A Formal Framework of Hybrid Test Cases Generation Applied to Embedded Systems

Tarik Nahhal
Hassan II University Casablanca
Faculty of Sciences, LIAD Laboratory BP 5366 Maarif,
Casablanca Morocco

Abstract—In this paper, we describe a formal framework for conformance testing of continuous and hybrid systems, using the international standard 'Formal Methods in Conformance Testing' FMCT. We propose a novel test coverage measure for these systems, which is defined using the star discrepancy notion. This coverage measure is used to quantify the validation 'completeness'. It is also used to guide input stimulus generation by identifying the portions of the system behaviors that are not adequately examined. We then propose a test generation method, which is based on a robotic motion planning algorithm and is guided by the coverage measure. The approach is illustrated by its application to some embedded system benchmarks.

1 Introduction

Hybrid systems have been recognized as a high-level model appropriate for embedded systems, since this model can describe, within a unified framework, the logical part and the continuous part of an embedded system. Due to the gap between the capacity of exhaustive formal verification methods and the complexity of embedded systems, testing is still the most commonly-used validation method in industry. Its success is probably due to the fact that testing suffers less from the 'state explosion' problem. Indeed, the engineer can choose the 'degree of validation' by the number of tests. In addition, this approach can be applied to the real system itself and not only to its model. Generally, testing of a reactive system is carried out by controlling the inputs and checking whether its behavior is as expected. Since it is impossible to enumerate all the admissible external inputs to the hybrid system in question, much effort has been invested in defining and implementing notions of coverage that guarantee, to some extent, that the finite set of input stimuli against which the system is tested is sufficient for validating correctness. For discrete systems, specified using programming languages or hardware design languages, some syntactic coverage measures can be defined, like exercising every statement or transition, etc. In this work, we treat continuous and hybrid systems that operate in a metric space (typically R^n) and where there is not much inspiration coming from the syntax to the coverage issue. On the other hand, the metric nature of the state space encourages more semantic notions of coverage, namely that all system trajectories generated by the input test patterns form a kind of dense network in the reachable state space without too many big unexplored 'holes'. In this work we adopt a model-based testing approach [1]. This approach allows the engineer to perform validation during the design, where detecting and correcting errors on a model are less expensive than on an implementation. The main contributions of the paper can be summarized as follows. We define a formal framework for conformance testing of continuous and hybrid systems, using the international standard for formal conformance testing FMCT [24]. We propose a test coverage measure for these systems, which is defined using the star discrepancy notion from statistics. This coverage measure is used to quantify the validation 'completeness'. It is also used to guide input stimulus generation by identifying the portions of the system behaviors that are not adequately examined. We propose an algorithm for generating tests from hybrid automata. This algorithm is based on the RRT (Rapidly-exploring Random Tree) algorithm [18] from robotic motion planning and guided by the coverage measure. The rest of the paper is organized as follows. We first describe our conformance testing framework and our test coverage measure.We then present the test generation algorithm and show how to use the coverage measure to guide the test generation process. We also prove the completeness property of the algorithm. Finally, we describe some experimental results. Before concluding, we discuss related work.

2 Testing Problem

As a model for hybrid systems, we use hybrid automata [2]. In most classic versions of hybrid automata, continuous dynamics are defined using ordinary differential equations (ODEs). The behavior of continuous and hybrid systems are however described using differential

1. This work was supported by the CNRST 'Centre National pour la Recherche Scientifique et Technique' (DEV-RS 14).

algebraic equations (DAEs). We thus adapt the model to capture this particularity. Mathematically, the behavior of a (non-linear) continuous system can be described by a set of DAEs: $f(x(t), \dot{x}(t), u(t), w) = 0$ where $x \in \mathbb{R}^n$ denotes the state variables (internal voltages, currents, and outputs), $\dot{x}$ denotes their time derivatives, $w \in W \subset \mathbb{R}^m$ is the system parameter vector, and $u : \mathbb{R}^+ \to U$ is the input signal. When considering embedded systems that exhibit both logical and continuous, their behavior can be naturally modeled using hybrid automata. Before presenting the model, we remark that DAEs and ODEs are different in both theoretical and numerical properties.

2.1 Hybrid Automata

A hybrid automaton is an automaton augmented with continuous variables.

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

- *$\mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state space. We denote by $V(\mathcal{A})$ the set of continuous variables of $\mathcal{A}$.*
- *Q is a finite set of locations (or discrete states).*
- *E is a set of discrete transitions.*
- *$F = \{F_q \mid q \in Q\}$ such that for each $q \in Q$, $F_q = (f_q, U_q, W_q)$ defines a DAE $f_q(x(t), \dot{x}(t), u(t), w) = 0$ where $w \in W_q \subset \mathbb{R}^m$ is the parameter vector, and $u : \mathbb{R}^+ \to U_q \subset \mathbb{R}^p$ is the input signal. We assume the existence and uniqueness of solutions of these differential algebraic equations. Note that during the evolution of the system, the parameter w is constant.*
- *$\mathcal{I} = \{\mathcal{I}_q \subseteq \mathbb{R}^n \mid q \in Q\}$ 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 $e \in E$, $\mathcal{G}_e \subseteq \mathcal{I}_q$.*
- *$\mathcal{R} = \{\mathcal{R}_e \mid e \in E\}$ is a set of reset maps. For each $e = (q, q') \in E$, $\mathcal{R}_e : \mathcal{G}_q \to 2^{\mathcal{I}_{q'}}$ defines how x may change when $\mathcal{A}$ switches from q to q'.*

A *hybrid state* is a pair (q, x) where $q \in Q$ and $x \in \mathcal{X}$ and the hybrid state space is $\mathcal{S} = Q \times \mathcal{X}$. A state (q, x) can change in two ways: by *continuous evolution* and by *discrete evolution*. In location q the continuous evolution of x is governed by the DAE $f_q(x(t), \dot{x}(t), u(t), p) = 0$. Let $\phi(t, x, u(\cdot), p)$ be the solution of the DAE with the initial condition x, the parameter p and under the input $u(\cdot)$. A *continuous transition* $(q, x) \overset{u(\cdot),h}{\rightarrow} (q, x')$ where $h > 0$ is a positive real number means that $x' = \phi(h, x, u(\cdot), p)$ and for all $t \in [0, h] : \phi(t, x, u(\cdot), p) \in \mathcal{I}_q$. In other words, x' is reached from x' under the input $u(\cdot)$, and we say that $u(\cdot)$ is *admissible* starting at (q, x) for h time. For a state (q, x), if there exists a transition $e = (q, q') \in E$ and $x \in \mathcal{G}_e$, then the transition e is enabled, the system can switch from location q to q' and the continuous variables can be assigned to a new value $x' \in \mathcal{R}_e(x)$. This is denoted by $(q, x) \overset{e}{\rightarrow} (q', x')$, and we say that the discrete transition e is admissible at (q, x). We use the notation $(q, x) \to (q', x')$ to simply indicate that (q', x') is reachable from (q, x). We assume that discrete transitions are instantaneous. The hybrid automata we consider are assumed to be *non-Zeno*.

