Generation of Signals under Temporal Constraints for CPS Testing

Benoît Barbot[1], Nicolas Basset[2], and Thao Dang[2]

[1] LACL, Université Paris-Est Créteil, France
[2] VERIMAG/CNRS, Université Grenoble Alpes, France

Abstract. This work is concerned with validation of cyber-physical systems (CPS) via sampling of input signal spaces. Such a space is infinite and in general too difficult to treat symbolically, meaning that the only reasonable option is to sample a finite number of input signals and simulate the corresponding system behaviours. It is important to choose a sample so that it best "covers" the whole input signal space. We use timed automata to model temporal constraints, in order to avoid spurious bugs coming from unrealistic inputs and this can also reduce the input space to explore. We propose a method for low-discrepancy generation of signals under temporal constraints recognised by timed automata. The discrepancy notion reflects how uniform the input signal space is sampled and additionally allows deriving validation and performance guarantees. To evaluate testing quality, we also show a measure of uniformity of an arbitrary set of input signals. We describe a prototype tool chain and demonstrate the proposed methods on a Kinetic Battery Model (KiBaM) and a $\Sigma\Delta$ modulator.

1 Introduction

Cyber-physical systems (CPS) are integrations of computation with physical processes, and have become a predominant component of modern engineering systems. A major challenge in proving correct operations of industrial CPS is the absence of rigorous mathematical models, and even when such models are available they are often intractable by exhaustive formal verification techniques, due to computational complexity. Falsification methods, based on black-box testing, are often used for industrial-size complex systems. These methods rely on a tester that can execute/simulate the system under some input stimuli and observe the corresponding outputs; their goal is to search for the worst case behaviours by minimising robustness of satisfaction of some temporal logic formula (see for example [18,15,2,29]). The most popular tools are S-Taliro [3] and Breach [16]. Generally, a challenge in this approach is the limitation of global optimisation solvers which may converge to local optima. Also, most optimisers do not take into account input constraints and may lead to trivial solutions that do not correspond to realistic scenarios. Statistical model checking based on Monte Carlo methods has also been applied to CPS [13,1,7]. The reader is referred to a survey on CPS validation approaches [8]. We defer a discussion on related work to Section 6 after our approach is described in detail.

It is clear that the efficiency of such a validation process depends on the class of input signals under consideration. On one hand, this class should be sufficiently expressive to capture all feasible configurations of the environment. On the other hand, such permissible classes can be very large; therefore, it is desirable to consider only the input stimuli which are realistic or relevant w.r.t. the operation context of the system. In this work, we are interested in classes of signals which satisfy temporal constraints modelled as timed automata (TA). Indeed, in a CPS, computation processes interact with physical processes via devices (such as sampling, measurement, actuation) the timing imprecision of which can be appropriately modelled using TA. We now formulate the problems we want to solve in order to address the above issues.

1. Generate a set of input signals satisfying some temporal constraints. Using these signals to simulate or execute the system, one expects to find a behaviour that falsifies the property. When no such behaviour is found, it is important to provide guarantees, such as the portion of correct behaviours, or the average robustness of property satisfaction.
2. Given an arbitrary set of input signals, determine its testing quality in terms of property checking or performance evaluation.

To address the first problem, we extend the method for uniform random generation of runs of timed automata [6] to propose a new method based on low discrepancy. The generated timed words are then mapped to input signals. To address the second problem, we employ the well-known Kolmogorov-Smirnov statistic [26] to measure the goodness of fit of a sample w.r.t. a given distribution. Interestingly, this statistic can be interpreted in terms of the star discrepancy [23] largely used in quasi-Monte Carlo methods.

The paper is organised as follows. Section 2 recalls important concepts, namely the star discrepancy, timed automata and timed polytopes, and quantitative guarantees. The next three sections assume that a timed automaton describing a class of input signals of interest is given, and focus on the problem of generation of timed words. Section 3 presents a transformation from the unit box to a timed polytope (corresponding to the constraints on time delays along a path). In the forward direction, by sampling over the unit box and then applying the transformation we obtain a sampling over the timed polytope. In Section 4, we describe a new method for low-discrepancy sampling which yields a quasi-Monte Carlo method. Section 5 presents a measure of uniformity degree of an arbitrary sample and discusses how this measure can be estimated, by using again the above-described transformation but in the backward direction (which is known as the Rosenblatt's transformation [26]). Finally, based on these results we propose in Section 6 a framework for testing CPS with guarantees. We include here a comparison with related work to highlight the novelty of our approach. We describe a tool chain which integrates an implementation of these methods, and demonstrate the proposed methods on a Kinetic Battery Model and a $\Sigma\Delta$ modulator.

2 Preliminaries

Star Discrepancy. By the (n-dimensional) unit box we mean the set $[0,1]^n$. Given a point $\boldsymbol{b} = (b_1, \ldots, b_n)$ inside the unit box, we define the box $[\mathbf{0}, \boldsymbol{b}] = [0, b_1] \times \cdots \times [0, b_n]$. The star discrepancy of a finite set S of points in the unit box is defined as:

$$D_\star(S) = \sup_{\boldsymbol{b} \in [0,1]^n} \left| \mathtt{Vol}([\mathbf{0}, \boldsymbol{b}]) - \frac{|S \cap [\mathbf{0}, \boldsymbol{b}]|}{|S|} \right|.$$

Intuitively, the star discrepancy is a measure of how equi-distributed a set of points is over the unit box, or how different its distribution is compared to the uniform distribution. This notion is used in number theoretic and quasi-Monte Carlo methods. The lower the discrepancy is, the better the space is "filled" with points. Asymptotically a sequence of uniform random points will homogeneously fill the space $\mathcal{P}$ that is sampled such that for every subset of $\mathcal{P}$, the density of points in this subset will be proportional to its volume. In this work we use two well-known low-discrepancy sequences: Halton [19] and Kronecker [25]. The star discrepancy is a way of quantifying how homogeneously a sample covers the sampling space for finite sequences. Its link with the n-dimensional Kolmogorov-Smirnov statistic is provided later (see Section 5.2).

