State estimation and property-guided exploration for hybrid systems testing

Thao Dang and Noa Shalev *

VERIMAG/CNRS,
Centre Equation
2 avenue de Vignate, 38610 Gières, France
Thao.Dang@imag.fr, Noa.Shalev@imag.fr

Abstract. This paper is concerned with model-based testing of hybrid systems. The first result is an algorithm for test generation which enhances the coverage of critical trajectories by using a random walk. The second result is a framework for practical testing that includes a state estimator. When the state of a system under test cannot be directly observed, it is necessary to reconstruct the trajectory of the real system in order to produce a verdict whether the system violates a property. To do so, we integrate in our tester a hybrid observer, the goal of which is to provide an estimate for the current location and the continuous state of the system under test based on the information on the input and the output of the system.

1 Introduction

We describe some recent progress in model-based testing of hybrid systems, systems combining continuous and discrete dynamics. Such systems have been widely accepted as a mathematical model for many applications in embedded systems and cyber-physical systems. In our previous work [7], we introduced a test coverage measure, based on the notion of star discrepancy of a set of points, which indicates how well the states visited by a test suite represent the whole reachable space. We then designed a test generation algorithm, called gRRT [7] which can be seen as a coverage guided version of the RRT (Rapidly-exploring Random Tree) algorithm for robotic planning [12]. Our algorithm has been successfully applied to a number of analog circuits and control applications. In this paper, we propose a new version of the test generation algorithm that is not only guided by the coverage but also by the property to verify. We focus on covering the trajectories that violate a property of interest and exploiting the discrete structure of the hybrid system. More concretely, we are interested in finding a search strategy that allows achieving a high probability of reaching some set of states. In the current version of the gRRT algorithm, we sample a

* This work was supported by the Agence Nationale de la Recherche (VEDECY project - ANR 2009 SEGI 015 01).

goal hybrid state according to some distribution reflecting the current coverage information. The probability that a state is sampled to be a goal state depends on the current state coverage, which tends to a 'uniform' coverage over the global hybrid state space. In this work, we propose a new sampling method in order to biase the exploration towards some critical paths. To do so, we specify some desired stationary probability distribution over the regions (which reflect the objective of our biased exploration) and use the Metropolis-Hastings algorithms to compute a transition probability matrix of a random walk on a discrete abstraction of the hybrid system under study. An advantage of this random exploration is that one can use the bound on the expected number of transitions necessary to reach a given region as a criterion to determine the desired stationary probability distribution and guide the exploration towards critical behaviors more efficiently.

In addition, we address the partial observability problem to extend our framework to more practical testing settings. When not all discrete transitions and not all components of the continuous states are observable, to produce a verdict it is necessary to reconstruct the trajectory of the system under test. To do so, we integrate in our tester a hybrid observer, the goal of which is to provide an estimate for the current location and the continuous state of the system under test, based on the information on the input and the output of the system.

Hybrid systems testing has recently attracted the attention of researchers, which is attested by a number of publications on the topics (see for example [9,4,17,10] and references therein). However, to our knowledge, partial observability is not yet considered. Our work on testing under partial observability is inspired by a number of existing results in testing for discrete and timed systems (see for example [11,8,1]). In some of these works (such as [8]), game theoretic approaches are used. Applying this idea to hybrid systems is however difficult since this often requires complex and expensive set computations. Our approach is rather based on the idea of estimating the state of the system using the well-established results on observer design for continuous systems [14,5]. Concerning the use of random walks in test generation, this idea has recently been intensively applied for software testing, in particular, one could mention the tool MaTeLo developped by the company All4tec[1]. In this work, we make use of Metropolis-Hastings algorithms which have been successful for network protocol design (see for example [16]).

The paper is organized as follows. We first recall the testing framework and basic definitions. We then show how to combine a random walk with the coverage-guided test generation algorithm. The next section is devoted to the problem of designing an algorithm for test execution under partial observability. Finally we present some experimental results obtained for two well-known analog and mixed-signal circuit benchmarks.

[1] http://www.all4tec.net/

2 Model-based testing of hybrid systems

2.1 Model

We use a conformance testing framework based on the hybrid automaton model [2]. Intuitively, a hybrid automaton is an automaton where each location is associated with a distinct continuous mode and the switching between the continuous modes is described by discrete transitions. In this work we focus on discrete-time hybrid automata.

