Towards concolic testing for hybrid systems

Pingfan KONG

Yi LI

Xiaohong CHEN

Jun SUN
Singapore Management University, junsun@smu.edu.sg

Meng SUN

See next page for additional authors

Citation

Towards Concolic Testing for Hybrid Systems

Pingfan Kong[1(✉)], Yi Li[2], Xiaohong Chen[1,3], Jun Sun[1(✉)], Meng Sun[2], and Jingyi Wang[1]

[1] Singapore University of Technology and Design, Singapore, Singapore
{pingfan_kong,sunjun}@sutd.edu.sg
[2] LMAM & DI, School of Mathematical Sciences, Peking University, Beijing, China
[3] University of Illinois at Urbana-Champaign, Champaign, USA

Abstract. Hybrid systems exhibit both continuous and discrete behavior. Analyzing hybrid systems is known to be hard. Inspired by the idea of concolic testing (of programs), we investigate whether we can combine random sampling and symbolic execution in order to effectively verify hybrid systems. We identify a sufficient condition under which such a combination is more effective than random sampling. Furthermore, we analyze different strategies of combining random sampling and symbolic execution and propose an algorithm which allows us to dynamically switch between them so as to reduce the overall cost. Our method has been implemented as a web-based checker named HyChecker. HyChecker has been evaluated with benchmark hybrid systems and a water treatment system in order to test its effectiveness.

1 Introduction

Hybrid systems are ever more relevant these days with the rapid development of the so-called cyber-physical systems and Internet of Things. Like traditional software, hybrid systems rely on carefully crafted software to operate correctly. Unlike traditional software, the control software in hybrid systems must interact with a continuous environment through sensing and actuating. Analyzing hybrid systems automatically is highly nontrivial. With a reasonably precise model of the entire system (e.g., in the form of a hybrid automaton), analyzing its behaviors (e.g., answering the question whether the system would satisfy a safety property) is challenging due to multiple reasons. Firstly, the dynamics of the environment, often composed of ordinary differential equations (ODE), is hard to reason about. For instance, there may not be closed form mathematical solutions for certain ODE. Secondly, unlike in the setting of traditional model checking problems, the variables in the hybrid models are often of real type and the (mode) transitions are often guarded with propositional formulas over real variables. There have been theoretical studies on the complexity of analyzing hybrid systems. For instance, it has been proved that non-trivial verification and control problems on non-trivial nonlinear hybrid systems are undecidable [19,22]. As a result, researchers have proposed to either work on approximate models of hybrid systems [18,23], or adopt approximate methods and tools on the hybrid system models [5,6,17].

J. Fitzgerald et al. (Eds.): FM 2016, LNCS 9995, pp. 460–478, 2016.
DOI: 10.1007/978-3-319-48989-6_28

One line of research (which we believe is relevant) is on analyzing the behaviors of hybrid systems through *controlled* sampling. One example of those sampling-based methods is [17]. The idea is to approximate the behavior of a hybrid system probabilistically in the form of *discrete time Markov chains* (DTMC). The complex dynamics in hybrid automata model is approximated using numeric differential equations solvers, and the mode transitions are approximated by probabilistic transition distributions in Markov chains. Afterwards, methods like hypothesis testing can be applied to the Markov chain to verify, probabilistically, properties against the original hybrid model.

While sampling-based methods like [17] are typically more scalable, there are limitations. Arguably, the most important one is that random sampling does not work well when the system contains *rare events*, i.e., events which by definition are unlikely to occur through random sampling. When systems get complicated, every event becomes rarer in a way. Existing remedies for this problem include importance sampling [6] and importance splitting [25], which work by essentially increasing the probability of the rare events. Both approaches are however useful only in certain limited circumstances.

One potential remedy for the problem is concolic testing, which is a technique proposed for analyzing programs [15,36]. The idea is: if random sampling fails to fire certain transitions in certain state (i.e., a potential rare event), we apply symbolic execution to generate the specific inputs which would trigger the transition or to show that the transition is infeasible. In this work, we investigate the possibility of applying concolic testing to hybrid systems. In particular, we study two fundamental questions. One is under what condition combining random sampling and symbolic execution is beneficial, i.e., given a property, is it guaranteed to find a counterexample with a smaller number of samples? The other is, among different strategies of combining random sampling and symbolic execution (i.e., when and how to apply symbolic execution), how do we define and identify the more effective strategies? We remark that the latter question is particularly relevant to the analysis of hybrid systems as symbolic execution for hybrid automata is often very time consuming and thus a good strategy should perhaps be: applying symbolic execution as minimum as possible. Based on the answers, we then design an algorithm which adopts a strategy to dynamically switch between random sampling and symbolic execution. Intuitively, it works by continuously estimating whether certain transition is rare or not and applying symbolic execution only if the transition is estimated to be rarer than certain threshold. Furthermore, the threshold is calculated according to a cost model which estimates the cost of symbolic execution using certain constraint solver. Our method has been implemented as a self-contained web-based checker named HyChecker and evaluated with benchmark hybrid systems as well as a water treatment system in order to test its effectiveness.

The remainders of the paper are organized as follows. In Sect. 2, we define a DTMC interpretation of hybrid system models. In Sect. 3, we view symbolic execution as a form of importance sampling and establish a sufficient condition for importance sampling to be beneficial. In Sect. 4, we discuss strategies on combining random sampling and symbolic execution. In Sect. 5, we present our implementation and evaluate its effectiveness. In Sect. 6, we conclude and review related work.

2 A Probabilistic View

In this section, we present a probabilistic interpretation of hybrid system models, which provides the foundation for defining and comparing the effectiveness of random sampling, symbolic execution or their combinations. In this work, we assume that hybrid systems are modelled as hybrid automata [20]. The basic idea of hybrid automata is to model different discrete states in a hybrid system as different *modes* and use differential equations to describe how variables in the system evolve through time in the modes. For simplicity, we assume the differential equations are in the form of ordinary differential equations (ODEs).

Definition 1. *A* hybrid automaton *is a tuple* $\mathcal{H} = (Q, V, q_0, I, \{f_q\}_{q \in Q}, \{g_{(q,p)}\}_{\{q,p\} \subseteq Q})$ *such that* Q *is a finite set of modes;* V *is a finite set of state variables;* $q_0 \in Q$ *is the initial mode;* $I \subseteq \mathbb{R}^n$ *is a set of initial values of the state variables;* f_q *for any* $q \in Q$ *is an ODE describing how variables in* V *evolve through time at mode* q*; and* $g_{(q,p)}$ *for any* $q, p \in Q$ *is a guard condition on transiting from mode* q *to mode* p*.*

For simplicity, we often write $q \xrightarrow{g} p$ to denote $g_{(q,p)}$. For example, the hybrid automaton shown on the left of Fig. 1 models an underdamped oscillatory system [16], such as a spring-mass or a simple pendulum with a detector that raises an alarm whenever the displacement x exceeds the threshold a. The initial displacement $x(0) = 0$, while its tendency to deviate from the equilibrium $x'(0) \in [0, 2\pi]$. An alarm is raised when the system enters mode q_e, which is reachable only through the transition $q_0 \xrightarrow{x>a} q_e$.

