Monitoring of a Class of Timed Discrete Events systems

Adib Allahham, Hassane Alla

Monitoring of a Class of Timed Discrete Events Systems

Adib Allahham, Hassane Alla

Abstract— **This paper extends the notion of residuals for fault detection, well known in the continuous system to the timed discrete events systems. The aim is to design the fault indicators, two problems are considered. Firstly, the dynamics of clocks which monitor the system must be sensible to the behavior change resulting from a fault. Secondly, we must specify the inequality clocks constraints for which the system is normal as long as they are satisfied. We introduce the notion of acceptable system behavior modeled by stopwatch automata. This behavior is supervised by two clocks for each interruptible task of the system. The time sub-spaces in the stopwatch automaton locations delimit exactly the range of acceptable behavior. They are synthesized using an algorithm based on the reachability analysis techniques of stopwatch automata. One of the main results of designing this time space is the detection the system faults as early as possible.**

I. Introduction

Monitoring the complex systems plays an important role for economic reasons and for security and reliability motives. One of the used monitoring methods is the model-based method. The monitoring task consists in determining the occurrence of any faults (called fault detection) and to identify its type or origin (called fault diagnosis). In this paper, we consider the problem of fault detection in systems that, for this purpose, can be modeled as timed discrete-event system.

In the cadre of timed discrete-event system (TDES), the possible kinds of faults are the drastic faults, which dispossess the system its ability to perform its task, and the partial faults that result in change in timing characteristics of the system's resources. Another type of categorization of faults arises from the manner in which faults are to disappear after they occur [12]. Permanent faults which disappear due to a repair of the fault and intermitting faults. The intermitting faults can appear several times during the task execution of the system and disappear without any external action on the system. Apparition of these faults changes the dynamic of system.

The design of analytical models to fault detection in continuous system needs a central problem to be considered. One has to design fault indicators, called residuals, which are sensitive to faults. The residuals express the discrepancies between actual data (measures) and the system model, which is always modeled by both equality and inequality constraints. Equality constraints model the dynamic behavior of system and the inequality constraints express the operating range where the system have to remain inside for normal operating.

Adib Allahham, Hassane Alla are with the GIPSA-Lab., Control systems Departement. adib.al-lahham@inpg.fr, hassane.alla@inpg.fr

In this paper, we extend the notion of residuals applied in the continuous system to the TDES for design its fault indicators. To realize this aim, two problems must be considered. Firstly, the dynamics of clocks which monitor the system must reflect the behavior change resulting from a fault. Secondly, we must specify the inequality constraints for which the system is normal as long as they are satisfied.

We represent the system by a linear hybrid automaton that combines a finite state automaton with dense time. In that representation, the oriented arcs are annotated by switching conditions between the different states. Each location corresponds to a given system state: differential and algebraic equations that represent the dynamic behavior together with the acceptable state-subspace are indicated. The state-subspace in each location delimits the range of the system's variables for normal behavior in this state.

All prior works on fault monitoring in TDES consider a fault have occurred in a system if a faulty event occurs. [5], [7], reaching a faulty state [9], [10] or more generally violating system specification. The used mechanism for monitoring the system specification is based on watchdogs which detect a fault if the expected observation is produced early or late with respect to certain time bounds [11], [8]. In these methods, it is not possible to detect the occurrence of a fault immediately when it occurs because the system must wait the time bound expiration. It is desirable to detect the fault occurrence as early as possible, without waiting for the expiration of timed bounds. One of the main result of delimiting exactly the acceptable behavior in each state, is the detection of faults as early as possible, as we will see.

II. Problem Formulation

We consider the fault detection in the complex systems where their physical components can be subjected to unpredictable faults during the execution of their tasks. These faults can be either permanent or intermitting. Because of intermitting faults, 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 (case of intermitting fault) or it leaves toward a faulty state (Fig.1.a). In this paper, we introduce the notion of *acceptable behavior* which is composed of the normal and intermediate states, and consider the faults that interrupt the task of a resource. We call the system which is subjected to this type of faults as *interruptible system* and its tasks as *interruptible task*. We call also the intermitting faults which cause task's interruption as *malfunction*.

