Active Fault-Tolerant Control of Timed Automata with Guards

Julien Niguez, Said Amari, Jean-Marc Faure

Active Fault-Tolerant Control of Timed Automata with Guards

Julien Niguez, Saïd Amari, Jean-Marc Faure

LURPA, ENS Paris-Saclay, Univ. Paris-Sud, Université Paris 13, Sorbonne Paris Cité, Supmeca, Université Paris-Saclay, 94235, Cachan, France.
{julien.niguez ; said.amari ; jean-marc.faure}@ens-paris-saclay.fr

Abstract: In this paper, an approach for active fault-tolerant control of discrete event systems modeled by timed automata with guards is proposed. Time is essential to detect some faults, and will be used as a criterion to select the control law. A model representing the behavior of the whole system that respects time constraints is first constructed. Hence, given a diagnosis result, a reconfigured control law is extracted from the previous model on the basis of the fastest execution time of desired tasks.

Keywords: Timed Discrete Event Systems, Timed Automata with Guards, Fault-Tolerant Control, Reconfiguration

1. INTRODUCTION

Availability of industrial processes within a company is a constant concern, with significant economic implications. It depends among others on the ability of the systems to adapt to faults before they can have a negative impact on production. Fault-Tolerant Control (FTC) is a means of dependability that allows the interaction with the system controller, in order to adapt the control to a faulty behavior of the plant. The production strategy can be accommodated before the productivity of the system is reduced. Basic definitions of FTC are presented in (Blanke et al. 2016).

Concerning FTC of Discrete Event Systems (DES), the different methods can be separated in two categories.

Passive FTC approaches generally consist of a single controller model that can be used for both nominal and faulty behavior. In (Seong-Jin Park and Jong-Tae Lim 1999), the controller is designed to respect the nominal specification with and without the occurrence of a fault. Some approach allows degraded modes of operation (Wen et al. 2008) (Wittmann, Richter, and Moor 2012). An extension of the latter introduces a module that hides the fault to the controller (Wittmann, Richter, and Moor 2013).

On the other hand, active FTC methods use several models of the controller that can be switched. In (Shu and Lin 2014), the controller model is selected in a bank of precomputed models according to the diagnosis result, while in (Paoli, Sartini, and Lafortune 2011), only the current state of the controller is adapted. Recently, approaches based on tracking controller reconfiguration were proposed, for both unambiguous (Schuh and Lunze 2016b) and ambiguous diagnosis (Schuh and Lunze 2016a).

In a previous work (Niguez, Amari, and Faure 2015), it has been shown that passive approaches require explicit models of faults, which is not feasible for an industrial application. Furthermore, there is no method for FTC of DES taking the physical time into account. This is particularly limiting as it is not possible to treat faults that are only detectable thanks to the measurement of time, and which results in most cases in a system failure.

This paper proposes a method for active FTC of DES modeled by timed automata with guards. This formalism has been selected because it allows to represent execution date of an event with an interval. This represents the fact that in practice, an event does not occur at the exact same time, and a task does not have an exact duration. Fig. 1 details the architecture of the system considered. A faulty plant $\mathcal{P}$ is controlled by a controller $\mathcal{C}$ through controllable events e_c, and reacts by generating uncontrollable events e_u. The diagnoser $\mathcal{D}$ is in charge of detecting the occurrence of a fault and to compute a diagnosis result. This result is sent to the reconfiguration block that consists of two units. The reconfiguration model $\mathcal{G}(R)$ can be seen as a database of acceptable behaviors. The reconfigurator $\mathcal{R}$ must select and extract a reconfigured control law from the reconfiguration model based on the diagnosis result. Then this new control law is sent to the controller $\mathcal{C}$ in order to accommodate the fault f.

The main contribution of this paper is the construction method of the reconfiguration block. For this reason, it has been chosen to use an existing solution for the diagnoser. Since time was a major criterion, the diagnoser proposed in (Schneider, Litz, and Danancher 2011) was selected.

