CEAI, Vol. 4, No. 4, pp. 47-56, 2002 Printed in Romania

MODELING AND ANALYSIS OF DISCRETE-TIME HYBRID SYSTEMS

Monika KUROVSZKY and Hassane ALLA

Laboratoire d'Automatique de Grenoble (UMR 5528)
INPG-UJF-CNRS - B.P. 46
38402 Saint Martin d'Hères – France
Monika.Kurovszky@inpg.fr, Hassane.Alla@inpg.fr

Abstract: *The behavior of hybrid systems is described by interacting discrete and continuous variables. In this paper, we suggest a modeling and analysis approach based on the discrete-time representation of the continuous dynamics of hybrid systems. Initially, the hybrid system is modeled using a discrete-time hybrid automaton. Starting from this model, by the analysis approach an equivalent timed automaton model is obtained. The approach that we propose here is a generalization of the "Clock Translation" method developed by Henzinger for a particular class of hybrid systems. This model, representing the result of our analysis approach, is built in the aim of a further supervisory control synthesis .*

Keywords: *Hybrid systems, modeling, Discrete-Time Hybrid Automaton, reachability analysis.*

1. INTRODUCTION

The behavior of hybrid systems is described by interacting discrete and continuous variables. Recently, the hybrid systems have attracted considerable research attention in the control and computer theory communities. The major motivation for the study of hybrid systems is given by the inadequacy of models based on continuous mathematics for describing certain classes of complex systems. There were attempts of mathematical formalization but in all these attempts, the interaction between the continuous and discrete dynamics were represented as differential equations with discontinuities. The reason for this is given by the existence of a rich theory of continuous systems and the lack of similar theory in the discrete side.

A look at the literature lets us to say that there are a number of theoretical frameworks developed: 1) to model hybrid systems ([1],[8]), 2) analyze their behaviors ([2],[4],[6]) and, 3) synthesize controllers such that the closed loop system satisfies certain specifications ([3], [5]). The major parts of these frameworks treat the case of hybrid systems having uncoupled continuous dynamics.

By our work, we address a class of hybrid systems having any continuous dynamics.

In this paper we propose an approach for modeling and analysis of hybrid systems in view of supervisory control synthesis. The approach is based on the use of discrete-time

representation for the continuous and hybrid automaton modeling the interactions between discrete and continuous dynamics of the hybrid system.

The steps that will be followed in order to obtain the timed equivalent model of the initial hybrid automaton are the following: (1) the continuous dynamics of the systems are sampled in order to obtain a discrete-time representation for this ones; the resulting model is called *discrete-time hybrid automaton* and (2) in order to ensure that the modeled system has specified the appropriate transition guards, some verifications must be performed, representing the goal of the analysis approach.

Generally, the hybrid systems analysis, modeled by hybrid automaton, is based on the reachable state space regions computation starting from a given initial condition. Hence, firstly, we must check if all the evolutions modeled by the hybrid model are actually possible in the system. To answer at this question implies to perform a *reachability analysis* of all the locations of the model.

The *reachability problem* can be formulated as follows: having a hybrid system and an initial state space region, find the state space regions that can be reached by the system evolution starting from this initial condition. In the case of hybrid automaton models, this problem can be restricted to check the reachability of guard conditions associated with the transitions of the model. The results of this analysis will be used in order to perform modifications in the initial model. Namely, if there are some guard conditions that are never reached (i.e., transitions that are never fired) by the system evolution, then the initial model can be simplified deleting all the transitions labeled by these guards. The resulting model is given by a *reachable automaton* modeling only the possible evolutions of the system starting from a given initial condition.

Therefore, the reachability analysis states the existence problem of one trajectory, which drive the system from a given initial region R_0 into a final region R_F by the system dynamics.

The analysis approach that we propose here is a generalization of an analytical method named "clock translation", which can be used only for a very restricted class of hybrid systems. Given the complexity of the continuous dynamics, in our case, only a numerical resolution is possible. Therefore, the generalization that we propose make possible to build an equivalent timed model of initial hybrid automaton model, passing trough numerical computation.

The rest of the paper is organized as follows: in section 2, we introduce the minimally necessary definitions concerning discrete and continuous behaviors and hybrid automata and we introduce the discrete-time hybrid automaton model. In section 3, the proposed generalization of the Clock Translation method is presented. In section 4, we present our analysis method. Throughout all these sections, theoretical examples illustrate the steps of our approach.

2. DISCRETE-TIME HYBRID SYSTEMS

Generally, hybrid systems are systems having theirs evolutions determined by interacting discrete and continuous dynamics. There are many approaches to modeling, analysis and synthesis of hybrid systems. Approaches differ with respect to emphasis on the complexity of the continuous and discrete dynamics.

