Kalpa Publications in Computing

Volume 4, 2018, Pages 124–143

28th International Workshop on
Principles of Diagnosis (DX'17)

Counterexample-Guided Abstraction-Refinement for Hybrid Systems Diagnosability Analysis

Hadi Zaatiti[1,2], Lina Ye[2,3], Philippe Dague[2], and Jean-Pierre Gallois[1]

[1] CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, Gif-sur-Yvette, 91191, France
firstname.lastname@cea.fr
[2] LRI, Univ. Paris-Sud & CNRS, Univ. Paris-Saclay, Orsay, 91405, France
firstname.lastname@lri.fr
[3] CentraleSupélec

Abstract

Verifying behavioral or safety properties of hybrid systems, either at design stage such as state reachability and diagnosability, or on-line such as fault detection and isolation is a challenging task. We are concerned here with abstractions oriented towards hybrid systems diagnosability checking. The verification can be done on the abstraction by classical methods developed for discrete event systems extended with time constraints, which provide a counterexample in case of non-diagnosability. The absence of such a counterexample proves the diagnosability of the original hybrid system. In the presence of a counterexample, the first step is to check if it is not a spurious effect of the abstraction and actually exists for the hybrid system, witnessing thus non-diagnosability. Otherwise, we show how to refine the abstraction, guided by the elimination of the counterexample, and continue the process of looking for another counterexample until either a final result is obtained or we reach an inconclusive verdict. We make use of qualitative modeling and reasoning to compute discrete abstractions. Abstractions as timed automata are particularly studied as they allow one to handle time constraints that can be captured at a qualitative level from the hybrid system.

1 Introduction

The increasing complexity of systems makes it challenging to detect and isolate faults. Hybrid systems are no exception, combining both discrete and continuous behaviors. Verifying behavioral or safety properties of such systems, either at design stage such as state reachability, diagnosability and predictability or on-line such as fault detection and isolation is a challenging task. Actually, computing the reachable set of states of a hybrid system is an undecidable matter due to the infinite state space of continuous systems. One way to verify those properties over such systems is by computing discrete abstractions and inferring them from the abstract system back to the original system. Methods have been proposed for diagnosability verification for continuous and discrete systems separately, few of them handle hybrid systems [2, 3, 10].

Diagnosability is a property describing the system ability to determine whether a fault has effectively occurred based on the observations, which has received considerable attention in the literature [7, 9, 15, 17, 18, 20]. However, most of the existing works are applied to discrete event systems.

Note that many modern technological processes are often considered as hybrid ones that exhibit both continuous evolution and discrete transitions, whose combination constitutes important phenomena of such systems. To the best of our knowledge, very few works handle diagnosability of hybrid systems with satisfactory results. Considering that the verification problem for even very simple hybrid systems is undecidable [12], we propose here, inspired by [10], an algorithm of counterexample-guided abstraction-refinement with timed automata as abstract models. The reason is that timed automata allow, by taking into account time constraints, finer abstractions than purely discrete models which are commonly used in all existing works concerned with hybrid systems diagnosability.

In this paper, we are concerned with abstractions oriented towards hybrid systems diagnosability checking. Our goal is to establish abstractions in order to verify, at design stage, if a fault occurring at runtime could be unambiguously detected. This verification can be done on the abstraction by classical methods developed for discrete event systems extended with time constraints, which provide a counterexample in case of non-diagnosability. The absence of such a counterexample proves the diagnosability of the original hybrid system. In presence of a counterexample, one should check if it is not a spurious effect of the abstraction and actually exists in the hybrid system, witnessing thus non-diagnosability. Otherwise, we show how to refine the abstraction, guided by the elimination of the counterexample, and continue to look for another counterexample until either a final result is obtained or we decide to stop the refinement process and end with an inconclusive verdict. We make use of qualitative modeling and reasoning to compute abstractions. Abstractions as timed automata are particularly studied as they allow one to handle time constraints that can be captured at a qualitative level from the hybrid system [4, 6].

2 Hybrid Dynamical Systems

We first describe hybrid systems and then propose a formal representation framework for hybrid automata and introduce the special case of timed automata.

2.1 Hybrid automata definition

Hybrid systems are dynamical systems that include discrete and continuous behaviors [11]. *Hybrid automaton* (HA) is a mean to model such systems; it is an infinite state machine frequently used for this purpose among the scientific community. Each state of the hybrid automaton is twofold with a discrete and a continuous part. The discrete part ranges over a finite domain while the continuous part ranges over the Euclidean space $\mathbb{R}^n$.