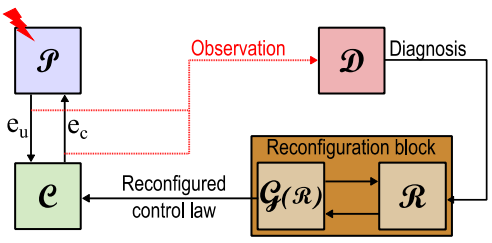

Fig. 1 – A fault-tolerant control loop

The paper is organized as follows: section 2 details the formalism of timed automata with guards and the hypotheses

of this work. In section 3 the construction of the reconfiguration model $\mathcal{G}(R)$ is detailed. Section 4 exposes the different cases of reconfiguration. Finally, an example of application on a sorting case is provided in section 5.

2. PRELIMINARIES

2.1. Timed automata with guards

Definition 1 (Cassandras and Lafortune 2008)**:** a *timed automaton with guards*, denoted by G, is a 6-tuple $G = (Q, \Sigma, Q_0, Q_m, Tra, C)$ where:

- Q is the set of states;
- $Q_0 \in Q$ is the initial state;
- $Q_m \subset Q$ is the set of final (or marked) states;
- Σ is a finite set of events;
- C is the set of clocks, $c_1, \dots, c_n$, with $c_i(t) \in \mathbb{R}^+$, $t \in \mathbb{R}^+$;
- Tra is the set of *timed transitions* of the automaton with $Tra \subseteq Q \times \mathcal{C}(C) \times \Sigma \times 2^C \times Q$ where $\mathcal{C}(C)$ is the set of admissible constraints for the clocks in the set C.

The set Tra of *timed transitions* is to be interpreted as follows: if $(q_{in}, guard, e, reset, q_{out}) \in Tra$, then there is a transition from q_{in} to q_{out} with the complete label $(e, guard, reset)$ where $guard \in \mathcal{C}(C)$, $e \in \Sigma$ and $reset \subseteq C$.

The set of admissible clock constraints $\mathcal{C}(C)$ is specified as follows:

- If $\boldsymbol{I} \subseteq \mathbb{R}^+$, then all conditions of the form $c_i(t) \in \boldsymbol{I}$ are in $\mathcal{C}(C)$;
- If g_1 and g_2 belong to $\mathcal{C}(C)$, then $g_1 \wedge g_2$ belongs to $\mathcal{C}(C)$;

Remarks:

- There is no need for the bounds of admissible clock constraints to be integer.
- All clocks are set to 0 when the system is initialized.
- $reset$ corresponds to the subset of clocks that will be reset when the transition is fired. This mechanism allows modeling systems in which duration is stated for sequences of events.

An example of graphical representation of Timed Automata with Guards (TAG) is presented in part 2.3.

The determinism of *timed automata with guards* can be defined in two ways:

- *Time-determinism:* an automaton is deterministic if for all events in all states, the guards of the outgoing transitions are mutually exclusive.
- *Event-determinism:* an automaton is deterministic if for all states, there is at most one outgoing transition triggered with the same event.

It can be denoted that any *event-determinist* TAG is also *time-determinist*.

2.2. Hypotheses

Several hypotheses and limitations can be stated:

- For small systems, only one clock is generally sufficient to operate the system. Concerning larger systems, they can be handled by using decentralized approaches, in which each sub-system is modeled with a single clock. In that specific case of single clock systems, the parallel composition could be simplified since the conjunction of two guards would become equivalent to the intersection of the intervals. If the result of that intersection is the empty set, then the guard can never be validated, and the associated transition can be deleted.
- All models will be *event-deterministic*.
- The repartition of occurrence dates of an event in a given interval will be modeled with a normal distribution.

