Hybrid Systems Diagnosability by abstracting faulty continuous Dynamics

Mehdi Bayoudh and **Louise Travé-Massuyès**
LAAS-CNRS
Toulouse, France
{bayoudh, louise}@laas.fr

Xavier Olive
Alcatel Alenia Space
France
Xavier.Olive@space.alcatel.fr

Abstract

On line model based reconfiguration is generally used to improve the ability of a system to tolerate faults. Recovery after fault occurrence relies on allowing the system to proceed with its mission from a new known nominal state. In this paper, we consider on-line reconfiguration from a novel point of view, having in mind to use reconfiguration actions to desambiguate the tracked estimated system state, i.e. to produce a more precise diagnosis. The choice of the best suited reconfiguration action(s) must hence be guided by the diagnosability properties of the system. However, diagnosability conditions known for continuous systems (CS) on one hand and for discrete event systems (DES) on the other hand cannot be applied directly because of the hybrid nature of the systems that we consider. Our work proposes a framework for analyzing the diagnosability of a hybrid system which stands on recent results establishing the formal equivalence of diagnosability definitions for DES and CS. The approach relies on merging the fault signatures exhibited at the continuous level into the Mode Automaton that represents the discrete dynamics of the system, so that DES diagnosability analysis can be performed on the resulting Behavior Automaton and the corresponding diagnoser. When the state of the system is ambiguous, an analysis of the diagnoser should allows to point at reconfiguration actions that safely move the system into a mode reducing ambiguity.

1 Introduction

Embedded systems found in nowadays cars, aircrafts and space vehicles are characterized by a mix of hardware and software components and limited instrumentation. They hence undergo complex hybrid dynamics that can only be partially observed, which makes tricky their on-board monitoring and diagnosis. They generally require to use stochastic and/or uncertain approaches which provide a belief state or in other words an ambiguous diagnosis [Hofbaur and Williams, 2004] [Benazera *et al.*, 2002] [Williams and Nayak, 1999]. In many cases, testing the system on line could be an interesting option to produce a more precise diagnosis. For instance, in the space domain, specific commands are often applied by the ground segment for getting more information about the state of a faulty spacecraft.

This kind of testing, that we call *active diagnosis*, involves reconfiguring the system so that new symptoms are exhibited through the existing sensor instrumentation. The choice of the best suited reconfiguration action (s) must hence be guided by the diagnosability properties of the system. Diagnosability analysis proves to be a requisite for several other tasks such as instrumentation design, end-of-line testing, etc and has deserved a lot of attention from the Model Based Diagnosis community in the last few years, both for the analysis of Continuous Systems (CS) and Discrete Events Systems (DES) [Sampath *et al.*, 1995] [Pencolé, 2004] [Travé-Massuyès *et al.*, 2004].

However, diagnosability conditions known for continuous systems (CS) on one hand, and for discrete event systems (DES) on the other hand, cannot be applied directly when the system has hybrid dynamics. We rely on very recent results establishing the formal equivalence of diagnosability definitions for DES and CS [Cordier *et al.*, 2006] and propose to abstract the faulty continuous dynamics of a hybrid automaton to produce a enriched discrete automaton that accounts for fault models. Fault models are obtained from fault signatures exhibited from the continuous dynamics constraints that are interpreted in terms of events.

DES diagnosability analysis can be performed on the resulting Behavior Automaton and the corresponding diagnoser. When the state of the system is ambiguous, an analysis of the diagnoser should allow to point at reconfiguration actions that safely move the system into a mode reducing ambiguity.

The paper is organized as follows. Section 2 introduces the hybrid modelling framework used for tracking the states of the system. Section 3 provides an insight into fault signatures as defined for continuous systems and gives the intuitions guiding our contribution. Section 4 then introduces the main DES diagnosability notions. Section 5 presents the procedure for building the Behavior Automaton from the Mode Automaton and fault signatures exhibited at the continuous behavior level. Finally, section 6 illustrates our approach with a motivational example. Related work, perspectives for future work and concluding discussion are provided in section 7.

