43rd IEEE Conference on Decision and Control
December 14-17, 2004
Atlantis, Paradise Island, Bahamas

Improving Reachability Analysis of Hybrid Automata for Engine Control

Alberto Casagrande*†, Andrea Balluchi†, Luca Benvenuti†‡, Alberto Policriti*,
Tiziano Villa†§ and Alberto Sangiovanni-Vincentelli†¶
*DIMI, Univ. di Udine, Via delle Scienze, 206, 33100 Udine, Italy
†PARADES, Via S.Pantaleo, 66, 00186 Roma, Italy
‡DIS, Univ. di Roma La Sapienza, Via Eudossiana, 18, 00184 Roma, Italy
§DIEGM, Univ. di Udine, Via delle Scienze, 208, 33100 Udine, Italy
¶EECS Dept., University of California, Berkeley, CA 94720, USA

***Abstract*— A new approach is presented for computing approximations of the reached sets of linear hybrid automata. First, we present some new theoretical results on termination of a class of reachability algorithms, which includes Botchkarev's, based on ellipsoidal calculus. The main contribution of the paper is a revised reachability computation that avoids the approximations caused by the union operation in the discretized flow tube estimation. Therefore, the new algorithm may classify as unreachable states that are reachable according to the previous algorithm because of the looser over-approximations introduced by the union operation. We implemented the new reachability algorithm and tested it successfully on a real-life case modeling a hybrid model of a controlled car engine.**

I. Introduction

Hybrid automata (see [1], [2]) are a formalism often used to model the interactions between a controller and its environment. A hybrid automaton is a "finite-state" machine with continuous variables that evolves according to discrete "nodes" characterized by systems of differential equations. Hybrid automata are used in various domains from control of industrial plants to biological systems and it is very important to be able to verify *safety* properties of systems modeled in such a way. Checking safety properties on automata reduces to the *reachability* problem; in particular, to prove that a safety property φ is always true for a hybrid automaton H, we only need to prove that all the states in which φ is false are not reachable from the initial states of H. Since, it has been proved in [3] that, in general, the *reachable set* is not computable, many works propose over-approximations techniques (see [4], [5], [6]).

This paper describes a work in progress on conservative reachability analysis of linear hybrid systems. The starting point is an algorithm for reachability analysis due to Botchkarev (see [7]), based on the representation of continuous regions with ellipsoids; it approximates the global flow-tube of reached sets by the union of partial flow-tubes obtained by discretizing integration time. We present a revised reachability computation that avoids the approximations caused by the union operation in the discretized flow-tube estimation. The new algorithm may classify as unreachable states that the previous one certifies as reachable due to looser over-approximations. Moreover, we give theoretical results on convergence of both algorithms, introducing a class of linear hybrid automata on which they terminate. Finally, checking the correct behaviour of an engine controller by reachability analysis of the corresponding hybrid automaton, we report results on verifying that a closed-loop engine control system satisfies a given reachability specification (see [8], [9]). The experimental study shows that our algorithm succeeds in proving correctness, whereas Botchkarev's algorithm may fail, due to looser over-approximations of the computational scheme.

The work has been conducted with partial support by the European Community Project IST-2001-33520 CC (Control and Computation).

II. Hybrid Automata

Hybrid automata have been introduced in [1], [2]. Formally, a hybrid automaton is defined as follows. Let $\mathcal{C}^m$ be the set of continuous functions from $\mathbb{R}^m$ to $\mathbb{R}^m$:

Definition 1: [Hybrid Automaton] A *hybrid automaton*, H, is a tuple $(\vec{X}, L, E, \mathcal{S}_0, \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{R})$ such that:

- $\vec{X}$ is a set of n continuous variables taking values in $\mathbb{R}$; a valuation $\vec{x} = (x_1, \ldots, x_n)$ of these variables represents the continuous component of a state.
- L is a finite set of *locations*; a valuation ℓ of L represents the discrete component of a state. A pair $(\ell, \vec{x}) \in L \times \mathbb{R}^n$ is called *state of the automaton*;
- E is the set of *arcs* (or *edges*) of the automaton. Each arc $e = (\ell, \ell')$ links a *source* location, $\ell \in L$, to a *target* location, $\ell' \in L$;
- $\mathcal{S}_0$ is the set of *initial* states;
- $\mathcal{F}$ is a function that associates to each location a system of differential equations. The system $\mathcal{F}(\ell)(\vec{X})$ is the dynamical system associated to the automaton's location ℓ;
- $\mathcal{I}$ is a function called *invariant* that associates to each location a subset of $\mathbb{R}^n$. A hybrid automaton cannot stay in a state $s = (\ell, \vec{x})$ such that $\vec{x} \notin \mathcal{I}(\ell)$;
- $\mathcal{A}$ is a function that associates to every arc a subset of $\mathbb{R}^n$. A hybrid automaton having a state $s = (\ell, \vec{x})$ may traverse an arc $e \in E$ if and only if $\vec{x} \in \mathcal{A}(e)$.

