Proceedings of the 12th Conference on Formal Methods in Computer-Aided Design (FMCAD 2012)

A quantifier-free SMT encoding of non-linear hybrid automata

Alessandro Cimatti
Fondazione Bruno Kessler
Email: cimatti@fbk.eu

Sergio Mover
Fondazione Bruno Kessler
Email: mover@fbk.eu

Stefano Tonetta
Fondazione Bruno Kessler
Email: tonettas@fbk.eu

***Abstract*—Hybrid systems are a clean modeling framework for embedded systems, which feature integrated discrete and continuous dynamics. A well-known source of complexity comes from the time invariants, which represent an implicit quantification of a constraint over all time points of a continuous transition.**

Emerging techniques based on Satisfiability Modulo Theory (SMT) have been found promising for the verification and validation of hybrid systems because they combine discrete reasoning with solvers for first-order theories. However, these techniques are efficient for quantifier-free theories and the current approaches have so far either ignored time invariants or have been limited to linear hybrid automata[1].

In this paper, we propose a new method that encodes a class of hybrid systems into transition systems with quantifier-free formulas. The method does not rely on expensive quantifier elimination procedures. Rather, it exploits the sequential nature of the transition system to split the continuous evolution enforcing the invariants on the discrete time points. This pushes the application of SMT-based techniques beyond the standard linear case.

I. Introduction

Embedded systems (e.g. control systems for railways, avionics, and space) feature the interaction of discrete systems with the environment by means of controlled and monitored variables that evolve continuously in time. The validation and verification of embedded systems designs must often take into account a model of the continuous evolution of such variables. *Hybrid systems* [3] are a clean modeling framework for embedded systems because they exhibit both continuous transitions ruled by flow conditions (i.e., constraints on the derivatives of continuous variables) and discrete changes represented with logical formulas. A fundamental step in the design of these systems is the validation and verification of the models, performed by checking properties such as invariants or reachability. In spite of the undecidability of these problems, several verification techniques have been developed and have proved to be applicable in a wide number of cases. Among these techniques, common approaches are the computation of the reachable states, and the use of abstraction or deduction systems (see [2] for a recent survey).

An emerging approach to the verification of hybrid systems is the application of verification techniques based on SMT [5]. The hybrid system is encoded into a symbolic transition system and reachability problems are represented by means of first-order formulas. The encoding allows the application of general-purpose SAT-based verification techniques such as Bounded Model Checking (BMC) [6], interpolation-based model checking [27], k-induction [32], and predicate abstraction [20]. Examples of such SMT-based approaches are [4], [1], [24], [22], [17], [25], [18], [23]. Specific techniques have also been proposed for networks of hybrid systems [9], [11], [13], and for requirements [14]. Also thanks to the strong progress in the field of SMT, these approaches are increasingly applied in real settings (e.g. the design of complex space systems [7], [8], [34]).

A well-known problem of this approach is the encoding of invariants. In order to preserve the semantics of the hybrid system, the formula representing a continuous (timed) transition between two time points t and t' must guarantee that the invariant holds along all points of the implicit continuous evolution between the state $s(t)$ and the state $s(t')$. A straightforward approach would create a quantified formula which treats the invariant as a formula $Inv(t)$ over the variable representing real time and quantifies the formula along all time points of the timed transition, i.e., $\forall \epsilon \in [t, t'], Inv(\epsilon)$. In general, it is an open question how to handle such quantifiers (see for example [1], [18]): the elimination of quantifiers is in general not possible, and when the elimination is theoretically possible (such as in the case of the theory of reals, i.e., polynomial constraints) it is in practice not feasible beyond the quadratic case. Only in particular cases (such as when the continuous evolution of variables is linear in time, and Inv is convex), the encoding is equivalent to the quantifier-free formula $Inv(t) \wedge Inv(t')$.

In this paper, we propose a new approach to efficiently encode invariants as quantifier-free formulas. Intuitively, the encoding can be thought of as generalizing the linear case, forcing the invariant before and after the timed transition ($Inv(t) \wedge Inv(t')$), and imposing the derivative of the invariant to be constant in sign throughout the timed transition. This reduces the invariant to a quantified formula over the derivatives of the continuous variables. Applying these reduction recursively, in some cases, one may obtain a quantifier-free encoding. This is guaranteed, for example, in the case of polynomial hybrid automata, where the derivatives eventually reduce to zero. We obtain a quantifier-free encoding also in interesting cases of non-linear hybrid automata with

[1] In the context of this paper, the terms “linear”, “non-linear”, and “polynomial” refer to the formulas over the time variable used to describe continuous and discrete transitions, and not to the type of ODE.

transcendental functions. As a result, a key contribution of the paper is a quantifier-free encoding of polynomial hybrid automata, which enables the application of SMT-based verification techniques to a broader class of hybrid systems. The approach has been implemented and evaluated on a set of benchmarks. The analysis shows that the proposed technique allows us to solve problems where an abstraction that simply ignores the invariants is too coarse to guarantee soundness and completeness.

The rest of this paper is structured as follows. In Sec. II we present some background. In Sec. III and IV we present the encoding, together with the statement of correctness. A comparison with related work is described in Section V, whilst the experimental evaluation is presented in Section VI. In Section VII we draw some conclusions, and outline directions for future work.

II. Background

A. First-order Transition Systems

Given a set V of variables, we denote with $V', \dot{V}, V^0, V^1, \ldots$ copies of such set. Given a first-order signature Σ, a first-order Σ-Transition System (TS) is a tuple $S = \langle V, \mathit{Init}, \mathit{Inv}, \mathit{Trans}\rangle$ such that:

- V is a set of variables;
- Init is a first-order Σ-formula over V (called initial condition);
- Inv is a first-order Σ-formula over V (called invariant condition);
- Trans is a first-order Σ-formula over $V \cup V'$ (called transition condition).

Let $\Sigma_{\mathbb{R}}$ be the standard signature of real ordered field. In the following, we will consider signatures Σ that are extensions of $\Sigma_{\mathbb{R}}$, the structure $\overline{\mathbb{R}}$ of the real ordered field extended with transcendental functions such as the exponential and the trigonometric functions, and formulas will be interpreted in an appropriate extension of the first-order theory of the real numbers for such structure $\overline{\mathbb{R}}$.

A *state* s is an assignment to the variables V. We denote with $s', \dot{s}, s^0, s^1, \ldots$ the corresponding assignment to the copy $V', \dot{V}, V^0, V^1, \ldots$ of V.

A sequence $s_0, s_1, \ldots, s_k$ of states is a model (also called *path*) of the transition system $S = \langle V, \mathit{Init}, \mathit{Inv}, \mathit{Trans}\rangle$ iff:

- s_0 satisfies Init;
- for every $0 \leq i \leq k$, s_i satisfies Inv;
- for every $0 \leq i < k$, s_i, s'_{i+1} satisfy Trans.

Many verification techniques for transition systems such as Bounded Model Checking (BMC) [6] are based on satisfiability checking interacting with queries to SAT/SMT solvers.

B. Hybrid traces

We denote with $\dot{f}$ the derivative of a real function f. Let I be an interval of $\mathbb{R}$ or $\mathbb{N}$; we denote with $le(I)$ and $ue(I)$ the lower and upper endpoints of I, respectively. We denote with $\mathbb{R}^+$ the set of non-negative real numbers.

Hybrid traces [15], [14] describe the evolution of variables in every point of time. Such evolution is allowed to have a countable number of discontinuous points corresponding to changes in the discrete part of the model.