Definition 1. *A discrete-time hybrid automaton is a tuple* $\mathcal{A} = (\mathcal{X}, Q, E, F, \mathcal{I}, \mathcal{G})$ *where*

- $\mathcal{X}$ *is the continuous state space and is a bounded subset of* $\mathbb{R}^n$*;*
- Q *is a (finite) set of locations;*
- $E \subseteq Q \times Q$ *is a set of discrete transitions;*
- $F = \{F_q \mid q \in Q\}$ *such that for each* $q \in Q$*,* $F_q = (f_q, U_q)$ *defines a difference equation*
$$x[k+1] = f_q(x[k], u[k])$$
for each location q*;* $x \in \mathcal{X}$ *is the continuous state,* $u(\cdot) \in \mathcal{U}_q$ *is the input of the form* $u : \mathcal{N}^+ \to U_q \subset \mathbb{R}^m$*. The set* $\mathcal{U}_q$ *is the set of admissible inputs.*
- $\mathcal{I} = \{\mathcal{I}_q \subseteq \mathcal{X} \mid q \in Q\}$ *is a set of staying conditions;*
- $\mathcal{G} = \{\mathcal{G}_e \mid e \in E\}$ *is a set of guards such that for each discrete transition* $e = (q, q') \in E$*,* $\mathcal{G}_e \subseteq \mathcal{I}_q$*;*

A *hybrid state* is a pair (q, x) where $q \in Q$ and $x \in \mathcal{X}$. The initial state of the automaton is denoted by (q_0, x_0). A state (q, x) of $\mathcal{A}$ can change in two ways as follows: by a *continuous evolution* (that is, the continuous state x evolves according to the dynamics f_q while the location q remains constant) and by a *discrete evolution* (that is, x satisfies the guard condition of an outgoing transition, the system changes the location by taking this transition).

Unlike *continuous evolutions*, *discrete evolutions* are instantaneous, which means that they do not take time. This model allows capturing *non-determinism* in both continuous and discrete dynamics. This non-determinism is useful for describing disturbances from the environment and imprecision in modelling and implementation.

2.2 Testing problem

Specification and System Under Test. Our testing goal is to study the conformance relation between the behaviors of a system under test (SUT) and a specification. The specification is modeled by a hybrid automaton $\mathcal{A}$ and the system under test by another hybrid automaton $\mathcal{A}_s$ (we may not know the hybrid automaton $\mathcal{A}_s$). The tester applies the control inputs to the SUT and

observes the outputs to produce one of the following verdicts: 'pass' (the observed behavior is allowed by the specification), 'fail' (the observed behavior is not allowed by the specification). In this work, we use the trace inclusion to define conformance relation. Intuitively, the system under test $\mathcal{A}_s$ conforms to the specification $\mathcal{A}$ if under every admissible control action sequence, the set of observation sequences of $\mathcal{A}_s$ is included in that of $\mathcal{A}$ (see [7] for more detail).

Inputs. An input of the system which is controllable by the tester is called a *control input*; otherwise, it is called a *disturbance input*. All the continuous inputs are assumed to be controllable by the tester. The discrete transitions could be controllable or uncontrollable.

We use the following assumption about the inputs: disturbance actions are of higher priority than control actions. This means that when a control action and a disturbance action are simultaneously enabled, the disturbance action takes place first.

Observations. In our previous work, the locations and the continuous states of the hybrid automata $\mathcal{A}$ and $\mathcal{A}_s$ are observable. In this work, we use a less restrictive assumption: the location and the continuous state are not directly observable, and the tester needs to deduce this information from some continuous observations. We define an observation function as follows: $h : \mathcal{X} \to \mathbb{R}^d$. In the following we consider only scalar observation functions, that is $d = 1$. Intuitively, the tester cannot directly observe the continuous state, and the outputs of its sensors can be modeled by: $y[k] = h(x[k])$.

Test cases and test executions. In our framework, a *test case* is represented by a tree where each node is associated with a hybrid state and each edge of the tree is associated with a control action. A physical test execution can be described as follows: the tester applies a control input sequence to the system and measures the observations. This procedure leads to a set of observation sequences since multiple runs are possible due to non-determinism. In practical systems, due to actuator and sensor imprecision, control inputs and observations are subject to errors. The issues related to error in measurements and actuators are treated in [7] and we do not discuss them here.

Coverage-guided test generation We proposed in [7] a notion of *state coverage* that describes how 'well' the visited states represent the reachable set. This measure is defined using the *star discrepancy* notion in statistics, which characterises the uniformity of the distribution of a point set within a region. Note that the reachable sets of hybrid systems are often non-convex with complex geometric form, therefore considering only corner cases does not always cover the behaviors that are important for reachabilily properties, especially in high

dimensions. Our current method for test generation is based on a randomized exploration of the reachable state space of the system. It is an extension of the Rapidly-exploring Random Tree (RRT) algorithm, a successful motion planning technique for finding robotic trajectories in an environment with obstacles [12]. Furthermore, we combine it with a guiding tool in order to achieve a good coverage of the system's behaviors we want to test. The new results presented in this paper are twofold. First, we provide a new test generation algorithm guided by the coverage measure and additionally by the property to test via a random walk. Furthermore, we address a testing problem under partial observability where the hybrid state should be deduced from a sequence of observations.