The set $\mathcal{A}(e)$ is called *activation region* or *guard* of the arc e;

$\mathcal{R}$ associates to every arc e a continuous function from $\mathbb{R}^n$ to $\mathbb{R}^n$, which reassigns values to the continuous variables when an arc is traversed. If the automaton H is in state $s = (\ell, \vec{x})$, by traversing $e = (\ell, \ell')$, H goes into state $s' = (\ell', \mathcal{R}(e)(\vec{x}))$. Extending the meaning of this function, given the set $R \subseteq \mathbb{R}^n$, $\mathcal{R}(e)(R)$ will be equal to the set $\{\mathcal{R}(e)(\vec{x}) | \vec{x} \in R\}$.

The execution of a hybrid automaton corresponds to a sequence of states of the automaton. Any infinite transition sequence starting from an initial state, s_0, will be called *automaton run* and it will be represented using the notation: $s_0 \Longrightarrow^*$. Hybrid automata have two kinds of transitions: flows, capturing the continuous evolution of a state, and steps, capturing the changes of location.

More formally, if f is a solution of $\mathcal{F}(\ell)(\vec{X})$ such that $f(0) = \vec{x}$ and if $\mathcal{I}(\ell)$ includes $f(t')$ for all $t' \in (0, t]$, then there exists a t-timed *flow* from $s = (\ell, \vec{x})$ to $s' = (\ell, f(t))$ and we write $s \xrightarrow{t} s'$. The set $\langle S \rangle_t^{\rightarrow} = \{s' | s \xrightarrow{t'} s' \wedge s \in S \wedge 0 \leq t' \leq t\}$ includes all the states that the automaton can reach from the set of states S using a t'-timed flow, where $t' \leq t$. This set is called *flow-tube* from S of time t. Given state $s = (\ell, \vec{x})$ and arc $e = (\ell, v')$, if $\vec{x} \in \mathcal{A}(e)$ then the automaton may take a *step* transition from $s = (\ell, \vec{x})$ to $s' = (\ell', \mathcal{R}(e)(\vec{x}))$. When state $s = (\ell, \vec{x})$ is such that $\vec{x} \in \mathcal{A}(e)$, we will say that arc e is *enabled*. Moreover, we will represent the set of states that are reachable from state set S with a step transition by the arc $e = (\ell, \ell')$, using the notation $[S]_e^{\rightarrow} = \{s' | (\ell, \vec{x}) \in S \wedge \vec{x} \in \mathcal{A}(e) \wedge s' = (\ell', \mathcal{R}(e)(\vec{x}))\}$.

In general, a state set S may include states belonging to more than one location and, indeed, it can be partitioned into a finite number of subsets $\{S_\ell\}_{\ell \in L}$ such that $S_\ell \subseteq \{\ell\} \times \mathbb{R}^n$. Thus, the reach set from S may be computed as the union of the reach sets from each single subset S_ℓ. Later on, to ease notation, we suppose that the initial set, $\mathcal{S}_0$, has the form $\{\ell\} \times R$, $R \subseteq \mathbb{R}^n$, because the proposed algorithms can be extended to the general case $\mathcal{S}_0 \subseteq L \times \mathbb{R}^n$, by simply iterating them for each single set of the suggested partition.

Given a set of states $S = \{(\ell, \vec{x}) |\ \vec{x} \in R\}$ where $\ell \in L$, we introduce the discrete projection operator $\downarrow$ that restricts it to the location ℓ and the continuous projection operator $\Downarrow$ that restricts it to the continuous component R (i.e., $[S]_\downarrow = \ell$ and $[S]_\Downarrow = R$). To ease the notation, we may omit a projection operator when it is clear from the context.

An important subset of hybrid automata is the class of *linear hybrid automata*.

Definition 2: A *linear hybrid automaton* is a hybrid automaton such that, for each arc e and for each location ℓ, both $\mathcal{F}(\ell)(\vec{X})$ and $\mathcal{R}(e)$ are linear.

Later on, we will focus on such automata and we will present two reachability analysis algorithms and theoretical results on their termination.

III. Previous Work

A technique used to represent and operate on sets in reachability analysis is sometimes called a *calculus method*. Calculus methods may differ for the technique employed to represent set of states as well the algorithmic aspects of continuous state set computations. Two calculi commonly used are polyhedral calculus (see [10], [11]) and ellipsoidal calculus (see [5], [7]), the former representing sets using polyhedra and the latter using ellipsoids.

An algorithm based on ellipsoidal representations was presented by Botchkarev in [7]. The algorithm over-approximates the reachable set of a linear hybrid automaton and in fact can be used with any calculus method. If Botchkarev's algorithm returns *UNREACHABLE* on a given linear hybrid automaton H, a set of states S_T, and a real number $\bar{t}$, then the over-approximation guarantees that H cannot reach S_T in time $\bar{t}$. To evaluate the reach set, the algorithm maintains a table $\mathcal{T}$ of pairs. Each pair has the form (S, t_S), where S is a set of states and t_S is a positive real value. When (S, t_S) is in $\mathcal{T}$, the set of states S is considered as reachable by automaton H in time t_S.