The hybrid systems of interest in this paper are represented by systems having the evolutions limited by theirs continuous specifications.

2.1. *Generalities*

The plant: Throughout our approach, we address a class of applications represented by hybrid systems in which the continuous dynamics are described by differential equations $\dot{x} = f(x(t), u(t), t)$, where $x(t) \in \Re^n$ denotes the vector of the continuous state space variables for the considered system and u(t) is the vector of external inputs.
The dynamical evolution of hybrid system is made by commutations between their discrete states and/or between their continuous dynamics. These commutations represent the discrete aspect of the behavior of hybrid systems and theirs are generated by *occurrence* of external or internal discrete events whose nature can be controllable or not.

The specifications introduce restrictions on the plant dynamics. In the case of hybrid systems, the specifications can be divided in two parts: specifications for the continuous part and specifications for the discrete part.
The *continuous specification* is introduced as constraints for the continuous variables x(.) which restrict theirs evolutions to stay in a specified state space region, named *desired*

region. Any outgoing evolution from this region is not desirable.
The discrete part can be viewed as a finite state machine. Usually, the *discrete specifications* are given as a logical condition describing for example the order of occurrence for the events in the plant, during its functioning.

The hybrid automaton represents the modeling tool that lets us to take in account all these aspects in the same structure. In the further paragraph of this paper, this formalism will be introduced.

2.2. *Hybrid automaton*

Generally, a hybrid system can be modeled by a set of systems with a continuous dynamic, which interacts, with one or several discrete event systems. Alur *et al*. (1993) introduce the framework of hybrid automata as a model and specification language for hybrid systems. In this model, at a given instant, the system can be in one of several discrete states in each of which its behavior is governed by a distinct continuous dynamical law.

The hybrid automaton can be viewed as an extension of a discrete automaton augmented with continuous variables whose dynamics in each discrete state are defined by differential equations. Transitions between the states are enabled by conditions on the values of these variables.

In order to define formally the hybrid automaton, an adaptation of definitions presented in [1], [8] and [9] is made, in the aim to relate these definitions to those used in the control literature. The result of this interpretation is presented below:

Definition 1. A *hybrid automaton* is a tuple A = (Q, X, *flow*, *inv*, *guard*, *Aff*, *init*) such that:

- $Q = \{q_1, q_2, ..., q_m\}$ a finite set of discrete states;
- $X \subseteq \Re^n$ is the continuous state-space. Elements of X are written $x = [x_1\, x_2 ... \, x_n]^T$;
- *flow*(q) is a function that assigns a continuous evolution to each discrete state. While the hybrid automaton stays in the discrete state q, the evolution of the continuous variables is governed by the differential equation

$$flow(q): \dot{x}(t) = f(x,u,t) \quad (1)$$

- *inv*(q) is a function that assigns with each location q, a condition that must be satisfied by the continuous variables if the automaton is to stay in the discrete state. The system can stay in the discrete state as long as the invariant associated with it is satisfied;
- *guard*(T) is a transition labeling function that assigns firing conditions with the transitions of the automaton;
- *Aff*(T) is a function that associates with each transition of the automaton one relation that allows to actualize the value of the continuous state space variables after the firing of a transition T. This function is specified as simple predicates such $x_i := c_i \mid x_i \in X$, where c_i is a constant value from X;
- *init* is a function which assign an initial state $x_0 \in X$ with the initial discrete state $q_{in} \in Q$.

At any time, the state of the hybrid system specifies a location and values for all continuous variables. The state can change in two ways: by an instantaneous transition that changes the entire state and by elapse of time that changes only the values of continuous variables in a continuous manner according to the dynamic *f*(q) of the current location. The system can evolve in a discrete state q only if the current continuous state satisfies the invariant condition associated with this state. These conditions can arise from constraints imposed by the physical systems or decisions in system design.

Intuitively, the states of the hybrid automaton allow differentiating the continuous dynamics of the system. The state of a hybrid system is characterized at any time instant by the couple (q_i, x) representing the *global state* of the system.

Definition 2. The *global state* of the system at the time instant t is given by the current discrete state $q_i \in Q$ and the current value of the state vector $x \in \Re^n$ at the considered time instant.

Graphically, the hybrid automaton can be represented by a directed graph whose vertices represent the locations and arcs represent the transitions (Fig. 1).
The invariant conditions and the differential equations, which model the continuous evolutions, can be written inside the vertices. With the arcs are associated guards and transition relations.