A *hybrid trace* over discrete variables V and continuous variables X is a sequence $\langle \overline{f}, \overline{I}\rangle := \langle f_0, I_0\rangle, \langle f_1, I_1\rangle, \ldots, \langle f_k, I_k\rangle$ such that, for all i, $0 \leq i \leq k$,

- the intervals are adjacent, i.e. $ue(I_i) = le(I_{i+1})$;
- $le(I_0) = 0$ and I_k is right closed;
- $f_i : V \cup X \to \mathbb{R} \to \mathbb{R}$ is a function such that, for all $x \in X$, $f_i(x)$ is differentiable, and for all $v \in V$, $f_i(v)$ is constant;
- if I_i is left open [right open] and $le(I_i) = t$ [$ue(I_i) = t$] then, for all $v \in V \cup X$, $f_i(v)(t) = f_{i-1}(v)(t)$, [$f_i(v)(t) = f_{i+1}(v)(t)$].

We say that a trace is a sampling refinement of another one if it has been obtained by splitting an open interval into two parts by adding a sampling point in the middle [15]. A *partitioning function* μ is a sequence $\mu_0, \mu_1, \mu_2, \ldots$ of non-empty, adjacent and disjoint intervals of $\mathbb{N}$ partitioning $\mathbb{N}$. Formally, $\bigcup_{i \in \mathbb{N}} \mu_i = \mathbb{N}$ and $ue(\mu_i) = le(\mu_{i+1}) - 1$. A hybrid trace $\langle \overline{f}', \overline{I}'\rangle$ is a *sampling refinement* of $\langle \overline{f}, \overline{I}\rangle$ (denoted with $\langle \overline{f}', \overline{I}'\rangle \preceq \langle \overline{f}, \overline{I}\rangle$) iff, there exists a partitioning μ such that for all $i \in \mathbb{N}$, $I_i = \bigcup_{j \in \mu_i} I'_j$ and, for all $j \in \mu_j$, $f'_j = f_i$. We extend the relation to set L_1 and L_2 of traces as follows: $L_1 \preceq L_2$ iff for every trace $\sigma_2 \in L_2$ there exists $\sigma_1 \in L_1$ such that $\sigma_1 \preceq \sigma_2$.

In the paper, we will assume that the evolution of predicates along time have the finite variability property: we say that a predicate $P(t)$ over a real variable t has *finite variability* [30] iff for any bounded interval J there exists a finite sequence of real numbers $t_0 < \ldots < t_n$ such that $t_0 = le(J)$, $t_n = ue(J)$, and for all $i \in [1, n]$, either for all $\epsilon \in (t_{i-1}, t_i)$, $P(\epsilon)$ or for all $\epsilon \in (t_{i-1}, t_i)$, $\neg P(\epsilon)$. The last condition means that the predicate is constant in the interval (t_{i-1}, t_i). If P is in the form $g(t) \bowtie 0$ with g continuous and $\bowtie \in \{\geq, \leq, <, >\}$, in the points in which P changes value, $g(t) = 0$. Thus, $g \bowtie 0$ has finite variability iff for any bounded interval J there exists a finite sequence of real numbers $t_0 < \ldots < t_n$ such that $t_0 = le(J)$, $t_n = ue(J)$, and for all $i \in [1, n]$, either for all $\epsilon \in [t_{i-1}, t_i]$, $g(\epsilon) \geq 0$ or for all $\epsilon \in [t_{i-1}, t_i]$, $g(\epsilon) \leq 0$. We denote this condition with *Constant*(P, t_{i-1}, t_i).

Proposition 1: Assuming that a predicate P has finite variability, for every hybrid trace σ, there exists a sampling refinement of σ for which which P is constant in the open part of every interval.

C. Hybrid systems

Hybrid systems [3] extend transition systems with continuous dynamics. A *Hybrid System* (HS) is a tuple $\langle V, X, \mathit{Init}, \mathit{Trans}, \mathit{Inv}, \mathit{Flow}\rangle$ where:

- V is the set of discrete variables,
- X is the set of continuous variables,
- Init is a $\Sigma_{\mathbb{R}}$-formula over $V \cup X$ (called the initial condition);
- Inv is a $\Sigma_{\mathbb{R}}$-formula over $V \cup X$ (called the invariant condition).

- $Trans$ is a $\Sigma_{\mathbb{R}}$-formula over $V \cup X \cup V' \cup X'$ (called the transition condition);
- $Flow$ is a $\Sigma_{\mathbb{R}}$-formula over $V \cup X \cup \dot{X}$ (called the flow condition).

Given a hybrid trace $\langle f_0, I_0\rangle, \langle f_1, I_1\rangle, \ldots, \langle f_k, I_k\rangle$, we denote with $s_{f_i}(t)$ the state assigning to every variable $v \in V \cup X$ the value $f_i(v)(t)$ and with $\dot{s}_{f_i}(t)$ the assignment that maps every variable $v \in X$ with the value $\dot{f}_i(v)(t)$.

A hybrid trace $\langle f_0, I_0\rangle, \langle f_1, I_1\rangle, \ldots, \langle f_k, I_k\rangle$ is a model (also called *path*) of the HS $S = \langle V, X, Init, Inv, Trans, Flow\rangle$ iff:

- $s_{f_0}(0)$ satisfies $Init$;
- for every $0 \leq i \leq k$, for all $t \in I_i$, $s_{f_i}(t)$ satisfies Inv;
- for every $0 \leq i < k$, if I_i is right closed with $ue(I_i) = t$ and I_{i+1} is left closed with $le(I_{i+1}) = t'$, then $s_{f_i}(t), s'_{f_{i+1}}(t')$ satisfies $Trans$;
- for every $0 \leq i \leq k$, for all $t \in I_i$, $s_{f_i}(t), \dot{s}_{f_i}(t)$ satisfy $Flow$.

The *language* $L(S)$ is the set of models of S.

Proposition 2: A sampling refinement of a path of an HS S is a path of S too.

Intuitively, sampling refinement just splits an interval into sub-intervals and therefore does not change either the initial state or the discrete transitions. Thus, the conditions remain satisfied by the corresponding points.

Sampling refinement preserves reachability properties in the sense that if $L' \preceq L(S)$ then there exists a trace in L' reaching a condition ϕ iff there exists a trace in $L(S)$ reaching ϕ (similarly for LTL properties without next operators [15] and HRELTL properties [14]).

Remark 1: In the above definition, the flow conditions are general predicates over the derivatives of X. In the following, we are considering HSs with continuous dynamics described by ODEs in form $\dot{X} = F(X)$ (i.e., for all $x \in X$, $\dot{x} = F_x(X)$).

D. Encoding of hybrid into transition systems

In this section, we show a standard encoding of HSs into a transition system with formulas over the reals. In general, for encoding, we mean a transition system that preserves the properties of interest. In this paper, we say that the transition system is an encoding of a HS if it represents its language or a sampling refinement thereof (thus preserving for example reachability).

In this encoding, we assume that the system of ODEs admits a primitive solution $f(V, t)$, which is uniquely determined by the state at the beginning of the timed transition. Moreover, we assume that the time intervals of the hybrid traces satisfying the HSs are all in the form either $[t_1, t_2)$ (i.e., left closed, right open) or $[t_1, t_1]$ (i.e., singular intervals). This simplifies the encoding but a more general encoding is possible (see for example [14]). Note also that the restriction does not affect the validity of Proposition 1, which regards only the open parts of the intervals.

Theorem 1: Given a HS S, there exists a TS S_D such that there exists a one-to-one mapping between the paths of S and the paths of S_D.

We call S_D the *encoding* of S.

