INTRODUCING A NEW MATHEMATICAL ABSTRACTION FOR MODELING REAL TIME SYSTEMS

Vasileios Deligiannis, Stamatis Manesis

University of Patras
Electrical & Computer Engineering Dept.
Division of Systems and Control
<bdeligiannis, stam.manesis>@ee.upatras.gr

Abstract: Due to the increasing complexity of industrial production systems, there exists a need for the development of efficient formal approaches for their analysis and control. Various methods have been proposed and examined from researchers, without being widely adopted for direct industrial use. In general, an industrial production line can be modelled as a Discrete Event System, but a more accurate representation would result, if we considered it as a real-time system. This paper presents a new mathematical abstraction for modelling real-time systems. In comparison with the conventional methods, the proposed method introduces new formulation parameters and handles variables in a different manner. It gives the opportunity to handle both discrete and real valued variables as inputs, outputs or both. A formal definition of the method is given and some examples of computations or runs of two typical examples are also presented.

Keywords: Real-time systems, discrete-event systems, formal method, automata, industrial production systems.

1. INTRODUCTION

In general, an industrial production line consists of various types of devices (e.g. robots, NC machines, actuators, sensors, etc.) controlled by either centralized or decentralized controllers. From a planning and control perspective, an industrial production line can be seen as a dynamic system whose states evolve according to the occurrence of abrupt physical events, thus exhibiting the characteristics of a Discrete Event System (DES). In the past, automated manufacturing DESs have usually been sufficiently simple that intuitive or ad-hoc control solutions have been adequate (Ramadge and Wonham, 1989). However, the increasing complexity of these systems and the requirement of fast system response have created a need of formal approaches for their analysis and control (Cassandras, 1993). Formal methods allow rigid proving of system properties in verification and validation.

Many methods have been proposed and examined from researchers for modelling DESs. Such methods are Petri nets, GRAFCET, finite automata and their extensions, hybrid, timed and PLC automata. These methods did not meet wide acceptance for industrial use, primarily because they are application depended.

- *Petri nets* were first introduced by C.A.Petri in the early 1960s and since then, they have become a powerful tool for modelling and analysis of dynamic DESs (Peterson, 1981).
- *GRAFCET* language was developed in 1977 as a tool for sequential systems aiming to be a formal specification method for logical controllers. In industry, GRAFCET, with minor changes, is better known under the name Sequential Function Charts (David, 1995).
- *Finite state automata* are probably the simplest mathematical abstractions of discrete event systems with a finite number of states and transitions between those states (Khoussainov, 2001).

- *Hybrid automata* are commonly used mathematical models for the analysis and design of hybrid systems. The hybrid automaton extends the classical notion of automaton by modelling the coupled interaction of discrete events and continuous dynamical systems (Allur, et al., 1993; Antsaklis, 2000; Henzinger, 1996).
- *Timed automata* were first introduced in (Allur and Drill, 1994) as a simple technique for modelling real-time systems. They are an extension of finite automata restricting transitions based on the values of multiple timers.
- *PLC automata* were first introduced in (Dierks, 1997) as a useful tool for the description of distributed real-time systems that are implemented on a PLC and are a subclass of timed automata. The main difference from them is that PLC automata are being restricted to only one clock and certain types of restraints on it.

On the contrary, an industrial production system would be defined more accurately as a real-time system. This paper, based on this assumption, presents a new mathematical abstraction for modelling real-time systems. The proposed method introduces new modelling parameters in comparison with conventional methods. It borrows some characteristics from several types of automata such as the control graph with a finite set of states and transition between those states. It can handle both discrete and real valued variables as inputs, outputs or both, combining flow conditions, invariants and guard conditions from hybrid automata, with clock constraints and delayed inputs from timed and PLC automata. In addition new parameters as reset table at each transition and hierarchical classification of executable events at each state are introduced.

The rest of the paper is organized as follows. Section 2 gives a formal definition of the new proposed method. In section 3, a comparison between the new method and conventional modeling methods, is given. Section 4 discusses about two case studies using the new formulation method. Finally, last section ends this paper with concluding remarks and open research problems.

2. A FORMAL APPROACH

In order to be able to model various forms of industrial systems, as real-time systems, was necessary to define a new formal method offering this convenience. Some common features, with other similar methods, have been conserved or extended to cover their weakness.