Next, we define the semantics of a hybrid automaton in the form of an infinite-state labeled transition system (LTS).

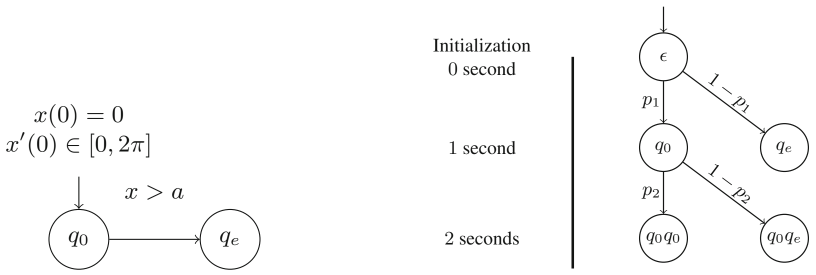

Fig. 1. An oscillatory system: the ODE at q_0 is $x'' + x' + 4\pi^2 x = 0$ and the one at q_e is $x' = 0$

Definition 2. *Let $\mathcal{H} = (Q, V, q_0, I, \{f_q\}_{q \in Q}, \{g_{(p,q)}\}_{\{p,q\} \subseteq Q})$ be a hybrid automaton. The semantics of $\mathcal{H}$, written as sem($\mathcal{H}$), is an LTS $(S, S_0, T, \rightarrow)$, where $S = \{(q, v) \mid q \in Q$ and $v \colon V \rightarrow \mathbb{R}^n\}$ is the set of all (concrete) states; $S_0 = \{(q_0, v) \mid v \in I\}$ is the set of initial states; $T = \mathbb{R}_+ \cup \{\epsilon\}$ is the set of transition labels, where ϵ is a label for all discrete jumps; and $\rightarrow \subseteq S \times T \times S$ contains two sets of transitions. One is time transitions, i.e., $(q, v) \xrightarrow{t} (q, u)$ if there exists a solution ξ to the differential equation $\mathrm{d}V/\mathrm{d}t = f_q(V)$ such that $\xi(0) = v$ and $\xi(t) = u$. The other is jumps, i.e., $(q, v) \xrightarrow{\epsilon} (p, v)$ where there exists a transition $q \xrightarrow{g} p$ in $\mathcal{H}$ such that v satisfies g.*

A finite *run* ρ of $\mathcal{H}$ is a finite sequence of transitions of *sem*($\mathcal{H}$). Since we are investigating random sampling and symbolic execution (both of which are limited to finite runs), we focus on runs of *bounded* length. For simplicity, we assume that all finite runs can be extended to an infinite non-Zeno run (such that time elapses unboundedly [20]). It is straightforward to see that there always exists a time unit $\Delta t > 0$ such that at most one jump (i.e., ϵ-transition) occurs with Δt time units. In the following, we simply assume that Δt is defined as one time unit. As a result, by observing the system mode at the end of every time unit, we can obtain a *trace* of $\mathcal{H}$ as $\pi = q_0 q_1 \ldots q_k$, i.e., the sequence of modes observed during the run. We remark that if there is no jump during the time unit, the same mode is observed.

In the following, we focus on reachability analysis of certain modes [17], i.e., certain modes in $\mathcal{H}$ are considered negative and we would like to check if any of them is reachable. We remark that the verification problem of properties expressed in BLTL formula [27] can be reduced to reachability analysis [17]. For instance, in the example shown in Fig. 1, the safety property is reduced to whether the negative mode q_e is reachable or not (within certain time). A trace is positive if it contains no negative mode. It is negative (a.k.a. a counterexample) if it contains at least one negative mode.

Next, we introduce a Markov chain interpretation of $\mathcal{H}$, adopted from [17]. Without loss of generality, we assume a uniform probability distribution on all initial states. This uniform distribution naturally induces a probability distribution over the traces of the system. Recall that a transition $q \xrightarrow{g} p$ of $\mathcal{H}$ can be fired only when g is satisfied. Suppose the system is in the state (q, v) initially and becomes (q, v_t) after a time transition of t. If v_t satisfies g, the transition is enabled. We denote the set of all points in time within $(0, 1)$ when the mode transition $q \xrightarrow{g} p$ is enabled as

$$T_q(v, g) = \{t \in (0, 1) \mid \theta_q(v, t) \text{ satisfies } g\}, \tag{1}$$

where $\theta_q(v, \cdot)$ is the solution of the ODE at mode q with the initial value v. If the transition is fired at some time point $t \in T_q(v, g)$, the following state is observed after 1 time unit: $(p, \theta_p(v_t, 1 - t)) = (p, \theta_p(\theta_q(v, t), 1 - t))$. That is, the new mode is p and the variables evolve according to the ODE at mode q for t time unit and then according to the ODE at mode p for $1 - t$ time unit. For simplicity, we write $v_{q,p}(v, t)$ to denote the variable state reached from state

(q, v) by firing transition $q \xrightarrow{g} p$ at time t, i.e., $v_{q,p}(v, t) = \theta_p(\theta_q(v, t), 1 - t)$. Furthermore, we write $v_{q,p}(v)$ to denote the set of all variable states reached from state (q, v) by firing transition $q \xrightarrow{g} p$ at any time the transition is enabled, i.e., $v_{q,p}(v) = \{v_{q,p}(v, t) \mid t \in T_q(v, g)\}$.

By our assumption on the uniform random sampling, there is a uniform distribution over $T_q(v, g)$. This uniform distribution, denoted as $\mathbf{U}(T_q(v, g))$, naturally induces the following probability distribution over $v_{q,p}(v)$

$$\mathbf{P}(Y) = \int_{t \in T_q(v,g)} \frac{[v_{q,p}(v, t) \in Y]}{\|T_q(v, g)\|} \mathrm{d}t \tag{2}$$

for any $Y \subseteq v_{q,p}(v)$, where $[\,\cdot\,]$ is the Iverson bracket [24] and $\|\cdot\|$ is the Lebesgue measure [29]. Intuitively, if initially the system is in the state (q, v), we obtain a probability distribution over all possible states after taking the transition.