3 Combining the coverage-guided exploration and random walks

The main steps of the coverage-guided test generation algorithm [7] are the following: (1) a goal state is sampled, and this sampling is guided so that the goal state lies in the regions where the local coverage of the visited states is still low; (2) a neighbor state of the goal state is determined, from which an appropriate control input is applied to steer the system towards the goal state.

When using the state coverage measure to guide the test generation process, the whole reachable space is 'equally' important for the exploration and the test generation tries not to leave a large part of the reachable space unvisited. The resulting test suite is appropriate when different qualitative behaviors of the system need to be explored. However, when some regions in the state space or some traces are of particular interest, we want to bias the execution towards those regions and traces. To this end, during the test generation we combine the coverage-guided sampling with a guiding method based on a random walk.

To define a random walk, we first partition the continuous state space into a set of regions and from there we construct a directed graph G which roughly overapproximates the specification automaton $\mathcal{A}$. Then, the sampling process consists of two steps:

1. Perform a random walk on the resulting graph G to determine a goal region.
2. Within the goal region we use the coverage-guided test generation algorithm [7] to sample the goal state.

In the following we explain the first step of the sampling process. Let $G = (V_G, E_G)$ be the underlying graph obtained by partioning the continuous state space; V_G is the set of nodes and E_G is the set of directed edges between the nodes. The partition must capture the discrete transitions of $\mathcal{A}$ and separate critical regions from the rest. In this work, we use axis-aligned hyperplanes to define a partition. As future work, more general polyhedral partitions will be considered and this requires adapting the coverage definition which is currently based on a box partition of the continuous state space.

Given a node $w \in V_G$, an adjacent node of w is a node $v \in V_G$ such that there exists an edge in E_G that connects w to v. Let $A_G(w)$ be the set of all the adjacent nodes of w. A random walk[2] on G specifies a run where the node to be visited next is selected from the adjacent vertices at random with a transition probability $P(G) = (p_{wv})_{w,v \in V_G} \in [0,1]^{V_G \times V_G}$ such that for all $w \in V_G$ $\sum_{v \in A_G(w)} p_{wv} = 1$ and for any $v \in V_G$ $p_{wv} = 0$ if $v \notin A_G(w)$.

A random walk on G starting at a vertex $w \in V_G$ under the transition matrix $P(G)$ is an infinite sequence of random variables $\eta_i \in V_G$ such that $\eta_0 = w$ with the probability 1, and for all $i \geq 0$ the probability that $\eta_{i+1} = w'$, provided that the probability that $\eta_i = v$ is $p(v, w')$. The hitting time from w to v under the transition matrix $P(G)$ is defined as $H_P(w, v) = \mathcal{E}[\inf i \mid \eta_i = v]$, which is the expectation of the smallest numbers of steps needed to reach v from w.

In order to bias the exploration towards the critical regions, we define a target probability distribution

$$\pi = \{\pi_v \mid v \in V_G\}.$$

The regions we want to explore are given a higher target probability. To achieve this target probability distribution, we use the Metropolis-Hastings method since it guarantees that the stationary distribution of such a random walk on the graph G is the target distribution π [16]. Given two nodes w and v, we assign a probability to the edge from w to v:

$$\begin{cases} p_{wv} = \dfrac{1}{deg(w)} min\{\dfrac{deg(w)\pi_v}{deg(v)\pi_w}, 1\} & \text{if } v \text{ is adjacent node of } w \\ p_{wv} = 1 - \displaystyle\sum_{w' \neq w} p_{ww'} & \text{if } v = w \\ p_{wv} = 0 & \text{otherwise} \end{cases}$$

where $deg(w)$ is the degree of the vertex w. The additional reason we choose to use the Metropolis-Hastings method in this work is that it has good hitting times, which are of $\mathcal{O}(rN_v^2)$ where N_v is the number of vertices and $r = \max\{\dfrac{\pi_w}{\pi_v} \mid w, v \in V_G\}$. As mentioned earlier, this algorithm has been successfully applied to many applications, in particulat in network protocols.

4 Hybrid state estimation

We now proceed with our second result. As mentioned earlier, the tester needs to deduce the current location and the current continuous state from the continuous observations.

[2] For a detailed introduction to random walks on a graph, the reader is referred to [15].

