Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Bounded Model Checking of Hybrid Dynamical Systems

Nicolò Giorgetti, George J. Pappas and Alberto Bemporad

Abstract— **Bounded model checking (BMC) has recently emerged as a very powerful methodology for the verification of purely discrete systems. Given a horizon of interest, bounded model checking verifies whether all finite-horizon trajectories satisfy a temporal logic formula by first translating the problem to a large satisfiability SAT-problem and then relying on extremely powerful state-of-the art SAT-solvers for a counterexample or a certification of safety. In this paper we consider the problem of bounded model checking for a general class of discrete-time hybrid systems. Critical to our approach is the abstraction of continuous trajectories under discrete observations with a purely discrete system that captures the same discrete sequences. Bounded model checking can then be applied to the purely discrete, abstracted system. The performance of our approach is illustrated by verifying temporal properties of a hybrid model of an electronic height controller.**

I. Introduction

In the quest of efficient automatic techniques for verification of purely discrete systems bounded model checking (BMC) has received great interest both from academia and industry [1]. This interest is mainly motivated by its efficiency to check safety temporal properties on a finite horizon, which in several cases outperforms other existing model checking techniques [1]. The efficiency of BMC relies on the capability of unfolding the discrete dynamics into a very big propositional formula and the use of very efficient satisfiability (SAT) solvers [2] to verify temporal properties.

However, in many real applications dynamical systems are hybrid in nature, exhibiting both continuous and discrete dynamics. Hybrid systems provide a unified framework for describing processes evolving according to continuous dynamics, discrete dynamics, and logic rules [3]. The interest in hybrid systems is mainly motivated by the large variety of practical situations, for instance real-time systems, where physical processes interact with digital controllers.

In this paper we introduce BMC to hybrid systems. There is limited but recent literature on this topic [4], [5]. In [4] and [5] BMC has been used to check safety properties of linear hybrid automata by using specialized SAT solvers which exploit the structure of the linear hybrid automata. These approaches are based on the particular structure of the hybrid system, in particular on the continuous dynamics, and on special BMC solvers for exploiting this structure. Following a different route, in this paper we want to consider quite general hybrid dynamics and exploit discrete abstractions to get a simpler and purely discrete representation of the hybrid system.

The goal of abstraction is to obtain a simpler description of the dynamics which preserves the properties being analyzed while hiding the details that are of no interest. Techniques for abstracting some classes of hybrid systems are presented in [6]. A technique to abstract continuous dynamics to discrete systems has been proposed in [7].

Several modeling formalisms have been developed to describe hybrid systems [8], among them the class of discrete-time hybrid automata (DHA) [8], a general modelling framework directly derived from hybrid automata [9], where discrete-time switched continuous dynamics are considered. Examples of real-world applications that can be naturally modeled within the DHA framework are reported in [8].

We propose an algorithm to derive a finite-time discrete transition system from the DHA dynamics which simulates the discrete behavior of the continuous dynamics. The composition of the discrete transition system with the discrete dynamics represents a purely discrete representation of the hybrid system and can be used by BMC for checking temporal properties. The algorithm is based on a recursive procedure which involves the alternation of a SAT solver and of computational geometry techniques.

The paper is organized as follows. Section II gives a brief presentation of BMC and properties expressible as linear temporal logic (LTL). Section III introduces the description of hybrid systems and motivates the reason of abstraction. Section IV presents the abstraction procedure and Section V shows on an industrial application the performance of the algorithm to check some temporal properties. Section VI summarizes the results presented in the paper and describes ongoing research.

II. Bounded Model Checking

Given a discrete model of hardware or software systems, and a desired specification expressed as a temporal logic formula, model checking algorithms attempt to verify that the system satisfies the formula, or provide a counterexample. Very recently, bounded model checking (BMC) has emerged as a novel technique to formally verify discrete systems [1]. Bounded model checking verifies that all trajectories of the discrete system for some pre-specified horizon N do satisfy the desired specification. The finite horizon assumption allows one to unfold the dynamics of the transition system to a large satisfiability (SAT) problem. Then very powerful

This work has been supported by the HYCON Network of Excellence, contract number FP6-IST-511368, and NSF Information Technology Research Grant 0121431.

N. Giorgetti and A. Bemporad are with the Dipartimento di Ingegneria dell'Informazione, University of Siena, via Roma 56, Siena, Italy {giorgetti, bemporad}@dii.unisi.it

G.J. Pappas is with the Dept. of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA pappasg@seas.upenn.edu

0-7803-9568-9/05/$20.00 ©2005 IEEE

satisfiability (SAT) solvers [10] are used to verify safety properties for systems with very large state spaces.