Next, we generalize the result so as to compare the probability of taking different transitions from different initial states. We assume a probability space $(X, \mathbf{P})$ where $X \subseteq \mathbb{R}^n$, and the automaton $\mathcal{H}$ starts from a state (q, v) with $v \sim \mathbf{P}$. Let $q \xrightarrow{g_i} p_i$ where $i \in \{1, \ldots, m\}$ be the transitions from q. Given an initial state (q, v), the time window in which the transition to p_i is enabled is $T_q(v, g_i)$. We assume that the system does not favor certain transitions and the probability of taking a transition is proportional to the size of the time window in which that transition is enabled. In other words, the probability of taking the transition $q \xrightarrow{g_i} p_i$ from state (q, v) is defined as $p_{q,p_i}(v) = \|T_q(v, g_i)\| / \sum_{j=1}^{m} \|T_q(v, g_j)\|$. According to the law of total probability, we have

$$p_{q,p_i} = \int_{v \in X} \frac{\|T_q(v, g_i)\|}{\sum_{j=1}^{m} \|T_q(v, g_j)\|} \mathrm{d}\mathbf{P}(v). \tag{3}$$

Furthermore, assume the transition $q \xrightarrow{g_i} p_i$ is fired. Given the condition that v is a fixed v_0, we know the conditional probability distribution over $v_{q,p}(v_0)$, for any $Y \subseteq v_{q,p}(v_0)$, is defined as:

$$\mathbf{P}(Y \mid v = v_0) = \int_{t \in T_q(v_0,g)} \frac{[v_{q,p}(v_0, t) \in Y]}{\|T_q(v_0, g)\|} \mathrm{d}t.$$

By the law of total probability, for any $Y \subseteq v_{q,p}(X)$, we have

$$\begin{aligned} \mathbf{P}(Y) &= \int_{v \in X} \mathbf{P}(Y \mid v) \mathrm{d}\mathbf{P}(v) \\ &= \int_{v \in X} \int_{t \in T_q(v,g)} \frac{[v_{q,p}(v, t) \in Y]}{\|T_q(v, g)\|} \mathrm{d}t \mathrm{d}\mathbf{P}(v) \\ &= \int_{v \in X} \int_0^1 \frac{[t \in T_q(v, g) \wedge v_{q,p}(v, t) \in Y]}{\|T_q(v, g)\|} \mathrm{d}t \mathrm{d}\mathbf{P}(v). \end{aligned} \tag{4}$$

Equations (3) and (4) effectively identify a discrete-time Markov chain (DTMC).

Definition 3. *Let $\mathcal{H} = (Q, V, q_0, I, \{f_q\}_{q \in Q}, \{g_{(p,q)}\}_{\{p,q\} \subseteq Q})$ be a hybrid automaton, and K be a bound of trace length. The DTMC associated with $\mathcal{H}$ is a tuple $\mathcal{M}_{\mathcal{H}} = (S, u_0, Pr)$ where a node in S is of the form $(q, X, \mathbf{P}_X)$ where $q \in Q$ is a mode, X is the set of values for V and $\mathbf{P}_X$ is a probability distribution of the values in X; the root $u_0 = (q_0, I, \mathbf{U}_I)$ where $\mathbf{U}_I$ is the uniform distribution over I; and for any $(q, X, \mathbf{P}_X) \in S$, the transition probability $Pr((q, X, \mathbf{P}_X), (p, v_{q,p_i}(X), \mathbf{P}_i))$, where the probability distribution $\mathbf{P}_i$ is defined as in Eq. (4).*

We remark that $\mathcal{M}_{\mathcal{H}}$ abstracts away the complicated ODE in $\mathcal{H}$ and replaces the guarded transitions with probabilistic transitions. A *path* of $\mathcal{M}_{\mathcal{H}}$ with non-zero probability always corresponds to a trace of $\mathcal{H}$ [17]. The partition of positive and negative traces in $\mathcal{H}$ naturally induces a partition of positive and negative paths in $\mathcal{M}_{\mathcal{H}}$. Notice that $\mathcal{M}_{\mathcal{H}}$ is by construction in the form of a tree. The degree of the tree is bounded by the number of modes in $\mathcal{H}$, and its depth is bounded by K, i.e., the bound on trace length.

For instance, following the above discussion, we can construct the DTMC of the oscillatory system shown on the right of Fig. 1. The root node is $s_0 = (q_0, I, \mathbf{U}_I)$ where $I = \{0\} \times [0, 2\pi]$ and $\mathbf{U}_I$ is the uniform distribution over I. There is one outgoing transition $q_0 \to q_e$ at mode q_0. Thus s_0 has two children nodes s_1 and s_2, where s_1 represents automaton taking the transition $q_0 \to q_e$ in the first second, and s_2 represents automaton staying in mode q_0. For this simple example, we can analytically compute the transition probability, e.g., p_1 and p_2 shown in Fig. 1. In general it is difficult.

3 Symbolic Execution as a Form of Importance Sampling

In this section, we analyze the effectiveness of random sampling and symbolic execution based on the DTMC interpretation of $\mathcal{H}$ developed in the previous section. In particular, we review symbolic execution as a form of importance sampling [39], which intuitively speaking alters the probability distribution of $\mathcal{M}_{\mathcal{H}}$ in certain way so that a negative path is more likely to be sampled. In the following, we first define a way of measuring the effectiveness of random sampling, symbolic execution and possibly other sampling methods.

3.1 Bayesian Inference

Recall that traces of $\mathcal{H}$ are partitioned into either positive trace, denoted as Π^+, or negative traces, denoted as Π^-. The probability of the system exhibiting a negative trace is called the *error probability* and is denoted as $\theta = \mathbf{P}(\Pi^-)$. Intuitively, after observing some sample traces (obtained either through random sampling or symbolic execution), we gain certain information on θ. Formally, we investigate the following questions: (1) how do we claim that the error probability θ is bounded by certain tolerance level δ and (2) how do we measure the confidence of the claim?

We answer the questions based on statistical inference. Intuitively, if we see many negative trace samples, we conclude *with certain confidence* that the system is likely to have an error probability that is larger than the tolerance level δ. If we identify few or even no negative traces, we conclude *with certain confidence* that the system is likely to have an error probability within the tolerance level δ. Formally, let random variable X denote whether a sample trace is positive or negative, i.e., $\mathbf{P}(X = 1)$ is the error probability θ. Let $B(N, \theta)$ denote the binomial distribution with parameters $N \in \mathbb{N}$ and $\theta \in [0, 1]$. We have $X \sim B(1, \theta)$. Given N independent and identically distributed sample traces, the number of negative traces is: $m = X_1 + X_2 + \dots X_N \sim B(N, \theta)$. Initially, before witnessing any sample trace, we may only estimate the value of θ based on historical data. We thus assume a *prior knowledge* of θ in the form of a *prior distribution* $f(\theta)$. If no historical data are available, we set the prior distribution to be a non-informative one. In the following, for simplicity, we adopt the non-informative prior distribution $f(\theta) \equiv 1$ where $\theta \in [0, 1]$.

