DESIGN AND IMPLEMENTATION OF A MONITORING SYSTEM USING GRAFCET

Adib Allahham and Hassane Alla
GIPSA-Lab, Department of Control, Institute National de Polytechnique de Grenoble
961 Rue de la Houille Blanche - Domaine universitaire BP 46, 38402 Saint Martin D'hères, France
adib.al-lahham@inpg.fr, hassane.alla@inpg.fr

Keywords: Monitoring, Fault detection, Manufacturing systems, Stopwatch automata, Reachability analysis, Grafcet.

Abstract: A monitoring system based on a stopwatch automaton is proposed to detect the system faults as early as possible. Each location in the automaton corresponds to a system's situation. Its time space delimits exactly the range of the normal behavior in the corresponding system's situation. The monitoring system detects a fault when the time space corresponding to the actual system's situation is violated. The stopwatch automaton provides a formal foundation to model the system's behavior and to synthesize the exactly time space in each location. This paper aims to provide the grafcet monitor that allows to link the design of the monitoring system of a system with its implementation in a programmable logic controller.

1 INTRODUCTION

Monitoring complex manufacturing systems plays an important role for economic and security reasons. A wide variety of methods has been considered this problem. These methods consider a fault have occurred in a system if a faulty event occurs (Ghazel et al., 2005), reaching a faulty state (S. H. ZAd and Wonham, 2003) or more generally violating system specifications. Most systems monitor the timed system specifications by using Watchdogs. They detect a fault if the expected observation is produced early or late with respect to certain time bounds.
The increasingly stringent requirements in monitoring and fault detection problems lead to the necessity to detect the fault as early as possible without waiting the expiration of certain bounds. For that, we have proposed in (A.allahham and alla, 2006) a monitoring method which extends the method of residuals, well-known in continuous system. In (A.allahham and alla, 2006), we have introduced the notion of acceptable behavior of a system detailed in the following section. We model this acceptable behavior by a stopwatch automaton. In that representation, each location corresponds to a state of the system and the arcs are labeled by switching conditions between the different states. In each state, the differential equations express the progression or suspension of the task represented by the stopwatch due to a fault. The time sub-space in each location represented by a set of algebraic inequalities, delimits the range of stopwatches in the corresponding system's situation in the acceptable behavior. The monitoring system detects a fault when the system exceeds this time sub-space.
The stopwatch automaton provides a formal basis to model the system's behavior and to analyze it in order to characterize the exact time sub-space in each location, corresponding to the acceptable behavior.
In this paper, our objective is to provide the grafcet model that allows to link the design of monitoring system of a manufacturing system with its implementation in the logic controller. We show that the grafcet fulfils not only the sequential specification of the applications but also the continuous behavior specified in the monitoring stopwatch automaton.
The grafcet corresponding to monitoring automaton models a location by a step and a stopwatch by a timer where the following problem is encountered. The behavior of a stopwatch goes beyond the ability of a timer representing the simplest way to include the time in grafcet model. This problem in turn affects the method to represent the time sub-space associating to the steps of grafcet. However, we will show that this problem can be overcome by complet-

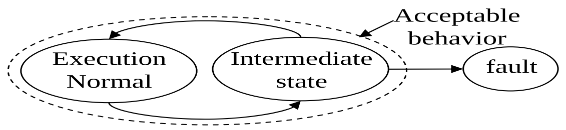

Figure 1: Acceptable behavior of a system.

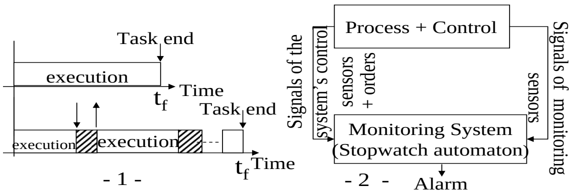

Figure 2: 1- Behavior of an interruptible task 2- Inputs\Output of monitoring system.

ing the grafcet by actions associated with steps. Also, the grafcet will monitor permanently the consistency of the stopwatches within its acceptable range.

Section 2 describes the acceptable behavior of a system and its model based on stopwatch automaton. Our approach is given and used to delimit the time space characterizing this behavior. In Section 3, the method to translate a monitoring automaton into a Grafcet model is detailed. We apply this method in an illustrative example in Section 4.