The key point in Botchkarev's approach is the discretization of time, by a *time step*, δ. Given the parameter δ, for each $k \in \{0, \ldots, \left\lceil \frac{\bar{t} - t_S}{\delta} \right\rceil\}$, the algorithm starts by computing the set, R_k, of points reachable from R at time $t = k\delta$. Since some of the states that are reachable with a t'-timed flow, where $k\delta \leq t' \leq (k+1)\delta$, may not be included in these approximations, the algorithm expands, using the Minkowski sum, each set R_k by a ball $B_\epsilon(\vec{0})$, centered in $\vec{0}$ and with radius ϵ. The radius ϵ is set in such a way to account for *every* state reachable at time $t \leq t' \leq t + \delta$. Lemma 1 (from [7]) shows how the radius ϵ can be evaluated beforehand as a function of δ and of the flow function for the current location.

Lemma 1: Let (ℓ, R) be a set of states and let $\dot{\vec{x}} = A_\ell \vec{x} + U_\ell$ be the differential equation associated to ℓ by $\mathcal{F}(\ell)(\vec{X})$, where A is a $n \times n$-matrix and $U \subseteq \mathbb{R}^n$ is compact and convex. For all $t \in [0, \delta]$ is true that $[\langle (\ell, R) \rangle_t^{\rightarrow}]_\Downarrow \subseteq R + B_{\epsilon_\delta}(\vec{0})$ where $\epsilon_\delta = \left(e^{N_A \delta} - 1\right) D + e^{N_A \delta} N_U \delta$ and $N_A = \|A\|$, $N_U = \max_{u \in U} \|u\|$ and $D = \max_{x \in R} \|x\|$.

Proof: See the proof in [7]. ∎

From now on, we will denote with $\chi^+_{\ell,\delta}(R, k)$ the function that evaluates the region reached with a $k\delta$-timed flow from the set R in the location ℓ. Furthermore, given a region R, a location ℓ and an arc $a = (\ell, \ell')$, the regions V_a^+ and W_a^+ will denote the over-approximation of the intersection between the flow tube and the guard of a and the over-approximation of the continuous part reset by a step transition over a respectively.

The parameters of Botchkarev's algorithm are: an automaton H, a set of states S_T whose reachability must be tested, and a time step δ. During the initialization phase the pair $(\mathcal{S}_0, 0)$ is inserted into $\mathcal{T}$. The steps of the algorithm can be described as follows:

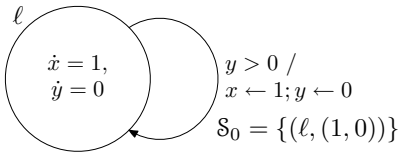


Fig. 1. This automaton is non-Zeno, but the reachability algorithm does not terminate on it.

1) if $\mathcal{T}$ is empty then stop and return *UNREACHABLE*. Otherwise, pick in $\mathcal{T}$ a pair (S, t_S), with the smallest t_S and where $S = (\ell, R)$, and remove it from the table;
2) for the selected pair (S, t_S):
   a) evaluate $R^+_{\ell,k} = \chi^+_{\ell,\delta}(R, k) + B_{\epsilon_\delta}(\vec{0})$, for $k \in \{0, \ldots, m\}$, where m is the minimum between $\left\lceil \frac{\bar{t} - t_S}{\delta} \right\rceil$ and the smallest k such that $R^+_{\ell,k} \cap \mathcal{I}(\ell) = \emptyset$;
   b) if $\ell = [S_T]_\downarrow$ and $R^+_{\ell,k} \cap [S_T]_{\Downarrow} \neq \emptyset$, for some $k \in \{0, \ldots, m\}$, then return *REACHABLE*;
   c) let $A_q = \{a_1, \ldots, a_l\}$ be the set of arcs leaving location ℓ.
      i) For each $a \in A_q$, set the regions V^+_a and W^+_a to $\emptyset$;
      ii) Let $j_{min(a_i)}$ be the minimum value between $m + 1$ and the smallest k such that $\mathcal{A}(a_i) \cap R^+_{\ell,k} \neq \emptyset$. If $0 \leq j_{min(a_i)} \leq m$ then compute $V^+_{a_i} \supseteq \bigcup_{j=j_{min(a_i)}}^{m} \left(R^+_{\ell,j} \cap \mathcal{A}(a_i) \right)$.
      iii) If $V^+_{a_i} \neq \emptyset$ then compute $W^+_{a_i} \supseteq \left[V^+_{a_i} \right]^{\rightarrow}_{a_i}$.
3) let τ_i be such that $\tau_i = j_{min(a_i)}\delta$ and let $a_i = (\ell, \ell_i)$. Then, for each $W^+_{a_i} \neq \emptyset$ computed at step 2(c)iii, add to $\mathcal{T}$ the pair $(S_i, t_p + \tau_i)$, where $S_i = (\ell_i, W^+_{a_i})$, and repeat from step 1.

Termination of Botchkarev's algorithm is not guaranteed. As a matter of fact, if we apply this algorithm to a *Zeno* automaton, the table $\mathcal{T}$ will not necessarily become empty.