Timed Automata and Timed Polytopes. Let X be a finite set of non-negative real-valued variables called *clocks*, which are assumed bounded by a constant $M \in \mathbb{N}$. A *clock constraint* has the form $x \sim c$ or $x - y \sim c$ where $\sim \in \{\leq, <, =, >, \geq\}$ with $x, y \in X,\ c \in \mathbb{N}$. A *guard* is a finite conjunction of clock constraints. For a clock vector $\boldsymbol{x} \in [0, M]^X$ and a non-negative real t, we denote by $\boldsymbol{x} + t$ the vector $\boldsymbol{x} + (t, \ldots, t)$. A *timed automaton* (TA) $\mathcal{A}$ is a tuple $(\Sigma, X, Q, i_0, \mathcal{F}, \Delta)$ where Σ is a finite set of events; X is a finite set of clocks; Q is a finite set of *locations*; i_0 is the initial location; $\mathcal{F} \subseteq Q$ is a set of final locations; and Δ is a finite set of *transitions.* A transition $\delta \in \Delta$ has an *origin* $\delta^- \in Q$, a *destination* $\delta^+ \in Q$, a label $a_\delta \in \Sigma$, a *guard* $\mathfrak{g}_\delta$ and a *reset* function $\mathfrak{r}_\delta$ determined by a subset of clocks $B \subseteq X$; this transition resets to 0 all the clocks in B and does not modify the other clocks. A *state* $s = (q, \boldsymbol{x}) \in Q \times [0, M]^X$ is a pair of a location and a clock vector. The initial state of $\mathcal{A}$ is $(i_0, \mathbf{0})$. A *timed transition* is a pair (t, δ) of a time *delay* $t \in [0, M]$ followed by a discrete transition $\delta \in \Delta$. The delay t represents the time before firing the transition δ. A run is an alternating sequence $(q_0, \boldsymbol{x}_0) \xrightarrow{t_1, \delta_1} (q_1, \boldsymbol{x}_1) \ldots \xrightarrow{t_n, \delta_n} (q_n, \boldsymbol{x}_n)$ of states and timed transitions with the following updating rules: q_i is the successors of q_{i-1} by δ_i, the vector $\boldsymbol{x}_{i-1} + t$ must satisfy the guard $\mathfrak{g}_\delta$ and $\boldsymbol{x}_i = \mathfrak{r}_\delta(\boldsymbol{x}_{i-1} + t)$. This run is *labelled* by the *timed word* $(t_1, a_1) \cdots (t_n, a_n)$ where for every $i \leq n$, a_i is the label of δ_i. The set of timed words that label all the runs leading from the initial state $(i_0, \mathbf{0})$ to a final state $(q_n \in \mathcal{F})$ is called the *timed language* of $\mathcal{A}$. Given a discrete path $\alpha = \delta_1 \cdots \delta_n$ of $\mathcal{A}$ the set of timed vectors $\boldsymbol{t} \in [0, M]^n$ such that $(i_0, \mathbf{0}) \xrightarrow{t_1, \delta_1} (q_1, t_1) \ldots \xrightarrow{t_n, \delta_n} (q_n, t_n)$ is called the *timed polytope* associated to the path α.

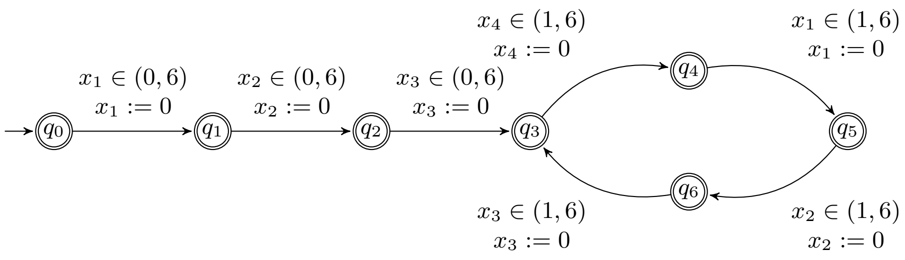


Fig. 1. A timed automaton for the running example.

Example 1 (Running example). We consider the TA in Fig. 1. This automaton has the property that after entering the cycle the time between 4 consecutive events is between 1 and 6. Intuitively, they are loosely periodic as transitions cannot be taken too early or too late. This automaton is used to model a quasi-periodic pattern of signals with uncertain period ranging between 1 and 6. Its first three locations before the cycle model the uncertain phase of the signals. To illustrate the timed polytope notion, we consider the path of length 2 starting at location q_0 and ending at q_2. The timed polytope corresponding to this path is the triangle $\{(t_1, t_2) \mid t_1 + t_2 < 6, t_1 > 0, t_2 > 0\}$. Its 2-dimensional volume is $6^2/2 = 18$. Uniform sampling in this polytope is depicted in Fig. 2 (b_1). More generally, the timed polytope associated to the (unique) path of length n is defined by $0 < t_{k-3} + t_{k-2} + t_{k-1} + t_k < 6$, for $k = 1, \ldots, 3$ with the convention that $t_j = 0$ for $j < 1$, and $1 < t_{k-3} + t_{k-2} + t_{k-1} + t_k < 6$, for $k = 4, \ldots, n-3$. The k^{th} constraint is due to the guard $x_i \in (1, 6)$ (to be precise, with $i = (k \bmod 4)+1$) because the clock $\boldsymbol{x}_i$ contains the sum of the 4 last delays before taking the k^{th} transition. Computing the volume of such a timed polytope requires dynamic programming algorithms (involving an integral operator per transition) which can be found in [4,6].

Quantitative Guarantees and Sampling-Based Estimation. Quantitative properties of CPS can be expressed by averaging some function f defined on the set of input signals. For instance, such a function can be the indicator function of the set of input stimuli that lead to incorrect behaviours, and the average gives the probability that an input signal falls in this set. This is more generally the problem of estimating a sub-language volume. In addition, given a property expressed using temporal logics, f can be the (satisfaction) robustness which is a function of the input. Such properties can thus be evaluated by sampling in the input signal space, as in Monte Carlo and quasi-Monte Carlo methods. To obtain accurate results, the sampling process should generate the signals as uniformly as possible to cover well the input signal space with high probability. In other words, the probability that a generated signal falls in a set should be proportional to the volume of this set. To solve this problem for timed automata we need a transformation from the unit box to a timed polytope, which is explained in the next section. We also use this transformation for

