution of points in space (line 5). For each of these points x, the probability $P(x \in \mathcal{B}) = P(f_K(x) \in I)$ is evaluated and the set $\{(x, P(x \in \mathcal{B})), x \in D_{grid}\}$ is then created. Afterwards, a candidate point x_{new} is sampled from D_{grid} proportionally to its associated probability (line 6) so to increase the sampling of points with higher estimated probability of belonging to $\mathcal{B}$. Consequently, $K(f)$ is upgraded and if $x \in \mathcal{B}$ then x is added to B (line 11). The procedure outputs also d, the minimum distance of the evaluated points from the interval I calculated during the procedure.

9.2 The Falsification Process

We adopt a black box optimization approach, meaning that we consider the target block diagram model as a black box dynamical system $\mathcal{M} = \{\mathbf{U} \times \mathbf{X}, \texttt{sim}\}$ (see Subsec-

[2] Φ is the CDF of the standard normal distribution.

Algorithm 7 DOMAINESTIMATION algorithm

Require: $maxIter, ce, m, f, I$
1: $i \leftarrow 0,\ B \leftarrow \emptyset,\ d \leftarrow +\infty$
2: INITIALIZE($K(f)$)
3: **while** ($|B| \leq ce$ **and** $i \leq$ maxIter) **do**
4: $f_{K(f)} \sim$ TRAINGAUSSIANPROCESS($K(f)$)
5: $D_{grid} \leftarrow$ LHS(m)
6: $x_{new} \leftarrow$ SAMPLE$\{(x, P(x \in \mathcal{B})), x \in D_{grid}\}$
7: $f_{new} \leftarrow f(x_{new})$
8: $d \leftarrow \min(d,$ DISTANCE$(f_{new}, I))$
9: $K(f) \leftarrow K(f) \cup \{(x_{new}, f_{new})\}$
10: **if** $f_{new} \in I$ **then**
11: $B = B \cup \{x_{new}\}$
12: **end if**
13: $i \leftarrow i + 1$
14: **end while**

tion 1.3.3).

A big effort during the prototyping process consists in verifying the requirements usually expressed as safety property, such as:

$$\forall(\mathbf{u}, x_0) \in \mathbf{U}^{\mathbb{T}} \times \mathcal{X}_0\ , \varrho(\varphi, (\mathbf{u}, \mathbf{x}), 0) > 0 \tag{9.3}$$

meaning that for each input function and initial state $x_0 \in \mathcal{X}_0 \subseteq \mathbf{X}$, the dynamics of the systems represented by the couple $(\mathbf{u}, \mathbf{x})$, where $\mathbf{u} \in \mathbf{U}^{\mathbb{T}}$ is the input function and $\mathbf{x} \in \mathbf{X}^{\mathbb{T}}$ is the state function, satisfies the STL formula φ. It is possible to interpret the safety condition (9.3) as a domain estimation problem associated with

$$\mathcal{B} = \{(\mathbf{u}, x_0) \in \mathbf{U}^{\mathbb{T}} \times \mathcal{X}_0\ , \varrho(\varphi, (\mathbf{u}, \mathbf{x}), 0) < 0\} \tag{9.4}$$

with the purpose of verifying its emptiness, which entails that (9.3) is satisfied. We call $\mathcal{B}$ the *counterexample set* and its elements counterexamples.

Solving the previous domain estimation problem could be extremely difficult because of the infinite dimensionality of the input space, which is a space of functions. For this reason, it is mandatory to parameterize the input function by means of an appropriate finite dimensional representation.

Let us consider for simplicity that $\mathbf{U} = U_1 \times \cdots \times U_m$, where $m < +\infty$. One of the most used parameterization—mainly for its simplicity—is the *fixed control point parameterization* (fixCP): after having fixed the times $(t_1^k, \ldots, t_{n_k}^k)$, the control points $\{(t_1^k, u_1^k), \ldots, (t_{n_k}^k, u_{n_k}^k)\}$ are chosen as parameter of the k-th input signals. We also need to choose an interpolation set of function with n_k degrees of freedom for each k-th input signals ($\mathcal{F}_{n_k}^k \subset \{\mathbb{T} \to U_k\}$, e.g., piecewise linear, polynomials of degree n_k, and so on (see [SF12])), the fixCP parameterization will associate with each control point $c_k = \{(t_1^k, u_1^k), \ldots, (t_{n_k}^k, u_{n_k}^k)\}$ the unique function $F_{c_k} \in \mathcal{F}_{n_k}^k$ satisfying $\forall i \leq n\ , F_{c_k}(t_i^k) = u_i^k$. Let us denote by $\mathcal{F}_{\mathbf{n}} = (\mathcal{F}_{n_1}^1, \ldots, \mathcal{F}_{n_m}^m)$ the set of interpolating functions.

It is clear that by increasing the number of control points, we will enlarge the set

of approximant functions $\mathcal{F}_{\mathbf{n}}$: $\mathbf{n} \leq \mathbf{m}$ implies $\mathcal{F}_{\mathbf{n}} \subset \mathcal{F}_{\mathbf{m}}$, where $\mathbf{n} \leq \mathbf{m}$ is intended pointwise. As piecewise linear or polynomial functions are known to be dense in the space of continuous functions, by choosing an appropriately large $\mathbf{n}$, we can approximate any input function with arbitrary precision.

Considering an $\mathbf{n}$-fixCP, which is a fixCP where $\mathbf{n} = (n_1, \ldots, n_m)$ represents the number of control points used for each input variables, it is possible to introduce the domain estimation problem (9.4) associated with the following set:

$$\mathcal{B} = \{(\bar{c}, x_0) \in \{U_1^{n_1} \times \cdots \times U_m^{n_m}\} \times \mathcal{X}_0, \quad \varrho(\varphi, (\mathcal{F}_{\mathbf{n}}(\bar{c}), x), 0) < 0\} \tag{9.5}$$

which , differently from (9.4), is a finite dimensional set described by using $\sum_{j=1}^{m} n_j + dim(\mathcal{X}_0)$ variables.

By the density argument it is clear that

$$\text{(9.4) has at least one element} \iff \exists \mathbf{n} \in \omega^m, \text{ (9.5) has at least one element}$$

A possible strategy is to solve the domain estimation problem associated with (9.5) by choosing the minimum $\mathbf{n}$ such that $\mathcal{F}_{\mathbf{n}} \times \mathcal{X}_0$ contains a counterexample. Applying that strategy, even in simple cases, could be cumbersome as shown in the following example.

