2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Controller Synthesis for a Class of Hybrid Systems

Sonia Batis and Hassane Alla, *Gipsa-lab, Grenoble*

***Abstract*—A timed control synthesis approach is proposed for hybrid systems modeled with rectangular hybrid automata. The control goal is to constrain the reachable state spaces by the addition of desired control specifications to the automaton transition guards, and to determine in a maximal permissive way the transition dates that respect those constraints. The approach is illustrated via a production system.**

I. INTRODUCTION

In this paper, we develop a global algorithm for the timed control of a class of systems broadly known as hybrid processes. This type of process can be found in numerous real-life automated systems.

This work belongs to the general framework of the Ramadge and Wonham theory [11] for the control of discrete event systems. These systems are modeled as generators of formal languages; the adjunction of a control structure allows limiting the language generated by the system by accordingly enabling and disabling events. Brandin and Wonham [3] have then extended the theory to the timed discrete event systems by adding discrete timing features to the initial systems.

Maler [10] introduced an extension to the Ramadge and Wonham theory, working in the timed automaton framework (suggested by Alur and Dill [1]) and defining the notion of timed games. Later, Cassez [4] used this notion to introduce an efficient on-the-fly algorithm for the analysis of timed automata.

Lots of automated systems evolve according to continuous sub processes which are started and stopped by discrete state orders. Therefore, processes have rarely a purely discrete or continuous behavior but a mixture between both of them. These dynamical systems with a double behavioral component are called hybrid systems and can be modeled by many tools among which there is the hybrid automaton [7]. This model can be considered as a generalization of the timed automaton model [1], where the continuous evolution is no longer represented by clocks but by differential equations on the continuous variables of the system.

Henzinger [8, 9] has studied the control problem of hybrid automata by introducing the hybrid games. The game proceeds in an infinite sequence of rounds and produces an ω-sequence of states. In each round, both players independently choose enabled moves; the pair of chosen moves either results in a discrete state change, or in a passage of time during which the continuous state evolves. In the special case of a rectangular game, the enabling condition of each move is a rectangular region of continuous state, and when time advances, then the derivative of each continuous variable is governed by a constant differential inclusion. The control problem for hybrid games asks: giving a hybrid game B and a formula φ over the discrete states of B, is there a strategy for player 1 so that all possible outcomes of the game satisfy φ?

Spathopoulos [12] considered the problem of supervisory control for rectangular hybrid automata and established a supervisory controller that can disable only discrete-event transitions in order to solve the non-blocking forbidden state problem.

Our control approach has been introduced in [14, 15] in a partial way. In fact, we have focused on local control on two Locations of the rectangular hybrid automaton to develop the main ideas.

In this article, we extend our preliminary approach to a general automaton and develop a global algorithm for the controller synthesis of hybrid systems modeled with rectangular hybrid automaton. The control is obtained via the automaton new transition guards modified by the addition of desired constraints to the system. It is based on the reachable state spaces computation for any location. A particular case is also studied in this paper. It concerns automata having Locations with more than one output transition.

The paper is organized as follows. In Section 2, we define the model assumptions, present the reachable state spaces and introduce the constrained automaton. Section 3 develops our approach of control. This approach will be applied to a production system in Section 4 and a conclusion will be given in Section 5.

II. CONTROL MODELING

In this section, we first present the model assumptions. Then, we introduce the notion of reachable state space. And finally, we define the constrained automaton.

A. The model assumptions and the reachable state spaces

The model chosen here is the Rectangular Hybrid Automaton (RHA). In this case, the continuous evolution is represented by differential equations on the continuous variables of the system. This model is interesting since the

S.Batis and H.Alla are with the Automatic Control Department, Gipsa-lab, 38402 Saint Martin d'Hères, Grenoble, France (e-mail: Sonia.Batis@gipsa-lab.fr, Hassane.Alla@gipsa-lab.fr)

flow is constant and this allows us to compute analytically the state change dates. This hypothesis isn't constraining since it represents a good approximation for most of the physical systems.

The model's definition is derived from the definition of Henzinger [7].