Sketched proof: We encode a HS S in the TS $S_D = \langle V_D, Init_D, Inv_D, Trans_D\rangle$ where:

- $V_D := V \cup X \cup \{t\}$
  (t is a real variable that stores the current real time of the system).
- $Init_D := t = 0 \wedge Init$.
- $Inv_D := Inv$
  (note that this does not guarantee that the invariants of S hold for the entire duration of a continuous transition. This is taken into account in $Trans_D$).
- $Trans_D :=$ TIMED $\vee$ UNTIMED where
  - TIMED $:= t' > t \wedge V' = V \wedge X' = f(V \cup X, t') \wedge \forall \epsilon \in [t, t'], Inv(V, f(\epsilon))$
  - UNTIMED $:= t' = t \wedge Trans(V, X, V', X')$.

Let the hybrid trace $\langle f_0, I_0\rangle, \langle f_1, I_1\rangle, \ldots, \langle f_k, I_k\rangle$ be a path of S. Then, the sequence of states $f_0(le(I_0)), f_1(le(I_1)), \ldots, f_k(le(I_k))$ is a path of S_D.

Let the sequence $s_0, s_1, \ldots, s_k$ be a path of S_D. Let us consider, for all $i \in [1, k]$, $f_i(v)(t) = f(s_i, t)(v)$. Let us define $I_i := [s_i(t), s_{i+1}(t))$ if $i < k$ and $s_{i+1}(t) > s_i(t)$, $I_i := [s_i(t), s_i(t)]$ if $i < k$ and $s_{i+1}(t) = s_i(t)$ or if $i = k$. Then, the hybrid trace $\langle f_0, I_0\rangle, \langle f_1, I_1\rangle, \ldots, \langle f_k, I_k\rangle$ is a path of S. ■

Remark 2: Notice that the timed transition involves a quantified sub-formula to encode that the invariant holds along each instant of the continuous evolution. This is an issue for using standard SMT solvers which typically handle quantifier-free formulas or are not complete for quantifiers (even if the full theory with quantifiers is theoretically decidable). When the primitive solution is known and is expressed in the theory of reals (a polynomial), the quantifier can be removed to yield an equivalent quantifier-free encoding. However, in practice, this solution is not feasible beyond the quadratic case.

Remark 3: It is usually very useful to strictly alternate timed and discrete transitions to simplify the encoding and improve the search (see e.g. [1]). The encoding of Theorem 1 does not force such alternation, to enable other forms of simplification. In the following, we will clarify when we use alternation.

Hereafter, we assume that every universal quantifier occurs positively in $Trans_D$ and that it is in the form $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$ with $\bowtie \in \{<, \leq, >, \geq, =\}$. As shown in Appendix A, we can generalize the approach to generic formulas.

III. REMOVING QUANTIFIERS FROM THE INVARIANTS

A. Reduction to flow invariants

In this section we present the main theorem of the paper. The goal of the theorem is to reduce the quantified formula of an invariant to a quantified formula over its derivatives. In some cases, this simplifies the quantified formula.

The following theorems assume the finite variability of predicates of the derivatives. Many functions have this property, in particular polynomials and some simple transcendental functions.

Theorem 2: If $g : \mathbb{R} \rightarrow \mathbb{R}$ is a differentiable function and $\dot{g} \bowtie 0$ ($\bowtie \in \{\geq, >, \leq, <\}$) has finite variability, then $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$ iff there exists a finite sequence of real numbers $t = t_0 < \ldots < t_n = t'$ such that $\bigwedge_{0 \leq i \leq n} g(t_i) \bowtie 0 \wedge \bigwedge_{0 < i \leq n} \mathit{Constant}(\dot{g} \geq 0, t_{i-1}, t_i)$.

Proof: Let us assume that $\bowtie \in \{\geq, >\}$.

($\Rightarrow$) Since $\dot{g} \bowtie 0$ has finite variability, there exists a finite sequence of real numbers $t_0 = t < \ldots < t_n = t'$ such that $\bigwedge_{0 < i \leq n} \mathit{Constant}(\dot{g} \geq 0, t_{i-1}, t_i)$ by definition. Moreover, since $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$, $g \bowtie 0$ holds also in the time points $t_0, \ldots, t_n$.

($\Leftarrow$) Assume by contradiction that there exists $t_b \in [t, t']$ such that $g(t_b) \bowtie 0$ is false. Since $\bigwedge_{0 \leq i \leq n} g(t_i) \bowtie 0$, there exists $i \in [1, n]$ such that $t_b \in (t_{i-1}, t_i)$. Since g is differentiable, by the mean value theorem, there exists a point $t'_b \in (t_{i-1}, t_b)$ such that $\dot{g}(t'_b) = \frac{g(t_b) - g(t_{i-1})}{(t_b - t_{i-1})}$ and therefore $\dot{g}(t'_b) < 0$. Similarly, there exists a point $t''_b \in (t_b, t_i)$ such that $\dot{g}(t''_b) = \frac{g(t_i) - g(t_b)}{(t_i - t_b)}$ and therefore $\dot{g}(t''_b) > 0$. Thus, $\dot{g}$ is not constant over (t_{i-1}, t_i) contradicting the hypothesis. We conclude that $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$.

The cases in which $\bowtie \in \{\leq, <\}$ can be proved similarly. ■

When the predicate is an equality, the reduction is simpler.

Corollary 1: If $g : \mathbb{R} \rightarrow \mathbb{R}$ is a differentiable function and $\dot{g} = 0$ has finite variability, then $\forall \epsilon \in [t, t'], g(\epsilon) = 0$ iff $g(t) = 0 \wedge g(t') = 0 \wedge \forall \epsilon \in [t, t'], \dot{g}(\epsilon) = 0$.

The definition of *Constant*() contains quantified subformulas in the form $\forall \epsilon \in [t, t'], \dot{g} \bowtie 0$. Therefore, the reduction can be iterated trying to remove the quantifiers.

Theorem 2 can be used to simplify the encoding of the invariant of an HS. Let the invariant be in the form $g(X) \bowtie 0$ ($\bowtie \in \{\geq, \leq, >, <, =\}$). Let $f : \mathbb{R} \rightarrow \mathbb{R}^{|X|}$ be the solution of the flow condition. If f and g are differentiable functions and $\frac{d}{dt}(g \circ f) \bowtie 0$ has finite variability, then $\forall \epsilon \in [t, t'], g(f(\epsilon)) \bowtie 0$ iff there exists a finite sequence of real numbers $t_0 = t < \ldots < t_n = t'$ such that $\bigwedge_{0 \leq i \leq n} g(f(t_i)) \bowtie 0 \wedge \bigwedge_{0 < i \leq n} \mathit{Constant}(\frac{d}{dt}(g \circ f) \geq 0, t_{i-1}, t_i)$.

The geometrical interpretation of $\frac{d}{dt}(g \circ f)$ is the scalar product of the gradient of the curve g and the derivative vector $\dot{f}$: in fact, $\frac{d}{dt} g(f(t)) = \bigtriangledown g \cdot \dot{f}$ where $\bigtriangledown g = \langle \frac{\partial g}{\partial x_1}, \ldots, \frac{\partial g}{\partial x_n} \rangle$. Therefore, in the theorem, the condition of $\dot{g} \geq 0$ of being constant in the interval means that the function f is uniformly getting closer to (or farther from) the curve g in that interval.

As a side note, in the case of ODEs $\dot{X} = F(X)$, the new quantified formula $\forall \epsilon \in [t, t'], \frac{d}{dt}(g \circ f) \geq 0$ is equivalent to the invariant $\bigtriangledown g \cdot F \geq 0$. Thus, the reduction can be also applied without need of the primitive solutions.

In the case that the invariants are polynomial and the continuous variables are polynomial functions of time, the derivative will eventually reduce to zero.