Note also that, because of the over-approximation of the flow-tube, the algorithm may fail to terminate even on a non-Zeno automaton. For instance, consider the example presented in Fig. 1: the flow starting from the initial states does not enable any arc, thus the automaton is not a Zeno automaton. Nevertheless, $N_A > 0$, $N_U = 0$ and $D > 0$ and then the ball $B_{\epsilon_\delta}(\vec{0})$ used in step 2a of the algorithm has radius $\epsilon_\delta > 0$. Indeed, R^+_0 intersects the arc's guard and the pair $(S, 0)$, where $S = (\ell, \{(1, 0)\})$, will be inserted into $\mathcal{T}$, bringing back the algorithm to the initial state.

IV. Some Theoretical Results on Termination

To guarantee convergence of the reachability algorithm, we would like to restrict non-Zeno automata to identify a class such that the expansions made by the algorithm at step 2a and the approximations produced by the calculus method do not cause an infinite sequence of arc activations. For this reason, we introduce the definitions of *ϵ_H-expansive hybrid automaton*, *approximation index of a calculus method* and *$(\mathcal{C}, \delta)$-compatibility*. The idea is that, if e is an arc and S is a set of states, if a hybrid automaton is ϵ_H-expansive, then expanding $[S]^{\rightarrow}_e$ by means of a ϵ_H-ball, no arc at the destination location is enabled.

Definition 3: [ϵ_H-expansive hybrid automaton] A hybrid automaton H is *ϵ_H-expansive*, where $\epsilon_H \in \mathbb{R}_{\geq 0}$ is the *expansion radius*, if $\left([\{\ell_1\} \times \mathcal{A}(e_1)]^{\rightarrow}_{e_1} + B_{\epsilon_H}(\vec{0}) \right) \cap \mathcal{A}(e_2) = \emptyset$, for each pair of arcs $e_1 = (\ell_1, \ell_2), e_2 = (\ell_2, \ell_3) \in E$.

To obtain termination, we also need to guarantee that the approximation made by the calculus method does not enable an arc incorrectly. For this reason, we introduce the *approximation index of a calculus method*. Given an arc $e = (\ell, \ell')$ and a calculus method $\mathcal{C}$, let $V^{\mathcal{C}}_e$ be the over-approximation of the set $V_e = \{\ell\} \times \mathcal{A}(e)$.

Definition 4: [Approximation index of a calculus method] The *approximation index* $\gamma^{\mathcal{C}}_H$ of calculus method $\mathcal{C}$ on hybrid automaton H is:

$$\gamma^{\mathcal{C}}_H = \max_{e \in E} h_+ \left(\left[V^{\mathcal{C}}_e \right]^{\rightarrow}_e, [\{\ell\} \times \mathcal{A}(e)]^{\rightarrow}_e \right), \qquad (1)$$

where h_+ is the *Hausdorff's semi-distance* between two sets of states.

Definition 5: [$(\mathcal{C}, \delta)$-compatibility] Let H be an ϵ_H-expansive hybrid automaton and, for each location ℓ, let $\dot{x} = A_\ell x + U_\ell$ be the flow system of ℓ. Suppose that $\gamma^{\mathcal{C}}_H$ is the approximation index of calculus method $\mathcal{C}$ on H. If there exists a $\delta \in \mathbb{R}_{>0}$ such that, for each arc $e = (\ell, \ell')$,

$$\epsilon_H - \gamma^{\mathcal{C}}_H > \left(e^{N_{A_{\ell'}} \delta} - I \right) D(e) + e^{N_{A_{\ell'}} \delta} N_{U_{\ell'}} \delta, \qquad (2)$$

where $D(e) = \max_{x \in [V^{\mathcal{C}}_e]^{\rightarrow}_e} \|x\|$, $N_{A_{\ell'}} = \|A_{\ell'}\|$ and $N_{U_{\ell'}} = \max_{u \in U_{\ell'}} \|u\|$, then, we say that H is *compatible* with calculus method $\mathcal{C}$ and time step δ, i.e., that H is $(\mathcal{C}, \delta)$-compatible.

The next theorem gives an upper bound on the number of cycles required by Botchkarev's algorithm when it runs on a $(\mathcal{C}, \delta)$-compatible automaton H.

Theorem 1: Let H be a $(\mathcal{C}, \delta)$-compatible hybrid automaton. Furthermore, let S_T be a set of states whose continuous part is a compact and convex region, let $\bar{t} \in \mathbb{R}_{>0}$. Then Botchkarev's algorithm, applied to H and S_T, with maximum flow time $\bar{t}$ and time step δ, performs at most $\psi_{\bar{t},\delta}(n)$ cycles, where n is the maximum number of arcs leaving a location and:

$$\psi_{\bar{t},\delta}(n) = \begin{cases} \left(\left\lfloor \frac{\bar{t}}{\delta} \right\rfloor + 1 \right) + 1 & \text{if } n = 1, \\ n \left(\frac{n \left\lfloor \frac{\bar{t}}{\delta} \right\rfloor + 1 - 1}{n - 1} \right) + 1 & \text{if } n \neq 1. \end{cases} \qquad (3)$$

Proof: See the proof in [12]. ∎

From this theorem, the following termination result follows.