Toy example. Consider a simple black box model which accepts a single piecewise constant function $u : [0,1] \to [0,1]$ as input function and returns the same function $x = u$ as output. Considering the following requirement $\varphi := \neg(\mathbf{G}_{[0,0.51]}\, 0 < x < 0.2 \wedge \mathbf{G}_{[0.55,1]}\, 0.8 < x < 1)$, it is evident that it could be falsified only in a control point parameterization having at least the point (t_i, u_i) such that $t_i \in [0.51, 0.55]$. Otherwise if this points does not exists it means the output signals will assume a constant value in $[0.51, 0.55]$ which implies that or $\mathbf{G}_{[0.55,1]}\,(0.8 < x < 1)$ or $\mathbf{G}_{[0,0.51]}\,(0 < x < 0.2)$ is false, meaning that φ is not falsified. The minimum number of uniformed fixed control points necessary to achieve it is 9, which entails a considerable computational effort.

A natural way to overcome the limitation of the fixCP consists in considering the times of the control points as variables. An $\mathbf{n}$-*adaptive Control Points parameterization* ($\mathbf{n}$-adaCPP) consists in a function $\bar{F}^k_{n_k} : \mathbb{T}^{n_k} \times U_k^{n_k} \to \mathcal{F}^k_{n_k}$, which has twice as much parameters than the fixed version: values at control points and times (which are constrained by $\forall i < n\ t_i \leq t_{i+1}$). The adaptive parameterization is preferable with respect to the fixed one because of its ability to describe functions with local high variability even with a low number of control points. In fact it is possible to concentrate any fraction of the available control points in a small time region, inducing a large variation in this region, while letting the parameterized function vary much less outside it.

9.2.1 Adaptive Optimization

The idea of the adaptive optimization approach consists in falsifying (9.3) starting from a simple input function and increasing its expressiveness by increasing the number of control points. After having defined a parameterization for each of the m input signals (which compose the input function $\mathbf{u}$), Algorithm 8 works as following:

- At the first iteration a parameterization $F_{\mathbf{n_0}} = \{F_1^0, \dots, F_n^0\}$ with zero control points for each signals ($\mathbf{n_0} = (0, \dots, 0)$) is considered (line 1). Zero control points means defining input signals which are constant functions. The final counterexample set (B) is set to empty, which ensures the optimization is run at least once (line 2).

- In the iterative loop, the algorithm first checks if the number of counterexamples (ce) or if the maximum global number of iterations (mgi) has been reached. In this case, the method stops returning the counter example set (B). Otherwise, the falsification problem is solved by using the domain estimation procedure DOMAINESTIMATION (Algorithm 7) which returns the counterexample set and the minimum value of the robustness found by using that parameterization (see Section 9.1 for details). The parameterization is then expanded by picking a coordinate of the input signal (line 5 - 9) and adding a new control point (line 10), obtaining a new parameterization $F_{\mathbf{n_{i+1}}}$.

Algorithm 8 ADAPTIVEGPFALSIFICATION algorithm

Require: mgi, mii, ce, m, φ
1: **Input:** mgi, mii, ce, m, φ
2: $\mathbf{n_0} \leftarrow (0, \dots, 0)$
3: $B \leftarrow \emptyset$, $k_0 \leftarrow 0$, $i \leftarrow 0$, $d_0 \leftarrow +\infty$
4: **while** ($|B| \leq ce$ **and** $i \leq mgi$) **do**
5: $[B^-, d_{i+1}] =$ DOMAINESTIMATION$(mii, \mathbf{n}_i, ce - |B|, m, \varrho(\varphi, \cdot, t), (-\infty, 0))$
6: **if** $d_{i+1} > d_i$ **then**
7: $k_{i+1} \leftarrow k_i$
8: **else**
9: $k_{i+1} \leftarrow (k_i + 1) \mod n$
10: **end if**
11: $\mathbf{n}_{i+1} \leftarrow \mathbf{n}_i + \mathbf{e}_k$
12: $i \leftarrow i + 1$
13: $B \leftarrow B \cup B^-$
14: **end while**

The general idea of this approach is to keep low the number of parameters by starting from constant signal and gradually increasing the number of control points of the input functions. In the adaptive control points parameterization, adding a control point means adding two new degrees of freedom (one for the time and one for the value of the control point). This means, on the one hand, having more expressiveness and so more chances to falsify the system, but on the other hand this complicates the optimization process and increases the dimension of the search space as well, hence, the minimum number of simulations required to solve it. For this reason it is convenient to add control points only where it is truly necessary.

9.3 Probabilistic Approximation Semantics

Gaussian processes can be used to estimate the probability that a given input falsifies a system as described in Section 9.1 and Section 9.2. This fact offers the possibility to define an approximate semantics which generalizes the concept of probability of falsification that we can infer considering the knowledge of the system we have collected. The basic idea is to decompose an STL formula as a Boolean combination of temporal modalities, propagating the probability of the temporal operators, estimated by GPs, through the Boolean structure. Formally, let $\mathcal{L}_0$ be the subset of STL containing only atomic propositions and temporal formulae of the form $\varphi_1 \mathbf{U}_{[T_1,T_2]}\varphi_2$, $\mathbf{F}_{[T_1,T_2]}\varphi$ and $\mathbf{G}_{[T_1,T_2]}\varphi$, that cannot be equivalently written as Boolean combinations of simpler formulae. For example $\mathbf{F}_T(\varphi_1 \vee \varphi_2)$ is not in $\mathcal{L}_0$ because $\mathbf{F}_T(\varphi_1 \vee \varphi_2) \equiv \mathbf{F}_T\varphi_1 \vee \mathbf{F}_T\varphi_2$. Furthermore, let $\mathcal{L}$ be the logic formed by the Boolean connective closure of $\mathcal{L}_0$. Clearly $\mathcal{L}$ is isomorphic to STL. [3]

