Falsification of Cyber-Physical Systems with Constrained Signal Spaces

Benoît Barbot[1], Nicolas Basset[2], Thao Dang[2], Alexandre Donzé[3], James Kapinski[5], Tomoya Yamaguchi[4]

[1] Univ Paris Est Creteil, LACL, F-94010 Creteil, France
[2] VERIMAG/CNRS, Université Grenoble Alpes, France
[3] Decyphir SAS, France
[4] Toyota Motors North America R&D, USA
[5] Unaffiliated

Falsification has garnered much interest recently as a way to validate complex CPS designs with respect to a specification expressed via temporal logics. Using their quantitative semantics, the falsification problem can be formulated as robustness minimization problem. To make this infinite-dimensional problem tractable, a common approach is to restrict to classes of signals that can be defined using a finite number of parameters, such as piecewise-constant or piecewise-linear signals with fixed time intervals). A major drawback of this approach is that when the input signals must satisfy non-trivial temporal constraints, encoding these constraints into bounded domains for parameters can be difficult. In this work, to better capture temporal constraints on the input signal space, we use timed automata (TA) and make use of a transformation that allows sampling TA traces by sampling points in the unit box. We exploit this transformation to efficiently encode constrained CPS signals in the robustness minimization problem. This transformation also allows us to define an effective coverage measure of the constrained signal space so as to provide quantitative guarantees when no falsifying behaviour is found. In addition, this coverage is used to improve the black-box optimisation performance by detecting situations where the search is stuck near a local optimum. The approach is demonstrated on a $\Delta\Sigma$ modulator and a model of car automatic transmission subject to constraints describing usual driving patterns.

1 Introduction

Cyber-physical systems (CPS) are found in many safety-critical applications, like aircraft, medical devices, and automobiles, hence it is vital that they behave in a manner consistent with their design expectations. CPS models are growing rapidly in complexity and size, which often go beyond the scalability of formal verification techniques based on exhaustive analysis. As of today, industrial validation is carried out mostly by sampling a finite number of input stimuli and checking the corresponding behaviors obtained by model simulation or system execution.

Another approach to CPS validation is requirement falsification using black-box optimization. Falsification can be thought of as testing where requirements are expressed in a formal specification language such as metric temporal logic (MTL) and signal temporal logic (STL) [32, 35], which are appropriate for specifying behaviors defined using real-valued signals over dense time. A key feature of such logics is that they are equipped with *quantitative* semantics, and for a given behavior, a real value, called the *robustness*,

quantifies the property satisfaction level of the behavior [20, 24]. Using such semantics, the falsification problem can be formulated as a robustness minimization problem, so as to automatically find behaviors that violate (falsify) the property. Falsification techniques have been applied to many CPS systems and are finding applications in industry (see a recent survey [9]), with the development of tools like S-TaLiRo and Breach [4, 17]. This *optimization-based approach* is faced with the following major challenges.First, the existing optimization solvers expect decision variables in a space of finite dimension whereas the search space of the CPS falsification problem includes continuous-time input signal spaces of infinite dimension. This gives rise to the problem of encoding CPS signal spaces. A common practice so far (initiated in [17, 37]) is to restrict to classes of signals that can be defined using a finite number of parameters. Another major challenge is that, for cases where the inputs must satisfy non-trivial temporal constraints, encoding these constraints into bounded domains for parameters can be difficult. Ad hoc rejection sampling methods become inefficient when the portion of signals satisfying the constraints is small. Also, the cost function (that is, the robustness) is often non-convex and contains discontinuities, and for such problems, in general there are no algorithms that can guarantee to find a global optimum. Hence, we can see the importance of approximation quality measures of behavior sets, in order to quantify how complete the search is. Therefore another challenge is to define meaningful coverage measures. When the input signals are subject to complex temporal constraints, the resulting constrained signal space may be difficult to encode and measure.

In this paper we address the above challenges in the following two ways: (1) introducing in the optimization-based falsification framework a new encoding of input signal spaces subject to temporal constraints specified using timed automata [3]; (2) this encoding method also leads to a new coverage measure for constrained signal space, which we use to improve the efficiency of an iterative black-box optimization procedure. For clarity of explanation, before describing our contributions and comparing them with the current state of the art, we provide an overview of the existing approaches and their limitations.

2 Requirement Falsification Problem

CPS Models and Specification. We model the behaviors of a CPS using the following input-output mapping:

$$y = \mathcal{F}(u), \tag{1}$$

where $u \in \mathcal{U}$ is a function of time that represents the input signals to the system, that is $u : \mathcal{I} \to U$, where $\mathcal{I}$ is an interval of the form $[0, T]$ with $T \in \mathbb{R}_{>0}$, and U is some metric space of finite dimension. Note that initial conditions as well as other parameters (some finite set of variables influencing the system's behavior) can be captured as constant input signals. Similarly, we assume that each output signal $y \in \mathcal{Y}$ is a function $\mathcal{I} \to Y$, where Y is some metric space of finite dimension. To specify the correct or expected behaviors for the system (1) in an unambiguous form that can be efficiently measured and quantified, we use the Signal Temporal Logic (STL) language [35].

Overview of STL. An STL formula φ consists of atomic predicates along with logical and temporal connectives. Atomic predicates are defined over signal values and have the

form $f(y(t)) \sim 0$, where f is a scalar-valued function over the signal y evaluated at time t, and $\sim \in \{<, \leq, >, \geq, =, \neq\}$. Temporal operators "always" ($\Box$), "eventually" ($\Diamond$), and "until" ($\mathcal{U}$) have the usual meaning and are scoped using intervals of the form (a, b), $(a, b]$, $[a, b)$, $[a, b]$, or (a, ∞), where $a, b \in \mathbb{R}_{\geq 0}$ and $a < b$. If I is a time interval, the following grammar defines the STL language.

$$\varphi \ := \ \top \mid f(y(t)) \sim 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 : \quad \sim \in \{<, \leq, >, \geq, =, \neq\} \tag{2}$$

The $\Diamond$ operator is defined as $\Diamond_I \varphi \triangleq \top \mathcal{U}_I \varphi$, and the $\Box$ operator is defined as $\Box_I \varphi \triangleq \neg(\Diamond_I \neg\varphi)$. When omitted, the interval I is taken to be $[0, \infty)$. Given a signal y and an STL formula φ, we use the quantitative semantics for STL, which is defined formally in [20]. The quantitative semantics defines a function ρ such that a positive sign of $\rho(\varphi, y, t)$ indicates that (y, t) satisfies φ, and its absolute value estimates the *robustness* of this satisfaction. If φ is an inequality of the form $f(y) > b$, then its robustness is $\rho(\varphi, y, t) = f(y(t)) - b$. When t is omitted, we assume $t = 0$ (i.e., $\rho(\varphi, y) = \rho(\varphi, y, 0)$). For the conjunction of two formulas $\varphi := \varphi_1 \wedge \varphi_2$, we have $\rho(\varphi, y) = \min(\rho(\varphi_1, y), \rho(\varphi_2, y))$, while for the disjunction $\varphi := \varphi_1 \vee \varphi_2$, we have $\rho(\varphi, y) = \max(\rho(\varphi_1, y), \rho(\varphi_2, y))$. For a formula with until operator as $\varphi := \varphi_1 \mathcal{U}_I \varphi_2$, the robustness is computed as $\rho(\varphi, y) = \max_{t' \in I} \left(\min\left(\rho(\varphi_2, y, t'), \min_{t'' \in [t, t']} (\rho(\varphi_1, y, t''))\right)\right)$.

