A Game-Theoretic approach to Fault Diagnosis of Hybrid Systems*

Davide Bresolin Marta Capiluppi
Università di Verona
Dipartimento di Informatica
Verona, Italy
davide.bresolin@univr.it, marta.capiluppi@univr.it

Physical systems can fail. For this reason the problem of identifying and reacting to faults has received a large attention in the control and computer science communities. In this paper we study the fault diagnosis problem for hybrid systems from a game-theoretical point of view. A hybrid system is a system mixing continuous and discrete behaviours that cannot be faithfully modeled neither by using a formalism with continuous dynamics only nor by a formalism including only discrete dynamics. We use the well known framework of hybrid automata for modeling hybrid systems, and we define a Fault Diagnosis Game on them, using two players: the environment and the diagnoser. The environment controls the evolution of the system and chooses whether and when a fault occurs. The diagnoser observes the external behaviour of the system and announces whether a fault has occurred or not. Existence of a winning strategy for the diagnoser implies that faults can be detected correctly, while computing such a winning strategy corresponds to implement a diagnoser for the system. We will show how to determine the existence of a winning strategy, and how to compute it, for some decidable classes of hybrid automata like o-minimal hybrid automata.

1 Introduction

In modern complex systems continuous and discrete dynamics interact. This is the case of wide manufacturing plants, agents systems, robotics and physical plants. This kind of systems, called hybrid in their behaviour, need a specific formalism to be analysed. In order to model and specify hybrid systems in a formal way, the notion of *hybrid automata* has been introduced [2, 22]. Intuitively, a hybrid automaton is a "finite-state automaton" with continuous variables that evolve according to dynamics characterizing each discrete state. In the last years, a wide spectrum of modeling formalism and algorithmic techniques has been studied in the control and computer science communities to solve the problems of simulation, verification and control synthesis for hybrid systems. Much scarce attention have been posed to the problem of dealing with faults. When a hybrid system fail, the failure propagates throughout the system both in continuous and discrete evolutions. Nevertheless the interaction of continuous and discrete dynamics leads to the need of studying new theories for fault tolerance.

A *fault* is a deviation of the system structure or the system parameters from the nominal situation [6]. This implies that after the occurrence of a fault the system will have a behaviour which is different from the nominal one. Hence *Fault Tolerance* is the property of reacting to faults. In particular the analysis of fault tolerance consists in establishing if a given system is still able to achieve its tasks after the occurrence of a given fault, whereas the synthesis of fault tolerance resides in providing a given system the tools to react to a given faulty situation. The fault tolerance problem can be divided in two tasks: fault detection and isolation (FDI) and control redesign. FDI produces a diagnostic result including detection

*This research was partly supported by the EU Project FP7-ICT-223844 CON4COORD.

Giovanna D'Agostino, Salvatore La Torre (Eds.):
Proceedings of the Second International Symposium on
"Games, Automata, Logics and Formal Verification" (GandALF 2011)
EPTCS 54, 2011, pp. 237–249, doi:10.4204/EPTCS.54.17

and location of the fault, and if possible an estimate of the dimensions of the fault. In this paper we concentrate our attention to the problem of fault detection and isolation for hybrid systems.

Fault tolerance and fault tolerant systems have been studied by the control community since the late '70s, as in [16] where fault detection for chemical processes is introduced, and later in [25]. One of the first surveys on fault detection is [17], which is dated 1984, and where some methods based on modelling and estimation are introduced. Much later the interesting book [24] collects some results on Fault Detection and Isolation (FDI) methods. For a complete outline of the recent improvements in this field, it is worth citing [26] where a quite new approach to fault detection in industrial (batch) systems is introduced and [19], an overview on fault tolerant techniques for flight control.

In the computer science community fault tolerance is also known as *Fault localization and correction*, and it is usually viewed as the problem of finding and fixing bugs in a software program or in a digital circuit. One of the most systematic approaches in this area is Model Based Diagnosis, where an oracle provides an example of correct behavior that is inconsistent with the behavior of the faulty system, and a correct model of the system is usually not necessary [11]. Model based diagnosis can be distinguished between abduction-based and consistency-based diagnosis. Abduction-based diagnosis [27] assumes that it is known in which ways a component can fail. Using a set of fault models, it tries to find a component and a corresponding fault that explains the observation. Consistency-based diagnosis [12, 28] considers the faulty behavior as a contradiction between the actual and the nominal behavior of the system. It does not require the possible faults to be known, and it proceeds by dropping the assumptions on the behavior of each component in turn. If this removes the contradiction, the component is considered a candidate for correction. More recently, applicability of discrete game theory to fault localization and automatic repair of programs have been proposed in [18]. In this alternative setting, the specification of the correct behaviour is given in Linear Temporal Logic and the correction problem is stated as a game, in which the protagonist selects a faulty component and suggests alternative behaviours.

Not many attempts have been made until now in the field of fault diagnosis for hybrid systems. This can be due in first instance to the hard task of state estimation in this kind of systems. Indeed to know if a fault has occurred it has to be detected if the system is behaving in an unusual way, that is based on the knowledge of the state in which the system is working. When dealing with hybrid systems a state estimator must provide both the continuous and the discrete state. The accomplishment of this task is generally difficult because of the coupling of the two dynamics.

Among the first methods for fault detection of hybrid systems it is worth citing the ones presented in [23] and [29]. These two methods are quite different, because they are based on opposite models of hybrid systems. The first one deals with mixed logical dynamical (MLD) systems, and mainly with faults on the continuous dynamics, whereas the second one uses quantised systems, then it deals mainly with the discrete part. The method introduced in [13] presents some results based on Hybrid Input/Output Automata [21] and extends the theory of diagnosability for discrete events systems to the hybrid case. As usual in this kind of discrete event approach to hybrid systems, the two dynamics are kept separated, which means that the diagnoser has to first check if some fault has occurred in the current (discrete) mode, then to check the continuous dynamics inside the mode, finally a supervisor will decide which kind of fault has occurred and where. Nevertheless the diagnosability is tested on the hybrid dynamics, using the notion on hybrid traces.

In this paper we choose to start from the modeling framework of [21], where Hybrid Automata assume a distinction between internal and external actions and variables. We add faults to this model, by using a distinguished *fault* action. This is not a restrictive assumption, since every kind of fault can be modeled as an internal action of an automaton, supposing the fault action leads from a nominal state to a faulty one in the system. We assume that after a fault the system remains in its faulty situation and never

recovers.

We choose to use game theory applied to fault diagnosis of hybrid systems because it allows us not to split the continuous and the discrete behaviours. A hybrid game is a multiplayer structure where the players have both discrete and continuous moves and the game proceeds in a sequence of rounds. In every round each player chooses either a discrete or a continuous move among the available ones [30]. Hybrid games has been successfully applied to solve the controller synthesis problem for timed [4] and hybrid automata [7, 15], and to the fault diagnosis problem for timed automata [9]. In our setting we model the fault diagnosis problem as a game between two players, the environment and the diagnoser. The environment controls the evolution of the system and chooses whether and when a fault occurs. The diagnoser observes the external behaviour of the system and announces whether a fault has occurred or not. Existence of a winning strategy for the diagnoser implies that faults can be identified correctly, while computing such a winning strategy corresponds to implement a diagnoser for the system. In contrast with the usual definition of hybrid game, our game is asymmetric, since the environment is more powerful than the diagnoser, and is under partial observability, since the diagnoser is blind to the value of internal variables and to the occurrence of internal events. We define two notions of diagnosability, and we prove that the fault diagnosis problem is solvable for the weakest notion of diagnosability for all classes of hybrid automata that admit a bisimulation with finite quotient that can be effectively computed.