As the *FDI* (Fault Detection and Isolation [2]) in continuous systems, the fault detection system must be capable to

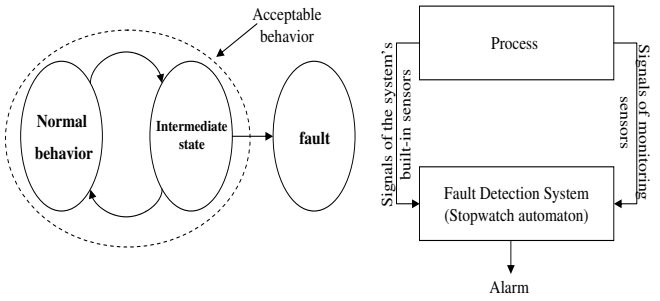

Fig. 1. a- Considered system behavior b- Structure of proposed monitoring system

follow the change of the system's dynamic resulting from the faults. More precisely: it is necessary to observe the system trajectory between normal and intermediate states. For that, we represent the system by a stopwatch automaton [1] that combines a finite state automaton with a continuous time model. In that representation, the dynamic behavior is represented by the increase rate of the stopwatches in each location. This rate can be switched between 1 and 0 to express the progression or suspension of the variable represented by stopwatch. Figure 1.b shows the structure of our detection system where the monitoring signals represent the outputs of logic sensors. It is then possible to follow the system behavior between the normal and intermediate states, for all the system tasks.

As Figure 1 shows, the monitoring system sets on the alarm when the monitored system leaves from an acceptable state to a faulty one. In other words, when the algebraic inequalities which characterize the acceptable behavior are violated. Then, we must determine the time sub-space in each location which characterizes this behavior. This leads to calculate all the trajectories corresponding to the normal behavior, and to the trajectories for which the system comes back from the intermediate state to normal state. It is necessary to notice that the presence of intermitting faults which leads to the intermediate state, can violate the inequalities constraints, since it prevents the system to execute its tasks during its specified durations. This effect corresponds to some trajectories in a stopwatch automaton SWA. Then, it is necessary to delimit only the trajectories that satisfy the property: " all the tasks are executed during its defined durations" and eliminate the others.

So, the problem of determining the time sub-state of this automaton becomes a problem of synthesis of a SWA^* which satisfy the property mentioned above (Fig.2). We can synthesize these time sub-spaces using the reachability analysis methods of Stopwatch automata, as we will see.

The rest of this paper is organized as follows: Section III defines the stopwatch automaton and some of its technique of time analysis. These techniques will be used to synthesize the time-subspace corresponding to acceptable behavior in each location. Section IV describes the behavior of an interruptible task and our method to monitor its behavior. Section V models this behavior by a stopwatch automaton. Section VI presents the method of reachability synthesis of this automaton for determining the inequalities delimiting the acceptable behavior in each location of the system. We apply this method in an illustrative example in Section VII. Finally, Section VIII concludes the paper with a summary and future works.

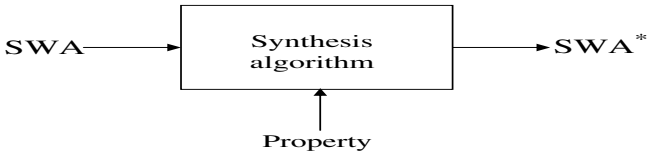

Fig. 2. Reachability synthesis of *SWA* for monitoring purposes

III. Stopwatch Automata

We basically define stopwatch automata as a class of linear hybrid automaton where the time derivative of a clock in a location can be either 0 or 1 [1].

Definition 1: A Stopwatch automaton is a 7-tuple $(L, l_0, X, \Sigma, A, I, Dif)$ where

- L is a finite set of locations,
- l_0 is the initial location,
- X is a finite set of positive real-valued stopwatches,
- Σ is a finite set of labels,
- $A \subset L \times C(X) \times \sigma \times 2^X \times L$ is a finite set of arcs. $a = (l, \delta, t, R, l') \in A$ is the arc between the locations l and l', with the guard δ, the label name t and the set of stopwatches to reset R. $C(X)$ is the set of constraints over X.
- $I \in C(X)^L$ maps an invariant to each location,
- $Dif \in (\{0,1\}^X)^L$ maps an activity to each location, $\dot{X}$ being the set of time derivatives of the stopwatches w.r.t time.$\dot{X} = Dif(l)(x)_{x \in X}$. Given a location l and a clock x, we will denote $Dif(l)(x)_{x \in X} = \{0, 1\}$. □

A. Techniques of Reachability Analysis For SWA

A state of the *SWA* is a pair (L, E) where L is a location of *SWA* and E is its time state space which is a polyhedron. As we will see, there are two types of evolutions from a state (L, E), namely the continuous evolution by letting the time progress and the discrete evolution by firing a transition. Accordingly, we define two types of successors: those by continuous evolution, which we call continuous-successors, and those by discrete evolution, which we call discrete-successors. The successor operators are basic ingredients in forward reachability analysis. For backward reachability analysis, we introduce subsequently the predecessor operators: continuous-predecessors and discrete-predecessors.

