State Identification In The Hybrid Automata Description Of Dynamical Systems

ISABELLA KOTINI, GEORGE HASSAPIS
Dept. of Electrical and Computer Engineering
Aristotle University of Thessaloniki
54006, Thessaloniki
GREECE

Abstract: Hybrid automata can be used to describe formally dynamical systems with discrete and continuous components. Looking for the existence of certain states in the reachability tree of an automaton is a way of verifying the satisfaction of behavioural modes of the system that can be related to these states. The method of reachability tree construction attempts to compute the set of reachable states by successive approximation, starting from a set of initial states and adding new reachable states until convergence to a state is achieved (Post Procedure). In some special cases the reachability construction does not terminate and as a result it fails to find the reachable region of a hybrid automaton. To solve this problem the reachability verification method has been proposed. According to this method the user makes a guess for a region of reachable states, by studying the results of the Post Procedure. To avoid guessing for a reachable region in such cases, a new method is proposed in this paper, for finding algorithmically the reachable region, which the required state belongs to. The duration δ that the system stays at each location of the automaton is estimated and the relationships that exist between the successive values of each duration are found. As a result, each variable in each location is expressed in terms of duration. Then, from the relationships found among the values of duration, an arithmetic progression is derived expressing each location variable in terms of its initial state and of another variable i, the values of which are natural numbers. This last variable i, represents the number of iterations of the Post Procedure. The use of the proposed method is demonstrated by its application to the automaton that describes a water-tank controller.

Key-Words: formal methods, hybrid automata, real-time, specification, verification.

CSCC'99 Proceedings: - Pages 3131-3135

1 Introduction

Hybrid automata are generalised finite state machines, which can be used for analysing the behaviour of systems involving mixed continuous and discrete evolutions of the variables [1]. Exploring the state space of the automaton that describes the dynamic operation of a physical system in order to find whether one or more states exist and are reachable can base this analysis.

Two methods have been reported in the literature for examining whether a state can be reached from an initial state. The first is the reachability tree construction [4]. It attempts to compute the set of reachable states by successive approximation, starting from a set of initial states and adding new reachable states until convergence to a state is achieved. The second is the reachability verification according to which the user makes a guess for a region of reachable states that may contain the required one, and verifies its existence by the use of a theorem prover [4]. The first method cannot find the required state when the reachability tree does not terminate and the second might require a large number of trials until the right guess is made.

The scope of this paper is to present a different method that finds the reachable region, instead of making arbitrary guesses. This method establishes relationships between values of duration in each location of the hybrid automaton. As a result, the reachable region is expressed in the form of a logical formula, which is true for the reachable states.

The paper is organised as follows. In Section 2, the subsection 2.1 and the subsections 2.2 and 2.3 the linear hybrid automaton model, the semantics of hybrid automata, and the existing verification methods are presented respectively. In Section 3 the proposed method for finding the reachable region is presented and through an example it is demonstrated its application to

the verification of the occurrence of a specific state. For this example the method of the reachability tree construction fails.

2 Linear Hybrid Automata And Reachability Analysis

Hybrid automata [1], are finite automata enriched with a finite set of real-valued variables. In each location, the variables evolve continuously according to differential equations, as long as the invariant of the location is true, then when the guard of a transition becomes true, the control may proceed to another location. Before this, it resets some variables to new values.

In figure 1, a hybrid automaton is shown, which models a computer-based controller that opens and shuts the overflow of a water tank. The automaton has two

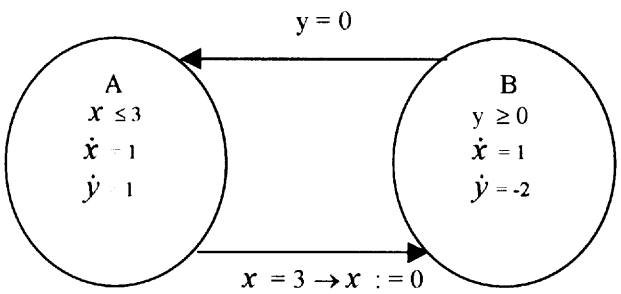

Figure 1. The water-tank automaton.