2 THE ACCEPTABLE BEHAVIOR

The possible kinds of faults that affect the resources in a manufacturing system are the permanent faults, which dispossess a resource's ability to perform its task and the intermitting faults. These faults can appear several times during the task execution and disappear without any external action on the system while permanent faults disappear due to a repair of the fault (Huang et al., 1996). Our work considers only the intermitting faults that interrupt the task of a resource. We call it *malfunctions* and the task subjected to these malfunctions as *interruptible task*. The system containing these tasks is called as *interruptible system*. Because of malfunctions, an intermediate state can appear between a normal state and a faulty one. In this state, the system can come back to the normal behavior or it leaves toward a faulty state (Fig.1). We refer to this behavior by *acceptable behavior*. These malfunctions occur often in a manufacturing system, so the system's designer accepts to some extent this behavior for productivity motives. The question to answer is: how the designer takes into account these malfunctions in his system.

Let be a task $Task_i \in Task_{int}$ where $Task_{int}$ represents the set of interruptible tasks in a complex system S. $Task_i$ has a known execution duration $[\alpha_i, \beta_i]$ which is given in the technical characteristics of the resources that execute $Task_i$ or measured directly. Because of the interruptions resulting from malfunctions, the designer accepts a tolerated duration to execute $Task_i$. It is given by the interval $[\alpha_i, \gamma_i)$ where $\beta_i < \gamma_i$. We call $[\alpha_i, \beta_i]$ and $[\alpha_i, \gamma_i)$ respectively the normal and acceptable durations of $Task_i$.

2.1 Monitoring of an Interruptible Task

We refer to the apparition and disappearing of a fault by its effect on the task execution, then we refer it by *interruption* and *resuming* of the task.

Hypothesis 1 *The execution speed is supposed to be constant or to vary sightly around a mean value.* □

Considering the properties of the tasks mentioned above, we distinguish the behavior of an interruptible task shown in Figure 2.1. Either $Task_i$ is executed without interruption, then $t_f \in [\alpha_i, \beta_i]$ or $Task_i$ has been executed but with several interruptions. After each interruption, the system resumes from the position at which it has been interrupted. In this case: $t_f \in [\alpha_i, \gamma_i)$.

To monitor $Task_i$, we use the timers x_i and y_i. The timers x_i and y_i have a values "0" when the task begins. x_i will be used to check that $Task_i$ has completed before the expiration of its tolerated deadline. y_i is used to monitor the effective time of execution. Then, $Task_i$ is correctly executed if $y_i \in [\alpha_i, \beta_i]$ and $x_i \in [\alpha_i, \gamma_i)$ when the task end occurs.

The arrows $\downarrow$ and $\uparrow$ in Figure 2.1 represent respectively the signal of logical sensor which detects the interruption and resuming of $Task_i$. These signals represent an input of our monitoring system (Fig. 2.2).

2.2 Modeling of an Interruptible System

We use the stopwatch automata *SWA* to model the interruptible system. It is a class of linear hybrid automaton where the time derivative of a clock in a location can be either 0 or 1 (Cassez and Larsen, 2000).

Definition 1 *A stopwatch automaton is a 7-tuple* $(L, l_0, X, \Sigma, A, I, \dot{X})$ *where:*
• *L is a finite set of locations, l_0: the initial location,*
• *X is a finite set of stopwatches,*
• *Σ is a finite set of labels,*
• *A is a finite set of arcs. $a = (l, \delta, \sigma, R, l') \in A$ is the arc between the locations l and l', with the guard $\delta \in C(X)$, the label name σ and the set of stopwatches to reset R. $C(X)$ is the set of constraints over X.*

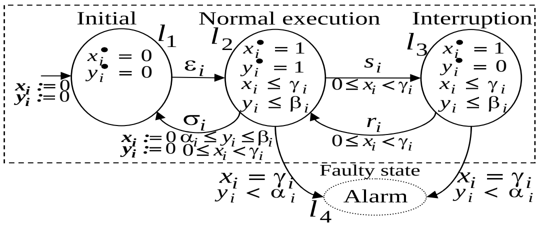

Figure 3: Stopwatch automaton of an interruptible task.