Definition 1: **The general model**

A Rectangular Hybrid Automaton (RHA) is defined by the tuple $H=(Q, X\cup\{h\}, \Sigma, T, inv, flow, q_0 E_0^{in})$, where:

1. $Q=\{q_1,\ldots,q_k\}$ is a finite set of locations representing the discrete states of the system;
2. $X\cup\{h\}$ is a finite set of real variables and h is a global clock;
3. Σ is a set of controllable events; $\sigma \rightarrow [a, b] \in \{\Re^+ \times (\Re^+ \cup \infty\}$, we associate to each event a clock interval;
4. $T \subseteq Q \times \Sigma \times Pred\ (X) \times Pred\ (X) \times 2^X \times Q$ is a finite set of transitions where $Pred\ (X)$ being the set of predicates on X;
5. inv: $Q \rightarrow Rect\ (X\cup\{h\})$ is a function which associates with each location $q \in Q$ a rectangular constraint for any variable $x_i \in X$ and for the clock h where $Rect$ being a rectangular predicate over X and h;
6. $flow$: $Q \rightarrow Rect\ (X' \cup \{h'=1\})$ is the function that assign to each location a dynamic for the continuous evolution;
7. q_0 is the initial location;
8. E_0^{in} is the initial input space of q_0; E_0^{in} is a convex region. ◊

All calculations to be used are starting from formally determining the reachable state spaces by a RHA. This determination is based on the application of a forward analysis of the system's automaton. We obtain a set of inequalities that gives the relations between the different state variables.

- **Forward analysis:** The forward analysis operators are used in order to compute all possible trajectories of the system. This leads to compute the state spaces associated to the stays of the system in each location of the model. One location can be reached with different continuous state spaces at its entrance, especially when the model contains cycles. Consequently, the state space associated to a location is equal to the conjunction of the continuous state spaces of all possible visits to the location.

The forward analysis is realized using the software PHAver [5]. This software provides commands for computing reachable sets of states and simulation relations plus a number of commands for the manipulation and output of data structure. Its language is as user friendly as possible.

B. Constrained Automaton

While imposing certain constraints to a transition between two locations, the system commutes at a certain moment that reduces the state space (Fig.1). This behavior joins the idea of forbidden states, introduced by Wonham [13] in the classical supervisory control.

Definition 2: Constraint: A constraint is a conjunction of polyhedral inequalities over X of the form $c_1x_1+\ldots+c_kx_k \sim c$, where $x_1,\ldots x_k \in X$, $c, c_1,\ldots, c_k \in Z$ and $\sim \in \{<, \leq, =, \geq, >\}$. It is called a polyhedral predicate over X and is denoted by $C(X)$.◊

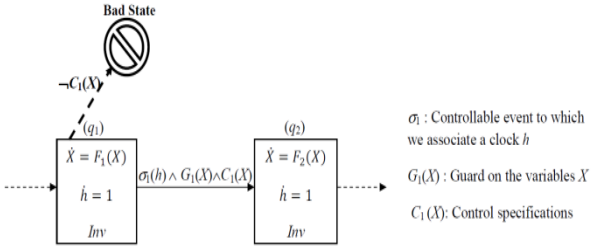

Fig. 1. Specification of the control

Definition 3: Control specification: $T \rightarrow (X \rightarrow C\ (X))$, a control specification is a function that assigns to each transition a constraint over X. ◊

For example, in Fig.1 above, a constraint $C_1(X)$ has been added in the guard between locations (q_1) and (q_2).

Definition 4: Constrained RHA:

A Constrained Rectangular Hybrid Automaton (CRHA) is defined by the tuple $H=(Q, X\cup\{h\}, \Sigma, T, inv, flow, q_0, E_0^{in})$, as in Definition 1 of a RHA, except for the fourth point which becomes:

4. $T \subseteq Q \times \Sigma \times Pred\ (X) \times C(X) \times Pred\ (X) \times 2^X \times Q$ is a finite set of transitions where $Pred\ (X)$ being the set of predicates on X and $C(X)$ being the set of constraints over X. ◊

Remark 1:

a. All the reachable state spaces of the automaton are convex. In fact, in each state space, we have only inequalities and intersections between them. It has been proven [6] that the image of a convex set by an affine application is convex and that the intersection preserves convexity.

b. We denote by W_i the convex set of inequalities for a Location (q_i) on the state variables x_j and the global clock h.