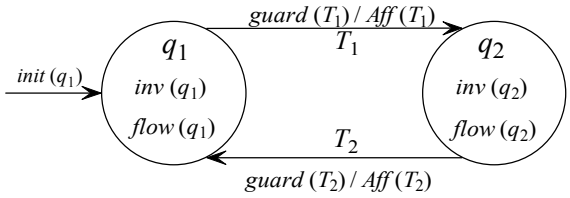

Fig. 1. Hybrid automaton model.

The function describing the continuous dynamics of each location is given by the linear differential equation:

$$\dot{x} = \mathrm{A}x + \mathrm{B}u \text{ , (where } x \in \Re^n \text{)} \quad (2)$$

which represents the state space representation for the continuous dynamics and where $x \in \Re^n$ denotes the vector of the continuous state space variables.

2.3. *Discrete-time hybrid automaton*

In order to compute the trajectories generated by systems modeled as a hybrid automaton numerical computation are performed using the computation of digital processors. Therefore, the computation is performed using a digitized model of the continuous dynamic. The digitized model is obtained by discretization into fixed intervals of the original continuous one. The sampling period is chosen so that the Shannon's theorem is satisfied.

In the aim of a reduced complexity of the calculus, in all steps of our approach we use a *discrete-time representation* for the continuous dynamics of the system. Therefore, the continuous dynamics are modeled by the equation:

$$x_{k+1} = A_d x_k + B_d u_k \qquad (k = \overline{0,n}; k \in \mathrm{N}) \quad (3)$$

where: x_k represents the vector of continuous variables at k^{th} sampling time, A_d and B_d represent the discrete equivalent of matrix A and vector B used in the continuous state representation for the continuous dynamics. The sampling period is dictated by the nature of the continuous dynamics and can be chosen respecting the conditions imposed by the Shannon theorem.

Let us to define now the *discrete-time hybrid automaton*.

Definition 3. The definition of the *discrete-time hybrid automaton* is similar to the definition of the hybrid automaton (Def. 2), except that a discrete-time model (relation 3) gives the evolution of the continuous variables.

In the *discrete-time hybrid automaton*, the discrete-state transitions can occur only at valid sampling times. We assume that the transitions occur immediately when a guard condition is satisfied. We have transformed an asynchronous-synchronous model given by the *continuous-time hybrid automaton* into a synchronous model given by the *discrete-time hybrid automaton*. We suppose that the modeling errors carried by this transformation are sufficiently small and negligible. This problem does not constitute the goal of this paper and is well studied in [11].

In the further steps, the discrete-time hybrid model will be is used

3. GENERALISATION OF THE "CLOCK TRANSLATION" METHOD

The "Clock Translation" method proposed in [9] can be used for a restricted class of hybrid systems. The method consists in the construction of a timed bisimilar equivalent of the hybrid automaton, by replacing all the continuous variables by clocks.

For a hybrid automaton A, the variable t is a *clock* if t is linear and all flow conditions of A imply $\dot{t} = 1$. We can replace a continuous variable x by a clock t_x if at any time the value of x can be determined uniquely from the value of t_x.

3.1. *Clock Translation*

In order to apply the method proposed in (Henzinger *et al*., 1998), the variable x must satisfy some *solvability conditions*:

- the variable x must be independent of the other variables in flow, jump, initial and final conditions;
- the variable x follows a unique flow from any starting point;
- in order to determine the truth of predicates such as $x \geq c$, for any constant $c \in \Re$, from the value of t_x, the flows of x must be strictly monotone;
- the initial value of the current flow of x and the elapsed time must be known at any instant.

If all these conditions are satisfied, the construction of the bisimilar hybrid automaton can be realised. The construction technique is illustrated throughout the following example:

Example 1. Consider the hybrid automaton model illustrated in Figure 2.a., modeling the functioning of a thermostat. From this model, we want to obtain a bisimilar timed automaton applying the steps of the "clock translation" method.

The timed equivalent of the initial hybrid automaton is obtained by analytical resolution of the differential equation describing the continuous evolution of the system in each discrete state. The obtained model is illustrated in the Figure 2.b.