Falsification Problem. Given a system model such as (1) and a requirement φ specified as an STL formula, we want to find an input $u \in \mathcal{U}$ such that $y = \mathcal{F}(u)$ does not satisfy φ, denoted $y \not\models \varphi$. Such a behavior y is called a counter-example, which is identified when $\rho(\varphi, y) < 0$. This is usually solved by formulating the following optimization problem:

$$\min_{u \in \mathcal{U}} \rho(\varphi, y) \ s.t. \ y = \mathcal{F}(u)$$

This formulation has been the focus of numerous research [9]. We next discuss the challenges in solving this optimization problem and some existing approaches.

Input Signal Encoding. The input signals are taken from an infinite-dimensional space (*i.e.*, they can be a partial functions over a continuous time-domain), one thus needs a finite encoding of the signals. As mentioned earlier, most of the existing approaches restrict to classes of input signals that are *finitely parameterizable*, that is, input signals u can be uniquely characterized by a finite set of parameters. Therefore, the infinite-dimensional optimization problem (3) becomes finite-dimensional. For example, a right-continuous piecewise constant input signal u with discontinuities occurring at monotonically increasing instants $t_1, \ldots, t_m$ where $0 = t_1 < t_m < T$, can be uniquely characterised by m values $v_i = u(t_i)$. By fixing the number m of time intervals, the time points $t_1, \ldots, t_m$ and the corresponding signal values are the decision variables for the search.

Minimizing the Robustness. Fixing an input signal parametrization, the optimization problem (3) becomes finite-dimensional but is still challenging for a number of reasons. First, the input-output mapping $\mathcal{F}$ is not specified explicitly; rather, it enforces that y is the output signal of the dynamical system model $\mathcal{F}$, given the input signal u. As $\mathcal{F}$ can be a nonlinear hybrid system modelled using heterogenous formalisms (such as Simulink/Stateflow), the output y can only be determined approximately using numerical simulation.

This also gives rise to the hard problem of determining the gradients of the cost function, often required by traditional continuous optimization techniques. Additionally, the cost function ρ is often non-convex and contains discontinuities. For such problems, in general there are no algorithms that can guarantee to find a global optimum [26]. Hence, the robustness minimization step is often done using the black-box optimization approach because it does not require derivative information [38]. This approach relies on the search techniques, called metaheuristics [21], which aim to combine the strengths of existing algorithms for discrete and continuous domains. Such a search consists of a sequence of moves from one candidate solution to another. In each move if the candidate satisfies the falsification goal, a counter-example is found, otherwise, the candidate is updated. The updating heuristics in general perform well for simple search spaces, for instance, multi-dimensional boxes, or linear algebraic constraints [38]. This is one reason why in practice the input signal parametrization is often chosen in such a way that induces a search space that is a box. These essential ideas are summarized by the following abstract algorithm.

Algorithm 1 Optimization-based Falsification Algorithm

$k = 1$, $\rho_m = +\infty$
Select a set $\mathcal{U}_s \subset \mathcal{U}$ of input signals
repeat
  $\rho_m = \min\{\rho_m, \min_{u \in \mathcal{U}_s}\{\rho(\varphi, y) \mid y = \mathcal{F}(u)\}\}$
  if $\rho_m < 0$ **then**
    Report the falsifying behavior. Exit
  end if
  $k = k + 1$
  $\mathcal{U}_s = Update(\mathcal{U}_s)$ ▷ (using black-box optimization)
until $k = K_{max}$
No falsifying behavior found.

Quantitative Guarantees. When no falsifying behavior is found, it is of great interest to provide a quantitative guarantee expressed by a measure of the set of behaviors that was tested. Such a 'coverage' measure was proposed only for point spaces (see related work on coverage measures in Section 5), which are appropriate only for properties defined over the system states (such as, safety). It is thus useful to use a more general notion of signal/function space coverage, a problem that we address in this work.

Limitations of the Existing Solutions and Our Approach. Concerning signal encoding: using fixed parametrizations restricts the searchable space, and the falsification performance depends on the selected parametrizations, which requires validation engineers to use intuition to select the number of intervals and their duration. Furthermore, as mentioned in the introduction, input signals in practical applications are often subject to constraints imposed by their generators. Examples of such signals include noises from specific environments or controls from under-actuated controllers. In these cases the input signals must satisfy non-trivial temporal constraints, and encoding these constraints in the forms that can be efficiently handled by the existing optimizers can be difficult; the optimizers

often treat such constraints using ad hoc manners, such as using rejection sampling. Little attention has been given to these considerations in the falsification-related literature but [16, 39] propose some strategies that involve incrementally increasing the number of time intervals. If these constraints are not taken into account, there are two consequences. First the optimizers can come up with trivial non-realistic solutions, such as Zeno behaviors switching between extreme values. Second, the unconstrained search space may be too conservative compared to the valid search space, which makes rejection sampling inefficient, as we will show in an example involving a rather intuitive temporal constraint.

In this work, to capture temporal constraints on the input signal space, we use timed automata (TA) [3]. Such constraints are previously considered in a procedure to uniformly generate random signals [6], which relies on the calculation of a transformation from the unit box to timed polytopes (allowing sampling timed words of a TA by sampling points in the unit box) [5]. We extend this transformation to encode constrained input signal spaces, which constitutes a crucial ingredient in the optimization process. Unlike the work [6] where the falsification process is based on a given set of uniformly sampled timed words, in this work we perform optimization in a search space that satisfies both signal timed pattern and value constraints. In other words, this encoding allows us not only to consider signals uniformly but also to perform best-case search strategies according to an objective function, which enhances the falsification performance as shown by the experimental results. This transformation also allows us to define an effective coverage measure of the constrained signal space in order to provide quantitative guarantees. In addition, this coverage will be used to improve the black-box optimization performance by detecting situations where the search is trapped near a local optima and to make online decisions about when and how to switch from one optimization strategy to another.