Corollary 1: Let H be a $(\mathcal{C}, \delta)$-compatible hybrid automaton. Furthermore, let S_T be a set of states whose continuous part is a compact and convex region, and let $\bar{t} \in \mathbb{R}_{>0}$. If S_T is reachable by H in time $t \leq \bar{t}$, then Botchkarev's algorithm, applied to H and S_T, with maximum flow time $\bar{t}$ and time step δ, terminates and returns *REACHABLE*.

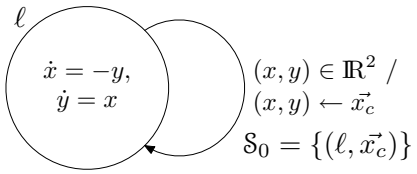


Fig. 2. This automaton is a critical example for Botchkarev's approximation algorithm: many unreachable states are classified as reachable.

V. An Improved Approximation Algorithm

The approximation of the flow tube made by Botchkarev's algorithm tends to be too conservative because step 2(c)ii may include in the reachable set many unreachable states, which we call *trash states*. For instance, consider the automaton in Fig. 2 and suppose we want to evaluate the states that are flow-reachable from $\mathcal{S}_0 = \{(\ell, \vec{x_c})\}$, where $\vec{x_c} = (0, -1)$. The result of applying Botchkarev's algorithm at the end of step 2a is shown in Fig. 3(a). The set indicated as V_a^+ in the algorithm is the ellipsoid shown in Fig. 3(b). The grey area in Fig. 3(c) is the set of all unreachable states, which are classified as reachable because of the over-approximation of the union made at step 2(c)ii.

To decrease the number of trash states, we propose a new algorithm that avoids performing the union operation. This algorithm is more expensive computationally, but it obtains a tighter flow-tube approximation and has a better theoretical approximation index than Botchkarev's one. The new algorithm is based on the fact that a state s is reachable from the set $\bigcup_{j=j_{min(a_i)}}^{m} \left(R^+_{\ell,j} \cap \mathcal{A}(a_i)\right)$ if and only if s is reachable from the set $R^+_{\ell,j} \cap \mathcal{A}(a_i)$, for some $j \in \{j_{min(a_i)}, \dots, m\}$. Thus, we can evaluate the reach set of $\bigcup_{j=j_{min(a_i)}}^{m} \left(R^+_{\ell,j} \cap \mathcal{A}(a_i)\right)$ as the union of the reach sets of each $R^+_{\ell,j} \cap \mathcal{A}(a_i)$. In particular, in the example of Fig. 3, this new algorithm avoids the approximation due to replacing the union of $R^+_{\ell,j}$'s by V_a^+ and instead evaluates the reach set of each single $R^+_{\ell,j}$.

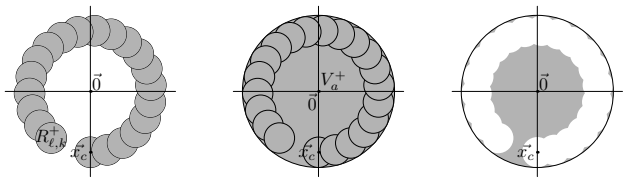
This idea could be implemented using a recursive function called *verify* whose parameters are S, S_T, t_S, $\bar{t}$ and δ, where S and S_T are sets of states, while t_S, $\bar{t}$ and δ are real numbers. This function uses a time step of length δ and returns *TRUE*, if any state in S_T is classified as reachable from set S in time $(\bar{t} - t_S)$.

The main function of the algorithm does the following.

1) if *verify*$(\mathcal{S}_0, \bar{t}, 0, S_T, \delta)$ =*TRUE*, where $\mathcal{S}_0$ is the set of initial states, then return *REACHABLE*;
2) return *UNREACHABLE*.

The function *verify*$(S, \bar{t}, t_S, S_T, \delta)$ performs the following steps.

1) let ℓ be the location of all the states of S and $R = [S]_{\Downarrow}$. For each $j \in \{0, \dots, \left\lceil \frac{\bar{t} - t_S}{\delta} \right\rceil\}$:
   a) evaluate $R^+_{\ell,j} = (\chi^+_\ell(R, j\delta) + B_{\epsilon_{\ell,\delta}}(\vec{0})) \cap \mathcal{I}(\ell)$;
   b) if $\ell = [S_T]_{\downarrow}$ and $R^+_{\ell,j} \cap [S_T]_{\Downarrow} \neq \emptyset$ for some j then return *TRUE*;
   c) for each arc $e = (\ell, \ell')$:
      i) if $R^+_{\ell,j} \cap \mathcal{A}(e) = \emptyset$, then set $V^+_{e,j} = \emptyset$, else evaluate $V^+_{e,j} \supseteq R^+_{\ell,j} \cap \mathcal{A}(e)$;
      ii) if $V^+_{e,j} \neq \emptyset$:
         A) compute $W^+_{e,j} \supseteq \left[V^+_{e,j}\right]^{\rightarrow}_{e}$;
         B) if *verify*$(S_{e,j}, \bar{t}, t_S + j\delta, S_T, \delta)$, where $S_{e,j} = (\ell', W^+_{e,j})$, returns *TRUE*, then return *TRUE*;
2) return *FALSE*.