Forward Analysis: Consider an initial state (L_0,E_0^a) where E_0^a is the time space at the entry of L_0. The forward reachability method consists in starting with this initial state and computing iteratively theirs successors until the desired state (L_n,E_n) is reached. For that, the automaton follows two types of evolutions:

• remaining in the same location while the time progresses. The reachable state following this evolution is calculated by the continuous-successor $Succ_t$. For example, the state (L_0, υ'), where $\upsilon' = \upsilon + t$ and $\upsilon \in E_0^a$, is the continuous-successor of (L_0, υ), denoted by $Succ_t(L_0, \upsilon)$, if:

$$\exists t \in R^+ s.t. \forall t' \leq t, \upsilon + t' \models I(L_0) : (L_0, \upsilon') = Succ_t(L_0, \upsilon)$$
where $\upsilon' \in E_0$: the reachable space in L_0.

• firing a transition $a = (L_0, g_{0,n}, t_i, R_{0,n}, L_n) \in A$. The state (L_n, υ') is the discrete-successor of the state (L_0, υ) where $\upsilon \in E_0$ and denotes as $succ_d(L_0, \upsilon)$ if:

$$\upsilon' \models g_{0,n} \wedge (\upsilon' = \upsilon[R_{0,n}]) \wedge (\upsilon[R_{0,n}] \models I(L_n)) : (L_n, \upsilon') = succ_d(L_0, \upsilon)$$
where $\upsilon' \in E_n^a$: time space at the entery of E_n

Backward Analysis: Given the states (L_n, E_n) and (L_m, E_m). The Backward reachability starts with any state (L_n, E_n), calculates iteratively its predecessors until the desired state (L_m, E_m) is reached. For this type of analysis, the automaton follows the following evolutions:

• remaining in the same location while the time evolves. The reachable state following this evolution is calculated by the continuous-predecessor Pre_t. For example, the state (L_n, υ') is the continuous-predecessor of (L_n, υ) where$\upsilon = \upsilon' + t$, and denoted by $Pre_t(L_n, \upsilon)$, if:

$$\exists t \in R^+ s.t. \forall t' \leq t, \upsilon' + t' \models I(L_n) : (L_n, \upsilon') = Pre_t(L_n, \upsilon)$$
where $\upsilon',\ \upsilon \in E_n$: the reachable space in L_n.

• firing a transition $a = (L_m, g_{m,n}, t_i, R_{m,n}, L_n) \in A$. The state (L_m, υ') is the discrete-predecessor of the state (L_n, υ) where $\upsilon \in E_n$ and denotes as $Pre_d(L_n, \upsilon)$ if:

$$\upsilon' \models g_{m,n} \wedge (\upsilon = \upsilon'[R_{m,n}]) \wedge (\upsilon'[R_{m,n}] \models I(L_n)) : (L_m, \upsilon') = Pre_d(L_n, \upsilon')$$
where $\upsilon' \in E_m$: time space at the entery of L_m

IV. Behavior of interruptible tasks

Suppose that S a complex system which is composed of a set of tasks referred as *Task*. $Task_{int}$ represent the set of interruptible tasks in S where $Task_{int} \subset$ *Task*.

Assume that $Task_i \in Task_{int}$. It has a known execution duration $[\alpha_i, \beta_i]$ which we call *the normal duration* of $Task_i$. It is given in the technical characteristics of the resources which execute $Task_i$ or measured directly. Because of the interruptions resulting from malfunctions and for productivity motives, the designer accepts a tolerated duration for the execution of $Task_i$. It is given by the interval $[\alpha_i, \gamma_i)$ where $\beta_i < \gamma_i$. We call it: *the acceptable duration* of $Task_i$.