We briefly review bounded model checking for discrete systems. A discrete system can be described as a transition system $T_D = (Q, Q^0, \Sigma, \rightarrow, O, M)$ which consists of

- a (possibly infinite) set Q of states,
- a (possibly infinite) set $Q^0 \subseteq Q$ of initial states,
- a finite set $\Sigma = \{\sigma_1, \ldots, \sigma_l\}$ of labels (or events),
- a transition relation $\rightarrow \subseteq Q \times \Sigma \times Q$,
- a finite set $O = \{o_1, o_2, \ldots, o_p\}$ of observations,
- a map $M : Q \rightarrow 2^O$ that associates at each state $q \in Q$ a set of observations $M(q) \subseteq O$.

We say that T_D is *finite* when Q is finite, and *infinite* otherwise. Let q_t be the state of the transition system at time $t \in \mathbb{N}$. A *trajectory* $q[t]$ is defined as the infinite sequence $q_t \overset{\sigma_t}{\rightarrow} q_{t+1} \overset{\sigma_{t+1}}{\rightarrow} q_{t+2} \overset{\sigma_{t+2}}{\rightarrow} \ldots$, originating from $q_t \in Q$. A k-length trajectory $q[t]^k$ is defined as the finite sequence $q_t \overset{\sigma_t}{\rightarrow} q_{t+1} \overset{\sigma_{t+1}}{\rightarrow} q_{t+2} \overset{\sigma_{t+2}}{\rightarrow} \ldots \overset{\sigma_{t+k-1}}{\rightarrow} q_{t+k}$ originating at q_t.

For transition systems T_D, we are interested in verifying properties that are expressible in linear temporal logic (LTL) whose syntax and semantics are now defined.

LTL Syntax: The atomic propositions π of LTL are captured by the observation symbols of O of T_D. Based on these atomic symbols, the LTL formulas are defined according to the following grammar:

$$\phi \quad ::= \quad \pi \mid \neg\phi \mid \phi \vee \phi \mid \bigcirc \phi \mid \phi \, \mathcal{U} \phi$$

As usual, the Boolean constants $\top$ and $\bot$ are defined as $\top = \pi \vee \neg\pi$ and $\bot = \neg\top$ respectively. Given negation ($\neg$) and disjunction ($\vee$), we can define conjunction ($\wedge$), implication ($\Rightarrow$), and equivalence ($\Leftrightarrow$). Furthermore, we can also derive additional temporal operators such as eventuality $\Diamond\phi = \top\mathcal{U}\phi$ and safety $\Box\phi = \neg\Diamond\neg\phi$.

LTL Semantics: LTL formulas are interpreted over all trajectories of the transition system starting from initial states $q_0 \in Q^0$. Given an LTL formula ϕ, and any trajectory $q[t]$, we say that $q[t] \models \phi$ denotes the satisfaction of the formula ϕ over the trajectory $q[t]$ starting at q_t. The semantics of any formula can be recursively defined as:

- $q[t] \models \pi$ iff $M(q_t) = \pi$
- $q[t] \models \neg\phi$ if $q[t] \not\models \phi$
- $q[t] \models \phi_1 \vee \phi_2$ if $q[t] \models \phi_1$ or $q[t] \models \phi_2$
- $q[t] \models \bigcirc\phi$ if $q[t+1] \models \phi$
- $q[t] \models \phi_1 \mathcal{U} \phi_2$ if there exists $s \geq t$ such that $q[s] \models \phi_2$ and for all s' with $t \leq s' < s$ we have $q[s'] \models \phi_1$

Therefore, the formula $\phi_1 \mathcal{U} \phi_2$ intuitively expresses the property that over the trajectory $q[t]$ ϕ_1 is true until ϕ_2 becomes true. Formula $\Diamond\phi$ indicates that over the trajectory the formula ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true over the trajectory $q[t]$ for all time $t' \geq t$. The transition system satisfies the property, that is $T_D \models \phi$, if for all $q_0 \in Q^0$ we have that $q[0] \models \phi$.