According to the Bayesian law, the *post distribution* of θ after witnessing m negative samples and $n = N - m$ positive samples is defined as follows.

$$f(\theta \mid n, m) = \frac{f(\theta) f(n, m \mid \theta)}{\displaystyle\int_0^1 f(\theta) f(n, m \mid \theta) \mathrm{d}\theta} = \frac{\theta^m (1-\theta)^n}{\mathrm{B}(m+1, n+1)}$$

where $\mathrm{B}(\,\cdot\,,\,\cdot\,)$ is the Beta function [3]. The *confidence* we have about the claim that $\theta < \delta$, denoted as $c(n, m, \delta)$, can be defined naturally as the probability of $\theta < \delta$ conditioned on observing the negative and positive samples. Formally,

$$c(n, m, \delta) = \int_0^\delta f(\theta \mid n, m) \mathrm{d}\theta = \frac{\mathrm{B}(\delta; m+1, n+1)}{\mathrm{B}(m+1, n+1)}$$

where $\mathrm{B}(\delta\,;\,\cdot\,,\,\cdot\,)$ is the incomplete Beta function [3].

The following proposition shows that our definition of confidence is consistent with our intuition, i.e., the more positive samples we observe, the more confidence we have.

Proposition 1. *For any tolerance* $0 < \delta < 1$*,* $c(n, 0, \delta) \to 1$ *as* $n \to \infty$*; and for any* $m > 0$*,* $c(n, m, \delta) \to 1$ *as* $n/m \to \infty$*.*

Thus, if we have dominantly sufficient positive samples, we would always be able to reach a target confidence level. In practice, however, we are always limited in the budget or time, we thus would like to reach a certain confidence level at a low cost. For instance, instead of random sampling, we can apply idea like importance sampling [39] so as to increase the probability of sampling a negative sample and hope to gain the same confidence level with fewer samples. Recall that we can view symbolic execution as a particular way of importance sampling. Compared with random sampling, it essentially alters the probabilistic distribution of the traces so that more probability is associated with those traces following a given path. In the following, we investigate the idea of importance sampling in our setting and establish a condition which must be satisfied so that importance sampling (including symbolic execution) must satisfy in order to be more effective in achieving the same confidence level.

3.2 Importance Sampling

Importance sampling is a widely-used technique in Monte Carlo method in order to approximate the expectation of a probability distribution. The intuition is after observing many positive samples, we should have more confidence in the system's correctness, *if the samples are generated by a method that is more likely to sample a negative one.* We remark that the notion of importance sampling we adopt here has nothing to do with the expectation approximation [39], but rather shares the same idea with the importance sampling in the Monte Carlo method.

Recall that θ is the error probability. The probability of a specific sampling method finding a negative trace is a function of θ, denoted as $\varphi(\theta)$. We refer to $\varphi(\theta)$ as the *effectiveness function* of the sampling method. Furthermore, $\varphi(\theta)$ is assumed to be continuous and strictly increasing on $[0,1]$ with $\varphi(0) = 0$ and $\varphi(1) = 1$. Given a specific sampling method (e.g., random sampling or symbolic execution), we may be able to approximate its effectiveness through empirical study. In certain special cases, we might identify a closed form of the effectiveness function for a specific sampling method. For instance, in the case of random sampling, the effectiveness function $\varphi(\theta) = \theta$. A sampling method with effectiveness $\varphi(\theta)$ is said to be *more effective* than another with effectiveness $\phi(\theta)$, if $\varphi(\theta) > \phi(\theta)$ for all θ. Two sampling methods are called incomparable if no one is more effective than the other.

In the following, we show that a more effective sampling method leads to a higher confidence level. Without loss of generality, we focus on effectiveness functions which can be expressed in the form of a power function $\varphi(\theta) = \theta^\alpha$ where $\theta \in [0,1]$ for $0 < \alpha \le 1$. The reason for the assumption is that effectiveness functions in this form can be compared easily.

Following the discussion in Sect. 3.1, suppose that the effectiveness of a testing method is $\varphi(\theta) = \theta^\alpha$ and we have witnessed m negative samples and $n = N - m$ positive samples. The post distribution can be calculated as follows.

$$f(\theta \mid n, m) = \frac{\theta^{\alpha m}(1-\theta^\alpha)^n}{\displaystyle\int_0^1 \theta^{\alpha m}(1-\theta^\alpha)^n \mathrm{d}\theta} \tag{5}$$

Accordingly, the confidence is defined as follows.

$$c_\varphi(n, m, \delta) = \int_0^\delta f(\theta \mid n, m)\mathrm{d}\theta = \frac{\displaystyle\int_0^\delta \theta^{\alpha m}(1-\theta^\alpha)^n \mathrm{d}\theta}{\displaystyle\int_0^1 \theta^{\alpha m}(1-\theta^\alpha)^n \mathrm{d}\theta}. \tag{6}$$

The following theorem then establishes that a *more effective* sampling method would always result in more confidence.

Theorem 1. *Let* $\varphi(\theta) = \theta^{\alpha}$ *and* $\psi(\theta) = \theta^{\beta}$ *be the effectiveness function of two testing methods. If* $1 \geq \alpha > \beta > 0$*, then* $\varphi(\theta) \leq \psi(\theta)$ *for all* $\theta \in [0, 1]$*, and* $c_{\varphi}(n, m, \delta) \leq c_{\psi}(n, m, \delta)$ *for all* $n, m \in \mathbb{N}$ *and* $\delta \in (0, 1)$*.* $\square$

The (rather involved) proof is presented in [28]. This theorem endorses our intuition that we should have more confidence in the systems' correctness when observing many positive samples, if the samples are generated by a method like symbolic execution (with a bias on negative samples). In general we cannot compare the confidence of two incomparable sampling methods. We remark that this result has not been formally proved before and it serves the foundation for the approach we propose next.

Based on the theorem, in order to apply symbolic execution to achieve better confidence than applying random sampling, we should apply it such that it is more likely to sample a negative trace. There are multiple different strategies on how/when to apply symbolic execution. For instance, we could symbolically execute a path which ends with a negative mode, or a part of the path (e.g., we solve for input values which are required to trigger the first few transitions of a path leading to a negative mode, if we have reasons to believe that only those transitions are unlikely to be fired through random sampling), or even simply symbolically execute a path which has not been visited before if all existing samples are positive. In the next section, we discuss how to compare these different strategies based on cost and propose a cost-effect algorithm.