Hypothsis 1: The execution speed of the task $Task_i$ is supposed to be constant or it varies slightly around a mean value. This speed variation is taken into account by the interval $[\alpha_i, \beta_i]$. This interval takes also into account the predictable normal stops during the task execution. □

This type of tasks represents the most services executed by the resources in many systems such as manufacturing systems (processing, transport, filling, .etc.) which represent one of our interest domains. So, this hypothesis is realistic and necessary in our modeling point of view.

We refer to the apparition and disappearing of a malfunction or a permanent fault by its affect on the task execution, then we refer it by *Interruption* and *resuming* of the task. Considering the properties of the tasks mentioned above, we distinguish the following behavior of an interruptible task given in Figure 3. In this figure the arrows ↓ and ↑ represent respectively the signal of logical sensor which detects the interruption and resuming of $Task_i$.

- $Task_i$ is executed without interruption, therefore the execution duration belongs to the interval $t_f \in [\alpha_i, \beta_i]$.
- $Task_i$ has been executed but with several interruptions. For each interruption, the system resumes a little later from the position at which it has been interrupted. In this case: $t_f \in [\alpha_i, \gamma_i)$.

To monitor the task $Task_i$, we shall use the following timers:

- x_i measures the time since the monitored task was released for execution. In acceptable behavior, $x_i \in [\alpha_i, \gamma_i)$;
- y_i accumulates the time of partial execution of $Task_i$. When $Task_i$ finishes, y_i will belong to $[\alpha_i, \beta_i]$.

The timers x_i and y_i are reset to zero 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 achieved amount of $Task_i$. $Task_i$ is correctly executed if when the task end occurs, $y_i \in [\alpha_i, \beta_i]$ and $x_i \in [\alpha_i, \gamma_i)$.

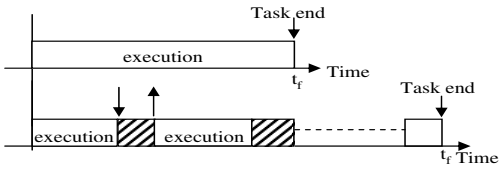

Fig. 3. Behavior of an interruptible task

V. Modeling of an interruptible task

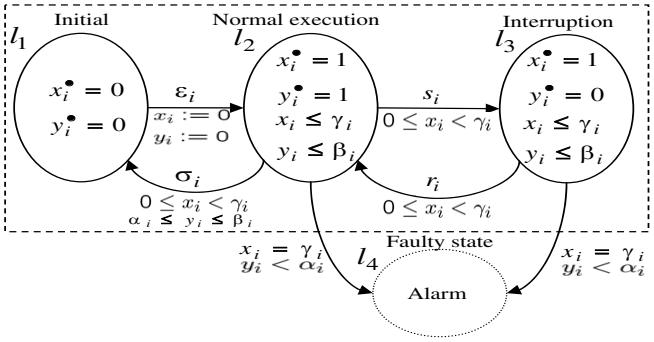

Fig. 4. Stopwatch automaton of an interruptible task

To model the acceptable behavior of $Task_i$, we need to use a Stopwatch automaton with three states such that the automaton can move back and forth between "Initial",

"normal execution" and "Interruption" as shown in Fig. 4. The state l_1 indicates that the resource is waiting to start the task, l_2 indicating that the resource is executing its task and l_3 indicating the task is interrupted after having started.

Definition 2: Let $(L, l_1, X, \Sigma_i, A, I, Dif)$ the Stopwatch automaton of task $Task_i$ where

- $L = \{l_1, l_2, l_3\}$, and l_1 is the initial location,
- $X = \{x_i, y_i\}$,
- $\Sigma_i = \{\sigma_i, s_i, r_i, \varepsilon_i\}$,
- $I(l_3) = I(l_2) = \{x_i \leq \gamma_i, y_i \leq \beta_i\}$;
- $Dif(l_2)(x_i) = Dif(l_3)(x_i) = 1$, $Dif(l_1)(x_i) = 0$ $Dif(l_2)(y_i) = 1, Dif(l_3)(y_i) = Dif(l_1)(y_i) = 0$ □