$W_j^i = [v_j^i, w_j^i]$: the existence interval corresponding to each variable x_j in Location (q_i). The limits v_j^i and w_j^i can be obtained by linear programming.
In this case, as W_i is convex, W_j^i is also convex.

c. Most of the real-life systems have reset loops. This feature leads us to guarantee the termination of the algorithm. It is expressed as follows: For any loop, it exists at least one transition $(q_i, \mathbf{v_i}) \rightarrow (q_{i+1}, \mathbf{v_{i+1}})$ such that we have the assignment over the global state X and the clock h, $(X,$

h):=(X_0, h_0); (X_0, h_0) is a convex region. These transitions will be called initialized transitions. ◊

The constraints are added as predicates in conjunction with the guards associated to the transitions. Consequently, there is a modification of the guards except for those where the state is initialized or reset. Hence the idea of the global automaton decomposition into branches, each branch being delimited by an initialized arc.

C. Decomposition into branches

Definition 5: Branch: An automaton branch is a sequence of Locations such that the initial and final Locations (q_0 and q_N) have an initialized exit arc. ◊

The constrained automaton is finite because the number of locations is finite. Since each loop is reset, the constrained automaton is decomposed into M branches and this number M is finite.

The model of the branch on which our control computation will be based is given by Fig.2.

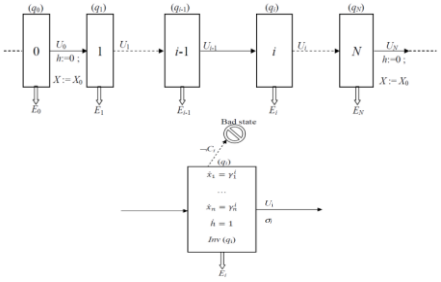

Fig. 2. a) A branch model b) Zoom on Location (q_i)

For each Location (q_i):

- The number of variables is equal to n. Each variable is denoted by x_j ($j\in[1,n]$).
- U_i is the guard in the transition between two locations (q_i) and (q_{i+1}), it is the conjunction between the predicate on the global clock h, the old guards on the variables X and the constraints on X: U_i= $Pred_i(h)\wedge G_i(X)\wedge C_i(X)$.
- The arc has the same number as the Location upstream.
- E_i refers to the reachable state space of Location (q_i). This means that it is the state space obtained after forward analysis of the constrained automaton.

Remark 2: In case of a loop, (q_N) and (q_0) correspond to the same Location.◊

The first step of our control approach is to make a forward analysis of the constrained automaton. This analysis leads to the state space reduction and can also lead to a state removal. It means that some states may not be reachable. This can be due to the fact that the constraints are stronger than the dynamics. If there is no state removal, it means that there exists a non-empty possible control for the constrained automaton. The obtained automaton is called *reachable constrained automaton*.

III. CONTROL COMPUTATION

Finding the maximal permissive controller consists in finding all maximal guards of commutation that respect both the invariants and the constraints.

The base of the control computation amounts to solve the general problem for a branch (Fig.2a).

A. Branch control computation

The constraints are defined regardless of the process. There is then no guarantee on their achievability. In fact, they can be inconsistent with the dynamic of the location. For a location, it can happen that an input value of the constrained variable is greater than its output value, then the duration of stay in the corresponding Location is negative in case of a positive flow. Since the duration of stay is a time value, it must always be positive. This will be the key idea of our controller synthesis approach.

We have cut the problem into parts by resetting the state in each branch (or loop), so that we don't have any link with the downstream. Then, the computation of the guard is done starting where the state is reset. As we mentioned earlier in Remark 1.c, the transition where the state is reset is called the initialized transition.

The new computed guard for a transition between two locations (q_i) and (q_{i+1}), which will lead to the maximal permissive controller, is denoted by V_i. It is calculated in a recursive way from the old guard U_i and the reachable state space E_i.

Theorem 1: Maximal permissive control computation for a branch

a. The new guards for the initialized transitions are: $V_0=E_0\wedge U_0$ and $V_N=E_N\wedge U_N$.

b. Let V_i be the new computed guard from location ($q_{i(i\in[0,N]}$), then the maximal permissive control V_{i-1} from (q_{i-1})is computed as follows:

- Case $\gamma_j^i>0$: $V_{i-1}=E_{i-1}\wedge U_{i-1}\wedge(\wedge_{i=1}^{n}[0,\beta_j^i])$ (1)
- Case $\gamma_j^i<0$: $V_{i-1}=E_{i-1}\wedge U_{i-1}\wedge(\wedge_{j=1}^{n}[\alpha_j^i,\infty])$ (2)