Definition 1 (Hybrid automata (HA))**.** *An n-dimensional hybrid automaton (HA) is a tuple $H = (Q, X, S_0, \Sigma, F, Inv, T)$ where:*

- *Q is a finite set of modes (or locations), that can be possibly defined as the valuations set of a finite number of finite valued variables, and represents the* discrete *part of H. X is a set of n real-valued variables (which are continuously differentiable functions of time), whose valuations set $\boldsymbol{X} \subseteq \mathbb{R}^n$ represents the* continuous *part of H. $S = Q \times \boldsymbol{X}$ is the state space of H, whose elements, called states, are noted $(q, \boldsymbol{x})$ with q and $\boldsymbol{x}$ the respective discrete and continuous parts of the state.*
- *$S_0 \subseteq S$ is the set of initial states. If unique, the initial state is noted $(q_0, \boldsymbol{x}_0)$.*
- *Σ is a finite set of events.*
- *$F : S \to 2^{\mathbb{R}^n}$ is a mapping assigning to each state $(q, \boldsymbol{x}) \in S$ a set $F(q, \boldsymbol{x}) \subseteq \mathbb{R}^n$ constraining the time derivative $\dot{\boldsymbol{x}}$ of the continuous part of the mode q by $\dot{\boldsymbol{x}} \in F(q, \boldsymbol{x})$. If there is no uncertainty on*

the derivative, then F is a function $S \to \mathbb{R}^n$ specifying the flow condition $\dot{\boldsymbol{x}} = F(q, \boldsymbol{x})$ in each mode q (the dynamics in each mode is thus given by a set of n first-order ordinary differential equations (ODEs)).

- *$Inv : Q \to 2^{\boldsymbol{X}}$ assigns to each mode q an invariant set $Inv(q) \subseteq \boldsymbol{X}$, which constrains the value of the continuous part of the state while the discrete part is q. We require, for all $q \in Q$, that $\{\boldsymbol{x} \mid (q, \boldsymbol{x}) \in S_0\} \subseteq Inv(q)$.*
- *$T \subseteq S \times \Sigma \times S$ is a relation capturing discontinuous state changes, i.e., instantaneous discrete transitions from one mode to another one. Precisely, $t = (q, \boldsymbol{x}, \sigma, q', \boldsymbol{x}') \in T$ represents a transition whose source and destination states are $(q, \boldsymbol{x})$ with $\boldsymbol{x} \in Inv(q)$ and $(q', \boldsymbol{x}')$ with $\boldsymbol{x}' \in Inv(q')$, respectively, and labeled by the event σ. It represents a jump from $\boldsymbol{x}$ in mode q to $\boldsymbol{x}'$ in mode q'.*

We will call *(concrete) behavior* of H any sequence of continuous solution flows and discrete jumps, rooted in an initial state, satisfying all the constraints above defining H. Hybrid systems are typically represented as finite automata with (discrete, i.e., modes) states Q, initial states $Q_0 = \{q \in Q \mid \exists \mathbf{x} \in Inv(q)\ (q, \mathbf{x}) \in S_0\}$ and transitions δ defined by $\delta = \{(q, \sigma, q') \in Q \times \Sigma \times Q \mid \exists \mathbf{x}, \mathbf{x}'\ (q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T\}$. To each state $q \in Q_0$ is associated an initial (continuous) nonempty set $Init(q) = \{\mathbf{x} \in Inv(q) \mid (q, \mathbf{x}) \in S_0\}$. To each transition $\tau = (q, \sigma, q') \in \delta$ are associated a nonempty guard set $G(\tau) = \{\mathbf{x} \mid \exists \mathbf{x}'\ (q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T\} \subseteq Inv(q)$ and a set-valued reset map $R(\tau) : G(\tau) \to 2^{Inv(q')}$ given by $R(\tau)(\mathbf{x}) = \{\mathbf{x}' \mid (q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T\}$. It is actually equivalent in the definition to provide either T or δ, G and R. In the last case, H is denoted by $(Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$ and we have: $\forall (q, \mathbf{x}), (q', \mathbf{x}') \in S$, $\forall \sigma \in \Sigma$, $((q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T \Leftrightarrow \tau = (q, \sigma, q') \in \delta \wedge \mathbf{x} \in G(\tau) \wedge \mathbf{x}' \in R(\tau)(\mathbf{x}))$. A discrete event automaton $D = (Q, Q_0, \Sigma, \delta)$ can be seen as a hybrid automaton without continuous part. At the opposite, a continuous system $CS = (X, S_0, F, Inv)$ can be seen as a hybrid automaton without discrete part and guards.

It can be in some cases more convenient to adopt a relational-based representation than a set-based representation and to use predicates instead of subsets. By a slight abuse of notation, for each mode q, $Init(q)$ (for $q \in Q_0$), $F(q)$ and $Inv(q)$ indicate then predicates whose free variables are respectively from X, $X \times \dot{X}$ and X and $Init(q)(\mathbf{x})$, $F(q)(\mathbf{x}, \dot{\mathbf{x}})$ and $Inv(q)(\mathbf{x})$ being true means respectively $\mathbf{x} \in Init(q)$, $\dot{\mathbf{x}} \in F(q, \mathbf{x})$ and $\mathbf{x} \in Inv(q)$. In the same way, for each mode transition τ, $G(\tau)$ and $R(\tau)$ indicate predicates whose free variables are respectively from X and $X \times X$ and $G(\tau)(\mathbf{x})$ and $R(\tau)(\mathbf{x}, \mathbf{x}')$ being true means respectively $\mathbf{x} \in G(\tau)$ and $\mathbf{x}' \in R(\tau)(\mathbf{x})$. Guards in any mode q will be assumed non-intersecting: $\forall q \in Q, \forall \tau_1 = (q, \sigma_1, q_1) \in \delta, \forall \tau_2 = (q, \sigma_2, q_2) \in \delta, (\tau_1 \neq \tau_2 \Rightarrow G(\tau_1) \cap G(\tau_2) = \emptyset)$. Figure (1) illustrates a simple thermostat system modeled as a hybrid automaton such that $Q = \{off, on\}$, $X = \{x\}$ where x represents the temperature, $S_0 = (off, [80, 90])$, $\Sigma = \{B_{on}, B_{off}\}$, $F(off) = \{\dot{x} = -x\}$, $F(on) = \{\dot{x} = -x + 100\}$, $\delta = \{\tau_1 = (off, B_{on}, on), \tau_2 = (on, B_{off}, off)\}$, $Inv(off) = x \geq 68$, $Inv(on) = x \leq 82$, $G(\tau_1) = 68 \leq x \leq 70$, $G(\tau_2) = 80 \leq x \leq 82$, $R(\tau_1) = R(\tau_2) = \{x = x'\}$.