2 Hybrid modeling framework

Embedded systems combine continous dynamics with discrete events (which can be commanded or spontaneous). Hence, the hybrid formalism is appropriate for modelling complex dynamic systems. An hybrid system is described by a hybrid automaton defined as a tuple $S = (X, Q, \Sigma, T, C)$, where:

- X is the set of continuous variables, which includes observable and non observables variables. Those variables are linked with constraints that vary from one mode to another.
- Q is the set of discrete variables. Each state $q_i \in Q$ represents a functional mode of the system.
- Σ is the set of events. Events correspond to command value switches, spontaneous mode changes and fault events.
- $\Sigma o \subseteq \Sigma$ is the set of observable events. Without loss of generality we can assume that fault events are unobservable.
- T is the transition function, $\mathrm{Q} \times \Sigma \rightarrow Q$.
- C is the set of constraints which may be qualitative or quantitative. Associating a subset of constraints $C_i \subseteq C$ to functional mode q_i allows one to describe the system behavior evolution in this mode. Quantitative constraints can be derived from algebraic or differential equations.
- (x_0, q_0) is the initial condition.

The discrete part of the hybrid automaton, given by $M = (Q, \Sigma, T, q_0)$, is a discrete automaton that describes the discrete dynamics of the system, i.e. the possible evolutions between operating modes in Q. We refer to this automaton as the *Mode Automaton*. Modes include nominal and fault modes as well as an unknown mode which stands for all the non anticipated fault situations. The unknown mode has no specified underlying behavior and hence no associated constraints.

3 Fault Signatures

Following the parity space approach, the constraints in each mode q_i can be brought back to a set of analytical redundancy relations ($SARR_{q_i}$) by eliminating non observable variables [Cordier *et al.*, 2004]. An ARR can be expressed as $r = 0$, where r is called the residual of the ARR. The ARRs are constraints that only contain observable variables. They can be determined off-line and then be evaluated on-line with the incoming observations, allowing one to check the consistency of the observed against the predicted system's behavior. They are satisfied if the observed behavior satisfies the model constraints, in which case the associated residuals are zero. In the opposite case, all or some of the residuals are non zero. The set of residuals hence results in a boolean fault indicator tuple. The expected boolean value pattern for a given fault provides the *fault signature*. In our hybrid framework, the set of ARRs linked with each functional system mode is generally different, although some ARRs may be shared. A fault hence manifests by the fact that a subset of residuals switches to a non zero value, whereas other residuals may switch from an undetermined value to zero.

Definition 1 *Given a set $[r_1, ..., r_n]$ of n residuals and a set $F = [F_1, F_2, ..., F_m]$ of m faults, the signature of a fault F_j is given by the binary vector $FS_i = [s_{1j}, ..., s_{nj}]^T$, $s_{ij} = 1$ if some components affected by F_j are involved in ARR_i, $s_{ij} = 0$ otherwise.*

Residuals and fault signatures provide an abstracted information about the continuous dynamics of the system which is sufficient for characterizing the system's nominal or faulty state. When a fault occurs, fault signatures can be interpreted in terms of events referring to the residuals switching values. Our goal is to take advantage of this event driven information to enrich the system's mode automaton, abstracting the continuous dynamics of the hybrid automaton into an extended discrete automaton that we call the *Behavior Automaton*. The diagnosability of the hybrid system can thus be analysed from the Behavior Automaton, by using discrete event systems criteria [Sampath *et al.*, 1995].

