Modeling and Performance Evaluation of Hybrid Systems

Isabella Kotini, George Hassapis

Dept. of Electrical and Computer Engineering,
Aristotle University of Thessaloniki
54006, Thessaloniki,Greece
kotini@egnatia.ee.auth.gr, chasapis@eng.auth.gr

Abstract. This paper addresses the issue of evaluating the behavior of system models that involve mixed continuous and discrete evolutions of variables and have been developed by using the rectangular hybrid automata abstraction. The evaluation is based on mapping the requirements of system behavior to the hybrid automata states and checking the states that can be reached. Reaching a state requires to find all the transitions from the initial state to all the other states until it reaches the final state. Specifically, a method is proposed for finding whether there is a finite number of state changes in reaching the desired state. According to this method, the duration of the model stay at a discrete state is estimated and the values of continuous-time variables at which transition from a discrete state to another occurs, are expressed as functions of this duration. An industrial case study demonstrates the application of this method.

1 Introduction

Chemical process control, air traffic control, automotive control, robotics and automated manufacturing are some of the complex engineering applications that are implemented by embedded computing devices. The design of the embedded computer software is a rather complicated process and involves the realization of the control functions and the communication of the embedded computer with other machines and humans. During the early phases of the design and many other times, the developer wishes to make sure that certain critical specifications of the operation of the process or machine under the automatic control or guidance of the embedded computer will be satisfied. Such specifications are not restricted to a single norm-based measure of some important variables but include complex descriptions of state trajectories that the process has to follow and combination of measures of important variables. To find out whether such specifications are satisfied, abstract graphical, mathematical or formal models of the design are worked out with the purpose of describing modes of the operation of the physical process or machine under the control of the embedded computer.

Each mode of operation may be viewed as a hybrid system that involves mixed continuous-valued variables (the process or machine state variables) and discrete event variables (the program state). The continuous-valued variables can evolve

Proceedings of the 1st South-East European
Workshop on Formal Methods, SEEFM'03
November 2003, Thessaloniki, Greece

within certain ranges of values or under certain conditions. When these conditions cease to be valid and/or changes of discrete event variables are made by the program, discontinuous changes of the continuous variables may occur. When a discontinuous change takes place the variables can evolve under different conditions whereas changes on different discrete event variables may cause the next discontinuous change. Different approaches have been proposed for modeling this hybrid system behavior. Early approaches tended to focus on discrete event modeling formalisms, such as finite state machines, process algebras, Petri-nets, temporal logic, etc [4] and switched systems [16]. Later, the approach of extracting an equivalent discrete event model of the continuous subsystem which could be supervised by a discrete-event supervisor [15] was proposed, and the use of differential Petri-nets was introduced [6] A framework with significant potential for practical usage was the extension of the symbolic model checking approach (SMC) [11] for finite state machines to hybrid systems. The result was the hybrid automaton [2]. These automata are generalizations of traditional finite state machines in which event transitions are conditioned on truth value of logical propositions defined over a set of continuous-valued dynamical processes.

The problem of verifying the satisfaction of specifications by an automaton model of a hybrid system by the SMC method can be expressed in a more formal way as follows. Given a class of hybrid system automata models H and a class of specification properties P, the aim is to find a computational procedure that will decide if there is a state in the state space of the model that satisfies the specification and whether this state can be reached from an initial state in a finite number of computational steps. If such a computation procedure exists, then this class of verification problem is called decidable. By far the best-known computational method is the so called reachability construction [10]. This is an iterative procedure according to which reachable states are computed by successive approximation, starting from a set of initial states and repeatedly adding new reachable states. However, because of the unaccountability of the state space of a hybrid automata model, this computational method for the general case of the hybrid automaton is undecidable. Instead, the maximal class of hybrid automata with a decidable reachable states computation has been proved [8] to be the class of rectangular automata when specifications are expressed in linear temporal logic (LTL). This finding is of significance because hybrid systems with very general dynamics can be locally approximated by using rectangular automata [7]. Although the decidability proof makes certain that there is a finite number of computational steps in finding whether a set of states is reachable, it does not give a practical LTL computation algorithm, as long as the number of the computational steps is not known. If this number is very large the computation becomes intractable.

The main focus of this work is to present a procedure other from that of a typical reachability construction for which the number of computations can be predicted when a rectangular automata model is available.

The article is organized as follows. In section 2, formal definitions of the general hybrid automata and the rectangular hybrid automata formalisms are introduced. In sections 3 the proposed algorithm for finding the number of computations that the specification states can be reached is described. In section 4 the case study of verify-

ing the satisfaction of a basic specification of the control of the cement milling process demonstrates the application of the algorithm. Conclusions drawn from this work are given in section 5.

2 Hybrid Automata

The hybrid automaton is defined by the tuple (N, Δ, L, I, R, T). N is a marked directed graph characterized by the ordered pair (V,A), where V is a set of vertices and A is a set of directed arcs between the vertices. A way of modeling the discontinuous jumps of its continuous-time state is by mapping each discrete state that the system can occupy to a vertex and the discrete state-transition function to an arc of the N graph. In a notation format this can be expressed by a vector $\overline{\mu}$ in which each element of it is a value of the function μ:V→{0,1}. This function associates a one with the vertex that represents the discrete state that the system is at time τ and a zero with each one of the other vertices. Usually, this vector is called the *marking vector* of the hybrid system.