2.3. Graphical representation and notations

Fig. 2 depicts an example of a system modeled with a TAG, called $\mathcal{G}(P_1)$. It consists of two processes $\mathcal{A}$ and $\mathcal{B}$. Each process can be started with controllable events S_a and S_b, respectively followed by the sequences of uncontrollable events ac and bc. The objective of the system is achieved when the event c is generated. Both processes end with the occurrence of the event c, which means that process $\mathcal{B}$ can be seen as a redundancy of the process $\mathcal{A}$. The system can be restarted with the controllable event R. State P_0 is considered initial (shown with an incoming arrow). State P_5 is considered as final (shown with an outgoing arrow). Each time the clock is reset, it is stated in the transition with the element r (for example, in the transition $R, \mathbb{c}(t), r$ from state P_5 to state P_0). Otherwise, it is indicated with $-$. In this example, every uncontrollable event is expected to occur before an upper bound m time units (t.u.), while every controllable event is considered as occurring instantly at the current clock value $c(t)$ when entering a new state (the interval of these transitions should be $[c(t); c(t)]$). However, for the sake of clarity, the notation $\mathbb{c}(t)$ is used instead of $[c(t); c(t)]$ in the graphical representations of TAGs.

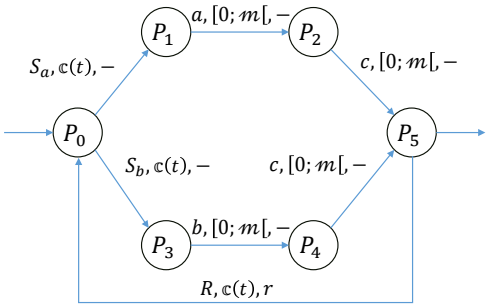

Fig. 2 – Timed automaton with guards $\mathcal{G}(P_1)$

Controllable (resp. uncontrollable) events are represented by uppercase (resp. lowercase) letters.

2.4. Composition of timed automata with guards

Definition 2 (Cassandras and Lafortune 2008)**:** consider two timed automata with guards:

$$G_1 = (Q_1, \Sigma_1, Q_{0,1}, Q_{m,1}, Tra_1, C_1)$$
$$G_2 = (Q_2, \Sigma_2, Q_{0,2}, Q_{m,2}, Tra_2, C_2)$$

The *parallel composition* of G_1 and G_2 is the automaton $G_{1||2} = \boldsymbol{Ac}(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, Q_{0,1} \times Q_{0,2}, Q_{m,1} \times Q_{m,2}, Tra_{1||2}, C_1 \cup C_2)$

where $\mathcal{Ac}$ corresponds to accessible states, $Tra_{1||2}$ is defined as follows:

$$Tra_{1||2} \subseteq (Q_1 \times Q_2) \times \mathcal{C}(C)_{1||2} \times (\Sigma_1 \cup \Sigma_2) \times 2^{C_1 \cup C_2} \times (Q_1 \times Q_2)$$

- For all $e \in \Sigma_1 \cap \Sigma_2$, if

$(q_{i,in}, guard_i, e, reset_i, q_{i,out}) \in Tra_i$ for $i = 1,2$, then

$$\left((q_{1,in}, q_{2,in}), guard_1 \wedge guard_2, e, reset_1 \cup reset_2, (q_{1,out}, q_{2,out})\right) \in Tra_{1||2}$$

- For all $e_1 \in \Sigma_1 \backslash \Sigma_2$ and $q_2 \in Q_2$ if

$(q_{1,in}, guard_1, e_1, reset_1, q_{1,out}) \in Tra_1$, then

$$\left((q_{1,in}, q_2), guard_1, e_1, reset_1, (q_{1,out}, q_2)\right) \in Tra_{1||2}$$

- For all $e_2 \in \Sigma_2 \backslash \Sigma_1$ and $q_1 \in Q_1$ if

$(q_{2,in}, guard_2, e_2, reset_2, q_{2,out}) \in Tra_2$, then

$$\left((q_1, q_{2,in}), guard_2, e_2, reset_2, (q_1, q_{2,out})\right) \in Tra_{1||2}$$

3. CONSTRUCTION OF THE RECONFIGURATION MODEL

The objective of this part is to provide a construction method of the reconfiguration model $\mathcal{G}(R)$ of Fig. 1. This step of the approach must be done *offline*.