The remainder of the paper is organized as follows. In Section 3 we briefly recall timed automata [3] and the transformation from the unit box to timed polytopes [5, 6]. We then show how this transformation can be used to encode constrained signals and to define coverage measures for the space of such signals. Section 5 describes the falsification algorithm and Section 6 presents our experimental results.

3 Preliminaries on Timed Automata and Timed Word Generation

3.1 Timed Automata

A *timed automaton* $\mathcal{A} = (Q, X, \Sigma, \Delta, Inv, i_0)$ is a tuple where Q is a finite set of locations with i_0 as initial location; X is a finite set of clocks which are assumed bounded by a constant $M \in \mathbb{N}$; Δ is a finite set of transitions. Each transition is the form $\delta = (q, \psi, a, r, q')$ where $q, q' \in Q$ are the source and destination locations; ψ is the guard, and $a \in \Sigma$ is a label; r is the reset map; Inv associates with each location q a conjunction of clock constraints, called the invariant of q. A state of $\mathcal{A}$ is a pair $(q, \boldsymbol{x})$ where $q \in Q$ and $\boldsymbol{x}$ is a clock valuation[6]. The transitions of the automaton are of two types: timed transitions and discrete transitions. Timed transitions correspond to the evolution of the clocks within a location as long as the clock valuation satisfies the invariant of the location. Concerning discrete transitions, if the transition $\delta = (q, \psi, a, r, q')$ is enabled at the state $(q, \boldsymbol{x})$ (that

[6] A clock valuation, denoted by the letter $\boldsymbol{x}$ in bold, is a vector of clock values, while x_i denotes the i^{th} clock of the automaton, as in Fig. 1

is $\boldsymbol{x}$ satisfies the guard ψ), the discrete transition from q to q' can take place (if the clock valuation after applying the reset map r satisfies the invariant $Inv_{q'}$ of q'). The *reset* map r is determined by a subset of clocks $B \subseteq X$ and this transition resets to 0 all the clocks in B and does not modify the other clocks. The initial state of $\mathcal{A}$ is $(i_0, \mathbf{0})$. A *trace* is an alternating sequence $(i_0, \boldsymbol{x}_0) \xrightarrow{a_1,\tau_1} (q_1, \boldsymbol{x}_1) \dots \xrightarrow{a_n,\tau_m} (q_m, \boldsymbol{x}_m)$ of states and timed transitions with the following updating rules: q_i is the successors of q_{i-1} by transition δ_i, the vector $(\boldsymbol{x}_{i-1} + (\tau_i, \dots, \tau_i))$ must satisfy the guard of δ_i and applying the reset map to it gives $\boldsymbol{x}_i$. This trace is *labelled* by the *timed word* $\gamma = (\tau_1, a_1), \dots, (\tau_m, a_m)$ where a_i are transition labels and τ_i are *time delays* between two consecutive transitions, $(\tau_1, \dots, \tau_m)$ is called a *timed vector* and $(a_1, \dots, a_m)$ a *discrete pattern*. Given a discrete path $\alpha = \delta_1, \dots, \delta_n$ of $\mathcal{A}$ the set of timed vectors $(t_1, \dots, t_n) \in [0, M]^n$ such that $(i_0, \mathbf{0}) \xrightarrow{t_1,\delta_1} (q_1, t_1) \dots \xrightarrow{t_n,\delta_n} (q_n, t_n)$ is called the *timed polytope* associated to the path α. The set of timed words that label all the traces from the initial state is called the *timed language* of $\mathcal{A}$. As an example, we consider the TA in Fig. 1 that will be used in our

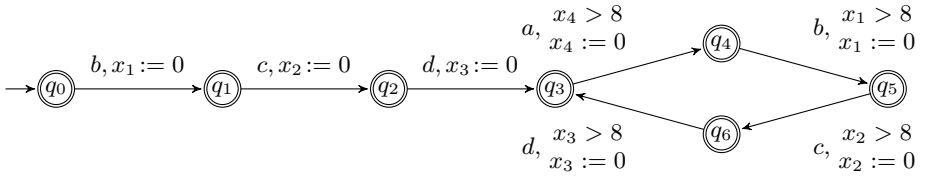

Fig. 1. A timed automaton used in our experiments. To avoid overloading the figure, a global clock x (reset to 0 at each transition) and the global invariants $x_1, x_2, x_3, x_4 < 12$ and $x < 4$ are not depicted.

experiments. This automaton models a quasi-periodic pattern of signals with uncertain period ranging between 8 and 12. It has the property that after entering the cycle the time lapse between 4 consecutive transitions is contained in the interval $[8, 12]$. Intuitively, the traces of this automaton are loosely periodic as transitions cannot be taken too early or too late. Moreover the global invariant condition $x < 4$ (not depicted in Fig. 1) ensures that each duration is bounded from above by 4. An example of timed word in the timed language is $(3.4, b)(3.6, c)(1.1, d)(2.3, a)(3.3, b)$.

3.2 Transformation from the unit box to a timed polytope

We want that the exploration within the domains of optimization variables reflects the exploration within the timed language, in terms of coverage. To this end, we will use a volume-preserving transformation developed in [5, 6], which is briefly recalled in the following. The reader is referred to a more detailed summary of the method in the Appendix.

From a timed automaton $\mathcal{A}$ we can define inductively the volume v_n of the language of words of length n accepted by the automaton. Indeed, in [5] we show that by decomposing the automaton into a zone graph with additional constraints ensuring that guards of the automaton are linear, the volume v_n can be written as a ratio of two polynomials and computed efficiently (in polynomial time). Then, the transformation in question is

defined as the cumulative probability distributions for sequentially sampling each transition and time delay. This is exactly the transformation that we need to encode a signal space constrained by a timed automaton. Indeed, to generate a timed word of length n, one starts with a sequence $(u_i)_{i=1}^{2n} \in [0,1]$ of real values, which corresponds to a point in the unit box of dimension $2n$. Starting from the initial state of the automaton and the clock valuation equal to $\mathbf{0}$, the transition and the delay at step i are chosen using the inverse transform method for the distribution sampling with the reals (u_{2i}, u_{2i+1}) over the distribution indexed by $(n-i)$. We also remark that in previous work [6] three tools were used to perform this sampling: Prism [34] for computing the zone graph, SageMath [42] for computing distributions and Cosmos [7] for the sampling. In the present work, the tool WordGen [8] combining the three steps has been developed which greatly increases the usability of the method.