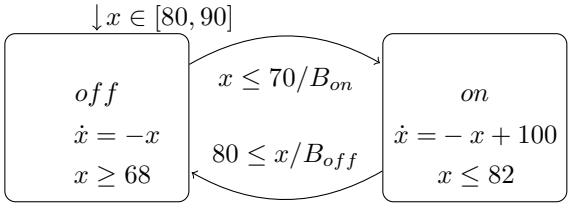

Figure 1: Hybrid automaton modeling a thermostat

2.2 Hybrid automata semantics

We denote by $q_0, q_1, ...$ the modes of Q, and by $x_1, x_2, ..., x_n$ the variables of X.

Definition 2 (Hybrid automaton semantics)**.** *The semantics of a hybrid automaton H, denoted by $[[H]]$, is the set of all executions, which are labeled sequences of states from S with labels in $L = \Sigma \cup \mathbb{R}_+$: $(q_0, \boldsymbol{x}_0) \xrightarrow{l_0} (q_1, \boldsymbol{x}_1)...(q_i, \boldsymbol{x}_i) \xrightarrow{l_i} ...$ such that $(q_0, \boldsymbol{x}_0) \in S_0$ and, for any two successive states $(q_i, \boldsymbol{x}_i) \xrightarrow{l_i} (q_{i+1}, \boldsymbol{x}_{i+1})$ in the sequence, one of the following is true:*

- $l_i = \sigma_i \in \Sigma$ *and* $(q_i, \boldsymbol{x}_i, \sigma_i, q_{i+1}, \boldsymbol{x}_{i+1}) \in T$*;*
- $l_i = d_i \in \mathbb{R}_+$, $q_i = q_{i+1}$, $\boldsymbol{x}_i, \boldsymbol{x}_{i+1} \in Inv(q_i)$ *and* $\exists x : [0, d_i] \to \boldsymbol{X}$ *continuously differentiable function, with* $x(0) = \boldsymbol{x}_i$, $x(d_i) = \boldsymbol{x}_{i+1}$ *and* $\forall t \in (0, d_i)\ \dot{x}(t) \in F(q_i, x(t))$ *and* $x(t) \in Inv(q_i)$.

The trace of an execution h, i.e., the sequence of its labels, is a word from $L^\star$ (or L^ω for infinite h), denoted as $trace(h)$. We denote the total time duration of h by $time(h) \in \mathbb{R}_+ \cup \{+\infty\}$, which is calculated as the sum of all time periods in the trace of h: $time(h) = \sum d_i$. The part of execution $h = (off, 80) \xrightarrow{0.15} (off, 69) \xrightarrow{B_{on}} (on, 69) \xrightarrow{0.5} (on, 81) \xrightarrow{B_{off}} (off, 81)...$ is valid for the thermostat example, thus $h \in [[H]]$. Let $\overline{S} = \bigcup_{q \in Q}(\{q\} \times Inv(q)) \subseteq S$ the (infinite) set of invariant satisfying states of H, $\overline{S}_0 = \bigcup_{q \in Q_0}(\{q\} \times Inv(q)) \subseteq S_0$ the subset of invariant satisfying initial states and $\to \subseteq \overline{S} \times L \times \overline{S}$ the transition relation defined by one or the other condition in Definition 2. The semantics of H is actually given by the labeled transition system $S^t_H = (\overline{S}, \overline{S}_0, L, \to)$, i.e., $[[H]]$ is the set of all paths of S^t_H issued from an initial state. S^t_H, called the *timed transition system of* H, is thus a discretization of H with infinite sets of states and of transition labels. It just abstracts continuous flows by timed transitions retaining only information about the source, the target and the duration of each flow and constitutes the finest abstraction of H we will consider. The timeless abstraction of S^t_H, called the *timeless transition system of* H, is obtained by ignoring also the duration of flows and thus defined as $S_H = (\overline{S}, \overline{S}_0, \Sigma \cup \{\epsilon\}, \to)$, obtained from S^t_H by replacing any timed transition $(q_i, \mathbf{x}_i) \xrightarrow{d_i} (q_{i+1}, \mathbf{x}_{i+1})$ with $d_i \in \mathbb{R}_+$ by the ϵ transition $(q_i, \mathbf{x}_i) \xrightarrow{\epsilon} (q_{i+1}, \mathbf{x}_{i+1})$, that can be considered as a silent transition. It has infinite set of states but finite set of transitions labels. It constitutes the finest timeless abstraction of H we will consider.