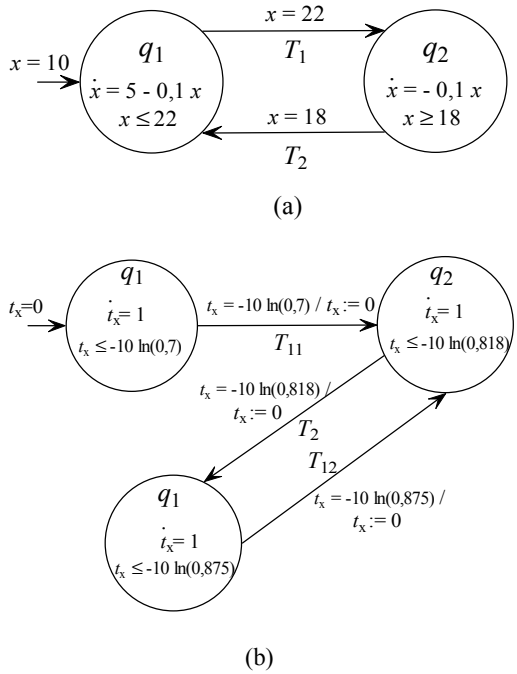

Fig.2. Hybrid automaton modeling a thermostat: (a) hybrid model and (b) timed model.

Therefore, the bisimilar model of the initial hybrid automaton, obtained following the steps of the before presented Clock Translation method, is given as a timed automaton.

It can be concluded that the presented approach is efficient but applies only for a restricted class of continuous variables. This is an analytical method, so the solvability of the continuous variables is always requested.

3.2. *Generalization*

The method that we propose is dedicated to obtain an equivalent timed automaton starting from an initial hybrid. The hybrid systems of interest has any linear continuous dynamics, so the constraints which must be satisfied by the continuous variable x in the case of the Clock Translation method presented before, are not verified.

The principle of our method is illustrated in the Figure 3.

Starting from a hybrid automaton having any dynamics, we propose a reachability analysis approach based on the discrete-time representation of the continuous dynamics of the hybrid systems.

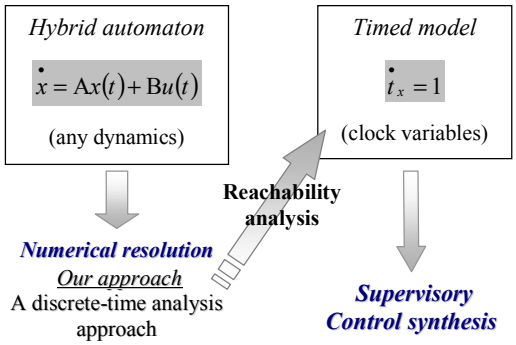

Fig. 3. Generalized "Clock Translation" method.

The approach is based on the numerical computation of trajectory for continuous variables in each location of the hybrid automaton in order to test the reachability of the guard associated with its output transitions. Once the output transition is firable (i.e., when the guard condition is satisfied), the current discrete-time instant is retained as the minimal duration of staying in the current location. The transitions guards are given as continuous state-space regions. So, once the guard is reached by the continuous dynamics, the trajectory of the system in the current location is computed until it becomes no longer reachable or the invariant associated with the current location is no longer satisfied.

The reachability method and the technique letting us to build the equivalent timed model will be detailed in the following section of this paper. The timed model will represent the starting model for supervisory control synthesis.

4. ANALYSIS APPROACH

Having a formal model for hybrid systems and their behaviors, we need methods to check all the possible evolutions of the system. We will adopt an algorithmic verification methodology.

In this paper, we concentrate our interest on finding all trajectories of the system, which reach the state space regions representing the transition labeling conditions. In order to do this, we develop a forward reachability method for the discrete-time hybrid automaton, which models the studied system.

4.1. *Reachable state space computation*

The reachability analysis problem for any discrete systems can be solved since the

transition functions, the initial set, the target set and the reachable states accumulated over the execution are finite and can be represented explicitly. Now, if extended to hybrid systems, the reachability analysis involve the computation of following functions over subsets of the state space of hybrid systems: successors computation, union of sets in order to accumulate the reachable state space regions and intersection in order to check the emptiness or whether the system reaches the target region.

In order to be able to compute these functions, the first ingredient we need is a finite syntactic representation of the state space regions encountered during the reachability analysis.

The continuous state space X of hybrid automata is in $\Re^n$. Hence, unlike the finite-state systems, subsets of X do not admit an enumerative representation and can only be represented symbolically, such us by formulas of some logic.

The *polyhedral sets* as symbolic representation of state space regions of $\Re^n$ are geometrical objects that, from computational point of view, are easier to describe and manipulate. One major characteristic of these objects is stated in the following property:

Property 1. Any polyhedron in $\Re^n$ can be uniquely defined by an ordered list of its vertices.

Example 2. In order to illustrate this property, let us to consider a state space region, like illustrates the Figure 4.