2 Hybrid Automata with Faults

Throughout the paper we fix the *time axis* to be the set of non-negative real numbers $\mathbb{R}^+$. An *interval* I is any convex subset of $\mathbb{R}^+$, usually denoted as $[t_1,t_2] = \{t \in \mathbb{R}^+ : t_1 \leqslant t \leqslant t_2\}$. For any interval I and $t \in \mathbb{R}^+$, we define $I+t$ as the interval $\{t'+t : t' \in I\}$.

We also fix a countable universal set $\mathcal{V}$ of *variables*, where every variable $v \in \mathcal{V}$ has a type $\mathrm{Type}(v)$ which defines the domain over which the variable ranges. Elementary types include *booleans*, *integers* and *reals*. Given a set of variables $V \subseteq \mathcal{V}$, a *valuation* over V is a function that associate every variable in V with a value in its type. We often refer to valuation as *states*, and we denote them as $\mathbf{x},\mathbf{y},\mathbf{z},\ldots$. The set $\mathrm{Val}(X)$ is the set of all valuations over X. Given a valuation $\mathbf{x}$ and a subset of variables $Y \subseteq X$, we denote the *restriction* of $\mathbf{x}$ to Y as $\mathbf{x}|Y$. The restriction operator is extended to sets of valuations in the usual way.

A notion that will play an important role in the paper is the one of *trajectory*. A trajectory over a set of variables X is a function $\tau : I \mapsto \mathrm{Val}(X)$, where I is a left-closed interval with left endpoint equal to 0. With $\mathrm{dom}(\tau)$ we denote the domain of τ, while with $\tau.ltime$ (the *limit time* of τ) we define the supremum of $\mathrm{dom}(\tau)$. The *first point* of a trajectory is $\tau.fval = \tau(0)$, while, when $\mathrm{dom}(\tau)$ is right-close, the *last point* of a trajectory is defined as $\tau.lval = \tau(\tau.ltime)$. We denote with $Trajs(X)$ the set of all trajectories over X. Given a subset $Y \subseteq X$, the *restriction* of τ to Y is denoted as $\tau|Y$ and it is defined as the trajectory $\tau' : \mathrm{dom}(\tau) \mapsto \mathrm{Val}(Y)$ such that $\tau'(t) = \tau(t)|Y$ for every $t \in \mathrm{dom}(\tau)$.

A trajectory τ' is a *prefix* of another trajectory τ if and only if $\tau'.ltime \leqslant \tau.ltime$ and $\tau'(t) = \tau(t)$ for every $t \in \mathrm{dom}(\tau')$. Conversely, we say that τ' is a *suffix* of τ if there exists $t \in \mathbb{R}^+$ such that $\tau'.ltime = \tau.ltime - t$ and $\tau'(t') = \tau(t'+t)$ for every $t' \in \mathrm{dom}(\tau')$. Given two trajectories τ_1 and τ_2 such that $\tau_1.lstate = \tau_2.fstate$, their concatenation $\tau_1 \cdot \tau_2$ is the trajectory with domain $\mathrm{dom}(\tau_1) \cup (\mathrm{dom}(\tau_2) + \tau_1.ltime)$ such that $\tau_1 \cdot \tau_2(t) = \tau_1(t)$ if $t \in \mathrm{dom}(\tau_1)$, $\tau_1 \cdot \tau_2(t) = \tau_2(t - \tau_1.ltime)$ otherwise. We extend the concatenation operation to countable sequences of trajectories in the usual way.

We model hybrid systems with faults by using the formalism of Hybrid Automata (HA) as defined by Lynch, Segala, and Vandraager in [21], enriched with a distinguished *fault* action, and with a partition of

the state space into faulty and non-faulty states. We assume a single type of faults for simplicity reasons. However, all the results presented in the paper can be easily generalized to a finite number of faults.

Definition 2.1. *A* Hybrid Automaton with Faults *is a tuple* $\mathcal{A} = \langle W, X, Q, Q_f, \Theta, E, H, f, D, \mathcal{T} \rangle$*, where:*

- *W and X are two finite sets of* external *and* internal *variables, disjoint from each other. We define $V = W \cup X$;*
- *$Q \subseteq \mathrm{Val}(X)$ is the set of* states*;*
- *$Q_f \subset Q$ is the set of* faulty states*. We define Q_n the set of* non-faulty states *such that $Q = Q_n \cup Q_f$ and $Q_n \cap Q_f = \emptyset$.*
- *$\Theta \subseteq Q_n$ is a nonempty set of* initial states*;*
- *E and H are two finite sets of* external *and* internal *actions, disjoint from each other. We define $A = E \cup H$;*
- *$f \in H$ is a distinguished* fault *action;*
- *$D \subseteq Q \times A \times Q$ is the set of* discrete transitions *respecting the following properties:*
  - **D1** *for every $\mathbf{x} \in Q_n$, there exists $\mathbf{x}' \in Q_f$ such that $(\mathbf{x}, f, \mathbf{x}') \in D$;*
  - **D2** *for every $(\mathbf{x}, f, \mathbf{x}') \in D$, $\mathbf{x} \in Q_n$ and $\mathbf{x}' \in Q_f$;*
  - **D3** *for every $(\mathbf{x}, a, \mathbf{x}') \in D$ such that $a \neq f$, $\mathbf{x} \in Q_f$ iff $\mathbf{x}' \in Q_f$;*
- *$\mathcal{T}$ is a set of trajectories on V. Let $\tau.fstate = \tau.fval|X$ and $\tau.lstate = \tau.lval|X$, if τ closed: we require $\mathcal{T}$ to respect the following properties:*
  - **T1** faulty state invariance*: for every τ, either $\tau(t)|X \in Q_f$ for every $t \in \mathrm{dom}(\mathcal{T})$, or $\tau(t)|X \in Q_n$ for every $t \in \mathrm{dom}(\mathcal{T})$;*
  - **T2** prefix closure*: for every τ' prefix of τ, $\tau' \in \mathcal{T}$;*
  - **T3** suffix closure*: for every τ' suffix of τ, $\tau' \in \mathcal{T}$;*
  - **T4** concatenation closure*: for every (possibly infinite) sequence of trajectories $\tau_0, \tau_1, \tau_2, \ldots \in \mathcal{T}$ such that $\tau_i.lstate = \tau_{i+1}.fstate$, the concatenation $\tau_0 \cdot \tau_1 \cdot \tau_2 \cdot \ldots \in \mathcal{T}$;*

Condition **D1** implies that a fault can occur at any time of the evolution. Conditions **D2** and **D3** implies that the only discrete action that can switch between non-faulty and faulty states is the fault action f, while condition **T1** implies that trajectories cannot switch between faulty and non-faulty states. Conditions **T2**, **T3**, and **T4** express some natural closure properties on $\mathcal{T}$.