Theorem 1 (Correction and completeness of the semantics)**.** *Any concrete behavior of H is timed (resp. timeless) abstracted into an $\overline{S}_0$ rooted path in S^t_H (resp. S_H). Conversely, any path in S^t_H (resp. S_H) that alternates continuous and discrete transitions (in particular any single transition) abstracts a part of a concrete behavior of H and, if F is a singleton function (i.e., deterministic derivative), any $\overline{S}_0$ rooted path in S^t_H (resp. S_H) abstracts a concrete behavior of H. In this latter case, there is thus no spurious abstract behavior in S^t_H (resp. S_H), which expresses faithfully the behavior of H.*

2.3 Timed automata

The form of the dynamics F determines primarily the class of the hybrid automaton. For example the rectangular class, for which the dynamics valuations are a cartesian product of intervals, lies on the boundary of the decidability over reachability problem with some restrictions [12]. We are particularly interested in timed automata, a class of hybrid automata where the continuous variables x_i, $1 \leq i \leq n$, with values in $\mathbb{R}_+$, called clocks, have all first order derivatives equal to one. So time elapses identically for all clocks. The set $C(X)$ of constraints over a set of clocks X is defined as follows: a constraint is either a primitive constraint of the form $x_i\ op\ c_i$, where $c_i \in \mathbb{R}_+$ (at the theoretical level because, in practice, $\mathbb{Q}$ is used instead of $\mathbb{R}$ for computer implementation reasons) and op is one of the operators

$<, \leq, =, \geq, >$, or a finite conjunction of primitive constraints. The satisfiability set of a constraint is thus a rectangle in $\mathbb{R}_+^n$, i.e., the product of n intervals of the half real line $\mathbb{R}_+$, and we will identify $C(X)$ to the set of rectangles.

Definition 3 (Timed automaton (TA))**.** *A timed automaton (TA) is a hybrid automaton* $T = (Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$ *such that:*

- $\boldsymbol{X} = \mathbb{R}_+^n$.
- $S_0 = Q_0 \times \{\boldsymbol{0}\}$.
- $\forall q \in Q\ F(q, .) = \boldsymbol{1}$*, which means that the dynamics of clocks evolution in each mode* q *is given by* $\dot{x}_i = 1$.
- $Inv : Q \to C(X)$ *associates to each mode* q *a rectangle invariant in* $\boldsymbol{X}$*. We require* $\boldsymbol{0} \in Inv(q_0)$.
- $G : \delta \to C(X)$ *associates to each discrete transition* (q, σ, q') *a rectangle guard in* $Inv(q)$.
- $\forall \tau \in \delta\ \exists Y(\tau) \subseteq X\ \forall \boldsymbol{x} \in G(\tau)\ R(\tau)(\boldsymbol{x}) = \{\boldsymbol{x}'\}$ *with* $\boldsymbol{x}'_i = 0$ *if* $x_i \in Y(\tau)$ *and* $\boldsymbol{x}'_i = \boldsymbol{x}_i$ *otherwise, i.e., clocks in* $Y(\tau)$ *are reset to zero with transition* τ*, the others keeping their values.*

The notation of a timed automaton T is generally simplified as $T = (Q, X, Q_0, \Sigma, Inv, (\delta, G, Y))$. The semantics of T as a hybrid automaton, given by Definition 2, can be simplified by merging together in an execution successive timed transitions between two discrete transitions and summing up their time period labels. An execution in $[[T]]$ is thus a sequence h of alternating time steps (possibly with 0 time period) and discrete steps of the form $(q_0, \mathbf{x}_0) \xrightarrow{d_1} (q_0, \mathbf{x}_0 + d_1) \xrightarrow{\sigma_1} (q_1, \mathbf{x}_1) \xrightarrow{d_2} ...$ whose trace $trace(h)$ is the timed word $d_1\sigma_1 d_2 ... \in \mathbb{R}_+(\Sigma\mathbb{R}_+)^*$ and duration is $time(h) = \sum d_i$.

3 Diagnosability of Hybrid Systems

3.1 Hybrid system model for diagnosability

Remind that diagnosability is a system property allowing one to determine with certainty, at the design stage, a fault occurrence, based on available observations. Precisely, in a given system model, the existence of two infinite behaviors, with the same observations but exactly one containing the considered fault, violates diagnosability. Hence, to be able to analyze such property, it is necessary to define what can be observed for given systems as well as what are considered as faults. In practice, the observations are partial, only parts of the system are known and are usually obtained from sensors. In whole generality we will consider that both some discrete jumps between modes and some continuous variables inside a mode may be observable. The sets of observable events and variables are assumed to be time invariant, the second one being also assumed to be independent of the mode for sake of simplicity. Events are observed together with their instantaneous occurrence time and variables values are assumed to be observed at any moment. E.g, for our thermostat system in Figure 1, transitions B_{on}, B_{off} and temperature x are assumed to be observable. For what concerns faults, we will suppose that they are modeled, as for discrete event systems, by some unobservable discrete jumps, between precisely a normal mode and a faulty mode, translating often in a change of dynamics. This it well adapted for abrupt faults but progressive faults or degraded modes (as a shift of parameter) can be also represented in this way, the designer abstracting a slow evolution in a sudden change when he estimates that the behavior variation induced (that he will model by means of the invariant and the guard) cannot any more let consider the given mode as normal. To sum up, we obtain the following definition.