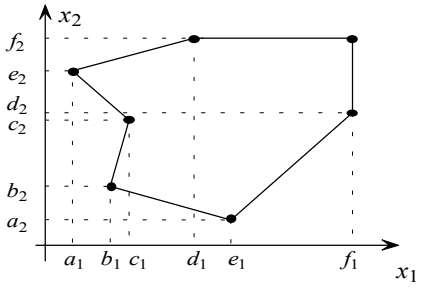

Fig.4. Polyhedral state space region in $\Re^2$.

The considered region is represented in $\Re^2$ and can be described by the ordered set of its vertices. Each element of the set represents a grid point. Therefore, the region can be entirely described by the ordered set: $\{(a_1, e_2), (d_1, f_2), (f_1, f_2), (f_1, d_2), (e_1, a_2), (b_1, b_2), (c_1, c_2)\}$. Hence, we can conclude that this description gives an unique representation for the considered state space region.

In consequence, all the operations on the real state space regions are replaced by operations on their polyhedral representations. Therefore, a polyhedral region will represent the continuous successor at t time instant. Hence, the reached region by the hybrid system evolution starting from a given initial region will be obtained by the union of its continuous successors.

Polyhedral sets can be divided into two types: *convex* and *non-convex*. The former is simpler and allows efficient implementation of intersection, membership and equivalence testing. However, the limitation to use only convex polyhedra, is very restrictive because whenever we make unions of continuous successors in order to obtain the reachable state space region the result is non-convex polyhedra. The approximation of this result by a convex polyhedron might result in a too coarse approximation. Consequently, the use of non-convex polyhedra is necessary in the aim to describe more precisely the reachable state space by the evolution of the system.

In the following, we suppose that convex state space regions give the initial regions and the target regions representing the guard conditions of the hybrid automaton model. We want to build the real trajectory of the systems based on the continuous successors computation.

4.2. *Continuous successors computation*

In the computation of the continuous successors we are based on an important property of linear systems which is stated and proved in the following lines.

Consider a continuous linear system $C = (X, f)$ where $X \subseteq \Re^n$. The continuous dynamic of C is given by

$$\dot{x}(t) = \mathrm{A}x(t) + \mathrm{B}u(t) \tag{4}$$

where: $\mathrm{A} \in \Re^{n\times n}$, $\mathbf{u}(\mathrm{t}) \in \Re^m$ and $\mathrm{B} \in \Re^{n\times m}$. Let us to state the fundamental property of linear systems:

Lemma 1 If $F \subseteq X$ is a *convex region* then for every $t \geq 0$ the image of this region by a linear application f at any time instant t is convex.

Proof: Let ξ_x be the trajectory of C starting from a point $x \in X$. By solution of equation (4), we have:

$$\xi_x = e^{\mathrm{A}t}x + \int_0^t e^{\mathrm{A}(t-\tau)}\mathrm{B}u(\tau)d\tau$$

Let us consider $f(t,F)$ the set of states reachable from F in t time units. Therefore, this set can be written as:

$$f(t,F)=e^{At}F+\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

The matrix exponential function e^{At} is a linear operator. Hence, preserve convexity, which implies that $f(t,F)$ is convex.

We suppose further that convex polyhedral regions represent the initial region and the transition labeling guards conditions. Therefore, based on the *Property* 1, we can conclude that the continuous successor computation is resumed to the computation of trajectories generated by each vertex of the initial region.

The use of discrete-time representation for the continuous dynamics of hybrid systems let us to reduce the complexity of the calculus. Based on the property stated in *Lemma* 1, for each sampling time a convex region will be obtained by continuous successor computation.

Figure 5 illustrates the continuous successor computation in two dimensions, starting from an initial state space region R_0. R_k denotes the continuous successor region reached in k sampling periods from R_0. This region is completely described by its four vertices.

The vertices of R_k are denoted as $succ_k(s_i)$, where $i=\overline{1,4}$. The vertex $succ_k(s_i)$ is obtained by computing the trajectory of the initial vertex s_i. The trajectory of the system satisfies the dynamic described by discrete-time flow equation, $x_{k+1}=A_d x_k+B_d u_k$ in k sampling periods and for $x_0 = s_i$. In this way, all the vertices of the continuous successor region R_k are computed.

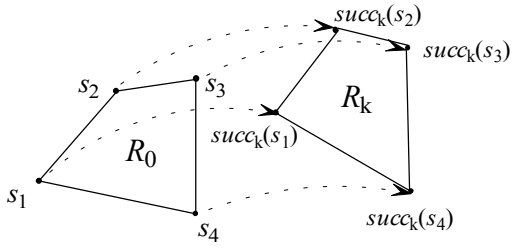

Fig. 5. Successors computation starting from R_0.