In this automaton, the interruption and resumption are modeled by transitions to and from l_3 in which the clock y_i does not progress. 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 stopwatch y_i adds up the durations of partial execution of $Task_i$ while x_i measures the total execution duration. When the stopwatches values reach $\alpha_i \leq y_i \leq \beta_i \wedge \alpha_i \leq x_i < \gamma_i$ the automaton leaves l_2 to the initial state.
In this automaton, we find the following guards:
• $l_2 \xrightarrow{g_2} l_3$: $g_2 = 0 \leq x_i < \gamma_i$. The stop can happen at any instant during the execution of the task, without exceeding the acceptable duration;
• $l_3 \xrightarrow{g_3} l_2$: $g_3 = 0 \leq x_i < \gamma_i$. The resumption of $Task_i$ must happen without exceeding the acceptable duration;
• $l_2 \xrightarrow{g_4} l_1$: $g_4 = \alpha_i \leq y_i \leq \beta_i \wedge \alpha_i \leq x_i < \gamma_i$ represents the execution of $Task_i$ in its acceptable duration.
Figure 4 shows the acceptable behavior and faulty state. The task 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 do not respect guard g_4. They are identical and given by: $g_5 = \neg g_4$, then g_5 = $(x_i = \gamma_i \wedge y_i < \alpha_i)$.

Definition 3: A possible trajectory in SWA which represents an interruptible task can be defined as:
• Initially: Start from the state $(l_1$,$(x_i = 0,\ y_i = 0))$,
• Discrete evolution: Sequence of states as $(l_1.\ l_2.\ l_3.\ l_2.\ l_1)$.
• Continuous evolution: Stopwatches evolution $(x_i,\ y_i)$.
Example, $(0,0)$,...,(α_i, α_i) or $(0,0)$,...,$(\gamma_i - \tau_i, \alpha_i)$ where τ_i a small amount of time. □

Proprerty 1: The trajectories which lead $Task_i$ to the state l_1 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 evolution in the acceptable behavior permitting to execute $Task_i$ correctly. □

The valuations of the stopwatches that satisfy the acceptable execution of task $Task_i$ given by g_4 define a polyhedron which is denoted as d_i, and called the desired space.
Note that the trajectories mentioned in Property 1 satisfy only the guard g_4. As long as $g_5 = \neg g_4$, then these trajectories do not lead to the faulty state.

VI. Characterizing the time space state of the system in its acceptable behavior

As we saw in the previous section, the task can evolve either inside the states of acceptable behavior or towards a faulty state. In this section, we concentrate on the time space synthesis that characterizes the evolution in the acceptable states for a complex system S.
The acceptable behavior of a system S is a stopwatch automaton given by Definition 1. It is destined to execute many tasks where some of them are interruptible. The behavior of these tasks are described in Definition 2.

Definition 4: Let $\mathbb{A} = (L, l_0, X, \Sigma, A, I, Dif)$ and $\mathbb{A}^* = (L', l_0, X, \Sigma, A, I, Dif)$ be two identical stopwatch automata. In these automata, the corresponding locations have the same invariants, the same stopwatches dynamics, and the corresponding arcs have the same guards and the same set of initialized stopwatches. Let E_i and E_i^* be the reachable time space in two corresponding locations where $l_i \subset L$ and $l_i' \subset L'$ respectively. We say that $\mathbb{A}^*$ is more restrictive (in terms of behavior) than $\mathbb{A}$, denoted by $\mathbb{A}^* \preceq \mathbb{A}$, if $E_i^* \subseteq E_i$. □

problem 1: Reachability synthesis for monitoring purposes. Let $\mathbb{A} = (L, l_0, X, \Sigma, A, Inv, Dif)$ be a stopwatch automaton and Q be a subset of trajectories $L \times X$ which represents a natural generalization of Property 1 for all the interruptible tasks of the system S represented by $\mathbb{A}$. The reachability synthesis problem is to find the automaton $\mathbb{A}^* \preceq \mathbb{A}$ such that for each trajectory q in $\mathbb{A}^* \Rightarrow q \in \mathsf{Q}$. □

We make use the following algorithm to solve this problem.