3.1. Problem statement

The main idea is to construct a reconfiguration model of the system that describes the entire behavior complying with a set of timed rules. Two kinds of models can be used to obtain this result: *Plant models* and *Specification models.* These models are then composed in order to obtain the reconfiguration model. Every succession of states that leads from the initial state to the final state correspond to a sequence of operations that meets the time constraints and performs the expected tasks.

3.2. Plant models

Plant models are used to represent the components of the system. They correspond to their logical behavior, without taking time constraints into account. The TAG $\mathcal{G}(P_1)$ of Fig. 2 can be considered as a *plant model.* We will consider that:

- Controllable events are generated as soon as they are expected, which correspond to the current clock value when entering a new state. This is represented by the interval $\mathbb{c}(t)$ in the associated transitions.
- Uncontrollable events are expected to occur between 0 and m t.u., with m an unknown upper bound. The corresponding interval depicts the fact that the date of the occurrence of the event is not constant. This is represented by the interval $[0; m[$ in the associated transition.

The TAG of Fig. 2 depicts the two sequences of events that include the event c from the initial state.

3.3. Specification models

Specification models are graphical representations of the timed rules that the system must satisfy to operate in its nominal conditions. They are used to specify the intervals of the transitions associated with uncontrollable events.

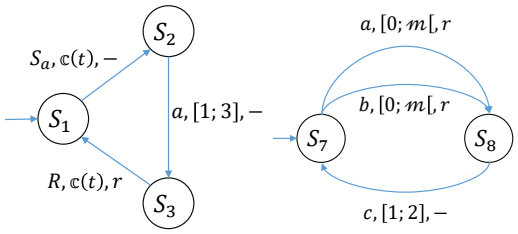

Fig. 3 - Two models of timed specifications – From left to right: $\mathcal{G}(S_1)$ and $\mathcal{G}(S_3)$

Fig. 3 shows two specifications that ensure timed rules on the system of Fig. 2. $\mathcal{G}(S_1)$ states that the event a must occur between 1 and 3 t.u. after the occurrence of the event S_a, and that the system is reinitialized through event R before any other cycle of process $\mathcal{A}$. It can be noted that it is not necessary to reset the clock on the occurrence of S_a since S_a is supposed to occur instantly when the transition from S1 to S2 is fired. $\mathcal{G}(S_3)$ describes the fact that c must occur between 1 and 2 t.u. after the occurrence of either a or b. The specification $\mathcal{G}(S_2)$ (not presented here) is similar to $\mathcal{G}(S_1)$ in that it ensures that b occurs between 2 and 5 t.u. after S_b. For specifications of Fig. 3, all states are considered as final, but the outgoing arrows were deleted for the sake of readability.

3.4. Reconfiguration model

Given the *plant* and *specification models* determined as explained above, the following algorithm is proposed to compute the *reconfiguration model.*

Algorithm 1: Construction of the reconfiguration model

Given: plant models $\mathcal{G}(P_i)$, specification models $\mathcal{G}(S_j)$

Compute the reconfiguration model $\mathcal{G}(R)$ of the system, defined by:

$$\mathcal{G}(R) = \mathcal{G}(\mathcal{P}) || \mathcal{G}(\mathcal{S})$$

with $\mathcal{G}(\mathcal{P}) = ||_i \mathcal{G}(\mathcal{P}_i)$ and $\mathcal{G}(\mathcal{S}) = ||_j \mathcal{G}(\mathcal{S}_j)$

Result: reconfiguration model $\mathcal{G}(R)$

If there is no final state in $\mathcal{G}(R)$, this means that the specifications are too restrictive. One or more restrictions must be relaxed in order for the system to perform its expected behavior. The TAG of Fig. 4 presents the reconfiguration model obtained by composition of $\mathcal{G}(P_1)$, $\mathcal{G}(S_1)$, $\mathcal{G}(S_2)$ and $\mathcal{G}(S_3)$, and represents all the evolutions of the components that respect the time constraints of the specification. Both states sequences $R_0R_1R_2R_5$ and $R_0R_3R_4R_5$ lead from the initial state R_0 to the final state R_5. However, it can be denoted that the first sequence is on average faster to execute than the second one for a normal distribution of occurrence dates (resp. 3,5 t.u. and 5 t.u.).