variables x and y and two locations A and B. The variable x represents a clock of the water-level controller and the variable y represents the water level in the tank. The derivative of x is always 1 because it is a clock. In location A the following controller action is described. When the out flow of the water tank is shut the water level increases 1 inch per sec ($\dot{y} = 1$). In location B the described action is the following. When the outflow of the water tank is open the water level decreases 2 inches per sec ($\dot{y} = -2$). The transition from B to A represents the shutting down of the outflow when the water tank is emptied. The guard y = 0 on the transition ensures that the transition may be taken only when the water level is 0; the constraint $y \geq 0$ on B ensures that the transition to A must be taken before the water level becomes negative. The transition from A to B (open the outflow) is taken every 3 sec. The transition is taken whenever x = 3, and the transition restarts the clock x at 0. If the automaton is started from the state A $\wedge$ x = y = 0, then figure 2 shows how the water level y changes as a piecewise-linear function of time.

2.1 Syntax of linear hybrid automata.

A convex linear predicate is a system of linear inequalities of variables. A linear predicate is a finite disjunction of convex linear predicates. A linear hybrid automaton consists of the following elements:

- a finite set $X = \{x_1, x_2, x_3, \ldots, x_n\}$ of *variables;*
- a finite set L of locations;
- a finite multiset of transitions $T \subseteq L \times L$;
- for each location $\ell \in L$;
  - an invariant Inv(ℓ), which is a convex linear predicate on the variables;
  - an activity Act(ℓ), which is a tuple of differentials laws of the form $\dot{x} = A(\ell, x)$. The $A(\ell, x)$ is called also slope of he variable x at location ℓ;
  - an initial condition Init(ℓ), which is a convex linear predicate on the variables;
- for each transition $T \in T$:
  - a guard Guard(t), which is a convex linear predicate on the variables;
  - a reset Reset(t) , which is a set of variables Reset(t) $\subseteq X$.

Semantics: A valuation is a function υ: $X \xi \mathbf{R}$ that associates a real number $\upsilon(x)$ to each variable $x \in X$. Given a variable valuation υ and a linear predicate P

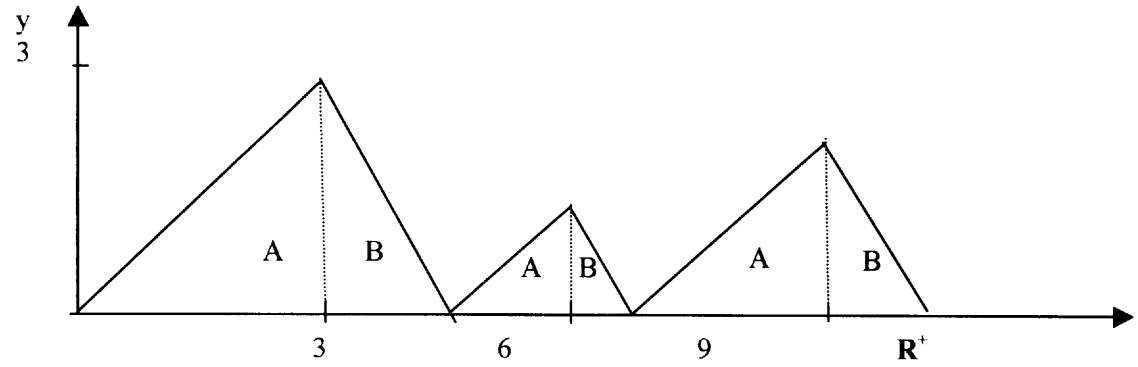

Figure 2. A run of the water-tank automaton.

over the variables, υ satisfies P, written $P(\upsilon)$ = true, if by replacing in P each variable x with its value $\upsilon(x)$, we can obtain a true statement. Given a valuation υ, a location $\ell \in L$, and a non-negative real $\delta \in \mathbf{R}^+$, we write $\upsilon +_\ell \delta$ for the valuation that assigns to each variable x in X the value $\upsilon(x) + A(\ell, x) \cdot \delta$.