Definition 4 (Partially observable hybrid automaton (POHA)). *A partially observable hybrid automaton (POHA) is an hybrid automaton H according to Definition 1 where:*

- $\Sigma = \Sigma_o \uplus \Sigma_u \uplus \Sigma_f$, *i.e., the set of events is the disjoint union of the set Σ_o of observable (normal) events, the set Σ_u of unobservable normal events and the set Σ_f of unobservable fault events.*
- $X = X_o \uplus X_u$, *i.e., the set of continuous real-valued variables is the disjoint union of the set X_o of observable variables and the set X_u of unobservable variables.*

Definition 5 (Execution (timed) observation). *Given an execution $h \in [[H]]$ of a POHA H, $h = (q_0, \boldsymbol{x}_0) \xrightarrow{l_0} (q_1, \boldsymbol{x}_1)...(q_i, \boldsymbol{x}_i) \xrightarrow{l_i} ...$, with $l_i \in \Sigma \cup \mathbb{R}_+$, the (timed) observation of h is defined as $Obs(h) = \boldsymbol{x}_0^o, l_0^o, \boldsymbol{x}_1^o ... \boldsymbol{x}_i^o, l_i^o, ...$, where:*

- $\boldsymbol{x}_i^o$ *is obtained by projecting $\boldsymbol{x}_i$ on variables in X_o.*
- $l_i^o = l_i$ *if $l_i \in \Sigma_o \cup \mathbb{R}_+$. Otherwise, $l_i^o = \varepsilon$, which is then removed from $Obs(h)$.*

Similarly, one can define observation for timed automata. The difference is that we do not assume any information about continuous clocks, so there is no $\mathbf{x}_i^o$. Then, the observation is obtained from the trace (a timed word) by erasing all unobservable events and by adding up the periods between any two successive observable events in the resulting sequence.

3.2 System diagnosability definition

A fault is modeled as a fault event that alters the system from a normal mode to an abnormal mode. There may exist different fault events in a given system. For the sake of reducing complexity (from exponential to linear in the number of different fault events) and of simplicity, in the following only one fault type, i.e., fault event, at a time is considered but multiple occurrences of this event are allowed, and the other types of fault events are thus processed as unobservable normal events. However, this framework can be extended in a straightforward way such that a number of different faults can be considered simultaneously. Now we adapt to hybrid systems the diagnosability definition [17] introduced for discrete event systems. h^F denotes a finite execution whose last label is a first occurrence of the fault event F considered. Given a finite execution $h \in [[H]]$ such that $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1)...(q_i, \mathbf{x}_i)$, the set of post-executions of h in $[[H]]$ is defined as $[[H]]/h = \{h' = (q_i, \mathbf{x}_i) \xrightarrow{l_i} ... \mid h.h' \in [[H]]\}$, where $h.h'$ is obtained by merging the final state of h and the first state of h', both should be the same.

Definition 6 (Hybrid automaton diagnosability). *A fault F is diagnosable in a POHA H iff*

$$\exists \triangle \in \mathbb{R}_+ \; \forall h^F \in [[H]] \; \forall h' \in [[H]]/h^F \; (time(h') \geq \triangle$$
$$\Rightarrow \forall h \in [[H]](Obs(h) = Obs(h^F.h') \Rightarrow F \in trace(h)).$$

The above definition states that F is diagnosable iff there is always a lapse of time Δ such that, for all executions h^F in $[[H]]$ and all post-executions h' that last at least Δ, then every execution that is observably equivalent to $h^F.h'$ should contain F. In other words, the existence of two behaviors, that at any time are indistinguishable, with exactly one of them containing F with arbitrarily long time after F, violates the diagnosability for hybrid automata. Inspired from the framework of discrete event systems, we define critical pairs for partially observable hybrid automata as follows (for a faulty execution h, we note $time(h, F)$ the period of time in h from the first occurrence of F).

Definition 7 (Critical pair). *A pair of executions h, $h' \in [[H]]$ is called a Δ-critical pair with respect to F iff: $F \in trace(h)$ and $F \notin trace(h')$ and $time(h, F) \geq \Delta$ and $Obs(h) = Obs(h')$. It is called a critical pair if $time(h) = +\infty$.*

Theorem 2. *A fault F is diagnosable in a POHA H iff, for some Δ, there is no Δ-critical pair in $[[H]]$ with respect to F (i.e., there is no arbitrarily long time after F critical pair).*
F is diagnosable in a partially observable timed automaton T iff there is no critical pair in $[[T]]$ with respect to F. Checking diagnosability of T is PSPACE-complete [19].

4 Hybrid Automata Abstraction

We now introduce finite state space decomposition of a hybrid automaton. We will then present an abstraction based on such a decomposition that incorporates reachability and time constraints. Later on, in the next section, we will discuss the refinement of the abstraction yielding constraints with better precision than before refinement.