The set Δ is a set of elements that represent the continuous-time state trajectories that the hybrid system generates as long as it stays in a discrete state. It may be characterized by:

$$\Delta = \{F(1,X), F(2,X) \ ,..., F(n,X) \ \} \tag{1}$$

where *X* denotes the ν real-valued variables of the state space, defined by

$$X = (x_1, x_2, \cdots, x_v) \tag{2}$$

and representing the hybrid system that generates state trajectories at the ith discrete state through the differential inclusion:

$$\dot{X}(t) \in F(i,X) \tag{3}$$

for any $F(i,X) \in \Delta$, defined by the set of elements:

$$F(i,X) = \{f(1,x_1), f(i,x_2), \cdots, f(i,x_v)\} \tag{4}$$

where $f(i,x_j)$ is a constraint on the xj coordinate.

Equation (3) labels the ith vertex. The continuous state of the system is characterized by the value that the following four-tuple $z = (\dot{X}, X, \tau_0, X_0)$ takes over the time, where $\dot{X}$ is one of the mappings in Δ, i.e. F(i,X), and $\tau_0 \in R$ and $X_0 \in R^n$ are referred to as the initial time and initial value of the set of the continuous-valued variables respectively.

$$X_0 = (x_{01}, x_{02}, \cdots, x_{0v}) \tag{5}$$

Combining z with $\bar{\mu}$ the ordered pair σ = (z, $\bar{\mu}$) is produced, each value of which comprises the *hybrid state* of the automaton. The set of all the values that the ordered pair (z, $\bar{\mu}$) takes is denoted as *H* and will be called the hybrid state space.

The elements of the set L are also subsets of logical propositions which can be generated recursively by the conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$) of atomic relationships. They label the arcs and determine how the continuous and discrete parts of the hybrid system interact. The atomic relationships are elementary formulae or equations that have the form $g(X)>0$, where $g:R^v \rightarrow R$ is a function defined over the continuous states R^v. Each subset is assigned to an arc, *l*, and is usually called the guard set of the arc, denoted by the symbol *Guard(l)*. The elements of the set I are subsets of relationships that label the vertices and impose constraints on the continuous part *X* of the state. Each subset is associated with a vertex, it consists of relationships of the form $a_{ij} \le x_j \le b_{ij}$ or $x_j \in [a_{ij}, b_{ij}]$ is called the invariant set of the vertex and is denoted by the symbol *Inv(i)*. The elements of the set R are subsets of reset conditions to the continuous-time state and they label also the arcs. The subset that is associated with an arc *l*, is called the reset set of the arc and is denoted by the symbol *Reset(l)*. Each reset condition has the form $h(X_0) = 0$. The elements of the *T* set, are the subsets of initial conditions associated with each vertex being of the form $X=X_0$ and $\tau_0 = \tau$, where τ denotes the current time. Each subset is denoted by the symbol *Init(i)*.

The dynamics of the hybrid automaton are characterized by the generated hybrid trajectories. The trajectories that have the same value of the marking vector, $\bar{\mu}$, in their sequence of hybrid states, satisfy the differential inclusions of the vertex that is associated with the non-zero element of this marking. This marking is changed when the relationships of the guard set of an arc are satisfied by the values that the continuous part of the hybrid state takes at a certain time. Then, the hybrid automaton moves to a hybrid state with different inclusions of the form shown in (3). The initial time and initial values of the continuous-time variables are reset to the values that satisfy the guard relationships. They may be given different reset values if reset relationships are associated with the transition to the new state.

2.1 Rectangular Automata Models

Consider the space R^v with the variables $x_1, x_2, \ldots, x_v$. A rectangular set is defined by the conjunction of linear inequalities and equalities of the form $x_j \approx c$, where ≈ is one of $<, \le, =, \ge, >$ and c is a constant. For a rectangular set *B*, let B_j be its projection onto the jth coordinate. Thus, the rectangular set is of the form $B=B_1 \times B_2 \times \ldots \times B_v$, where each B_j is a bounded or unbounded interval. On the basis of this definition, a rectangular automaton is a hybrid automaton that satisfies the following constraints.

- For every vertex $i \in V$, the sets *Init(i) and Inv(i)* are rectangular sets.
- For every vertex $i \in V$, there is a rectangular set B_i such that

$$F(i,X) = B_i \quad (6)$$

- For every arc $a \in A$, the set *Guard(α)* is a rectangular set and there is a rectangular set B_α and a subset J_α, $J_\alpha \subseteq \{1,2,...,v\}$ such that for all $x \in R^v$

$$\text{Reset}(\alpha) = \{(x'_1, x'_2, ..., x'_v) \in \Re^v \mid \text{for all } 1 \le j \le v, \quad (7)$$

$$\text{if } j \in J_\alpha \text{ then } x_j \in B_\alpha \text{ else } x'_j = x_j\}$$

Therefore, in a rectangular automaton the derivative of each variable stays between two fixed bounds, which may be different in different locations. With each discrete jump across an arc the value of a variable is either left unchanged or reset non-deterministically to a new value within some fixed, constant interval.