Definition 1. We define an automaton the structure of which is composed by the following sets:

- The system's variables:
  - Real-valued variables: $X = \{x_1, x_2, x_3, ..., x_m\}$
  - Discrete variables: $Z = \{z_1, z_2, z_3, ..., z_k\}$
- The set of states: $Q = \{q_1, q_2, q_3, ..., q_n\}$
- The alphabet or set of events: $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, ..., \sigma_\lambda\}$, which can be:
  - Discrete variables.
  - Conditions over the real-valued variables.
  - Any combination of them.
- Initial conditions: Init
  - $X = X_0$
  - $Z = Z_0$
  - q_0
- Flow conditions:
  - $F(X, \dot{X}) = 0$
  - $Z_{i+1} = G(Z_i)$
- Invariant conditions: $L = \{\ell_1, \ell_2, \ell_3, ..., \ell_n\}$
- Restrictions or safe values: $S = \{s_1, s_2, s_3, ..., s_n\}$
- The set of events to be ignored until the satisfaction of restrictions: $W = \{w_1, w_2, w_3, ..., w_n\}$ with $w_i \subseteq \Sigma$.
- The set of transitions: $E \subseteq Q \times Q \times \Sigma \times R_X \times R_Z$
- Reset table for each transition:
  - $X = R_X$
  - $Z = R_Z$

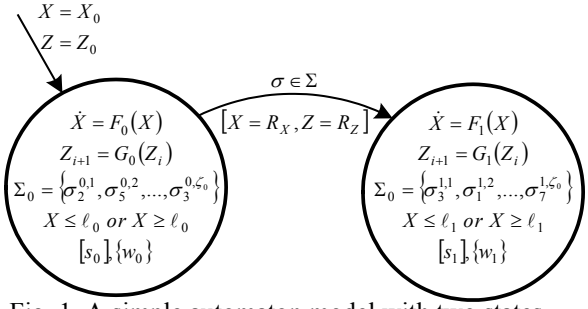

Fig. 1. A simple automaton model with two states.

Each set $(q, q', \sigma, r_X, r_Z)$ represents a transition from state q to state q', which is caused by the event $\sigma \in \Sigma$. The set $r_X \subseteq R_X$ gives the real-valued variables to be resettled during this transition, while the set $r_Z \subseteq R_Z$ gives the discrete variables.

Each state q_i has a corresponding set of parameters, which are:

- Flow conditions:
  - $F_i(X, \dot{X}) = 0$
  - $Z_{j+1} = G_i(Z_j)$
- Active events at the present state: $\Sigma_i \subseteq \Sigma$. Set Σ_i has, by definition, ζ elements, each one of which belongs to set Σ. $\Sigma_i = \{\sigma_j^{i,k}\}$, where i is the present state, $k = 1,2,...,\zeta$ and $j \in [1, \lambda]$. Index k also denotes transitions priority caused by different events. If two events occur simultaneously and cause two different transitions, transition with the lower index *k* will take place.
- Invariant conditions: ℓ_i
- Restrictions or safe values: s_i
- The set of events to be ignored until the satisfaction of restrictions: w_i.

3. VALIDATION BY COMPARISON WITH ALTERNATIVE METHODS

When a new method for solving a problem is presented, it must be compared with previous methods to prove whether it is better or not. In this section the proposed method is being validated by comparison with existing methods such as timed and hybrid automata.

3.1 Comparison with Timed Automata

Timed automata were introduced by (Allur and Drill 1994) and since then have become one of the most well studied and widely used models for real-time systems. They are an extension of finite automata.

Definition 2. Timed Automata: Formally a timed automaton is defined as a 5-tuple $A = \langle Q, \Sigma, i, C, E \rangle$ where:

- Q is a finite nonempty set of states,
- Σ is a finite nonempty set of events, called alphabet,
- $i_A \in Q_A$ is the initial state,
- C is a finite set of clocks and
- $E \subseteq Q \times Q \times \Sigma \times 2^C \times \Phi(C)$ gives the set of the transitions, where $\Phi(C)$ is defined as a set of clock constraints**.**

An edge *(q, q', a, λ, φ)* represents a transition from state q to state q' on input signal a. The set $\lambda \subseteq C$ gives the clocks to be resettled with this transition and φ is a clock constraint over C.