For simplicity of presentation, we first assume a single initial state of the automaton denoted by (q_{init}, x_{init}), and additionally all the paremeter set P_q are singletons. An extension of the framework to a set of initial states and sets of parameters will be discussed when we present our test generation algorithm.

Note that this model is non-deterministic. Indeed, in continuous dynamics the non-determinism is represented by the set of admissible input functions. The discrete transitions are non-deterministic because there might be continuous states at which the system can either continue with the same continuous dynamics or make a transition. Also, multiple transitions can be enabled at the same continuous states, and additionally, the reset maps could also be set-valued. This non-determinism is useful for describing disturbances from the environment and imprecision in modeling and implementation. To define our testing framework, we need the notions of inputs and observations.

Inputs

An input of the system which is controllable (by the tester) is called *control input*; otherwise, it is called *disturbance input*. We consider the following inputs:

- **Continuous inputs.** All the continuous inputs of the system are controllable. Since we want to implement the tester as a computer program, we are interested in piecewise-constant continuous input functions. Hence, a *continuous control action*, such as $(\bar{u}_q, h)$ where $\bar{u}_q$ is the value of the input and h is the *time step*, specifies that the system continues with the dynamics F_q under the input $u(t) = \bar{u}_q$ for exactly h time. We say that $(\bar{u}_q, h)$ is admissible at (q, x) if $u(t) = \bar{u}_q$ is admissible starting at (q, x) for h time.
- **Discrete inputs.** The discrete transitions are partitioned in controllable and uncontrollable discrete transitions. Those are controllable correspond to discrete control inputs, and the others to discrete disturbance inputs. The tester emits a discrete control action to specify whether the system should take a controllable transition (among the enabled ones) or continue with the same continuous dynamics. In the latter case, it can also control the values assigned to the continuous variables by the associated reset map. We denote a discrete control action of this type by the corresponding transition, such as (q, q').

In the remainder of the paper, for simplicity of explanation, we use the following assumption: a continuous control action is of higher priority than any discrete input actions. This means that after a continuous control action $(\bar{u}_q, h)$ is applied, no discrete transition can occur during h time, i.e. until the end of that continuous control action. This assumption is not restrictive, from a modeling point of view. As we shall see later, by considering all the possible values of h we can capture the cases where a discrete transition can occur before the termination of a continuous control action.

Definition 2 (Admissible Input Sequence)**.** *For a state (q,x), a sequence of input actions $\iota_0, \iota_1, \iota_2, \ldots, \iota_k$ is admissible starting at (q,x) if the following conditions are satisfied:*

- *The first input action ι_0 is admissible at (q,x).*
- *For each $i = 1, 2, 3, \ldots, k$:*
  - *ι_i is admissible at (q_i, x_i) where (q_i, x_i) is the state such that $(q_{i-1}, x_{i-1}) \overset{\iota_{i-1}}{\to} (q_i, x_i)$.*

Intuitively, this means that an admissible control sequence does not cause the automaton $\mathcal{A}$ to be blocked.

Definition 3 (Trace)**.** *Given an admissible input sequence $\gamma = \iota_0, \iota_1, \ldots, \iota_k$, the* trace *starting at (q,x) under γ is defined as follows:*

$$\tau((q,x),\gamma) = \{(q_0,x_0),\ldots,(q_k,x_k) \mid (q_0,x_0) = (q,x) \\ \wedge \forall i \in \{1,2,\ldots,k\} : (q_{i-1},x_{i-1}) \overset{\iota_{i-1}}{\to} (q_i,x_i) \}. \quad (1)$$

We use the notation $(q,x) \overset{\gamma}{\to} (q',x')$ to indicate that (q',x') is reachable from (q,x) after γ.

Admissible Control Sequences

It follows from the above assumption that uncontrollable discrete transitions cannot occur during a continuous control action. However, they can occur between control actions. Hence, the result of applying a control action is non-deterministic. Given a state (q,x) and a control action c, let $\sigma = \sigma_0, \sigma_1, \ldots \sigma_k$ be an input sequence that satisfies the following:

- $\sigma_0 = c$ and all the other σ_i with $i > 0$ are disturbance input actions.
- σ is an admissible input sequence starting at (q,x).

Let $\Sigma(c,(q,x))$ be the set of all sequences like σ. We can now define the set of traces reachable from (q,x) after a control action c as:

$$Tr((q,x),c) = \{\tau((q,x),\sigma) \mid \sigma \in \Sigma(c,(q,x))\}.$$

We also define the set of successors of (q,x) after a control action c as $Reach((q,x),c) = \{(q',x') \mid \exists\sigma \in \Sigma(c,(q,x))\, (q,x) \overset{\sigma}{\to} (q',x')\}$.

We can now proceed to define an *admissible control action sequence* (or admissible control sequence for short). Given a sequence of two control actions $\gamma = c_0, c_1$, we define

$$\Sigma(\gamma,(q,x)) = \{\sigma_0 \oplus \sigma_1 \mid \sigma_0 \in \Sigma(c_0,(q,x)) \\ \wedge \exists(q',x') \in Reach((q,x),c_0) : \sigma_1 \in \Sigma(c_1,(q',x'))\},$$

where $\oplus$ is the concatenation operator. Note that if c_0 is admissible at (q,x) then $Reach((q,x),c_0)$ is not empty; therefore, we say that $\gamma = c_0, c_1$ is admissible starting at (q,x) if $\Sigma(\gamma,(q,x))$ is not empty.