For simplicity, let us denote by ϑ a parameter and describe the input function by u_ϑ and the initial state by $x_{0\vartheta}$. We write x_ϑ to indicate the path generated by the simulator, given as input u_ϑ and $x_{0\vartheta}$, according to Section 1.3.3. We want to define an (approximate) semantics giving the probability that a path x_ϑ satisfies a given formula $\psi \in \mathcal{L}$ (without simulating it). The idea is to evaluate the quantitative semantics of the atomic formulae $\varphi_j \in \mathcal{L}_0$ of ψ on a finite collection of parameters $(\{\vartheta_i\}_{i\leq n})$, then building GPs in order to estimate the probability that the quantitative semantics of each formula φ_j is higher than zero on a target parameter. This is again a Domain Estimation Problem (Section 9.1), where the function is the robustness associated with the STL formula φ_j and the interval I is $(0, +\infty)$. We propagate this probability through the Boolean structure of ψ according to the following:

Definition 24 (Probabilistic Approximation Semantics of $\mathcal{L}$)**.** *The probabilistic approximation function $\gamma : \mathcal{L} \times \mathbf{U}^{\mathbb{T}} \times [0, \infty) \to [0, 1]$ is defined by:*

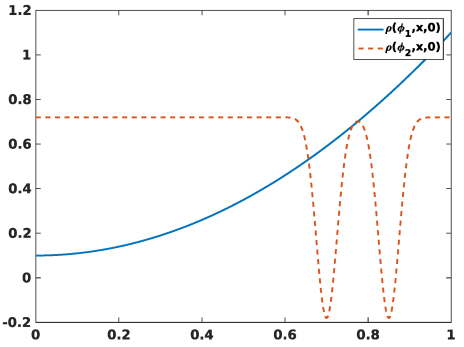
- $\gamma(\varphi, \vartheta, t) = P(f_{K(\varphi)}(\vartheta) > 0)$
- $\gamma(\neg\psi, \vartheta, t) = 1 - \gamma(\psi, \vartheta, t)$
- $\gamma(\psi_1 \wedge \psi_2, \vartheta, t) = \gamma(\psi_1, \vartheta, t) * \gamma(\psi_2, \vartheta, t)$
- $\gamma(\psi_1 \vee \psi_2, \vartheta, t) = \gamma(\psi_1, \vartheta, t) + \gamma(\psi_2, \vartheta, t) - \gamma(\psi_1 \wedge \psi_2, \vartheta, t)$

where $K(\varphi) = \{\vartheta_i, \varrho(\varphi, \vartheta_i, t)\}_{i=1,..,n}$ is the partial knowledge of the satisfiability of $\varphi \in \mathcal{L}_0$ that we have collected performing n simulations for parameters $(\vartheta_i)_{i=1,..,n}$. $f_{K(\varphi)}$ is the GP trained on $K(\varphi)$, and P refers to its probability. For simplicity we use $\gamma(\psi, \vartheta, t)$ to mean $\gamma(\psi, (u_\vartheta, x_\vartheta), t)$.

In the previous definition, the probability $P(f_{K(\varphi)}(\vartheta) > 0)$ is easily computed, as $f_{K(\varphi)}(\vartheta)$ is normally distributed.

Including the Probabilistic Approximation Semantics (PAS) in our falsification procedure (Algorithm 8) is straightforward. Given the formula we have to falsify, first we negate and decompose it in order to identify the $\mathcal{L}$ formula and ist atom in $\mathcal{L}_0$ associated

[3] $\varphi \in \mathcal{L}$ iff $\psi := \varphi \,|\, \neg\psi \,|\, \psi \vee \psi \,|\, \psi \wedge \psi$, with $\varphi \in \mathcal{L}_0$.



(a) PAS example

STL Formulae
$\phi_1^{AT}(\bar{v},\bar{\omega}) = \mathbf{G}_{[0,30]}(v \leq \bar{v} \wedge \omega \leq \bar{\omega})$
$\phi_2^{AT}(\bar{v},\bar{\omega}) = \mathbf{G}_{[0,30]}(\omega \leq \bar{\omega}) \rightarrow \mathbf{G}_{[0,10]}(v \leq \bar{v})$
$\phi_3^{AT}(\bar{v},\bar{\omega}) = \mathbf{F}_{[0,10]}(v > \bar{v}) \rightarrow \mathbf{G}_{[0,30]}(\omega \leq \bar{\omega})$

(b) Automatic Transmission Req.

Figure 9.1: **(a)** Example on the use of the probabilistic semantics. The curve treating the formula as a single one is the minimum of the two curves. **(b)** Requirements for the Automatic Transmission example of Section 9.4.

with it. Then we pass all the basic $\mathcal{L}_0$ formulae to the DOMAINESTIMATION procedure (Algorithm 7) and train a GP for each of them (instead of considering a single function (Algorithm 7, line 5). Subsequently we calculate its probabilistic approximation semantics to drive the sampling strategy (Algorithm 7, line 7). The rest of the algorithm remains the same.

Remark. Consider the STL formula $\varphi = \mathbf{G}_{[0,30]}(v \leq 160 \wedge \omega \leq 4500)$. This formula is not in $\mathcal{L}_0$, as it can be rewritten as $\mathbf{G}_{[0,30]}(v \leq 160) \wedge \mathbf{G}_{[0,30]}(\omega \leq 4500)$. We could have defined the set $\mathcal{L}_0$ in different ways (e.g., by including also φ), our choice corresponding to the finer decomposition of temporal formulae. Even if this leads to an increased computational cost (more GPs have to be trained), it also provides more flexibility and allows us to exploit the Boolean structure in more detail, as discussed in the following example.

Example. To clarify the advantages of the PAS, consider the functions $\varrho(\varphi_1, x, 0) = x^2 + 1$ and $\varrho(\varphi_2, x, 0) = -0.2 + 0.9(1 - h(x, 0.7, 0.035) - h(x, 0.85, 0.035))$ representing the robustness associated with unspecified formulae φ_1 and φ_2 at time 0 and for input parameter x, respectively. Here $h(x, m, s)$ is a Gaussian function with mean m and standard deviation s. We compare two approaches. In the first one, we calculate the probability of its negation i.e. $\gamma(\neg(\varphi_1 \wedge \varphi_2), x, 0) = 1 - \gamma((\varphi_1 \wedge \varphi_2), x, 0)$ by means of a single Gaussian process. In the second one, we decompose the conjunction and calculate its PAS $\gamma(\neg(\varphi_1 \wedge \varphi_2), x, 0) = 1 - \gamma(\varphi_1, x, 0) * \gamma(\varphi_2, x, 0)$ by means of two separated Gaussian processes. Functions used by the method to drive the sample are represented in Figure 9.1a. In the first case, the signal which is smooth in $[0, 0.65]$ and highly variable in $(0.65, 1]$ forces the method to sample many points near $x = 0$, as the function is close to zero near this point. This requires 55.35 ± 45.10 function evaluations. On the contrary the second approach shows a rapid discovery of the falsification area, i.e., 17.19 ± 7.71 evaluations, because the two components are treated independently, and the method quickly finds the minima regions of $\gamma(\varphi_2, x, 0)$, after an initial phase of