B. Applications

1) Application to polynomial hybrid automata: We consider the class of HS where the invariants and the primitive solution of the ODEs are polynomial functions of time (see also [19]). The polynomial may contain some discrete variables as coefficients to account for uncertainties in the inputs, model parameters, etc. Note that several classes of HS with linear ODE can be expressed as a polynomial hybrid automaton. This is because the primitive solution to the ODEs can be expressed as a quantifier free formula in the theory of reals for several classes of linear systems [26][2].

Theorem 3: The invariant of a polynomial hybrid automata can be encoded with a quantifier-free formula.

Proof: In the case of polynomial hybrid automata, the invariant $g \bowtie 0$ is encoded into a formula in the form $\forall \epsilon \in [t, t'], g(f(\epsilon)) \bowtie 0$. If g and f are polynomials, $g \circ f$ is also a polynomial. The derivative of a polynomial has a lower degree than the polynomial itself. Thus, at every application of Theorem 2, the degree of the polynomial inside the quantifier strictly decreases. Thus, after a finite number of applications of the theorem, we obtain a quantifier-free formula. ■

Example 1: Let us consider the classical example of the bouncing ball. Suppose the ball moves in two dimensions x and y, where x is the horizontal coordinate, with $\dot{x} = v_0$, and y is the vertical coordinate, with $\dot{y} = w$ and $\dot{w} = -g$. Thus, the primitive solution is $x(t) = v_0 t + x_0$, $y(t) = -\frac{g}{2}t^2 + w_0 t + y_0$, and $w(t) = -gt + w_0$. Suppose the ball is bouncing on a parabolic hill, a curved surface with equation $y + ax^2 + bx + c = 0$. The invariant of the continuous transition is $y + ax^2 + bx + c \geq 0$ and its encoding is $\forall \epsilon \in [t, t'], y(\epsilon) + ax^2(\epsilon) + bx(\epsilon) + c \geq 0$, which is quadratic in ϵ. After applying the Theorem 2 twice, we obtain the following quantifier-free formula: $y(t) + ax^2(t) + bx(t) + c \geq 0 \wedge$
$y(t_1) + ax^2(t_1) + bx(t_1) + c \geq 0 \wedge$
$y(t') + ax^2(t') + bx(t') + c \geq 0 \wedge$
$((w(t) + 2av_0 x(t) + bv_0 \geq 0 \wedge w(t_1) + 2av_0 x(t_1) + bv_0 \geq 0) \vee$
$(w(t) + 2av_0 x(t) + bv_0 \leq 0 \wedge w(t_1) + 2av_0 x(t_1) + bv_0 \leq 0)) \wedge$
$((w(t_1) + 2av_0 x(t_1) + bv_0 \geq 0 \wedge w(t') + 2av_0 x(t') + bv_0 \geq 0) \vee$
$(w(t_1) + 2av_0 x(t_1) + bv_0 \leq 0 \wedge w(t') + 2av_0 x(t') + bv_0 \leq 0))$

2) Application to non-linear hybrid automata: In the general case of non-linear hybrid automata (here meant as hybrid systems with non-polynomial functions), the reduction of Theorem 2 may result in more complex quantified formulas. Even if we restrict to polynomial invariants, their composition with transcendental primitive solutions may yield complex derivatives. However, in many cases, we can convert the derived quantified formula into a polynomial which is simpler than the original[3].

Example 2: Let us consider a temperature controller. The system is parameterized by the lower and upper temperature limits m and M, the outside temperature u, the rate b of temperature exchanged with the outside, the rate c of temperature increase due to the heater. The constraints on the parameters

[2] In particular, given a linear system $\dot{X} = AX + BU$, the reachability problem can be expressed in the theory of reals if the matrix A has a particular structure: A is nilpotent, A is diagonalizable with all rational eigenvalues, A is diagonalizable with all imaginary eigenvalues. While in the first case obtaining a primitive solution in the theory of reals is straightforward also in the presence of symbolic coefficients of the matrix, the other two cases are more involved and require to perform several substitutions to remove exponential or trigonometric functions, which assume to have constant coefficient. We refer the reader to [26] for the details.

[3] This conversion is not currently automated.

are $u < m < M \wedge c > 0 \wedge b > 0$. The HS is defined as follows:

- $V = \{h\}$ where h is a variable representing the heater.
- $X = \{x\}$ where x represents the temperature.
- $Init := m \leq x \leq M$.
- $Inv := (h = 0 \rightarrow x \geq m) \wedge (h = c \rightarrow x \leq M)$.
- $Trans := (h = 0 \rightarrow (x = m \wedge h' = c)) \wedge (h = c \rightarrow (x = M \wedge h' = 0)) \wedge x' = x$.
- $Flow := \dot{x} = b(u - x) + h$.

The primitive of the ODE is $x(t) := u + \frac{(x(0)-u)}{b} e^{(-b*t)} + \frac{c}{b}$. Its derivative is $x(t) := -(x(0) - u)e^{(-b*t)}$, which never changes sign. Therefore, applying Theorem 2, $\forall \epsilon \in [t, t'], x \geq m$ is translated into the formula $x(t) \geq m \wedge x(t') \geq m$ and similarly for $\forall \epsilon \in [t, t'], x \leq M$.

Example 3: Consider the Traffic Collision Avoidance System (TCAS) example (cfr. e.g. [29]). The continuous dynamics of a safe circular maneuver is described by the following equations $\dot{x_1} = d_1, \dot{x_2} = d_2, \dot{d_1} = -\omega d_2, \dot{d_2} = \omega d_1, \dot{y_1} = e_1, \dot{y_2} = e_2, \dot{e_1} = -\rho e_2, \dot{e_2} = \rho e_1, (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$.

The primitive solution of the differential equations are:

$$\begin{aligned} x_1 = \frac{1}{\omega} sin(\theta),\ x_2 = -\frac{1}{\omega} cos(\theta), \\ d_1 = cos(\theta),\ d_2 = sin(\theta),\ \theta = \omega t + t_0, \\ y_1 = \frac{1}{\rho} sin(\xi),\ y_2 = -\frac{1}{\rho} cos(\xi), \\ e_1 = cos(\xi),\ e_2 = sin(\xi),\ \xi = \rho t + t_0 \end{aligned}$$

Substituting the primitive solution into the invariant $(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$ we obtain the formula:

$$\frac{1}{\omega^2} + \frac{1}{\rho^2} - \frac{2}{\omega\rho} sin(\theta) sin(\xi) - \frac{2}{\omega\rho} cos(\theta) cos(\xi) \geq p^2.$$

which can be rewritten into: $\phi := \frac{1}{\omega^2} + \frac{1}{\rho^2} - \frac{2}{\omega\rho} cos(\theta - \xi) \geq p^2$.

The standard quantified encoding is $\forall t \in [0, \delta], \phi(t)$. Applying Theorem 2, we obtain the formula:

$$\begin{aligned} \phi(0) \wedge \phi(\delta) \wedge \quad & (\forall t(-sin(\theta - \xi)(\omega - \rho) \geq 0) \vee \\ & \forall t(-sin(\theta - \xi)(\omega - \rho) \leq 0)). \end{aligned}$$

The quantified sub-formulas can be rewritten into polynomials over θ and ξ. For example, $\forall t(-sin(\theta - \xi)(\omega - \rho) \geq 0)$ can be rewritten into $\forall t(\omega - \xi \geq 0 \wedge (\pi \leq \theta - \rho \leq 2\pi) \vee \omega - \xi \leq 0 \wedge (0 \leq \theta - \rho \leq \pi))$. Since θ and ρ are linear, this can be converted into an equivalent quantifier-free one.