So the semantic features of a hybrid automaton are:

- a state is a pair (ℓ, υ), where ℓ is a location and υ is a valuation satisfying the $Inv(\ell)(\upsilon)$ = true;
- for a non-negative real $\delta \in \mathbf{R}^+$, there is a continuous step of duration δ between two states (ℓ, υ) and (ℓ, υ') denoted $(\ell, \upsilon) \xrightarrow{\delta} (\ell, \upsilon')$, if $\upsilon' = \upsilon +_\ell \delta$;
- for a transition $\alpha = (\ell, \ell') \in T$, there is a discrete step of a label α between two states (ℓ, υ) and (ℓ', υ') denoted $(\ell, \upsilon) \xrightarrow{\alpha} (\ell', \upsilon')$, if Guard (t)($\upsilon$) = true and $\upsilon' = \upsilon[Reset(t) := 0]$;
- a run is a finite sequence of continuous and discrete steps $(\ell_0, \upsilon_0) \rightarrow (\ell_1, \upsilon_1) \rightarrow \ldots \rightarrow (\ell_m, \upsilon_m)$ such that the first state (ℓ_0, υ_0) is initial.

On the basis of the above the two verification method are briefly presented in the next section.

2.2 Reachability construction

This method performs a symbolic execution of the hybrid automaton. This approach consists in successively approximating the reachable region, starting from the initial region, and iterating successor operations until the computation converges. The continuous successor of a region is the region that contains all the states that can be reached from states, by a single continuous step. Suppose that variable x evolves in location ℓ according to the type $x' = x +_\ell \delta$. It can be shown that the elimination of the duration δ can be performed, and it again produces a linear predicate in variable.

Reachability construction consists in iterating the following Post procedure:

Input: set A of linear regions.

Output: set B of linear regions, initially empty.
For each linear region $<\ell, P>$ in the set A, for each transition (ℓ, ℓ') with the origin ℓ:

- let P_1 be the intersection of P with the guard of transition (ℓ, ℓ')
- let P_2 be the projection of P_1 over transition (ℓ, ℓ')
- let P_3 be the intersection of P_2 with the invariant of ℓ'
- let P_4 be the extension of P_3 at state ℓ'
- let P_5 be the intersection of P_4 with the invariant of ℓ'
- add $<\ell', P_5>$ to set B

The $Post^k(I)$ is the set of regions by applying k times the post operation to the set of initial regions where I denotes the set of initial region. The reachable region $Post^*(I)$ is the countable infinite union

$$\bigvee_{k \in N} Post^k(I)$$

If for some $k \in \mathbb{N}$, it is the case that

$$Post^{k+1}(I) \subseteq Post^k(I)$$

then reachability construction terminates in finitely many steps. This does not happen in general for linear hybrid automata.

2.3 Reachability Verification

Reachability Verification method [4] can succeed in cases when reachability construction fails. Reachability verification consists of two steps: first, to guess the reachable region; second to verify that the guess is correct. A suitable guess can be fount using simple heuristics, and the verification can be performed by induction, using a theorem prover. It appears that when reachability construction does not terminate, the reachable region of a hybrid automaton can still behave in a regular manner.

A typical situation is to guess a reachable region written using only one natural-number variable μ that represents the number of iterations of the Post procedure. In this case, the guessed region is called directly inductive, and if for all $\mu \in \mathbb{N}$, $Post^\mu(I) = R(\mu)$, where $R(\mu)$ is the reachable region involves some new variables (in this case the variable μ), then the guessed region is correct. In other situations the guessed region may not be directly inductive, but it can be made so by introducing new variables (one of which represents the iteration number) and constraints connecting the existing and the new variables. Finally, even when a guess is not directly inductive, it can be useful (as an invariant of the system) to prove safety properties.