4.1 Continuous state estimation

We first describe a method for estimating the state of a continuous system, which is used in the next section to handle hybrid systems. Thus, for simplicity of presentation, we drop the location index q from the equations of the dynamics, that is

$$\begin{aligned} x[k+1] &= f(x[k], u[k]) \\ y[k] &= h(x[k]) \end{aligned}$$

This method is based on the Newton observer method for non-linear systems [14]. During the test execution, to estimate $x[k]$ at each time point k, the tester needs a sufficient long sequence of observations. This is indeed related to observability of the system f and is explained in the following.

Let $U_{k,k+N-1}$ be a vector of N consecutive inputs that is to be applied to the system at time k:

$$U_{k,k+N-1} = \begin{pmatrix} u[k] \\ u[k+1] \\ \dots \\ u[k+N-1] \end{pmatrix}$$

Under this continuous input sequence $U_{k,k+N-1}$ and starting from the state $x[k]$ (that we need to estimate), the system under test produces a vector of observations

$$Y_{k,k+N-1} = \begin{pmatrix} y[k] \\ y[k+1] \\ \dots \\ y[k+N-1] \end{pmatrix}$$

We define the following vector of functions:

$$H(x, U_{0,N-1}) = \begin{pmatrix} h(x) \\ h \circ f^{u[0]}(x) \\ \dots \\ h \circ f^{u[N-1]}(x) \circ \ldots \circ f^{u[0]}(x) \end{pmatrix}$$

where $\circ$ denotes the following composition operator: given two functions $\alpha : X \to Y$ and $\beta : Y \to Z$, the function resulting from composing α with β is $\beta \circ \alpha : X \to Z$ such that for a given $y \in Y$, $\beta \circ \alpha(y) = \beta(\alpha(x))$. In the above since the inputs are fixed in the functions f, we write them as superscripts of f.

From the results on observativility of continuous systems, we know that if the system is N-observable (with $N \geq 1$) at state $\tilde{x}$ if there any sequence U of N control inputs such that $\tilde{x}$ is the unique solution of the following equation:

$$H(\tilde{x}, U) = H(\xi, U)$$

In the above, ξ is the unknown variable. Here N is the minimum number of observations required to reconstruct the state.

We assume now that the system is N-observable [14], and thus to estimate the state at time $k \geq 0$ it suffices to solve the following equation:

$$Y_{k,k+N-1} - H(\xi, U_{k,k+N-1}) = 0 \tag{1}$$

Intuitively, to estimate the state at time point k, we need to apply a sequence $U_{k,k+N-1}$ of N next input values to obtain a sequence $Y_{k,k+N-1}$ of observations. Then, we solve the above equation with $\xi \in \mathbb{R}^n$ as the unknown variables. We let $N = n$, so that the Jacobian matrix of the vector H of functions is square. Then, to determine ξ, we can use Newton's algorithm as follows[3]:

$$\xi^{i+1} = \xi^i + [\frac{\partial H}{\partial x}(\xi^i, U_{k,k+N-1})]^{-1}(Y_{k,k+N-1} - H(\xi^i, U_{k,k+N-1})$$

A detailed discussion on the standard convergence theorem for this algorithm can be found in [13]. The convergence of Newton's algorithm depends on the initial estimate and the second derivatives $||\frac{\partial^2 H}{\partial x^2}||$ which measures the nonlinearity degree of the equation (1). For linear systems, the initial estimate can be arbitrarily far from the exact solution; however, when $||\frac{\partial^2 H}{\partial x^2}||$ is large (that is, the system is very nonlinear), the initial estimate needs to be more accurate.

4.2 Testing execution with hybrid state estimation

Let T be the tree generated from the specification $\mathcal{A}$, starting from the initial continuous state x_0. Now we use this tree to test the system against the specification $\mathcal{A}_s$. From the above discussion on observablity of continuous systems, to extend to the hybrid systems we need to assume that the discrete transitions can occur at times of multiples of N steps. The test execution procedure described in Algorithm 1 uses the following assumption. At any time step of the algorithm, the tester can apply many input sequences and observe the corresponding observation sequences (which may require restarting the execution of the system from the initial state to restore the current state). Initially, the system could be in any location; thus S_{init} covers all the locations.

At each iteration i (which corresponds to a time segment of length N), the tester keeps a set S_i of possible states visited before time iN. Since the discrete transitions are instantaneous, we also need to include all possible discrete successors (represented by the operator $Succ_d$).

For each of possible states (q, x) in S_i, the tester chooses from the tree a possibly feasible sequence of N inputs. It is important to note that, according to our assumption during the next N steps no discrete transition can occur. The tester then applies the chosen sequence to the system under test and observes the corresponding sequence Y of outputs. Using Newton's algorithm, we compute an estimator ξ of the state at time iN. There are two cases:

[3] In the case that there are more equations than states, the inverse should be replaced by a pseudo-inverse [13].