Given the rectangular automaton model of a hybrid system, one may use it to check whether performance specifications are satisfied under certain modes of hybrid system operation. To describe these specifications expressive methods, such as the propositional or other types of logic [5], [1], [9] must be adopted for capturing the designer's requirements. In this work the propositional logic was chosen to express the guards of the automaton. As it is proved [14] in selecting the LTL logic to express requirements and automata constraints, the verification problem with the model checking approach can be decided for initialized rectangular automata, provided every proposition occurring in temporal formulae is either an automaton vertex or a rectangular set. Once specifications have been expressed in this logic formalism, the problem of verifying the satisfaction of these specifications is raised. If as a region of states is defined the pair $<\overline{\mu}, P>$, where $\overline{\mu}$ is the marking vector and P is a logical proposition on the automaton variables, then, the region is said that contains the state $s=(z, \overline{\mu})$ when $\overline{\mu} = \overline{\mu}$ and z satisfies P. As it was explained in the introduction, one of the methods that is used to verify the satisfaction of a specification is the construction of the tree of the regions that are reachable from an initial region or state. Algorithms and tools for constructing this tree have become available [3] for linear and rectangular hybrid automata in which the guards depend on one of the variables of a vertex. Although the previous discussion leads to the conclusion that the construction of the tree of state regions will terminate, the computations involved may be time consuming and require a very large number of computational steps. In order to circumvent this problem, in this work a method is proposed for estimating by a much shorter number of computations the number of the discrete state changes required to reach the region of the continuous time states that a specification defines. Having found that such a finite number exists we may conclude that the specification is satisfied without constructing the whole state space tree that leads from an initial region to the desired region of states. In the following, the method is presented.

3 Specification Verification

According to what has been said in the previous section, a way of verifying that a specification expressed in LTL logic is satisfied or not by a rectangular automata model of a hybrid system is to find if the region of states that are defined by the specification is reached in a finite number of automaton iterations. As an automaton iteration is defined the sequence of discrete state changes that an automaton model takes until it returns to the initial state. As there is always a duration of time that the automaton stays in a vertex or discrete state which is determined by the invariant of the vertex and the guard of the arc emanating from this vertex, the number of iterations can be derived from this duration the way explained in the following.

Let us consider a rectangular and deterministic hybrid automaton, A, with a continuous state-tuple $z = (\dot{X}, X, \tau_o, X_0)$, where *X, X0* and τ_0 have been defined in the previous section and the set *F(i,X),* at the i^{th} vertex of the automaton is defined by:

$$F(i,X) = \left\{ (k_{1i,min}, k_{1i,max}),\ (k_{2i,min}, k_{2i,max}),\ \dots\ (k_{\nu i,min}, k_{\nu i.,max}) \right\} \quad (8)$$

where $k_{si,min}$ and $k_{si,max}$ are the minimum and the maximum value respectively of the first derivative of the variable x_s, $s \in [1, \nu]$, in the vertex i. Let us now denote by:

- $x'_{d,i}$ the value of the variable x_d in the guard of the transition from the i^{th} to the $(i+1)^{th}$ vertex that enables the transition
- $x'_{d,i,,k,min}$ and $x'_{p,i,,k,max,}$ the minimum and maximum values respectively of the variable x_d, in the i^{th} vertex, after k iterations and when the automaton exits the i^{th} vertex,
- $x_{d,i,k,min}$ and $x_{d,i,k,max}$ the minimum and the maximum value of the x_d variable after k iterations and when the automaton enters the i^{th} vertex, and
- $\Delta_{i,k}=[\delta_{ik,min}, \delta_{ik,max}]$ is the range of values of the duration of the i^{th} vertex at the k^{th} iteration that is the time taken from the instant the automaton entered the i^{th} vertex until it exits from this vertex. $\Delta_{i,k}$ is empty when the following conditions exist:

$$\Delta_{i,k} = \varnothing \Rightarrow (x'_{d,i} \langle x_{d,i,k,min} \wedge k_{d,i,min} \geq 0) \vee (x'_{d,i} \rangle x_{d,i,k,max} \wedge k_{d,i,max} \leq 0) \quad (9)$$

An automaton is considered to be deterministic if there is only one transition from one vertex to another and a specific minimum and maximum duration for each vertex

Let us consider that any value of the differential inclusion $\dot{x}_{p,i} \in [k_{p,i,min}, k_{p,i,max}]$ at the k^{th} iteration can be computed from the differential approximation:

$$\dot{x}_{p,i,k} = \frac{x'_{p,i,k} - x_{p,i,k}}{\delta_{i,k}} \quad (10)$$

When the guard condition is satisfied, it holds $x'_{p,i,k} = x'_{d,i}$. When $\Delta_{i,k}$ is not empty, then depending on whether $k_{d,i\,min}$ and $k_{d,i,max}$ are positive or negative numbers

the limits of $\Delta_{i,k}$ can be estimated [13]: from the following formulae which are derived from (10)