A. Synthesis Algorithm

Our algorithm for constructing the restrictive automaton $\mathbb{A}^*$, involves two phases. Firstly, we calculate by using the forward analysis, the reachable time space of $\mathbb{A}$ which corresponds to the set of all possible trajectories including those leading to faulty state. In the second phase, a procedure based on backward analysis is given to synthesize from the previous set of trajectories, only those satisfying Property 1, for each interruptible task in S.
We will introduce the used notations in the algorithm:
• E_i Time reachable space in a location i.
• E_i^a Time space at the entry of L_i for all the visits.
• $e_{i,j}^a$ Time reachable space at the entry of L_j by firing a_{ij}.
• P_1 The memory that saves the location's name and the space at its entry.
• E_j^F The space in L_j resulting from Forward analysis.
• D = $d_1, ..., d_m$ The set of time desired space. $\forall\ task_i \in task_{int}$: $d_i \in D$. m number of interruptible tasks in S.
• P_2 The memory that saves the location's name that have a desired space at its outgoing arc. An element of P_2 has the following form: $\{(L_i, d_i) \in P_2 : L_i \in L\ d_i \in D\ \}$
• P_3 The memory that saves the arcs to be analyzed in backward analysis. An element of P_3 has the following form: $(L_i, a_{i,j}, L_j)$ where L_i location source, L_j location destination and $a_{i,j}$ the arc to be analyzed.

• $E_j^{(i)}$ The space in L_j calculated by backward analysis at iteration i.
• $Project_{L_j}(E)$ function which gives the projection of the space E on the active and suspended stopwatches in L_j.
We construct the space of $\mathbb{A}^*$ by using the following algorithm:

• **Forward Analysis**

1) Initialize P_1: $P_1 = \{(L_0, e_0^a)\}$,
   E_0 =....= E_i =....=E_m = $\emptyset$
   E_0^a =....= E_i^a =....=E_m^a = $\emptyset$.
2) Pick an element from P_1. Let be (L_i, e_i^a).
3) Calculate the reachable space in L_i:
   $E_i = Succ_t(e_i^a) \cup E_i$
4) For each outgoing arc from L_i, do:
   $S_{i,j} = Succ_d(E_i) : L_j$ the destination location
   $e_{ij}^a = Project_{L_j}(S_{i,j})$
   if $e_{ij}^a \nsubseteq E_j^a$:$E_j^a = E_j^a \cup e_{i,j}^a$
   add $(L_j, e_{i,j}^a)$ to P_1.
   If $P_1 \neq \emptyset$, go to step (1).

• **Backward Analysis**

1) Initialize $E_0^{(0)} = E_0^F$, ...=$E_m^{(0)} = E_m^F$,
   P_2 = { (L_i, d_n),...,(L_j, d_k),...,(L_n, d_m)}, $P_3 = \{\emptyset\}$.
2) Pick an element from P_2. Let be (L_j, d_k).
3) Calculate the space in L_j which allows to reach d_k:
   $E_j^{(i)} = Pre_t(d_k)$.
   if $E_j^{(i-1)} \subseteq E_j^{(i)}$ go to step (2).
   otherwise $E_j^{(i)} = E_j^{(i)} \cap E_j^{(i-1)}$.
4) Add to P_3 all the entering arcs to be analyzed. Let for L_j be $P_3 = \{(L_n, a_{n,j}, L_j), (L_l, a_{l,j}, L_j)\}$
   4.1) Pick the last element in P_3. Let be $(L_l, a_{l,j}, L_j)$
   4.2) Calculate the space at entry of L_j permitting to reach $E_j^{(i)}$: $E_j^a = E_j^a \cap E_j^{(i)}$
   4.3) Calculate the space in L_l permitting to reach E_j^a by firing $a_{l,j}$: $E_l^{(i)} = Pre_d(E_j^a)$
   if $E_l^{(i-1)} \nsubseteq E_l^{(i)}$: $E_l^{(i)} = E_l^{(i)} \cap E_l^{(i-1)}$ go to step (4).
   If $P_3 \neq \emptyset$ go to step (4.1).
   If $P_2 \neq \emptyset$, go to step (2).

Using this algorithm for an interruptible task mentioned in Figure 4, we have found the reachable time space in l_2 and l_3 by a forward analysis (Fig. 5.a). In these spaces, we can see that there are some trajectories that do not permit to reach the desired space d_i shown in (Fig. 5.b). These trajectories are eliminated by the backward analysis step. The resulting space (Fig. 5.c) represents all the possible evolution permitting to execute task $Task_i$ according to the guard g_4. We call this resulting space: *exact space of* $Task_i$.