4 Encoding Constrained Signal Space in the Optimization Problem

A timed automaton can naturally provide a qualitative description, annotated with timing information, for a class of CPS signals of interest. In addition, we can consider quantitative constraints on signal values by associating them with the transitions labels of the automaton. More concretely, each transition label a is associated with a predicate of the form $\pi_a(v) \leq 0$ where $v \in \mathbb{R}$ is the signal value.

To perform optimization over the space of such signals, we need an efficient representation of this space. For simplicity of explanation, we focus only on the signals corresponding to the timed words of $\mathcal{A}$ having a single discrete pattern $\alpha = (a_1 \cdots a_m)$. The timed polytope $\mathcal{P}_\tau$, defined by the delays τ between the transitions that are subject to the clock constraints (imposed by the guards, resets and invariants along the transition sequence), is the search space for timed words with the fixed pattern α. To couple it with the search space for signal values, we couple $\mathcal{P}_\tau$ with the set of signal values satisfying the associated predicates: $\mathcal{P}_v = \{v \mid \forall i \in \{1,\ldots,m\}\ \pi_{a_i}(v) \leq 0\}$. In this work, we assume that each π_{a_i} is an interval predicate[7], and the set $\mathcal{P}_v$ is thus a box, called a *valued box*. Hence, this coupling of time and value constraints leads to a polytope in $\mathbb{R}^{2m}$: $\mathit{\Pi} = \{(\tau, v) \mid \tau \in \mathcal{P}_\tau \wedge v \in \mathcal{P}_v\}$, called a *timed-valued polytope*. The signal constructed from any point (τ, v) in $\mathit{\Pi}$ is guaranteed to satisfy the constraints specified by the timed automaton $\mathcal{A}$ and its associated predicates. Thus the constrained signal space in question is encoded by this *timed-valued polytope*.

To generate candidate solutions from a timed-valued polytope, as mentioned earlier, we make use of the transformation that maps the unit box to this timed polytope and extend it to a timed-valued polytope, in order to reduce the search space to a box domain (instead of complex polytopic domains). Indeed, since a timed-valued polytope $\mathit{\Pi}$ is the product of a timed polytope $\mathcal{P}_\tau$ and a valued box $\mathcal{P}_v$, it is not hard to see that the transformation for $\mathit{\Pi}$, denoted by $\mathcal{S}$, is composed of $\mathcal{S}_\tau$ for the timed polytope $\mathcal{P}_\tau$ and $\mathcal{S}_v$ for the valued box $\mathcal{P}_v$. Note that $\mathcal{S}_v$ is simply an affine function transforming $\mathcal{P}_v$ to the unit box $[0,1]^m$. In short, using the transformation $\mathcal{S}$, the initial search domain, which is a timed-valued polytope, becomes the unit box $[0,1]^{2m}$. Let us write it as the product of two unit boxes, that is $\mathcal{B}_\tau \times \mathcal{B}_v$.

[7] Using more general predicates, such as linear predicates, leads to a more complicated problem of defining the transformation from the unit box, which we plan to consider in future work. This is indeed related to the problem of uniform sampling within a convex polytope.

This transformation was implemented in the tool WordGen to generate a timed word from a point in the unit box $\mathcal{B}_\tau$. Then, to construct CPS signals corresponding to a given timed word, we use the tool Breach [18]. This tool is also used to simulate the system behaviours and evaluate their robustness. To recap, the input signal construction is done as follows:

1. Pick a point p_τ in the unit box $\mathcal{B}_\tau$. Pick a point p_v in the unit box $\mathcal{B}_v$.
2. Use WordGen to generate a timed word w from p_τ.
3. Use Breach to generate a signal u from w and p_v.

Note that the above first step is done by the procedure of updating candidate input signals. This procedure is based on a combination of metaheuristics that we discuss in the sequel.

5 Guided Combination of Metaheuristics

One natural strategy for updating candidate solutions is to use methods related to gradient descent, wherein new points are selected based on some estimate of the gradient of the cost function near promising previously evaluated points. Such a descent strategy may not lead to a global optimum, leaving the search stuck around a local optimum. When this occurs, it is possible to restart the search from a new set of candidate solutions, but this can become expensive when there are many local optima. Metaheuristics [21] are one way to go about this problem, by accepting from time to time candidates that do not improve the cost function value. In this work we propose a method for combining a number of well-known metaheuristics. The method switches between two different types of solvers or search algorithms which, borrowing the terminology from [11, 21], are called exploitation-driven and exploration-driven.

The exploitation-driven algorithms try to make greedy changes (often small) around the current candidate. We make use of a number of well-known solvers in this type[8], namely Simulated Annealing [31], Global Nelder-Mead algorithms [2, 36], and CMAES (Covariance Matrix Adaptation Evolution Strategy) [28]. This type of solver is used to explore locally around promising candidates. On the other hand, the exploration-driven solvers explore the parameter space widely, and thus quickly enlarge the exploration space. Such solvers are particularly useful to help the search escape a local optimum, where the cost value has stagnated. The exploration-driven solver we use in this work is based on the low-discrepancy and uniform sampling method in [6].

It is of great interest to be able to synergize exploration and exploitation by adaptive switching between the two strategies using appropriate measures for exploitation and exploration performance. The trade-offs between exploitation and exploration have been explored for the purposes of falsification for CPS [33]. Exploitation performance can be measured by the reduction in the cost value (that is the robustness value). Exploration performance can be measured using the notion of search space coverage. For our framework, we introduce in the subsequent section a *signal space coverage measure*.

[8] The exploitation-driven and exploration-driven characterization refers only to the behaviors of the solvers seen on a global level, since the above-mentioned metaheuristics contain both exploitation-driven and exploration-driven aspects.

5.1 Signal Space Coverage Measure

We define a signal space coverage measure based on a partition of the variable domains, called *cell occupancy*. A similar measure was already used in our previous work [1] but was restricted to the parameter space corresponding to the space of signal values over fixed time parametrizations. Equiped with the transformation from the unit box, we can now extend it to signals. Let G be a partition of the unit box $[0,1]^{2m}$ into N_t rectangular cells with equal side length. Cell occupancy is based on the ratio between the number N_o of cells occupied by points and the total number N_t of cells. Then, the cell occupancy measure is given as $\frac{\log N_o}{\log N_t}$. Logarithm functions are used because the total number of cells could be very large as compared to the number of occupied cells. A major advantage of the cell-occupancy measure is that it is easy to compute; however, it is clear that when the cell size is large this measure does not reflect levels of uniformity or equi-distributivity as the Kolmogorov-Smirnov statistic [6].