Algorithm 1 Test execution

```
/* Input: Test tree T */

i = 0
S_init = {(q, x_0) | q ∈ Q}
S_n = S_init ∪ Succ_d(S_init)
repeat
   S_i = S_n ∪ Succ_d(S_n)
   S_n = ∅
   for all (q, x) ∈ S_i do
      U = InputSeq(T, (q, x))        /* Choose an input sequence U of length N and
      feasible at (q, x) */
      Y = Observation(U) /* Apply the input sequence U to the system and observe
      the corresponding outputs */
      ξ = Newton(q, U, Y)  /* Using Newton's algorithm to estimate the state at the
      current time iN */

      if (||ξ − x|| ≤ ε) then
         S_n = S_n ∪ {Succ_c((q, x), U)} /* Adding all the continuous successors to the
         set S_n, they will be explored in the next iteration */
      end if
   end for
   if (S_n = ∅) then
      RETURN 'fail' verdict
   end if
   i++
until i = i_max ∨ S_n = ∅
```

1. If the estimator ξ is ε-close to the corresponding state x in the tree, we add it to the set S_n of new possible states which will be treated in the next iteration. The threshold ε is used to account for numerical error in Newton's algorithm and possible measurement error.
2. Otherwise, we continue with another possible current state in S_i.

After applying the above treatment to all the possible states in S_i, if the set S_n of new possible states is empty, the algorithm declares the verdict 'fail'; otherwise it continues. Note that to initialize the estimates in Newton's algorithm we use the last estimates of the previous iteration, in order to obtain a good convergence of the estimation algorithm.

Theorem 1. *If the algorithm returns the 'fail' verdict, the system under test does not conform to the specification.*

Proof. We first prove the following proposition by induction. Let us suppose that the set S_{i+1} at the beginning of the loop 'REPEAT... UNTIL' contains all the visited state during the interval $[iN, (i+1)N]$ (this is true for $i = 0$), we prove that the set S_{i+2} contains all the visited states up to time $(i+2)N$.

Indeed, for any state $(q, x) \in S_{i+1}$, all the states which can be reached from (q, x) by a discrete transition are already included in S_{i+1}. Hence, in order to include all the state reachable in N next step from S_{i+1}, it suffices to consider only the continuous dynamics. Since by applying Newton's algorithm to each state $(q, x) \in S_{i+1}$ as above we can estimate the continuous states, if the estimates do not match the expected states stored in the tree T, the successors of (q, x) are not the states visited so far. It then follows that in the iteration $i + 2$, the algorithm discards only the states which cannot be visited up to time $(i+2)N$, and at the same time the algorithm includes all the states which are possibly visited up to time $(i+2)N$. Thus, the proposition is proved.

From this proposition, it is easy to see that when the algorithm returns 'fail', that is the set of possible states is empty, the system under test is not conform to the specification. ■

5 Experimental results

We implemented the above algorithms and incorporated their implementations in the tool HTG [6]. The enhancement of the test generation algorithm allowed us to increase the efficiency of the tool. In addition, the tool can now be used for test execution in practical settings with partial observability. In this section, to show the improvement in the test generation, we present the results obtained for a well-known benchmark of ring oscillator circuit. The second case study is a Delta-Sigma circuit, which is used to illustrate the state estimation feature.

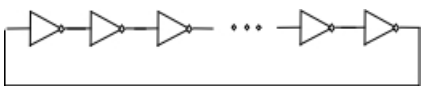

Fig. 1. Ring oscillator circuit.

5.1 Ring oscillator

The ring oscillator circuit is described in Figure 1, given in SPICE netlist formalism[4]. This circuit has one input variable, which is the source voltage of the circuit. Its values is between $1.6V$ and $2.4V$. There are 9 state variables, which are the output voltages of each inverter. We want to test whether the output voltages could reach a value lower that $-2.15V$.

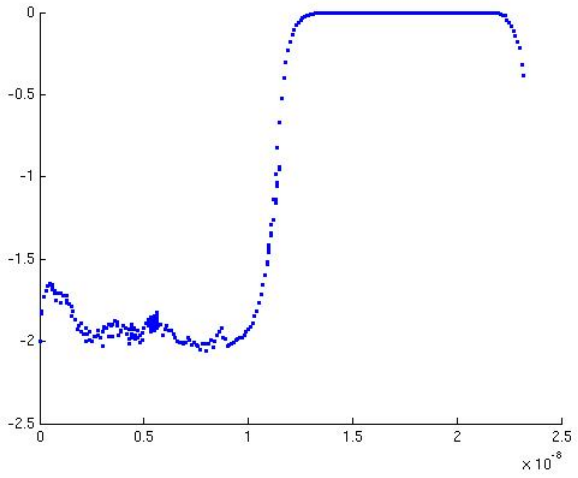

Fig. 2. The output voltage of the last inverter, obtained without using random walk.