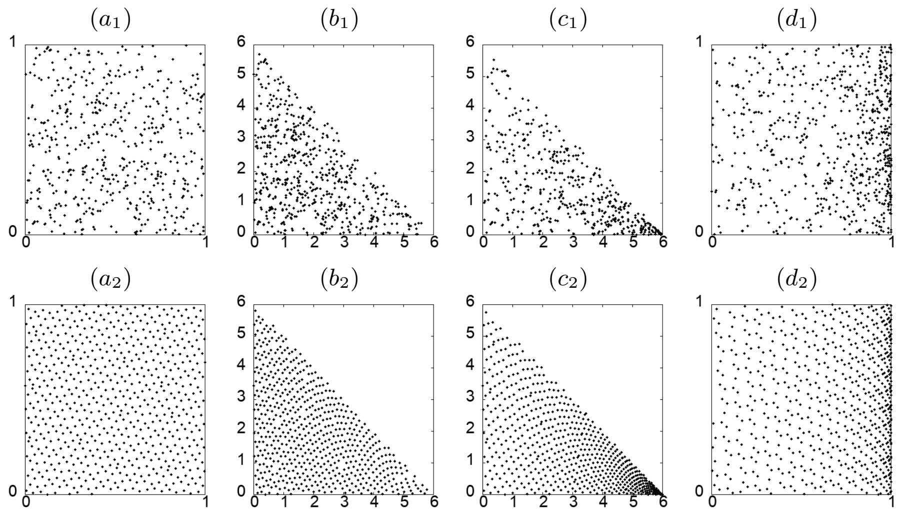


Fig. 2. First line, left to right. (a_1): we first draw 450 points uniformly at random in the unit box; (b_1): we then apply the inverse sampling method with the CDF of the uniform sampling (see Theorem 1) to get 450 timed vectors in the timed polytope; (c_1): we do the same as for (b_1) but using the CDF of the isotropic sampling; (d_1): the sample in (c_1) is mapped back to the unit box with the CDF of the uniform sampling to check its star discrepancy. Second line, we do the same as for the first line, but starting with a low-discrepancy sequence of 450 points in the unit box (a_2).

a low-discrepancy generation method presented in Section 4. We further exploit this transformation in Section 5 to estimate a measure of uniformity degree for an arbitrary sample of timed words.

3 Tranformation from the Unit Box to a Timed Polytope

Defining Probability Distributions to Sample Timed Languages. We first emphasise that we want to sample uniformly timed words of a given length from the timed language of a given TA. This *uniform sampling* is such that every timed word of the language has the "same chance" of being sampled. Note that the sampled space is uncountably infinite since the delays are real numbers. This uniform sampling should not be confused with an intuitive sampling method, called *isotropic*, which at each discrete step makes a uniform choice among the possible delays. This isotropic sampling method is used as a "default" sampling in several work (see [10] and references therein). The difference between the two sampling methods will be illustrated in Ex. 2.

For a given TA, our previous work [6] proposes a method for generating uniformly timed words of a given length. Such a uniform sampling is done by adding probability distributions on time delays along a run of the automaton. In [9] we treat the case of generating infinite timed words, based on the notion of maximal entropy. Let us now explain the essence of these ideas. The constraints on the delays along a discrete path define a timed polytope; sampling runs along a discrete path thus reduces to sampling over a timed polytope. For simplicity of presentation, we will explain only the generation of timed vectors in a timed polytope, or in a sequence of timed polytopes along a single path. Discrete branching can be handled similarly as in [6].

Given a discrete path $\alpha = \delta_1 \cdots \delta_n$ of the TA $\mathcal{A}$, let $\mathcal{P}$ be the timed polytope associated to this path. We want to evaluate $\int_{\mathcal{P}} f(\boldsymbol{t}) d\boldsymbol{t} / \texttt{Vol}(\mathcal{P})$, the average over $\mathcal{P}$ of a function $f : \mathcal{P} \to \mathbb{R}$ (used to express a quantitative property). Here the normalising constant $\texttt{Vol}(\mathcal{P}) = \int_{\mathcal{P}} 1 d\boldsymbol{t}$ is the n-dimensional volume of $\mathcal{P}$. More generally, we can give different weights to different timed vectors of $\mathcal{P}$ by using a *probability density function* (PDF), namely a function $\omega : \mathcal{P} \to \mathbb{R}^+$ such that $\int_{\mathcal{P}} \omega(\boldsymbol{t}) d\boldsymbol{t} = 1$. Then the integral we want to evaluate becomes $\int_{\mathcal{P}} f(\boldsymbol{t}) \omega(\boldsymbol{t}) d\boldsymbol{t}$, which is also called the expectation $E(f(\boldsymbol{T}))$ of the random variable $\boldsymbol{T} = (T_1, \ldots, T_n)$ distributed according to the PDF ω (and that takes values as timed vectors in $\mathcal{P}$).

The uniform distribution assigns the density of probability $\omega(\boldsymbol{t}) = 1/\texttt{Vol}(\mathcal{P})$ to every timed vector $\boldsymbol{t} \in \mathcal{P}$. A sampled timed vector $\boldsymbol{t}$ thus falls in a given subset A of the timed polytope $\mathcal{P}$ with probability $\texttt{Vol}(A)/\texttt{Vol}(\mathcal{P})$. If we define f as the indicator function 1_B of a set B of "bad" behaviours, and if $\boldsymbol{T}$ is distributed according to the uniform distribution, then the expectation $E(1_B(\boldsymbol{T}))$ measures the portion of bad behaviours in $\mathcal{P}$, formally $E(1_B(\boldsymbol{T})) = \texttt{Vol}(B)/\texttt{Vol}(\mathcal{P})$.

To define the random variables $\boldsymbol{T}$, it suffices to give its n-dimensional cumulative distribution function (CDF) $F(\boldsymbol{t}) = \text{Prob}(\boldsymbol{T} \leq \boldsymbol{t})$ where the partial order $\leq$ is defined by $(T_1, \ldots, T_n) \leq (t_1, \ldots, t_n)$ iff $T_i \leq t_i$ for every $i = 1 \ldots n$. This CDF is usually given by the following sequence of conditional CDF: $F_i(t_i \mid t_1, \ldots, t_{i-1}) = \text{Prob}(T_i \geq t_i \mid T_1 = t_1, \ldots, T_{i-1} = t_{i-1})$. The following chain rule gives the relation between the conditional CDF and the CDF of $\boldsymbol{T}$:

$$F(t_1, \ldots, t_n) = F_1(t_1) F_2(t_2 \mid t_1) \ldots F_n(t_n \mid t_1, \ldots, t_{n-1}).$$

In [6] and in some other work, the conditional CDF $F_i(t_i \mid \boldsymbol{t})$, used to sample t_i, depends only on the current state $(q_{i-1}, \boldsymbol{x}_{i-1})$, that is $F_i(t_i \mid \boldsymbol{t}) = G_i(t_i \mid (q_{i-1}, \boldsymbol{x}_{i-1}))$ for some conditional CDF G_i. For the uniform distribution on a timed polytope, the conditional CDF are characterised in [6], via the definition of the conditional PDF (which are the derivatives of the CDF). These conditional CDF for uniform sampling play a particular role in our subsequent development, and is denoted specifically by $\mathfrak{F} = (\mathfrak{F}_1, \ldots, \mathfrak{F}_n)$. Theorem 1 summarises the characterisation of the CDF for the uniform sampling of timed words. These CDF can be effectively computed and their computation was implemented in the tool chain of [6].