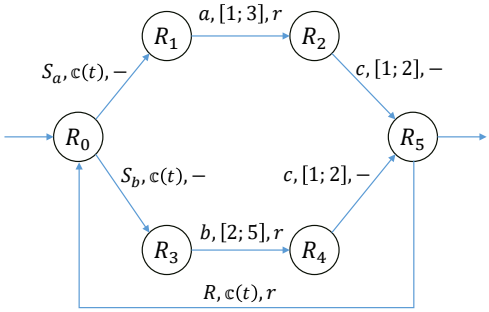

Fig. 4 - Reconfiguration model $\mathcal{G}(R)$

4. RECONFIGURATION OF THE CONTROLLER

The objective of this part is to detail the method of reconfiguration of the controller given the reconfiguration model and the diagnosis result, which will be performed by the reconfigurator unit $\mathcal{R}$ of Fig. 1. Since the reconfiguration step is done accordingly to the diagnosis result, it must be only done *online*.

4.1. Reconfiguration cases

For the nominal behavior, the control law is directly extracted from the reconfiguration model by selecting the fastest-execution-time path on the average from the initial state to the final state. In the example Fig. 4, it corresponds to the sequence of states $R_0R_1R_2R_5$, with an average execution time of 3,5 t.u.

Concerning the behavior in case of a fault, we will distinguish cases based on the three types of diagnosis results considered (Schneider, Litz, and Danancher 2011):

4.1.1. *Residual: $\{e\}$ – event e was expected but did not occur*

Algorithm 2: Reconfiguration for $\{e\}$

Given: reconfiguration model $\mathcal{G}(R)$, diagnosis residual $\{e\}$

I. Delete all the transitions labeled with the event e.
II. Extract the fastest-execution-time path if there exists one going from the initial state to the final state. This path corresponds to the control law $\mathcal{G}(\mathcal{C})$

Result: control law $\mathcal{G}(\mathcal{C})$

If it is not possible to reach a final state after step 1, it means that it is not possible to reconfigure the system. In practice, this means that the system possesses no redundancy for the component associated to the faulty event. Let us consider the diagnosis residual $\{a\}$. In Fig. 4, the transition from R_1 to R_2 must be deleted. However, it is still possible to reach the final state through the sequence of events $R_0R_3R_4R_5$. The sub-model extracted from this sequence corresponds to the reconfigured control law $\mathcal{G}(\mathcal{C})$, with an average execution time of 5 t.u.

4.1.2. *Residual: $\{e\}, late/early$ – event e was expected but occurred too late/early*

Algorithm 3: Reconfiguration for $\{e\}, late$ or $\{e\}, early$

Given: reconfiguration model $\mathcal{G}(R)$, diagnosis residual $\{e\}, late$ or $\{e\}, early$

I. Modify the bounds of all the transitions labeled with the event e so that the date of occurrence of the event is no longer out of bounds.
II. Extract the fastest-execution-time path if there exists one going from the initial state to the final state. This path corresponds to the control law $\mathcal{G}(\mathcal{C})$

Result: control law $\mathcal{G}(\mathcal{C})$

Let us consider the diagnosis residual $\{a\}, late$, with a date of execution of 5. In order to compute the new control law, the transition from R_1 to R_2 is modified to $a, [1; 5], r$. The average execution-time of the sequence $R_0R_1R_2R_5$ become 4,5 t.u., which is still faster than the average execution-time of the sequence $R_0R_3R_4R_5$. Hence, the reconfigured control law can be extracted from states $R_0R_1R_2R_5$.