4.1 Geometric decomposition of the state space

Definition 8 (Continuous space partition)**.** *A (finite) partition P of the Euclidean space $\mathbb{R}^n$ is a finite set of nonempty connected subsets of $\mathbb{R}^n$ such that every point $x \in \mathbb{R}^n$ is in one and only one of those subsets. We can write $\mathbb{R}^n = \biguplus_{p \in P} p$. An element p of P will be referred to as a partition element and we will call it a region. For a subset E of $\mathbb{R}^n$, we will denote by $P(E)$ the subset of regions of P that have a nonempty intersection with E.*

The only smoothness hypothesis we will impose for the moment over a partition w.r.t. a given continuous dynamics is that any (finite) path solution of the dynamics crosses only a finite number of times each region, more precisely, $\forall x : [0, 1] \to \mathbb{R}^n$ a continuously differentiable function satisfying the flow condition, $\forall p \in P$ a region, $x^{-1}(p)$ is a finite union of intervals. In practice, partitions are chosen enough regular and smooth, with regions in any dimension from 1 to n such as (from simpler to more complex) rectangles, zonotopes, polytopes or defined by a set of polynomial inequalities. The choice among different partitions is guided by the dynamics and the property we wish to verify. For example, consider a continuous system with dynamics F. A coarse but helpful way to obtain a highly abstract reachability mapping would be to identify regions of the state space that conserve the sign vector of F, i.e., for all elements of the same region the derivative signs for each dimension would keep the same value, i.e., either negative or positive or null. Thus, the regions would be the connected components of the 3^n subsets E_s of $\mathbb{R}^n$ parametrized by sign vectors $s \in \{-1, 0, +1\}^n$: $E_s = \{\mathbf{x} \in \mathbf{X} \mid \forall i, 1 \leq i \leq n,\ \dot{\mathbf{x}}^i < 0 \text{ if } s^i = -1,\ \dot{\mathbf{x}}^i = 0 \text{ if } s^i = 0,\ \dot{\mathbf{x}}^i > 0 \text{ if } s^i = +1\}$.

If the considered system is a hybrid automaton, it is practical to allow different partitions in different modes. We define thus a decomposition of the hybrid state space as follows.

Definition 9 (Hybrid state space decomposition)**.** *Given a hybrid automaton H and a set $\mathfrak{P}$ of partitions of the valuations set of X, $\boldsymbol{X} \subseteq \mathbb{R}^n$, we say that $\mathfrak{P}$ decomposes H if there is an onto function $d : Q \to \mathfrak{P}$ which associates to each $q \in Q$ a partition $d(q) \in \mathfrak{P}$.*

The initial and invariant sets and the guards satisfiability domains and variables reset domains are primary elements to take into consideration while abstracting. For $q \in Q$ and $\tau = (q, \sigma, q') \in \delta$, we denote the regions families $d(q)(Init(q))$, $d(q)(Inv(q))$, $d(q)(G(\tau))$ by $d_{Init}(q)$, $d_{Inv}(q)$, $d_G(q, \tau) \subseteq d(q)$ and, for a region $p \in d_G(q, \tau)$, we denote $d(q')(R(\tau)(p \cap G(\tau)))$ by $d_R(q', \tau, p) \subseteq d(q')$. When possible, we will try to define d such that $Init(q)$, $Inv(q)$, $G(\tau)$ and $R(\tau)(p)$ are exactly the unions of the regions in those families (if not, those regions families over-approximate them).

4.2 Encoding hybrid automata reachability constraints

We have defined in 2.2, the timeless transition system S_H of a hybrid automaton H as the finest timeless abstraction that can be obtained. However, in practice S_H can only be computed for very restricted classes of hybrid automata. We will define a less granular time-abstract transition system based on a set of partitions and define the relations between adjacent regions.

Definition 10 (Adjacent regions)**.** *Two distinct regions p_1, p_2 of a partition P of $\mathbb{R}^n$ are adjacent if one intersects the boundary of the other: $p_1 \cap \overline{p_2} \neq \emptyset$ or $\overline{p_1} \cap p_2 \neq \emptyset$, where $\overline{p}$ refers to the closure of p.*

Definition 11 (Decomposition-based timeless abstract automaton of a hybrid automaton)**.** *Given a hybrid automaton $H = (Q, X, S_0, \Sigma, F, Inv, T)$ and a decomposition $(\mathfrak{P}, d)$ of H, we define the timeless abstract (finite) automaton of H with respect to $\mathfrak{P}$ as $DH_{\mathfrak{P}} = (Q_{DH}, Q_{0_{DH}}, \Sigma_{DH}, \delta_{DH})$ with:*