Theorem 1 ([6]). *Given a path in a TA one can compute the CDF $\mathfrak{F}_i$ in polynomial time w.r.t. the length of the path. These CDF can be written in the following form $\mathfrak{F}_i(t_i \mid t_1, \ldots, t_{i-1}) = \pi_i(t_1, \ldots, t_{i-1}) / \gamma_i(t_1, \ldots, t_i)$with π_i and γ_i polynomials of degree at most i.*

Example 2 (Ex. 1 continued). To show the difference between the isotropic and uniform methods, we consider again the path $q_0\, q_1\, q_2$ of the automaton of Ex. 1.

We sample timed vectors (t_1, t_2) in the 2-dimensional timed polytope associated to this path (shown in the first line of Fig. 2). Using the isotropic sampling, t_1 is chosen uniformly in $(0, 6)$ and then t_2 is chosen uniformly in $(0, 6 - t_1)$. This is why in Fig. 2-(c_1) the set of points generated by the isotropic sampling gets more and more dense along the t_1-axis. In particular, with the isotropic sampling the set $\{(t_1, t_2) \mid t_1 \in (0, 1), t_1 + t_2 < 6\}$ has the same probability as the small triangle $\{(t_1, t_2) \mid t_1 \in (5, 6), t_1 + t_2 < 6\}$, while the former is 11 times bigger than the latter. This is in contrast with the uniform sampling where the chance of falling in a set is proportional to its area. With the uniform sampling (see Fig. 2-(b_1)), t_1 is chosen according to the probability density function (PDF) $t_1 \mapsto (1 - t_1)/18$, and t_2 according to $t_2 \mapsto 1/(1 - t_1)$. The PDF of a timed vector (t_1, t_2) is hence $((1 - t_1)/18)1/(1 - t_1) = 1/18 = 1/\texttt{Vol}(\mathcal{P})$, as expected.

Transformation from the Unit Box to a Timed Polytope. We observe further that if we use the conditional CDF $F = (F_i)_{i=1..n}$ to transform a timed vector $\boldsymbol{t}$ to a vector $\boldsymbol{u}$ as follows: $u_1 = F_1(u_1)$, and for $i = 2, \dots, n$ $u_i = F_i(t_i | t_1, \dots, t_{i-1})$, then $\boldsymbol{u} = (u_1, \dots, u_n)$ is in $[0, 1]^n$. The following theorem allows going back and forth between a timed polytope and the unit box.

Theorem 2 (Rosenblatt's tranformation [26]). *Let $F = (F_i)_{i=1..n}$ be a sequence of conditional CDF. Define the transformation $\boldsymbol{U} = F(\boldsymbol{T})$ between the random vectors $\boldsymbol{U}$ and $\boldsymbol{T}$ by $U_1 = F_1(T_1)$, $U_i = F_i(T_i \mid T_1, \dots, T_{i-1})$ for every $i = 2 \dots n$. Then $\boldsymbol{T}$ is distributed according to the CDF F iff $U_1, \dots, U_n$ are i.i.d uniformly distributed random variables on* $[0, 1]$.

This theorem allows us to make use of the transformation $\mathfrak{F}^{-1}$ for generating timed words, similarly to random sampling according to CDF as in *inverse transform sampling*. Once $t_1, \dots, t_{i-1}$ are sampled, the next delay t_i is randomly sampled as follows. A real number u_i is drawn uniformly in $[0, 1]$, and then one finds the unique t_i such that $\mathfrak{F}_i(t_i \mid t_1, \dots, t_{i-1}) = u_i$ using for instance the Newton's method. Ultimately, from n i.i.d uniformly distributed random numbers $u_1, \dots, u_n$ in the unit interval, we get a timed vector $(t_1, \dots, t_n) = \mathfrak{F}^{-1}(u_1, \dots, u_n)$. This transformation $\mathfrak{F}^{-1}$ implicitly underlies the uniform sampling method presented in [6]. We can now use it for the two problems stated in the introduction:

- *In a forward manner for low-discrepancy generation (see Section 4).*
- *In a backward manner to evaluate the generation quality (see Section 5.2).*

4 Low-Discrepancy Generation and Quasi-Monte Carlo Methods for Timed Polytopes

We exploit the forward use of $\mathfrak{F}^{-1}$ (from the unit box to a timed polytope), to generate points in a timed polytope with low discrepancy. To this end, it suffices to start with a low-discrepancy set of points in the unit box and then apply to it the transformation $\mathfrak{F}^{-1}$. To obtain a low discrepancy point set in the unit box, in this work we use, as mentioned earlier, the well-known low-discrepancy sequences Halton [19] and Kronecker [25].

The use of low-discrepancy generation is motivated by the fact that when considering finite sequences, some (deterministic) low-discrepancy sequences behave better than uniform sequences in terms of homogeneous space-filling. Our generation procedure indeed yields a quasi-Monte Carlo method [24] for estimating or averaging integral functions which express quantitative properties of interest. Note that using the uniform random generation, one can only provide statistical guarantees, as Monte Carlo methods. This new generation method, in contrast, allows characterising *deterministic* error bounds (that is, without probabilistic uncertainty) in approximating the multi-dimensional integral of a function by the average of the function values on a sample of points. A popular characterisation is the Koksma-Hlawka (KH) inequality [20]. Formally, given a function $g : [0,1]^n \to \mathbb{R}$ and a sample $S = (\boldsymbol{p}^{(k)})_{k=1..N}$, the Koksma-Hlawka inequality is

$$\left| \frac{1}{N} \sum_{n=1}^{N} g(\boldsymbol{p}^{(n)}) - \int_{[0,1]^n} g(\boldsymbol{r}) d\boldsymbol{r} \right| \leq V^*(g)(D_\star(S))^{1/n} \tag{1}$$

where $D_\star(S)$ is the star discrepancy of the set S, $V^*(g)$ is the variation in the sense of Hardy and Krause, which does not depend on S, so it is constant when we fix g. Using low-discrepancy sequences yields an upper-bound $D_\star(S) \leq C_n \log(N)^n/N$ where the constant C_n depends on the point dimension n and on the type of the sequence but not on the number N of sampled points.