4.2. Case of ambiguous diagnosis

The case of ambiguous diagnosis corresponds to the situation when the diagnoser proposes a set of faulty events instead of single one. It is possible to treat this case by successively applying step 1 of algorithms 2 and 3 for each possibly faulty event and then apply step 2.

4.3. Case of multiple final states

It is necessary to distinguish two cases:

- All final states have the same signification for the system (e.g. two processes that product the same pieces. One of the processes can be seen as a redundancy).

In this case, it is sufficient to find a path from the initial state to any of the final states, since they all share the same physical meaning.

- Final states have different meanings for the system (e.g. a machine producing pieces depending on the input raw piece)

It is necessary to compute a sub-control law for every set of final states that holds a different signification. Hence, the control law is obtained by composition of all the sub-control laws. An example of this kind of system is treated in section 5.

5. APPLICATION: SORTING SYSTEM

In this section, the reconfiguration method is applied for illustration purpose. The example used for this application is a turntable from a sorting system (Fig. 5), whose purpose is to separate packages of two different sizes arriving from conveyer B, small packages sent to the right, large packages sent to the left. This system is inspired by the one which is proposed in the ITS PLC software and has been modified to highlight the interest of the method with the addition of a second controller to rotate the table.

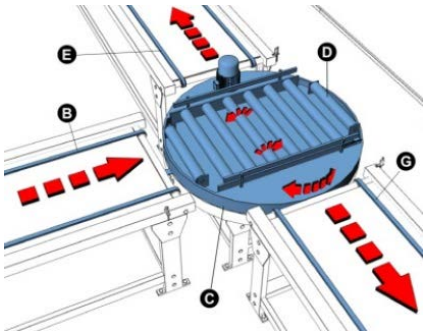

Fig. 5 – Turntable

5.1. Presentation of the system

The turntable is composed of a table (C) and a set of rollers (D) which can both rotate in two directions. This specificity allows to distribute the packages on each side in two different ways, so the system can be reconfigured in case of faults. Conveyors B, E and G are not considered in this paper.

The table below lists all the events that are used to model the system.

Table 1 - Event of the turntable model

Label	C/U	Description
$T_+, \overline{T_+}$	C	Clockwise rotation of the table
$T_-, \overline{T_-}$	C	Counterclockwise rotation of the table
$R_+, \overline{R_+}$	C	Loading rotation of the rollers
$R_-, \overline{R_-}$	C	Unloading rotation of the rollers
p_s	U	Detection of a small package
p_b	U	Detection of a large package
p_c	U	Detection of a package on the table
p_r	U	Detection of a package on the right conveyor
p_l	U	Detection of a package on the left conveyor
t_r	U	Table facing the right conveyor
t_m	U	Table in initial position
t_l	U	Table facing the left conveyor

In the case of controllable events, E (resp. $\bar{E}$) means that the actuator is set to 1 (resp. 0).

5.2. Construction of the reconfiguration model of the turntable

For simplicity reasons, only the reconfiguration model of the turntable will be presented in Fig. 6. It was built using two plant models (one for the table and one for the rollers) and three specification models (one ensuring that large (small) packages are delivered to the left (right), one for the delay of loading/unloading of the rollers and another one for the delay of the table rotations).

The reconfiguration model of the turntable has several specificities. The upper part of the graph (all states from R_1 to R_{14}) corresponds to the treatment of a large package, while the lower part (all states from R_{15} to R_{28}) corresponds to the small package. Sequences of states $R_5R_6R_7R_8R_9$ and $R_{10}R_{11}R_{12}R_{13}R_{14}$ (resp. $R_{19}R_{20}R_{21}R_{22}R_{23}$ and $R_{24}R_{25}R_{26}R_{27}R_{28}$) describe the only two sequences of events that ensure the distribution of a large package (resp. a small package) to the left (resp. right): T_+ followed by R_- or T_- followed by R_+ (resp. T_+ followed by R_+ or T_- followed by R_-).