In the water-tank automaton (figure 1) a suitable guess is the following [3]:

For $i \geq 0$,

- $S_{2i+1} = (B, (y + 2x = 2 + \frac{(-1)^i}{2^i}) \wedge (y \geq 0))$
- $S_{2i} = (A, (x - y = 1 + \frac{(-1)^{i+1}}{2^i}) \wedge (x \leq 3))$

A linear hybrid automaton is additive-inductive [4], if its reachable region can be expressed as a set of pairs $\{<\ell, P\ell> \mid \ell \in L\}$ such that for each location $\ell \in L$, $P\ell$ is a formula of the theory $(\mathbb{R}, \mathbb{N}, +, \leq)$ in which all natural-number variables are outermost existentially quantified.

3 The Proposed Method Of Computing The Reachable Region

In the above semantics has been mentioned that there is a duration δ between two successive states. In the reachability construction approach, the duration is eliminated by producing a linear predicate in variables $x_1, x_2, x_3, \ldots, x_n$

It is observed that the values that the duration of each location takes at different iterations satisfy a certain relationship. Furthermore, each variable can be expressed in each location as relations of values of the duration. In this way, the reachable region can be found which involves a quantifier over a new natural-number variable i (where i is a counter that is increased by 1 after a full turn of an iteration). Then, as it has been described in the reachability verification approach, the identified reachable region can be verified with the theorem prover PVS [5].

Our method is restricted to additive-inductive hybrid automata, because observations on a number of studied cases have disclosed that the sums of the duration values follow an arithmetic progression. By using the procedure for the reachability construction, which has been already described in section 2.2, however retaining the δ duration (without eliminating it), the following approach is proposed for finding the relationship between successive duration values.

1. during the first iteration the maximum value of duration for each location is found;
2. the relation for each location is found that holds between values of duration;
3. if these relations remain the same for a number of iterations, it is assumed that the generalised form of relations is the same for future iterations in the same location;
4. as a result of the above assumptions, in each location each variable is expressed on the basis of values of duration and on initial state;
5. finally, arithmetical progressions are used to transform the above expressions and to obtain the desired reachable regions.

The automaton of the control system of figure 1 belongs to the additive-inductive classification and the reachability construction does not terminate. In order to check whether at the timing instant x=3 the level of the tank will be at y=2 we need to verify that the automaton state $T = (A \wedge x = 3 \wedge y = 2)$ can be reached from the initial state $S = (A \wedge x = y = 0)$.

The application of the above method leads to the following results:

Step 1:

In the first iteration of each location, the maximum duration in location A is 3 and in location B is 3/2. Then, in location A the following relationship will hold:

- $x - \dot{x}\delta_{11} = x_0 = 0 \Rightarrow x - \delta_{11} = 0 \Rightarrow x = \delta_{11}$
- $y - \dot{y}\delta_{11} = y_0 = 0 \Rightarrow y - \delta_{11} = 0 \Rightarrow y = \delta_{11}$

A transition from location A to B will occur when the value of variable x becomes 3. Consequently, the maximum value of duration δ_{11} is 3.

In location B, it will hold:

- $x - \dot{x}\delta_{21} = x_0 = 0 \Rightarrow x - \delta_{21} = 0 \Rightarrow x = \delta_{21}$
- $y - \dot{y}\delta_{21} = y_0 = 3 \Rightarrow y + 2\delta_{21} = 3 \Rightarrow y = 3 - 2\delta_{21}$

A transition from location B to A when y becomes zero. Therefore, the maximum value of δ_{21} is 3/2.