A comparison that one can make between timed automata and the proposed type, will lead them to the conclusion that a timed automaton can be translated to the new type without any problems. This primarily because most of the parameters are the same for both types and the rest of them can be replaced by other similar. Considering the definition of timed automata, as given above, one has to replace the 5-tuple $A = \langle Q, \Sigma, i, C, E \rangle$ with equivalent parameters. The first two, sets of states and events, are common and have the same concept for both types. The third one is the initial state. The new type's *Init,* except the initial state, has also initial conditions for all system variables. The last but one parameter is the set of clocks, which, in the new type, can be reproduced as real-valued variables governed by flow conditions of type $\dot{X} = 1$. Finally, the set of transitions, with the clock constraints and the resettled clocks, can be replaced with a suitable combination of new type's transitions set, reset table and ignored events at each transition.

A complete comparison between the two types must also show which are the benefits of using the new modelling method. The new type can handle, in addition, discrete variables with different flow conditions in each state. Besides, as already mentioned above, initial conditions regard all the automaton's variables and not only the initial state. Moreover, new type has restrictions for every variable and not only for clock variables. This, in combination with ignored events at each state, gives the opportunity to the designer to handle even the most complicated real-time systems.

3.2 Comparison with Hybrid Automata

Hybrid automata have been proposed as a formal model for hybrid systems and are a hyper-set of finite state automata, as they have in addition, continuous dynamics corresponding to each discrete state of the automaton. These dynamics are typically modelled via differential equations.

Definition 3. Hybrid Automata: A hybrid automaton, as defined by (Henzinger, 1996), consists of the following components:

- *Variables.* A finite set X of real-valued variables.
- *Control graph.* A finite directed multigraph *(V,E)*, where V are the states or locations and E are the transitions or switches.
- *Initial conditions* for every variable and the initial state.
- *Invariant conditions.* Limits for every variable.
- *Flow conditions,* according to which variables change their value. Flow conditions are, in general, differential equations.
- *Jump conditions,* which cause a transition between two states.
- *Events.* A finite set of events, whose members are assigned to each transition.

The most typical example (Antsaklis, 2000) of hybrid automata is depicted at figure 2 and models a house thermostat. The real valued variable x represents the temperature. The two states are labelled as *On* and *Off* and each one has a flow condition according to which the temperature rises or falls. Initially the temperature is 20 degrees and the heater is off. According to the jump condition the temperature falls, governed by the differential equation $\dot{x} = -0.1x$, until the threshold of 18 degrees. Then the heater turns on and the temperature rises according to the flow condition $\dot{x} = 10 - 0.1x$, until the upper limit of 23 degrees, where the heater returns to off mode.

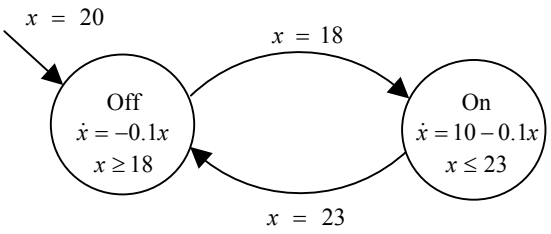

Fig. 2. A thermostat's hybrid automaton model.

The automaton shown at figure 2 could be also an automaton, which has been designed based on the new formal method. This derives from the fact that the new type is a hyper-set of hybrid automata including all hybrid automata's parameters. Illustrating this conclusion we have to make a

comparison between the two types. Most of hybrid automata's components are exactly the same as at the new type, such as variables, events, initial, invariant and flow conditions. Control graph has been replaced by the set of states and the set of transitions. Finally, jump conditions are including at the set of events, as defined at section 2.

As denoted above, the new modelling method, in comparison with hybrid automata, has an additional group of parameters. First of all new type has the ability to handle discrete variables with initial conditions and different flow conditions at each state. In addition, new automata have transition restrictions if the relative criteria are not accomplished. Restrictions take effect only to the ignored events at each state. Finally, the proposed method resets all the system variables at each transition.