Related Work on Coverage Measures. In the context of CPS a signal space coverage measure should be defined over continuous-time signals, such as the input signal space or the system behaviour space. The latter option is more difficult because the space of all possible system behaviours is in general unknown. When an input signal space is finitely parameterized, a point coverage measure can be defined on its associated parameter space. Measures like *dispersion* try to capture the size of the empty space between points that have been explored [23]. A related and simple measure, partitions the search space into cells and measures the proportion of cells that are occupied by explored points [41]. This method is related to the combinatorial entropy notion from the domain of physics to measure the degree of randomness in a distribution of points [27]. The *star discrepancy* measure from statistics to measure the degree to which a set of points are equidistributed [29], was also used for measuring the coverage of reachable states [1, 15, 22]. In this work where the specification imposes on the input signals complex temporal constraints, the resulting parameter space is difficult to define. However, using the above-described volume preserving transformation any point coverage can be defined over the unit box and carried over to the signal space. Hence, we can use in principle any existing point coverage. In this work, we choose to use the cell occupancy measure, since it can be efficiently computed for high dimensional search spaces encountered in our case studies.

5.2 Algorithm for Guided Combination of Metaheuristics

We describe our algorithm for guided combination of metaheuristics, summarized in Algorithm 2. The guiding is based on the robustness and coverage measures.

The algorithm is organized in iterations, and in each iteration the solvers (or metaheuristics) are sequentially called, based on the current search results. Throughout the search process, we maintain a set G of *intermediate visited states*. By 'visited state', we mean the pair (p, ρ) where p is a candidate point (in the search domain which is the unit box) and ρ is its associated cost value, and by 'intermediate' we mean the points successively computed by the solver scheme. The procedure starts with $Exploitation$, which runs each of the exploitation-driven solvers and updates the set G of visited states. Then $updateCoverage$ updates the coverage c of G (using the cell-occupancy measure).

Algorithm 2 Abstract Algorithm for Combining Metaheuristics

▷ s: solver index; $\mathcal{S}_\rho$: set of exploitation-driven solvers; G: set of visited states; ρ^* and c: sequences such that $\rho^*[k]$ and $c[k]$ are respectively the best robustness value and the coverage value up to iteration k

$k = 1$
while $k \leq k_{max}$ **do**
  $\{\rho^*, G\} = Exploitation(\mathcal{S}_\rho, G)$ ▷ run all the exploitation-driven solvers
  $c = updateCoverage(c, G)$
  $blocking = DetectBlocking(c, \rho^*)$ ▷ based on coverage and robustness
  if $(blocking)$ **then**
    $s = Rand$
    $(\rho^*, G) = Run(s, T_s)$ ▷ run a sampling-based solver for T_s time
  end if
  $k++$
end while

Next, the procedure $DetectBlocking$ determines whether the search has entered a blocking situation. If it has, the exploration-based search $Rand$, using quasi-random (that is, low-discrepancy) or uniform methods, is run for T_s seconds.

Switching to Exploration to Escape a Local Minimum. The search is said to be *blocking*, if it does not improve the cost value after some execution time limit, without increasing the coverage. Such a blocking situation often indicates a local optimum, and an exploration-driven solver, either the uniform or low-discrepancy sampling methods, is used to escape it. We monitor the coverage and robustness evolution, to detect if they do not increase and decrease respectively by some predefined amounts, for a predefined number of iterations. Due to the monotonicity of the coverage and robustness evolution with respect to the number of visited points, the detection can be done by comparing the coverage and the robustness values of the current iteration to those of the previous iteration.

Exploitation to Improve Best Candidates. An exploitation-driven solver with index s runs from a set P of initial points for T_s time (see Algorithm 2). The corresponding best cost value is stored in ρ^*. The reason we store the visited states is that they can reflect the relation between the cost function and the decision variables and can thus indicate promising regions, so as to derive good initializations for subsequent solvers.

Solver Initialization. We select initial points for a solver using the following heuristics:

- Select an initial point or a population of initial points from the best points obtained from previous iterations.
- Pick initial points according to a distribution that is dynamically updated based on the previous results, as inspired by the population based methods such as the CMAES. As described above, after each iteration we keep the points visited in the previous iterations. We select a set of best points, the robustness values of which are below some threshold, and use them to define the sampling distribution for new candidates. Let p be a parameter point and p_i denote its i^{th} coordinate. For any point p in G, let $[\underline{p}_i, \overline{p}_i]$ be the bounding interval such that each coordinate $p_i \in [\underline{p}_i, \overline{p}_i]$. In the k^{th} iteration,

the sampling distribution of p_i can be a normal distribution $\mathcal{N}(\overline{p}_i^k, \sigma_i^k)$, where the mean $\overline{p}_i^k$ is one of the most promising candidates from the previous iteration, selected based on the robustness value. The standard deviation σ_i^k in the k^{th} iteration can be determined by: $\sigma_i^k = (\overline{p}_i - \underline{p}_i)(\frac{1}{N^k})^{k/n}$ which decreases iteration after iteration. The number N^k of candidates can vary, being large at the beginning and decreasing gradually. In the first iteration where no information is available, we can sample candidate points according to the uniform distribution.

6 Experimentation

We use two case studies to evaluate our algorithms: a model of $\Delta\Sigma$ modulator and an automatic transmission control system, through which we show the efficiency of using the encoding constrained signal space, and evaluate the advantage of combining different metaheuristics. The combination algorithm is implemented in MATLAB® and uses 4 metaheuristics (integrated in Breach [19]): Simulated Annealing (SA) [31], CMAES [28], a globalized version of the Nelder Mead algorithm proposed by Luersen and Le Richec [2] abbreaviated by LRNM, and another globalized version of the Nelder Mead algorithm combining the classical Nelder Mead algorithm [36] with some corner searches, abbreaviated by GNM. The tool Breach [19] also provides robustness evaluation and signal construction from timed words. The generation of timed words from points in the unit box is done by the tool WordGen. Our experiments were performed on a computer with a 1.4GHz processor with 4GB RAM, running MATLAB® R2015a 64-bit version.