States R_9, R_{14}, R_{23} and R_{28} are final. However, R_9 and R_{14} mean that a small package has been successfully transferred to the right, while R_{23} and R_{28} have the same meaning for a large package delivered to the left. Hence, for the step of control law selection, it is necessary to keep one of the states R_9 and R_{14} and one of the states R_{23} and R_{28}, as well as the sequences leading to these states.

5.3. Reconfiguration scenarios

In this part, the selection of the control law for the controller will be detailed in different cases of reconfiguration.

5.3.1. *First scenario: faultless case*

In the case where no fault has occurred, it is possible to extract the control law directly from the reconfiguration model. Since the execution time is not a discriminant criterion here, the control law can be obtained arbitrarily as long as it contains exactly one of the states R_9 and R_{14} and one of the states R_{23} and R_{28}. A possible solution for the control law can be obtained from the reconfiguration model of Fig. 6 without states R_{10}, R_{11}, R_{12}, R_{13}, R_{14}, R_{24}, R_{25}, R_{26}, R_{27}, R_{28}, R_{32}, R_{33} and R_{34}.

5.3.2. *Second scenario: faulty case 1*

In this case, the actuator allowing the counterclockwise rotation of the rollers cannot be activated. The corresponding diagnosis result emitted by the diagnoser is : $\{R_-\}$, that can be interpreted as "the event R_- was expected but did not occur". According to the Algorithm 2 of the section 4.1.1, the first step consists in the suppression of all transitions labeled with the faulty event. According to Fig. 6, transitions from R_7 to R_8 and from R_{26} to R_{27} must be deleted. The consequence is that final states R_9 and R_{28} cannot be reached anymore, but states R_{14} and R_{23} are still accessible. Hence, the only possible solution for the reconfigured control law corresponds to the reconfiguration model of Fig. 6 without states R_5, R_6, R_7, R_8, R_9, R_{24}, R_{25}, R_{26}, R_{27} and R_{28}.

5.3.3. *Third scenario: faulty case 2*

In this case, the sensor t_r is subject to activation delays, valued at 0,5 t.u. The diagnosis result is $\{t_r\}, late$. According to the Algorithm 3 of the section 4.1.2, transitions from R_5 to R_6 and from R_{19} to R_{20} are both adjusted to $t_r, [2{,}9; 3{,}6], r$, resulting in a difference in the average time of sequences leading to final states. Namely, it is faster in terms of execution time to reach states R_{14} and R_{28}. Hence, the reconfigured control law corresponds to the reconfiguration model of Fig. 6 without