4 DES diagnosability Analysis

4.1 Diagnosability definition

Diagnosability is the property of a system and its observables, i.e. set of all the possible observations, that guarantees that a set of anticipated fault situations can be assessed and distinguished. Diagnosability definitions have been provided independently for CS and for DES [Cordier *et al.*, 2006] [Sampath *et al.*, 1995]. However, recent results have proved that definitions on both sides are formally equivalent [Cordier *et al.*, 2006]. We take benefit of this result and propose to interpret the CS fault signatures that are the key diagnosability concept in terms of an automaton that can be merged into the discrete dynamics model. In this way, the diagnosability problem for the hybrid system is brought back to the diagnosability problem for an extended DES system. In consequence, this section restricts the presentation to the DES diagnosability definition and analysis through the so-called diagnoser [Sampath *et al.*, 1995]. A DES is modeled by a finite state machine $M = (Q, \Sigma, T, q_0)$ where Q is the set of states, Σ is the set of events, $T \subseteq (Q \times \Sigma \times Q)$ the transition function and q_0 the initial state, as already defined in section 2. The event set Σ is partitioned as $\Sigma = \Sigma_{uo} \cup \Sigma_o$, where Σ_{uo} is the unobservable event set and Σ_o the observable event set. Observable events are system commands or events generated from the sensors. In our approach, these later observable events are the residual value switches. We consider $\Sigma_f \cup \Sigma_{uo}$ as the set of fault events to be diagnosed. In the DES community, the diagnosis consists in the deduction of unobservable fault events from the observable traces generated by the system.

Definition 2 *A fault F is diagnosable iff for each trajectory s_F containing the fault event, there exists a finite sequence of observable events $[o_1, o_2, ..., o_n]$ that allows us to diagnose F with certainty.*

Formally, the fault F is diagnosable iff there exists a subset of observable events $OBS_F \subseteq \Sigma_o$, such that, for each trajectory s_F containing the fault event, there exists a finite number n such that, for each continuation t of s_F, if the length of

$s_F t$ is greater than n, then $\mathcal{P}roj|_{\Sigma_o}(s_F t) = OBS_F$, where $\mathcal{P}roj|_{\Sigma_o}(s_F t)$ is the projection of $s_F t$ on the observable set Σ_o.

4.2 The diagnoser

We assume that M has no unobservable cycles (i.e cycles containing unobservable events only). The set of fault events Σ_F is partitioned into disjoint sets corresponding to different failure types, $\Sigma_F = \Sigma_{F_1} \cup \Sigma_{F_2} \cup ... \cup \Sigma_{F_n}$ and $\Sigma_{F_i} \cap \Sigma_{F_j} = \emptyset$, $for\, i \neq j$. The aim of the diagnosis is to make inferences about past occurrences of failure types on the basis of the observed events. In order to solve this problem the system model is directly converted into the diagnoser.
The diagnoser $Diag(M) = (Q_{Diag}, \Sigma_{Diag}, T_{Diag}, q_{0\,Diag})$ is a deterministic finite state machine built from the system model $M = (Q, \Sigma, T, q_0)$ [Sampath *et al.*, 1995].
$q_{0Diag} = \{(q_0, \{\emptyset\})\}$ is the initial state of the diagnoser.
$\Sigma_{Diag} = \Sigma_o$ is the set of observable events of the system.
Q_{Diag} is the set of states of the diagnoser: $Q_{Diag} \subseteq 2^{Q \times 2^{\Sigma_F}}$ or $Q_{Diag} \subseteq \mathcal{P}(Q \times \mathcal{P}(\Sigma_F))$, where $\mathcal{P}(E)$ denotes the power set of E. The states of the diagnoser provide the set of diagnosis candidates as a set of couples whose first element refers to the state of the original system and the second is a label providing the set of faults on the path leading to this state. For example, when the diagnoser is in the state $q_{Diag} = \{(q_1, \{\,\}), (q_2, \{\Sigma_{F_1}, \Sigma_{F_3}\})\}$, it means that the system M is in one of the states q_1, q_2 as developed in table 1[1].
T_{Diag} is the diagnoser transition function which is built by a recursive process which consists in computing all the reachable states from the diagnoser initial state and by propagating the diagnosis information. For more details see [Sampath *et al.*, 1995].