- $I \in C(X)^L$ *maps an invariant to each location,*
- $\dot{X} \in (\{0,1\}^X)^L$ *maps an activity to each location.* □

- ***SWA of an interruptible task***

We model the acceptable behavior of $Task_i$ by the Stopwatch automaton shown in Fig. 3. The location l_1 indicates that the resource is waiting to start the task, l_2 that the resource is executing its task and l_3 that the task is interrupted after having started. In this automaton, the clock y_i in l_3 does not progress while x_i evolves to express that the task is interrupted but the time remains progressing. The labels s_i and r_i represent respectively the stop and the resumption of $Task_i$ in the physical system, while label σ_i corresponds to the end of this task. ε_i which is the always true event, represents the necessary condition to start the task. Here it starts immediately.

The guard g_2 of the arc $l_2 \xrightarrow{g_2} l_3$ expresses that the interruption can occur at any instant during the acceptable duration while the guard g_3 associated to $l_3 \xrightarrow{g_3} l_2$ expresses that the resumption must occur before exceeding the acceptable duration. The execution of $task_i$, during its acceptable duration is represented by the guard g_4 of the arc $l_2 \xrightarrow{g_4} l_1$.

Figure 3 shows that $Task_i$ leaves the acceptable behavior to faulty state l_4 either from the location l_2 or l_3. The guards of arcs towards l_4 are identical and given by $g_5 = \neg g_4 = (x_i = \gamma_i \wedge y_i < \alpha_i)$. It expresses the fact that the acceptable duration of execution was expired and $Task_i$ is not executed.

2.3 Time Space State Delimiting the Acceptable Behavior

The acceptable behavior of a system S is represented by a stopwatch automaton $\mathbb{A}$. It is obtained by the composition of the different tasks automata according to the system specifications which represent the relation between these tasks.

Property 1 *The trajectories which lead $Task_i$ to the state $l_1 \times (0,0)$ from $l_2 \times (x_i, y_i)$ where $x_i \in [\alpha_i, \gamma_i)$ and $y_i \in [\alpha_i, \beta_i]$, represent all the possible evolutions characterizing the execution of $Task_i$.* □

The trajectories specified in Property 1 represent only a part of the possible ones. Thus, the synthesis problem of monitoring can be set as follows: given a stopwatch automaton $\mathbb{A}$ representing a system S, restrict the possible trajectories of this automaton in a way that all remaining ones satisfy Property 1, for all the tasks of S. As a result, we obtain an automaton $\mathbb{A}^*$ where all its trajectories characterize the acceptable execution of S. The calculation of the time space containing these trajectories E^* of $\mathbb{A}^*$ is the core of our synthesis algorithm. This is realized using of the Forward and backward reachability analysis. (Alur et al., 1995)

- ***Forward analysis of monitoring SWA:***

We use the forward analysis operators to calculate all the possible trajectories in the system. In other words: the reachable time space E in the automaton $\mathbb{A}$ mentioned above. The forward operators look for all the reachable states of a stopwatch automaton from its initial state remaining in the locations of automaton while the time progresses or by firing its transitions. The reachable time space by forward analysis in locations l_2 and l_3 of the automaton shown in Figure 3 is given in Figure 4.1. Note that the values of the stopwatches given by g_4 in Figure 3 define a polyhedron. We denote it as D_i, and call it as *the desired space* of $Task_i$ (Fig 4.2). Note also that the trajectories specified in Property 1 lead the task only to D_i. These trajectories represent only a part of the ones which are contained in reachable time space (Fig. 4.1). Thus,we must delimit the time space containing only these trajectories to characterize the acceptable execution.

- ***Backward analysis of monitoring SWA:***

It is not hard to see that the time space E^* of $\mathbb{A}^*$ can be obtained by removing from the time space of $\mathbb{A}$ the states from which system's evolutions do not lead to D_i of each interruptible task. In other words, one needs first to apply the backward operators (called as predecessors and annotated as *Pre* operators) to the guards of arcs representing the desired space of all the tasks over the automaton $\mathbb{A}$. Then, $E^* = E \cap (\bigcup Pre(D_i))$. The intuition behind the using the predecessors operators for a guard representing D_i of $Task_i$ is that we look for all the states that lead to this space D_i from the initial state of $\mathbb{A}$.