In the result of the test generation without using random walk (shown in Figure 2), the maximum value of the output voltage of the last inverter is 0, and its minimum value is $-2.1458V$. The computation time for generating 10000 points is $60s$.

In the next experiment, we partitioned the continuous state space into two regions corresponding to two 'binary' states of the outputs of the inverters:

$$v_1 = (low, high, low, high, low, high, low, high, low)$$

[4] The tool HTG can accept SPICE netlists as input. For more detail on the tool, see [6].

$$v_2 = (high, low, high, low, high, low, high, low, high)$$

We used a random walk with the target probability distribution for these two regions: $\pi_1 = 0.8$ and $\pi_2 = 0.2$. The Metropolis-Hastings algorithm produced the following transition probability matrix:

from $\rightarrow$ to	0	1
0	0.875	0.125
1	0.5	0.5

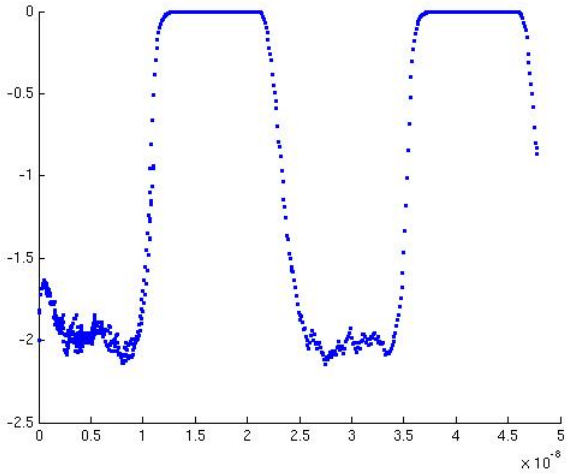

Fig. 3. The output voltage of the last inverter, obtained with a random walk.

In the result of the test generation using random walk (shown in Figure 3), the maximum value of the output voltage of the last inverter is 0, and its minimum value is $-2.1725V$. The computation time for generating 10000 points is $55s$. We can see that by favoring the region corresponding the low level of the voltage, a lower value of the voltage was discovered, and a violation of the property was detected.

5.2 Delta-Sigma circuit

The second case study is a third-order Delta-Sigma modulator [3], which is a mixed-signal circuit shown in Figure 4. When the input is positive and its value is less than 1, the output takes the $+1$ value more often and the quantization error is fed back with negative gain and accumulated in the integrator $\frac{1}{z-1}$. Then,

when the accumulated error reaches a certain threshold, the quantizer switches the value of the output to -1 to reduce the mean of the quantization error.

The specification of a third-order Delta-Sigma modulator is modeled as a hybrid automaton, shown in Figure 4. The discrete-time dynamics of the system is as follows: $x[k+1] = Ax[k] + bu[k] - sign(y[k])a$, $y([k] = c_3x_3[k] + b_4u[k]$ where $x[k] \in \mathbb{R}^3$ is the integrator states, $u[k] \in \mathbb{R}$ is the input, $y[k] \in \mathbb{R}$ is the input of the quantizer. Thus, its output is $v[k] = sign(y[k])$, and one can see that whenever v remains constant, the system's dynamics is affine continuous.

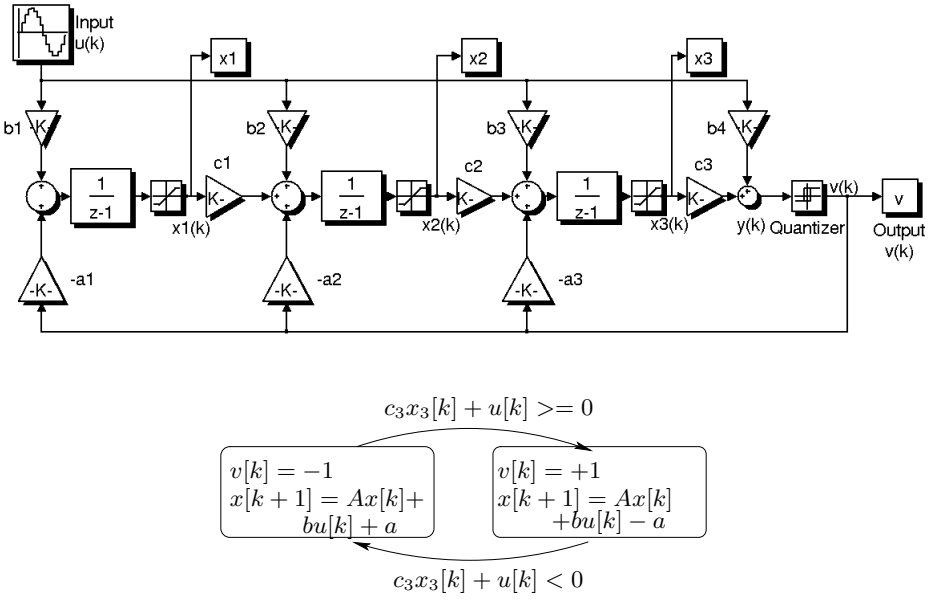

Fig. 4. Model of a third-order modulator: Saturation blocks model saturation of the integrators.