(a) Sets $R^+_{\ell,k}$ are evaluated by Botchkarev's algorithm at step 2a.

(b) The grey area represents an over-approximation of the union of sets $R^+_{\ell,k}$.

(c) States lying in the grey area are included in the reachable set, but actually they are not reachable.

Fig. 3. These images show how Botchkarev's algorithm may over-approximate a flow-tube. The proposed algorithm avoids the union of sets $R^+_{\ell,k}$ evaluating all their reach sets in place of the reach set of the union.

Although termination is not guaranteed for this new algorithm either, we can prove a complexity bound analogous to Theorem 1.

Theorem 2: Let H be a $(\mathcal{C}, \delta)$-compatible hybrid automaton. Furthermore, let S_T be a set of states whose continuous part is a compact and convex region and let $\bar{t} \in \mathbb{R}_{>0}$. The new algorithm, applied to H and S_T, with maximum flow time $\bar{t}$ and time step δ, calls the function *verify* at most $(n+1)^{\left\lceil \frac{\bar{t}}{\delta} \right\rceil}$ times, where n is the highest number of arcs leaving a location.

Proof: See the proof in [12]. ∎

To gauge the accuracy of the new algorithm, we introduce a new kind of index called *algorithmic approximation index*. This new index gives an error estimate of the flow-tube approximation to compare the quality of different reachability algorithms.

Definition 6: [Algorithmic approximation index] Let H be a hybrid automaton, α a reachability analysis algorithm and $t \in \mathbb{R}_{\geq 0}$. Furthermore, for each pair of edges e and e', let $F^{+\ \alpha}_{e,e'}$ be the over-approximation by algorithm α of the set

$$F_{e,e'} = \left[\langle [\{\ell_1\} \times \mathcal{A}(e)]^{\rightarrow}_{e} \rangle^{\rightarrow}_{t}\right]_{\Downarrow} \cap \mathcal{A}(e'), \qquad (4)$$

which is the subset of $\mathcal{A}(e')$ reachable from $\mathcal{A}(e)$ by a discrete step and a flow. The *algorithmic approxima-*

tion index of α on H at time t is the real number $\mu_H^\alpha = \max_{e,e' \in E} h_+(F_{e,e'}^{+\,\alpha}, F_{e,e'})$. The rationale is to upper bound the over-approximations obtained by algorithm α over all flows induced by all edge transitions.

Note that the smaller is the approximation index of an algorithm, the better is the precision in flow-tube evaluation. As a matter of fact, let us consider two algorithms, α and β that check reachability of the same set of states with the same maximum flow time. Suppose that they have algorithmic approximation indices at time t respectively equal to μ^α and μ^β, where $\mu^\alpha \leq \mu^\beta$. If α verifies reachability then β does too. On the other hand, if β verify reachability, α may return *UNREACHABLE*. Therefore, the set of unreachable states that algorithm α marks as *REACHABLE* is a subset of the states that β marks as *REACHABLE*. For this reason, it is better to use an algorithm with a low algorithmic approximation index.

Theorem 3: Let H be a linear hybrid automaton. If μ^α and μ^β are the algorithmic approximation indices on H, respectively, of the new algorithm and of Botchkarev's algorithm, then $\mu^\alpha \leq \mu^\beta$.

Proof: See the proof in [12]. ■

Thus, even though the calculus method used by both algorithms cannot express exactly the union of sets $R_{\ell,j}^+ \cap \mathcal{A}(e)$, the algorithmic approximation index of the proposed algorithm is at least as small as the one of Botchkarev's algorithm.

VI. Test Case Results

To test the quality of the proposed algorithm, we verified the behavior of an automotive engine subject to idle speed control. The objective was to prove that, for the given engine model, the proposed controller maintains a crankshaft speed between 750 and 850 rpm, for a least 0.5 seconds. For that purpose, we modeled the engine and its controller using hybrid automata. Furthermore we defined forbidden regions of the controlled system and we checked whether they could be reached, by using both algorithms discussed above.

We used the engine model presented in [8], [9]. It is composed of three main blocks: the *intake manifold*, the *cylinders* and the *powertrain*.

The intake manifold's pressure, p, determines the amount of air, m, loaded by the cylinders. Its evolution is subject to the linear dynamics $\dot{p} = a_p p + b_p \alpha$, where α is the engine control input and denotes the throttle valve position. Parameters a_p and b_p depend on geometric features of the intake mainfold. For small variations of the crankshaft revolution speed, n, the amount of air loaded by the engine can be assumed to be proportional to the intake manifold pressure, i.e. $m = kp$. The torque, T, generated by cylinders depends on the air–fuel mixture loaded from the intake manifold and on the spark ignition time. Assuming fuel injection proportional to air intake, so to have stoichiometric air–fuel mixture, the torque generated can be described by $T = Gm\eta(\phi)$, where ϕ is the *spark advance*, G is the air–to–torque gain, and $\eta(\phi) = \frac{1}{175}(2\phi + 135)$ is the *ignition efficiency function*.