Notice that, following the same approach as Lynch, Segala, and Vandraager, we have defined the state of a Hybrid Automaton with Faults to depend only on the values of the internal variables X. However, the choice of the set of trajectories $\mathcal{T}$ can constrain the admissible values for the external variables in W. For this reason, we define the set of *extended states* as $S = \{\mathbf{v} \in \mathrm{Val}(V) | \exists \tau \in \mathcal{T} \text{ s.t. } \tau.fval = \mathbf{v}\}$. By **T1** we have that $S|X = Q$, and thus the definition of extended states is sound. The set of *faulty extended states* S_f and the set of *non-faulty extended states* S_n can be defined in a similar way.

Given a set of variables V and a set of actions A, a (V,A)-*sequence* is a possibly infinite sequence $\alpha = \tau_0 a_1 \tau_1 a_2 \tau_2 \ldots$ such that

1. τ_i is a trajectory on V, for every $i \geqslant 0$,
2. a_i is an action in A, for every $i \geqslant 0$,
3. if α is finite then it ends with a trajectory, and
4. if τ_i is not the last trajectory of α, then $\mathrm{dom}(\tau_i)$ is right-closed.

If $V' \subseteq V$ and $A' \subseteq A$, then the (V',A')-*restriction* of α (denoted $\alpha|(V',A')$ is the (V',A')-sequence obtained by first projecting all trajectories of α on the variables in V', then removing the actions not in A', and finally concatenating all adjacent trajectories. (V,A)-sequences are used to give the semantics of Hybrid Automata in terms of *executions* and *traces*.

Definition 2.2. *An* execution *of a Hybrid Automaton $\mathcal{A}$ from a state $\mathbf{x} \in Q$ is a (V,A)-sequence $\alpha = \tau_0 a_0 \tau_1 a_1 \tau_2 a_2 \ldots$ such that:*

1. *every τ_i is a trajectory in $\mathcal{T}$;*
2. *$\tau_0.fstate = \mathbf{x}$;*
3. *if τ_i is not the last trajectory in α, then $\tau_i.lstate \xrightarrow{a} \tau_{i+1}.fstate$, with $a \in A$.*

The corresponding trace, *denoted trace(α), is the restriction of α to external variables and external actions.*

We say that an execution $\alpha = \tau_0 a_0 \tau_1 a_1 \tau_2 a_2 \ldots$ is *faulty* if for some $i \geqslant 0$, $a_i = f$. An execution α is *maximal* if it starts from a state in Θ and either it is infinite or its last trajectory τ_n is such that *(i)* there exists no trajectory $\tau' \in \mathcal{T}$ such that τ_n is a prefix of τ', and *(ii)* there exists no discrete transition $(\mathbf{x}, a, \mathbf{x}')$ with $\mathbf{x} = \tau_n.lstate$. Moreover, we say that an execution α is *progressive* if it is infinite and it contains an infinite number of occurrences of external actions. Given a Hybrid Automaton $\mathcal{A}$, we denote by *Exec*$(\mathcal{A})$ the set of all maximal execution of $\mathcal{A}$, and by *Traces*$(\mathcal{A})$ the set of all maximal traces of $\mathcal{A}$, that is, the set $\{trace(\alpha) : \alpha \in Exec(\mathcal{A})\}$. $\mathcal{A}$ is *progressive* if all executions in *Exec*$(\mathcal{A})$ are progressive.

We say that a hybrid automaton with faults is *diagnosable* if (maximal) faulty executions can be distinguished from non-faulty ones by looking at the corresponding traces.

Definition 2.3 (Diagnosability)**.** *We say that a Hybrid Automaton with Faults $\mathcal{A} = \langle W, X, Q, Q_f, \Theta, E, H, f, D, \mathcal{T} \rangle$ is* diagnosable *if for any two maximal executions $\alpha_1, \alpha_2 \in Exec(\mathcal{A})$, if α_1 is faulty then either α_2 is faulty or $trace(\alpha_1) \neq trace(\alpha_2)$.*

The above definition of diagnosability is very general, and can be applied to a large class of faults, involving both the continuous and the discrete dynamics of the system. However, solving the fault-diagnosis problem can be very complex, if not impossible at all, under this definition.

In this paper we consider a weaker notion of diagnosability, that we call *time-abstract diagnosability*, for which the fault-diagnosis problem can be solved in a simpler way, leaving the treatment of the stronger diagnosability notion for a subsequent paper. We assume the system to be progressive, and we define the diagnoser as some kind of finite-state digital device, that monitors the evolution of the system by reacting to external actions and by measuring the values of external variables with a fixed and finite precision. We formally define the latter restriction by introducing the notion of *observation* for the external variables.

Definition 2.4. *Given the set of external variables W of a hybrid automaton with faults $\mathcal{A}$, an* observation *of W is any finite partition $\mathcal{O} = \{O_1, \ldots, O_2\}$ of* Val(W). *We call the elements O_i of the partition* observables *for W.*

In this setting, we say that a progressive system is time-abstract diagnosable if faults can be determined only by looking at the observables and at the occurences of external discrete actions, without considering the delays and the trajectories between them. To formally define such a notion, we first need to define *untimed observation traces* for hybrid automata.

Definition 2.5. *Given a trace $\beta = \tau_0 a_0 \tau_1 a_1 \tau_2 a_2 \ldots$ of a Hybrid Automaton $\mathcal{A}$, and an observation $\mathcal{O}$ for W, we define the corresponding* untimed observation trace *as the sequence $untime(\beta) = O_0 a_0 O_1 a_1 O_2 a_2 \ldots$ such that $\tau_i.fval \in O_i$ for each $i \geqslant 0$. Given an execution α of $\mathcal{A}$, we define $utrace(\alpha) = untime(trace(\alpha))$.*

Definition 2.6 (Time-abstract diagnosability)**.** *We say that a Hybrid Automaton with Faults $\mathcal{A} = \langle W, X, Q, Q_f, \Theta, E, H, f, D, \mathcal{T} \rangle$ is* time-abstract diagnosable *if it is progressive and, for any two maximal executions $\alpha_1, \alpha_2 \in Exec(\mathcal{A})$, if α_1 is faulty then either α_2 is faulty or $utrace(\alpha_1) \neq utrace(\alpha_2)$.*

Since $utrace(\alpha_1) \neq utrace(\alpha_2)$ implies that $trace(\alpha_1) \neq trace(\alpha_2)$, a hybrid automaton that is time-abstract diagnosable is also diagnosable, but the converse does not necessarily hold. Indeed, any fault that do not change the sequence of discrete actions performed by the system, but only the delays or the continuous trajectories between them is not time-abstract diagnosable.