We now show how the above result can be applied to our testing context where f is the function expressing the guarantee. Each timed word corresponds to an input signal. To average f, we can use the quasi-Monte Carlo approach for $g = f \circ \mathfrak{F}^{-1}$ (where $\mathfrak{F}$ is the above-described CDF of the uniform generation in a timed polytope $\mathcal{P}$) as follows. We first generate a low-discrepancy sequence of vectors in the unit box, next we apply $\mathfrak{F}^{-1}$ and then f to the sequence, compute the average to get an estimate of the expectation $\int_{\mathcal{P}} f(\boldsymbol{t})d\boldsymbol{t}/\mathtt{Vol}(\mathcal{P})$. Another application is to estimate the size (volume measure) of a subset E of the timed polytope $\mathcal{P}$ (corresponding for instance to the set of input stimuli leading to incorrect behaviours). To this end, it suffices to define $g = \chi_{\mathfrak{F}(E)}$ where χ_A is the indicator function of a set A.

Providing different types of guarantees (statistical vs. deterministic), the uniform random and low-discrepancy generation methods are complementary. It is important to emphasise that the low-discrepancy generation method does not require estimating the star discrepancy, which is indeed computationally costly. Only after the testing process is done, the star discrepancy or more generally the Kolmogorov-Smirnov statistic (which will be introduced in the next section) are estimated to evaluate the testing results. This information is useful for deciding whether additional test cases are needed.

5 Evaluating the Uniformity Degree

To evaluate the level of confidence in the testing results, we now address the second problem stated in the introduction. In the sampling-based framework described thus far, this problem can be formulated as evaluating the quality of an arbitrary sample in estimating a quantitative guarantee. Since the approximation quality of both Monte Carlo and quasi-Monte Carlo methods depends on the uniformity degree of the sampled point set (which indicates how close the distribution of this set is to the uniform distribution), we are interested in evaluating the uniformity degree of a given point set.

5.1 Visualising n-dimensional Uniformity Degree via Histograms

One practical way to evaluate the uniformity degree is visualisation. For 2-dimensional samples, we have already visualised in Fig. 2, the difference in the uniformity degree between the sets sampled using different methods. For clouds of points in dimensions higher than 2, we propose the following visualisation method based on histograms.

Example 3. We modify slightly the TA of Fig. 1 to ensure that every delay is bounded by 2. To do so, it suffices to add a clock y that is reset at each transition and must satisfy the condition $y < 2$ for the transition to be taken. This ensures that the timed polytope associated to the discrete path of length n is included in the box $[0, 2]^n$. We draw timed words (using a sampling method) and for each box $\prod_{i=1}^{n}[b_i, b_{i+1}]$ we count the number of times this box is *hit* by a sampled timed word. Since each box has volume 1, every box that is fully included in the language has probability to be hit equal to $1/\mathtt{Vol}(L_n)$. The other have a lower probability which is proportional to the volume of their intersection with the language. To visualise the boxes, we number each box with a binary representation given by the lower bounds of the box. Formally, $\prod_{i=1}^{n}[b_i, b_i + 1]$ is numbered by $\sum_{i=1}^{n} b_i 2^{n-i}$. For instance with $n = 5$, the number 25, the binary representation of which is 11001, is assigned to the box $[1, 2] \times [1, 2] \times [0, 1] \times [0, 1] \times [1, 2]$. Fig. 3 shows the histograms of the hitting count for each box included in $[0, 2]^5$ after drawing $5, 000, 000$ timed words. All the boxes intersect the language, and the purple bars correspond to the boxes fully included in the language that we call hereafter purple boxes. We can observe from the histograms a great similarity between the uniform and low-discrepancy sampling methods. As expected, when restricted to the purple bars, their histograms are flat because the probability for each purple box to be hit is the same and equal to $1/\mathtt{Vol}(L_5)$. We can see that the isotropic sampling is clearly not uniform on the purple boxes and it over-samples the green boxes.

5.2 Measuring the Uniformity Degree

Another evaluation method is to characterise the uniformity degree using the Kolmogorov-Smirnov (KS) test, which is a statistical test to measure how well a

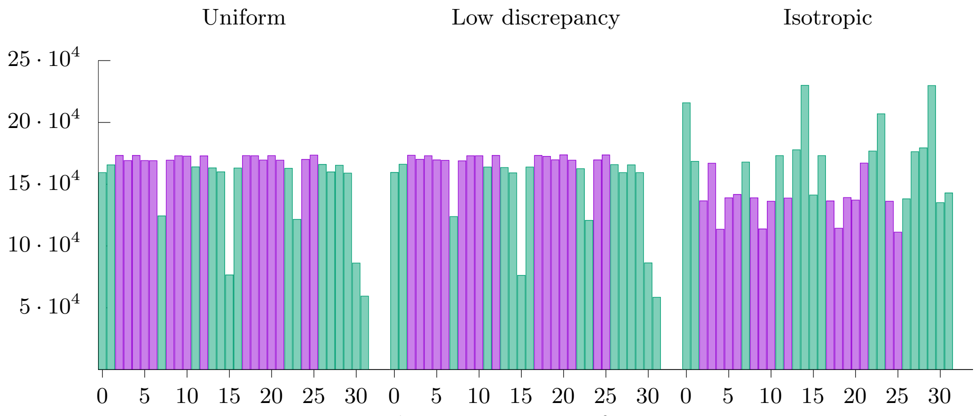


Fig. 3. Histograms for Ex. 3

sample S of points fits a distribution given by a known CDF F. We first point out the relation between this test and the star discrepancy, which allows us to exploit the backward use of $\mathfrak{F}^{-1}$ (from a timed polytope to the unit box) to estimate the KS statistic. The Kolmogorov-Smirnov statistic is defined by the following value (which is a random variable when the sample is drawn at random):

$$\mathbf{KS}(F, S) = \sup_{\boldsymbol{p} \in \mathbb{R}^n} |F(\boldsymbol{p}) - \tilde{F}_S(\boldsymbol{p})|$$