Where $x_j\in[\alpha_j^i,\ \beta_j^i]$ is the convex existence interval corresponding to each variable x_j in Location (q_i) (Remark 1.b) and γ_j^i is the flow of this variable in Location (q_i). ◊

Proof

a. After the addition of the constraints, E_0 and E_N are reduced by forward analysis. Then, the new guards are reduced. The reset associated to the output arc of the corresponding Locations q_0 and q_N initializes totally the flow vector. Then the computation of the new guards is direct. U_0 and U_N become V_0 and V_N by the expressions $V_0=E_0\wedge U_0$ and $V_N=E_N\wedge U_N$.

b. We suppose here that V_i is known and we want to compute V_{i-1}

For a location (q_i), we have: The entrance space for Location (q_i) is: $E^{in}= U_{i-1}\wedge E_{i-1}$, such that U_{i-1} corresponds to the old guard associated to the transition (q_{i-1}) →(q_i) (Fig.3)

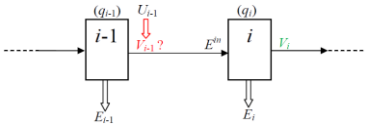

Fig.3. New guard computation

We consider that x_j^{in} is any element of E^{in}

Let's compute the duration of stay in the location (q_i) for the variable x_j:

$x_j = \gamma_j^i(h_i - h_i^{in})+x_j^{in}$ where $(h_i - h_i^{in})$ is the duration of stay in location (q_i) for the variable x_j before commutation, h_i being the instant of commutation.
→ $(h_i - h_i^{in})=(x_j - x_j^{in})/\gamma_j^i$ →This expression has to be positive because it is a time value →As $\gamma_j^i>0$ then $x_j^{in} \le x_j$, where $x_j \in V_i$ and x_j the variable value at the commutation state.
Let's retake a look at Remark 1.b and consider that $W_i=V_i$.

In this case, we call the convex interval corresponding to each variable $[\alpha_j^i, \beta_j^i]$, where:

- α_j^i is the minimal value taken by x_j respecting V_i
- β_j^i is the maximal value taken by x_j respecting V_i

$x_j \in V_i \rightarrow x_j \in [\alpha_j^i, \beta_j^i]$. As $x_j^{in} \le x_j$ then $x_j^{in} \in [0, \xi_j^i]$ where $\xi_j^i \in [0, \beta_j^i]$. This corresponds to a possible running of the automaton. **This running is maximal permissive when** $\xi_j^i = \beta_j^i$. Hence we find the relation (1) above.
For $\gamma_j^i < 0$, the proof is symmetric. ◊

Remark 3: The general control expression $V_{i-1}=E_{i-1}\wedge U_{i-1}\wedge (\wedge^n_{j=1} [0, \beta_j^i])$ is composed of two parts:

- A part due to the downstream control ($E_{i-1}\wedge U_{i-1}$) meaning that we have to leave the state as soon as the constraint in U_{i-1} is verified (in order not to go to a forbidden state).
- A part due to the upstream control ($\wedge^n_{j=1} [0, \beta_j^i]$) that corresponds to the satisfaction of the new guard V_i. ◊

B. Global control

Until now, we've considered the special case of a simple branch between two initializations.

Let's consider now the case where we have for a Location in the branch more than one output transition, as shown in Fig.4.

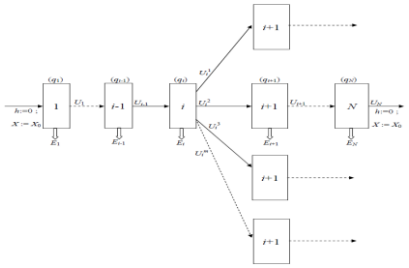

Fig.4. A location with more than one transition in the branch

We denote by U_i^k ($k \in [1, m]$) the old guards of the output transitions from (q_i).

Proposition 1:
In case of a Location (q_i) with more than one transition, the new guard V_{i-1} upstream is:

- Case $\gamma_j^i>0$: $V_{i-1}=E_{i-1}\wedge U_{i-1}\wedge(\wedge^m_{k=1}(\wedge^n_{i=1} [0, \beta_j^i]^k))$ (3)
- Case $\gamma_j^i<0$: $V_{i-1}=E_{i-1}\wedge U_{i-1}\wedge(\wedge^m_{k=1}(\wedge^n_{j=1} [\alpha_j^i, \infty]^k))$ (4)

◊

Proposition 1 is a direct generalization of Theorem 1. Since Location (q_i) has m transitions and the new guard upstream this transition depends on the downstream, V_{i-1} has to respect all guards downstream (q_i).