$$\delta_{i,k,min} = \begin{cases} \dfrac{x'_{d,i} - x_{d,i,k,min}}{k_{d,i,min}} & \text{if} \quad x'_{d,i} < x_{d,i,k,min} \wedge k_{d,i,min} < 0 \\ \dfrac{x'_{d,i} - x_{d,i,k,max}}{k_{d,i,min}} & \text{if} \quad x'_{d,i} > x_{d,i,k,max} \wedge k_{d,i,max} > 0 \\ 0 & \text{if} \quad x_{d,i,k,min} < x'_{d,i} < x_{d,i,k,max} \end{cases} \tag{11}$$

$$\delta_{i,k,max} = \begin{cases} \infty & \text{if} \quad \begin{array}{c} (x'_{d,i} < x_{d,i,k,min} \wedge k_{d,i,max} \geq 0 \vee \\ \vee (x'_{d,i} > x_{d,i,k,max} \wedge k_{d,i,max} > 0 \wedge k_{d,i,min} \leq 0) \vee \\ \vee (x_{d,i,min} \leq x'_{d,i} \leq x_{d,i,k,max} \wedge k_{d,i,min} \leq 0 \leq k_{d,i,max} \end{array} \\ \dfrac{x'_{d,i} - x_{d,i,k,max}}{k_{d,i,max}} & \text{if} \quad x'_{d,i} < x_{d,i,k,max} \wedge k_{d,i,max} < 0 \\ \dfrac{x'_{d,i} - x_{d,i,k,min}}{k_{d,i,min}} & \text{if} \quad x'_{d,i} \geq x_{d,i,k,max} \wedge k_{d,i,min} > 0 \end{cases} \tag{12}$$

As the vertex duration is the same for all the other variables of the vertex that are not related with the guard, once the duration is available, the limits of the range of the values that these variables may take when the automaton exits the vertex can be computed from the following relationships that are easily derived from (10) when the range of the values of the differential inclusion $\dot{x}_{p,i} \in [k_{p,i,min}, k_{p,i,max}]$ are negative or positive real numbers.

$$x'_{p,i,k,min} = \begin{cases} x_{p,i,k,min} + k_{p,i,min} * \delta_{i,k,max} & \text{if} \quad p \neq d \quad \wedge \quad k_{p,i,min} < 0 \\ x_{p,i,k,min} + k_{p,i,min} * \delta_{i,k,min} & \text{if} \quad p \neq d \quad \wedge \quad k_{p,i,min} \geq 0 \\ x'_{d,i,k} & \text{if} \quad p = d \end{cases} \tag{13}$$

$$x'_{p,i,k,max} = \begin{cases} x_{p,i,k,max} + k_{p,i,max} * \delta_{i,k,max} & \text{if} \quad p \neq d \quad \wedge \quad k_{p,i,max} < 0 \\ x_{p,i,k,max} + k_{p,i,max} * \delta_{i,k,min} & \text{if} \quad p \neq d \quad \wedge \quad k_{p,i,max} \geq 0 \\ x_{d,i} & \text{if} \quad p = d \end{cases} \tag{14}$$

Using equations (13) and (14) and assuming that there is not a transition resetting the value of a variable x_p, another set of relationships can be derived by induction. These relationships express the range limits of the values that a variable of a vertex

may take upon the automaton exit from the vertex as a function of the initial value of the variable and the cumulative sum of the durations of each vertex over a number of successive automaton iterations. The relationships are:

For $i \in [1, n]$ and $p \in [1, v]$

$$x'_{p,i,k,max} = x_{p,init} + k_{p1,max} \sum_{j=1}^{k} \delta_{1j} + k_{p2,max} \sum_{j=1}^{k} \delta_{2j} + \ldots + k_{p,i,max} \sum_{j=1}^{k} \delta_{ij} + \quad (15)$$

$$+ k_{p,i+1,max} + \sum_{j=1}^{k-1} \delta_{(i+1)j} + \ldots + k_{p,m,max} \sum_{j=1}^{k-1} \delta_{mj}$$

$$x'_{p,i,k,min} = x_{p,init} + k_{p1,min} \sum_{j=1}^{k} \delta_{1j} + k_{p2,min} \sum_{j=1}^{k} \delta_{2j} + \ldots + k_{pi,min} \sum_{j=1}^{k-1} \delta_{ij} + \ldots + \quad (16)$$

$$+ k_{p,m,min} \sum_{j=1}^{k-1} \delta_{mj}$$

As the formation of the sums in the above equations is based on the summation of the second term of equations (13) and (14) , the quantity $\delta_{i,j}$ will take values that depend on whether the slopes $k_{p,i,\min}$ and $k_{p,i,\max}$ have negative or positive values, that is:

$$\delta_{i,j} = \begin{cases} \delta_{i,j,min} & \text{if} \quad k_{p,i,min} \geq 0 \vee k_{p,i,max} < 0 \\ \delta_{i,j,max} & \text{if} \quad k_{p,i,min} < 0 \vee k_{p,i,max} \geq 0 \end{cases} \quad (17)$$

The letter m represents the last vertex the automaton is moved into, before its return to the initial vertex.