Given a finite transition system T_D, and an LTL formula ϕ, BMC checks whether ϕ is true in T_D by looking for a counterexample (i.e., a witness to the violation of ϕ) on a k-length trajectory $q[t]^k$. Given a finite transition system T_D, a finite horizon k, and an LTL specification formula ϕ, for example $\phi = \Box\psi(q)$, where ψ is a boolean formula on the states q, the dynamics of the system and the LTL specification can be unfolded to the following propositional formula,

$$[\![T_D, \phi]\!]_k = \mathcal{I}(q_0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(q_i, \sigma_i, q_{i+1}) \wedge \bigvee_{i=0}^{k} \neg\psi(q_i), \quad (1)$$

where $\mathcal{I}(q_0)$ is a propositional formula representing the initial conditions, and $\mathcal{R}(q_i, \sigma_i, q_{i+1})$ is a representation (in propositional form) of the transition relation from q_i to state q_{i+1} under event σ_i. If $[\![T_D, \phi]\!]_k$ is satisfiable, then the propositional model provides a counterexample to the desired specification (of horizon no larger than k). If $[\![T_D, \phi]\!]_k$ is unsatisfiable, then no counterexamples of length up to k are possible. In general, nothing can be said about the existence of counterexamples to $T_D \models \phi$ with larger horizons. The typical approach is to generate and solve $[\![T_D, \phi]\!]_k$ for increasing values of k, until either a counter-example is found, a given time-out is reached, or a completeness threshold is reached[1].

The bounded model checking approach described above relies heavily on the use of extremely powerful SAT solvers (e.g., zCHAFF [2]) for checking the satisfiability of $[\![T_D, \phi]\!]_k$.

III. Hybrid Dynamical Systems

The goal of this paper is to apply the framework and algorithms of bounded model checking to the class of discrete-time hybrid systems described in this section. A hybrid dynamical system can be modeled as the interconnection of logic dynamics with continuous dynamics, as shown in Figure 1(a). Many representations have been introduced to describe hybrid systems [8], [9]. In this paper we consider discrete-time hybrid automata (DHA) [8], where the continuous dynamics is represented by the switched affine dynamics (SAS)

$$x(t+1) = A_{q(t)} x(t) + B_{q(t)} u(t) + f_{q(t)}, \quad (2)$$

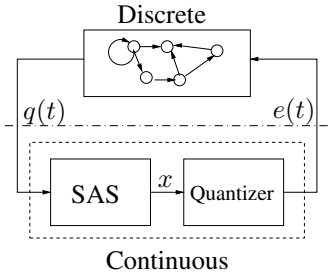
where $t \in \mathbb{Z}^+ = \{0, 1, \ldots\}$ is the time index, $q(t) \in \mathcal{Q} \subseteq \{0,1\}^{n_d}$ is the state of the discrete dynamics, $|\mathcal{Q}|$ the number of elements in $\mathcal{Q}$, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state defined on a polytope $\mathcal{X}$, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is a continuous exogenous input defined on a polytope $\mathcal{U}$, and $\{A_q, B_q, f_q\}_{q \in \mathcal{Q}}$ is a set of matrices of suitable dimensions.

The continuous dynamics generates through a quantizer a binary event signal $e(t) \in \mathcal{E} \subseteq \{0,1\}^{n_e}$. The quantizer is described by the following thresholds conditions

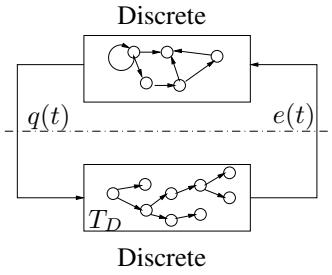
$$e_j(t) = \begin{cases} 1 & \text{if } C^j x(t) \leq D^j \\ 0 & \text{otherwise} \end{cases}, \ \forall j = 1, \ldots, n_e, \quad (3)$$

where C^j is a row vector of suitable dimensions and D^j is a scalar.

[1]A completeness threshold is a horizon k^* such that if the system is safe for horizon k^* then it is safe for the infinite horizon. The search for completeness thresholds for various LTL formulas is currently an active research topic.



(a) Hybrid dynamical system



(b) Equivalent purely discrete system

Fig. 1. Hybrid dynamical system and its discrete abstraction

The discrete dynamics is modeled as an automaton (or finite state machine). The automaton evolves according to the following logic state update function

$$q(t+1) = f_d(q(t), e(t)), \tag{4}$$

where $q \in \mathcal{Q} \subseteq \{0,1\}^{n_d}$ is the discrete state, $e \in \mathcal{E} \subseteq \{0,1\}^{n_e}$ is the endogenous input coming from the quantizer (3), and $f_d : \mathcal{Q} \times \mathcal{E} \to \mathcal{Q}$ is a deterministic Boolean function. In the sequel, with a slight abuse of notation, we will refer to the codomain of Boolean functions both as $\{0,1\}$ and as $\{\bot, \top\}$. In the context of Boolean functions and formulas, the equal sign $(=)$ should be interpreted as an equivalence condition $(\Leftrightarrow)$.