IV. Encoding Polynomial HS into Transition Systems

In this section, we show how Theorem 2 can be exploited to automatically encode a polynomial HS into a transition system with quantifier-free formulas.

A. Sequential encoding

Theorem 2 states the existence of the points $t_1, \ldots, t_n$ where the derivative changes sign. However, such points are unknown. The encoding of a HS into a transition system must thus implicitly represent when the derivative of the invariant changes sign. This is achieved by simply forcing that the sign of the derivative is constant throughout the timed transition. The encoding implicitly concatenates timed transitions one after the other, delegating to the search the task of finding the sequence of time points that split the interval, so that the sign of the derivative is uniformly constant in the resulting trace.

Given a formula T including the invariant condition $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$, the condition can be locally replaced with $g(t) \bowtie 0 \wedge g(t') \bowtie 0 \wedge \mathit{Constant}(\dot{g}, t, t')$ obtaining a new formula $\tau(T)$.

τ performs a recursive substitution of the quantified expressions. The recursion terminates when the quantified formula is a linear polynomial, thus allowing to trivially remove the quantifiers. τ is defined recursively as follows:

$$\begin{aligned} \tau(\psi_1 \wedge \psi_2) &:= \tau(\psi_1) \wedge \tau(\psi_2) \\ \tau(\psi_1 \vee \psi_2) &:= \tau(\psi_1) \vee \tau(\psi_2) \\ \tau(\neg\psi) &:= \neg\psi, (\psi \text{ is a predicate}) \\ \tau(\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0) &:= \begin{cases} g(t) \bowtie 0 \wedge g(t') \bowtie 0 & \text{if } g \text{ linear} \\ g(t) \bowtie 0 \wedge g(t') \bowtie 0 \wedge & \\ \tau(\mathit{Constant}(\dot{g}, t, t')) & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

The correctness of the transformation is given by the following theorem.

Theorem 4: If S_D is the encoding of the HS S and $\tau(S_D)$ is the transition system obtained by replacing $Trans$ with $\tau(Trans)$, then $\tau(S_D)$ is the encoding of a sampling refinement of S.

Proof: ($\Leftarrow$) If a sequence of states satisfies $\tau(S_D)$, then by Theorem 2, the sequence satisfies also S_D, and by Theorem 1, it represents a path of S. ($\Rightarrow$) Consider a hybrid trace $\langle f_0, I_0 \rangle, \langle f_1, I_1 \rangle, \ldots, \langle f_k, I_k \rangle$ which is a path of S. Assuming that $\dot{g}$ has finite variability, we can refine the hybrid trace into a new hybrid trace in which $\dot{g}$ is constant in every interval. The new hybrid trace also satisfies S by Theorem 2 and thus the corresponding discrete trace $s_0, \ldots, s_k$ satisfies its encoding S_D. At every i, if s_i satisfies $\forall \epsilon \in [t, t'], g(\epsilon) \bowtie 0$, then both $f(s_i, t)$ and $f(s_i, t')$ satisfy $g \bowtie 0$. Since $\dot{g}$ has constant sign in I_i, s_i satisfies also $\tau(Trans)$. Therefore the discrete trace satisfies also $\tau(S_D)$. ■

The recursive definition of τ in (1) creates a formula whose size is exponential in the degree of the polynomial inside the invariant. We use the following equivalence to keep the size of the encoding *linear* in the degree of the polynomial (here g is not linear):

$$\begin{aligned} \tau(\mathit{Constant}(g, t, t')) &= (g(t) \geq 0 \wedge g(t') \geq 0 \wedge \\ &\quad \tau(\mathit{Constant}(\dot{g}, t, t'))) \vee \\ &\quad (g(t) \leq 0 \wedge g(t') \leq 0 \wedge \\ &\quad \tau(\mathit{Constant}(\dot{g}, t, t'))) \\ &= ((g(t) \geq 0 \wedge g(t') \geq 0) \vee \\ &\quad (g(t) \leq 0 \wedge g(t') \leq 0)) \wedge \\ &\quad \tau(\mathit{Constant}(\dot{g}, t, t')) \end{aligned} \tag{2}$$

Another optimization that we implemented is the use of some *lemmas* that relate the value of polynomials to the value of their derivatives. More specifically, we optionally add to τ the following formulas:
$(\dot{g}(t) > 0 \vee \dot{g}(t') > 0) \rightarrow$
$((g(t) \geq 0 \rightarrow g(t') \geq 0) \wedge (g(t') \leq 0 \rightarrow g(t) \leq 0)) \wedge$
$(\dot{g}(t) < 0 \vee \dot{g}(t') < 0) \rightarrow$
$((g(t) \leq 0 \rightarrow g(t') \leq 0) \wedge (g(t') \geq 0 \rightarrow g(t) \geq 0))$

The formula means that when the derivative is positive g can only increase (thus cannot pass from positive to negative) and vice versa when $\dot{g}$ is negative g can only decrease (thus cannot pass from negative to positive).

B. Bound on required splitting

The sequential encoding may force the split of a continuous transition in several transitions, since the predicates introduced to remove the quantifiers forces the derivatives of the invariant conditions to be constant. While the encoding enables to remove the quantifier, the depth of the bounded model checking formula may increase due to the splitting. In incremental bounded model checking, the burden of finding how many splits are necessary is delegated to the search.

In the case of polynomial hybrid automata we can compute an upper bound on the number of consecutive continuous transitions (continuous transitions not separated by a discrete transition) needed to simulate the longest quantified continuous transition (the continuous transition with the maximum time elapse).

We can compute the upper bound on the number of intervals needed to "cover" the quantified continuous transition for the invariant predicate $\forall \epsilon \in [t, t'] g(\epsilon) \bowtie 0$. If $\Omega(g)$ is the degree of the polynomial, then the maximum number of intervals that have to be considered is $ub(g) = \frac{\Omega(g) * (\Omega(g) - 1)}{2}$. In fact, the i-th derivative of g has degree $\Omega(g) - i$ and thus changes sign $\Omega(g) - i$ times.

C. Layering

In the BMC settings we usually perform a search where we check if the target is violated for an increasing path length. In principle, the removal of the quantifiers requires more continuous transitions, thus increasing the size of the formula passed to solver. It is convenient to use a "layered" approach, where we first reach the target in an over-approximation of the HS, where invariants are not guaranteed to hold, and then we check if there exists a path that reaches the target and where invariants hold.

V. Related work

The quantifier-free encoding that we propose is related to quantifier elimination procedures (see, e.g., [16]). It is not a quantifier elimination procedure in that it contains new variables that are implicitly existentially quantified. In fact, we apply the reduction even in some cases of transcendental functions. The burden to remove the quantifiers is delegated to the verification techniques if necessary. We claim that quantifier elimination is somehow an overkill: the verification techniques does not often need the precise region of points where the invariant holds; it is usually sufficient either to pick some "good" values (in case of reachability) or to find "good" invariants (in case of safety verification).