The following algorithm synthesizes the steps of the continuous successor computation:

Algorithm 1: *Continuous Successors computation*

Step 1: Initialize the successor region $succ_0$ by R_0;
Step 2: Initialize the counter k:=0;
Step 3: **REPEAT**
3.1. For each vertex of the current region, compute the continuous successor reached by $x_{k+1}=A_d x_k+B_d u_k$ in one sampling period. Symbolically, we denote the continuous successor of the region R_k as $R_{k+1}:=\varphi(R_k)$, where $\varphi(R_k)$ is the function allowing to compute the continuous successors of all the vertices of R_k reached by the elapse of one sampling period.
3.2. Actualize the counter k:= k+1.
UNTIL $R_{k+1}=R_k$.

We can observe that the computation of all the continuous successors of an initial region ends when the convergence of the continuous dynamic in one hybrid location is achieved.

Formally, the convergence of the continuous dynamics is checked by the evaluation of the distance between two continuous successor regions R_k and R_{k+1} of a given initial region. This distance is obtained by the computation for each vertex of R_{k+1} the gap between its vertices and the vertices of R_k. Therefore, we can define the convergence of two regions as follows:

Definition 4. The continuous dynamic in a location of a discrete-time hybrid automaton converge in k sampling periods, if for a given ε, the maximum of distances between the region R_k and all its continuous successors verifies the relation (5) for any $j\in \mathrm{N}, j\geq k$.

$$\max(\delta(R_k,R_{k+1}))\leq\varepsilon \qquad (5)$$

The computation of continuous successors in a hybrid location is illustrated throughout the following example.

Example 3. Let us to consider the hybrid location given in Figure 6.

The invariant $inv(q_i)$ associated with this location limits the evolution of the continuous dynamics in this location.

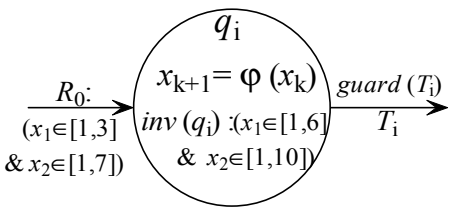

Fig.6. Hybrid location.

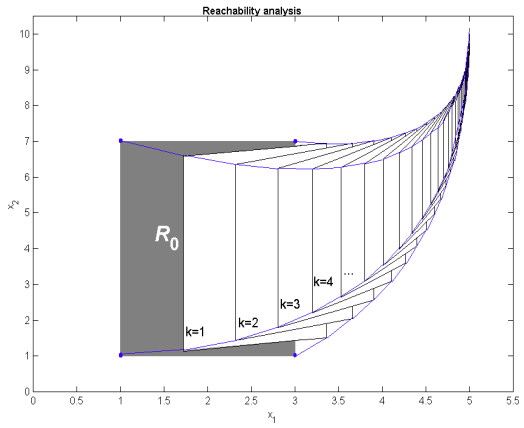

Fig. 7. Continuous successors of R_0.

The discret-time continuous dynamics is given by

$x_{k+1} = \varphi(x_k) = A_d x_k + B_d$, where

$A_d = \begin{pmatrix} 0{,}81 & 0 \\ 0{,}17 & 0{,}9 \end{pmatrix}$ and $B_d = \begin{pmatrix} 0{,}9 \\ 0{,}9 \end{pmatrix}$.

The initial region is represented by R_0. The ordered set of the initial region is {(1,1), (3,1), (3,7), (1, 7)}.

The continuous successors of the initial region R_0 are illustrated in the Figure 7. It can be observed that the convergence of continuous dynamics is done into the point (5, 10). Hence, all continuous successors of the initial state space region was computed.

4.3. *Transition enabling analysis*

In this section we are interested in the study of firability conditions for the transitions of discrete-time hybrid automaton model.

The hybrid automaton evolution is determined by interacting continuous and discrete dynamics.

The difficulty in the analysis is given by this interaction. Therefore, the instants of the commutations must be determined taking into account the continuous evolution of continuous state space variables. Hence, the enabling of a discrete transition is an important property that must be studied.

The principle of verification if an output transition of a currently analyzed location is enabled or not is illustrated in Figure 8. When the intersection between the currently reached region and the target region (transition guard) is not empty, the transition is enabled. The invariant limits the continuous evolution of the system and is used as a stop condition for the continuous successors computation.