4 Sampling Strategies

Recall that our objective is to check whether there is a trace which visits a negative mode. Theorem 1 certainly does not imply we should abandon random sampling. The simple reason is that it ignores the issue of time cost. In general, the cost of obtaining a negative trace through sampling (either random sampling or symbolic execution) is: c/pr where c is the cost of obtaining one sample and pr is the probability of the sample being a negative trace. In the case of random sampling, c is often low and pr is also likely low, especially so if the negative traces all contain certain rare event. In the case of symbolic execution, c is likely high since we need to solve a path condition to obtain a sample, whereas pr is likely high. Thus, in order to choose between random sampling and symbolic execution, we would like to know their time cost, i.e., c and pr. While knowing the cost of obtaining one sample through random sampling is relatively straightforward, knowing the cost c of symbolic execution is highly non-trivial. In this section, we assume there is a way of estimating that and propose an algorithm based on the assumption. In Sect. 5, we estimate c empirically and show that even a rough estimation serves a good basis for choosing the right strategy. We can calculate pr based on $\mathcal{M}_{\mathcal{H}}$. However, constructing $\mathcal{M}_{\mathcal{H}}$ is infeasible and thus we propose to approximate $\mathcal{M}_{\mathcal{H}}$ at runtime.

4.1 Probability Estimation

Initially, since we have no idea on the probability of obtaining a negative trace, we apply wishful thinking and start with random sampling, hoping that a negative trace will be sampled. If a negative trace is indeed sampled, we are done. Otherwise, the traces that have been sampled effectively identify a subgraph of $\mathcal{M}_{\mathcal{H}}$, denoted as $\mathcal{M}_{sub}$, which contains only modes and transitions visited by the sample traces. Without any clue on the transition probability between modes not in $\mathcal{M}_{sub}$, it is infeasible for us to estimate pr (i.e., the probability of reaching a negative mode). It is clear however that in order to reach a negative mode, we must sample in a way such that $\mathcal{M}_{sub}$ is expanded with unvisited modes. Thus, in the following, we focus on finding a strategy which is cost-effective in discovering new modes instead.

For each mode u in $\mathcal{M}_{sub}$ where there is an unvisited child mode v, we have the choice of either trying to reach v from u through more random sampling or through symbolic execution (i.e., solving the path condition). In theory, the choice is to be resolved as follows: random sampling if $c_t/q(u) < c_s$ and symbolic execution if $c_t/q(u) \geq c_s$, where c_t is the cost of generating a random sample; c_s is the cost of applying symbolic execution to generate a sample visiting an unvisited child of u in $\mathcal{M}_{sub}$; and $q(u)$ is the probability of finding a new mode from u, i.e., $q(u) = \sum_{v \in V \setminus V_0} p_{uv}$. Intuitively, for random sampling, the expected number of samples to find a new mode is $1/q(u)$ and thus the expected cost of using random sampling to discover a new mode is $c_t/q(u)$. Unfortunately, knowing $q(u)$ and $\mathcal{M}_{sub}$ exactly is expensive. The former is the subject of recent research on model counting and probabilistic symbolic execution [9,11,12,32], and the latter has been studied in [4]. Thus, in this work, we develop techniques to estimate their values.

In our approach, we actively estimate the probability of $q(u)$ (for each u in $\mathcal{M}_{sub}$) from historical observation through Bayesian inference, by recording how many times we sampled u. Assume $q = q(u)$ has a prior distribution $f(q)$. Let A denote the event that an unvisited child v remains unvisited after one sampling, and $\bar{A}$ denote the event that v becomes visited afterwards. Then,

$$f(q \mid A) = \frac{f(q)f(A \mid q)}{\displaystyle\int_0^1 f(q)f(A \mid q)\mathrm{d}q} = \frac{qf(q)}{\displaystyle\int_0^1 qf(q)\mathrm{d}q} \propto qf(q),$$

and similarly: $f(q \mid \bar{A}) \propto (1-q)f(q)$.

Suppose that mode u has been sampled for N times and for m out of N times, we end up with a child which has been visited previously. As a result, $n = N - m$ is the number of times we ended up with an unvisited child. We can compute the post distribution $f(q) \propto (1-q)^m q^n$ and the expectation as follows.

$$E(q) = \frac{\displaystyle\int_0^1 q(1-q)^m q^n \mathrm{d}q}{\displaystyle\int_0^1 (1-q)^m q^n \mathrm{d}q} = \frac{n+1}{m+n+2}.$$

Thus, we estimate $q(u)$ as $(n+1)/(m+n+2)$. Intuitively, the bigger m is, the less likely that an unvisited mode is going to be sampled through random sampling.

Next, we discuss how to apply symbolic execution in this setting. There are multiple strategies on how to apply symbolic execution to construct a sample for visiting v. For instance, we could solve a path condition from an initial mode to v so that it will surely result in a trace visiting v (if the path condition is satisfiable). Alternatively, we could take a sample trace which visits u and apply symbolic execution to see whether the trace can be altered to visit v after visiting u by letting a different amount of time elapsing at mode u. That is, assume that (u, X) is a concrete state of $sem(\mathcal{H})$ visited by a sample trace, where X is a valuation of V. We take the state (u, X) as the starting point and apply symbolic execution to solve a one-step path condition so that v is visited from state (u, X). This is meaningful for hybrid automata because, for every step, by letting a different amount of time elapsing, we may result in firing a different transition and therefore reaching a different mode. We remark that if solving the one-step path condition has no solution, it does not necessarily mean that v is unreachable from u. Nonetheless, we argue that this strategy is justified as, according to Theorem 1, such a sampling strategy would be more effective than random sampling. To distinguish these two strategies, we refer to the former as *global concolic sampling* and the latter *local concolic sampling*. The choice of strategy depends on c_s. We estimate c_s for particular solvers and systems in Sect. 5 and choose the right strategy accordingly.

4.2 Concolic Sampling

Based on the theoretical discussion presented above, we then present our sampling algorithm, which we call concolic sampling. The details are shown in Algorithm 1. The input is a hybrid automaton modeling a hybrid system, where some modes are identified as negative ones. The aim is to identify a trace which visits a negative mode or otherwise report that there is certain probabilistic guarantee on their absence. We remark that we skip the part on how the probabilistic guarantee is computed and refer the readers to [17] for details. We rather focus on our contribution on combining random sampling and symbolic execution for better counterexample identification in the following.

We maintain the set of sample traces as Π in the algorithm. Based on Π, we can construct the above-mentioned subgraph $\mathcal{M}_{sub}$ of $\mathcal{M}_{\mathcal{H}}$ systematically. Next, for each node u in $\mathcal{M}_{sub}$ which potentially has unvisited children, we maintain two numbers m and n as discussed above, in order to estimate the probability of $q(u)$. If according to our strategy, there is still some u such that it might be cheaper to discover a new child mode through random sampling, we proceed by generating a random sample using the algorithm presented in [17], which is shown as Algorithm 2.