- $Q_{DH} = \{(q, p) | q \in Q, p \in d(q)\}$.
- $Q_{0_{DH}} = \{(q, p_{Init}) | q \in Q_0, p_{Init} \in d_{Init}(q)\}$.
- $\Sigma_{DH} = \Sigma \cup \{\epsilon\}$.
- $((q_i, p_k), \sigma, (q_j, p_l)) \in \delta_{DH}$ *iff one of both is true:*
  - $q_i \neq q_j$ *and* $\sigma \in \Sigma$ *and* $p_k \in d_G(q_i, \tau)$ *and* $p_l \in d_R(q_j, \tau, p_k)$ *where* $\tau = (q_i, \sigma, q_j)$ *and* $\exists t = (q_i, \boldsymbol{x}, \sigma, \boldsymbol{x'}, q_j) \in T$ *such that* $\boldsymbol{x} \in p_k$ *and* $\boldsymbol{x'} \in p_l$.
  - $q_i = q_j$ *and* $\sigma = \epsilon$ *and* $p_k, p_l \in d_{Inv}(q_i)$ *are adjacent regions and* $\exists d \in \mathbb{R}_+^*$ *and* $\exists x : [0, d] \to \boldsymbol{X}$ *continuously differentiable function such that* $\forall t \in (0, d)\ \dot{x}(t) \in F(q_i, x(t))$, $\forall t \in [0, d]\ x(t) \in Inv(q_i)$, $x(0) \in p_k$, $x(d) \in p_l$, $\exists c\ 0 \leq c \leq d\ \forall t \in (0, c)\ x(t) \in p_k\ \forall t \in (c, d)\ x(t) \in p_l$ *and* $x(c) \in p_k \cup p_l$.

The defined timeless abstract automaton encodes reachability with adjacent regions of the state space, the events in Σ witnessing mode changes and ϵ transitions representing a continuous evolution between adjacent regions in the same mode. Notice that $((q_i, p_k), \sigma, (q_j, p_l)) \in \delta_{DH} \Rightarrow \exists \mathbf{x}_k \in p_k\ \exists \mathbf{x}_l \in p_l\ (q_i, \mathbf{x}_k) \xrightarrow{\sigma} (q_j, \mathbf{x}_l)$ in S_H, the converse being true for $\sigma \in \Sigma$. The mapping $\alpha_{\mathfrak{P}}$ defined by $\alpha_{\mathfrak{P}}((q, \mathbf{x})) = (q, p)$ with $p \in d(q)$ and $\mathbf{x} \in p$ defines an onto timeless abstraction function $\alpha_{\mathfrak{P}} : \overline{S} \to Q_{DH}$. If the flow condition F is a singleton, $\alpha_{\mathfrak{P}}$ maps any transition of S_H to a unique path in $DH_{\mathfrak{P}}$. The coarsest timeless abstract automaton is obtained when partitions of $\mathfrak{P}$ have all a unique region $p = \mathbf{X}$ and is thus (Q, Q_0, Σ, δ), i.e., the discrete part of H without its continuous part. It corresponds to the coarsest timeless abstraction function $\alpha_{\{\{\mathbf{X}\}\}}((q, \mathbf{x})) = q$. For our previous thermostat example, this gives $(\{off, on\}, \{off\}, \{B_{on}, B_{off}\}, \{(off, B_{on}, on), (on, B_{off}, off)\})$ and the abstraction of the execution h given previously (2.2) is just $off \xrightarrow{B_{on}} on \xrightarrow{B_{off}} off...$

Theorem 3 (Timeless abstraction completeness)**.** *Given a decomposition $\mathfrak{P}$ of H, any concrete behavior of H is timeless abstracted into a $Q_{0_{DH}}$ rooted path in $DH_{\mathfrak{P}}$ and any transition of $DH_{\mathfrak{P}}$ abstracts a part of a concrete behavior of H. If the flow condition F is a singleton function then the timeless abstraction function $\alpha_{\mathfrak{P}}$ defines a trace preserving mapping (still denoted by $\alpha_{\mathfrak{P}}$) from $\overline{S}_0$ rooted paths in S_H (i.e., timeless executions of H) to $Q_{0_{DH}}$ rooted paths in $DH_{\mathfrak{P}}$ and thus the language defined by S_H is included in the language defined by $DH_{\mathfrak{P}}$.*

Obviously, a path in $DH_{\mathfrak{P}}$ does not abstract in general a concrete behavior of H (as the behaviors parts abstracted by the individual transitions do not connect in general) which expresses that abstraction creates spurious behaviors.

If now H is a POHA, in the same way we defined the observation of a concrete execution in Definition 5 we define the observation of its timeless abstraction.

Definition 12 (Timeless abstraction observation)**.** *Given a POHA H and $h = (q_0, p_0) \xrightarrow{\sigma_0} (q_1, p_1)...$ $(q_i, p_i) \xrightarrow{\sigma_i}$ ..., with $\sigma_i \in \Sigma \cup \{\epsilon\}$, a timeless abstract path in $DH_{\mathfrak{P}}$, the observation of h is defined as $Obs(h) = p_0^o, \sigma_0^o, p_1^o ... p_i^o, \sigma_i^o, ...$, where*

- *p_i^o is obtained by projecting p_i on variables in X_o.*
- *$\sigma_i^o = \sigma_i$ if $\sigma_i \in \Sigma_o$. Otherwise, $\sigma_i^o = \varepsilon$, which is then removed from $Obs(h)$.*

4.3 Encoding hybrid automata time constraints

We are concerned with verifying temporal properties of hybrid systems and checking the diagnosability property using time constraints. For this reason, we define in this subsection, always related to a decomposition of the state space into partitions, an abstraction of the hybrid automaton as a timed automaton that partly captures the time constraints at the level of the regions. We will first introduce some intuitive ideas. Considering a partition P of the $\mathbb{R}^n$ state space of a continuous system with arbitrary dynamics F, the set of trajectories (i.e., the continuous solution flows) entering a region $p \in P$ is in one of these two cases: either at least one of the trajectories ends up trapped inside p for all future times or all of them exit p to an adjacent region within a bounded time under the continuity assumption. In the first case, no time constraint can be associated with the region p unless a reshaping of p is applied; in the latter, it is possible to compute time constraints satisfied by all trajectories entering and leaving the region p. We will give a formal definition of the timed automaton constructed from given hybrid automaton and partitions set and then discuss some cases where a time bound can be practically computed.