$\Delta\Sigma$ Modulator. We illustrate the application of our method of encoding constrained signal space with a $\Delta\Sigma$ modulator which is an important component of analog-to-digital converters. Practical quantizers have a limited input and output ranges which may lead them to saturation, and we want to check whether the output ever saturates. We use a behavioral model of a second-order modulator specified using Simulink®, which takes into account most non-idealities [12], including sampling jitter, integrator noise, and op-amp parameters (finite gain, finite bandwidth, slew-rate and saturation voltages). There exist simplified discrete-time $\Delta\Sigma$ modulator models without non-idealities, for which it is possible to derive its dynamics equations and thus can be solved using optimization [14] and statistical model-checking [13]. However, this Simulink model is heterogeneous, including embedded MATLAB code and mixing discrete-time and continuous-time components; it is too complex for existing formal verification tools. We consider the falsification of the absence of saturation of some quantizer signal *Out* under a certain class of nearly oscillatory inputs *In*. Formally *In* and *Out* must satisfy for some $\mathrm{t_s} \geq 0$ and $\forall t \geq 0$,

$$|\mathit{Out}(t)| < 2 \tag{3}$$

$$\exists T \in [8\mathrm{t_s}, 12\mathrm{t_s}] \text{ such that } \mathit{In}(t+T) = \mathit{In}(T) \tag{4}$$

Encoding (3) as an STL formula is trivial: $\varphi_{\neg\text{sat}} = \Box|\mathit{Out}| < p_{sat}$. However, enforcing that *In* satisfies (4) is not so simple. For instance, unbounded periodic properties are known to be beyond STL expressivity [35], and this is before considering that periods may be uncertain. We consider two approaches: one based on the above-described TA framework and another using only STL formulas. In both approaches, we use a signal

generator interpolating the signal values between points of a periodic discrete sequence of the form: $u_0\ \tau_0\ u_1\ \tau_1\ u_2\ \tau_2\ u_3\ \tau_3\ u_0\ \tau_4\ u_1\ \tau_5\ u_2\ \tau_6\ u_3\ \tau_7\ u_0\ \tau_N\ u_{\hat{N}} \dots$ The value $\mathit{In}(t)$ is obtained by finding k such that $\sum_0^k \tau_i \leq t < \sum_0^{k+1} \tau_i$ and interpolating between $u_{\overline{k}}$ and $u_{\overline{k}+1}$ where $\overline{k}$ is the remainder of $k/4$. Since the discrete sequence u_i is periodic, the resulting signal satisfies (4) iff $\forall i,\ 8\mathrm{t}_s \leq \tau_i + \tau_{i+1} + \tau_{i+2} + \tau_{i+3} \leq 12\mathrm{t}_s$. Note that this constraint is satisfied by the delays of the timed words of our TA of Fig. 1. Hence by using WordGen to generate timed words and mapping labels *a, b, c, d* to values u_0, u_1, u_2, u_3 we obtain the desired signals. To cross-validate this approach, we used

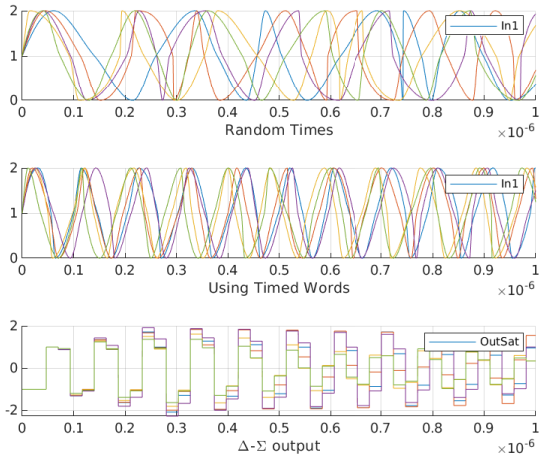

Fig. 2. Example traces for the $\Delta\Sigma$ modulator output (bottom) using inputs signals with random timings (top) and timings based on timed words from the TA of Fig. 1 (middle)

a simple formula: $\varphi_{per} = \Diamond_{[0,t_{end}]}(up \rightarrow upnext) \ \wedge\ \Diamond_{[0,t_{end}]}(down \rightarrow downnext)$, where $up = In1[t] > 1.9$ and $upnext = \Box_{[7.5*ts,12.5*ts]}(up)$, and $down$ and $downnext$ are defined similarly. We then defined the falsification problem as

$$\min_v \rho(\varphi_{\neg \text{sat}}, \mathit{Out}(v)) \tag{5}$$

$$\text{s.t. } \mathit{In}(v) \models \varphi_{\text{per}}. \tag{6}$$

where v is a parameter vector. In the TA based approach, $v \in \mathcal{P}_v$ as described in Section 4 whereas in the TA-free approach, v encodes directly delays between values, i.e., $v \in \{(\tau_0, \dots, \tau_N) \mid \tau_i \in [0, 4\mathrm{t}_s]\}$, and in this case the solver is responsible for the satisfaction of constraint (6). Breach implements a simple "optimized rejection" strategy where the constrained optimization problem (5-6) is basically replaced by a unconstrained $\min_v(J(v))$ where $J(v) = \rho(\varphi_{\neg \text{sat}}, \mathit{Out}(v))$ if φ_{per} is satisfied and $J(v) = -\rho(\varphi_{\text{per}}, \mathit{In}(v))$ otherwise. In other words, when in an infeasible region, Breach actively tries to satisfy φ_{per} with the current optimization strategy. This is a *rejection* strategy in the sense that when φ_{per} is not satisfied, v is not used, i.e., $\mathit{Out}(v)$ is not computed to avoid useless simulations. With these settings, we could confirm that the TA-based approach indeed generated only inputs satisfying φ_{per}, for arbitrarily long inputs. In addition, the

optimized rejection approach only works for short horizons. For instance, we considered simulations of duration $1e{-}6$ seconds with $\mathrm{t}_s = 1e{-}8$ seconds. To be able to satisfy φ_{per} we had to set the horizon t_{end} to $3e{-}7$ seconds, *i.e.* considering only about 3 periods. Longer horizons would result in the solver rejecting most of considered inputs, which can be explained by a small ratio of the volume of the language of valid inputs w.r.t. that of the language of all inputs.

For the saturation threshold $p_{sat} = 2$ used in the model [12], the property $\varphi_{\neg\text{sat}}$ was easily falsified in our optimization setting. In addition, we could compare the performance of different metaheuristics by continuing the optimization after falsification. Using our previous algorithm [6] based purely on a set of 10000 uniformly generated signals, the highest absolute output value is 2.32032. However, using the combined metaheuristics after exploring only 826 signals, a higher value, 2.322586, is found. More concretely, we fixed the saturation threshold p_{sat} to be 2.325 in $\varphi_{\neg\text{sat}}$ and ran the metaheuristics with the option of stopping at the first falsifying trace that is found. With some fixed seed[9] (100 in this case), all the stand-alone metaheuristics could not falsify the property, but the combined metaheuristics could (see Table 1). The combined metaheuristics first used Simulated Annealing and then LR Nelder Mead, which got stuck in a blocking situation where the robustness is not improved and the coverage does not increase significantly. It then switched to the CMAES metaheuristics but used the points explored by the previous metaheuristics to estimate a good initial distribution for this CMAES solver which could then falsify the property. The CMAES method seemed to have the best performance for this example, among the stand-alone metaheuristics, for this examples, we thus compared it with the combined metaheuristics using different seeds. The comparison results are summarized in Table 2, which indicates that the combined metaheuristics algorithm outperformed the stand-alone CMAES for seeds 1000 and 10000, and was less performant for seed 5000. This shows how initializations can affect the performance of the metaheuristics, and the combination guided by coverage and robustness can be thought of as a heuristic (on top of the metaheuristics) that tries to use the information gained through the search to lead it towards promising initializations.