4. TWO CASE STUDIES USING THE PROPOSED FORMAL METHOD

4.1 The three machines' stop problem.

Let us suppose that three similar machines start and stop manually through an equal number of start–stop buttons. For the start handling of the three machines the process does not demand any special requirement. For the stop handling however, due to operational reasons, the action takes place immediately for every machine except the last operated one, which must be stopped only if an input signal (e.g. a timer for 30sec) allows it. It is obvious that the other two machines can stop independently of the situation of the input when the corresponding stop button is pressed. Furthermore, the last operated machine is not predefined or constant. On the contrary, it is a stochastic parameter and hence can be any of the three machines. If input has been activated but the stop button of the last operated machine has not been pressed, the machine continues to operate. Summing up, we can claim that any of the three machines can be the last operated one, which must stop in combination with an input while the other two machines will stop in the usual way.

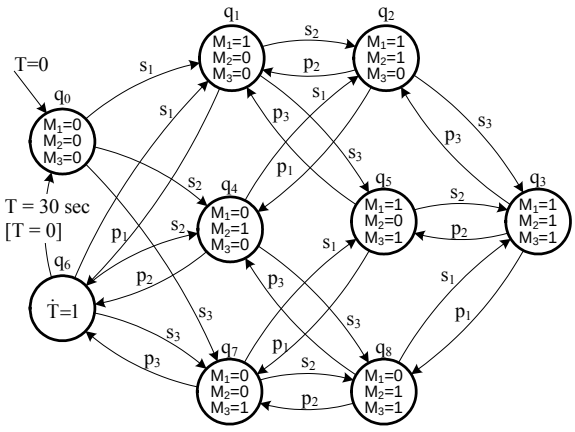

Fig. 3. Full state diagram for the three machine's stop problem.

In figure 3 the state diagram with all operating combinations of the three machines is shown. Let s_i and p_i denote the signals "start" and "stop" of machine i respectively and M_i denotes the operating status of each machine (i = 1, 2, 3). States q_1, q_4 and q_7 are the states where only one machine operates (hence is the last one) and from which the transition to q_0 requires the intermediate state q_6. This state diagram seems alike to common finite state diagrams, as it has the same number of states and equivalent transitions. But, if we try to take advantage of every capability the new type of automata offers, we can succeed to have state aggregation. The transformation merges states q_1, q_4 and q_7 to a new state labelled "Only one machine operates" and states q_2, q_5 and q_8 to a new state, where two machines operate. The new state diagram has only five states, compared with nine at the initial diagram, and is shown at figure 4.

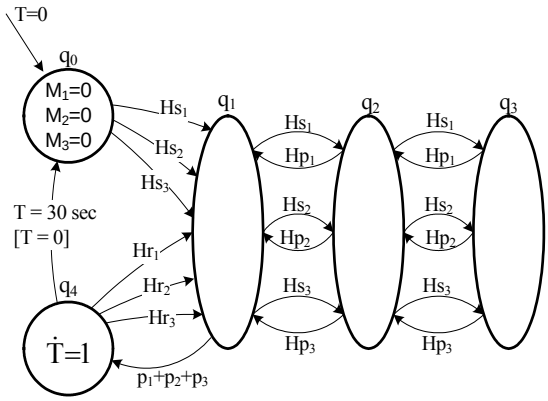

Fig. 4. Reduced state diagram with five states.

The notation at figure 4 has to be clarified. Firstly, states q_1 and q_2, have six couples of restrictions and events to be ignored. Particularly first three are, $[M_i=1]$, $\{s_i\}$, where $i=1,2,3$ and the latter three are $[M_i=0]$, $\{p_i\}$, where $i=1,2,3$. State q_3, where all machines operate, has no restrictions. All transitions labeled Hs_i are triggered by the events s_i and cause the set of discrete variables M_i to value 1 respectively. Equivalently, transitions Hp_i are triggered by the events p_i and cause the reset of discrete variables M_i to value 0. Finally, transitions Hr_i, which are caused by the events s_i, set discrete variables M_i to value 1 and reset discrete variables M_j (where $j \neq i$) to value 0.