Applying the backward analysis for the automaton given in Figure 3 gives the time space shown in Figure 4.3. The intersection of this space and that of forward analysis is given in Figure 4.4. It is the space characterizing the execution acceptable of $Task_i$. One of the trajectories contained in synthesized space (Fig. 4.4) shows that the task reaches a faulty state, only from the location l_3 with the dynamics $\dot{x} = 1$ and $\dot{y} = 0$. Figure 4.5 presents the final monitoring automaton $\mathbb{A}^*$.

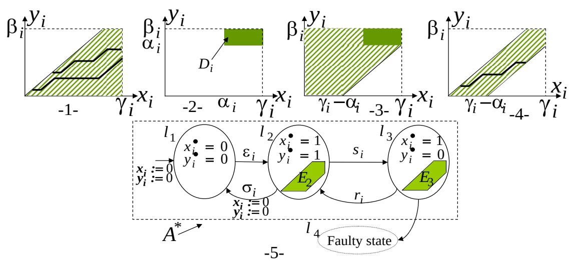

Figure 4: Time space in l_2 and l_3: (1) reachable by forward analysis, (2) desired, (3) reachable by backward analysis (4) delimiting acceptable execution (5) Synthesized Monitoring automaton of an interruptible task.

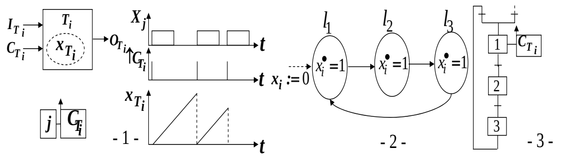

Figure 5: (1) Timer (2) A part of monitoring automaton (3) Corresponding grafcet G_1.

3 GRAFCET OF THE MONITORING SYSTEM

Grafcet and its international standard SFC (CEI/ IEC 60848 revised in 2002) are used for the implementation of discrete events models for manufacturing systems and many programmable logic controllers use it as a programming language. The basic concepts of the grafcet are: the step, action, transition and its associated receptivity (David, 1995). A Boolean variable X_i is associated with each step. Its value is 1 when step is active.

The general idea to translate the monitoring automaton $\mathbb{A}^*$ into a grafcet is to represent each location of the automaton by a step. The faulty state is also modeled by a step. Let $L = \{l_1, ..., l_n\}$ be the set of locations of $\mathbb{A}^*$. The set of steps corresponding to these locations is denoted by $\{1, ..., n\}$. An arc linking two locations is modeled by a transition linking the two corresponding steps. The transition receptivity is the label of the arc. The simplest way to include time in the grafcet model is to use timer objects, for that, each stopwatch will be modeled by a timer.

Figure 5 shows a timer (T_i) which is typically initialized with a value representing a duration (I_{T_i} input) and a control input (C_{T_i}) for starting the timer. This timer produces a boolean output (O_{T_i}). Associating an impulse action $\uparrow C_{T_i}$ with a step j will activate the timer T_i as soon as $\uparrow X_j = 1$. Here, we are not interested in the logic output of timer, but in the instantaneous value of the timer T_i denoted by x_{T_i}, which is

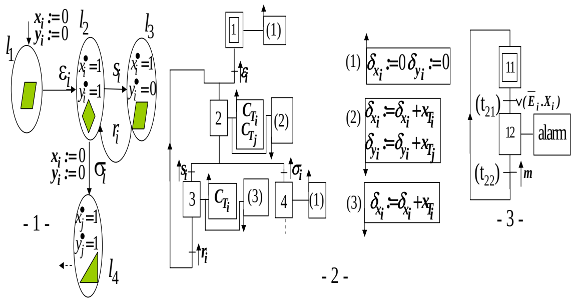

Figure 6: (1) A part of monitoring automaton (2) G_1 and shifting and initiation actions (3) G_2 model.

supposed to be readable and testable in real time. In fact, many *PLC* manufacturers provides products with timers equipped with functions permitting to read and test the value x_{T_i}.