If there is a transition which resets the value of the variable x_p, then the following relationships will hold:

$$x'_{p,i,k,min} = x_{p,,r,,reset} + k_{p,(r+1),min}\delta_{(r+1),k} + \ldots + k_{pi,min}\delta_{i,k} \quad (18)$$

$$x'_{p,i,k,max} = x_{pr,reset} + k_{p,(r+1),max}\delta_{(r+1),k} + \ldots + k_{p,i,,max}\delta_{i,(k-1)} \quad (19)$$

where r represents the number of the vertex before the reset. In this case the value of the variable depends on the duration values of the r^{th} up to the i^{th} vertices.

A direct result of the above analysis would be the following procedure for verifying the reachability of the region of states that is specified by the logic proposition P.

1. Starting from the vertex with the initial conditions and by applying equations (10) and (11) at every vertex and for successive automaton iterations, find the limits of the duration ranges for each vertex and for an arbitrary number of iterations.

2. Replace in the sums of (15) and (16), for $p \in [1, \nu]$ and $i \in [1, n]$ the duration limits by the values defined in (17) and compute the values of the partial sums that correspond to each variable of each vertex at every unit increase of the number of iterations.
3. Find either an analytic relationship of the values of the sums in terms of the number of the automaton iterations, if such a function exists, or alternatively find a polynomial approximation by curve fitting techniques.
4. Replace in (15) and (16) the equivalent functions of the sums and express the limits of the range of values that each variable in a vertex may take upon the exit of the automaton from this vertex as a function of the number of the automaton iteration.
5. In order to check that the region of states, defined by the P proposition, is reachable, equate the expressions of the previous step with the limiting values of the variables that are derived from the region that P defines. Solve the obtained equations with respect to the number of iterations. If there is a positive integer number of iterations for which all the above equations become true, then conclude that the region defined by P can be reached from the assumed initial region.

4 Case Study: Control of a Cement Mill

In the following the basic specification of a validated model of the control of a ce-

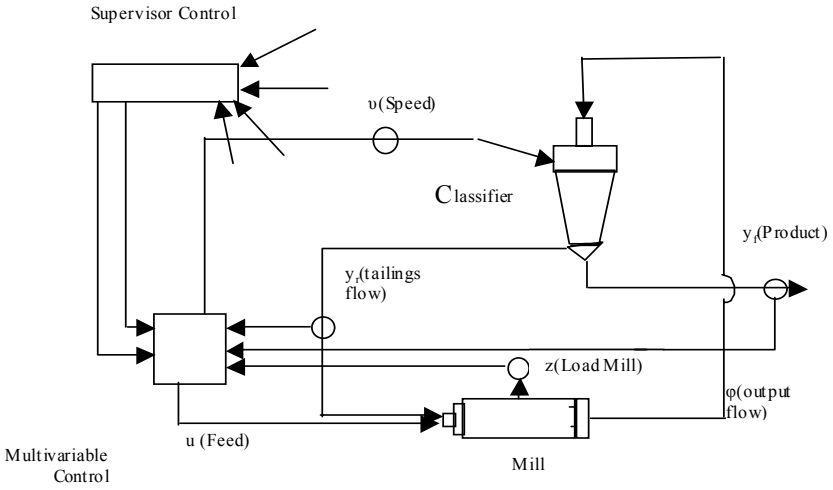

Fig. 1. Schematic diagram of the milling circuit

ment mill will be verified by the use of the proposed method and a comparison with the results of the classical analysis of the validated model will be made. In cement milling circuits control techniques are applied with the purpose of avoiding their unstable operation. This operation is explained in the following by considering the

schematic layout of a cement milling circuit shown in figure 1. The mill is fed with raw material (feed u). After grinding, the material is introduced in a high-efficiency classifier and separated in two classes. The tailings (flow yr) are fed back into the mill while the finished product (flow yf) exits the milling circuit. The classification of the material is driven by the rotational speed and by the airflow rate of the classifier. In steady-state operation, the product flow rate is equal to the feed flow rate while the tailings flow rate and the load in the mill may take any arbitrary constant values. The load in the mill(z) depends on the input (fresh feed plus tailings