where $\tilde{F}_S$ is the empirical CDF associated with the sample S defined as $\tilde{F}_S(\boldsymbol{p}) = |\{\boldsymbol{p}' \in S \mid \boldsymbol{p}' \leq \boldsymbol{p}\}|/|S|$, which is the ratio of the number of points in S that falls in the box $[-\infty, p_1] \times \ldots \times [-\infty, p_n]$. Let F_U be the CDF associated to n i.i.d. uniform random variables on $[0, 1]$, then $F_U(\boldsymbol{p})$ is the volume of the box $[\mathbf{0}, \boldsymbol{p}]$, and the KS statistic $\mathbf{KS}(F_U, S)$ is nothing else than $D_\star(S)$, that is the *star discrepancy* introduced in the preliminaries and used in the Koksma-Hlawka inequality (1). This connection is known (see for example [21,23]). Theorem 2 is a basis for the following observation. Given a sample S, one can translate $\mathbf{KS}(F, S)$ into $\mathbf{KS}(F_U, F(S))$. The former is the multi-dimensional KS statistic of S w.r.t. a CDF F, and the latter is the KS statistic of the transformed sample $F(S) = \{F(\boldsymbol{p}) \mid \boldsymbol{p} \in P\}$ w.r.t. the uniform distribution F_U on the unit box. The latter is, as mentioned before, the star discrepancy of $F(S)$. Applying this observation to the CDF $\mathfrak{F}$ of the uniform distribution on a timed polytope, we have $\mathbf{KS}(\mathfrak{F}, S) = D_\star(\mathfrak{F}(S))$. Note that when S is obtained via uniform (resp. low-discrepancy) sampling then $S = \mathfrak{F}^{-1}(S')$ where S' is a sample of uniform random vectors (resp. a low-discrepancy sample). So in that case $\mathbf{KS}(\mathfrak{F}, S) = D_\star(\mathfrak{F}(\mathfrak{F}^{-1}(S'))) = D_\star(S')$ and the KS test that requires the KS statistic to be below a threshold passes with high probability (resp. for sure).

Example 4. For the running example, we compute $D_\star(\mathfrak{F}(S))$ of a sample S of timed words drawn using the three sampling methods. After mapping the generated timed vectors back to the unit box, we estimate the star discrepancy of the resulting points. The estimation based on a grid provides a lower and an upper bound on the star discrepancy value [28]. For the set generated by the low-discrepancy method (Fig. 2-(b_1)), the star discrepancy estimation interval is

$[0.009, 0.020]$, by the uniform method (Fig. 2-(a_1)) $[0.040, 0.047]$; by the isotropic method with a uniform random sequence (Fig. 2-(a_4)) $[0.256, 0.266]$, and by the isotropic method with a low-discrepancy sequence (Fig. 2-(b_4)) $[0.250, 0.257]$. From these results, regarding the KS statistics, we observe that the low-discrepancy and uniform methods are clearly better than the isotropic method. It is worth noting that although the points in the unit box generated by the isotropic method from the low-discrepancy sequence look in the figure more "regular", their discrepancy is far larger.

6 Application to CPS Testing

6.1 CPS Testing

Our development thus far focuses on timed words, which can be thought of as an abstraction of real-valued signals. In order to achieve a procedure for CPS testing with guarantees, we show how to define real-valued signals from timed words. We also discuss how timed automata can be used to specify temporal constraints of input signals. Before continuing, let us sketch our procedure.

1. Specifying the temporal constraints on input signals by a timed automaton.
2. Generating a sample of timed words of the timed automaton, using either the uniform method or the low-discrepancy method.
3. Mapping the generated timed words to input signals.
4. Simulating the model or executing the system under the input signals.
5. Determining the guarantees: the uniform method produces statistical guarantees (such as, probability of satisfaction) while the low-discrepancy method produces deterministic ones (such as, error bound on the ratio of correct behaviours or on the satisfaction robustness).

If the testing process uses an arbitrary sample of timed words we can still evaluate its generation quality by the step 5. The steps 2 and 5 have been discussed in the previous sections. Before proceeding with the remaining steps, we point out the novelty of our approach. The existing approaches, such as S-Taliro [3] and Breach [16], use a parametrisation of input signals to reduce the involved infinite dimensional optimisation problem to a finite dimensional one. Such a parametrisation (based on a fixed time discretisation producing to a fixed sequence of time stamps) does not directly capture temporal constraints on input signals. This may lead to a large number of non-realistic test scenarios that are explored by the optimiser but then need to be discarded. With the ability of generating valid signals satisfying temporal constraints, our approach can consider a larger variety of time discretisations leading to better coverage. Also, our approach can reduce the search space and aim at good coverage only over the valid signal space. Furthermore, our generation methods can generate parametric signals; for example, the signal values for the time intervals between the transitions need not be fixed but are represented as parameters over which optimisation can be used, as in the existing optimisation-based approaches. In terms of complexity, if we use the existing optimisation-based approaches, the number of the parameters (that is the number of the optimisation variables) corresponds to the path length in our approach. Both of the generation methods (low-discrepancy and uniform) require computing the CDF which, as mentioned earlier, can be done in polynomial time w.r.t. the path length. In other words, compared to these methods, our signal generation methods do not add much computation efforts and enable more efficient search since all the generated signals are relevant.

Specifying temporal constraints on signals using timed automata. Timed automata are a popular tool for specifying temporal properties of various types of systems [8]. In this section, we only illustrate the usefulness of timed automata in specifying two common properties of signals arising in CPS applications. The first is *bounded variability*, meaning that within any time period of duration T_p there cannot be more than m events. Another definition is to state that for every integer $1 \leq k \leq n-m$ the sum of delays $t_k + \ldots + t_{k+m-1}$ is always greater than T_p. This can be measured by a clock that can be reused every m transitions, so it suffices to have m clocks (one per congruence class modulo m). This is illustrated in our running example: every sequence of $m = 4$ delays needs more than $T_p = 1$ time units to occur.

The second property is a perturbed periodic pattern, which specifies that some m events occur during a period of $[T_p, T_p + \varDelta_p]$ time units and that during this period the delays are in the prescribed intervals. This perturbed periodic pattern is used in the sequel to create input signals to a model of $\varSigma\varDelta$ modulator.