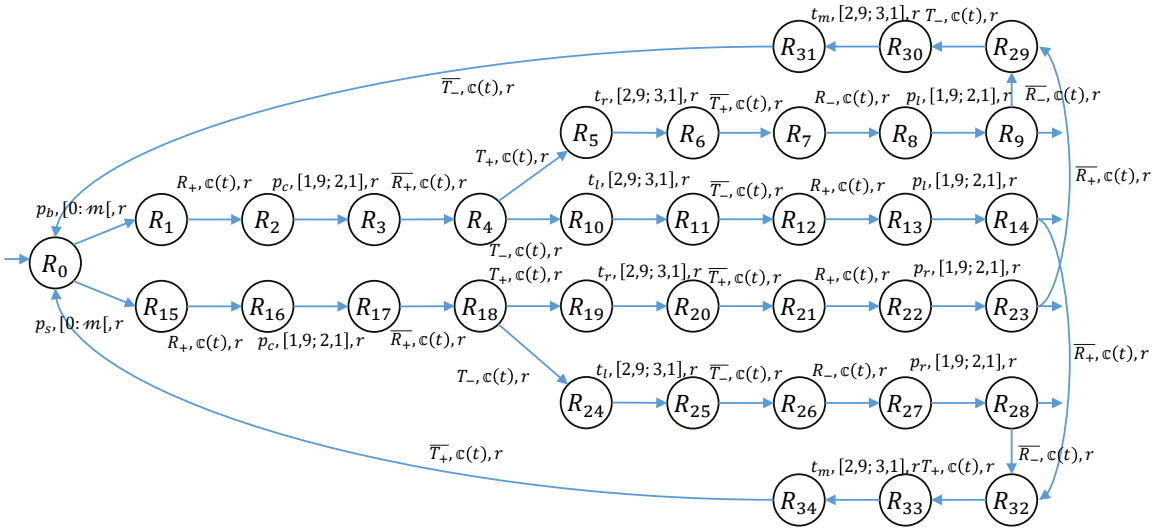

Fig. 6 - Reconfiguration model of the turntable

states R_5, R_6, R_7, R_8, R_9, R_{19}, R_{20}, R_{21}, R_{22}, R_{23},R_{29}, R_{30} and R_{31}.

6. CONCLUSION

A method of fault-tolerant control of timed automata with guards has been presented, based on the diagnosis obtained with timed-residuals. The reconfiguration is performed in two steps. First, the reconfiguration model is computed, representing the entire system behavior that respects timed rules. Secondly, this model and diagnostic results are used to search the fastest paths from the initial to the final states. Finally, these paths are used to compute the control law of the system for each case of operation. An example of application is provided on a simple system.

In future works, it would be interesting to use a linear representation of TAG (Niguez, Amari, and Faure 2016) in order to search for the fastest path during the reconfiguration step.

REFERENCES

Blanke, Mogens, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. 2016. *Diagnosis and Fault-Tolerant Control*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cassandras, Christos G., and Stéphane Lafortune. 2008. *Introduction to Discrete Event Systems*. 2. ed. New York, NY: Springer.

Niguez, Julien, Saïd Amari, and Jean-Marc Faure. 2015. "Fault-Tolerant Control of Discrete Event Systems: Comparison of Two Approaches on the Same Case Study." In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–4..

Niguez, Julien, Said Amari, and Jean-Marc Faure. 2016. "Analysis of Timed Automata with Guards in Dioids Algebra." In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 391–97.

Paoli, Andrea, Matteo Sartini, and Stéphane Lafortune. 2011. "Active Fault Tolerant Control of Discrete Event Systems Using Online Diagnostics." *Automatica* 47 (4): 639–49.

Schneider, Stefan, Lothar Litz, and Mickael Danancher. 2011. "Timed Residuals for Fault Detection and Isolation in Discrete Event Systems." In *3rd International Workshop on Dependable Control of Discrete Systems*, 35–40..

Schuh, Melanie., and Jan. Lunze. 2016a. "Fault-Tolerant Control of Deterministic I/O Automata with Ambiguous Diagnostic Result." In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 251–57.

———. 2016b. "Fault-Tolerant Control for Deterministic Discrete Event Systems with Measurable State." In *2016 American Control Conference (ACC)*, 7516–22.

Seong-Jin Park, and Jong-Tae Lim. 1999. "Fault-Tolerant Robust Supervisor for Discrete Event Systems with Model Uncertainty and Its Application to a Workcell." *IEEE Transactions on Robotics and Automation* 15 (2): 386–91.

Shu, Shaolong, and Feng Lin. 2014. "Fault-Tolerant Control for Safety of Discrete-Event Systems." *IEEE Transactions on Automation Science and Engineering* 11 (1): 78–89.

Wen, Qin, Ratnesh Kumar, Jing Huang, and H. Liu. 2008. "A Framework for Fault-Tolerant Control of Discrete Event Systems." *IEEE Transactions on Automatic Control* 53 (8): 1839–49.

Wittmann, Thomas, Jan Richter, and Thomas Moor. 2012. "Fault-Tolerant Control of Discrete Event Systems Based on Fault-Accommodating Models." *IFAC Proceedings Volumes*, 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, 45 (20): 854–59.

Wittmann, Thomas, Jan Richter, and Thomas Moor. 2013. "Fault-Hiding Control Reconfiguration for a Class of Discrete Event Systems." *IFAC Proceedings Volumes*, 4th IFAC Workshop on Dependable Control of Discrete Systems, 46 (22): 49–54.

ITS PLC software, Real Games, https://realgames.co/its-plc/.