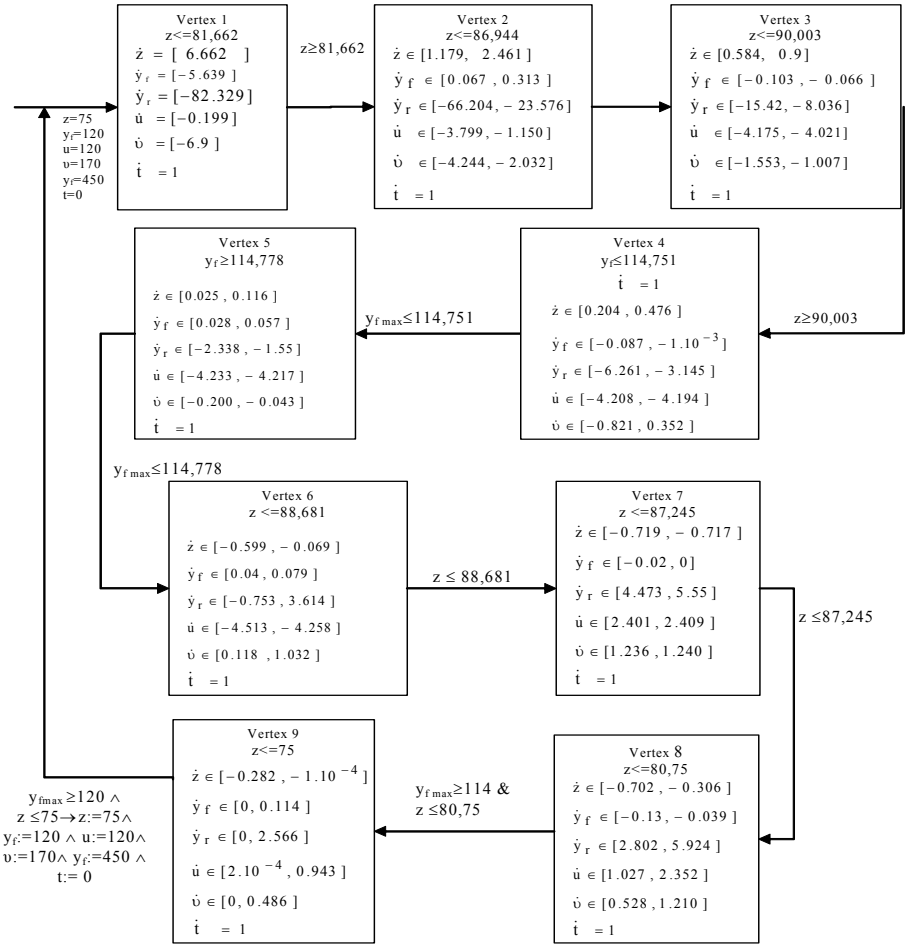

Fig. 2. The rectangular automaton model of the controller

flow rate) and on the output flow rate (φ). This flow rate depends in a nonlinear way on the load in the mill and on a very important time-varying quantity, the hardness of the material. Sometimes this non-linearity can cause the instability of the system, that is, the load changes from a very high level to a low level and vice versa leading re-

spectively to the obstruction of the mill or to a fast wear of the mill internal equipment. This phenomenon is usually called the plugging of the mill and requires the interruption of the grinding process. To avoid this phenomenon various control strategies are applied.

The rectangular automata model of Figure 2 approximates the typical strategy of the Linear Quadratic (LQ) control of a nonlinear model [12] of the milling process. Specifically it controls the load (z) and the product flow rate (yf) by adjusting the feed flow rate (u) and the rotational speed of the classifier (υ). This strategy succeeds to avoid the plugging phenomenon as long as the changes in the hardness of the material are within certain limits. Our aim is to verify that the proposed control strategy avoids the plugging effect for a range of hardness from 1.25 up to 1.34. The automaton consists of 9 vertices. In each vertex the first order time derivatives of the variables of the nonlinear model are varying within the range limits that are shown in figure 2. For example, the vertex 1 refers to the variable changes when the load of the mill is below 81 tons. The automaton moves to the second vertex when the load exceeds the value of the 81 tons and is less than 87 tons. The parameter conditions are marked on the automaton graph as vertex invariants and guard conditions on the arcs. Considering that the mill is at the state of its normal operation which is defined by the predicate:

$$S=\{\text{vertex1}\wedge z = 75\wedge u = 120\wedge yf = 120\wedge yr = 450\wedge v = 170\wedge d=1.25\wedge t = 0\} \quad (\mathbf{20})$$

and the hardness of the material changes from 1.25 to 1.34, we wish to check whether plugging of the mill may occur after the elapse of 100 iterations. The specific mill is considered to enter into a plugging effect situation when the load becomes higher than 100 tons and the product flow drops below 90 tons/h. Also, during the mill control operation the values of the manipulated variables of the feed and the rotational speed should remain within the limits of [0,200tons/h] and [100, 300 r/min] respectively. These control requirements can be expressed by the following predicate of prepositional logic:

$$T= \{z >100 \wedge y_f< 90 \wedge(0\leq u\leq 200)\wedge(100\leq \upsilon\leq 300) \wedge t < 100\} \quad (\mathbf{21})$$

Applying equations (11) and (12) for the first iteration, the minimum and maximum values of the durations of each vertex are found. The results for the first iteration are given in Table 1.

Table 1. Min and Max vertex duration

Vertices	Value of Duration	
	Minimum	Maximum
Vertex 1	1	1
Vertex 2	2,146	4,48
Vertex 3	3,399	5,238
Vertex 4	0	9,052
Vertex 5	0,475	0,964
Vertex 6	2.229	34,475

Vertex 7	1,997	2,003
Vertex 8	9,249	21,219
Vertex 9	0	20,399

In a similar way the duration values for the other 99 iterations are obtained. Then, the minimum and maximum values of the variables of the automaton are computed from the equations (18) and (19), by using the computed values for the durations of Table 1. The new results are listed in Table 2.