3 The Fault Detection Game

In this section we introduce the key notion of *Fault Detection Game* (for time-abstract fault diagnosis), played on a Hybrid Automaton with Faults $\mathcal{A}$ by two players, the *Environment* and the *Diagnoser*. A *position* in the game is a pair $(\mathbf{v}, d) \in \mathrm{Val}(V) \times \{yes, no\}$, such that $\mathbf{v}$ is an extended state of $\mathcal{A}$. Given a current position $(\mathbf{v}, d)$, we distinguish between the following kind of moves:

1. **Diagnoser move**: the Diagnoser chooses an answer $d' \in \{yes, no\}$. The game continues from position $(\mathbf{v}, d')$ with an Environment move, and we denote this by $(\mathbf{v}, d) \xrightarrow{d'} (\mathbf{v}, d')$.
2. **Environment move**: the Environment chooses one of the following possible moves
   (a) two valuations $\mathbf{v}', \mathbf{v}'' \in \mathrm{Val}(V)$, a trajectory $\tau \in \mathcal{T}$, and an external action $e \in E$ such that $\tau.fval = \mathbf{v}$, $\tau.lval = \mathbf{v}''$, and $\mathbf{v}''|X \xrightarrow{e} \mathbf{v}'|X$. The game continues from position $(\mathbf{v}', d)$ with a Diagnoser move, and we denote this by $(\mathbf{v}, d) \xrightarrow{e} (\mathbf{v}', d)$;
   (b) two valuations $\mathbf{v}', \mathbf{v}'' \in \mathrm{Val}(V)$, a trajectory $\tau \in \mathcal{T}$, and an internal action $h \in H$ such that $\tau.fval = \mathbf{v}$, $\tau.lval = \mathbf{v}''$, and $\mathbf{v}''|X \xrightarrow{h} \mathbf{v}'|X$. The game continues from position $(\mathbf{v}', d)$ with an Environment move, and we denote this by $(\mathbf{v}, d) \xrightarrow{h} (\mathbf{v}', d)$.

Notice that the Fault Detection Game is is asymmetric: in our framework the environment is more powerful than the diagnoser, since it can choose the continuous trajectory to follow and prevent the diagnoser to move by choosing an internal action. Moreover, the game is also under partial observability: as formally stated in the following, the diagnoser is blind to the value of internal variables and to the occurrence of internal events.

Definition 3.1 (Run of the Fault Detection Game)**.** *A* run *of the game is an infinite sequence* $\rho = (\mathbf{v}_0, d_0) \xrightarrow{m_1} (\mathbf{v}_1, d_1) \xrightarrow{m_2} \ldots$ *such that:*

1. $d_0 = no$,
2. m_1 *is a diagnoser move,*
3. *for every* $i \geqslant 1$, $(\mathbf{v}_{i-1}, d_{i-1}) \xrightarrow{m_i} (\mathbf{v}_{i-1}, d_{i-1})$ *is a valid move of the game;*
4. *for every* $i > 1$, m_i *is a diagnoser move if and only if* m_{i-1} *is an environment move with* $m_{i-1} \notin H$.

A run is winning *for the diagnoser if one of the two conditions hold:*

- *either for each* $i \geqslant 1$, $m_i \neq f$ *and, for each* $j \geqslant 1$, $d_j = no$, *or*
- *there exists* $i \geqslant 1$ *such that* $m_i = f$ *and* $j > i$ *such that* $d_j = yes$.

Given an observation $\mathcal{O}$ for the external variables, the corresponding *observation* of a run ρ is a sequence $obs(\rho) = (O_0, d_0) \xrightarrow{m_1} (O_1, d_1) \xrightarrow{m_2} \ldots$ obtained from ρ by replacing every maximal sequence of environment moves $(\mathbf{v}_j, d_j) \xrightarrow{m_{j+1}} \ldots \xrightarrow{m_{j+k}} (\mathbf{v}_{j+k}, d_{j+k})$ with $(\mathbf{v}_j, d_j) \xrightarrow{m_{j+k}} (\mathbf{v}_{j+k}, d_{j+k})$ and by restricting every position $(\mathbf{v}_j, d_j)$ to (O_j, d_j), where O_j is the unique observable such that $\mathbf{v}_j|W \in O_j$. We denote by $Obs_f(\mathcal{A})$ the set of finite observations for the Fault Detection Game played on $\mathcal{A}$. A strategy is a function that tells the Diagnoser which move to choose given a finite observation.

Definition 3.2. *A* strategy *is a partial function* λ *from* $Obs_f(\mathcal{A})$ *to* $\{yes, no\}$.

The strategy tells the diagnoser what answer to give at the current moment. Let ρ be a run of the game, $\sigma = obs(\rho)$ and let $\sigma_i = (O_0, d_0) \xrightarrow{m_1} \ldots \xrightarrow{m_i} (O_i, d_i) \ldots$ be the prefix of σ of length i. We say that ρ is consistent with the strategy λ when, for all i, if $\lambda(\sigma_i) = d$ then either $m_{i+1} = d$ or m_{i+1} is an environment move. A strategy λ is *winning from a state* $\mathbf{x} \in Q$ if for all $\mathbf{v}$ such that $\mathbf{v}|X = \mathbf{x}$, all runs starting in $(\mathbf{v}, no)$ compatible with λ are winning. The set of *winning states* is the set of states from which there is a winning strategy.

We can now define the fault diagnosis problems we will study.

Definition 3.3 (Time-abstract Diagnosability in a class $\mathfrak{C}$ of automata)**.** *Given a hybrid automaton with faults $\mathcal{A} \in \mathfrak{C}$, determine whether there exists a winning strategy in the Fault Detection Game played on $\mathcal{A}$ from the initial states Θ.*

Definition 3.4 (Time-abstract Diagnoser synthesis in a class $\mathfrak{C}$ of automata)**.** *Given a hybrid automaton with faults $\mathcal{A} \in \mathfrak{C}$, determine whether there exists a winning strategy in the Fault Detection Game played on $\mathcal{A}$ from the initial states Θ, and compute such a strategy if possible.*

4 Computing Strategies

In this section we will show how to solve the Time-abstract Diagnosability and the Time-abstract Diagnoser synthesis problems for some relevant classes of hybrid automata, exploiting the notion of bisimulation. Such a key notion has been introduced in many fields with different purposes (for instance, van Benthem proposed it as an equivalence principle between structures [5]). In our setting, we use bisimulation as an equivalence principle between states of a hybrid automaton. Roughly speaking, two extended states $\mathbf{v}$ and $\mathbf{v}'$ are *bisimilar* if every behaviour that starts from $\mathbf{v}$ can be matched by starting from $\mathbf{v}'$ and vice versa.

Definition 4.1 (Time-abstract bisimulation)**.** *Given a Hybrid Automaton with Faults $\mathcal{A} = \langle W, X, Q, Q_f, \Theta, E, H, f, D, \mathcal{T} \rangle$, a* time-abstract bismulation *is an equivalence relation $\sim \subseteq S \times S$ such that for every $\mathbf{v}_1, \mathbf{v}_1', \mathbf{v}_2 \in S$, the following two conditions are satisfied:*

$$\forall a \in A, \left(\mathbf{v}_1 \sim \mathbf{v}_1' \text{ and } \mathbf{v}_1|X \xrightarrow{a} \mathbf{v}_2|X\right) \Rightarrow \left(\exists \mathbf{v}_2' \in S \text{ s.t. } \mathbf{v}_2 \sim \mathbf{v}_2' \text{ and } \mathbf{v}_1' \xrightarrow{a} \mathbf{v}_2'\right), \text{ and}$$

$$\forall \tau \in \mathcal{T}, \left(\mathbf{v}_1 \sim \mathbf{v}_1' \text{ and } \mathbf{v}_1 = \tau.fval \text{ and } \mathbf{v}_2 = \tau.lval\right) \Rightarrow \left(\exists \tau' \in \mathcal{T}, \mathbf{v}_2' \in S \text{ s.t. } \mathbf{v}_2 \sim \mathbf{v}_2' \text{ and } \tau'.fval = \mathbf{v}_1' \text{ and } \mathbf{v}_2' = \tau.lval\right).$$

Given a hybrid automaton $\mathcal{A}$ and a time-abstract bisimulation $\sim \subseteq S \times S$, we say that two extended states $\mathbf{v}, \mathbf{v}' \in S$ are *bisimilar* if and only if $\mathbf{v} \sim \mathbf{v}'$. The *equivalence class of* $\mathbf{v}$, denoted by $[\![\mathbf{v}]\!]_\sim$ is defined as the set $[\![\mathbf{v}]\!]_\sim = \{\mathbf{v}' \in S | \mathbf{v}' \sim \mathbf{v}\}$ (in the following, we will omit the $_\sim$ subscript when clear from the context). A time-abstract bisimulation naturally induces a partition of S into equivalence classes, called *bisimulation quotient of* $\mathcal{A}$.