For a sequence γ of more than two control actions, $\Sigma(\gamma,(q,x))$ can be defined similarly. We denote by $S_{\mathcal{C}}(\mathcal{A})$ the set of all admissible control action sequences. We can also define the set of traces starting at (q,x) after an admissible control sequence γ as follows:

$$Tr((q,x),\gamma) = \{\tau((q,x),\sigma) \mid \sigma \in \Sigma(\gamma,(q,x))\}.$$

Observations

In this work, we focus on behavior of continuous systems and thus the properties of interest involve only continuous variables. We assume a set $V_o(\mathcal{A}) \subseteq V(\mathcal{A})$ of continuous variables of the hybrid automaton that are observable by the tester. We also assume that any change in location can be observed by the tester. Given a hybrid state $s = (q,x)$, $cont(s)$ gives the continuous component of s. The definition can be extended to a sequence of states $\tau = (q_0,x_0),\ldots,(q_k,x_k)$ as follows: $cont(\tau) = x_0,\ldots,x_k$. The projection of a continuous state x on $V_o(\mathcal{A})$, denoted by $\pi(x, V_o(\mathcal{A}))$, is called an *observation*.

Definition 4 (Observation Sequence)**.** *Let $\gamma = c_0, c_1, \ldots, c_k$ be an admissible control sequence. Let (q_{init}, x_{init}) be the initial state of $\mathcal{A}$. The set of* observation sequences *associated with γ is*

$$S_{\mathcal{O}}(\mathcal{A},\gamma) = \{\pi(cont(\tau), V_o(\mathcal{A})) \mid \tau \in Tr((q_{init},x_{init}),\gamma)\}.$$

2.2 Specification and System under Test

We assume that the specification is modeled as a hybrid automaton $\mathcal{A}$ and the system under test (such as an implementation) by another hybrid automaton $\mathcal{A}_s$ such that:

- $V_o(\mathcal{A}) \subseteq V_o(\mathcal{A}_s)$ and
- $S_{\mathcal{C}}(\mathcal{A}) \subseteq S_{\mathcal{C}}(\mathcal{A}_s)$.

Note that we do not assume that we know the model $\mathcal{A}_s$. For a given observation sequence $\mathcal{O}$ of $\mathcal{A}$, the operator of projecting $\mathcal{O}$ on a set V of variables is defined componentwise, denoted by $\pi(\mathcal{O}, V)$. Again, we can naturally extend this definition to a set of observation sequences $\pi(S,V) = \{\pi(\mathcal{O},V) \mid \mathcal{O} \in S\}$.

2.3 Testing Problem

The goal of testing is to make statements about the relation between the traces of a system under test and a specification [28]. The system under test $\mathcal{A}_s$ often operates within some environment. In our testing problem, the tester plays the role of the environment and it performs experiments on $\mathcal{A}_s$ in order to study the relation between $\mathcal{A}$ and $\mathcal{A}_s$. The tester works as follows. It emits an admissible control sequence to the system under test and measures the resulting observation sequence in order to produce a verdict $\boldsymbol{v} \in \{P, F, I\}$. The verdict P means 'pass' (the observation sequence is allowed by the specification), F means 'fail' (the observation sequence is not allowed by the specification), and I means 'inconclusive' (neither a 'pass' nor a 'fail' verdict can be assigned). The observations are measured at the

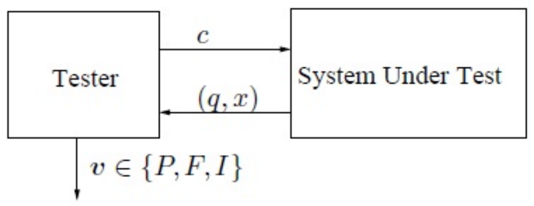

Fig. 1. Test architecture.

end of each continuous control action and after each discrete (disturbance or control) action.

Conformance is an important property of the relation between the system under test and the specification.

Definition 5 (Conformance)**.** *We say that the system under test $\mathcal{A}_s$ is conform to the specification $\mathcal{A}$, denoted by $\mathcal{A} \approx \mathcal{A}_p$, iff $\forall \gamma \in S_{\mathcal{C}}(\mathcal{A}) : \ \pi(S_{\mathcal{O}}(\mathcal{A}_s, \gamma), V_o(\mathcal{A})) \subseteq S_{\mathcal{O}}(\mathcal{A}, \gamma)$.*

Note that since $S_{\mathcal{C}}(\mathcal{A}) \subseteq S_{\mathcal{C}}(\mathcal{A}_s)$, a control sequence which is admissible for $\mathcal{A}$ is also admissible for $\mathcal{A}_s$.

2.4 Test case

A test case is represented by a (finite) tree where each node is associated with an observation and each path from the root with an observation sequence. Each edge of the tree is associated with a control action.

The observation sequences of the trees are grouped into three disjoint sets: the set O_p of observation sequences that cause a 'pass' verdict, the set O_f that cause a 'fail' verdict, and the set O_i that cause an 'inconclusive' verdict. Note that an observation sequence must cause a unique verdict.

Since a hybrid automaton might have an infinite number of infinite traces of a hybrid automaton; however, the tester can only perform a finite number of test cases in finite time. Therefore, we need to select a finite portion of the input space of $\mathcal{A}$ and test the conformance of $\mathcal{A}_s$ with respect to this portion. The selection is done using a coverage criterion that we formally define in the next section. Hence, our testing problem is formulated as to automatically generate a set of test cases from the specification model to satisfy this coverage criterion.

2.5 Test Coverage

Test coverage is a way to evaluate testing quality. More precisely, it is a way to relate the number of tests to carry out with the fraction of the system's behaviors effectively explored. As mentioned earlier, the classic coverage notions mainly used in software testing, such as statement coverage and if-then-else branch coverage, path coverage (see for example [26], [24]), are not appropriate for the trajectories of continuous and hybrid systems defined by differential equations. However, the geometry of the hybrid state space can be exploited to define a coverage measure which, on one hand, has a close relationship with the properties to verify and, on the other hand, can be efficiently computed or estimated.

In discrete circuit testing, fault coverage is an important concern. A fault is said to be detected by a test input pattern if, when applying the input pattern to the circuit, different output patterns can be observed, for the reference (non-faulty) circuit and the faulty circuit. Generally, faults can be categorized into catastrophic and parametric faults. Examples of catastrophic faults include a change in the circuit topology, a global deviation of the circuit behavior. Parametric faults refer to small changes in the parameters that do not affect the circuit functionality. For example, a band-pass filter which has a frequency response with the correct shape but it passes a larger range of frequencies. It is often assumed that beyond a deviation of 10% is considered to have caused faults.

For this reason, we are motivated in defining a coverage measure that describes how 'well' the visited states represent the set of all reachable states of the system. 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. We first briefly recall the star discrepancy. The star discrepancy is an important notion in equidistribution theory as well as in quasi-Monte Carlo techniques (see for example [5]).