We briefly introduce how Algorithm 2 works in the following. In a nutshell, the algorithm is designed to sample a run π according to an approximation of the probability distribution of $\mathcal{M}_{\mathcal{H}}$. The main idea is to use the Monte Carlo method to approximate the measure of time windows $\|T_q(v, g)\|$. Recall that

Algorithm 1. Concolic Sampling

```
1  Let $\Pi$ be the set of sampled runs, initialized to the empty set;
2  Let $\mathcal{M}_{sub}$ be the subgraph of $\mathcal{M}$, initialized to the root node;
3  repeat
4      Set $u = \arg\min_u \min(c_t/E(q(u)), c_s)$;
5      if $c_t/q(u) < c_s$ then
6          Invoke Random Sampling to generate a run $\pi$;
7      else
8          Apply symbolic execution to obtain a sample $\pi$ visiting a new child of $u$;
9      if $\pi$ visits a negative mode then
10         Present $\pi$ as a counterexample and terminate;
11     if $\pi$ visits an undiscovered mode then
12         $n_u := n_u + 1$;
13     else
14         $m_u := m_u + 1$;
15     Add $\pi$ into $\Pi$ and add $u$ to $\mathcal{M}_{sub}$;
16 until time out;
```

Algorithm 2. Random Sampling

```
   Input: A hybrid automaton $\mathcal{H}$ and a state $\langle q, v \rangle$
   Result: A successive state $\langle p, u \rangle$
1  Sample time points $t_1, \cdots, t_J$ uniformly from $[0, 1]$;
2  foreach outgoing transition $q \xrightarrow{g_i} q_i$ do
3      Set $T_i := \{t_j \mid \Phi_q(t_j, v) \models g_i\}$,
4  Choose a transition $q \xrightarrow{g_i} q_i$ with probability $\|T_i\| / \sum_i \|T_i\|$;
5  Sample a time point $t$ uniformly from $T_i$;
6  Set $u := \Phi_{q_i}(1 - t, \Phi_q(t, v))$ and $p := q_i$;
7  Return $\langle p, u \rangle$;
```

$T_q(v, g) = \{t \in (0, 1) \mid \theta_q(v, t)$ satisfies $g\}$. Therefore the measure $\|T_q(v, g)\|$ is the mean of $[\theta_q(v, \tau) \models g]$, where the random variable $\tau \sim \mathbf{U}(0, 1)$. According to the law of large numbers [30], the sample mean almost surely converges to the expectation. Thus we have

$$\frac{\sum_{i=1}^{n}[\theta_q(v, \tau_i) \models g]}{n} \xrightarrow{a.s.} \|T_q(v, g_j)\| \quad \text{as} \quad n \to \infty,$$

where $\tau_1, \tau_2, \ldots, \tau_n \overset{i.i.d.}{\sim} \tau$. To generate a K-step run, Algorithm 2 works by generating one random step at a time. In particular, after each time unit, at line 4, firstly a set of time points are uniformly generated. By testing how often each transition from the current mode is enabled at these time points, we estimate the transition probability in $\mathcal{M}_{\mathcal{H}}$. At line 7, we sample a transition according to the estimated probability and generate a step. This procedure finishes when a run which spans K time units is generated.

If random sampling is unlikely to be cost-effective in discovering a new mode according to our strategy, symbolic execution is employed at line 7 in Algorithm 1 to generate a sample to cover a new node in $\mathcal{M}_{\mathcal{H}}$. Among all the nodes in $\mathcal{M}_{sub}$, we identify the one which would require the minimum cost to discover a new child according to our estimation c_s, encode the corresponding path condition and apply an existing constraint solver that supports ODE (i.e., dReal [14]) to generate a corresponding run. Once we obtain a new run π at line 8, we check whether it is a counterexample. If it is, we report and terminate at line 10. Otherwise, we repeat the same procedure until it times out.

5 Evaluation

We implemented our approach in a self-contained toolkit called HyChecker, available online at [2]. HyChecker is implemented with 1575 lines of Python codes (excluding external libraries we used) and is built with a web interface. HyChecker relies on the dReal constraint solver [14] for symbolic execution. In the following, we evaluate HyChecker in order to answer the following research question: does our strategy on combining random sampling and symbolic execution (resulting from our theoretical analysis) allow us to identify rare counterexamples more efficiently?

Our test subjects include three benchmark hybrid systems which we gather from previous publications as well as a simplified real-world water treatment system.

- *Thermodynamic system.* We first test our method on a room heating system from [10]. The system has n rooms and $m \leq n$ heaters which are used to tune the rooms' temperature. The temperature of a room is affected by the environment temperature and also by whether a heater is warming the room. The system therefore aims to maintain the rooms' temperature within certain comfortable range by moving around and turning on and off the heaters. We consider in the experiment such a system with three rooms and two heaters. We verify the same property as in [17], i.e., whether the two heaters will be moved to other rooms in the first five days.
- *Navigation system.* Our second test subject is the navigation system from [10]. This system contains a grid of cells, where each cell is associated with some particular velocity. Whenever a floating object moves from one cell to the other, it changes its acceleration rate according to the velocity in that cell. If the object leaves the grid, the velocity is the one of the closest cell. We check whether an object in the grid will leave its initial cell and will not enter a dangerous cell, within six minutes.
- *Traffic system.* Our third model is from the long standing research on modeling traffic and examining causes of traffic jams and car crashes. We use the ODE in [34] to describe the dynamics of a vehicle. We consider in the experiment a circular road with $n = 5$ cars on it. We are interested whether there could be

a potential traffic accident in the closed system, and whether there could be a potential traffic jam.

– *SWaT system.* Lastly, we tested our method on a simplified real-world system model. The Secure Water Testbed (SWaT) is a raw water purification laboratory located at SUTD [38]. SWaT is a complicated system involving a series of water treatments like ultrafiltration, chemical dosing, dechlorination through an ultraviolet system. We build a hybrid automaton model of SWaT based on the control program in the programmable logic control (PLC) in the system. The modes are defined based on the discrete states of the actuators (e.g., motorized valves and motorized pumps). These actuators are controlled by the PLC. There are in total 23 actuators, which results in many modes. By focusing on the hydraulic process in the system only, we build a hybrid automaton with 2721 modes. We skip the details of the model due to the limited space here. The readers are referred to [1] for details. The property we verify is that the water level in the backwash tank must not be too high or too low (otherwise, the system needs to be shut down), with some extreme initial setting (e.g., the water level in the tank is close to be too low) to analyze the system safety.

Estimating Cost of Symbolic Execution. In order to apply Algorithm 1 with the right strategy, we need to estimate c_s. The underlying question is how efficient a constraint solver can check the satisfiability of a given path condition. We remark that it is a challenging research problem and perhaps deserves a separate research project by itself. There are a dozen of various factors that determines how a solver performs in solving a given constraint, including the number of variables, the number of operators, the number of differential equations, the length of witnesses (if there is any), etc. Even on the same problem, different solvers have different performance due to different search strategies, ways of pruning and reducing state space, etc. [26,31]. All these facts make a precise estimation of efficiency of symbolic solvers extremely difficult.