Given the initial condition $[x(0)'\ q(0)']' \in \mathcal{X}^0 \times \mathcal{Q}^0 \subseteq \mathcal{X} \times \mathcal{Q}$, and the input $u(t) \in \mathcal{U}$, $t \in \mathbb{Z}^+$, the state trajectory $x(t)$ of the system is recursively computed as follows:

1) Initialization: $[x(0)'\ q(0)']'$;
2) Recursion: Compute $e(t)$ with (3), $x(t+1)$ with (2), and $q(t+1)$ with (4).

In this paper we want to use available, off-the-shelf BMC solvers, such as [11], to check temporal properties on general hybrid systems, described as discrete-time hybrid automata. The key idea is based on the observation that the discrete dynamics and continuous dynamics communicate through binary signals, as shown in Figure 1(a). Replacing the continuous dynamics with a *finite horizon* discrete abstraction results in a purely discrete system on which BMC can be applied, as shown in Figure 1(b). This has the twofold benefit of i) exploiting the start-of-art techniques for bounded model checking techniques built in a BMC solver and ii) extending the use of BMC to generic hybrid systems.

IV. Finite Horizon Discrete Abstraction

A. Simulation relations

Our goal in this section is to abstract the discrete-time continuous dynamics with discrete outputs (as shown in Figure 1(a)) with a purely discrete transition system (as shown in Figure 1(b)) that *simulates* the continuous dynamics for a pre-specified horizon N, so that we can apply BMC on the purely discrete system.

Given a prespecified horizon N, we define the *finite-horizon* transition system $T_{C,N} = (Q_C, Q_C^0, \Sigma, \to_C, O, M_C)$ of the continuous dynamics as composed of:

- a set $Q_C = \mathcal{X} \times \{0, \dots, N\}$ of states,
- a set $Q_C^0 = \mathcal{X}^0 \times \{0\}$ of initial states,
- a set $\Sigma = \mathcal{Q}$ of labels,
- transition relation $\to_C \subseteq Q_C \times \Sigma \times Q_C$ defined as

$$\begin{aligned}(x,k) \xrightarrow{q}_C (x',k') \Leftrightarrow & k < N, k' = k+1, \\ & \exists u \in \mathcal{U}, \\ & x, x' \in \mathcal{X}\end{aligned}$$

with

$$x' = A_q x + B_q u + f_q$$

- a set $O = \mathcal{E}$ of observations,
- a map $M_C : Q_C \to O$ that associates to $X \subseteq \mathcal{X}$ at time k the set of observations $M_C(X \times k) \subseteq O$ as described by (3) for all $x \in X$.

Therefore the transition system captures the discrete-time dynamics up to the horizon of interest. Given a set $X \subset \mathcal{X}$ and $q \in \mathcal{Q} = \Sigma$, we define the following operator:

$$\begin{aligned}Post(X,q) = \{x' \in \mathcal{X} \mid & x' = A_q x + B_q u + f_q, \\ & x \in X, u \in \mathcal{U}\}\end{aligned} \tag{5}$$

Given a sequence $\mathbf{q}^{T+1} = q_0 q_1 \dots q_T q_{T+1}$ we can naturally define

$$\begin{aligned}&Post(X, \mathbf{q}^{T+1}) \\ &\quad = Post(Post(X, \mathbf{q}^T), q_{T+1})\end{aligned} \tag{6}$$

We now define a transition system $T_{D,N} = (Q_D, Q_D^0, \Sigma, \to_D, O, M_D)$ that simulates all transitions up to the specified horizon N. Each state of $T_{D,N}$ represents a possible sequence of $\mathcal{Q}$ that can be generated by (4), that is

$$\begin{aligned}Q_D = \{\mathbf{q}^T \mid \mathbf{q}^T = & q_0 q_1 \dots q_T,\ 0 \le T \le N, \\ & q_0 q_1 \dots q_T \text{ generated by (4)}\}\end{aligned} \tag{7}$$

The initial condition is simply $Q_D^0 = q_0$. We also have that $\Sigma = \mathcal{Q}$ and $O = \mathcal{E}$. The observation map M_D is defined as

$$M_D(\mathbf{q}^T) = M_C(Post(\mathcal{X}^0, \mathbf{q}^T)) \tag{8}$$

It remains to define the transition relation which is naturally defined as

$$\begin{aligned}\mathbf{q}^k \xrightarrow{q_j}_D \mathbf{q}^j \Leftrightarrow & 0 \le k < N, j = k+1, \\ & \mathbf{q}^k = \mathbf{q}^{j-1}\end{aligned}$$

The finite transition system $T_{D,N}$ defined above indeed simulates all finite horizon transitions of $T_{C,N}$ but under the assumption that the sequence of events that are input to $T_{D,N}$ and $T_{C,N}$ are generated by (4).