State	Diagnosis	Comments
q_1	$\{\emptyset\}$	the system my be in the nominal state q_1 (no fault)
q_2	$\{\{\Sigma_{F_1}, \Sigma_{F_3}\}\}$	the system my be in the faulty state q_3 with a diagnosis $\{\Sigma_{F_1}, \Sigma_{F_3}\}$ (Σ_{F_i} means that at least one fault of type Σ_{F_i} has occurred)

Table 1: The $\{\Sigma_{F_1}, \Sigma_{F_3}\}$ uncertain state of the Diagnoser

Definition 3 *Given a diagnoser state $q_{Diag} \in Q_{Diag}$, this state is Σ_{F_i}-uncertain iff Σ_{F_i} does not belong to all the labels of the state.*

Definition 4 *The system M is not diagnosable iff the associated diagnoser $Diag(M)$ contains an uncertain cycle, i.e. a cycle in which there is at least one Σ_{F_i}-uncertain state for some Σ_{F_i} and whose states also define a cycle in the original system M.*

[1]We do not use the ambiguous label used in [Sampath *et al.*, 1995], but we explicitly give the set of faulty system modes

5 Building the Behavior Automaton

In the classic FDI approach, a fault manifests itself as anticipated in the fault signature, which reduces the detection task to detecting the violation of a subset of set of ARRs. In this paper, we propose to model the (nominal and faulty) continuous behavior of the hybrid system based on events referring to the set of ARRs associated to the different modes. For each mode of the system (nominal and faulty), we associate a so- called *Local Behavior Automaton* constructed from the knowledge of the residuals that must switch value when transitioning to this mode: these residuals include a subset of the residuals associated to the departure modes that switch to non zero value and the residuals in the current mode that must switch to zero. Notice that the same procedure is indifferently applicable for transitions triggered by command events, fault event or spontaneous events. Local behavior automata hence evolve with the occurrence of residual value switches that define a set of events. They should allow us to determine the unobservable event (fault or spontaneous) that occurred at the transitioning between modes by the analysis of their observable trajectories.
The system's Behavior Automaton is obtained as an extension of the Mode Automaton by the local behavior automata for each transition.
Let $SARR_{q_i} = \{ARR_{i1}, ARR_{i2}, ..., ARR_{iN_{ARR}(q_i)}\}$ be the set of ARRs associated to mode q_i and $Sr_{q_i} = \{r_{i1}, r_{i2}, ..., r_{iN_{ARRS}(q_i)}\}$ the associated set of residuals, where $N_{ARR}(q_i)$ is the number of ARRs in mode q_i. We denote by $Sr_{system} = \bigcup_i Sr_{q_i}$ the set of all residuals for the system and we denote by $\mathcal{D} = \{0, 1, und\}$ the residuals value domain, where und stands for the undefined value that is used to represent the case when the associated ARR is not defined in one given mode.
Now, let us define the function e, which associates an event to every residual value switch:

$$\begin{aligned} e : Sr_{system} \times \mathcal{D} \times \mathcal{D} &\longrightarrow \Sigma_{behav} \\ (r_{ij}, l, k) &\longmapsto e_{ij}^{lk} \end{aligned}$$

The event e_{ij}^{lk} is hence associated to the residual r_{ij} switching from value l to value k.
The Local Behavior Automaton for a given system mode q_i (either nominal or faulty) is defined as $M^i = (Q^i, \Sigma^i, T^i, q_0^i)$ where:

- Q^i is the set of local behavior automaton states, each state $q_{i,k} \in Q^i$ is characterized by an instance of the global set of residuals Sr_{system} and the trajectories exhibit the different possible order for the residuals switches.
- Σ^i is the set of events, each event corresponds to one residual value switch.

As stated before, the system's Behavior Automaton is obtained as an extension of the Mode Automaton by the local behavior automata for each mode state. This procedure allows us to generate the system's Behavior Automaton in a mode-driven way, avoiding to enumerate all possible states, see (figure4). Indeed formally, the system's Behavior Automaton is the synchronous product of the automata defining

the residual value switches in which all non accessible states and impossible transitions, defined by the Mode Automaton, are discarded. The proof of equivalence is not provided in this paper.

6 Motivational Example

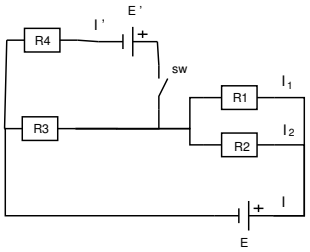

Figure 1: Circuit

In this section, we take example the electrical circuit, (figure1), which has two nominal operating modes N1 et N2, commanded by a switch sw. Without loss of generality, in this example, we only deal with the faulty modes involving the component R2, see (figure2). Otherwise, the centralized model of the whole system, can be obtained by taking into account the other component misbehaviors. The observable variables are: the voltage E, E' and the currents I, I', I_1, I_2 and I_3.