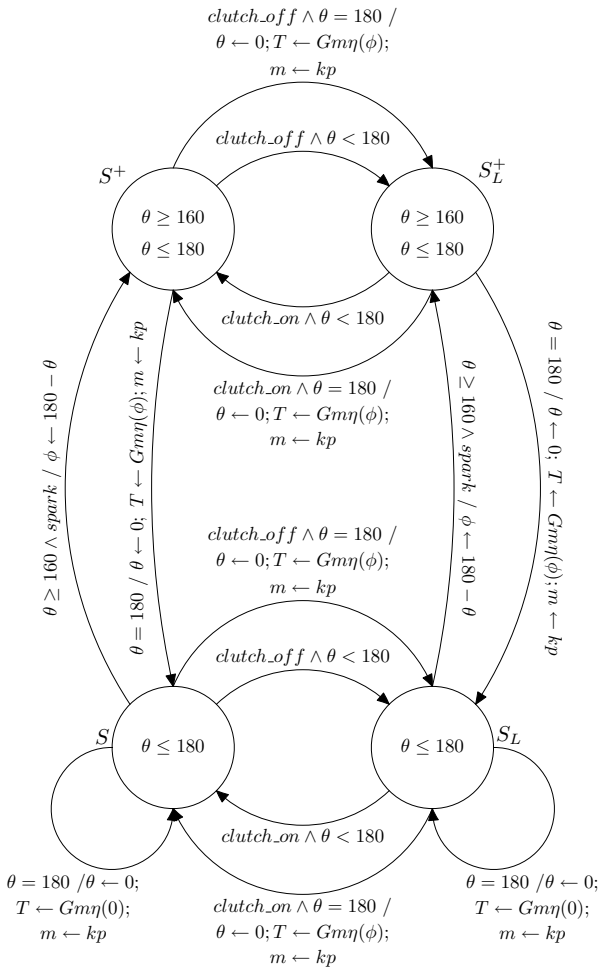


Fig. 4. Engine model

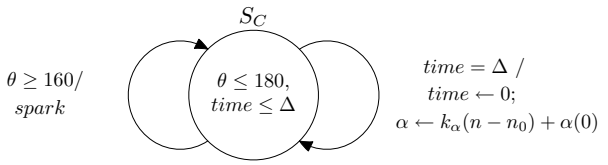


Fig. 5. Engine controller model

TABLE I
Flow equations of the model

$$\begin{cases} \dot{n}(t) = an(t) + bT(t) \\ \dot{\theta}(t) = 6n(t) \\ \dot{T}(t) = \dot{\phi}(t) = \dot{m}(t) = 0 \\ \dot{p}(t) = a_p p(t) + b_p \alpha(t) \end{cases}$$

(a) Flow equations in S and S^+ (the clutch is on).

$$\begin{cases} \dot{n}(t) = a_L n(t) + b_L T(t) \\ \dot{\theta}(t) = 6n(t) \\ \dot{T}(t) = \dot{\phi}(t) = \dot{m}(t) = 0 \\ \dot{p}(t) = a_p p(t) + b_p \alpha(t) \end{cases}$$

(b) Flow equations in S_L and S_L^+ (the clutch is released).

The powertrain dynamics depends both on the crankshaft

TABLE II

INITIAL CONDITIONS OF THE MODEL

$$n(0) \in [799, 801] \qquad n_0 = 800 \text{ rotation per minute}$$

$$\theta(0) = 0^\circ \qquad T(0) = -\frac{a}{b} n_0 = 12.81 \text{ Nm}$$

$$\phi(0) = 20^\circ \qquad m(0) = \frac{T(0)}{G} = 9.286e^{-5} \text{ Kg}$$

$$p(0) = \frac{m(0)}{k} = 7300 \text{ Pa} \qquad \alpha(0) = \frac{a_p}{b_p} p(0) = 7.02^\circ$$

speed, n, and the angle, θ, between the current crankshaft's position and the position of the previous dead center. This relation can be modeled by $\dot{n}(t) = an(t) + bT(t)$ and $\dot{\theta}(t) = k_n n(t)$, where the powertrain dynamics parameters a and b depend on the position of the clutch that can be either open (in state on) or closed (in state off). For this reason it is necessary to distinguish between two different systems: the former models the powertrain when the clutch is on, while the latter describes the powertrain when the clutch is released. The overall engine model and the corresponding flow equations and initial conditions are reported in Fig. 4, Table I and II respectively. This engine model was used to verify the correct behaviour of the proportional controller on the throttle valve control input, described in Fig. 5.

The proposed algorithm was implemented as part of a revised version of Verishift, a software package for reachability analysis of linear hybrid automata written by O. Botchkarev and S. Tripakis (see [7]), based on Botchkarev's original algorithm[1]. Although Botchkarev's algorithm reported as reachable some "bad" regions of the controlled system, the new algorithm could prove that such regions cannot be reached.