The clock is an important element in the system's control. It gives the date at which the commutation has to happen. It corresponds to the occurrence dates of the controllable events. Due to the previous calculations (Theorem 1 and Proposition 1), its values have been modified and often reduced. It is then necessary to compute the new occurrence intervals of the controllable events, which leads to compute the intervals defined in Remark 1.b.

In order to compute these intervals, we use the linear programming as follows:

Proposition 2: Clock commutation intervals
The clock commutation interval in the transition downstream Location (q_i) is $[\alpha_h^i, \beta_h^i]$, such that:

- α_h^i = Minimum h
- β_h^i = Maximum h

} respecting: V_i

◊

The interval $[\alpha_h^i, \beta_h^i]$ is never empty because Location (q_i) is reachable in the *reachable constrained automaton*.

Remark 4:

- The computation above has to be repeated for the whole automaton. The iterations' number stops as soon as we meet a state reset. Therefore, the general algorithm terminates since each loop or branch is reset.
- The structure of the offline controller automaton is exactly the same as the corresponding *reachable constrained automaton*. To obtain the timed controller, we empty each location of the *reachable constrained automaton* leaving only the clock h.

General control procedure

Step 1: Make a forward analysis of the *constrained automaton* → We obtain the *reachable constrained automaton*
Step 2: For l:=1 to M (each branch or loop)

- Apply Theorem 1.a in order to determine the new guard V_N for the initialized transition.
- For i:=N-1 to 1
  Apply Theorem 1.b in order to determine the new guard for the upstream transition.

Step 3: Construct the timed maximal permissive controller by emptying each location of the *reachable constrained automaton*, leaving only the clock h (computed by applying Proposition 2).

Remark 5: In case we have Locations with more than one transition, apply Proposition 1. ◊

IV. CASE STUDY: PRODUCTION SYSTEM

In order to apply our approach, we have chosen an example corresponding to a production system composed of an exclusive shared production center that supplies two buffers for consumption with a maximal capacity of 100 for each buffer. The system is shown in Fig.5.

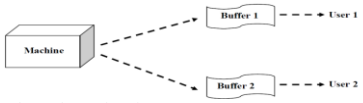

Fig.5. Exclusive shared production system

The state variables are the quantities in each buffer x_1 and x_2. We assume that, at start, the buffers are half full (x_1= x_2=50). We add to those variables a global clock, denoted by h, which determines the delivery dates in a defined interval (controllable events).

The machine is shared but in an exclusive way. Its production rate is constant and equal to 5. On the other hand the consumption fluctuates during the day.

The balance flow varies with time. The transition $(q_i) \rightarrow (q_{i+1})$ occurs following the controllable event σ_i. The corresponding automaton (for a normal daily behavior), without taking into account the expressions in bold, is shown in Fig.6.

The state changes are non-deterministic. It leads to a degree of freedom for control. For example, the controllable event σ_0 corresponds to an increase of consumption in buffer 1, which can occur for a value of $h \in [6,8]$.

The states (q_0), (q_1) and (q_2) are reserved to the production in buffer 1. The states (q_3), (q_4), (q_5) and (q_5') are reserved to the production in buffer 2.

The example meets all hypothesizes.

The following tables summarize the different values of production and consumption in each location and for each buffer.

TABLE I
PRODUCTION AND CONSUMPTION VALUES FOR BUFFER 1

Locations / Flows	q_0	q_1	q_2	q_3	q_4	q_5	q_5'	q_6
Production	5	5	5	0	0	0	0	0
Consumption	1	2	3	2	3	[5 6]	0	0

TABLE II
PRODUCTION AND CONSUMPTION VALUES FOR BUFFER 2

Locations / Flows	q_0	q_1	q_2	q_3	q_4	q_5	q_5'	q_6
Production	0	0	0	5	5	5	5	0
Consumption	2	[1 2]	1	2	3	5	4	0