	Adaptive PAS		Adaptive GP-UCB		S-TaLiRo		
Req	nval	times	nval	times	nval	times	Alg
$\varphi_1^{AT}(160, 4500)$	4.42 ± 0.53	2.16 ± 0.61	4.16 ± 2.40	0.55 ± 0.30	5.16 ± 4.32	0.57 ± 0.48	UR
$\varphi_1^{AT}(160, 4765)$	6.90 ± 2.22	5.78 ± 3.88	8.7 ± 1.78	1.52 ± 0.40	39.64 ± 44.49	4.46 ± 4.99	SA
$\varphi_2^{AT}(75, 4500)$	3.24 ± 1.98	1.57 ± 1.91	7.94 ± 3.90	1.55 ± 1.23	12.78 ± 11.27	1.46 ± 1.28	CE
$\varphi_2^{AT}(85, 4500)$	10.14 ± 2.95	12.39 ± 6.96	23.9 ± 7.39	9.86 ± 4.54	59 ± 42	6.83 ± 4.93	SA
$\varphi_2^{AT}(75, 4000)$	8.52 ± 2.90	9.13 ± 5.90	13.6 ± 3.48	4.12 ± 1.67	43.1 ± 39.23	4.89 ± 4.43	SA
$\varphi_3^{AT}(80, 4500)$	5.02 ± 0.97	2.91 ± 1.20	5.44 ± 3.14	0.91 ± 0.67	10.04 ± 7.30	1.15 ± 0.84	CE
$\varphi_3^{AT}(90, 4500)$	7.70 ± 2.36	7.07 ± 3.87	10.52 ± 1.76	2.43 ± 0.92	11 ± 9.10	1.25 ± 1.03	UR

Table 9.1: Results. All the times are expressed in seconds. Legend - nval: number of simulations, times: time needed to falsify the property, Alg: the algorithm used as described in Section 9.4.

homogeneous exploration. In addition, the paraboloid $\gamma(\varphi_1, x, 0)$ is smooth and requires few evaluations for a precise reconstruction.

9.4 Case Studies

In this section we discuss a case study to illustrate our approach, taken from [SF12]. We will compare and discuss the performance of a prototype implementation in Matlab of our approach with S-TaLiRo toolbox [FSUY12]. We use S-TaLiRo to compute the robustness, and the implementation of Gaussian Process regression provided by Rasmussen and Williams [RN10].

Automatic Transmission (AT). We consider a Simulink model of a Car Automatic Gear Transmission Systems. There are two inputs: the throttle and the brake angle dynamics describing the driving style. Modes have two continuous state variables, describing vehicle (v) and engine (ω) speed. The Simulink model is initialized with a fixed initial state $(w_0, v_0) = (0, 0)$, it contains 69 blocks (2 integrators, 3 look-up tables, Stateflow Chart, ...). The requirements are described by means of STL formulae as reported in Figure 9.1b. The first requirement (φ_1^{AT}) is a so called *invariant*, which says that in the next 30 seconds the engine and vehicle speed never reach $\bar{\omega}$ rpm and $\bar{v}$ km/h, respectively. The second requirement (φ_2^{AT}) says that if the engine speed is always less than $\bar{\omega}$ rpm, then the vehicle speed can not exceed $\bar{v}$ km/h in less than 10 seconds. Finally, the third requirement (φ_3^{AT}) basically says that if within 10 seconds the vehicle speed is above $\bar{v}$ km/h then from that point on the engine speed is always less than $\bar{\omega}$ rpm.

Results. We analyze the performance of our approach in terms of the minimum number of simulations and computational time needed to falsify the previous test cases. We have performed 50 optimization runs for each STL formula and compared its performance with the best statistics achieved among a Cross Entropy (CE), Montecarlo Sampling (SA) and Uniform Random Sampling (UR) approaches performed with the S-TaLiRo tool [ALFS11b] and the GP-UCB algorithm applied to falsification as described in [Aka16]. As the table shows, our approach (Adaptive PAS) has good results in terms

of the minimum number of evaluations needed to falsify the systems with respect to the STL formulae, outperforming in almost all tests the methods of the S-TaLiRo suite and the Adaptive GP-UCB approach. This is the most representative index, as in real industrial cases the simulations can be considerably expensive (i.e., cases of real measurements on power bench, time and computation intensive simulations). In these cases the total computational time is directly correlated with the number of simulations and the time consumed by the optimizer to achieve its strategy becomes marginal. Furthermore, we are testing our method with a prototype implementation which has not been optimized, in particular for what concerns the use of Gaussian processes. Despite this, the numerical results in terms of the minimum number of simulations are outperforming S-TaLiRo and GP-UCB approach.

Conditional Safety Properties. When we define a conditional safety property i.e. $\mathbf{G}_T(\varphi_{cond} \rightarrow \varphi_{safe})$ we would like to explore cases in which the formula is falsified but the antecedent condition holds (see [Aka16]). This is particularly relevant when the formula cannot be falsified, as it reduces the search space, ignoring regions where the formula is trivially true due to a false antecedent. Focusing on the region where φ_{cond} holds requires a straightforward modification of the sampling routine of the Domain Estimation Algorithm (Algorithm 7, line 6-7). Instead of performing the sampling directly on the input provided by the Latin Hypercube Sampling Routine (Algorithm 7, line 6), we previously define a set of inputs verifying the antecedent condition (by the standard Domain Estimation Algorithm using the Gaussian processes trained on the robustness of the antecedent condition) and then we sample from this set the candidate point (Algorithm 7, line 7).

To verify the effectiveness of this procedure we consider the STL formula $\mathbf{G}_{[0,30]}(\omega \leq 3000 \rightarrow v \leq 100)$ which cannot be falsified by any configuration of the AT model. We tray to falsify it and calculate the percentage of sampled inputs satisfying its antecedent condition (i.e., $\omega \leq 3000$). This percentage is 43% for the GP-UCB algorithm, but increases to 87% for the modified domain estimation algorithm.