Star Discrepancy

Let P be a set of k points inside $\mathcal{B} = [l_1, L_1] \times \ldots \times [l_n, L_n]$. Let Γ be the set of all sub-boxes J of the form $J = \prod_{i=1}^{n}[l_i, \beta_i]$ with $\beta_i \in [l_i, L_i]$ (see Figure 2 for an illustration). The local discrepancy of the point set P with respect to the subbox J is defined as follows: $D(P, J) = |\frac{A(P, J)}{k} - \frac{\lambda(J)}{\lambda(\mathcal{B})}|$ where $A(P, J)$ is the number of points of P that are inside J, and $\lambda(J)$ is the volume of the box J. The star discrepancy of P with respect to the box $\mathcal{B}$ is defined as:

$$D^*(P, \mathcal{B}) = sup_{J \in \Gamma} D(P, J) \qquad (2)$$

Note that $0 < D^*(P, \mathcal{B}) \leq 1$.

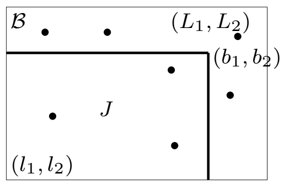

Fig. 2. Illustration of the star discrepancy notion.

Intuitively, the star discrepancy is a measure for the irregularity of a set of points. A large value $D^*(P, \mathcal{B})$ means that the points in P are not much equidistributed over $\mathcal{B}$. When the region is a hyper-cube, the star discrepancy measures how badly the point set estimates the volume of the cube. Since a hybrid system can only evolve within the staying sets of the locations, we are interested in the coverage with respect to these sets. For simplicity we assume that all the staying sets are boxes.

Definition 6 (Test Coverage)**.** *Let $\mathcal{P} = \{(q, P_q) \mid q \in Q \wedge P_q \subset \mathcal{I}_q\}$ be the set of states. The coverage of $\mathcal{P}$ is defined as: $Cov(\mathcal{P}) = \frac{1}{||Q||}\sum_{q \in Q} 1 - D^*(P_q, \mathcal{I}_q)$ where $||Q||$ is the number of locations in Q.*

If a staying set $\mathcal{I}_q$ is not a box, we can take the smallest oriented box that encloses it, and apply the star discrepancy definition in (2) to that box after an appropriate coordination change. We can see that a large value of $Cov(\mathcal{P})$ indicates a good space-covering quality. If $\mathcal{P}$ is the set of states visited by a test suit, our objective is to maximize $Cov(\mathcal{P})$.

3 Test Generation

Our test generation is a combination of the Rapidly-exploring Random Tree (RRT) algorithm (a successful robot motion planning technique [18]) and a guiding tool used to achieve a good coverage of the system's behaviors we want to test. We call the resulting algorithm gRRT. In this section, we present only the main ideas of gRRT, and a detailed description of the algorithm can be found in [19].

The algorithm constructs a tree denoted by $\mathcal{T}$, the root of which corresponds to the initial state $s_0 = (q_0, x_0)$. The construction of the tree is summarized in Algorithm 1. In each iteration, a hybrid state $s_{goal} = (q_{goal}, x_{goal})$ is sampled to indicate the direction towards which the tree is expected to evolve. Expanding the tree towards s_{goal} is done by making a continuous step as follows

- First, a starting state $s_{near} = (q_{near}, x_{near})$ for the current iteration is determined. It is natural to choose s_{near} to be a state near s_{goal}. The distance between two hybrid states is defined as an average length of the traces from s_{near} to s_{goal} (see [19] for more detail).
- Next, the procedure ContinuousStep tries to find the input $\bar{u}_{q_{near}}$ to take the system from s_{near} towards s_{goal} as closely as possible after one time step h. This results in a new continuous state x_{new}. A new edge from s_{near} to $s_{new} = (q_{near}, x_{new})$, labeled with the associated continuous control action is then added to the tree. To find s_{new}, when the set U is not finite it can be sampled, or one can solve a local optimal control problem.

Then, from s_{new}, we compute its successors by all possible discrete transitions. Each time we add a new edge, we label it with the associated control or disturbance action. The algorithm terminates after reaching a satisfactory coverage value or after some maximal number of iterations. To extract a test case from the tree, we project the states at the nodes on the observable variables of $\mathcal{A}$. In Algorithm 1, the function Sampling plays the role of guiding the exploration, which will be described later. To deal with a set of initial states, we can sample a finite number of initial states and thus the tree can have more

Algorithm 1 gRRT - Test generation algorithm

```
$\mathcal{T}.init(s_0)$, $j = 1$                          ▷ $s_0$: initial state
repeat
    $s_{goal} = \text{Sampling}(\mathcal{S})$          ▷ $\mathcal{S}$: hybrid state space
    $s_{near} = \text{Neighbor}(\mathcal{T}, s_{goal})$
    $(s_{new}, \bar{u}_{q_{near}}) = \text{ContinuousStep}(s_{near}, h)$
    $\text{DiscreteSteps}(\mathcal{T}, s_{new})$, $j{+}{+}$
until $j \geq J_{max}$
```

than one root. Similarly, to deal with parameter sets, we can consider a finite number of parameter values associated and associate them with each initial state. Along a path from each root, the parameter vector remains constant and is used to determine the corresponding continuous dynamics.

Coverage Estimation

To evaluate the coverage of a set of states, we estimate a lower and upper bound of the star discrepancy of a point set (exact computation is not an easy problem). These bounds as well as the information obtained from their estimation are used to decide which parts of the state space have been 'well explored' and which parts need to be explored more. Let us briefly describe this estimation method. Let $\mathcal{B} = [l_1, L_1] \times \ldots \times [l_n, L_n]$. We define a box partition of $\mathcal{B}$ as a set of boxes $\Pi = \{\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m\}$ such that $\cup_{i=1}^m \boldsymbol{b}^i = \mathcal{B}$ and the interiors of the boxes $\boldsymbol{b}^i$ do not intersect. Each such box is called an *elementary box*. Given a box $\boldsymbol{b} = [a_1, b_1] \times \ldots \times [a_n, b_n] \in \Pi$, we define $\boldsymbol{b}^+ = [l_1, b_1] \times \ldots \times [l_n, b_n]$ and $\boldsymbol{b}^- = [l_1, a_1] \times \ldots \times [l_n, a_n]$ (see Figure 3 for an illustration).