6.1 The modes automaton

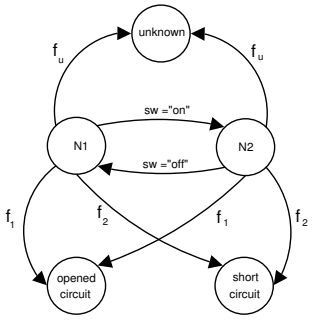

Figure 2: The modes automaton of the part in system which tacks into account only the misbehaviors invloving the component R2.

6.2 The ARRs set by each mode

We compute the several ARRs of the system in the two nominal modes and in the fault mode opened circuit.

In the nominal mode N1, $(sw = off)$

$$ARR_{11} : I_{1obs} = \frac{R_2}{R_1 + R_2} I_{obs} \quad (1)$$
$$ARR_{12} : E_{obs} = R_3 I_{obs} + R_1 I_{1obs} \quad (2)$$
$$ARR_{13} : E_{obs} = R_3 I_{obs} + R_2 I_{2obs} \quad (3)$$

In the nominal mode N2, $(sw = on)$

$$ARR_{2}1 : I_{1obs} = \frac{R_2}{R_1 + R_2} I_{obs} \quad (4)$$
$$ARR_{22} : E_{obs} = R_3(I_{obs} + I'_{obs}) + R_1 I_{1obs} \quad (5)$$
$$ARR_{23} : E_{obs} = R_3(I_{obs} + I'_{obs}) + R_2 I_{2obs} \quad (6)$$
$$ARR_{24} : E'_{obs} = R_4 I'_{obs} + R3(I_{obs} + I'_{obs}) \quad (7)$$

In the fault mode opened circuit, $(sw = off)$

$$ARR_{O11} : E_{obs} = R_3 I_{obs} + R_1 I_{1obs} \quad (8)$$
$$ARR_{O12} : I_{obs} = I_{1obs} \quad (9)$$

In the fault mode opened circuit, $(sw = on)$

$$ARR_{O21} : E_{obs} = R_3(I_{1obs} + I'_{obs}) + R_1 I_{1obs} \quad (10)$$
$$ARR_{O22} : E'_{obs} = R_4 I'_{obs} + R3(I_{obs} + I'_{obs}) \quad (11)$$
$$ARR_{O23} : I_{obs} = I_{1obs} \quad (12)$$

6.3 The Behavior Automaton

In this part, we present the part of the behavior automaton corresponding to the transitions from nominal modes to the fault mode opened circuit, see (figure3). Each mode of the behavior automaton is characterized by one instance of the residual set, this mapping is given in tables (2, 3, 4 and 5).

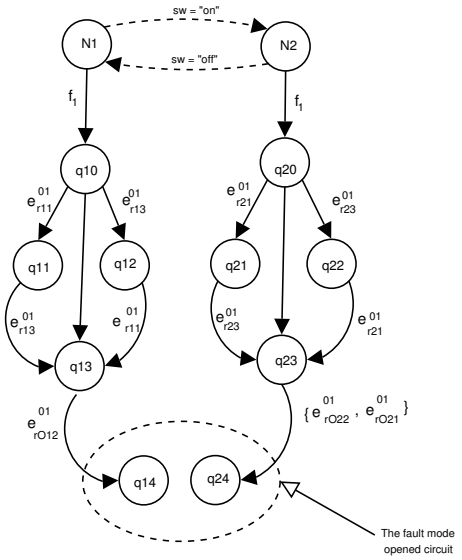

Figure 3: The Local Behavior Automaton associated to the fault mode opened circuit.