Note that in the worst case, the abstracted transition system may have up to $|\mathcal{Q}|^N$ states. This is clearly needed if we place no constraints on the possible sequences that are input to $T_{C,N}$. However, by requiring that sequences also satisfy the automaton dynamics (4) we may get significant savings if the system dynamics constrains the set of possible sequences.

B. Simulation Algorithm

Having established the formal relationship between the two models, we now focus on an algorithm which given $T_{C,N}$ computes $T_{D,N}$. The simulation algorithm is based on the idea that starting from an initial polytope $\mathcal{X}^0 \subset \mathcal{X}$ we compute at each instant time $t \in \mathbb{Z}^+$ the (forward) reach set $\mathcal{X}_{\mathbf{q}^t}$, recursively defined as

$$\begin{aligned} \mathcal{X}_{\mathbf{q}^0} &= \mathcal{X}^0, \\ \mathcal{X}_{\mathbf{q}^t} &= Post(\mathcal{X}_{\mathbf{q}^{t-1}}, q_t), \end{aligned} \tag{9}$$

where $\mathbf{q}^{t-1} = q_0 q_1 \dots q_{t-1}$, which represents the set of states reachable in 1 step from the set $\mathcal{X}_{\mathbf{q}^{t-1}}$ subject to the input set $\mathcal{U}$ and a feasible discrete transition q_t. At each set $\mathcal{X}_{\mathbf{q}^t}$ is uniquely associated a discrete state defined through a function $label$ recursively defined as

$$\begin{aligned} label(\mathcal{X}_{\mathbf{q}^0}) &= [], \\ label(Post(\mathcal{X}_{\mathbf{q}^{t-1}}, q_t)) &= label(\mathcal{X}_{\mathbf{q}^{t-1}}) \cdot q_t = \mathbf{q}^{t-1} \cdot q_t, \end{aligned} \tag{10}$$

which represents the feasible sequence of dynamics $\mathbf{q}^t = q_0 q_1 \dots q_{t-1} q_t$ that generates $\mathcal{X}_{\mathbf{q}^t}$ from the initial set $\mathcal{X}_0$.

The set of halfspaces (3) partition $\mathcal{X}$ in p regions $\mathcal{P}_i \subseteq \mathcal{X}$ such that i) $\mathcal{X} = \cup_{i=1}^{p} P_i$, ii) $P_i \cap P_t = \emptyset$ for all $i \neq t$. The i-th partition is defined as

$$P_i = \left\{ x \in \mathcal{X} : \begin{array}{ll} C^j x \leq D^j & j \in J_i \\ C^h x > D^h & h \in \bar{J}_i \end{array} \right\}, \tag{11}$$

where $J_i \subseteq \{1, \dots, n_e\}$, $\bar{J}_i = \{1, \dots, n_e\} \backslash J_i$. At each partition P_i is associated an observation E_i, a binary vector of length n_e (i.e. the number of hyperplanes in (3)) obtained by the inequalities in (11). The elements of E_i whose indices are in J_i are set to 1 whereas the rest of elements is set to 0. We define

$$M_C(\mathcal{X}_{\mathbf{q}^t}) = \{E_i \mid \mathcal{X}_{\mathbf{q}^t} \cap P_i \neq \emptyset\} \tag{12}$$

as the set of observations feasible for the set $\mathcal{X}_{\mathbf{q}^t}$. Therefore at each discrete state $label(\mathcal{X}_{\mathbf{q}^t})$ corresponds the set of observations $M_D(label(\mathcal{X}_{\mathbf{q}^t})) = M_C(\mathcal{X}_{\mathbf{q}^t})$.

The result of the exploration can be described as a *simulation tree* where the nodes of the tree represent sets from which a reach set evolution is computed, and an edge connects two nodes if a transition exists between the two corresponding sets. Each edge has associated a label which represents the dynamics enabling the transition. The root node of the tree corresponds to the initial set $\mathcal{X}^0$, from which the reach set evolution is computed.

Algorithm 1 Simulation algorithm

1: **Inputs:**
- Finite horizon transition system $T_{C,N}$,
- Initial set $\mathcal{X}_0$,
- Horizon N.

2: **Output:** Discrete transition system $T_{D,N}$.