Remark 1: Robustness of the exact space
Some partial faults that affect the timing characteristics may occur in the execution of the resources the $Task_i$. These faults can also be detected by the violation of the exact space.

B. Termination Criterion

Unlike timed automata, the reachability problem for Stopwatch automata is undecidable [6]. This is because the values of stopwatches in a location of automaton is described by non-convex polyhedron. Fortunately, thanks to the tools of the hybrid systems as model-checker *PHAVer* [3], the manipulation of these polyhedron is possible.
In *PHAVer*, if the invariants of the automaton is bounded, then the analysis termination can be forced by using simplification techniques of a complex polyhedra. This is because there are only a finite number of possible predicates. The simplification is done in a strictly conservative fashion of the reachable set of states [4].

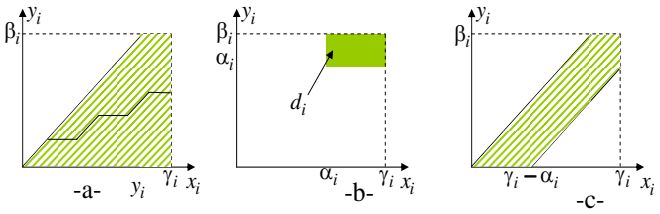

Fig. 5. Time state space of an interruptible task in l_2 and l_3: (a) reachable space and one of possible trajectory, (b) desired, (c) exact space

VII. Illustrative Example

To illustrate the way to design a monitoring system, we consider the following example. A part of manufacturing system is made up of a transfer station and a robot. The transfer station is composed of an actuator and a conveyor. This system and its working specification are shown in Figure 6. When the control system gives the order d, the

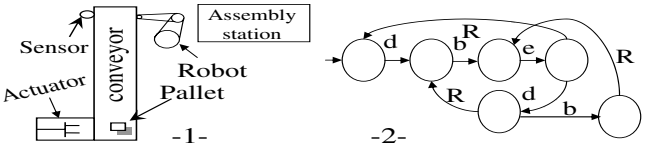

Fig. 6. -1- The manufacturing system -2- Working specification

actuator puts down a pallet on the conveyor. When the sensor B detects the transferred pallet $(event\ b)$, the transfer station comes back to its initial state, and the robot, if it is not busy $(event\ e)$, transfers the pallet to the assembly station. When the robot finishes its task $(event\ R)$, it returns to its initial state. The necessary information for building up the monitoring system is given in the following table. $t.u$ is the abbreviation for *"time units"*.

Task name	Conveyor task	Robot task
α_i (t.u)	3	2
β_i (t.u)	4	3
γ_i (t.u)	5	4
Used stopwatches	x_2 and y_2	x_4 and y_4
Event of task end	b	R
Interruption signal	s_2	s_4
Resuming signal	r_2	r_4

In Figure 7.1, we give the monitoring automaton of the considered system composed of 12 locations and focalize to a part of this automaton in Figure 7.2 for simplicity reasons.

In this figure, the event d comes from the controller and gives to the conveyor the starting order. It is monitored by clock x_1. Note that the automaton state is updated only by the set of events coming either from the controller or from the process permanently.

The time spaces in the locations L_2, L_7, L_8, L_9, L_{10} and

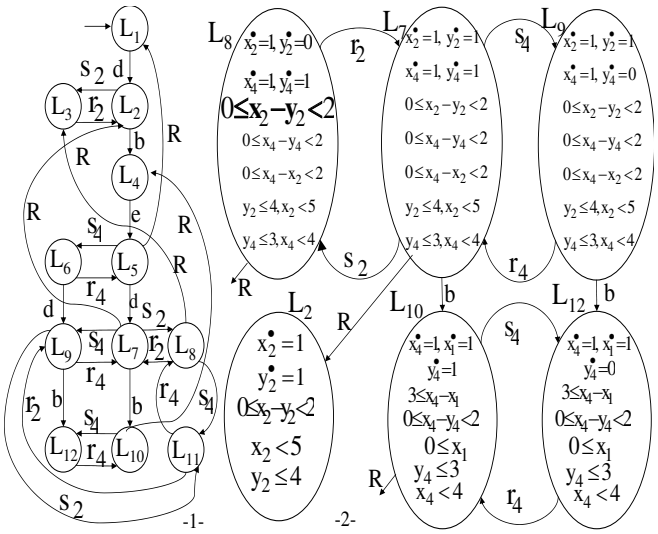

Fig. 7. 1- Automaton $\mathbb{SWA}^*$ 2- Scoped part of $\mathbb{SWA}^*$

L_{12} are represented by the algebraic inequalities and have been calculated by applying the proposed algorithm and by using the model-checker *PHAVer* [3]. It provides commands for computing reachable (Forward and backward) states and simulation relations, plus a number of commands for the manipulation and output of data structures.