In this study we first generated the test tree for the above hybrid automaton used as the specification automaton (see Figures 5 and 6). Our system under test is an implementation of the Delta-Sigma in SPICE netlists. The observation function $h(x) = 0.1x_1 + 0.2x_2 + 0.5x_3$. The initial state is in $[-0.01, 0.01]^3$ and the input values $u \in [-0.5, 0.5]$.

Figure 7 shows the result of the test execution using the hybrid state estimation, which state that for a bounded time the implementation is conform to the specification automaton. In this figure, the horizontal axis is time. The points drawn with $*$ sign are the estimates of x_2 obtained from the observations on the implementation. The circle points correspond to the possible states in the first location and the $+$ points correspond to the possible states in the second location.

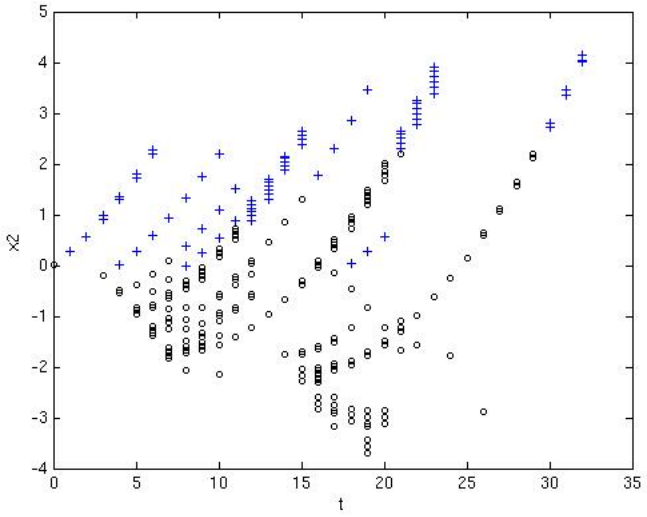

Fig. 5. Test generation result for the first location. The points drawn with the + sign correspond to the states from which a discrete transition to the second location takes place.

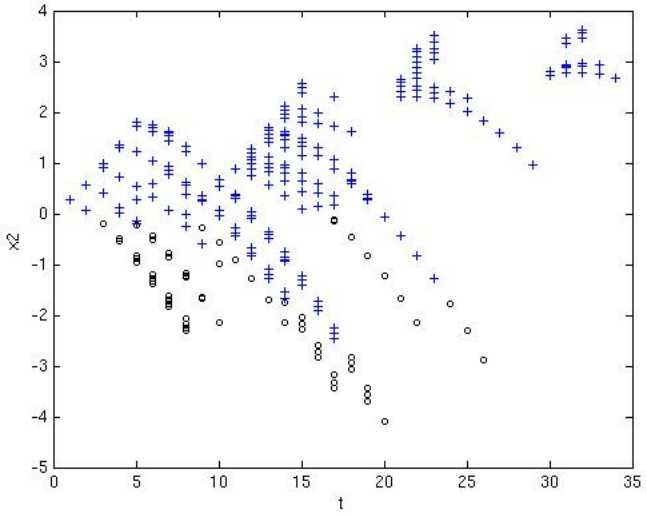

Fig. 6. Test generation result for the second location. The points drawn with the circle sign correspond to the states from which a discrete transition to the first location takes place.

With $\varepsilon = 1e - 2$, no violation of the conformance between the implementation and the specification automaton was detected.