Table 2. Minimum and maximum values of variables in each vertex

		Vert.1	Vert.2	Vert.3	Vert.4	Vert.5	Vert.6	Vert.7	Vert.8	Vert.9
Product flow rate	Min. Value	114.361	114.504	113.965	113.174	114.778	114.867	114.827	111.644	111.644
	Max Value		115.762	115.539	114.751	114.778	121.352	121.352	120.991	121.236
Feed flow rate	Min. Value	119.801	102.783	80.915	42.824	38.742	0	4.795	23.543	23.543
	Max Value		117.333	103.667	103.667	101.669	92.192	97.016	146.923	166.159
Mill Load	Min. Value	81.662	86.943	90.003	90.003	90.014	88.681	87.245	80.752	75
	Max Value		86.943	90.003	94.311	94.423	88.681	87.245	80.752	75
Classifier Speed	Min. Value	163.099	114.088	135.955	128.522	128.329	128.592	131.060	135.944	135.944
	Max Value		158.738	155.317	155.316	155.296	241.177	243.660	269.335	279.249
Tailings flow rate	Min. Value	367.667	195.110	114.343	57.668	55.413	0	1.683	27.599	27.599
	Max Value		317.072	289.760	289.760	289.026	280.982	292.095	417.797	470.139

Fitting these data to a polynomial curve, the functions in Table 3 are derived.

Table 3. Polynomial functions of automaton minimum and maximum variable values

Polynomial Functions		
Product flow rate	Min. Value	$113.882 + 0.086*t - 1.79*10^{-3}*t^2 + 6.641*10^{-6}*t^3$
	Max Value	$114.909 - 0.042*t + 4.498*10^{-3}*t^2 + 3.51*10^{-5}*t^3$
Feed flow rate	Min. Value	$125.7 - 2.094*t + 0.035*t^2 - 9.771*10^{-5}*t^3$
	Max Value	$131.073 - 6.139*t + 0.093*t^2 - 4.229*10^{-4}*t^3$
Mill Load	Min. Value	$82.483 + 0.699*t - 0.014*t^2 + 6.287*10^{-5}*t^3$
	Max Value	81.315 + 1.048*t - 0.023*t2 + 1.201*10-4*t3

Classifier Speed	Min. Value	$148.452 – 1.854*t + 0.04*t2 – 2.245*10\text{-}4*t3$
	Max Value	$163.75 - 1.732*t + 0.087*t^2 - 5.832*10^{-4}*t^3$
Tailings flow rate	Min. Value	$343.713 – 22.088*t + 0.402*t^2 – 2.139*10^{-3}*t^3$
	Max Value	$367.03 – 7.305*t + 0.142*t^2 - 5701*10^{-4}*t^3$

In Figure 3, a graphical presentation of the polynomial functions is given. As this