From timed words to signals. A mapping can be directly defined when the timed words yield directly Boolean signals which switch between True or False values after a time delay. As we will show later, such Boolean signals can model the fact that a battery is turned on or off in the KiBaM model used in our experimentation. Another straightforward mapping can be defined to obtain signals that are piecewise constant taking values in a finite set. In the above-described case of uncertain periodic pattern, during a period the signals take a predefined sequence of values (this can model for instance a discretised sinus function) and each change of value occurs after a time delay. A more general way of mapping is to use *retiming* functions, motivated by sampled-data systems and more generally embedded control systems. A retiming function can specify perturbation in terms of imprecision and delay in sampling and communication.

6.2 Experimentation

Our Tool Chain. We implemented our workflow using 4 tools: PRISM [22], SageMath [27], Cosmos [5] and Simulink®. The workflow is depicted in Fig. 4. The first steps are similar to those in [6], their output is a stochastic process generating timed words in the language of the automaton. Cosmos then simulates the stochastic process using Monte Carlo or quasi-Monte Carlo sampling (box “Signal Generation” in Fig. 4). The obtained timed traces are used to generate

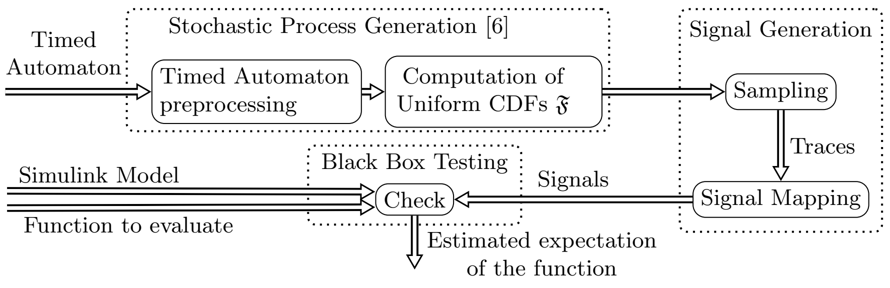
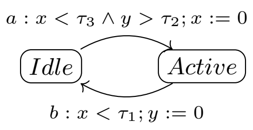

Timed Automaton
Stochastic Process Generation [6]
Timed Automaton preprocessing
Computation of Uniform CDFs $\mathfrak{F}$
Signal Generation
Sampling
Traces
Signal Mapping
Signals
Black Box Testing
Check
Simulink Model
Function to evaluate
Estimated expectation of the function

Fig. 4. Tool chain for black-box CPS validation with input temporal constraints.

real-valued signals which are then fed to the Simulink model we want to test.

Example 1 - KiBaM Model. We first illustrate our workflow on an example of a micro-controller powered by a battery using a Kinetic Battery Model (KiBaM). The goal is to show that if the micro-controller follows its specification in terms of energy usage pattern then the battery will last a certain amount of time. The KiBaM battery model is easily described by a system of ODE. The main feature of this model is that it reflects well the ability of the battery to "self-recharge" when idle. The controller oscillates between two states: the idle state where it consumes a very small amount of power and the active state where it consumes more. The energy usage pattern specification is that the controller may not stay active for more than τ_1 time units but needs to be active every τ_3 time units and when it is idle it waits at least τ_2 time units until it becomes active again. During the operation, the controller drains the battery if it stays active for too long, but the battery restores itself when the controller is idle. Eventually the battery will be completely drained. The property we want to check is that the battery lasts more than T time units: `always_[0,T] (BatteryCharge > 0)`. Fig. 6 depicts


$a : x < \tau_3 \wedge y > \tau_2 ; x := 0$
Idle
Active
$b : x < \tau_1 ; y := 0$

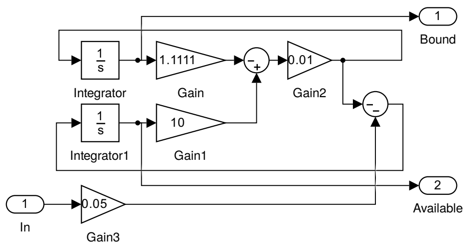


Fig. 5. KiBaM model: Simulink diagram and 2-state controller. When the state of the controller is *Idle* (resp. *Active*), the input port (In) receives the value 0 (resp. 1).

a simulation trace of the battery and the controller. One can observe that when the controller is active, the available charge quickly drops, but as the controller quickly returns to the idle state, the battery is able to self-recharge. This trace

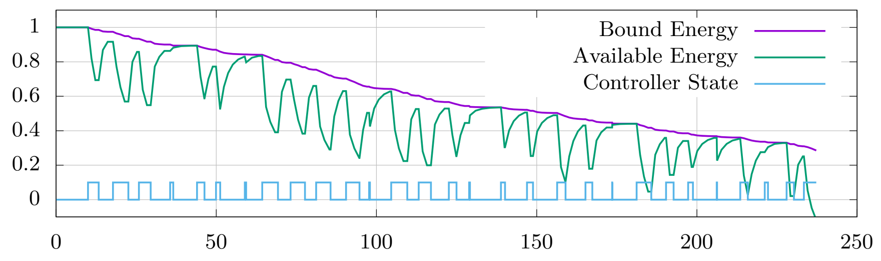


Fig. 6. Co-simulation of the automaton with uniform sampling and the KiBaM model. The energy unit is a fraction of initial capacity.

was sampled with uniform sampling. We performed experiments with the three sampling methods using $10,000$ trajectories. For $T = 271$, the property holds on every trajectory for all methods. For $T = 272$, the estimated satisfaction probability is in $[0.948, 0.959]$ for the uniform sampling and $[0.953, 0.963]$ for the low-discrepancy sampling using Halton's sequence. The confidence level is 0.99 and the total computation time is around 1 minute, where a dominating part of 45 seconds was used for the CPS simulation, and 15 seconds for the stochastic process simulation and signal generation. The CPS simulation requires to numerically solve differential equations which is costly.