In these translation rules, the behavior of a stopwatch goes beyond the ability of a timer. To show that, we consider the part of monitoring automaton shown in Figure 5.2. In this automaton the stopwatch x_i is newly activate in l_1 and remains active in l_2 and l_3. Translating this model into a grafcet by using the method described above, gives the model shown in Figure 5.3 where T_i is the timer corresponding to stopwatch x_i. In this grafcet, we activate the timer T_i as soon as the $\uparrow X_1 = 1$. T_i remains active in steps 2 and 3. However this is not sufficient to represent the behavior of the monitoring automaton since an important issue is the behavior at the firing the arc of automaton between l_3 and l_1. The stopwatch x_i persists active after the commutation and has a certain value at the instant of reaching l_1, while there will be an initialization of the value of corresponding timer T_i when $\uparrow X_1 = 1$ in the grafcet. However, we show that this problem can be overcome by completing the grafcet by actions and by using intermediate variables.

• ***Modeling of stopwatches by timers:***

Let us consider that the automaton given in Figure 6.1 follows the behavior given in Figure 7. T_i and T_j are the timers corresponding to stopwatches x_i and y_i. We express the dynamics $\dot{x}_i = 1$ and $\dot{y}_i = 1$ in the location l_2 by associating to step 2 the impulse actions $\uparrow C_{T_i}$ and $\uparrow C_{T_j}$. These actions will activate T_i and T_j as soon as $\uparrow X_2 = 1$. In a similar way, we express the dynamic $\dot{x}_i = 1$ in l_3. We will now give the method to represent the behavior of x_i and y_i whose values are 0 at the entry of l_2. Note that the value of x_i in a given location l_2 or l_3 is the sum of: the value of x_i when the system reaches this location and the passed time from the reaching instant to actual one.

The latter item corresponds to the value of timer T_i which is activated when the system reaches the step corresponding to the given location. For the for-

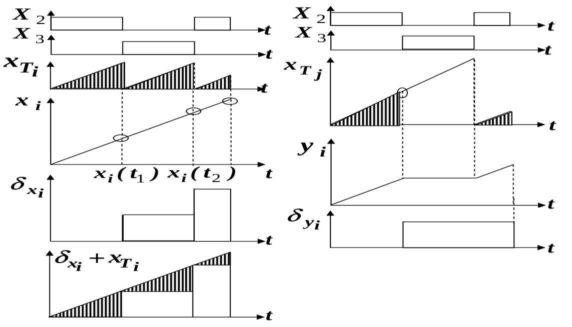

Figure 7: Representing behavior of stopwatches in grafcet.

mer item, an intermediate variable denoted by δ_{x_i} and called as *shifting variable* is used. δ_{x_i} is initialized when the automaton resets to 0 the stopwatch x_i. The value of δ_{s_i} corresponding to $x_i(t_1)$ in Figure 7 can be obtained by associating to the step 2 (Fig. 6.2) the impulse action $\downarrow \delta_{x_i} := \delta_{x_i} + x_{T_i}$ (Shifting action). It adds to δ_{s_i} whose initially has the value 0, the value of x_{T_i} representing the duration that the grafcet stays in step 2. The value of δ_{x_i} corresponding to $x_i(t_2)$ in Figure 7 can be obtained by associating to step 3 the same action. It adds to previous value of δ_{x_i} the duration that the system rests in step 3. The resulting values of δ_{x_i} are shown in Figure 7. They correspond to that of x_i at the instants of reaching l_2 and l_3 after each commutation between these two locations. As a result, $\delta_{x_i} + x_{T_i}$ is equivalent to that of x_i at any instant during the system dynamics either in l_2 or l_3.

The behavior of stopwatch y_i is different from that of x_i. y_i is suspended when the automaton fires from l_2 to l_3. y_i resumes in location l_2 from the same value when it was suspended, then we associate the action $\downarrow \delta_{y_i} := \delta_{y_i} + x_{T_j}$ to step 2 to memorize this value. δ_{y_i} is initialized when the automaton resets to 0 the stopwatch y_i. The describing exactly the given part of automaton is given in Figure 6.2.