Definition 4.2 (Bisimulation quotient)**.** *Given a Hybrid Automaton with Faults $\mathcal{A}$ and a time-abstract bisimulation $\sim \subseteq S \times S$, the* bisimulation quotient of $\mathcal{A}$ under $\sim$ *is defined as the set $S/_\sim = \{[\![\mathbf{v}]\!]_\sim | \mathbf{v} \in S\}$.*

A bisimulation $\sim$ has *finite index* if the number of equivalence classes in $S/_\sim$ is finite, and of *infinite index* otherwise. We say that a class $\mathfrak{C}$ of hybrid automata *admits a bisimulation with finite quotient* if for every $\mathcal{A} \in \mathfrak{C}$ there exists a time-abstract bisimulation $\sim$ with finite index. We say that such quotient can be *effectively computed* if there exists an algorithm that can compute $\sim$ and $S/_\sim$ for every $\mathcal{A} \in \mathfrak{C}$. In

the following we concentrate our attention on the classes of hybrid automata that admits a bisimulation with finite quotient that can be effectively computed, and we will prove that the Diagnosability and the Diagnoser synthesis problems are decidable in this case.

In the case of hybrid automata with faults, we have that the equivalence classes of a bisimulation respect the partition between faulty and non-faulty states, as formally proved by the following lemma. In the following, we denote with $S_f/_\sim$ the set of equivalence classes of the faulty extended states of $\mathcal{A}$, and with $S_n/_\sim$ the set of equivalence classes of the non-faulty extended states of the automaton.

Lemma 4.3. *Given a hybrid automaton with faults $\mathcal{A}$ and a time-abstract bisimulation $\sim \subseteq S \times S$, we have that for every $\mathbf{v} \in S_n$ and $\mathbf{v}' \in S_f$, $\mathbf{v} \not\sim \mathbf{v}'$.*

Proof. Suppose by contradiction that there exists $\mathbf{v}_1 \in S_n$ and $\mathbf{v}'_1 \in S_f$ such that $\mathbf{v}_1 \sim \mathbf{v}'_1$. By **D1** we have that there must exists $\mathbf{v}_2 \in S_f$ such that $(\mathbf{v}_1|X, f, \mathbf{v}_2|X) \in D$. By the definition of bisimulation, this implies that there exists $\mathbf{v}'_2 \in S$ such that $(\mathbf{v}'_1|X, f, \mathbf{v}'_2|X) \in D$, in contradiction with **D2**, since $\mathbf{v}'_1|X \in Q_f$. □

Given an observation $\mathcal{O}$ of $\mathrm{Val}(W)$, we say that a bisimulation $\sim \subseteq S \times S$ *respects* $\mathcal{O}$ if for every $\mathbf{v}, \mathbf{v}' \in S$, $\mathbf{v} \sim \mathbf{v}'$ implies that $\mathbf{v}|W$ and $\mathbf{v}'|W$ belong to the same observable of $\mathcal{O}$. From now on we assume that $\sim$ respects the observation of external variables.

We are now ready to define the key notion of *state estimator* of a hybrid automaton with faults. Intuitively, a state estimator is a finite automaton that given an untimed observation trace β of $\mathcal{A}$, provides the set of states that can be reached by $\mathcal{A}$ under all possible executions compatible with β.

Definition 4.4 (State estimator)**.** *Given a hybrid automaton with faults $\mathcal{A} = \langle W, X, Q, Q_f, \Theta, E, H, f, D, \mathcal{T}\rangle$, an observation $\mathcal{O}$ for the external variables, and a bisimulation with finite index $\sim \subseteq S \times S$ that respects $\mathcal{O}$, we define the* state estimator of $\mathcal{A}$ *as the transition system $\mathcal{E} = \langle 2^{Q/_\sim}, \Pi, \Delta\rangle$ such that:*

E1 *$2^{S/_\sim}$ is the powerset of $S/_\sim$;*

E2 *$\Pi \subseteq 2^{S/_\sim}$ is the set of initial states defined as*
$$\Pi = \{\mathcal{S} \in 2^{S/_\sim} | \exists O \in \mathcal{O} \text{ s.t. } \forall \mathbf{v} \in S, (\mathbf{v}|X \in \Theta \wedge \mathbf{v}|W \in O) \Rightarrow [\![\mathbf{v}]\!] \in \mathcal{S}\};$$

E3 *$\Delta : 2^{S/_\sim} \times A \times \mathcal{O} \mapsto 2^{S/_\sim}$ is the transition function such that $\Delta(\mathcal{S}, a, O) = \mathcal{S}'$ iff for all finite executions $\alpha = \tau_0 a_0 \ldots a_n \tau_n$ of $\mathcal{A}$,*
$$(a_n = a \wedge [\![\tau_0.fval]\!] \in \mathcal{S} \wedge utrace(\alpha) = O_0 a O) \Rightarrow [\![\tau_n.fval]\!] \in \mathcal{S}'.$$

The state estimator is a deterministic automaton, since the transition function associate a unique successor state to every pair of input symbols (a, O). Hence, with a little abuse of notation, we can define the function Δ on untimed observation traces as follows. Given an untimed observation trace $\beta = O_0 a_0 O_1 a_1 \ldots$ and a state $\mathcal{S} \in 2^{S/_\sim}$, we define $\Delta(\mathcal{S}, \beta)$ is the sequence of estimator states $\mathcal{S}_0 \mathcal{S}_1 \ldots$ such that *(i)* $\mathcal{S}_0 = \mathcal{S}$, and *(ii)* $\mathcal{S}_i = \Delta(S_{i-1}, a_{i-1}, O_i)$ for all $i \geqslant 1$. Moreover, we define $\Delta(\beta) = \Delta(\mathcal{S}_0, \beta)$, where $\mathcal{S}_0$ is the unique state in Π such that $\mathcal{S}_0 = \{[\![\mathbf{v}]\!] \in S/_\sim \text{ s.t. } \mathbf{v}|X \in \Theta \text{ and } \mathbf{v}|W \in O_0\}$. The following lemma proves that the state estimator is correctly defined, and can be seen ad a consequence of the fact that time-abstract bisimulation preserves traces.

Lemma 4.5. *Given a hybrid automaton with faults $\mathcal{A}$, and a state estimator $\mathcal{E}$ for it, let β be a finite untimed observation trace of $\mathcal{A}$, and $\Delta(\beta) = \mathcal{S}_0 \mathcal{S}_1 \ldots \mathcal{S}_n$. Then, for every $[\![\mathbf{v}]\!] \in S/_\sim$, $[\![\mathbf{v}]\!] \in \mathcal{S}_n$ if and only if there exists a finite execution $\alpha = \tau_0 a_0 \tau_1 a_1 \ldots a_{m-1} \tau_m$ such that $utrace(\alpha) = \beta$, $\tau_0.fstate \in \Theta$, and $\tau_m.fval \in [\![\mathbf{v}]\!]$.*

Proof. Let $\beta = O_0 a_0 O_1 a_1 \ldots a_{n-1} O_n$ be a finite untimed observation trace of $\mathcal{A}$. We prove the lemma by induction on the length of β.