7 Discussion and conclusions

This paper proposes a framework for analyzing the diagnosability of a hybrid system which stands on recent results establishing the formal equivalence of diagnosability definitions for DES and CS. The approach relies on merging

modes	$[r_{11}, r_{12}, r_{13}]$	$[r_{21}, r_{22}, r_{23}, r_{24}]$
$q10$	$[0, 0, 0]$	$[und, und, und, und]$
$q11$	$[1, 0, 0]$	$[und, und, und, und]$
$q12$	$[0, 0, 1]$	$[und, und, und, und]$
$q13$	$[1, 0, 1]$	$[und, und, und, und]$
$q14$	$[1, 0, 1]$	$[und, und, und, und]$

Table 2: a-1, Mapping between the modes of the Local behavior automaton and the instances of the residuals.

modes	$[r_{O11}, r_{O12}]$	$[r_{O21}, r_{O22}, r_{O23}]$
$q10$	$[und, und]$	$[und, und, und, und]$
$q11$	$[und, und]$	$[und, und, und, und]$
$q12$	$[und, und]$	$[und, und, und, und]$
$q13$	$[und, und]$	$[und, und, und, und]$
$q14$	$[0, 0]$	$[und, und, und, und]$

Table 3: a-2, Mapping between the modes of the Local behavior automaton and the instances of the residuals.

modes	$[r_{11}, r_{12}, r_{13}]$	$[r_{21}, r_{22}, r_{23}, r_{24}]$
$q20$	$[und, und, und]$	$[0, 0, 0, 0]$
$q21$	$[und, und, und]$	$[1, 0, 0, 0]$
$q22$	$[und, und, und]$	$[0, 0, 1, 0]$
$q23$	$[und, und, und]$	$[1, 0, 1, 0]$
$q24$	$[und, und, und]$	$[1, 0, 1, 0]$

Table 4: b-1, Mapping between the modes of the Local behavior automaton and the instances of the residuals.

modes	$[r_{O11}, r_{O12}]$	$[r_{O21}, r_{O22}, r_{O23}]$
$q20$	$[und, und]$	$[und, und, und]$
$q21$	$[und, und]$	$[und, und, und]$
$q22$	$[und, und]$	$[und, und, und]$
$q23$	$[und, und]$	$[und, und, und]$
$q24$	$[und, und]$	$[0, 0, 0]$

Table 5: b-2, Mapping between the modes of the Local behavior automaton and the instances of the residuals.