Step 2:

After five iterations, we observe that the relations between values of duration remain the same, as the followings:

$\delta_{11} = 2\delta_{21}$ $\quad$ $\delta_{12} + \delta_{21} = 3$

$\delta_{12} = 2\delta_{22}$ $\quad$ $\delta_{13} + \delta_{22} = 3$

$\delta_{13} = 2\delta_{23}$ $\quad$ $\delta_{14} + \delta_{23} = 3$

$\delta_{14} = 2\delta_{24}$ $\quad$ $\delta_{15} + \delta_{24} = 3$

Step 3:

After five iterations we assume that the above relationships remain the same for each location. Expressing them in a general form we have:

for $i \geq 1$

$$\delta_{1i} = 2\delta_{2i}$$

$$\delta_{1i} = 3 - \delta_{2i-1}$$

The $\delta_{\ell i}$ denotes the duration in location ℓ where i is the number of iterations. From the above equations one can derive that:

$$\delta_{2i} = \frac{3 - \delta_{2i-1}}{2} \qquad (1)$$

Step 4:

Taking into consideration the guard condition for the transition from A o B one can easily see that:

$$x = \delta_{2i-1} + \delta_{1i} \text{ and } y = \delta_{1i}. \qquad (2)$$

In location B the variables are:

$$x = \delta_{2i} \text{ and } y = \delta_{1i} - 2\,\delta_{2i}. \qquad (3)$$

So in location A from equations (2):

$$x = \delta_{2i-1} + y$$

and in location B from equations (3):

$$y = \delta_{1i} - 2x.$$

Although these expressions include the initial state as was mentioned in step 4 it is not obvious as $x = y = 0$.

Step 5:

The equation (1) can be rewritten as:

$$\delta_{2i} = 1 - \frac{(-1)^i}{2^i}$$

As a result in location B we obtain the following equation:

$$y + 2x = 3 - (1 - \frac{(-1)^i}{2^i}) = 2 + \frac{(-1)^i}{2^i}$$

and in location A :

$$(x = y + 1 - \frac{(-1)^i - 1}{2^{i-1}} \wedge x \leq 3),$$

because $x \leq 3$ is an invariant in location A.

Therefore, the reachable region will be:

- $S_{2i} = (B, y + 2x = 2 + \frac{(-1)^i}{2^i})$
- $S_{2i-1} = (A, x = y + 1 - \frac{(-1)^i - 1}{2^{i-1}} \wedge x \leq 3)$

From the above it becomes obvious that the final state T can not be reached from the initial state S. Therefore, the way the tank level is controlled can not guarantee the stabilisation of the tank level at the value of $y = 3$.

4 Conclusion

A new approach to the problem of identyifying the existence of specific states of hybrid automata has been presented. The method is applicable even in cases where the reachability tree of the automaton does not terminate. Compared to other methods, this method eliminates the need to make a guess for a reachable region. The idea is to trace algorithmically a reachable region of a hybrid automaton by using the values of duration δ in every location and finding the relations between them.

This method has been applied to other applications of hybrid automata, as the example with the leaking gas burner automaton [1], the bouncing-ball automaton [3], and chemical reactor-controller automaton [2].

References:

[1] Alur R., C. Courcoubetis, N. Halbwachs, T.A. Henziger, P-H Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, Vol.138, 1995, pp. 3-34.

[2] Hassapis G., I. Kotini, Z. Doulgeri, Validation of a SFC Software Specification by Using Hybrid Automata, In *9th Symposium on Information Control in Manufacturing*, 1998.

[3] Henzinger T. A, and Pei-Hsin Ho, A Note on Abstract Interpretation Strategies for Hybrid Automata, In Hybrid Systems II (P. Antsaklis, W. Kohn, A. Nerode, s. Sastry, eds), Lecture Notes in Computer Science 999, 1995, pp. 252-264.

[4] Henzinger A.T, V. Rusu, Reachability Verification for Hybrid Automata, *International Workshop on Hybrid Systems: Computation and Control (HSCC' 98)*, Berkeley (California, USA), LNCS 1386, 1998, pp. 190-204.

[4] Owre S., S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas, PVS: Combining specification, proof checking and model checking, In *Proc. of the 8th Conference on Computer-Aided Verification,* CAV' 96, LINCS 1102, 1996, pp. 411-414.