Several works focus on the reachability problem for hybrid systems, but they use less expressive invariants or they restrict the class of the analyzed hybrid automata. We extend the bounded model checking encoding of linear hybrid automata [4], [1], where invariants hold iff they hold at the first and the last instant of a timed transition, thus the resulting encoding is quantifier free. Other approaches [10], [18], [23] focus on non-linear hybrid automata. In [10], the authors solve the reachability problem for non-linear convex hybrid automata. The restriction to convex invariant and linear flow conditions, or to monotonic invariant and convex flow, allows to easily encode the invariants without quantifiers. Many examples, including those mentioned in this paper, do not fall in this class of automata. In [18] the authors propose an SMT solver modulo ODEs, that can be used to perform bounded model checking on hybrid automata where the flow conditions are ODEs. The only allowed invariants are of the form $x \in [l, u]$, where x is a continuous variable and $l, u \in \mathbb{R}$. Their main focus is on the integration of numerical methods to compute the initial value problem for ODEs, while they cannot manage more complex invariants (e.g. linear functions). ODEs are also handled directly in [23]. This is done by computing the precise intersection of the continuous flow with the guards of the hybrid automaton. The solver can in principle handle invariants, but the authors state that the implementation is not mature enough to evaluate the approach. Approaches based on motion planning [28] do not encode symbolically the invariants, since they simulate the ODEs using numerical methods. In contrast, we encode a set of continuous transitions.

The prominent approaches to the verification of HSs are based either on the exploration of the reachable states or on deductive systems. We refer the readers to [2] for a recent survey. The focus of our work is on the SMT-based paradigm, which, although less mature, seems promising.

Our settings also differs from the works that build abstractions for HSs. The approaches described in [33], [31] use techniques based on the sign of derivatives such as ours. However, the purpose is different in that they generate over-approximations of the HS.

Finally, we mention the "clock translation" described in [21], where invariants are translated into constraints on time. However, the translation is restricted to monotonic flows (plus other restrictions on the independence of variables).

VI. Experimental evaluation

We applied our approach to several benchmarks of non-linear hybrid automata, obtaining a quantifier-free encoding. For the polynomial subcase we used the ETCS benchmark [22], an industrial case study of the braking control system of trains, the classic bouncing ball, and a simple ballistics example. For the bouncing ball, we used four variants: a ball moving vertically in one dimension and bouncing on a

	# vars	Max degree	RedLog	QepCAD
etcs_braking	4	2	0.14	0.05
ball_1d_plain	4	2	0.10	0.03
ball_2d_plain	4	2	0.10	0.03
ball_2d_hill	5	2	0.15	T.O. > 3600.00
ball_2d_slope	5	4	N.A.	T.O. > 3600.00
simple_ballistics	5	4	N.A.	T.O. > 3600.00

TABLE I
Results of applying quantifier elimination to the polynomial benchmarks (max degree is the maximum degree of the quantified variable,T.O.is a time out of 3600 seconds, N.A. means not applicable).

	quantifier-free encoding	qelim (qepcad)	qelim (redlog)
etcs_braking	66.75 / 17	161.52 / 17	168.16 / 17
ball_1d_plain.01	0.05 / 2	0.05 / 2	0.03 / 2
ball_1d_plain.02	25.50 / 6	0.09 / 4	0.06 / 4
ball_1d_plain.03	31.43 / 10	0.28 / 6	0.40 / 6
ball_1d_plain.04	36.23 / 14	0.46 / 8	0.65 / 8
ball_1d_plain.05	151.41 / 18	1.27 / 10	1.51 / 10
ball_2d_plain.01	0.08 / 2	0.18 / 2	0.28 / 2
ball_2d_plain.02	4.20 / 6	3.14 / 6	3.64 / 6
ball_2d_plain.03	16.04 / 10	15.90 / 10	62.64 / 10
ball_2d_hill.01	1.30 / 4	na / na	0.94 / 2
ball_2d_hill.02	118.67 / 8	na / na	15.36 / 4
ball_2d_slope.01	to / na	na / na	na / na
simple_ballistics	8.31 / 1	na / na	na / na

TABLE II
Results (running time / path length) of BMC with the different encodings.

plain floor, a two-dimensional variant with constant horizontal speed, a third variant still in two-dimensions but bouncing on a hill (vertical parabola), and a fourth variant bouncing on a slope (horizontal parabola). As for the ballistics example, we modeled an object that flies above an obstacle keeping below a certain ceiling. As for nonlinear benchmarks with transcendental functions, we used the temperature controller, the TCAS benchmark and the steering car mentioned in Sec. III-B2. All the benchmarks are publicly available at http://es.fbk.eu/people/mover/tests/FMCAD12/.

The techniques discussed in the previous sections have been implemented in an extension of NuSMV[4], which is able to deal with HSs formalized in the HyDI language [12]. The NuSMV extension features an SMT-based approach to the verification of HSs, including bounded model checking and inductive reasoning. We automatically encode the invariants for polynomial hybrid automata, while we manually encode the invariants for the other benchmarks. iSAT[5] is used as the backend to solve the resulting satisfiability queries.

We evaluated the alternative use of quantifier elimination procedures, within their range of applicability, i.e. polynomial hybrid automata. We experimented with Cylindrical Algebraic Decomposition (CAD) (using QepCAD[6]) and Virtual Substitution (using RedLog[7]). Table I reports, for each polynomial benchmark, the time needed to obtain a quantifier free formula of the invariants using QepCAD and RedLog. The Virtual Substitution approach of RedLog can only handle formulas quantified over a quadratic variable. QepCAD is slightly more general, but de facto less useful: the results highlight the dramatic computational complexity of the procedure (e.g. ball_2d_hill, with 5 variables, times out in one hour). Thus, the quantifier elimination approach cannot even handle the polynomial benchmarks ballistic and ball_2d_slope (in addition to the benchmarks with transcendental functions).

We used the bounded model checking functionalities enabled by our approach to validate the various models and to evaluate the performance of the invariant encoding. For each model we generated different reachability properties which are falsified by traces with an increasing length. We evaluated the encoding of the invariant by comparing the time needed to find these traces with BMC. When quantifier elimination was able to produce a result, we also compared it with our approach using the same SMT-based technique, in order to evaluate the overhead caused by the splitting. The results are shown in Table II. The encoding time of our approach is instantaneous in all cases. In the cases where quantifier elimination is feasible, the resulting encoding may induce traces with a smaller number of steps, because timed transitions must not be split. This happens for the *ball_1d_plain* and the *ball_2d_hill* benchmarks. The reduced number of steps also reduces the time needed to generate the trace.

Our approach was also able to prove a simple invariant on the ballistics example, that was beyond the applicability of SMT-based techniques. We chose as obstacle a circle shape with center in $(c, 0)$ and radius r. If the ceiling level is less than r, the object cannot clearly pass. This has been proved with NuSMV and iSAT. Ignoring the invariant along the timed transitions (keeping it only on the discrete points) allows for spurious traces that forbid the inductive proof. Note that this small example is beyond the applicability of quantifier elimination (see Table I).

Some remarks are in order. Our approach strongly depends on the availability of SMT solvers for quantifier-free theories of nonlinear arithmetic, to solve the formulas resulting from our SMT-based verification engines. To this end, we tried to use all the available solvers for nonlinear arithmetic: Z3[8], SMT-RAT[9], CVC3[10], miniSMT[11], RAHD[12], hydlogic[13], dReal[14]. and iSAT[15]. Z3 and SMT-RAT implement two complete decision procedures for the non-linear arithmetic over reals. Both solvers still do not integrate a layering with the linear arithmetic solver: in this case all the linear arithmetic constraints are handled using the non-linear solver,

[4] http://nusmv.fbk.eu/

[5] http://isat.gforge.avacs.org/

[6] http://www.usna.edu/cs/ qepcad/B/QEPCAD.html

[7] http://redlog.dolzmann.de/

[8] http://research.microsoft.com/en-us/um/redmond/projects/z3/

[9] http://smtrat.sourceforge.net/

[10] http://cs.nyu.edu/acsys/cvc3/

[11] http://cl-informatik.uibk.ac.at/software/minismt/

[12] http://homepages.inf.ed.ac.uk/s0793114/rahd/

[13] http://code.google.com/p/hydlogic/

[14] http://www.cs.cmu.edu/ sicung/dReal/

[15] http://isat.gforge.avacs.org/

thus resulting in an inefficient approach. This is the case for our BMC case studies, which have a significant part of linear constraints. Instead, CVC3 and miniSMT implement an incomplete decision procedure for non-linear arithmetic (and miniSMT is tailored only to check satisfiable formulas). As a result, these solvers turned out to return "unknown" on most of the queries generated from our benchmarks. The hydlogic system turned out to be immature, while RAHD exports functionalities that are closer to a theory solver than a full SMT solver, requiring an explicit treatment of disjunctions. iSAT and dReal differ from the other solvers, since they can also provide non-precise solutions. dReal returns an unsatisfiable answer or a satisfiable answer if the formula is satisfiable under a bounded numerical perturbations. iSAT may return "unknown" exposing the results of interval constraints propagation: it produces the intervals found in the search, if these are below a user-defined threshold, as a candidate solution. In many practical cases, this is not spurious, and represents a satisfying assignment of the formula.