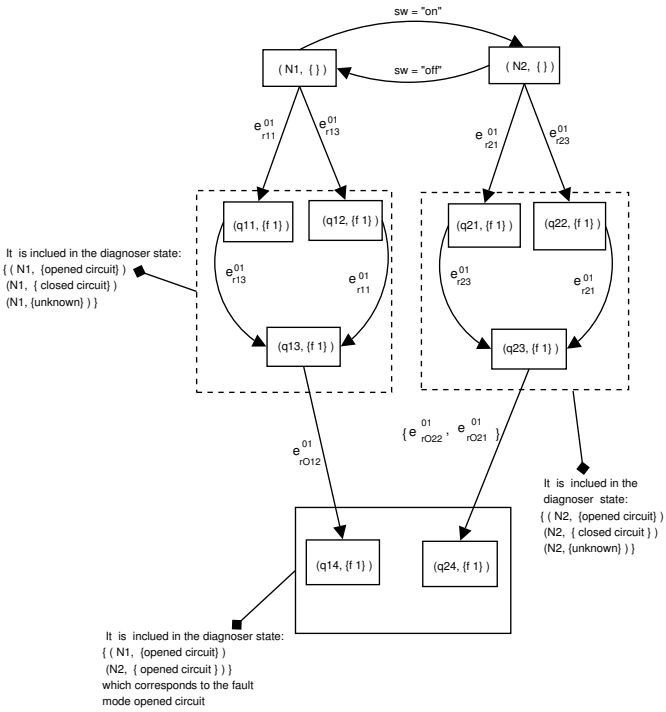

Figure 4: The Associated Local Diagnoser

the fault signatures exhibited at the continuous level into the Mode Automaton that represents the discrete dynamics of the system, so that DES diagnosability analysis can be performed on the resulting Behavior Automaton and the corresponding diagnoser. When the state of the system is ambiguous, an analysis of the diagnose should allow to point at reconfiguration actions that safely move the system into a mode reducing ambiguity. To our knowledge there is no existing work proposing a method to analyze the diagnosability of a hybrid system. The method that we propose interprets the continuous dynamics of the system in terms of events and gives a procedure to merge this knowledge into the discrete dynamics model. Our approach can be related to the work by Lunze which uses Quantized Automata [Lunze, 2000a] [Lunze, 2000b]. Lunze starts with a continuous system and discretizes the continuous variable value domains. From this discretization, he is able to produce a behavior automaton that accounts for all the variable value switches. The behavior automaton that he produces is hence oriented towards behavior prediction and simulation purposes and its semantics are quite different from the behavior automaton that we produce. In our case, we have pursued the goal to obtain the same behavior automaton as used by the model-based DES diagnosis community [Sampath *et al.*, 1995] [Puig *et al.*, 2005] [Lamperti and Zanella, 2002], so that their results can then be applied as so. For this purpose, the abstraction of the continuous dynamics is performed from the continuous subspaces that characterize the different modes of the system and the switches undergone by the system state. The subspaces are generated by the ARRs and the switches corre-

spond to value switches for their corresponding residuals. In this framework, fault signatures are uniquely defined, which is not the case when basing the abstraction on a state variable value partionning of the state space as used by Lunze [Lunze, 2000a]. This paper is in continuation with the work done by the French Imalaia group [Cordier *et al.*, 2004] and the Bridge Task Group within the MONET network of Excellence. It hence uses the knowledge and results obtained by these two groups and establishes yet another bridge between two model based communities, namely the continuous and the DES model based communities. Future work will be devoted to the problem of using diagnosability assessment for selecting the best reconfiguration action so that the system is driven into a mode in which new symptoms are exhibited, hence desambiguating the diagnosis. This problems goes beyond selecting and applying a discrete action. Indeed, some physical constraints may require to planify a sequence actions and the hybrid nature of the system may call for hybrid control.

References

[Benazera *et al.*, 2002] E. Benazera, L. Travé-Massuyès, and P. Dague. State tracking of uncertain hybrid concurrent systems. In *13th International Workshop on Principles of Diagnosis (DX'02)*, 2002.

[Cordier *et al.*, 2004] M.O. Cordier, P. Dague, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès. Conflicts versus analytical redundancy relations : A comparative analysis of the model-based diagnostic approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man and Cybernetics, Part B.*, 34(52163-2177), 2004.

[Cordier *et al.*, 2006] M.O. Cordier, L. Ttavé-Massuyès, and Xavier Pucel. Comparing diagnosability criterions in continuous systems and descrete events systems. *Rapport LAAS N06070*, 2006.

[Hofbaur and Williams, 2004] M.W. Hofbaur and B.C. Williams. Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B.*, 34(5):2178–2191, 2004.

[Lamperti and Zanella, 2002] Gianfranco Lamperti and Marina Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artif. Intell.*, 137(1-2):91–163, 2002.

[Lunze, 2000a] Jan Lunze. Diagnosis of quantized systems based on a timed discrete-event model. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 30(3):322–335, 2000.

[Lunze, 2000b] Jan Lunze. Diagnosis of quantized systems by means of timed discrete-event representations. In *HSCC*, pages 258–271, 2000.

[Pencolé, 2004] Y. Pencolé. Diagnosability analysis of distributed discrete event systems. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, pages 43–47, 2004.

[Puig *et al.*, 2005] V. Puig, J. Quevedo, T. Escobet, and B. Pulido. On the integration of fault detection and isolation in model based fault diagnosis. In *Proceedings of DX'05*, pages 227–232, 2005.

[Sampath *et al.*, 1995] M. Sampath, R. Sengputa, S. Lafortune, K. Sinnamohideen, and D. Teneketsis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40:1555–1575, 1995.

[Travé-Massuyès *et al.*, 2004] L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *Rapport LAAS N04080, To appear in IEEE Transactions on Systems, Man and Cybernetics, Part A*, 2004.

[Williams and Nayak, 1999] Brian C. Williams and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In Jack Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999*, College Park, Maryland, 1999. Computer Science Department, University of Maryland.