In this work, following previous work on this topic [4], we estimate c_s as follows. First, we construct a sequence of increasingly more complicated random constraints (composed of constraints on ODE as well as ordinary constraints, which we obtain from examples in dReal). We then measure the time needed to solve them using dReal one-by-one. Based on the results, we observe that the dominant factor is the length of the formula and thus heuristically decide c_s to be a function of the length of constraints. Next, we apply a function fitting method to obtain a function for predicting c_s. The function we obtained is $\exp(1.73l - 1.65) - 1$ where l is the length of the formula, which suggests that the solving time is exponential in the length of the formula. It implies that dReal has problem solving path conditions containing two or more steps, which in turn means that our choice of strategy should be the *local concolic sampling*. We remark that this is unlikely a precise estimation. Nonetheless, as we show below, even a rough estimation would be useful in guiding when and how to do symbolic execution.

Table 1. Experiments results

		Random		Dynamic	Global	Local (our strategy)
Thermodynamic system	result	ct-eg found	pass	pass	pass	ct-eg found
	time(s)	340.4	600	600	600	41.25
	#samples	13K	22K	n/a	134	551
Navigation system	result	ct-eg found		pass	pass	ct-eg found
	time(s)	91.7		600	600	4.33
	#samples	354		n/a	4	5
Traffic system	result	pass		pass	pass	ct-eg found
	time(s)	600		600	600	28.86
	#samples	1240		n/a	2	2
SWaT system	result	ct-eg found		pass	pass	ct-eg found
	time(s)	102.4		600	600	64.6
	#samples	169		n/a	24	68

Experiment Results. Table 1 shows the experiment results. All experiment results are obtained in Ubuntu Linux 14.04 on a machine with an Intel(R) Core(TM) i5-4950, running with one 3.30 GHz CPU core (no parallel optimization), 6M cache and 12 GB RAM. We set a timeout of 10 min for each experiment, i.e., if no counterexample is identified after 10 min, the property has passed the test. Each experiment is repeated for 10 times and we report the average time. All details on the experiments are at [1].

We compare four approaches in order to show the effectiveness of our chosen strategy. The first is the random sampling approach proposed in [17]. The results are shown in column **random**. Note that there are two results for the thermodynamic system. This is because due to the randomness in the approach, the results are not always consistent (e.g., in one experiment, a counterexample is found, whereas none is found in another). The second approach is the concolic testing approach in [36] (i.e., applying random testing once and applying symbolic execution to visit the alternative path in the last branch and so on). The results are shown in column **dynamic**. The last two columns report the result of applying **global** concolic sampling and **local** concolic sampling respectively.

We have the following observations based on the results. First, among the four approaches, local concolic testing is able to spot counterexamples more efficiently in all cases. Compared with random sampling, the number of samples explored by local concolic sampling is significantly smaller. This confirms the result of the theoretical analysis in previous sections. Second, symbolic execution for hybrid systems are clearly constrained by the limited capability of existing hybrid constraint solvers like dReal. For all four cases, both concolic testing and global concolic sampling time out whilst waiting for dReal to solve the first path condition. This is because the path condition (composed of constraints from multiple steps) is complex and dReal takes a lot of time trying to solve it. The only difference is that while concolic testing got stuck after the first sample, global concolic sampling got stuck after it has randomly sampled a few traces and switched to symbolic execution. On the contrary, local concolic sampling uses dReal to solve a one-step path condition each time and is able to smartly switch between random sampling and symbolic execution, and eventually found a counterexample. Third, the experiment results suggest that the formula that we applied for estimating c_s turned out to be an under-approximation, i.e., the actual time cost is often much larger. If we modify the function to return a much larger cost for solving a path composed of two or more steps, global concolic sampling would be equivalent to random sampling as symbolic execution would never be selected due to its high cost.

6 Conclusion and Related Works

In this work, we investigated the effectiveness of different sampling methods (i.e., random sampling and symbolic execution) for hybrid systems. We established theoretical results on comparing their effectiveness and we developed an approach for combining random sampling and symbolic executions in a way which is provably cost-effective.

In the following, we discuss the related work, in addition to those discussed already. This work is inspired by [7], which initialized the discussion on the efficiency of random testing. Our work aims to combine random sampling and symbolic execution to identify rare counterexamples efficiently. It is thus closely related to work on handling rare events in the setting of statistical model checking [5,6,25]. In [5], the authors set up a theoretical framework using coupling theory and developed an efficient sampling method that guarantees a variance reduction and provides a confidence interval. In [6] the authors proposed the first importance sampling method for CTMC to provide a true confidence interval. In [25] the authors motivated the use of importance splitting to estimate the probability of a rare property. Our work is different as we complement sampling with symbolic execution to identify rare events efficiently.

This work borrows idea from work on combining program testing with symbolic execution (a.k.a. dynamic symbolic execution or concolic testing). In [15], the authors proposed a way of combining program testing with symbolic execution to achieve better test coverage. Random testing is first applied to explore program behaviors, after which symbolic execution is used to direct the test towards different program branches. Similar ideas later have been developed in [8,33,36,37]. Our work is different in two ways. One is that we target hybrid systems in work, which has different characteristics from ordinary programs. One of them is that symbolic execution of hybrid automata is considerable more expensive, which motivated us to find ways of justifying the use of symbolic execution. The other is that, based on the probabilistic abstraction of hybrid models, we are able to formally compare the cost of random sampling against symbolic execution to develop cost-effective sampling strategies. We remark that the same idea can be applied to concolic testing of programs as well.

HyChecker is a tool for analyzing hybrid systems and thus it is related to tools/systems on analyzing hybrid systems. In [35], the authors developed a theorem prover for hybrid systems. Users are required to use differential dynamic logic to model hybrid systems. Afterwards, the prover can be used interactively to find a sound and complete proof of certain properties of the system. It has been shown that the prover works for safety critical systems like aircrafts [35]. HyChecker is different as it is fully automatic. *dReach* [13] is a recent tool developed for verifying hybrid systems. It is based on the SMT solver *dReal* [14] developed by the same authors. *dReach* focuses on bounded δ-complete reachability analysis. It provides a relatively easy-to-use interface for modeling hybrid systems and verifies whether a system is δ-safe under given safety demands. We observe since *dReach* attempts to solve every path in a hybrid automaton, its performance suffers when the system becomes more complicated. HyChecker relies on *dReal* and tries to improve *dReach* by combining random sampling to avoid solving many of the paths. HyTech [21] is one of the earliest tools on verifying hybrid systems. It is limited to linear hybrid automata.