If $n = 0$ then $\beta = O_0$ and the claim trivially follows from the definition of $\Delta(\beta)$.

If $n > 0$, let $\beta_{n-1} = O_0 a_0 O_1 a_1 \dots a_{n-2} O_{n-1}$, and suppose by inductive hypothesis that the claim holds for β_{n-1}. Now, let $\alpha = \tau_0 a_0 \dots a_{m-1} \tau_m$ an execution of $\mathcal{A}$ such that $utrace(\alpha) = \beta$ and $\tau_0.fstate \in \Theta$. By the definition of untimed execution trace, let $\alpha_{n-1} = \tau_0 a_0 \dots a_{l-1} \tau_l$ be the prefix of α such that $utrace(\alpha_{n-1}) = \beta_{n-1}$, and let $\Delta(\beta_{n-1}) = \mathcal{S}_0 \dots \mathcal{S}_{n-1}$. By inductive hypothesis, we have that $[\![\tau_l.fval]\!] \in \mathcal{S}_{n-1}$. Consider now the finite execution $\alpha' = \tau_l a_l \dots a_{m-1} \tau_m$ such that $\alpha = \alpha_{n-1} \alpha'$. By the definition of untimed observation trace, we have that $utrace(\alpha') = O_{n-1} a_{n-1} O_n$ and thus, by the definition of Δ, that $[\![\tau_m.fval]\!] \in \Delta(\mathcal{S}_{n-1}, a_{n-1}, O_n) = \mathcal{S}_n$. To prove the converse implication, let $[\![\mathbf{v}]\!] \in \mathcal{S}_n$. By definition of Δ, this implies that there exists a finite execution $\gamma = \tau_0 a_0 \dots a_{m-1} \tau_m$ such that $[\![\tau_0.fval]\!] \in \mathcal{S}_{n-1}$, $utrace(\gamma) = O_{n-1} a_{n-1} O_n$, and $\tau_m.fval \in [\![\mathbf{v}]\!]$. By inductive hypothesis, we have that there exists a finite execution $\gamma' = \tau'_0 a'_0 \dots a'_{l-1} \tau'_l$ such that $utrace(\gamma') = \beta_{n-1}$, $\tau'_0.fstate \in \Theta$, and $\tau'_l.fval \in \mathcal{S}_{n-1}$. Hence, the finite execution $\zeta = \tau'_0 a'_0 \dots a'_{l-1} \tau_0 a_0 \dots a_{m-1} \tau_m$ is a valid execution of $\mathcal{A}$ respecting the desired properties. □

Given the partition of the equivalence classes in $S/_\sim$ between faulty and non-faulty ones, we can distinguish between three different kinds of states $\mathcal{S} \in 2^{S/_\sim}$ of the state estimator:

faulty states such that $\mathcal{S} \subseteq S_f/_\sim$,
non-faulty states such that $\mathcal{S} \subseteq S_n/_\sim$, and
indeterminate states that contains both faulty and non-faulty equivalence classes.

It turns out that there exists a winning strategy for the diagnoser on the Fault Detection Game played on $\mathcal{A}$ if and only if there are no loops of indeterminate states reachable from the initial states of the estimator.

Theorem 4.6. *Given a hybrid automaton with faults $\mathcal{A}$, an observation $\mathcal{O}$ for the external variables, and a bisimulation with finite index $\sim \subseteq S \times S$ that respects $\mathcal{O}$, we have that there exists a winning strategy for the diagnoser in the Fault Detection Game played on $\mathcal{A}$ from the initial sates Θ if and only if there are no loops of indeterminate states reachable from the initial states Π of the state estimator for $\mathcal{A}$.*

Proof. Let $\mathcal{E} = \langle 2^{Q/_\sim}, \Pi, \Delta \rangle$ be the state estimator for $\mathcal{A}$, and suppose that there are no loops of indeterminate states reachable from the initial states Π. Then we show how to define a winning strategy for the diagnoser in the Fault Detection Game played on $\mathcal{A}$ from the initial states Θ. Given a finite observation for the fault diagnosis game $\sigma = (O_0, d_0) \xrightarrow{m_1} (O_1, d_1) \xrightarrow{m_2} \dots \xrightarrow{m_n} (O_n, d_n)$, we define the corresponding untimed observation trace $utrace(\sigma) = O_0 a_0 \dots a_{l-1} O_l$ by removing all diagnoser moves and ignoring the d_i component of the positions. Let $\Delta(O_0 a_0 \dots a_{l-1} O_l) = \mathcal{S}_0 \dots \mathcal{S}_l$. We define the strategy λ on σ as follows:

$$\lambda(\sigma) = \begin{cases} yes & \text{if } \mathcal{S}_l \text{ is a faulty state of } \mathcal{E} \\ no & \text{otherwise} \end{cases}$$

Now, let $\rho = (\mathbf{v}_0, d_0) \xrightarrow{m_1} (\mathbf{v}_1, d_1) \xrightarrow{m_2} \dots$ be an infinite run of the game compatible with λ, let $\alpha = utrace(obs(\rho)) = O_0 a_1 O_1 a_2 \dots$ be the corresponding infinite untimed observation trace, and let $\Delta(\alpha) = \mathcal{S}_0 \mathcal{S}_1 \mathcal{S}_2 \dots$. Two cases may arise:

- ρ is faulty. Since there are no loops of indeterminate states in $\mathcal{E}$, from Lemma 4.5 we can conclude that there exists $i \geqslant 0$ such that for every $j \geqslant i$ $\mathcal{S}_j$ is a faulty state of the estimator. Hence, the strategy λ is such that there exists k such that $m_k = yes$, and thus ρ is winning for the diagnoser.
- ρ is non-faulty. From Lemma 4.5 we can conclude that all $\mathcal{S}_i$ are either non-faulty or indeterminate. Hence, the strategy λ is such that $m_i = no$ for every diagnoser move, and ρ is winning for the diagnoser.

In both cases the diagnoser wins the game, so we can conclude that λ is a winning strategy for the diagnoser in the Fault Detection Game on $\mathcal{A}$ from the initial states Θ.

Conversely, suppose that there exists a loop of indeterminate states reachable from Π in $\mathcal{E}$. This implies that there exist an indeterminate state $\mathcal{S}$ and two time-abstract observation traces $\alpha = O_0 a_0 \dots a_{n-1} O_n$ and $\beta = O_n a_n \dots a_{m-1} O_m$ such that:

1. $\Delta(\mathcal{S}_0, \alpha) = \mathcal{S}_0 \dots \mathcal{S}_n$ is such that $\mathcal{S}_0 \in \Pi$ and $\mathcal{S}_n = \mathcal{S}$, and
2. $\Delta(\mathcal{S}_n, \beta) = \mathcal{S}_n \dots \mathcal{S}_m$ is such that $\mathcal{S}_n = \mathcal{S}_m = \mathcal{S}$ and $\mathcal{S}_i$ is an indeterminate state for each $n \leqslant i \leqslant m$.