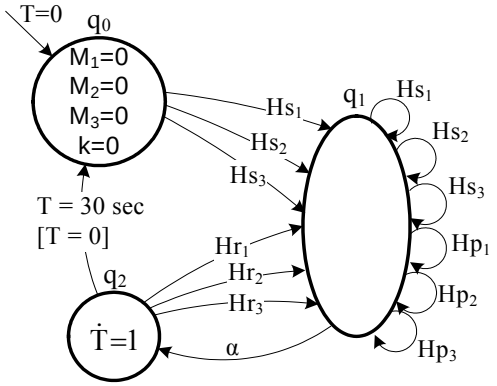

Fig. 5. Reduced state diagram with three states.

Another approach, adding a new variable *k* that is the number of operating machines, is shown at figure 5. The new state diagram has only three states, as it merges states q_1, q_2 and q_3 to a new state in which at least one machine operates.

State q_1 now has seven couples of restrictions and events to be ignored. Six first are exactly the same as before, $[M_i=1]$, $\{s_i\}$, and $[M_i=0]$, $\{p_i\}$, where $i=1,2,3$, and seventh is $[k=1]$, $\{p_1, p_2, p_3\}$. Similarly to the previous approach, all transitions labeled Hs_i are triggered by the events s_i and cause the set of discrete variables M_i to value 1 and increase counter's k value by one unit. Equivalently, transitions Hp_i are triggered by the events p_i and cause the reset of discrete variables M_i to value 0 and the decrease of counter k. Transitions Hr_i, which are caused by the events s_i, set discrete variables M_i to value 1, reset discrete variables M_j (where $j \neq i$) to value 0 and initialize counter k to value 1. Finally, event α is defined as $\alpha = M_1\overline{M}_2\overline{M}_3 p_1 + \overline{M}_1 M_2 \overline{M}_3 p_2 + \overline{M}_1\overline{M}_2 M_3 p_3$. A run or computation of this automaton on an input sequence $u = s_2, s_1, p_1, \alpha, s_3, \alpha, T = 30\,\text{sec}$ is

$$\left.\begin{array}{c} q_0 \\ M=(0,0,0) \\ T=0 \\ k=0 \end{array}\right| \xrightarrow{s_2} \left.\begin{array}{c} q_1 \\ M=(0,1,0) \\ T=0 \\ k=1 \end{array}\right| \xrightarrow{s_1}$$

$$\left.\begin{array}{c} q_1 \\ M=(1,1,0) \\ T=0 \\ k=2 \end{array}\right| \xrightarrow{p_1} \left.\begin{array}{c} q_1 \\ M=(0,1,0) \\ T=0 \\ k=1 \end{array}\right| \xrightarrow{\alpha}$$

$$\left.\begin{array}{c} q_2 \\ M=(0,1,0) \\ T=0 \\ k=1 \\ \dot{T}=1 \end{array}\right| \xrightarrow{s_3} \left.\begin{array}{c} q_1 \\ M=(0,0,1) \\ T=0 \\ k=1 \end{array}\right| \xrightarrow{\alpha}$$

$$\left.\begin{array}{c} q_2 \\ M=(0,0,1) \\ T=0 \\ k=1 \end{array}\right| \xrightarrow{T=30\,\text{sec}} \left.\begin{array}{c} q_0 \\ M=(0,0,0) \\ T=0 \\ k=0 \end{array}\right|$$

The notation:

$$\left.\begin{array}{c} q_0 \\ X=X_0 \\ Z=Z_0 \end{array}\right| \begin{array}{c} \dot{X}=F_0(X) \\ Z_{i+1}=G_0(Z_i) \end{array} \xrightarrow{\sigma} \left.\begin{array}{c} q_1 \\ X=R_X \\ Z=R_Z \end{array}\right| \begin{array}{c} \dot{X}=F_1(X) \\ Z_{i+1}=G_1(Z_i) \end{array}$$

denotes the transition from state q_0 to state q_1 caused by the event σ. The two equations under each state initialize the system variables according to reset table for transition σ. The other two differential equations are current state's flow conditions.

The diagram of figure 5 after the appropriate modification can be used for modeling a system with n machines without adding new states. The new automaton is depicted at figure 6.

All the changes made concern primarily the number of transitions between the states maintaining the type of existing transitions. Another change has been made at event α, which now is $\alpha = M_1\overline{M}_2 \ldots \overline{M}_n p_1 + \overline{M}_1 M_2 \ldots \overline{M}_n p_2 + \ldots + \overline{M}_1\overline{M}_2 M_n p_n$.