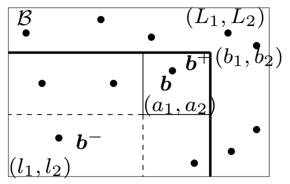

Fig. 3. Illustration of the star discrepancy estimation.

For any finite box partition Π of $\mathcal{B}$, the star discrepancy $D^*(P, \mathcal{B})$ of the point set P with respect to $\mathcal{B}$ satisfies: $C(P, \Pi) \leq D^*(P, \mathcal{B}) \leq B(P, \Pi)$ where the upper and lower bounds are: $B(P, \Pi) = \max_{\boldsymbol{b} \in \Pi} \max\{\frac{A(P, \boldsymbol{b}^+)}{k} - \frac{\lambda(\boldsymbol{b}^-)}{\lambda(\mathcal{B})}, \frac{\lambda(\boldsymbol{b}^+)}{\lambda(\mathcal{B})} - \frac{A(P, \boldsymbol{b}^-)}{k}\}$ and $C(P, \Pi) = \max_{\boldsymbol{b} \in \Pi} \max\{|\frac{A(P, \boldsymbol{b}^-)}{k} - \frac{\lambda(\boldsymbol{b}^-)}{\lambda(\mathcal{B})}|, |\frac{A(P, \boldsymbol{b}^+)}{k} - \frac{\lambda(\boldsymbol{b}^+)}{\lambda(\mathcal{B})}|\}$.

Coverage-Guided Sampling

In each iteration, we want to bias the goal state sampling distribution according to the current coverage of the visited states. More concretely, we first sample a discrete location and then a continuous state. Let $\mathcal{P} = \{(q, P_q) \mid q \in Q \wedge P_q \subset \mathcal{I}_q\}$ be the current set of visited states. The discrete location sampling distribution depends on the current continuous state coverage of each location:

$$Pr[q_{goal} = q] = \frac{D^*(P_q, \mathcal{I}_q)}{\sum_{q' \in Q} D^*(P_{q'}, \mathcal{I}_{q'})}.$$

We now show how to sample x_{goal}, assuming that we have already sampled a discrete location $q_{goal} = q$. The sampling process consists of two steps. In the first step, we sample an elementary box $\boldsymbol{b}_{goal}$ from the set Π; in the second step we sample a point x_{goal} in $\boldsymbol{b}_{goal}$ uniformly. The elementary box sampling distribution in the first step is biased in order to optimize the coverage. The intuition behind this guiding strategy is to favor the selection of an elementary box such that a new point x added in this box results in a reduction of the lower and upper bounds. In words, the essential idea of this guiding method is that we use the information about the current coverage in order to improve it (see [19]).

An important remark is that when the propery of interest involves only a subset of continuous variables, we can define a coverage measure with respect to the projection of the state space on these variables, and use it to guide the sampling process.

Before continuing, we recall that the gRRT algorithm preserves the *probabilistic completeness* of the RRT algorithm. The proof can be found in [19]. Roughly speaking, the probabilistic completeness property states that if the trace we seach for is feasible, then the probability that the algorithm finds it approaches 1 as the number k of iterations approaches infinity. This property is a way to explain a good space-covering property of the RRT algorithm and its successes in solving practical motion planning problems [18]. Our test generation algorithm has been tested on a number of control applications, which proved its scalability to high dimensional systems [19].

4 Application to Embedded system Benchmarks

4.1 Voltage controlled oscillator

We examined in this section the voltage controlled oscillator (VCO). This circuit is taken from [12], shown in Figure 4. The behavior of this circuit can be represented by a system of differential-algebraic equations DAEs with 55 continuous variables. The input is modeled by an ideal voltage controlled current which is mirrored by $TN1$, $TP1$, $TN2$, $TP2$, $TN5$ charging the capacitance C_2. Assuming the output voltage v_{C_1} to be at V_{DD}, C_2 is charged up linearly by the input current through the switch $TN3$, $TP3$ controlled by the inverter ($TN4$, $TP4$). As v_{C_2} exceeds the positive threshold voltage of the Schmitt trigger, determined by the resistors R_1 and R_2, v_{C_1} changes to V_{SS}. Consequently, C_2 is discharged until the initial status is reached leading to an oscillation.

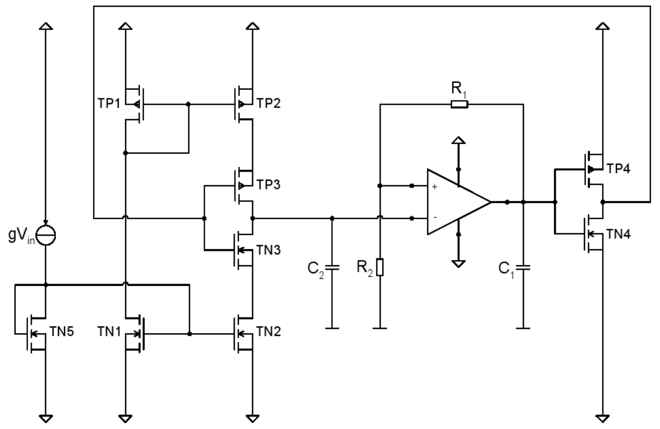

Fig. 4. Voltage controlled oscillator (VCO) circuit.

Testing: We are interested the oscillating frequency of the variables v_{C_1} and v_{C_2}. Indeed, this frequency is a linear function of the input voltage. We study the influence of a time-variant perturbation in C_2 on the frequency. This perturbation is modeled as an input signal. In this example we show that, in addition to conformance relation, using this framework, we can test a property of the input/output relation. More precisely, for a given input sequence, we want to compare the observation sequences of the specification $\mathcal{A}$ and those of the system under test $\mathcal{A}_s$, with respect to a property. As an example, the oscillating period $T \pm \delta$ of a variable $x_1 = v_{C_2}$ can be expressed using a simple automaton with one clock y in Figure 5 (similar to a monitor automaton in [11]). The question is to know if given an oscillating trace in $\mathcal{A}$, its corresponding trace in $\mathcal{A}_s$ is also oscillating with the same period. This additional automaton can be used to determine test verdicts for the traces in the computed test cases. If we are additionally interested in a property which states that all traces of the system under test oscillate with the period $T \pm \delta$. If an observation sequence corresponds to a path entering the 'Error' location, then it causes a 'fail' verdict. Since we cannot use finite traces to prove a safety property, the set of observation sequences that cause a 'pass' verdict is empty, and therefore the remaining observation sequences (that do not cause a 'fail' verdict) causes a 'inconclusive' verdict.