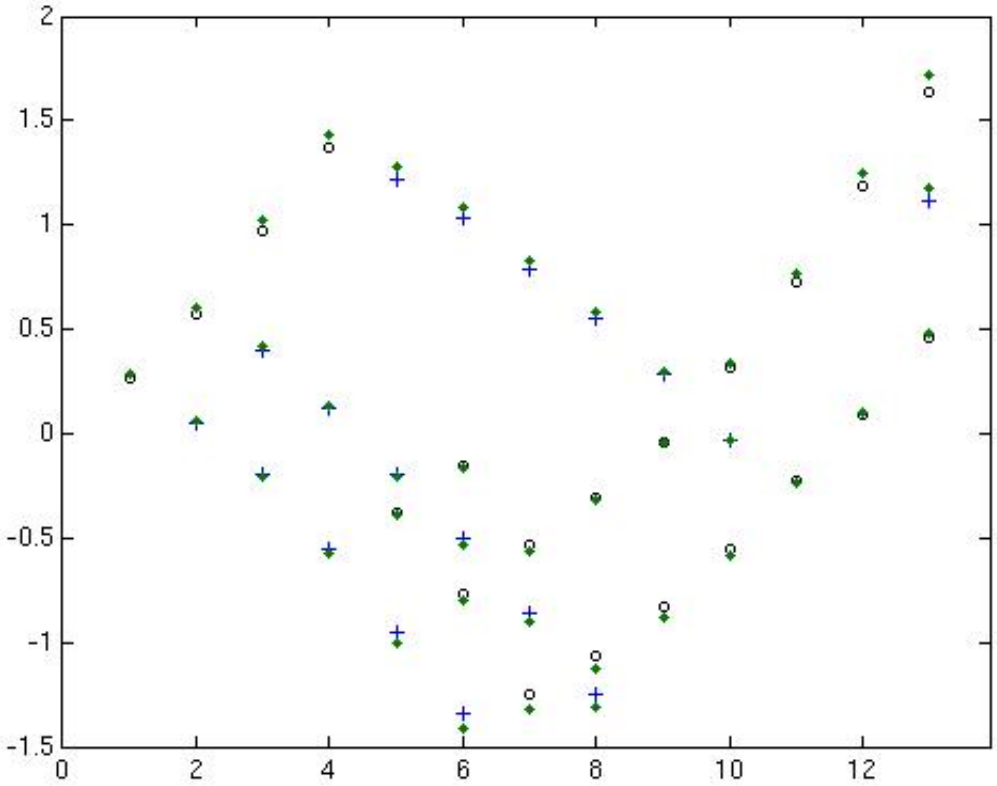

Fig. 7. Test execution result (over time) for the Delta-Sigma circuit.

6 Conclusion

In this work, we contributed two new results for hybrid systems testing: one is a method for test generation guided by the properties to test, and the other is a procedure for test execution with partial observability. The results were implemented and successfully applied to two case studies in circuit validation. One direction for future work is the development of a coverage measure which can capture interesting qualitative behaviors. We also plan to use this procedure

for test execution to tackle the problem of checking equivalence between different models with different abstraction levels and to model identification.

References

1. Bernhard K. Aichernig, Harald Brandl, and Franz Wotawa. Conformance testing of hybrid systems with qualitative reasoning models. *Electron. Notes Theor. Comput. Sci.*, 253(2):53–69, October 2009.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
3. P. M. Aziz, H. V. Sorensen, and J. van der Spiegel. An overview of sigma-delta converters. *Signal Processing Magazine, IEEE*, 13(1):61–84, 1996.
4. A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *HSCC*, pages 142–156, 2004.
5. E. Biyik and M. Arcak. Hybrid newton observer design using the inexact newton method and gmres. In *Proc. 2006 American Control Conf. ACC'06*, pages 3334–3339, 2006.
6. Thao Dang. Model-based testing of hybrid systems. In *Model-Based Testing for Embedded Systems*. CRC Press, 2010.
7. Thao Dang and Tarik Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.
8. Alexandre David, Kim G. Larsen, Shuhao Li, and Brian Nielsen. Timed testing under partial observability. In *2nd IEEE International Conference on Software Testing ICST'09*. IEEE Computer Society Press, 2009.
9. J. Esposito, J. W. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.
10. A. Agung Julius, Georgios E. Fainekos, Madhukar Anand, Insup Lee, and George J. Pappas. Robust test generation and coverage for hybrid systems. In *HSCC*, pages 329–342, 2007.
11. Moez Krichen and Stavros Tripakis. Conformance testing for real-time systems. *Form. Methods Syst. Des.*, 34(3), 2009.
12. S. LaValle and J. Kuffner. Rapidly-exploring random trees: Progress and prospects, 2000. In Workshop on the Algorithmic Foundations of Robotics.
13. D.G. Luenberger. *Optimization by Vector Space Methods*. New York Wiley, 1969.
14. P. Moraal and J.W. Grizzle. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 40(3), 1995.
15. R. Motowani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.
16. Yoshiaki Nonaka, Hirotaka Ono, Kunihiko Sadakane, and Masafumi Yamashita. The hitting and cover times of metropolis walks. *Theor. Comput. Sci.*, 411(16-18):1889–1894, 2010.
17. E. Plaku, L. Kavraki, and M. Vardi. Hybrid systems: From verification to falsification. In W. Damm and H. Hermanns, editors, *International Conference on Computer Aided Verification (CAV)*, volume 4590, pages 468–481. Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Berlin, Germany, 2007.