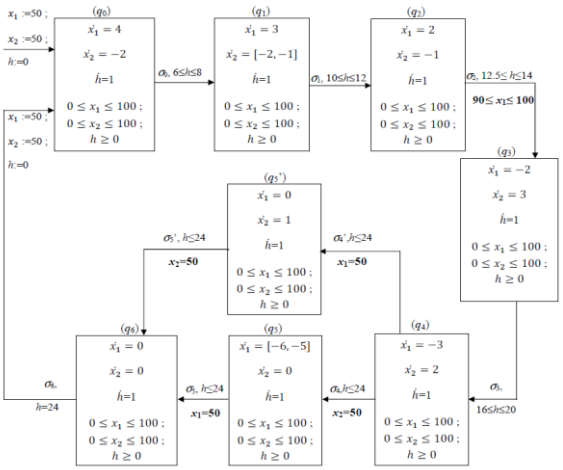

Fig.6. The production system automaton model

We need to impose the following constraints to the system (added in bold in the automaton model shown in Fig.5. above):

- Transition $(q_2) \rightarrow (q_3)$: We need to stop the production in buffer 1 when the level in $x_1 \in [90;100]$
- Transitions $(q_4) \rightarrow (q_5)$, $(q_5) \rightarrow (q_6)$, $(q_4) \rightarrow (q_5')$ and $(q_5') \rightarrow (q_6)$: We want that by the end of the day, the quantities in both buffers return to their initial values, i.e. x_2=50 and x_1=50.

The other transitions don't have constraints. In order to determine the offline timed controller of this system, we apply our approach developed in Section 4. In the following, we detail the different steps of our control approach that lead to the objective achievement.

Step 1: Make a forward analysis of the *constrained automaton*. We obtain 7 state spaces shown in Fig.7. The *reachable constrained automaton* is at the same time reversible and non-blocking. Then, there exists an optimal controller that respects the specifications where the spaces have been reduced compared to the non-constrained automaton. Therefore, there is a modification of the guards, which are computed in step 2.

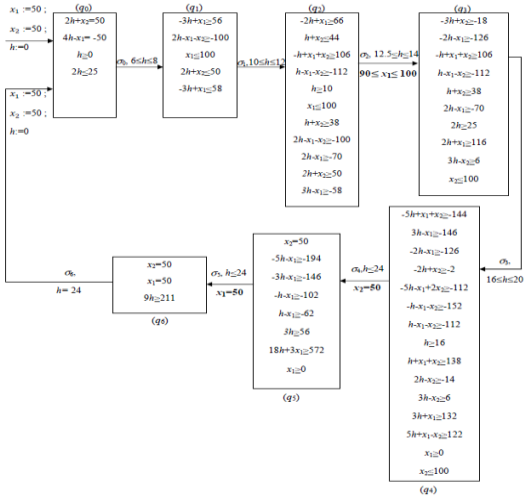

Fig.7. Forward analysis of the constrained automaton

The obtained state spaces express the linear inequalities between the different variables. They provide a bijective characterization of the state space.

We notice that Location (q_5') has been delayed by the forward analysis. This means that (q_5') isn't reachable by its upstream constraint x_1=50.

Step 2: New guards computation

In our automaton, there is only one state reset, which is in the transition σ_6. Then, it is considered as our starting point. After applying our algorithm, we obtain the new guards as follows:

- V_6={h=24; x_1=50; x_2=50}
- V_5={23.4≤h≤24; x_1=50; x_2=50}
- V_4={18.7≤h≤24; 64≤ x_1≤78.7; x_2=50}
- V_3={16≤h≤20; 80.4≤ x_1≤84; 42≤ x_2≤47.5}
- V_2={12.5≤h≤14; 91≤ x_1≤94; 27.5≤ x_2≤28.3}
- V_1={10≤h≤12; 86≤ x_1≤92; 30≤ x_2≤32}
- V_0={6≤h≤8; 74≤ x_1≤82; 34≤ x_2≤38}

Step 3: Controller construction

The obtained maximal permissive timed controller is shown in Fig.8.

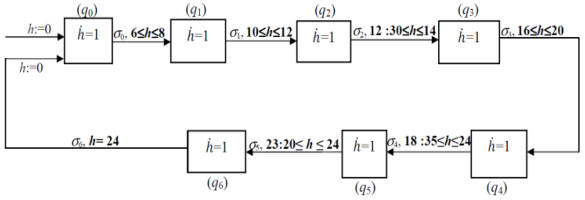

Fig.8. Maximal permissive timed controller of the production system

We notice that all initial values of the global clock are reachable, except for the following controllable transitions:

- (q_4) → (q_5): the initial clock interval (h≤24) becomes 18:35≤h≤24. The minimal limit 18:35 is due to the constraint x_2=50. This means that we have to change the consumption rate from 18:35 to satisfy that buffer 2 is half full.
- (q_5) → (q_6): the initial clock interval (h≤24) becomes 23:20≤h≤24.