VII. CONCLUSIONS AND FUTURE WORK

In this report we described a new algorithm for reachability analysis that yields a tighter approximation than Botchkarev's algorithm, from which it derives. The new algorithm avoids the approximations caused by the union operation, and so it can prove that a region is unreachable by a given hybrid automaton, also when Botchkarev's algorithm returns otherwise, due to the over-approximations of the union operation (declaring reachable states which are not). Moreover, we introduced the notions of ϵ_H-*expansive automaton*, *approximation index of a calculus method* and $(\mathcal{C}, \delta)$-*compatibility*, to characterize a class of hybrid automata for which termination of the two algorithms can be proved. We also defined the *algorithmic approximation index* to measure the quality of reachability analysis algorithms, showing that the new algorithm has a better index than the original one. Finally, we tested both algorithms on a real life problem, proving with the new one the correct behaviour of a controlled car engine that could not be verified with the previous algorithm.

[1]The modified version of Verishift and models presented in section VI are available at *http://fsv.dimi.uniud.it/papers/improving_EC2004*

Future work includes establishing when to use either the polyhedral or the ellipsoidal calculus based on the ideas presented in this paper, and designing a reachability analysis procedure that approximates a given set with more than one calculus method, choosing the best one according to the different shapes of the set's boundary. The correct choice would decrease the approximation index of the calculus method and the algorithmic approximation index.

Acknowledgments. The authors thank O. Botchkarev and S. Tripakis for providing the source code of Verishift and S. Tripakis for technical discussions.

REFERENCES

[1] 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*, ser. Lecture Notes in Computer Science, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds., vol. 736. Springer-Verlag, 1993, pp. 209–229.

[2] O. Maler, Z. Manna, and A. Pnueli, "From timed to hybrid systems," in *Real-Time: Theory in Practice*, J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, Eds., vol. 600. Springer-Verlag, 3–7 June 1991, pp. 447–484.

[3] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, Las Vegas, Nevada, 29 May–1 June 1995, pp. 373–382.

[4] N. Halbwachs, Y.-E. Proy, and P. Raymond, "Verification of linear hybrid systems by means of convex approximations," in *Static Analysis Symposium*, B. Le Charlier, Ed. Springer-Verlag, 1994, pp. 223–237.

[5] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *HSCC*, ser. Lecture Notes in Computer Science, N. A. Lynch and B. H. Krogh, Eds., vol. 1790. Springer-Verlag, 2000, pp. 202–214. [Online]. Available: citeseer.nj.nec.com/kurzhanski00ellipsoidal.html

[6] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *HSCC*, ser. Lecture Notes in Computer Science, N. A. Lynch and B. H. Krogh, Eds., vol. 1790. Springer-Verlag, 2000, pp. 73–88.

[7] O. Botchkarev, "Ellipsoidal techniques for verification of hybrid systems," 2000. [Online]. Available: http://fsv.dimi.uniud.it/downloads/vshift.pdf

[8] A. Balluchi, L. Benvenuti, M. D. D. Benedetto, and A. L. Sangiovanni-Vincentelli, *Nonlinear and Hybrid Systems in Automotive Control*. Springer-Verlag, 2002, ch. Idle Speed Control Synthesis using an Assume–guarantee Approach, pp. 229–243. [Online]. Available: http://www.parades.rm.cnr.it/∼balluchi/papers/ps/H2C.pdf

[9] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, G. M. Miconi, U. Pozzi, T. Villa, H. Wong-Toi, and A. L. Sangiovanni-Vincentelli, "Maximal safe set computation for idle speed control of an automotive engine," in *HSCC*, ser. Lecture Notes in Computer Science, N. A. Lynch and B. H. Krogh, Eds., vol. 1790. Springer-Verlag, 2000, pp. 32–44.

[10] T. A. Henzinger and P.-H. Ho, "Model checking strategies for linear hybrid systems," in *Proc. Workshop on Hybrid Systems and Autonomous Control*, Ithaca, NY, 1994.

[11] R. Alur, S. Kannan, and S. L. Torre, "Polyhedral flows in hybrid automata," in *HSCC*, ser. Lecture Notes in Computer Science, N. A. Lynch and B. H. Krogh, Eds., vol. 1790. Springer-Verlag, 2000, pp. 5–18. [Online]. Available: citeseer.nj.nec.com/alur99polyhedral.html

[12] A. Casagrande, A. Balluchi, L. Benvenuti, A. Policriti, T. Villa, and A. L. Sangiovanni-Vincentelli, "Improving reachability analysis of hybrid automata for engine control," Department of Mathematics and Computer Science at University of Udine, Udine, Italy, Tech. Rep., February 2004. [Online]. Available: http://fsv.dimi.uniud.it/papers/improving_EC2004

[13] N. A. Lynch and B. H. Krogh, Eds., *Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, Pittsburgh, PA, USA, March 23-25, 2000, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1790. Springer-Verlag, 2000.