3: $Q_D := label(\mathcal{X}^0)$; $Q_D^0 := label(\mathcal{X}^0)$;
4: $O_D := O_C$;
5: $\Sigma_D := \Sigma_C$;
6: SIMULATE($\mathcal{X}^0$,[],0); ▷ Initial step

7: **function** SIMULATE($\mathcal{X}_{\mathbf{q}^t}$,$q$,$T$)
8: **if** $T == N$ **then return** ;
9: **end if**
10: $M_D := M_D \cup M_D(label(\mathcal{X}_{\mathbf{q}^t})) = M_C(\mathcal{X}_{\mathbf{q}^t})$;
11: **for all** $q' \in f_d(q, M_C(\mathcal{X}_{\mathbf{q}^t}))$ **do**
12: $\mathcal{X}_{\mathbf{q}^{t+1}} := Post(\mathcal{X}_{\mathbf{q}^t}, q')$;
13: $Q_D := Q_D \cup label(\mathcal{X}_{\mathbf{q}^{t+1}})$;
14: $\rightarrow_D := \rightarrow_D \cup label(\mathcal{X}_{\mathbf{q}^t}) \xrightarrow{q'}_D label(\mathcal{X}_{\mathbf{q}^{t+1}})$;
15: SIMULATE($\mathcal{X}_{\mathbf{q}^{t+1}}$,$q'$,$T+1$);
16: **end for**
17: **end function**

Algorithm 1 summarizes the simulation procedure. The algorithm builds the simulation starting from $\mathcal{X}_{\mathbf{q}^t}$, assuming that $\mathcal{X}_{\mathbf{q}^t}$ has been generated through the dynamics q at time T. The simulation begins at line 1.6 with $\mathcal{X}^0$, $T = 0$ and $q = []$ since it represents the root node.

Function SIMULATE is recursively called on all possible dynamics (line 1.11) that can be activated by the discrete dynamics (4) given a feasible observation $M_C(\mathcal{X}_{\mathbf{q}^t})$ and assuming as current state the last dynamics q. Internally, line 1.11 solves the following satisfiability (SAT) problem:

$$\begin{array}{ll} \text{Find} & q' \\ \text{s.t.} & q' = f_d(q, e) \\ & e \in M_C(\mathcal{X}_{\mathbf{q}^t}). \end{array} \tag{13}$$

This problem represents the constraint which avoids the enumeration of all $|\mathcal{Q}|^N$ possible combinations. All possible feasible sequences of length N are enumerated with a depth-first strategy which is obtained by using recursion.

Proposition 1: Algorithm 1 generates in finite time a discrete transition system $T_{D,N}$ which simulates the continuous finite horizon transition system $T_{C,N}$.

Remark 1: Let us assume we have a discrete transition system $T_{D,N}$ generated by Algorithm 1 for a certain horizon N. If we want to compute a discrete transition system $T_{D,N'}$ from the same initial set for a longer horizon $N' > N$ we do not need to recompute the simulation tree from scratch. In fact, it is sufficient to start Algorithm 1 from the leaves of the simulation tree $T_{D,N}$ and compute the simulation on the horizon $N' - N$.

Remark 2: The simulation tree represents an over-approximation of the discrete behavior of the continuous dynamics. In fact, the paths on the simulation tree from the

root node to the leaf nodes determine a superset of feasible sequences $\{q_0, \ldots, q_N\}$.

V. Electronic Height Controller

In this section we consider the Electronic Height Controller (EHC) examined in [12]. In this section we want to verify that EHC satisfies certain safety properties by using BMC.

A. Description of the System

The aim of the EHC is to increase the driving comfort by adjusting the chassis level. This is achieved by a pneumatic suspension at each of the four wheels. The chassis level can be increased by pumping air into the suspension of the wheels and decreased by blowing air off. To simplify the discussion we consider a restricted model including only one wheel [12]. The suspension system is commanded by a three states logic controller described in [12]. The controller switches the compressor on when the level of the chassis is below a certain outer tolerance OTl, off when it reaches again an inner tolerance ITl, while it opens the escape valve when the level is above OTh, and closes it again when the level decreases below ITh. Because of high frequency disturbances due to irregularities of the road, the controller switches based on a filtered version $f(t) = \frac{1}{1+as}h(t)$ of the measured level h of the chassis. The filter is reset to $f = 0$ each time f returns within the range [ITl,ITh]. The compressor can lift the chassis at a rate $cp(t) \in [cp_{min}, cp_{max}]$, and the escape valve can lower it at a rate $ev(t) \in [ev_{min}, ev_{max}]$. All parameter values are reported in [12].