In Figure 6.1, x_i and y_i are initialized by firing the arc $l_2 \rightarrow l_4$. Our grafcet does this resetting by allocating to zero the variables δ_{x_i} and δ_{y_i} after the firing from step 2 to 4. The action resetting the shifting variables will be associated to the step 4. The initial step of is associated by an impulse action resetting all the shifting variables used in the grafcet.

The grafcet monitor checks permanently the time space associated to the actual step. The faulty step is reached when the system violates this time range. This fact can be represented in the grafcet model by using the concept of hierarchy. It is easy to imagine that a grafcet G_1 has an influence on anther grafcet G_2. G_1 is the Grafcet resulting from structural translation described above (Fig. 6.2). G_2 has two steps: initial and faulty steps (Fig. 6.3). The activation of initial step of G_2 expresses that the system's behavior is acceptable. G_2 evolves to faulty step when the time space is violated. Let $E_1,..E_i,..,E_n$ be the time subspace in the locations $l_1,..,l_i,..,l_n$ permitting to evolve to the faulty state. The corresponding steps in grafcet G_1 are $1,..,i,..,n$. The receptivity of t_{21} in Figure 6.3 is: $[X_1.\overline{E_1} + .. + X_i.\overline{E_i} + .. + X_n.\overline{E_n}]$. In Figure 6.3, the event $\uparrow m$ represents the reparation operation.

4 APPLICATION

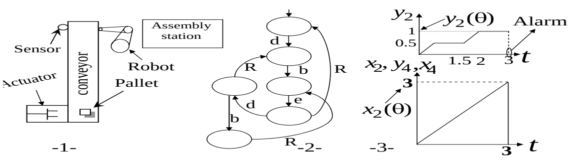

Figure 8: -1- Workshop -2- Working specification -3- A scenario of working.

Figure 8 shows a manufacturing system and its working specification. In this system, when the control system gives the order d, the actuator puts down a pallet on the conveyor. When the sensor B detects the transferred pallet (*event b*), and if the robot is not busy (*event e*), it transfers the pallet to the assembly station. The actuator comes back to its initial state and waits again d. When the robot finishes its task (*event R*), it returns to its initial state. The information concerning the interruptible tasks is given in the following table. $t.u$ is the abbreviation for *"time units"*.

Task name	Conveyor task	Robot task
$[\alpha_i,\beta_i]$ (t.u)	[3,4]	[2,3]
$[\alpha_i,\gamma_i)$ (t.u)	[3,5)	[2,4)
Used stopwatches	x_2 and y_2	x_4 and y_4
Monitoring signals	s_2 and r_2	s_4 and r_4

In Figure 9.1, we give the monitoring automaton of the considered system composed of 12 locations and focalize to a part of it in Figure 9.2. The time spaces in the locations have been calculated by using the model-checker *PHAVer* (Frehse, 2005).

Figure 8.3 shows a scenario of working where the robot and conveyor start their tasks simultaneously. This situation is represented by location L_7 as the stopwatches dynamic's show. In this scenario, the conveyor is interrupted 2 $t.u$. Then, the system fires to L_8. The inequality in bold in L_8 detects a fault in the considered behavior at the instant $x_2(\theta) = 3$. The corresponding value of y_2 is $y_2(\theta) = 1$. This result can be explained as follows: to finish the conveyor task correctly, one needs to have at least the duration $\alpha_2 - y_2(\theta) = 3 - 1 = 2\ t.u$. The corresponding value of x_2 will be $x_2 = x_2(\theta) + (\alpha_2 - y_2(\theta)) = 3 + 2 = 5$. This value exceeds the maximum permitted duration of conveyor's task. Figure 10.1 shows the monitoring

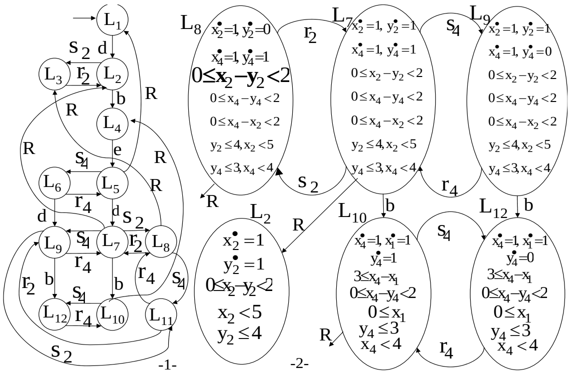

Figure 9: 1- Automaton $\mathbb{A}^*$ 2- Scoped part of $\mathbb{A}^*$.