Now, suppose by contradiction that there exists a winning strategy λ for the diagnoser, and consider the infinite time-abstract observation trace $\gamma = \alpha\beta\beta\beta\dots$. Two cases may arise:

- For every finite prefix γ_i of γ, $\lambda(\gamma_i) = no$. By Lemma 4.5, since every state in β contains a faulty equivalence class, we have that there exists a faulty execution α of $\mathcal{A}$ such that $utrace(\alpha) = \gamma$. This implies that it is possible to build an infinite faulty run of the game that is winning for the environment, against the hypothesis that λ is winning for the diagnoser.
- There exists a finite prefix γ_i of γ such that $\lambda(\gamma_i) = yes$. By Lemma 4.5, since every state in β contains a non-faulty equivalence class, we have that there exists a non-faulty execution α_i of $\mathcal{A}$ such that $utrace(\alpha_i) = \gamma_i$. This implies that it is possible to build a run of the game that is winning for the environment, against the hypothesis that λ is winning for the diagnoser.

In both cases a contradiction is found, and the thesis is proved. □

Let $\mathfrak{T}$ be a logical theory. If all the components of a hybrid automaton with faults $\mathcal{A}$ are definable in $\mathfrak{T}$, we say that $\mathcal{A}$ is *definable in* $\mathfrak{T}$. Moreover, a class of hybrid automata with faults $\mathfrak{C}$ is *definable in* $\mathfrak{T}$ if every $\mathcal{A} \in \mathfrak{C}$ is definable in $\mathfrak{T}$. The previous theorems shows that the state estimator can be used to define a winning strategy for the diagnoser in the Fault Detection Game. However, it does necessarily implies that we can compute such a strategy, since the theory used to define the automata is not necessarily decidable. Moreover, even when $\mathfrak{T}$ is decidable it is not guaranteed that a bisimulation with finite quotient that can be effectively computed. The following theorem states that if some conditions on the considered theory and on the observation of external variables are respected, then Theorem 4.6 provides an algorithmic solution to the diagnosability and the diagnoser synthesis problem.

Theorem 4.7. *Let $\mathfrak{T}$ be a decidable theory. Let $\mathfrak{C}$ be a class of Hybrid Automata with Faults that can be defined in $\mathfrak{T}$ and such that for every $\mathcal{A}$ in $\mathfrak{C}$, there exists a bisimulation with finite quotient $\sim$ that can be effectively computed. Then the time-abstract diagnosability problem in the class $\mathfrak{C}$ is decidable for every observation $\mathcal{O}$ definable in $\mathfrak{T}$. Moreover, a winning strategy for the diagnoser can be computed, if possible.*

Proof. To prove that that both the time-abstract diagnosability and the time-abstract diagnoser synthesis problems are decidable we have to show how to compute a state estimator $\mathcal{E}$ for the automaton $\mathcal{A}$.

First of all, let $\mathcal{O}$ be a definable observation for the external variables, and let $\sim$ a bisimulation with finite quotient for $\mathcal{A}$. In general, it is not guaranteed that $\sim$ respects $\mathcal{O}$. However, since $\mathcal{O}$ is definable in $\mathfrak{T}$, and $\mathfrak{T}$ is decidable, we can always refine $\sim$ to a finer bisimulation $\approx$ respecting $\mathcal{O}$ by using the bisimulation algorithm given in [10, 14]. Since both $\mathcal{O}$ and $S/_{\approx}$ are finite sets, to prove that $\mathcal{E}$ can be effectively computed it is sufficient to prove that the the transition relation Δ is computable. Given a state $\mathcal{S}$ of the estimator, an action $a \in E$, and an observable $O \in \mathcal{O}$, computing the successor state $\Delta(\mathcal{S}, a, O)$ can be reduced to a reachability problem on $\mathcal{A}$. Since it is known that reachability is decidable for all classes of hybrid automata for which there exists a bisimulation with finite quotient that can be effectively computed, then Δ is computable and there exists an algorithm that can build the state estimator for $\mathcal{A}$.

Once that the state estimator $\mathcal{E}$ has been built, we can use it for solving both the time-abstract diagnosability and the time-abstract diagnoser synthesis problems as follows.

- From Theorem 4.6 we know that there exists a winning strategy for the diagnoser in the Fault Detection Game if and only if there are no loops of indeterminate states in $\mathcal{E}$. Since the state estimator is a finite automaton, existence of such loops can be determined by computing a depth-first visit of $\mathcal{E}$, and thus the time-abstract diagnosability problem is decidable.
- The proof of Theorem 4.6 shows how the state estimator can be used to define a winning strategy for the diagnoser in the Fault Detection Game. Since the the state estimator can be effectively computed, we have that such a strategy can be computed.

Hence, both problems are decidable under the considered assumptions. □

This decidability results is very general: examples of classes of hybrid automata that respects the conditions of Theorem 4.7 are Timed Automata [3], Simple Multirate Automata [2], O-minimal Hybrid Automata [10, 20], and STORMED Hybrid Automata [31]. Hence, for all such classes of systems, the time-abstract diagnosability problem and the time-abstract diagnoser synthesis problem is decidable. Moreover, the discovery of more classes of hybrid automata respecting the conditions of the theorem immediately leads to new classes of systems for which the two fault-diagnosis problems considered in this paper are decidable.

The complexity of the two problems depends on the size of the bisimulation quotient $S/_\sim$: if n is the number of equivalence classes, then the size of the state estimator $\mathcal{E}$ is exponential in n. Since computing a depth-first visit on a finite transition system is in LOGSPACE, we have that the time-abstract diagnosability problem is solvable with polynomial space w.r.t. n. Theorem 4.7 proves that solving the time-abstract diagnoser synthesis problem corresponds to compute the state estimator $\mathcal{E}$ for the considered system. Hence, this second problem can be solved using an exponential amount of time w.r.t. n.

It is worth to notice that for most classes of hybrid automata, like Timed Automata, Initialized Rectangular Automata, and of o-minimal systems, like Pfaffian Hybrid Automata, the number of equivalence classes in $S/_\sim$ is exponential in the size of the automaton. Hence, for those classes the time-abstract diagnosability problem is in EXPSPACE and the time-abstract diagnoser synthesis problem is in 2-EXPTIME.

5 Conclusions

In this paper we studied the fault-diagnosis problem for hybrid systems from a game-theoretical point of view. We used the formalism of hybrid automata for modeling hybrid systems with faults and to define the notions of diagnosability and time-abstract diagnosability. We focused our attention on time-abstract diagnosability and we defined a Fault Diagnosis Game on hybrid automata with faults between two players, the environment and the diagnoser. Existence of a winning strategy for the diagnoser implies that faults can be identified correctly, while computing such a winning strategy corresponds to implementing a diagnoser for the system. Finally, we shown how to determine the existence of a winning strategy, and how to compute it, for all classes of hybrid automata definable in a decidable theory $\mathfrak{T}$ and such that a bisimulation with finite quotient can be effectively computed, like timed automata and o-minimal hybrid automata.

The results presented in the paper can be extended in many directions. First of all, by considering the stronger notion of diagnosability instead of time-abstract diagnosability. Then, by extending the results also to undecidable classes of hybrid automata, by exploiting abstraction refinement and approximation techniques. Finally, in the current framework there is no upper bound on the time that elapses between

the occurrence of the fault and the detection by the diagnoser. We envision the extension of our approach to reward and priced hybrid games [1, 8] as a possible way to provide minimal-delay strategies for the diagnoser.