Generally, two situations can be encountered:

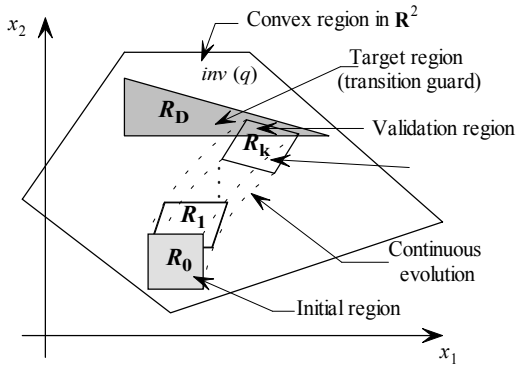

Fig. 8. Continuous dynamics in one hybrid location.

1. The target region is reached and the invariant is also satisfied. Therefore, the transition is can be fired.
2. The target region is not reached but the invariant is no longer satisfied. There is incoherence at the modeling level.

In order to illustrate the transition enabling analysis, let us to consider a hybrid location like illustrates Figure 9, corresponding with the simplest case that can be encountered in a hybrid automaton model. Namely, the case of one hybrid location with a single input and a single output transitions (SI&SOT – *Single Input & Single Output Transitions*).

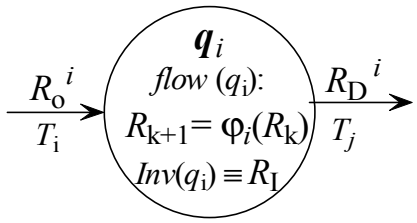

Fig. 9. Single Input Single Output Transitions.

The continuous evolution of the system in the considered location is governed by the equation $R_{k+1} = \varphi_i(R_k)$, where the region R_{k+1} represents the continuous successor of R_k and it will be computed as it was shown above. The *invariant* associated with this location is is denoted by $Inv(q_j) \equiv \boldsymbol{R}_D$. The initial region is represented by the guard associated with the input transition R_0^i and the region for which the reachability must be checked, is given by the guard associated with the output transition R_E^i.

Definition 4. For any hybrid automaton A, the *staying time* δ_i in one hybrid location $q_i \in Q$ is given by the duration of stays of the system in

the location until the output transition is enabled.

Property 2. For any location $q_i \in Q$, an output transition T_j is enabled only if there existe a finite value $\delta_i \in \Re^+$ for which the target region is reached.

The value of $\delta_i \in \Re^+$ depends on the initial condition (i.e., initial region), the enabling condition of the output transition T_j and the trajectory defined by the flow equations in the current location.

Property 3. If the output transition T_j is never enabled and the invariant is always satisfied, then the evolution of the system stays indefinitely in the location q_i (i.e., $\delta_i \to \infty$).

Remark 1. The *reached region* is the image of the continuous evolution of the system in one hybrid location, until the output transition is enabled or the invariant conditions associated to the location are not verified.

The reachability analysis, in the case of one hybrid location having more than one output transitions ends when one of its output transitions is enabled or if the invariant conditions associated with this location are no longer verified. The case of hybrid locations having multiple input and multiple output transitions can be reduced always to the reachability analysis for one hybrid location having single input and multiple output transitions. The analysis method for such situations is presented in [10] and it will be not detailed here.

4.4. *Reachable automaton*

In general, it is not possible to represent, much less to compute, the set of all reachable states required to build a finite transition system for a hybrid system.

Here, we present the approach, which lets us to obtain the hybrid automaton model in which all the transitions are reachable by the system evolution. This model is called the *reachable hybrid automaton*. In order to be able to build this model, the transition enabling analysis, must be performed for the entire hybrid automaton. We have shown that the transition enabling analysis is in fact a reachability analysis of the transitions guards by the system evolution.

The reachability analysis represents the central problem in the verification of properties of hybrid systems. This allows better knowing of the global system evolution. Consequently, the information concerning the hybrid system behavior obtained by analysis will be used further for control synthesis. Therefore, it is necessary to retain the complete evolution of the system. In this aim, during the reachable automaton construction, the continuous evolution of the system in each hybrid location will be stored. Therefore, a *trace* vector is associated with each location of the hybrid automaton. In this vector, the initial input condition as well as the complete evolution of the system will be retained.

Example 4. In order to illustrate this, let us to consider a hybrid location with one output transition (Fig.10.a.). The output transition drives the system state into a forbidden state (where the specifications are no longer verified). The initial conditions are represented by the region R_0: [1,3]x[1,7]. The continuous dynamics are the same that in example 3. Therefore, the continuous successors computations results in the evolution represented in Figure 7. By reachability analysis we can conclude that the output transition is enabled after 3 sampling periods and once this transition enabled, the forbidden state can be reached.