9.5 Conclusion and Future Works

In this chapter, we propose an adaptive strategy to find bugs in black box systems. We search in the space of possible input functions, suitably parameterized in order to make it finite dimensional. We use a separate parameterization for each different input signal, and we use an adaptive approach, increasing gradually the number of control points as the search algorithm progresses. This allows us to solve falsification problems of increasing complexity, looking first for simple functions and then for more and more complex ones. The falsification processes are then cast into the domain estimation problem framework, which uses the Gaussian Processes (GP) to constructs an approximate probabilistic semantics of STL formulae, giving high probability to regions where the formula is falsified. The advantage of using such an approach is that it leverages the Bayesian emulation providing a natural balance between exploration and exploitation, which are the key ingredients in a search-based falsification algorithm. In addition to a novel use of GP, we also rely on a new adaptive parameterization, treating the time of each control point as a variable, thus leading to an increase in expressiveness and flexi-

bility, as discussed in Section 9.2. Moreover with a slight modification of our algorithm we efficiently manage the falsification of the conditionsl safety properties, increasing the efficiency of the usual GP-UCB algorithm in focussing the search on the region of points satisfying the antecedent.

The experimental results are quite promising, particularly as far as the number of simulations required to falsify a property is concerned, which is lower than other approaches. The computational time of the current implementation, however, is in some cases higher than S-TaLiRo. The main problem is in the cost of computing predictions of the function emulated with a GP. This cost, in fact, is proportional to the number of already evaluated inputs used to train the GP. To reduce this cost, we can leverage the large literature about sparse representation techniques for GP [RW06]. Furthermore, with the increase in the number of control points, we face a larger dimensional search space, reflecting in an increased number of simulations needed to obtain an accurate representation of the robustness for optimization, with a consequent increase of computational time. We can partially improve on this problem, typical of naive implementations of the Bayesian approach, by refining the choice of the covariance function and/or constraining some of its hyperparameters so as to increment the exploration propensity of the search. In the future, we also plan to improve the adaptive approach which is in charge of increasing the control points of an input signal, with the goal of dropping control points that are not useful. In the current work, we use the GP-based sampling scheme to deal efficiently with falsification. However, our approach can be modified to deal with the coverage problem [DDD+15], i.e., the identification of a given number of counterexamples which are homogeneously distributed in the falsification domain. Our idea is to modify the sampling algorithm (Algorithm 7, line 7) by adding a memory of already visited areas, so to distribute samples homogeneously in space.

Concluding Remarks

This thesis combines formal methods and machine learning approaches to face the new challenges of model-based development. Our research starts from the consideration that even if formal methods are currently used in some specific areas, they are not used in many others where they might be successfully applied. As already discussed in the Introduction, the motivations are somewhat complex. First, there is a lack of knowledge in this area of computer science, for example, many engineers do not even encounter them in their academic studies. Second, standard model checking techniques are not able to deal with real industrial cases. In this thesis, machine learning approaches have been used to make formal methods applicable even in complex industrial scenarios by including, in addition, some common requirements of this context, such as the black box modeling and the minimization of the computational cost derived from model simulations.

Conclusions and future works have been already discussed at the end of each contribution. In this final chapter, we would like to focus on some section of this thesis which we think are more interesting, give some future directions and a critic review of some contributions.

Signal convolution logic, introduced in Chapter 5, is one of the most interesting contributions from the *formal methods side* that we presented in this thesis. But at the same time, I think, it needs more improvements than the other contributions. First of all its motivation is clear and reasonable, and goes in the direction of enriching the expressivity of "standard" temporal logic such as *Signal Temporal Logic* (STL) to manage the complexity of new models and systems, i.e., cyber-physical systems. Other authors had tackled this goal at different levels. In the falsification domain, for example, Takumi al. in [AH15] extend STL by introducing a novel logic called *Average Signal Temporal Logic* (AvSTL) which implements new average temporal operators. Their new language continues the mission which is well represented by the slogan: *expressivity of temporal logic should help falsification.* At the level of Boolean semantics, STL and AvSTL are identical. Differences arise at the level of quantitative semantics where time inhomogeneity can also be described. For example, the robustness of AvSTL can differentiate among two trajectories that both satisfies – *after heavy braking, the airbag must operate within 10 ms* – by giving more importance to the trajectory which realizes the airbag opening as soon as possible. Our approach is different and works at the level of Boolean semantic but shares with [AH15] the idea of increasing the expressivity of STL to achieve different purposes. A common problem of extending a language to get more expressivity is a consequent increase in the complexity of its verification monitoring. The algorithm we propose is rather simple and need improvements which have been planned for a future publication. Another common problem is related to its

usability. In our extension the interpretation of the flat kernel is rather simple and can find a natural domain of applicability, for example, $\langle \texttt{flat}_{[0,30min]}, 0.50 \rangle \varphi$ measures if in the next 30 minutes φ holds for at least 15 minutes in total. The introduction of other kernels is useful because permits to weight not uniformly the time where φ is satisfied. As an example, while measuring the glucose level in blood, it is more dangerous if its level is high just before a meal (see Chapter 5). To capture this, one could give different weights if the formula is satisfied, or not, at the end of a specific time interval, i.e., by consider an increasing exponential kernel such as $\langle \texttt{exp}_{[0,60min]}, 0.80 \rangle (\mathbf{G}(t) \geq 180)$, where $\mathbf{G}(t)$ is the glucose level, 180 is the hyperglycemia threshold, and the meal will be provided at the end of the next 60 min. A problem with these kinds of kernels is their usability. Indeed, if in the previous example it is quite obvious which type of kernel to consider (i.e., an increasing kernel), it is much less clear which kind of increasing function should be chosen (e.g., $\texttt{exp}[2](x)$ or $\texttt{exp}[3](x)$, etc.). An interesting use of signal convolution logic can be related to the classification of trajectories that we have discussed in Chapter 6. In that contribution, we have used *Parametric Signal Temporal Logic* (PSTL) to define a suitable classifier capable of classifying between anomalous and regular trajectories. We use a heuristics algorithm to solve this task and relay on the translation property of PSTL formula (i.e., translating all the atomic predicates of the same quantity the value of the robustness increase or decrease of the same amount as well). A simple improvement consists in using the monotonicity properties of PSTL formulas to learn the parameter of a formula. We think that the high expressivity of SCL can be useful in this domain and a parameterized version of SCL can even be used to identify suitable classifiers. A prerequisite is an efficient quantitative monitoring Algorithm for such language which is planned for future works.