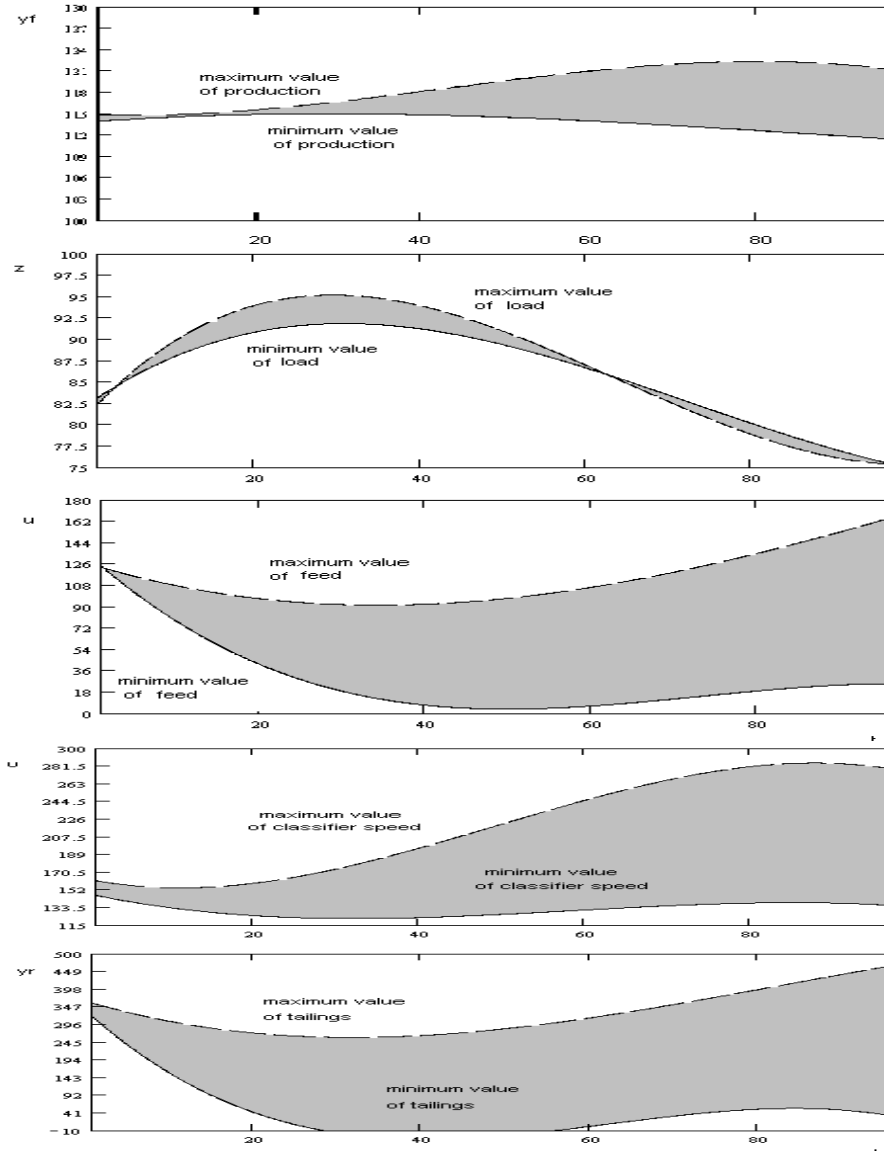

Fig. 3. Diagrams of minimum and maximum variable values

figure shows the maximum value for the load of the mill can never reach the value of 100 tons and the minimum value for the product flow to be less than 90 tons/h. So, plugging can never occur within the 100 model iterations. As every model iteration may be associated with a time unit, we may consider that within a period of 100 time units of mill operation the plugging effect is effectively controlled by the designed LQ controller. The same conclusion is reached by the solution of nonlinear equations of a validated model of the operation of the cement mill [12]. The compliance of the results obtained by applying the proposed state space analysis of the rectangular automata model ectness of twith those of a validated model is an indication of the correctness of the procedure.

5 Conclusions

In this work a method was proposed for verifying the satisfaction of specifications of a rectangular automata model of any hybrid system. Algebraic mathematical relationships were derived for finding whether a region of states that can be associated with the specifications of the rectangular automata model exists, without constructing the reachability tree of the automaton. These relationships express the range of values that each variable takes in each vertex as a function of the vertex duration. This duration in turn is approximated by a polynomial function of the number of the automaton iterations by curve fitting techniques. The vertex duration is defined as the time that the automaton stays in each vertex. The required data for applying the curve fitting technique are obtained from running the automaton an initial number of iterations and recording the observed times. Replacing the values that the variables take at the region of the states that the specifications define and checking by the use of the algebraic equations if there is a finite number of iterations which satisfy the algebraic equations, we verify the satisfaction of the specifications. A case study concerning the control of a cement mill has been used to demonstrate the application of the method. The comparison of the results with those obtained by solving a validated model of the mill consisting of a set of nonlinear differential equations has shown that the proposed method has predicted equally well the satisfaction of the control specifications.

References

1. Alur, R., Courcoubetis, C., Dill, D.: Model checking in dense real time systems, Information and Computation, Vol. 104, (1993) 2-34
2. Alur, R., Courcoubetis, C., Henzinger, T.A.. Ho, P-H: Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, Proceedings of Hybrid Systems I, Lecture Notes in Computer Science Vol. 736, (1993),209-229
3. Alur, R, Henzinger, T.A., Ho, P.H.: Automatic symbolic verification of embedded systems, IEEE Transactions on Software Engineering, Vol. 22, (1996) 181-201

4. Branicky, V., Borkar, S., Mitter, S.K.: A unified framework for hybrid control, Proceedings of the. eerd Conferance. Decision and Control, Lake Buena Vista (1994) 4228-4234.
5. Clarke, E.M., Emerson, E.A: Characterizing properties of algorithms as fixed points, Proceedings of the. 7th Int. Collog. Automata Languages and Programming, Lecture Notes in Computer Science, Vol. 85 (1981)
6. Demongogin, I., Koussoulas, N.T.: Differential Petri Nets: Representing Continuous Systems in a Discrete-Event World, IEEE Trans.Autom.Control, Vol 43, No 4, (1998) 573-579
7. Henzinger, T. A. Majumdar, R.: Symbolic Model Checking for Rectangular Hybrid Systems, S. Graf and M. Schwartzbach (eds.):TACAS/ETAPS, LNCS 1785, (2000) 142-156.
8. Henzinger, T. A., Kopke, P.T.: Discrete-time control for rectangular hybrid automata, Theoretical Computer Science Vol. 221, (1999) 369-392.
9. Henzinger, T.A., Copke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata?, Proceedings of the STOC'95, ACM, (1995) 373-382
10. Henzinger T.A., Rusu, V.: Reachability Verification for Hybrid Automata, HSCC' 98, LNCS, Vol. 1386, (1998) 190-204.
11. Macmillan, K.: Symbolic Model Checking, Kluwer Academic (1993)
12. Magni, L., Bastin, G., Wertz V.: Multivariable Nonlinear Predictive Control of Cement Mills. IEEE Transactions on Control Systems Technology. Vol. 7, No 4, (1999) 502- 508
13. Preubig, J., Kowalewski, S., Wong-Toi, H., Henzinger, T.A.: An Algorithm for the Approximative Analysis of Rectangular Automata, Proceedings of the Fifth International Symposium on Formal Techinques in Real-time and Fault-tolerant Systems (FTRTFT), Lecture Notes in Computer Science, Vol. 1486 (1998) 228-240
14. Raisch, J., O'young, S.D.: Discrete approximations and supervisory control of continuous systems, IEEE Trans. Automatic Control, Vol. 43, No 4, (1998) 569-573
15. Stiver, J.A., Antsaklis, P.J., Lemmon, M.D.: A logical DES approach to the design of hybrid control systems, Mathematical Computer Modeling, Vol. 23, (1996) 55-76.
16. Witsenhausen, H.S.: A class of hybrid -state continuous time dynamic systems, IEEE Transactions of. Automatic Control. Vol. 11, No. 2, (1966) 161-167