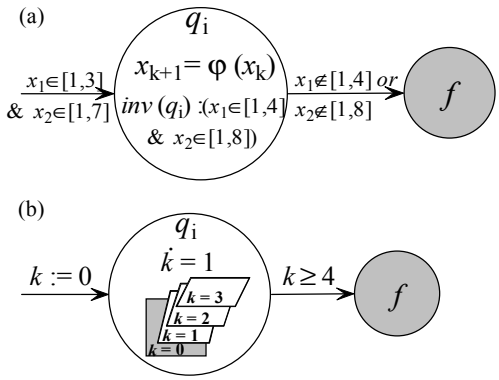

Fig. 10. Construction of reachable automaton.

In the reachable automaton, this result will be represented by a timed location in which the continuous evolution of the system is also stored (Fig. 10.b.).

Usually, real systems have a cyclical evolution in the means that the system starts from a specified initial discrete state and after a sequence of other discrete states, visited during the system functioning, the system returns in the initial state. Hence, we can conclude that each discrete state is visited several times during the system functioning.

Property 3 The reachable region associated with one discrete state is given as a union of regions reached, by the continuous evolution of the system, at each visit in this state.

Remark 2. Given the evolutions of the continuous variables of the systems, the initial condition for ne visit in a hybrid location of the automaton is generally different from that obtained for a previous visit.

The reachable state space for a hybrid automaton *A* is determined by the union of the continuous successors after the execution. Therefore, the successors computation, and consequently, the reachable state space computation is stopped when convergence of dynamics. However, the convergence can never occurs explained by the fact that the reacable state space computation is a non-decidable problem for hybrid automaton [7]. Hence, the global stability of the system can not be a priori checked. Therefore, the construction of reachable automaton is stoped when the gap between the initials input conditions in one hybrid location are less then a fixed precision ε. This value impose the precision of the resulting model.

The complete algorithm that let us to build the reachable automaton given as a timed model starting from an initial discrete-time hybrid automaton is given in [10] and it will not be detailed here.

5. CONCLUSIONS

In this paper an approach for modeling and analyzing hybrid systems in view of supervisory control synthesis was presented. The analysis approach represents a generalization of the analytical method "Clock Translation" in the mean that propose a technique for the construction of a reachable automaton given as a timed model, which models only the possible trajectories of the system. For this model the classical theory of supervisory control can be suitably applied and therefore the optimality of the control results can be guaranteed.

6. REFERENCES

[1] Alur, R., Courcoubetis, C., Henzinger, Th.A. and Ho, P.-H. - "Hybrid Automata: an algorithmic approach to the specification and verification of hybrid systems", *Hybrid Systems - Lecture Notes in Computer Science*, 736, **209-229**, 1993.

[2] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, Th.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S. "The algorithmic analysis of hybrid systems", *Theoretical Computer Science*, 138, **3-34**, 1995.

[3] Antsaklis, P. and Koutsoukos, X. – "On hybrid control of complex systems: a survey", *Symposium ADPM'98*, Reims, France, 1998.

[4] Asarin, E., Bournez, O., Dang, T. and Maler, O.- "Approximate reachability analysis of piecewise-linear dynamical systems", *Hybrid Systems: Computation and Control*, *Lecture Notes in Computer Science* 1790*,* **20-31**, 2000.

[5] Asarin, E., Bournez, O., Dang, T., Maler, O. and Pnueli, A.. – "Effective synthesis of switching controllers for linear systems", *Proceedings of IEEE,* 2000.

[6] Dang, T. – "Vérification et synthèse des systèmes hybrides", *Ph.D. Thesis*, INPG - Verimag, France, 2000.

[7] Henzinger, Th.A., Kopke, P. and Varaiya P. – "What's decidable about hybrid automata? The algorithmic analysis of hybrid systems", *Proceedings of the 27th Annual Symposium on Theory of Computing,* **373-382**, 1995

[8] Henzinger, Th.A. – "The theory of Hybrid Automaton", *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, **278-292**, 1996.

[9] Henzinger, Th.A., Ho, P.-H.and Wong-Toi, H. – "Algorithmic analysis of nonlinear hybrid systems", *IEEE Transactions on Automatic Control*, 43(4), **540-554**, 1998.

[10] Kurovszky, M. – "Etude des systèmes dynamiques hybrides par représentation d'état discrète et automate hybride", *PhD Thesis,* Laboratoire d'Automatique de Grenoble, France, 2002.

[11] Silva, B. and Krogh, B. - "Modeling and verification of sampled-data hybrid systems", *Proceedings of 4th International Conference on Automation of Mixed Processes*: *Hybrid Dynamic Systems*, **323-328**, 2000.