At the level of falsification, the contribution of Chapter 9 combines the power of Gaussian processes with the expressivity of STL. The most exciting aspect, I think, is the black box assumption which permits to define a general approach to the falsification of block diagram models. This kind of assumption is usually adopted in the industrial sectors for different reasons already described in the introduction and Chapter 9. An essential requirement that we consider consists in achieving the falsification of a model with a minimum number of simulations. This requirement is mainly motivated by the typical situation, in the industrial fields, that simulations are very costly. The number of simulations is, for this reason, an essential parameter to considers but it is not the tightest we can find. We can indeed consider the total simulation time in an independent manner w.r.t the number of simulation. If we perform 100 simulations of one minute each, we perform a total of 100 minutes simulation which is better than 50 simulations of 3 minutes each (i.e., an overall simulation time of 150 minutes). Therefore, minimizing the number of simulations is a right goal if all the simulations have the same duration, or a mandatory requirement if we cannot predict the simulation time for different model configurations. A possible improvement of our methods consists in leveraging the GP-UCB algorithm and using online monitoring for STL robustness. At each iteration, the GP-UCB algorithm for minimization will sample a new point which has a higher probability to be lower than the minimum value found so far. However, it could happen that this point is not so interesting (i.e., its value is higher than the already achieved minimum). My guess is that it is not necessary to discover its precise value to permit the GP-UCB to find the minimum, but it should be enough to find that its value is

above a given threshold. This information can be further generalized to nearby points, and that particular area will no more be considered by the GP-UCB algorithm, which will explore new and more promising areas. The nice part is that the online monitoring of STL [DDG+17] can be used efficiently to discover such inequalities before the planned simulation ends, with a consequent saving of the total simulation time.

Bibliography

[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995. 99

[ADMN11] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, pages 147–160, 2011. 67, 68, 69

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996. 23, 78

[AFS+13] Houssam Abbas, Georgios E. Fainekos, Sriram Sankaranarayanan, Franjo Ivancic, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embedded Comput. Syst.*, 12(2s):95:1–95:30, 2013. 100

[AH15] Takumi Akazaki and Ichiro Hasuo. Time robustness in MTL and expressivity in hybrid system falsification. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, pages 356–374, 2015. 27, 49, 111

[Aka16] Takumi Akazaki. Falsification of conditional safety properties for cyber-physical systems with gaussian process regression. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 439–446, 2016. 100, 107, 108

[ALFS11a] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Proc. of TACAS*, 2011. 68

[ALFS11b] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011. 99, 100, 107

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. 67

[BBNS15] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. System design of stochastic models using robustness of temporal properties. *Theor. Comput. Sci.*, 587:3–25, 2015. 30, 31, 67, 78, 89, 90, 96

[BBS14a] Ezio Bartocci, Luca Bortolussi, and Guido Sanguinetti. Data-driven statistical learning of temporal logic properties. In *Formal Modeling and Analysis of Timed Systems - 12th International Conference, FORMATS 2014, Florence, Italy, September 8-10, 2014. Proceedings*, pages 23–37, 2014. 67, 68, 69, 70, 76

[BBS+14b] Sara Bufo, Ezio Bartocci, Guido Sanguinetti, Massimo Borelli, Umberto Lucangelo, and Luca Bortolussi. Temporal logic based monitoring of assisted ventilation in intensive care patients. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, pages 391–403, 2014. 67, 68, 69, 70, 72, 74, 75, 76

[BCH+11] Benoît Barbot, Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Efficient CTMC model checking of linear real-time objectives. In *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 128–142, 2011. 78

[Bec03] Kent Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003. 1

[BHHK03] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003. 27, 28

[BHLM13a] Luca Bortolussi, Jane Hillston, Diego Latella, and Mieke Massink. Continuous approximation of collective system behaviour: A tutorial. *Perform. Eval.*, 70(5):317–349, 2013. 15, 18, 82

[BHLM13b] Luca Bortolussi, Jane Hillston, Diego Latella, and Mieke Massink. Continuous approximation of collective system behaviour: A tutorial. *Perform. Eval.*, 70(5):317–349, 2013. 77

[Bil99] Patrick Billingsley. *Convergence of probability measures*. Wiley series in probability and statistics. Probability and statistics section. Wiley, 2nd ed edition, 1999. 17

[BMS15] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. U-check: Model checking and parameter synthesis under uncertainty. In *Quantitative Evaluation of Systems, 12th International Conference, QEST 2015, Madrid, Spain, September 1-3, 2015, Proceedings*, pages 89–104, 2015. 93

[BMS16] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous-time markov chains. *Information and Computation*, 247:235–253, 2016. 7, 35, 36, 78

[BPS16] Luca Bortolussi, Alberto Policriti, and Simone Silvetti. Logic-based multi-objective design of chemical reaction networks. In *Hybrid Systems Biology - 5th International Workshop, HSB 2016, Grenoble, France, October 20-21, 2016, Proceedings*, pages 164–178, 2016. 5, 7, 31, 78, 89

[Bra08] Fred Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008. 93

[BS18] Luca Bortolussi and Simone Silvetti. Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, pages 396–413, 2018. 7

[BvOC11] Gábor Balázsi, Alexander van Oudenaarden, and James J Collins. Cellular decision making and biological noise: from microbes to mammals. *Cell*, 144(6):910–925, 2011. 94

[BVP+16] Giuseppe Bombara, Cristian Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016*, pages 1–10, 2016. 67, 68, 69, 74, 75, 76

[CDKM11] Taolue Chen, Marco Diciolla, Marta Z. Kwiatkowska, and Alexandru Mereacre. Time-bounded verification of ctmcs against real-time specifications. In *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, pages 26–42, 2011. 28, 29

[CDKM13] T. Chen, M. Diciolla, M. Z. Kwiatkowska, and A. Mereacre. A simulink hybrid heart model for quantitative verification of cardiac pacemakers. In *Proceedings of HSCC*, pages 131–136. ACM, 2013. 68