Acknowledgement. The project is supported by the NRF project IGDSi1305012 in SUTD and by the National Natural Science Foundation of China under grant no. 61532019, 61202069 and 61272160.

References

1. http://sav.sutd.edu.sg/?page_id=2803
2. http://sav.sutd.edu.sg/SMC/
3. Abramowitz, M.: Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables. Dover Publications, New York (1974). Incorporated
4. Aziz, M.A., Wassal, A.G., Darwish, N.M.: A machine learning technique for hardness estimation of QFBV SMT problems. In: 10th International Workshop on Satisfiability Modulo Theories (SMT), pp. 57–66 (2012)
5. Barbot, B., Haddad, S., Picaronny, C.: Coupling and importance sampling for statistical model checking. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 331–346. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28756-5_23
6. Barbot, B., Haddad, S., Picaronny, C., et al.: Importance sampling for model checking of continuous time markov chains. In: SIMUL, pp. 30–35 (2012)
7. Böhme, M., Paul, S.: On the efficiency of automated testing. In: 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE-22), pp. 632–642 (2014)
8. Cadar, C., Dunbar, D., Engler, D.R.: KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs. In: 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 209–224 (2008)
9. Chistikov, D., Dimitrova, R., Majumdar, R.: Approximate counting in SMT and value estimation for probabilistic programs. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 320–334. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46681-0_26
10. Fehnker, A., Ivančić, F.: Benchmarks for hybrid systems verification. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 326–341. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24743-2_22
11. Filieri, A., Frias, M.F., Păsăreanu, C.S., Visser, W.: Model counting for complex data structures. In: Fischer, B., Geldenhuys, J. (eds.) SPIN 2015. LNCS, vol. 9232, pp. 222–241. Springer, Heidelberg (2015). doi:10.1007/978-3-319-23404-5_15

12. Filieri, A., Pasareanu, C.S., Visser, W., Geldenhuys, J.: Statistical symbolic execution with informed sampling. In: 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE-22), pp. 437–448 (2014)
13. Gao, S., Kong, S., Chen, W., Clarke, E.: Delta-complete analysis for bounded reachability of hybrid systems. arXiv preprint arXiv:1404.7171 (2014)
14. Gao, S., Kong, S., Clarke, E.M.: dReal: an SMT solver for nonlinear theories over the reals. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 208–214. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38574-2_14
15. Godefroid, P., Klarlund, N., Sen, K.: Dart: directed automated random testing. SIGPLAN Not. **40**(6), 213–223 (2005)
16. Gordon, J., Serway, R., McGrew, R.: Physics for Scientists and Engineers, vol. 2. Cengage Learning, Boston (2007)
17. Gyori, B.M., Liu, B., Paul, S., Ramanathan, R., Thiagarajan, P.S.: Approximate probabilistic verification of hybrid systems. In: Abate, A., Šafránek, D. (eds.) HSB 2015. LNCS (LNBI), vol. 9271, pp. 96–116. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26916-0_6
18. Hahn, E.M., Hartmanns, A., Hermanns, H., Katoen, J.: A compositional modelling and analysis framework for stochastic hybrid systems. Formal Methods Syst. Des. **43**(2), 191–232 (2013)
19. Henzinger, T.A.: The theory of hybrid automata. In: 11th Annual IEEE Symposium on Logic in Computer Science (LICS), pp. 278–292 (1996)
20. Henzinger, T.A.: The theory of hybrid automata. In: Inan, M.K., Kurshan, R.P. (eds.) Verification of Digital and Hybrid Systems. NATO ASI Series, vol. 170, pp. 265–292. Springer, Heidelberg (2000)
21. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: HyTech: a model checker for hybrid systems. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 460–463. Springer, Heidelberg (1997). doi:10.1007/3-540-63166-6_48
22. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**(1), 94–124 (1998)
23. Henzinger, T.A., Majumdar, R.: Symbolic model checking for rectangular hybrid systems. In: Graf, S., Schwartzbach, M. (eds.) TACAS 2000. LNCS, vol. 1785, pp. 142–156. Springer, Heidelberg (2000). doi:10.1007/3-540-46419-0_11
24. Iverson, K.E.: A Programming Language. Wiley, New York (1962)
25. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39799-8_38
26. Jha, S., Limaye, R., Seshia, S.A.: Beaver: engineering an efficient SMT solver for bit-vector arithmetic. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 668–674. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02658-4_53
27. Kamide, N.: Bounded linear-time temporal logic: a proof-theoretic investigation. Ann. Pure Appl. Logic **163**(4), 439–466 (2012)
28. Kong, P., Li, Y., Chen, X., Sun, J., Sun, M., Wang, J.: Towards concolic testing for hybrid systems. In: Fitzgerald, J., et al. (eds.) FM 2016, LNCS 9995, pp. X–XY. Springer, Heidelberg (2016)
29. Lebesgue, H.: Intégrale, longueur, aire. Annali di Matematica Pura ed Applicata **7**(1), 231–359 (1902)
30. Leon-Garcia, A.: Probability and Random Processes For EE's, 3rd edn. Prentice-Hall Inc., Upper Saddle River (2007)
31. Lu, F., Iyer, M.K., Parthasarathy, G., Wang, L.-C., Cheng, K.-T., Chen, K.C.: An efficient sequential sat solver with improved search strategies. In: The Conference on Design, Automation and Test in Europe (DATE), 2005, pp. 1102–1107 (2005)

32. Luckow, K.S., Pasareanu, C.S., Dwyer, M.B., Filieri, A., Visser, W.: Exact and approximate probabilistic symbolic execution for nondeterministic programs. In: ACM/IEEE International Conference on Automated Software Engineering (ASE), pp. 575–586 (2014)
33. Majumdar, R., Sen, K.: Hybrid concolic testing. In: 29th International Conference on Software Engineering (ICSE 2007), pp. 416–426. IEEE (2007)
34. Orosz, G., Wilson, R.E., Szalai, R., Stépán, G.: Exciting traffic jams: nonlinear phenomena behind traffic jam formation on highways. Phys. Rev. E. **80**, 046205 (2009)
35. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010). Incorporated
36. Sen, K.: Concolic testing. In: 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 571–572. ACM (2007)
37. Sen, K., Agha, G.: CUTE and jCUTE: concolic unit testing and explicit path model-checking tools. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 419–423. Springer, Heidelberg (2006). doi:10.1007/11817963_38
38. Swat, S.: A test bed for secure water treatment (2015). http://academics.sutd.edu.sg/news-events/event/news/media-release-swat-a-test-bed-for-secure-water-treatment-swat/
39. Veach, E., Guibas, L.J.: Optimally combining sampling techniques for monte carlo rendering. In: 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 419–428 (1995)