Overall, despite some recent progress, our experience has shown that the field still requires additional research to deliver what our approach requires, both in terms of completeness, and performance. However, we argue that our method is valuable regardless of the current status of SMT for nonlinear arithmetic. First, we proposed a solution to a problem that was a show-stopper for SMT-based verification. In fact, we are now able to solve some benchmarks that cannot be solved by overapproximation, just forgetting about the quantified invariants. Second, we are hopeful that the field of SMT can deliver quick progress in quantifier-free nonlinear arithmetic. In fact, the development of SMT solving for non-linear arithmetic has been influenced by benchmarks from other domains (e.g. most of the SMT-LIB benchmarks in NRA are from the software domain). To this extent, we generated and submitted to the SMT-LIB a vast number of benchmarks, that will trigger additional research in practically relevant directions.

VII. Conclusions

In this paper, we tackled the problem of dealing with invariant constraints in non-linear hybrid automata in the setting of SMT-based verification. This is largely an open problem, due to the presence of the universal quantifiers required to encode that the invariant must hold throughout all time instants in delay transitions.

We proposed new methods that allow for the reduction to quantifier-free theories, at the cost of introducing additional variables. Our approach is comprehensive (deals with disjunctive invariants), encompasses a large class of hybrid systems (nonlinear polynomials), and is open to new patterns of reduction, when an algorithmic solution is not possible in general. As a result, we extend the applicability of SMT-based verification methods, and were able to verify some novel benchmark problems.

In the future, we plan to proceed along the following directions. We will experiment with the application of the proposed methods as a way to concretize the abstract paths. Then, we will generalize the approach to the analysis of networks of hybrid automata; in particular, we will exploit the locality of the splits of the continuous transitions in the local time semantics framework. We will also apply a layered approach to the analysis of non-linear constraints, where less expensive (e.g. linear) solvers are applied whenever possible before resorting to expensive but more precise nonlinear solvers such as RAHD. Finally, we will apply the proposed techniques to the analysis of requirements expressed in HRELTL logic [14]. In fact, HRELTL requires the predicates to be constant in arbitrary intervals of time.

References

[1] E. Ábrahám, B. Becker, F. Klaedtke, and M. Steffen. Optimizing Bounded Model Checking for Linear Hybrid Systems. In *VMCAI*, pages 396–412, 2005.

[2] R. Alur. Formal verification of hybrid systems. In *EMSOFT*, pages 273–278, 2011.

[3] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

[4] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying Industrial Hybrid Systems with MathSAT. *ENTCS*, 119(2):17–32, 2005.

[5] C.W. Barrett, R. Sebastiani, S.A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, pages 825–885. 2009.

[6] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *TACAS*, pages 193–207, 1999.

[7] M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll, and M. Roveri. Safety, Dependability and Performance Analysis of Extended AADL Models. *Comput. J.*, 54(5):754–775, 2011.

[8] M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll, M. Roveri, and R. Wimmer. A Model Checker for AADL. In *CAV*, pages 562–565, 2010.

[9] L. Bu, A. Cimatti, X. Li, S. Mover, and S. Tonetta. Model Checking of Hybrid Systems using Shallow Synchronization. In *FORTE*, 2010.

[10] L. Bu, J. Zhao, and X. Li. Path-Oriented Reachability Verification of a Class of Nonlinear Hybrid Automata Using Convex Programming. In *VMCAI*, pages 78–94, 2010.

[11] A. Cimatti, S. Mover, and S. Tonetta. Efficient Scenario Verification for Hybrid Automata. In *CAV*, pages 317–332, 2011.

[12] A. Cimatti, S. Mover, and S. Tonetta. HyDI: a language for symbolic hybrid systems with discrete interaction. In *EUROMICRO-SEAA*, 2011.

[13] A. Cimatti, S. Mover, and S. Tonetta. Proving and Explaining the Unfeasibility of Message Sequence Charts for Hybrid Systems. In *FMCAD*, 2011.

[14] A. Cimatti, M. Roveri, and S. Tonetta. Requirements Validation for Hybrid Systems. In *CAV*, pages 188–203, 2009.

[15] L. de Alfaro and Z. Manna. Verification in Continuous Time by Discrete Reasoning. In *AMAST*, pages 292–306, 1995.

[16] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. In *Alg. Algebra and Number Theory*, pages 221–247. Springer, 1998.

[17] A. Eggers, M. Fränzle, and C. Herde. SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems. In *ATVA*, pages 171–185, 2008.

[18] A. Eggers, N. Ramdani, N. Nedialkov, and M. Fränzle. Improving SAT Modulo ODE for Hybrid Systems Analysis by Combining Different Enclosure Methods. In *SEFM*, pages 172–187, 2011.

[19] M. Fränzle. What Will Be Eventually True of Polynomial Hybrid Automata? In *TACS*, pages 340–359, 2001.

[20] S. Graf and H. Saïdi. Construction of Abstract State Graphs with PVS. In *CAV*, pages 72–83, 1997.

[21] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic Analysis of Nonlinear Hybrid Systems. 1998.

[22] C. Herde, A. Eggers, M. Fränzle, and T. Teige. Analysis of Hybrid Systems Using HySAT. In *ICONS*, pages 196–201, 2008.

[23] D. Ishii, K. Ueda, and H. Hosobe. An interval-based SAT modulo ODE solver for model checking nonlinear hybrid systems. *STTT*, 13(5):449–461, 2011.

[24] S. Jha, B. A. Brady, and S. A. Seshia. Symbolic Reachability Analysis of Lazy Linear Hybrid Automata. In *FORMATS*, pages 241–256, 2007.

[25] T. King and C. Barrett. Exploring and Categorizing Error Spaces using BMC and SMT. In *SMT*, 2011.

[26] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[27] K.L. McMillan. Interpolation and SAT-Based Model Checking. In *CAV*, pages 1–13, 2003.

[28] E. Plaku, L.E. Kavraki, and M.Y. Vardi. Hybrid systems: from verification to falsification by combining motion planning and discrete search. *Formal Methods in System Design*, 34(2):157–182, 2009.

[29] A. Platzer and E.M. Clarke. Formal Verification of Curved Flight Collision Avoidance Maneuvers: A Case Study. In *FM*, pages 547–562, 2009.

[30] A.M. Rabinovich. On the Decidability of Continuous Time Specification Formalisms. *J. Log. Comput.*, 8(5):669–678, 1998.

[31] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *CAV*, pages 686–702, 2011.

[32] M. Sheeran, S. Singh, and G. Stålmarck. Checking Safety Properties Using Induction and a SAT-Solver. In *FMCAD*, pages 108–125, 2000.

[33] A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1):57–83, 2008.