[CDKP14] Milan Ceska, Frits Dannenberg, Marta Z. Kwiatkowska, and Nicola Paoletti. Precise parameter synthesis for stochastic biochemical systems. In *Computational Methods in Systems Biology - 12th International Conference, CMSB 2014, Manchester, UK, November 17-19, 2014, Proceedings*, pages 86–98, 2014. 6, 78

[CDP+17] Milan Ceska, Frits Dannenberg, Nicola Paoletti, Marta Kwiatkowska, and Lubos Brim. Precise parameter synthesis for stochastic biochemical systems. *Acta Inf.*, 54(6):589–623, 2017. 6, 78, 79, 80, 84, 85

[CFMS15] Fraser Cameron, Georgios E. Fainekos, David M. Maahs, and Sriram Sankaranarayanan. Towards a verified artificial pancreas: Challenges and solutions for runtime verification. In *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, pages 3–17, 2015. 60

[Cha17] D.K. Chaturvedi. *Modeling and Simulation of Systems Using MATLAB and Simulink*. CRC Press, 2017. 2

[CL04] Carlos A. C. Coello and Gary B. Lamont. *Applications of Multi-objective Evolutionary Algorithms*. Advances in natural computation. World Scientific, 2004. 89

[CMH06] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 325–336, 2006. 5, 89

[Coc02] Alistair Cockburn. *Agile software development*, volume 177. Addison-Wesley Boston, 2002. 1

[CPP+16] Milan Ceska, Petr Pilar, Nicola Paoletti, Lubos Brim, and Marta Z. Kwiatkowska. PRISM-PSY: precise gpu-accelerated parameter synthesis for stochastic systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pages 367–384, 2016. 78, 85

[DAPM02] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002. 5, 42, 91

[DDD+15] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V. Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, pages 127–142, 2015. 100, 109

[DDG+17] Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1):5–30, 2017. 113

[DDL+13] Alexandre David, Dehui Du, Kim Guldstrand Larsen, Axel Legay, and Marius Mikucionis. Optimizing control strategy using statistical model checking. In *NASA Formal Methods, 5th International Symposium, NFM 2013, Moffett Field, CA, USA, May 14-16, 2013. Proceedings*, pages 352–367, 2013. 78

[Deb01] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001. 41, 89

[Deb14] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014. 96

[DFM13] Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 264–279, 2013. 27, 55, 58

[DJJ+15] Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. Prophesy: A probabilistic parameter synthesis tool. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, pages 214–231, 2015. 78

[DJKM15] Jyotirmoy V. Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic local search for falsification of hybrid systems. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, pages 500–517, 2015. 100

[DM10] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings*, pages 92–106, 2010. 25, 27, 49, 52, 100

[Don10] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010. 27, 68, 100

[DTL16] DTL4STL. `http://sites.bu.edu/hyness/dtl4stl/`, 2016. 74

[EKVY07] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of markov decision processes. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, pages 50–65, 2007. 5, 89

[FKP12] Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, pages 317–332, 2012. 5, 89

[Fle13] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013. 38

[FP09] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009. 25, 49, 52, 53

[FSUY12] Georgios E. Fainekos, Sriram Sankaranarayanan, Koichi Ueda, and Hakan Yazarel. Verification of automotive control applications using s-taliro. In *American Control Conference, ACC 2012, Montreal, QC, Canada, June 27-29, 2012*, pages 3567–3572, 2012. 107

[FTHC14] Jie Fu, Herbert G. Tanner, Jeffrey Heinz, and Jane Chandlee. Adaptive symbolic control for finite-state transition systems with grammatical inference. *IEEE Trans. Automat. Contr.*, 59(2):505–511, 2014. 68

[GBCB16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 50

[GCC00] Timothy S Gardner, Charles R Cantor, and James J Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, 2000. 19, 92, 93

[Gil77] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. 17, 18, 91

[Gil01] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001. 17

[GMW81] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*. Academic press, 1981. 38

[Haa04] Peter J. Haas. Stochastic petri nets for modelling and simulation. In *Proceedings of the 36th conference on Winter simulation, Washington, DC, USA, December 5-8, 2004*, pages 101–112, 2004. 77

[HAF14] Bardh Hoxha, Houssam Abbas, and Georgios E. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, April 14, 2014 / ARCH@CPSWeek 2015, Seattle, WA, USA, April 13, 2015.*, pages 25–30, 2014. 100

[HBS73] Carl Hewitt, Peter Boehler Bishop, and Richard Steiger. A universal modular ACTOR formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, USA, August 20-23, 1973*, pages 235–245, 1973. 19

[HCC+04] Roman Hovorka, Valentina Canonico, Ludovic J Chassin, Ulrich Haueter, Massimo Massi-Benedetti, Marco Orsini Federici, Thomas R Pieber, Helga C Schaller, Lukas Schaupp, Thomas Vering, et al. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological measurement*, 25(4):905, 2004. 60

[HDF18] Bardh Hoxha, Adel Dokhanchi, and Georgios E. Fainekos. Mining parametric temporal logic properties in model-based design for cyber-physical systems. *STTT*, 20(1):79–93, 2018. 68

[HF10] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, 2010. 1

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994. 27

[HJW11] Thomas A. Henzinger, Barbara Jobstmann, and Verena Wolf. Formalisms for specifying markovian population models. *Int. J. Found. Comput. Sci.*, 22(4):823–841, 2011. 15

[HKM08] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS 2008, Barcelona, Spain, 30 November - 3 December 2008*, pages 173–182, 2008. 78

[IHS14] Malte Isberner, Falk Howar, and Bernhard Steffen. The TTT algorithm: A redundancy-free approach to active automata learning. In *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, pages 307–322, 2014. 67

[JDDS15] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. Mining requirements from closed-loop control models. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(11):1704–1717, 2015. 67

[JDK+14] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Kenneth R. Butts. Powertrain control verification benchmark. In *17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC'14, Berlin, Germany, April 15-17, 2014*, pages 253–262, 2014. 100

[JL11] Sumit Kumar Jha and Christopher James Langmead. Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. *Theor. Comput. Sci.*, 412(21):2162–2187, 2011. 78

[JRSS18] Susmit Jha, Vasumathi Raman, Dorsa Sadigh, and Sanjit A. Seshia. Safe autonomy under perception uncertainty using chance-constrained temporal logic. *J. Autom. Reasoning*, 60(1):43–62, 2018. 49