B. DHA and Finite Discrete Abstraction

The continuous dynamics of the filter and of the car, sampled by exact discretization (by introducing a zero-order hold), are

$$f(t+1) = e^{-aT_s}f(t) + (1 - e^{-aT_s})h(t), \tag{14a}$$
$$h(t+1) = h(t) + T_s(d(t) + c(t)), \tag{14b}$$

where T_s is the sampling time, $d(t)$ is an unknown but bounded noise which ranges within $[d_{min}, d_{max}]$, and

$$c(t) = \begin{cases} 0 & \text{in "no Action"} \\ cp(t) & \text{in "Compresson ON"} \\ ev(t) & \text{in "Escape Valve OPEN"} \end{cases}, \tag{15}$$

that is the continuous input is selected by the state of the logic controller. The transitions of the automaton are defined as the following set of thresholds

$$[e_1 = 1] \Leftrightarrow [f \leq \text{ITh}], \quad [e_2 = 1] \Leftrightarrow [f \leq \text{ITl}], \tag{16a}$$
$$[e_3 = 1] \Leftrightarrow [f \geq \text{OTh}], \quad [e_4 = 1] \Leftrightarrow [f \geq \text{OTl}], \tag{16b}$$

which partition the continuous state space in 5 regions.

The three states of the automaton are represented by a two-dimensional logic state

$$q(t) = [q_1(t)\ q_2(t)]' = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{"No Action"} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{"Compressor ON"} \\ \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{"Valve OPEN"} \end{cases}, \tag{17}$$

and logic dynamics can be written as

$$q_1(t+1) = (\neg q_1(t) \wedge e_3(t)) \vee (q_1(t) \wedge \neg e_1(t)) \tag{18a}$$
$$q_2(t+1) = (q_2(t) \wedge e_2(t)) \vee (\neg q_1(t) \wedge \neg e_4(t)) \tag{18b}$$

When the logic controller switches to the "No Action" state the filter is reset to $f = 0$.

The EHC system can be conveniently described as a DHA system. Dynamics (14) with input (15) represent the SAS dynamics (2). At each instant time the continuous input is selected through the discrete state $q(t)$. Thresholds (16) represent the quantizer (3) and discrete dynamics (18) represent the automaton (4).

In order to verify safety properties on the EHC system we compute the abstraction of the dynamics (14), (15) and quantizer (16). An example for $N = 7$, starting from the initial set $\mathcal{X}^0 = \{(f,h)|\ f = 0,\ h \in [ITl, ITh]\}$, is shown in Figure 2. The shape of each state represents the number of outputs feasible for that state. Each transition is labeled with the dynamics which activates the transition.

C. Numerical Results

Algorithm 1 has been implemented in Matlab and uses the Multiparametric toolbox [13] for polytope manipulation and the zCHAFF solver [2] for the satisfiability test. As BMC solver we used the nuSMV solver [11].

For the initial set $\mathcal{X}^0 = \{(f,h)|\ f = 0,\ h \in [ITl, ITh]\}$ we have computed the abstraction for different horizons, as reported in Table I. We can see that longer the horizon, longer the time to compute the abstraction. The exponential growth of time spent for computing the abstraction depends both on the continuous and the discrete dynamics of the system. The continuous dynamics influences the number of feasible observation e for each state whereas the discrete dynamics constraints the possible feasible values of q.

We can use the abstraction in conjunction with the discrete controller in order to verify some safety properties. One question is to verify that the escape valve is never open the instant after the compressor is on. This question can be posed as the following LTL formula:

$$\square(\text{"Compressor ON"} \Rightarrow \bigcirc\text{"Escape valve OPEN"}). \tag{19}$$

Bounded model checking proves in 0.41 s that the abstraction for $N = 18$ and the discrete controller cannot reach this condition.

The abstraction for $N = 18$ can be used to check more complex questions, as "If the system is in the "Escape Valve OPEN" state can we observe "No Action" within 8 s (8 steps)?". To answer to this question we first introduce the notion of n-next operator

$$\bigcirc_n \phi = \underbrace{\bigcirc(\bigcirc(\ldots(\bigcirc}_{n}\phi)\ldots), \tag{20}$$

where ϕ is a temporal formula, and says that any trajectory $q[t] \models \bigcirc_n \phi$ if $q[t+n] \models \phi$. The natural language question