Table 1. Using the different methods on the $\Delta\Sigma$ model with seed 100

Search method	Min robustness; Max ($\lvert Out \rvert$)	Nb fct eval	Comp time (s)
CMAES	0.003746; 2.321254	10000	6103.282974
SA	0.027244; 2.297756	10000	8036.702422
GNM	0.031889; 2.293111	10000	6763.065164
Uniform Rand	0.00338031; 2.32161969	10000	4539.560286
LRNM	0.07562901; 2.24937099	10000	4854.569456
Combined Meta.	-0.002414; 2.327414	826	431.434701

Automatic Transmission Control. This model [30] has been used as a benchmark for evaluating hybrid systems validation techniques[10]. Here we extend it to capture constraints on the input signals that reflect usual driving patterns, based on the data from

[9] The seed here refers to the index for a sequence of random numbers in MATLAB.

[10] See `http://cps-vo.org/node/12116`

Table 2. Comparing the combined metaheuristics and the CMAES with different seeds

Search method	Seed	Min robustness; Max ($\|Out\|$)	Nb fct eval	Comp time (s)
CMAES	1000	0.002282; 2.322718	10000	6430.215422
Combined Meta.	1000	-0.00323532; 2.32823532	936	489.150343
CMAES	5000	-0.00164623; 2.32664623	1081	463.796410
Combined Meta.	5000	-0.00201822; 2.32701822	1100	536.904802
CMAES	10000	0.00226337; 2.32273663	10000	7747.896409
Combined Meta.	10000	-0.000305395; 2.324694605	766	310.282428

the study in [40]. The system has two inputs: throttle α and brake β, and two outputs: the engine speed w (RPM) and the vehicle speed v (mph). We consider the input signals that satisfy the constraints of the timed automaton with two clocks x and y in Fig. 3. 'Coasting' means that both the brake and acceleration pedals are not pressed, that is the

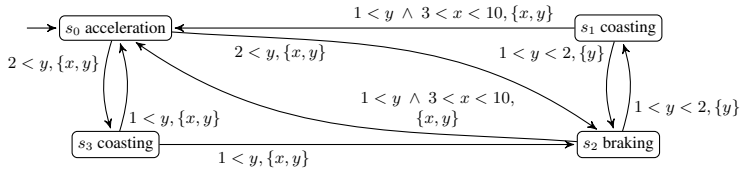

Fig. 3. A timed automaton describing the driving patterns of interest. A global invariant $y < 15$ (meaning that the location changes within at most 15 seconds) is not depicted.

two inputs are 0. The loop consisting of locations s_0, s_3 describes accelerating behaviors with coasting. At the location 'acceleration', braking can happen after accelerating for at least 2 and not more than 19 seconds, indicated by the transition from 'acceleration' to 'braking'. The loop between 'braking' and 'coasting' models the fact that the driver can push and release the brake pedal successively a number of times to adjust the vehicle speed. The clock x which is not reset in the transitions between 'braking' and 'coasting' allows to measure the time staying in this loop before returning to 'acceleration' by one of the two transitions both guarded by $x > 3$. In other words, the driver must stay in the braking-coasting (s_1-s_2) loop for at least 3 seconds. The transition labels are associated with the following range constraints on the input signal values: s_0 to s_3 (acceleration to coasting), $\alpha = 0$, $\beta = 0$; s_3 to s_2 (coasting to brake) $\alpha = 0$, $\beta = [100, 325]$; s_3 to s_0 (coasting to acceleration) $\alpha = (0, 500]$, $\beta = [100, 325]$; s_0 to s_2 (acceleration to braking) $\alpha = 0$, $\beta = 0$; s_1 to s_2 (coasting to brake) $\alpha = 0$, $\beta = [100, 325]$; s_2 to s_1 (brake to coasting) $\alpha = 0$, $\beta = 0$; s_2 to s_0 (brake to acceleration) $\alpha = (0, 500]$, $\beta = 0$; s_1 to s_0 (coasting to acceleration) $\alpha = (0, 500]$, $\beta = 0$. In terms of values, we use piece-wise constant signals satisfying the ranges associated to the transition labels. The property to check states that if the gear is 3 the vehicle speed should not be too slow, which is described by a STL formula: $\phi = \square_{[20,100]} \neg((gear = 3) \wedge (v < v_{min}))$. We seek a driving behavior (that is the input signals of throttle and brake) that leads to a violation of this property. For $v_{min} = 19.76$ (mph) the combined metaheuristics algorithm falsified it after 326 seconds, while GNM alone took 974 seconds and CMAES took 650 seconds to falsify. This experiment shows that these metaheuristics, when used alone, spent much time around local optima.

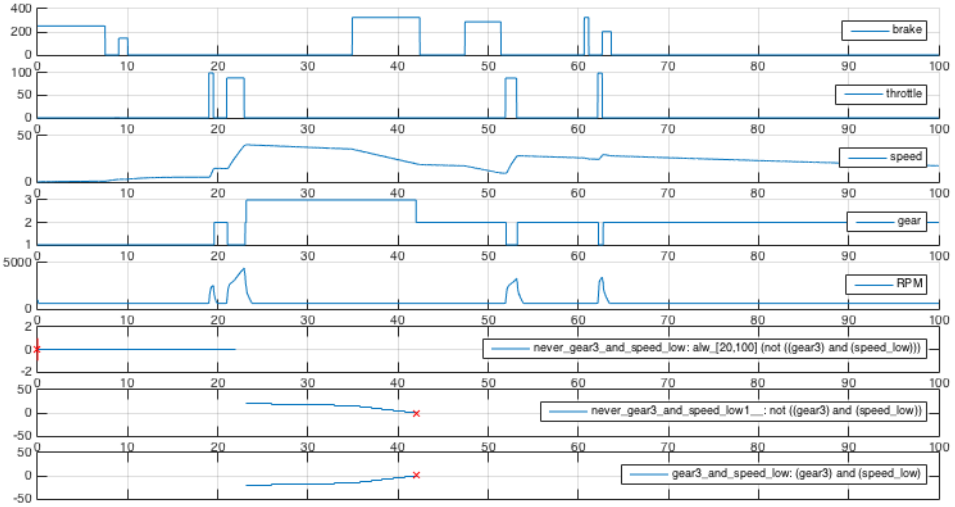

Fig. 4. A falsifying trace of the automatic transmission control system found by the combined metaheuristics algorithm. The (red) cross on the last plots indicates the instant of worst violation as computed by the diagnostics algorithm of [25] which allows ignoring quantitative information from the gear signal to focus on the speed signal only, which explains why robustness is not plotted in certain intervals.