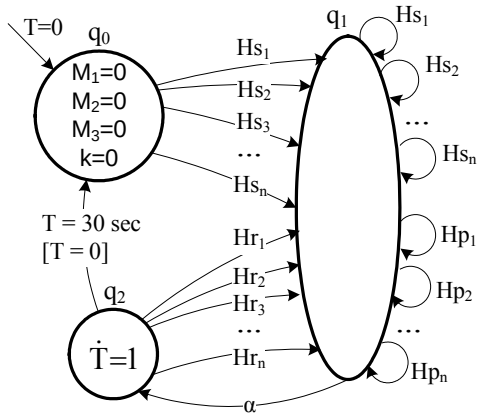

Fig. 6. State diagram for the n machine's stop problem according to the proposed formal method.

4.2 Token Passing Bus Protocol.

The automaton of figure 7 models a local area network with four nodes using the token passing bus protocol. The four states reconstruct the virtual ring between the nodes. When the automaton is in q_i state, *i* node has the token and the capability to use the network. The variable *t* represents the time and τ is an additional clock variable. The variables M_1, M_2, M_3 and M_4 represent the number of queuing packets in each node. The discrete variables s_1, s_2 s_3 and s_4 represent the arrival of a new packet at the queuing list in each node. The constant parameter T_s is the token's transmission time, while P_s is a single data packet's transmission time.

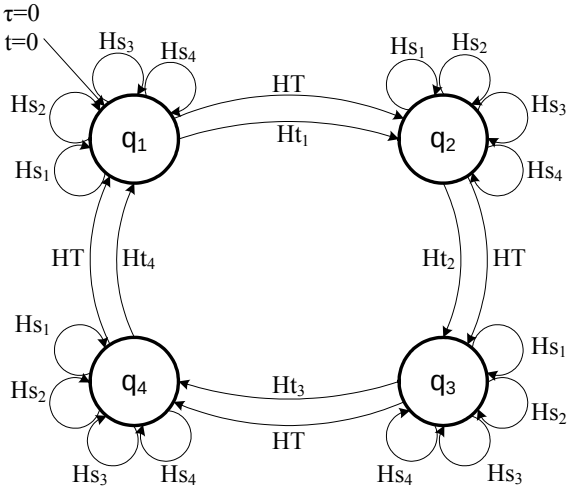

Fig. 7. Token Passing Bus Model with four nodes.

All transitions labeled Hs_i are triggered by the events s_i and increase counters M_i by one unit, for $i=1,2,3,4$. In the other hand, transitions Ht_i are triggered by the event $\tau = T_s + P_s$ and cause the reset of variable τ to zero value and decrease counters M_i by one unit, for $i=1,2,3,4$. Finally, transitions HT are caused be the event $\tau = T_s$ and reset variable τ to zero value. In

each state are two flow conditions, one for each clock variable: $\dot{t}=1$ and $\dot{\tau}=1$. There is also a couple of an ignored event and a restriction, which are: $[\tau=T_s],\{M_i>0\}$. This means that when the automaton is at q_i state and $M_i>0$, the event $\tau=T_s$ will be ignored. So, a transition from state q_i to state q_{i+1} takes place after T_s time units, if $M_i = 0$, or else after T_s+P_s time units.

Initially the automaton is in q_1 state and stays there until $t=T_s$, if $M_1=0$, or until $t=T_s+P_s$, if $M_1>0$, where M_1 is the number of queuing packets at the queuing list of node 1. The above condition means that if node 1 has not any packets to send, the bus's management will pass to node 2 after the token's time (T_s) elapses. In the other case ($M_1>0$), node 1 will send the first packet of his list and the transition to state 2 will take place after T_s+P_s time units (token's time and time for one packet respectively). These two different types of transitions cause the change from one state to the following one.

Each state has also four loop transitions caused by the events s_i, which represent the arrival of a new packet at the queuing list of node i. Consequently, the corresponding counter M_i increases by one unit.