Another important remark to mention is that the obtained controller gives all possible trajectories. This controller is non-deterministic. In order to make it deterministic, we have to choose a particular trajectory, for example, the one that changes consumption rate at start as soon as possible. In our example, this means that the first transition has to be made at 6 am, which corresponds to a quantity of 74 in buffer 1 and a quantity of 38 in buffer 2.

V. CONCLUSION

In this paper, we have introduced an approach for the control of a sub class of hybrid systems modeled by rectangular hybrid automata and characterized by a single global clock. This sub class is interesting since it can represent a great class of real life automated systems. The control is based on the computation of the clock commutation intervals respecting the specification constraints. They are deduced from the reachable spaces of the process and the constrained behavior. For the offline control, we have focused our presentation on a simple branch between two initializations containing only controllable events since it contains the main original ideas. Then we have extended the approach to the case of Locations having more than one output transition. Finally, we have studied a global case of a production system.

In future work, we will develop an online procedure to show how to fix a clock value in its corresponding interval and extend our approach to the case of systems containing both controllable and uncontrollable events.

REFERENCES

[1] R.Alur and D.Dill, "A theory of timed automata", Theoritical Computer Science, vol.126(2) , 1994, pp. 183-235.

[2] E.Asarin, O.Maler, A. Pnueli and J.Sifakis, " Controller synthesis for timed automata", In Proc. IFAC Symp. On System Structure & Control, 1998, pp. 469-474. Elsevier Science.

[3] B.A.Brandin and W.M. Wonham, "The supervisory control of timed discrete-event systemsé", IEEE Transactions on Automatic Control, vol.39, 1994, pp. 3357-3362.

[4] F.Cassez, A.David, E.Fleury, K.G.Larsen and D.Lime, "Efficient on-the-fly algorithms for the analysis of timed games, *CONCUR2005*, 2005, pp. 66-80.

[5] G.Frehse, "PHAVer: Algorithmic Verification of Hybrid Systems past Hytech", International journal on software tools for technology transfer (STTT), 2008, volume 10 number 3.

[6] J.Ch.Gilbert, "Ensembles convexes, Eléments d'optimisation différenciable : Théorie et Algorithmes" , pp.13-45, unpublished.

[7] T.A. Henzinger, "The theory of hybrid automata", Hybrid Systems II, LNCS, vol.999, 1996, pp. 278-292.

[8] T.A.Henzinger and P.W.Kopke, "Discrete-time control for rectangular hybrid automata", In Proceedings of the 24th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 1256, Spring-Verlag, 1997, pp. 582-593.

[9] T.A.Henzinger, B.Horowitz and R.Majumdar, "Rectangular hybrid games", In Proceedings of the Tenth International Conference on Concurrency Theory, Lecture Notes in Computer Science 1664, 1999, pp. 320-335, Springer-Verlag.

[10] O.Maler, A.Pnueli, J.Sifakis, "On the synthesis of discrete controllers for timed systems", In Proc. 12th Symp. On Theoretical Aspects of Computer Science (STACS'95), vol. 900, 1995, pp. 229-242. Springer.

[11] P.Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event systems", SIAM, J.Control and Optimisation, vol. 25, 1987, No. 1, 206-230.

[12] M.P.Spathopoulos, "Supervisory control for rectangular hybrid automata", In Proceedings of the 39th IEEE Conference on *Decision and Control*, Sydney, Australia, 2000, pp. 35-41.

[13] W.M. Wonham, "Supervisory control of discrete event systems", System Control Group, ECE Dept, University of Toronto, http://www.control.toronto.edu/DES, Revised July 2011

[14] S.Batis and H.Alla, "Timed control for rectangular hybrid systems", In 14th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'12, Bucharest, Romania, 2012, p.375-380.

[15] S.Batis and H.Alla, "Maximal permissive timed control for a class of hybrid systems", In 11th International Workshop on Discrete Event systems, WODES'12, Guadalajara, Mexico, 2012, p.157-162.