7 Conclusion

We presented a new falsification algorithm based on a method for encoding input signals subject to timed automaton constraints. We defined a coverage measure for such constrained signal spaces. We also proposed a combination of different metaheuristics to exploit their complementary properties. Switching between the metaheuristics, based on the coverage information, allows escaping local optimum situations. We successfully demonstrated the efficacy and advantage of the new algorithms through two case studies. Ongoing work includes considering the usage of other coverage measures, such as combinatorial entropy. Furthermore, the metaheuristic switching currently depends on global coverage and robustness improvement thresholds determining blocking situations, and a biased switching can be defined using local coverage measures based on multi-resolution partitions. We also plan to use ideas from the racing algorithms [10] for identifying and dropping inferior candidates during the search.

Bibliography

[1] Arvind S. Adimoolam, Thao Dang, Alexandre Donzé, James Kapinski, and Xiaoqing Jin. Classification and coverage-based falsification for embedded control systems. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 483–503, 2017.

[2] Marco A.Luersen and Rodolphe Le Richec. Globalized nelder-mead method for engineering optimization. *Computers and Structures*, 82(23):2251 – 2260, 2004. Computational Structures Technology.

[3] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[4] Yashwanth Annapureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *TACAS*, pages 254–257, 2011.

[5] Benoît Barbot, Nicolas Basset, Marc Beunardeau, and Marta Kwiatkowska. Uniform sampling for timed automata with application to language inclusion measurement. In *Quantitative Evaluation of Systems - 13th International Conference, QEST 2016, Quebec City, QC, Canada, August 23-25, 2016*, pages 175–190, 2016.

[6] Benoît Barbot, Nicolas Basset, and Thao Dang. Generation of signals under temporal constraints for CPS testing. In *NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings*, pages 54–70, 2019.

[7] Benoît Barbot, Béatrice Bérard, Yann Duplouy, and Serge Haddad. Integrating Simulink models into the model checker Cosmos. In Victor Khomenko and Olivier H. Roux, editors, *Application and Theory of Petri Nets and Concurrency*, pages 363–373. Springer International Publishing, 2018.

[8] Benoît Barbot. WordGen. `https://git.lacl.fr/barbot/wordgen`, 2019.

[9] Ezio Bartocci, Jyotirmoy V. Deshmukh, Alexandre Donzé, Georgios E. Fainekos, Oded Maler, Dejan Nickovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, pages 135–175. Springer, 2018.

[10] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[11] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003.

[12] S. Brigati, F. Francesconi, P. Malcovati, D. Tonietto, A. Baschirotto, and F. Maloberti. Modeling Sigma-Delta modulator non-idealities in Simulink. In *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI*, volume 2, pages 384–387 vol.2, May 1999.

[13] Edmund M. Clarke, Alexandre Donzé, and Axel Legay. On simulation-based probabilistic model checking of mixed-analog circuits. *Formal Methods in System Design*, 36(2):97–113, 2010.

[14] Thao Dang, Alexandre Donzé, and Oded Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In *Formal Methods in Computer-Aided Design, 5th International Conference, FMCAD 2004, Austin, Texas, USA, November 15-17, 2004, Proceedings*, pages 21–36, 2004.

[15] Thao Dang and Tarik Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[16] Jyotirmoy V. Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic local search for falsification of hybrid systems. In *ATVA*, volume 9364 of *Lecture Notes in Computer Science*, pages 500–517. Springer, 2015.

[17] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.

[18] Alexandre Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010.*, pages 167–170, 2010.

[19] Alexandre Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, pages 167–170, 2010.

[20] 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.

[21] Johann Dreo, Patrick Siarry, Alain Petrowski, and Eric Taillard. *Metaheuristics for hard optimization : methods and case studies*. Springer-Verlag, 2006.

[22] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Patrick 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.

[23] Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Adaptive rrts for validating hybrid robotic control systems. In *WAFR*, 2004.

[24] G.E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications. In *Proceedings of FATES/RV*, volume 4262 of *LNCS*, pages 178–192. Springer, 2006.

[25] Thomas Ferrère, Dejan Nickovic, Alexandre Donzé, Hisahiro Ito, and James Kapinski. Interface-aware signal temporal logic. In *HSCC*, pages 57–66. ACM, 2019.

[26] Christodoulos A. Floudas and Panos M. Pardalos, editors. *Encyclopedia of Optimization, Second Edition*. Springer, 2009.

[27] D.M. Gabbay, P. Thagard, J. Woods, J. Butterfield, and J. Earman. *Philosophy of Physics: Handbook of the Philosophy of Science*. Elsevier Science, 2006.

[28] N. Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[29] Stefan Heinrich. Some open problems concerning the star-discrepancy. *Journal of Complexity*, 19(3):416 – 419, 2003. Oberwolfach Special Issue.

[30] 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.

[31] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.

[32] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, October 1990.

[33] Jan Kuřátko and Stefan Ratschan. Combined global and local search for the falsification of hybrid systems. In Axel Legay and Marius Bozga, editors, *Formal Modeling and Analysis of Timed Systems*, pages 146–160, Cham, 2014. Springer International Publishing.

[34] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. CAV'11*, 2011.

[35] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.

[36] John A. Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[37] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivanciec, Aarti Gupta, and George J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *HSCC'10 - Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 211–220, 2010.

[38] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, Jul 2013.

[39] Simone Silvetti, Alberto Policriti, and Luca Bortolussi. An active learning approach to the falsification of black box cyber-physical systems. In Nadia Polikarpova and Steve Schneider, editors, *Integrated Formal Methods*, pages 3–17, Cham, 2017. Springer International Publishing.

[40] Gyubin Sim, Seongju Ahn, Inseok Park, Jeamyoung Youn, Seungjae Yoo, and Kyunghan Min. Automatic longitudinal regenerative control of evs based on a driver characteristics-oriented deceleration model. *World Electric Vehicle Journal*, 10:58, 09 2019.

[41] P. Skruch. A coverage metric to evaluate tests for continuous-time dynamic systems. *Central European Journal of Engineering*, 1(2):174–180, 2011.

[42] W. A. Stein et al. *Sage Mathematics Software (Version 6.9)*. The Sage Development Team, 2015. `http://www.sagemath.org`.