Figure 8 shows a scenario of working where the robot and

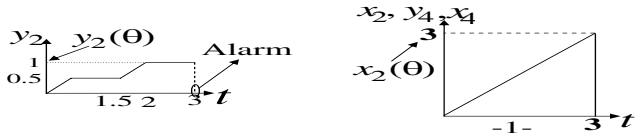

Fig. 8. Scenario of working

conveyor start their tasks simultaneously. This situation is presented 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 the fault presented in Figure 8 at instant $x_2(\theta) = 3$ $t.u$. The corresponding value of y_2 is $y_2(\theta) = 1$ $t.u$. We can explain the setting on the alarm by following reason: To finish the conveyor task correctly, one needs to have at least the duration $\alpha_2 - y_2(\theta) = 3 - 1 = 2$ $t.u$. In this case, the corresponding value of x_2 is $x_2 = x_2(\theta) + (\alpha_2 - y_2(\theta)) = 3 + 2 = 5$ $t.u$. This execution duration will exceed the maximum permitted duration of conveyor's task $\gamma_2 - \varepsilon$ where ε is an amount of time infinity small. So, one detects this fault as early as possible at instant 3 $t.u$. As long as using the other method based on watchdog mechanism will detect this fault at instant 5 $t.u$.

VIII. Conclusion

In this paper, we have proposed a method for the real-time monitoring systems, based on stopwatch automata. It takes into account the behavior of some tasks called '*interruptible tasks*'. The locations of this automaton represent the system reachable state in this behavior. The state of monitoring system is updated only by the set of events coming either from the controller or from the process. The acceptable behavior of each interruptible task is supervised by using two clocks. The time sub-state space in each location of this automaton represents the temporal constraints that the system must respect. These time sub-state space are represented by a set of algebraic inequalities. The monitoring system detects any violation of these constraints. Consequently, it detects the system faults as early as possible. The time state space for a complex system are calculated by an algorithmic method based on forward and backward reachability analysis techniques of stopwatch automata.

The future works will take into account more complex modification of the dynamic of the task resulting from faults. These faults will lead the system to multi mode behavior (instead of two: "execution" and "interruption"). For example, we will consider the faults that can accelerate or slow down a resource during its task execution. For that, one needs to use the linear hybrid automata where the time derivative of the variable in each location can be $x^{\cdot} = c$ where $c \in \mathbb{R}$.

References

[1] F. Cassez and K.J. Larsen. The impressive power of stopwatch. Number 1877, pages 138–152. Lecture Notes in Computer Science, Springer-Verlag, 2000.

[2] J. Chen and R.J. Patton. *Robust model-based fault diagnosis for dynamic systems*. Kluwer Academic Publishers, 1999.

[3] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Proceedings of the Fifth International Workshop on Hybrid Systems: Computation and Control*, pages 258–273, 2005.

[4] G. Frehse, B.H. Krogh, and R.A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. volume 1, 2006.

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

[6] T. A. Henzinger, P. W. Kopke, A. Puri, and V. Varaiya. What's decidable about hybrid automata? *Journal of Computer Sciences*, 57, 1998.

[7] S. Lafortune K. Sinnamohideen M. Sampath, R. Senqupta and C. Teneketzis. Failure diagnosis using discrete- event models. *IEEE Transaction on Control Systems Technology*, 4(2):105–124, March 1996.

[8] D. N. Pandalai and L. E. Holloway. Template languages for fault monitoring of timed discrete event processes. *IEEE Transactions On Automatic Control*, 45(5), May 2000.

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

[10] R. Kumar S. Jiang and H. E. Garcia. Diagnosis of repeated/intermittent failure in discrete event systems. *IEEE Transaction On Robotic and Automation*, 19(2):310–323, 2003.

[11] V.S. Srinivasan and M.A. Jafari. Fault detection/monitoring using time petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 23:1155–1162, 1993.

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