References

[1] B. Adler, L. de Alfaro & M. Faella (2005): *Average Reward Timed Games*. In: *Formal Modeling and Analysis of Timed Systems, Lecture Notes in Computer Science* 3829, Springer Berlin / Heidelberg, pp. 65–80, doi:10.1007/11603009_6.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. h. Ho, X. Nicollin, A. Olivero, J. Sifakis & S. Yovine (1995): *The Algorithmic Analysis of Hybrid Systems*. *Theoretical Computer Science* 138, pp. 3–34, doi:10.1016/0304-3975(94)00202-T.

[3] R. Alur & D. L. Dill (1994): *A theory of timed automata*. *Theoretical Computer Science* 126(2), pp. 183 – 235.

[4] E. Asarin, O. Maler, A. Pnueli & J. Sifakis (1998): *Controller Synthesis For Timed Automata*. In: *Proceedings of the IFAC Symposium on System Structure and Control*, Elsevier Science Publishers, pp. 469–474.

[5] J. van Benthem (1978): *Modal correspondence theory*. Ph.D. thesis, Department of Mathematics, University of Amsterdam, Amsterdam, The Netherlands.

[6] M. Blanke, M. Kinnaert, M. Staroswiecki & J. Lunze (2003): *Diagnosis and fault-tolerant control*. Springer-Verlag.

[7] P. Bouyer, T. Brihaye & F. Chevalier (2010): *O-Minimal Hybrid Reachability Games*. *Logical Methods in Computer Science* 6(1:1), pp. 1–48.

[8] P. Bouyer, T. Brihaye, M. Jurdziński, R. Lazić & M. Rutkowski (2008): *Average-Price and Reachability-Price Games on Hybrid Automata with Strong Resets*. In: *Formal Modeling and Analysis of Timed Systems, Lecture Notes in Computer Science* 5215, Springer Berlin / Heidelberg, pp. 63–77, doi:10.1007/978-3-540-85778-5_6.

[9] P. Bouyer, F. Chevalier & D. D'Souza (2005): *Fault Diagnosis Using Timed Automata*. In: *Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science* 3441, Springer Berlin / Heidelberg, pp. 219–233, doi:10.1007/978-3-540-31982-5_14.

[10] T. Brihaye, C. Michaux, C. Rivière & C. Troestler (2004): *On O-Minimal Hybrid Systems*. In: *Proceedings of Hybrid Systems: Computation and Control (HSCC'04), Lecture Notes in Computer Science* 2993, Springer, pp. 219–233, doi:10.1007/978-3-540-24743-2_15.

[11] L. Console & P. Torasso (1991): *A spectrum of logical definitions of model-based diagnosis*. *Computational Intelligence* 7(3), pp. 133–141, doi:10.1111/j.1467-8640.1991.tb00388.x.

[12] G. Fey, S. Staber, R. Bloem & R. Drechsler (2008): *Automatic Fault Localization for Property Checking*. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 27(6), pp. 1138 –1149.

[13] G.K. Fourlas, K.J. Kyriakopoulos & N.J. Krikelis (2001): *A Framework for Fault Detection of Hybrid Systems*. In: *Proceedings of the IEEE MED 2001 Conference*, Dubrovnik, Croatia.

[14] T. A. Henzinger (1996): *The theory of hybrid automata*. In: *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society, pp. 278–292, doi:10.1109/LICS.1996.561342.

[15] T. A. Henzinger, B. Horowitz & R. Majumdar (1999): *Rectangular Hybrid Games*. In: *Proceedings of the 10th International Conference on Concurrency Theory, LNCS* 1664, Springer-Verlag, pp. 320–335.

[16] D.M. Himmelblau (1978): *Fault detection and diagnosis in chemical and petrochemical processes*. Chemical engineering monographs, Elsevier Scientific Pub.

[17] R. Isermann (1984): *Process Fault Detection Based on Modeling and Estimation Methods - A Survey*. *Automatica* 20(4), pp. 387–404, doi:10.1016/0005-1098(84)90098-0.

[18] B. Jobstmann, S. Staber, A. Griesmayer & R. Bloem (2011): *Finding and Fixing Faults*. *Journal of Computer and System Sciences (JCSS)* Available at `http://www-verimag.imag.fr/~jobstman/bib/files/findingandfixing.pdf`. To appear.

[19] C.N. Jones & J.M. Maciejowski (2005): *Fault Tolerant Flight Control - An Overview*. Cambridge University - Engineering Department, Technical Report.

[20] G. Lafferriere, G. J. Pappas & S. Sastry (2000): *O-Minimal Hybrid Systems*. *Mathematics of Control, Signals, and Systems* 13, pp. 1–21, doi:10.1007/PL00009858.

[21] N. Lynch, R. Segala & F. Vaandrager (2003): *Hybrid I/O automata*. *Information and Computation* 185(1), pp. 105 – 157.

[22] O. Maler, Z. Manna & A. Pnueli (1991): *From Timed to Hybrid Systems*. In J. W. de Bakker, C. Huizing, W. P. de Roever & G. Rozenberg, editors: *Real-Time: Theory in Practice*, 600, Springer-Verlag, pp. 447–484, doi:10.1007/BFb0032003.

[23] D. Mignone, A. Bemporad & M. Morari (1999): *Moving Horizon Estimation for Hybrid Systems and Fault Detection*. In: *Proceedings of the American Control Conference*, San Diego, California, pp. 2471–2475.

[24] R.J. Patton, P.M. Frank & R.N. Clark (2000): *Issues of Fault Diagnosis for Dynamical Systems*. Springer-Verlag.

[25] L.F. Pau (1981): *Failure Diagnosis and Performance Monitoring*. Marcel Dekker.

[26] I.R. Petersen & D.C McFarlane (2004): *A methodology for Robust Fault Detection in Dynamic Systems*. *Control Engineering Practice* 12, pp. 123–138, doi:10.1016/S0967-0661(03)00004-2.

[27] D. Poole, R. Goebel & R. Aleliunas (1987): *Theorist: a logical reasoning system for defaults and diagnosis'*. In N. Cercone & G. McCalla, editors: *The Knowledge Frontier: Essays in the Representation of Knowledge*, Springer-Verlag, pp. 331–352.

[28] R. Reiter (1987): *A theory of diagnosis from first principles*. *Artificial Intelligence* 32(1), pp. 57–95, doi:10.1016/0004-3702(87)90062-2.

[29] J. Schröder (2003): *Modelling, State Observation and Diagnosis of Quantised Systems*. Lecture Notes in Control and Information Sciences, Springer, doi:10.1007/b94129.

[30] C.J. Tomlin, J. Lygeros & S.S. Sastry (2000): *A game theoretic approach to controller design for hybrid systems*. *Proceedings of the IEEE* 88(7), pp. 949–970, doi:10.1109/5.871303.

[31] V. Vladimerou, P. Prabhakar, M. Viswanathan & G. Dullerud (2008): *STORMED Hybrid Systems*. In: *Automata, Languages and Programming, Lecture Notes in Computer Science* 5126, Springer Berlin / Heidelberg, pp. 136–147, doi:10.1007/978-3-540-70583-3_12.