[Kat16] Joost-Pieter Katoen. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 31–45, 2016. 78

[KJB17] Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Trans. Automat. Contr.*, 62(3):1210–1222, 2017. 67, 68

[Knu92] Donald E Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992. 52

[KSK12] John G. Kemeny, J. Laurie Snell, and Anthony W. Knapp. *Denumerable Markov chains: with a chapter of Markov random fields by David Griffeath*, volume 40. Springer Science & Business Media, 2012. 12

[Kur70] Thomas G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of applied Probability*, 7(1):49–58, 1970. 18

[LS16] Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2016. 3, 99

[MA04] R. Timothy Marler and Jasbir S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004. 41

[Mat] MathWorks. *Simulink®*. https://mathworks.com/. 2, 19

[MBC79] Michael D. McKay, Richard J. Beckman, and William J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. 39, 101

[Mie99] K. Miettinen. *Nonlinear Multiobjective Optimization*. International Series in Opefrations Research & Management Science. Springer US, 1999. 41

[MMP91] Oded Maler, Zohar Manna, and Amir Pnueli. Prom timed to hybrid systems. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 447–484. Springer, 1991. 99

[MN04] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*, pages 152–166, 2004. 23, 24, 25, 27, 54, 55, 78, 79

[Mod] Modelica Association et al. *Modelica®*. https://www.modelica.org/. 19

[Mon17] Douglas C Montgomery. *Design and analysis of experiments*. John wiley & sons, 2017. 39

[Nat] National Instrument. *LabVIEW®*. https://www.ni.com/. 2, 19

[NKJ+17] Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy V. Deshmukh, Ken Butts, and Taylor T. Johnson. Abnormal data classification using time-frequency temporal logic. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, April 18-20, 2017*, pages 237–242, 2017. 67

[Nor98] James R. Norris. *Markov chains*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998. 15

[Pnu77] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57, 1977. 23

[RBNG17] Alena Rodionova, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. Temporal logic as filtering. In *Dependable Software Systems Engineering*, pages 164–185. 2017. 27, 49

[RK13] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Information Science and Statistics. Springer New York, 2013. 101

[RN10] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010. 107

[Rud98] Günter Rudolph. Evolutionary search for minimal elements in partially ordered finite sets. In *Evolutionary Programming VII, 7th International Conference, EP98, San Diego, CA, USA, March 25-27, 1998, Proceedings*, pages 345–353, 1998. 43

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. 33, 36, 78, 85, 109

[SD94] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994. 42

[SF12] Sriram Sankaranarayanan and Georgios E. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012*, pages 125–134, 2012. 102, 107

[SHI12] Bernhard Steffen, Falk Howar, and Malte Isberner. Active automata learning: From dfas to interface programs and beyond. In *Proceedings of the Eleventh International Conference on Grammatical Inference, ICGI 2012, University of Maryland, College Park, USA, September 5-8, 2012*, pages 195–209, 2012. 67

[SK16] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016*, 2016. 49

[SKC+17] Sriram Sankaranarayanan, Suhas Akshar Kumar, Faye Cameron, B. Wayne Bequette, Georgios E. Fainekos, and David M. Maahs. Model-based falsification of an artificial pancreas control system. *SIGBED Review*, 14(2):24–33, 2017. 61, 67

[SKKS12] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Trans. Information Theory*, 58(5):3250–3265, 2012. 72

[SNBB17] Simone Silvetti, Laura Nenzi, Luca Bortolussi, and Ezio Bartocci. A robust genetic algorithm for learning temporal specifications from data. *CoRR*, abs/1711.06202, 2017. 7

[SNBB18] Simone Silvetti, Laura Nenzi, Ezio Bartocci, and Luca Bortolussi. Signal convolution logic. *arXiv preprint arXiv:1806.00238*, 2018. 7

[SPB+17a] Fedor Shmarov, Nicola Paoletti, Ezio Bartocci, Shan Lin, Scott A. Smolka, and Paolo Zuliani. Smt-based synthesis of safe and robust PID controllers for stochastic hybrid systems. In *Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference, HVC 2017, Haifa, Israel, November 13-15, 2017, Proceedings*, pages 131–146, 2017. 60

[SPB17b] Simone Silvetti, Alberto Policriti, and Luca Bortolussi. An active learning approach to the falsification of black box cyber-physical systems. In *Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings*, pages 3–17, 2017. 6, 7, 67, 100

[Ste01] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001. 36

[TTS+08] Huan-Liang Tsai, Ci-Siang Tu, Yi-Jie Su, et al. Development of generalized photovoltaic model using matlab/simulink. In *Proceedings of the world congress on Engineering and computer science*, volume 2008, pages 1–6. San Francisco, USA, 2008. 2

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 327–338, 1985. 28

[Vin98] B. Vinnakota. *Analog and Mixed-signal Test*. Prentice Hall PTR, 1998. 99

[WHSX16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pages 370–378, 2016. 85

[WN15] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1775–1784, 2015. 85

[XJ18] Zhe Xu and A. Agung Julius. Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Trans. Automation Science and Engineering*, 15(1):264–277, 2018. 68

[YHF12] Hengyi Yang, Bardh Hoxha, and Georgios E. Fainekos. Querying parametric temporal logic properties on embedded systems. In *Testing Software and Systems - 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, November 19-21, 2012. Proceedings*, pages 136–151, 2012. 67

[YS06] Håkan L. S. Younes and Reid G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409, 2006. 29

[ZKH03] Q. Zhao, B.H. Krogh, and P. Hubbard. Generating test inputs for embedded control systems. *IEEE control systems*, 23(4):49–57, 2003. 99

[ZPC13] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. *Formal Methods in System Design*, 43(2):338–367, 2013. 29

[ZQL+11] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011. 41, 89

[ZRWT17] Jun Zhou, R. Ramanathan, Weng-Fai Wong, and P. S. Thiagarajan. Automated property synthesis of odes based bio-pathways models. In *Proc. of CMSB*, pages 265–282, 2017. 67

[ZSD+15] Aditya Zutshi, Sriram Sankaranarayanan, Jyotirmoy V. Deshmukh, James Kapinski, and Xiaoqing Jin. Falsification of safety properties for closed loop control systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, pages 299–300, 2015. 67

[ZSKP13] Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 462–470, 2013. 91, 98