[34] Y. Yushtein, M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, Th. Noll, X. Olive, and M. Roveri. System-software co-engineering: Dependability and safety perspective. In *SMC-IT*, pages 18–25. IEEE CS Press, 2011.

APPENDIX

In this section, we explain how the method can be extended to handle generic predicates for the invariants (i.e. disjunctive invariants and open intervals). We describe the extension only in the appendix because, first, it complicates the presentation, second, in practice we do not have disjunctive invariants in the benchmarks.

A. Encoding of hybrid into transition systems

We modify the encoding of a HS into a transition system with atomic quantified formulas. First, we modify the encoding considering quantification over open intervals instead of closed intervals. Later, we will prove that we can push the quantification inside the disjunctions under the assumption of finite variability.

We redefine $S_D = \langle V_D, Init_D, Inv_D, Trans_D \rangle$ as follows:

- $V_D := V \cup X \cup \{t\}$
  (t is a real variable that stores the current real time of the system).
- $Init_D := t = 0 \wedge Init$.
- $Inv_D := Inv$.
- $Trans_D :=$ TIMED $\vee$ UNTIMED
  where
  - TIMED $:= t' > t \wedge V' = V \wedge X' = f(V \cup X, t') \wedge \forall \epsilon \in (t, t'), Inv(V, f(\epsilon))$
  - UNTIMED $:= t' = t \wedge Trans(V, X, V', X')$.

Theorem 5: There exists a one-to-one mapping between the paths of S and the paths of S_D.

We call S_D the *encoding* of S.

Sketched proof:

Let the hybrid trace $\langle f_0, I_0 \rangle, \langle f_1, I_1 \rangle, \ldots, \langle f_k, I_k \rangle$ be a path of S. Then, the sequence of states $f_0(le(I_0)), f_1(le(I_1), \ldots, f_k(le(I_k)$ is a path of S_D.

Let the sequence $s_0, s_1, \ldots, s_k$ be a path of S_D. Let us consider, for all $i \in [1, k]$, $f_i(v)(t) = f(s_i, t)(v)$. Let us define $I_i := [s_i(t), s_{i+1}(t))$ if $i < k$ and $s_{i+1}(t) > s_i(t)$, $I_i := [s_i(t), s_i(t)]$ if $i < k$ and $s_{i+1} = s_i(t)$ or if $i = k$. Then, the hybrid trace $\langle f_0, I_0 \rangle, \langle f_1, I_1 \rangle, \ldots, \langle f_k, I_k \rangle$ is a path of S. ■

Without loss of generality we can assume that the quantified formula in $Trans$ is atomic. This is not correct in general and exploit the particular position of the quantified sub-formula in the transition condition.

Theorem 6: Assuming that the predicates ϕ and ψ have finite variability, if we replace a formula $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \psi(\epsilon)$ with $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \forall \epsilon \in (t, t'), \psi(\epsilon)$ inside $Trans_D$, we obtain the encoding of a sampling refinement to the original HS.

Proof: Clearly, $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \forall \epsilon \in (t, t'), \psi(\epsilon)$ implies $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \psi(\epsilon)$. The opposite does not hold in general. However, consider a hybrid trace $\langle f_0, I_0 \rangle, \langle f_1, I_1 \rangle, \ldots, \langle f_k, I_k \rangle$ which is a path of a HS S. Assuming that the predicates ϕ and ψ have finite variability, we can refine the hybrid trace into a new hybrid trace in which ϕ and ψ are constant in every interval. The new hybrid trace also satisfies S by Proposition 2 and thus the corresponding discrete trace $s_0, \ldots, s_k$ satisfies its encoding S_D. At every i, if s_i satisfies $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \psi(\epsilon)$, then $f(s_i, \epsilon)$ satisfies $\phi \vee \psi$ for all $\epsilon \in (t, t') = (le(I_i), ue(I_i))$, and thus either ϕ or ψ (since ϕ and ψ are constant in the open part of I_i). Therefore the discrete trace satisfies also the encoding with $\forall \epsilon \in (t, t'), \phi(\epsilon) \vee \forall \epsilon \in (t, t'), \psi(\epsilon)$. ■

B. Reduction to flow invariants

Theorem 7: If $g : \mathbb{R} \rightarrow \mathbb{R}$ is a differentiable function and $\dot{g} \bowtie 0$ ($\bowtie \in \{\geq, >\}$) has finite variability, then $\forall \epsilon \in (t, t'), g \bowtie 0$ iff there exists a finite set of real numbers $t = t_0 < \ldots < t_n = t'$ such that $g(t) \geq 0 \wedge g(t') \geq 0 \wedge \bigwedge_{0<i<n} g(t_i) \bowtie 0 \wedge \bigwedge_{0<i\leq n} \mathit{Constant}(\dot{g} \geq 0, t_{i-1}, t_i)$ if $\bowtie = \geq$, $g(t) \geq 0 \wedge g(t') \geq 0 \wedge \bigwedge_{0<i<n} g(t_i) \bowtie 0 \wedge \bigwedge_{0<i\leq n} \mathit{Constant}(\dot{g} \geq 0, t_{i-1}, t_i) \wedge (g(t) = 0 \rightarrow \dot{g}(t) > 0) \wedge (g(t') = 0 \rightarrow \dot{g}(t) < 0)$, if $\bowtie = >$.

Proof: ($\Rightarrow$) Since $\dot{g} \bowtie 0$ has finite variability, there exists a finite set of real numbers $t = t_0 < \ldots < t_n = t'$ such that $\bigwedge_{0<i\leq n} \mathit{Constant}(\dot{g} \geq 0, t_{i-1}, t_i)$ by definition. Moreover, since $\forall \epsilon \in (t, t'), g \bowtie 0$, $g \bowtie 0$ holds also in the time points $t_1, \ldots, t_{n-1}$. $g(t) \geq 0$ and $g(t') \geq 0$ for the continuity of g. Finally, $(g(t) = 0 \rightarrow \dot{g}(t) > 0) \wedge (g(t') = 0 \rightarrow \dot{g}(t) < 0)$, if $\bowtie = >$.

($\Leftarrow$) Assume by contradiction that there exists $t_b \in (t, t')$ such that $g(t_b) \bowtie 0$ is false. Since $\bigwedge_{0<i<n} g(t_i) \bowtie 0$, there exists $i \in [1, n]$ such that $t_b \in (t_{i-1}, t_i)$. Let us consider first the case in which $g(t) \bowtie 0$ and $g(t') \bowtie 0$ or $i \in [2, n-1]$. Since g is differentiable, for the mean value theorem, there exists a point $t'_b \in (t_{i-1}, t_b)$ such that $\dot{g}(t'_b) = \frac{g(t_b) - g(t_{i-1})}{(t_b - t_{i-1})}$ and therefore $\dot{g}(t'_b) \bowtie 0$ is false. Similarly, there exists a point $t''_b \in (t_b, t_i)$ such that $\dot{g}(t''_b) = \frac{g(t_i) - g(t_b)}{(t_i - t_b)}$ and therefore $\dot{g}(t''_b) \bowtie 0$ is true. Thus, $\dot{g}$ is not constant over (t_{i-1}, t_i) contradicting the hypothesis. Let us now consider the case in which $\bowtie = >$, $i = 1$ and $g(t) = 0$ or $i = n$ and $g(t') = 0$. Similarly as before there exists a point that contradicts $(g(t) = 0 \rightarrow \dot{g}(t) > 0) \wedge (g(t') = 0 \rightarrow \dot{g}(t) < 0)$.

We conclude that $\forall \epsilon \in (t, t'), g(\epsilon) \bowtie 0$. ■