Example 2 - $\Sigma\Delta$ Modulator. $\Sigma\Delta$ analog-to-digital converters are widely used for analog signals of a large range of frequencies. This is a typical mixed-signal circuit comprising of an analog component (modulator) and a digital component (digital signal processor for filter and decimation). The most basic architecture of the modulator contains a 1-bit DAC (comparator), a 1-bit DAC (switch), and one or more integrators. $\Sigma\Delta$ modulator stability analysis is a challenging problem. When instability occurs, low frequency signal at the input port of the quantizer alternates between the minimum and maximum magnitudes, which causes the quantizer output to get saturated and the modulator can no longer track the input signal. This constitutes a major non-linearity of the modulator. In this work we apply our methods of signal generation to test if a saturation can occur in a $\Sigma\Delta$ modulator. We use a behavioral model of a second-order modulator specified using Simulink®, which takes into account most non-idealities [11] (see Fig. 7), including sampling jitter, integrator noise, op-amp parameters (finite gain, finite bandwidth, slew-rate and saturation voltages). In terms of model complexity, this Simulink model is heterogeneous including embedded Matlab code and mixing discrete-time and continuous-time components, which goes beyond the applicability of the existing formal verification tools. We also remark that formal verification has previously applied to check the saturation occurrence for a much simpler discrete-time $\Sigma\Delta$ modulator model without non-idealities, for which it is possible to derive its dynamics equations and thus optimization can be formulated and solved [14]. This simple model was also treated by a statistical model-checking approach which picks uniformly an input value at random at each time step [12]. We consider a class of quasi-periodic signals with the frequency spectrum satisfying the nominal range required for the correct

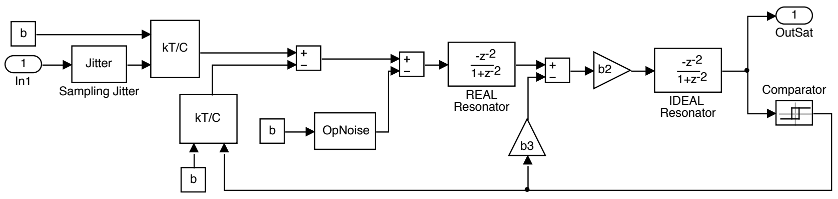


Fig. 7. High-level view of the $\Sigma\Delta$ model with non-idealities [11]

operation of the modulator. The temporal pattern of the considered signals is specified by a variant of the automaton in Figure 1. Each period ranges between 10 et 16, and the delay between two transitions between 1 and 3. The signal value range is discretised into 4 integer values from 0 to 3. We generate a set of 100 timed words of length 300 with the uniform sampling and low-discrepancy methods. The signals are constructed by linear interpolation between the values at the time stamps and then fed to the Breach tool [16], which evaluates the robustness of simulation traces. The STL specification [17] expressing the absence of saturation is `always_[0, sim_time] (abs(OutSat1[t]) < 2)`. Note that we focus on the first integrator since its non-idealities cannot be attenuated by the noise shaping. To test different frequency range, we scale the time stamps with different factors. For the scaling factors $\kappa \geq 0.8 \times 10^{-7}$, the two methods detected a saturation situation. With $\kappa = 0.6 \times 10^{-7}$ the low-discrepancy method detected a saturation while the uniform method did not. For $\kappa \leq 0.5 \times 10^{-7}$, both methods did not detect a saturation, which can be explained by these high frequencies getting closer to the oversampling frequency ($F_s = 42MHz$). This experiment showed the interest of the low-discrepancy generation method. The timed word and signal generation took about 30 seconds, while the average Simulink simulation time was 58 seconds for simulating 100 trajectories.

7 Conclusion

We have extended the work on uniform random generation of runs of timed automata, leading to two new contributions. The first one is a new method for low-discrepancy generation, which is an alternative to the uniform random generation, providing deterministic guarantees. The second contribution is a method for validation of complex CPS models which go beyond the scalability of formal verification and are treated in our approach as black boxes. The ability to handle temporal constraints on input signals is also a novelty in this context. This work opens a number of directions for future work. First, the star discrepancy calculation is a difficult problem. The grid-based estimation method used in this work becomes expensive in high dimensions when a good estimate is needed. We plan to explore methods based on the points in the sample to identify points with jumps in the empirical CDF that affect the supremum result. Additionally, we plan to combine the sampling-based approach with optimisation within the signal value space.

Acknowledgements. This work is partially supported by the IDEX project SYMER, funded by Université Grenoble Alpes.

References

1. Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 12(2s), May 2013.
2. 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, Part I*, pages 483–503, 2017.
3. 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.
4. Eugene Asarin, Nicolas Basset, and Aldric Degorre. Entropy of regular timed languages. *Information and Computation*, 241:142–176, 2015.
5. Paolo Ballarini, Benoît Barbot, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: A new approach for performance evaluation and model checking from concepts to experimentation. *Performance Evaluation*, 90(0):53 – 77, 2015.
6. 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.
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. 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. 2018.
9. Nicolas Basset. A maximal entropy stochastic process for a timed automaton. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP (2)*, LNCS 7966, pages 61–73. Springer, 2013.
10. Dimitri Bohlender, Harold Bruintjes, Sebastian Junges, Jens Katelaan, VietYen Nguyen, and Thomas Noll. A review of statistical model checking pitfalls on real-time stochastic models. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, volume LNCS 8803. 2014.
11. S. Brigati, F. Francesconi, P. Malcovati, D. Tonietto, A. Baschirotto, and F. Maloberti. Modeling Sigma-Delta modulator non-idealities in simulink(r). In *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI*, volume 2, pages 384–387 vol.2, May 1999.
12. 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.
13. Edmund M. Clarke and Paolo Zuliani. Statistical model checking for cyber-physical systems. In *Automated Technology for Verification and Analysis, 9th International*

Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings, pages 1–12, 2011.

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. 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.
16. 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.
17. 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.
18. 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.
19. J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, December 1964.
20. E. Hlawka. Discrepancy and Riemann integration. *Studies in Pure Mathematics*, page 121–129, 1971.
21. Ana Justel, Daniel Peña, and Rubén Zamar. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997.
22. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. CAV'11*, 2011.
23. Jia-Juan Liang, Kai-Tai Fang, Fred Hickernell, and Runze Li. Testing multivariate uniformity and its applications. *Mathematics of Computation*, 70(233):337–355, 2001.
24. Harald Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.
25. Martin Roberts. The unreasonable effectiveness of quasirandom sequences. Online article available at `http://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/`.
26. Murray Rosenblatt. Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23(3):470–472, 1952.
27. W. A. Stein et al. *Sage Mathematics Software (Version 6.9)*. The Sage Development Team, 2015. `http://www.sagemath.org`.
28. Eric Thiémard. An algorithm to compute bounds for the star discrepancy. *Journal of complexity*, 17(4):850–880, 2001.
29. Shakiba Yaghoubi and Georgios Fainekos. Falsification of temporal logic requirements using gradient based local search in space and time. In *6th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2018, Oxford, UK, July 11-13, 2018*, pages 103–108, 2018.