A run of this automaton is shown above presenting the system's states and variables in connection with time. For this example we assumed that $T_s=1$ and $P_s=2$.

$$\left.\begin{array}{c|c|c|c|c|c|c} q_1 & & q_2 & & & & \\ M=(1,0,2,2,0) & \xrightarrow{\tau=3} & M=(0,0,2,2,0) & \xrightarrow{\tau=1} & & & \\ t=0 & \dot{t}=1 & t=3 & \dot{t}=1 & & & \\ \tau=0 & \dot{\tau}=1 & \tau=0 & \dot{\tau}=1 & & & \end{array}\right.$$

$$\left.\begin{array}{c|c|c|c|c|c|c} q_3 & & q_3 & & & & \\ M=(0,0,2,0,0) & \xrightarrow{s_2} & M=(0,1,2,0,0) & \xrightarrow{\tau=3} & & & \\ t=4 & \dot{t}=1 & t=5 & \dot{t}=1 & & & \\ \tau=0 & \dot{\tau}=1 & \tau=1 & \dot{\tau}=1 & & & \end{array}\right.$$

$$\left.\begin{array}{c|c|c|c|c|c|c} q_4 & & q_1 & & & & \\ M=(0,1,1,1,0) & \xrightarrow{\tau=1} & M=(0,1,1,1,0) & \xrightarrow{\tau=1} & & & \\ t=7 & \dot{t}=1 & t=8 & \dot{t}=1 & & & \\ \tau=0 & \dot{\tau}=1 & \tau=0 & \dot{\tau}=1 & & & \end{array}\right.$$

$$\left.\begin{array}{c|c|c|c|c|c|c} q_2 & & q_2 & & & & \\ M=(0,1,1,1,0) & \xrightarrow{s_1} & M=(1,1,1,1,0) & \xrightarrow{\tau=3} & & & \\ t=9 & \dot{t}=1 & t=11 & \dot{t}=1 & & & \\ \tau=0 & \dot{\tau}=1 & \tau=2 & \dot{\tau}=1 & & & \end{array}\right.$$

5. CONCLUSIONS – FURTHER WORK

Industrial systems are usually described as DESs and mostly controlled by Programmable Logic Controllers (PLC). Modelling DESs is an open research field and many modelling methods have been proposed, such as Petri nets and controlled automata. The main problem is that there is not consensus on which is the most suitable as the "lingua franca" for DESs, primarily because most methods are strongly problem dependent. In addition, most industrial applications require mixed modelling with both continuous and discrete components. Based on this assumption, an industrial production system would be defined more precisely as a real-time system.

This paper presents a mathematical abstraction for modelling real-time systems as a new formal method. This method has some common features with existing methods and introduces some new modelling parameters. It handles both discrete and real valued variables and seems to be application independent, as shown at the given examples. Of course further work must be done so as to have a formal representation method for real-time systems. Additionally, a new software tool for modelling, verification and simulation of industrial systems must be developed, based on this new formal method.

ACKNOWLEDGEMENTS

This research work has been partially supported by the Caratheodory Program of the Research Commission of the University of Patras.

REFERENCES

Allur, R., C. Courcoubetis, T.A. Henzinger and P-H Ho (1993). Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems", *Hybrid Systems,* Lecture Notes in Computer Science, **Vol. 736**.

Allur, R. and D.L. Drill (1994). A Theory of Timed Automata, *Theoretical Computer Science*, 126:183-235.

Antsaklis, P.J. (2000). A Brief Introduction to the Theory and Applications of Hybrid Systems, *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications,* **Vol. 88**, pp. 879-887.

Cassandras, C.G. (1993). *Discrete Event Systems: Modeling and Performance Analysis*, Richard D. Irwin Inc., Boston (MA).

David, R. (1995). GRAFCET - a powerful tool for specification of logic controllers. *IEEE Trans. on Control Systems Technology,* **Vol. 3**, No. 3, pp. 253-268.

Dierks, H. (1997). PLC-Automata: A New Class of Implementable Real-Time Automata, *ARTS'97*, LNCS, Springer Verlag.

Henzinger, T.A. (1996). The theory of hybrid automata, In Proceedings of the *11th Annual Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 278-292.

Khoussainov, B. and A. Nerode (2001). *Automata theory and its applications*. Birkhauser, Boston.

Peterson, J.L. (1981). *Petri Net Theory and the Modelling of Systems*, Prentice-Hall Inc. New Jersey.

Ramadge, P. and W.M. Wonham (1989). The Control of Discrete Event Systems, *Proceedings of the IEEE*, **Vol. 77**, No.1, pp.81-98.