Results: We consider a constant input voltage $u_{in} = 1.7$. The generated test case shows that after the transient time, under a time-variant deviation of C_2 which ranges within $\pm 10\%$ of the value of $C_2 = 0.1e-4$, the variables v_{C_1} and v_{C_2} oscillate with the period $T \in [1.25, 1.258]s$ (with $\varepsilon = 2.8e-4$). This result is consistent with the result presented in [12]. Figure 6 shows the observation sequence projected on v_{C_1} and v_{C_2}. We note that the number of generated states was 30000 with 5 sampled input values and the computation time was 14mn. In this experiment, the coverage measure was defined on

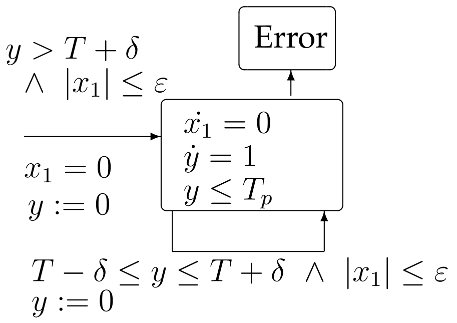
Error
$y > T + \delta$
$\wedge\ |x_1| \le \varepsilon$
$\dot{x}_1 = 0$
$\dot{y} = 1$
$y \le T_p$
$x_1 = 0$
$y := 0$
$T - \delta \le y \le T + \delta\ \wedge\ |x_1| \le \varepsilon$
$y := 0$

Fig. 5. Automaton for an oscillation specification.

the projection of the state space on v_{C_1} and v_{C_2}, and not on the full-dimensional state space.

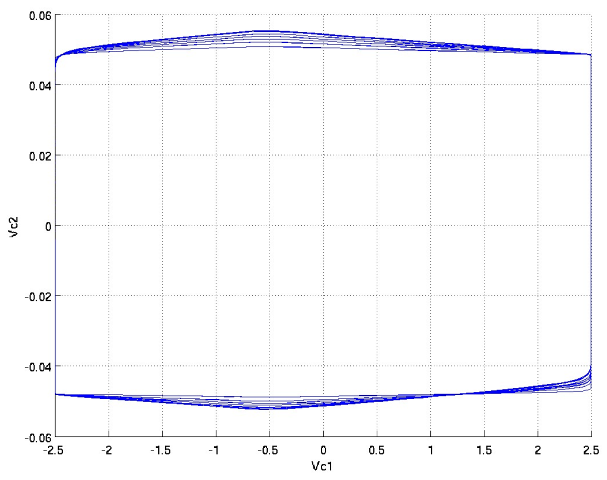

Fig. 6. Variables v_{C_1} and v_{C_2} projection.

5 Delta-Sigma circuit

In this section we treat a mixed-signal circuit, which is a Delta-Sigma modulator ([20], [29]), This circuit is a very popular technique to perform analog to digital conversion. The principles of Delta-Sigma modulation [21] can be described by the following points:

- Anti-aliasing: used to ensure that the signal band width lies within a given range $[-f_b, f_b]$,
- Oversampling or sampling at a frequency greater than the Nyquist rate $2 \times f_b$,
- Noise shaping so that the quantization error is 'pushed' toward high frequencies outside the bandwidth of interest,
- Quantization, typically on few bits.

Figure 7 shows how the Delta-Sigma modulation works. When the input sinusoid is positive and its value is less than 1, the output takes the $+1$ value more often and the quantization error which is the difference between the input and the output of the quantizer is fed back with negative gain and 'accumulated' in the integrator

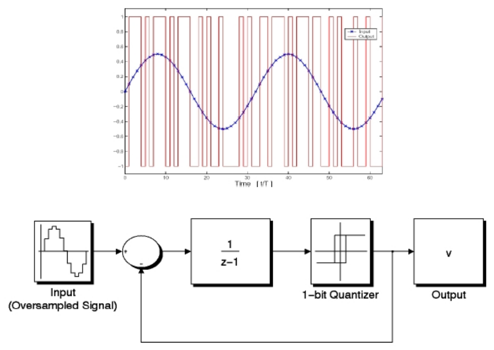

Fig. 7. A first order Delta-Sigma modulator and an example of an input-output plot.

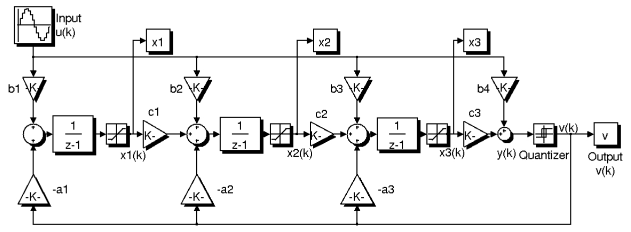

Fig. 8. Model of a third-order modulator: Saturation blocks model saturation of the integrator.

$\frac{1}{z-1}$. Then, when the accumulated error reaches a certain threshold, the quantizer switches the value of the output to -1 for some time, which reduces the mean of the quantization error.

A third-order Delta-Sigma modulator is modeled as a hybrid automaton, shown in Figure 9. It is called third-order since it uses a third order filter to process noise.

Higher-order modulators achieve better performance but induce stability problems. A modulator is said to be stable if under a bounded input, the states of its integrators are bounded. Stability analysis for such circuits is still a challenging problem [20], due to the presence of two sources of non-linearities: saturation and quantization. The discrete-time dynamics of the system is as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + bu(k) - sign(y(k))a, \qquad (3)\\ y(k) &= c_3x_3(k) + b_4u(k), \qquad (4) \end{aligned}$$

where matrix A, vectors a and b are constants depending on the various gains of the model, $x(k) \in \mathbb{R}^3$ represents 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 dynamics is affine continuous. A modulator is stable if under a bounded input, the states of its integrators are bounded.