TABLE I

ABSTRACTION FOR DIFFERENT HORIZONS

N	Time (s)	States
5	0.41	7
6	1.16	11
7	1.67	19
8	3.14	30
9	4.64	51
10	6.75	88
11	10.39	153
12	18.01	268
13	31.57	477
14	57.15	875
15	108.82	1667
16	215.41	3289
17	443.42	6775
18	970.04	14282

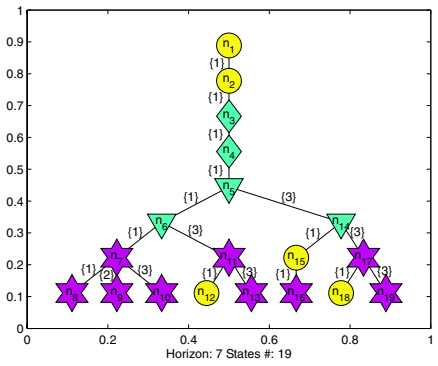


Fig. 2. Abstraction for $N = 7$ (No. of outputs: circle=1, diamond=3, triangle (down)=4, Hexagram > 4; Labels: 1 "No Action", 2 "Compressor ON", 3 "Escape Valve OPEN").

can be translated into the following temporal property:

$$\Box \left(\text{“Escape Valve OPEN”} \Rightarrow \bigvee_{n=1}^{8} \bigcirc_n \text{“No Action”} \right). \tag{21}$$

Bounded model checking proves that in 0.63 s starting from the initial condition $\mathcal{X}^0$ there is the following sequence of dynamics which violates the property:

$$\begin{aligned} \mathbf{q}^{14} = 1 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow 3 \rightarrow 3 \\ \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow 3 \\ \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow 1, \end{aligned}$$

where 1 corresponds to "No Action" and 3 to "Escape Valve OPEN".

Because the abstraction represents a superset of all possible discrete trajectories generated by the continuous dynamics the feasibility of the counterexample obtained by BMC has to be checked [14]. Feasibility can be simply tested via linear programming over the sets of linear equalities and inequalities obtained by unfolding the dynamics (2), (3), (4) on the sequence generated by BMC and translating logic formulas and if-then-else rules in linear inequalities as described in [3].

All computations have been done on an Intel Centrino 1.2 GHz with 640 Mb of RAM.

VI. CONCLUSIONS

Bounded model checking has been extended to discrete-time hybrid automata, a general class of hybrid systems. By means of an abstraction procedure the continuous dynamics is replaced by a finite-time discrete transition system which simulates the discrete behavior of the continous dynamics. Bounded model checking can be used for verifying safety properties on the purely discrete system, obtained by the composition of the discrete part of the hybrid system with the abstraction. The abstraction procedure and bounded model checking have been applied to an industrial case study.

Ongoing research is devoted to define abstraction procedures which bisimulate the continuous dynamics in order to use temporal properties for control of hybrid systems and use the current abstraction procedure in a model predictive verification setting where the horizon is moving in time.

ACKNOWLEDGMENTS

The authors would like to thank Fabio Torrisi for the fruitful discussions.

REFERENCES

[1] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58, 2003.

[2] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaf: Engineering an efficient SAT solver. In *39th Design Automation Conference*, Las Vegas, 2001.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[4] M. Fränzle and C. Herde. Efficient proof engines for bounded model checking of hybrid systems. *FMICS'04*, 2004.

[5] G. Audermard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying industrial hybrid systems with MathSAT. In *Proc. of the 2nd Intern. Workshop on Bounded Model Checking*, 2004.

[6] R. Alur, T.A. Henzinger, and G. Lafferriere G.J. Pappas. Discrete abstractions of hybrid systems. *Proc. of the IEEE*, 88(7), July 2000.

[7] D. Förstner, M. Jung, and J. Lunze. A discrete-event model of asynchronous quantised systems. *Automatica*, 38:1277–1286, 2002.

[8] F.D. Torrisi and A. Bemporad. HYSDEL - a tool for generating computational hybrid models. *IEEE Trans. Contr. Systems Technology*, 12(2):235–249, March 2004.

[9] 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 R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[10] J. Gu, P.W. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, number 35, pages 19–151. American Mathematical Society, 1997.

[11] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *International Conference on Computer-Aided Verification*, Copenhagen, Denmark, July 2002.

[12] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In F.W. Vaandrager and J.H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 1999.

[13] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi Parametric Toolbox (MPT). In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 448–462, Philadelphia, Pennsylvania, USA, March 2004. Springer Verlag.

[14] F.D. Torrisi and A. Bemporad. Discrete-Time Hybrid Modeling and Verification. In *IEEE Conference on Decision and Control*, Orlando, FL, December 2001.