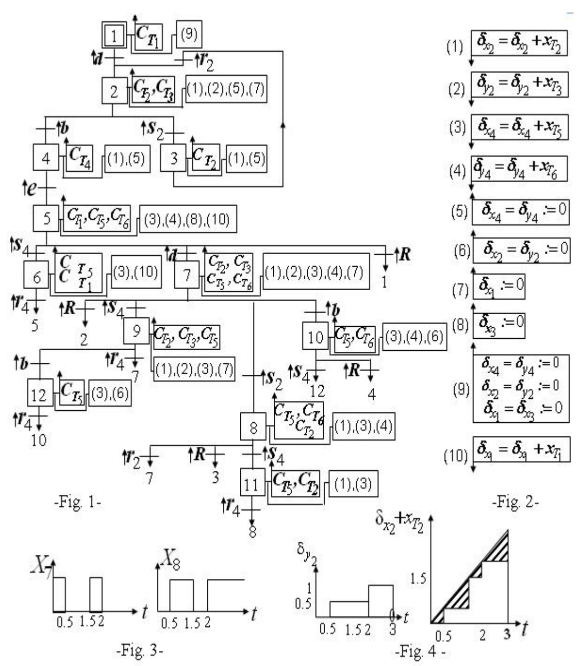

Figure 10: 1- G_1 2- Shifting and initiation actions 3- G_1 evolutions 4- evolution of Grafcet variables.

grafcet G_1 of the system. The timers T_1, T_2, T_3, T_5 and T_6 correspond respectively to stopwatches x_1, x_2, y_2, x_4 and y_4. The used shifting variables are : δ_{x_1}, δ_{x_2}, δ_{y_2}, δ_{x_4}, and δ_{y_4}. Figure 10.3 shows the evolution of G_1 according to the proposed scenario.
The receptivity of transition t_{21} in G_2 (Fig. 6.3) is: $(X_3.\overline{E_3}+X_6.\overline{E_6}+X_8.\overline{E_8}+X_9.\overline{E_9}+X_{11}.\overline{E_{11}}+X_{12}.\overline{E_{12}})$. Its predicate becomes true at the instant $t=3$ because $X_8=1$ and the inequality $(\delta_{x_2}+x_{T_2})-\delta_{y_2}\geq 2$ in $\overline{E_8}$ becomes true at this instant as shown in Figure 10.4.

5 CONCLUSION

Active approach has been carried out to provide solution to specific problem related to the fault detection which is the ability to detect the faults as early as possible. It is based on a stopwatch automaton which provides a formal support to this approach. The link between the design of monitoring system and its implementation in programmable logic controller is provided using grafcet tool. We have shown how the grafcet can be used to describe the monitoring stopwatch automaton's behavior.

REFERENCES

A.allahham and alla, H. (2006). Monitoring of timed discrete events systems: Application to manufacturing systems. In *The 32nd Annual conference of IEEE Industrial Electronics Society*.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzingerd, T., Hod, P., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algoritmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1).

Cassez, F. and Larsen, K. (2000). The impressive power of stopwatch. In *11th conference on concurrency theory*, number 1877, pages 138–152.

David, R. (1995). Grafcet: A powerful tool for specification of logic controllers. *IEEE transactions on control, systems technology*, 3(3).

Frehse, G. (2005). Phaver: Algorithmic verification of hybrid systems past hytech. In *The Fifth International Workshop on Hybrid Systems: Computation and Control*, pages 258–273.

Ghazel, M., Toguéni, A., and Bigang, M. (2005). A monitoring approach for discrete events systems based on a timed perti net model. In *Proceedings of 16th IFAC World Congress*.

Huang, Z., Chandra, V., Jiang, S., and Kumar, R. (1996). Modeling discrte event systems with faults using a rules based modeling formalism. *Mathematical Modeling of Systems*, 1(1).

S. H. ZAd, R. H. K. and Wonham, W. M. (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions On Automatic Control*, 48(7):1199–1212.