Results: We are intersted in testing the stability property of the modulator. The test generation algorithm

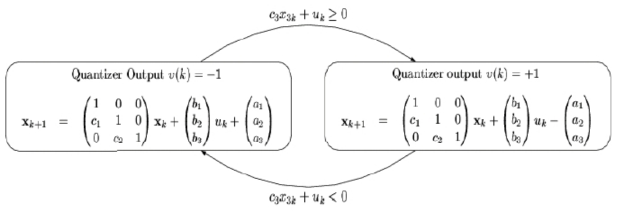
$c_3x_{3k} + u_k \geq 0$
Quantizer Output $v(k) = -1$
$\mathbf{x}_{k+1} = \begin{pmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ 0 & c_2 & 1 \end{pmatrix} \mathbf{x}_k + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} u_k + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$
Quantizer output $v(k) = +1$
$\mathbf{x}_{k+1} = \begin{pmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ 0 & c_2 & 1 \end{pmatrix} \mathbf{x}_k + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} u_k - \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$
$c_3x_{3k} + u_k < 0$

Fig. 9. A hybrid automaton model of the Delta-Sigma modulator.

was performed for the initial state $x(0) \in [-0.01, 0.01]^3$ and the input values $u(k) \in [-0.5, 0.5]$. After exploring only 57 states, saturation was already detected. The computation time was less than 1 second. Figure 10 shows the values of $(\sup x_1(k))_k$ as a function of the number k of time steps. We can see that the sequence $(\sup x_1(k))_k$ leaves the safe interval $[-x_1^{sat}, x_1^{sat}] = [-0.2, 0.2]$, which indicates the instability of the circuit. This instability for a fixed finite horizon was also detected in [8] using an optimization-based method.

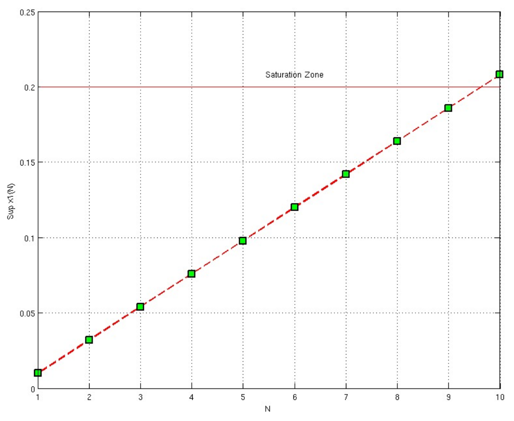

Fig. 10. Test generation result for the Delta-Sigma circuit.

6 AIRCRAFT COLLISION

This case study is one of the well-known benchmarks in the hybrid systems literature [1]. In this paper, the authors treated the problem of collision avoidance of two aircraft. To show the scalability of our approach we consider the same model with N aircraft.

As shown in Figure 11, all the aircraft are at a fixed altitude. Each aircraft i has three states (x_i, y_i, θ_i) where x_i and y_i describe the position and θ_i is the relative heading of the aircraft. Each aircraft begins in straight flight at a fixed relative heading (mode 1).

Then, as soon as two aircraft are within the distance R (km) between each other, they enter mode 2. In this mode each aircraft makes an instantaneous heading change of 90 degrees, and begins a circular flight for π time units. After that, they switch to mode 3 and make another

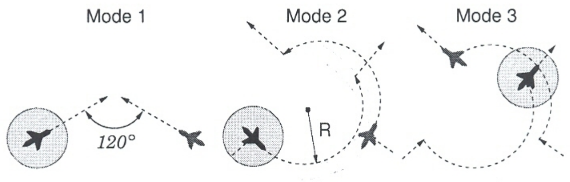

Fig. 11. Aircraft behavior in the three modes [1].

instantaneous heading change of 90 degrees and resume their original headings from mode 1.

The dynamics of the system are shown in Figure 12. The guard transition between mode 1 and mode 2 is given by $D(i, j) < R$ which means that the aircraft i is at R (km) distance from the aircraft j. The dynamics of each aircraft is as follows: For an ideal situation without external disturbances, we have:

$$\begin{aligned} \dot{x}_i &= vcos(\theta_i) + d_1 sin(\theta_i) + d_2 cos(\theta_i), \\ \dot{y}_i &= vsin(\theta_i) - d_1 cos(\theta_i) + d_2 sin(\theta_i) \\ \dot{\theta}_i &= \omega \\ \dot{\theta}_i &= 0 \end{aligned}$$

The continuous inputs are dx_i and dy_i describing the external disturbances on the aircraft (such as wind):

$$\begin{aligned} dx_i &= d_1 sin(\theta_i) + d_2 cos(\theta_i), \\ dy_i &= -d_1 cos(\theta_i) + d_2 sin(\theta_i), \end{aligned}$$

and $-\delta \leq d_1, d_2 \leq \delta$.

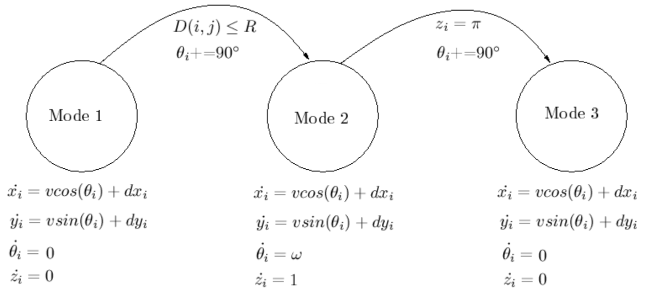
$D(i,j) \leq R$
θ_i+=90°
$z_i = \pi$
θ_i+=90°
Mode 1
Mode 2
Mode 3
$\dot{x_i} = vcos(\theta_i) + dx_i$
$\dot{y_i} = vsin(\theta_i) + dy_i$
$\dot{\theta}_i = 0$
$\dot{z}_i = 0$
$\dot{x_i} = vcos(\theta_i) + dx_i$
$\dot{y_i} = vsin(\theta_i) + dy_i$
$\dot{\theta}_i = \omega$
$\dot{z}_i = 1$
$\dot{x_i} = vcos(\theta_i) + dx_i$
$\dot{y_i} = vsin(\theta_i) + dy_i$
$\dot{\theta}_i = 0$
$\dot{z}_i = 0$

Fig. 12. System dynamics for the three modes.

Results: Thus for N aircraft, the system has $3N+1$ continuous variables (one for modeling a clock). For the case of $N = 2$ aircraft, when the collision distance is 5 no collision was detected after visiting 10000 visited states, and the computation time was 0.9 min. The result for $N = 8$ aircraft with the disturbance bound $\delta = 0.06$ is shown in Figure 13. For this example, the computation time for 50000 visited states was 10 min and a collision was found. For a similar example with $N = 10$ aircraft, the computation time was 14 minutes and a collision was also found. In Figure 13 we show the projected positions of the eight aircraft on a 2-dimensional space.

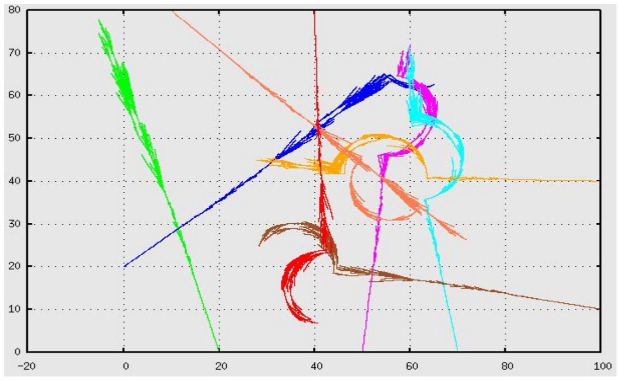

Fig. 13. Eight-aircraft collision avoidance (50000 visited states, computation time: 10 min.

REFERENCES

[1] I. Mitchell and C. Tomlin. Level Set Methods for Computation in Hybrid Systems HSCC, LNCS 1790, Springer, 2000.
[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*, 138:3–34, 1995.
[3] B. Ayai, N. Ben-Hamida, and B. Kaminska. Automatic test vector generation for mixed-signal circuits. In *Proc. European Design and Test Conference.*, pages 458-463, March 1995.
[4] V. Acary and F. Pérignon. SICONOS: A software platform for modeling, simulation, analysis and control of Non Smooth Dynamical system. *Proc. of MATHMOD*, ARGSIM Verlag, 2006.
[5] J. Beck and WWL. Chen. Irregularities of Distribution. Cambridge Univ. Press, 1987.
[6] B. Burdiek Generation of Optimum Test Stimuli for Nonlinear Analog Circuits Using Nonlinear Programming and Time-Domain Sensitivities. In *Proc. DATE*, pages 603-609, 2001.
[7] M. S. Branicky, M. M. Curtiss, J. Levine, and Stuart Morgan. Sampling-based reachability algorithms for control and verification of complex systems. *Proc. Thirteenth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, 2005.
[8] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *FMCAD*, LNCS, Springer, 2004.
[9] A. Bhatia and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. HSCC, LNCS 2993, pages 142-156, Springer, 2004.
[10] J.M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Int. Workshop on the Algorithmic Founddations of Robotics* , 2004.
[11] G. Frehse, B. Krogh, R. Ruttenbar, and O. Maler. Time domain verification of oscillator circuit properties. Proceeding of Workshop on Formal Verification of Analog Circuits (ETAPS Satellite Event), Edinburgh, ENTCS 153(3): 9-22, 2005.
[12] D. Grabowski, D. Platte, L. Hedrich and E. Barke Time Constrained Verification of Analog Circuits using Model-Checking Algorithms. Proceeding of Workshop on Formal Verification of Analog Circuits (ETAPS Satellite Event), Edinburgh, ENTCS 153(3): 37-52, 2005.
[13] E. Hairer, C. Lubich and M. Roche. The Numerical Solution of Differential-Algebraic Systems by Runge Kutta Methods. Lecture Notes in Mathematics 1409, Springer-Verlag, 1989.
[14] N. B. Hamida, K. Saab, D. Marche, B. Kaminska, and G. Quesnel. LIMSoft: Automated Tool for Design and Test Integration of Analog Circuits. In *Proc. International Test Conference*, 1996.
[15] N. Ben-Hamida, K. Saab, D. Marche, and B. Kaminska. A perturbation based fault modeling and simulation for mixed-signal circuits. In *Proc. IEEE Asian Test Symposium*, pages 182-187, 1997.
[16] H. Kerkhoff, R. Tangelder, H. Speek, and N. Engin. MISMATCH: A Basis for Semi-automatic Functional Mixed-Signal Test-Pattern Generation. In *Proc. IEEE Int. Conf. on Electronics, Circuits, and Systems*, pages 1072-1075, 1996.
[17] KuffnerLaValle00 J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 995–1001, 2000.
[18] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.
[19] T. Nahhal and T. Dang. Test Coverage for Continuous and Hybrid Systems In *Computer Aided Verification* CAV'07, LNCS, Berlin, 2007.
[20] B. Pérez-Verdú and F. Medeiro and A. Rodríguez-Vázquez. *Top-Down Design of High-Performance Sigma-Delta Modulators*, chapter 2. Kluwer Academic Publishers, 2001.
[21] Aziz, P. M. and Sorensen, H. V. and van der Spiegel, J. *An overview of sigma-delta converters*, VOLUME 2 pages 61–84. Signal Processing journal Magazine, IEEE, 1996
[22] R. Shi and W.T. Tian. Automatic Test Generator of Linear Analog Circuits Under Parameters Variations. In *Proc. Asian and South Pacific Design Automation Conference*, 1998.
[23] L. Tan, J. Kim, O. Sokolsky and I. Lee. Model-based Testing and Monitoring for Hybrid Embedded Systems. In *IRI*, 487-492, 2004.
[24] J. Tretmans. Testing Concurrent Systems: A Formal Approach. In *Int. Conference on Concurrency Theory CONCUR*, LNCS 1664, Springer, 1999.
[25] W. Verhaegen, G. Van der Plas, and G. Gielen. Automated test pattern generation for analog integrated circuits. In *Proc. IEEE VLSI Test Symposium*, pages 296-301, 1997.
[26] H. Zhu, P.A.V. Hall, and J.H.R. May. Software Unit Test Coverage and Adequacy. In *ACM Computing Surveys*, 29, 4. Dec. 1997.
[27] M. Zwolinski, S.J. Spinks, C.D. Chalk, and I.M. Bell. Generation and Verification of Tests for Analog Circuits Subject to Process Parameter Deviations. In *IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Paris, October 1997.
[28] Ananda Rao Akepogu Kiran Kumar J. An Approach to Cost Effective Regression Testing in Black-Box Testing Environment. In *In IJCSI Volume 8, Issue 3, May 2011.*
[29] Gururaj Balikatti, R.M Vani and P.V. Hunagund. Efficient Circuit Configuration for Enhancing Resolution of 8-bit flash Analog to Digital Convertor. In *In IJCSI Volume 9, Issue 6, November 2012.*

Prof. Tarik NAHHAL received his PhD in Computer Sciences at the University of Grenoble (France). He is a Associate Professor at the Department of Mathematics and Computer Sciences, University Hassan II of Casablanca. His research interests are related to Hybrid system Verification, Formal Methods in Conformance Testing and Embedded Systems design. He has published research papers at national and international journals, conference proceedings as well as chapters of books.