Definition 13 (Region time interval and time bounds)**.** *Given a continuous system CS, a partition P of $\mathbb{R}^n$ and $p \in P$ one of its regions, we say that $I_p = [t_{min}, t_{max}]$, with $t_{min}, t_{max} \in \mathbb{R}_+ \cup \{+\infty\}$, is a region time interval of p for CS if all trajectories of the CS entering p at time t leave p at time $t + t_{min}$ at least and $t + t_{max}$ at most. t_{min} and t_{max} are lower and upper time bounds of p.*

For a hybrid automaton, we denote a time interval relative to the region p in mode q as $I_{(q,p)}$.

Definition 14 (Decomposition-based timed abstract automaton of a hybrid automaton)**.** *Given a hybrid automaton $H = (Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$, a decomposition $(\mathfrak{P}, d)$ and the timeless abstract automaton $DH_{\mathfrak{P}} = (Q_{DH}, Q_{0_{DH}}, \Sigma_{DH}, \delta_{DH})$ of H with respect to $\mathfrak{P}$, we define the timed abstract automaton of H with respect to $\mathfrak{P}$ as $TH_{\mathfrak{P}} = (Q_{DH}, \{c\}, Q_{0_{DH}}, \Sigma_{DH}, Inv_{TH}, (\delta_{DH}, G_{TH}, Y_{TH}))$ such that, $\forall (q,p) \in Q_{DH}$ with a region time interval $I_{(q,p)} = [t_{min}, t_{max}]$:*

- *$Inv_{TH}((q,p)) = [0, t_{max}]$.*
- *$\forall \tau = ((q,p), \sigma, (q_1, p_1)) \in \delta_{DH}$, $G_{TH}(\tau) = [t_{min}, +\infty)$ if $\sigma = \epsilon$ and p does not intersect any reset set (i.e., $\forall \tau' = (q', \sigma', q) \in \delta_{DH}$ $p \notin d(q)(R(\tau')(G(\tau')))$) or $[0, +\infty)$ else.*

and, $\forall \tau \in \delta_{TH}$, $Y_{TH}(\tau) = \{c\}$.

The timed abstract automaton adds time constraints to those states (q,p) of the timeless abstract automaton for which an interval $I_{(q,p)}$ is computable as non-trivial (i.e., $I_{(q,p)} \neq [0, +\infty)$), by using one local clock c (reset at 0 in each state) that measures the sojourn duration t in each state (q,p), i.e., in each region p, and coding these constraints by means of invariant and guard of c in each state. The invariant codes the maximum sojourn duration as the upper time bound of the region p and the guard codes the minimum sojourn duration as the lower time bound of the region p when both entering and leaving the region are not the result of discrete jumps (controlled here directly for the out-transition and by requiring that p does not intersect any reset set for all possible in-transitions). In the thermostat example, consider the partition into two regions associated to the mode off given by the initial states set $(off, [80, 90])$

and by $(off, [68, 80))$. Then we take as time bounds for $(off, [80, 90])$ $t_{min} = 0$ and $t_{max} = 0.12$ (the exact upper bound, i.e., the time for the temperature to decrease from 90 to 80 is $Log(\frac{9}{8})$). It means that we define in the timed abstract automaton $Inv_{TH}((off, [80, 90])) = [0, 0.12]$. A beginning of execution of the timed abstract automaton is for example $(off, [80, 90]) \xrightarrow{0.08} (off, [68, 80])$.

Theorem 4 (Timed abstraction completeness). *Given a decomposition $\mathfrak{P}$ of H, any concrete behavior of H is timed abstracted into an execution in $TH_{\mathfrak{P}}$. If the flow condition F is a singleton function then the abstraction function $\alpha_{\mathfrak{P}}$ defines a mapping, denoted by $\alpha^t_{\mathfrak{P}}$, from $\overline{S}_0$ rooted paths in S^t_H (i.e., executions of H) to executions in $TH_{\mathfrak{P}}$. This mapping is trace preserving once ϵ labels are erased from executions traces in $TH_{\mathfrak{P}}$ and time period labels are added up between two consecutive events labels in both executions traces in S^t_H and in $TH_{\mathfrak{P}}$. This means that, for any execution $(q_0, \boldsymbol{x}_0) \xrightarrow{w}_* (q_i, \boldsymbol{x}_i) \in [[H]]$, with $w \in L^*$ (where $L = \Sigma \cup \mathbb{R}_+$), it exists a unique execution $(q_0, p_0) \xrightarrow{w'}_* (q_j, p_j) \in [[TH_{\mathfrak{P}}]]$, with $w' \in L'^*$ (where $L' = L \cup \{\epsilon\}$), $\boldsymbol{x}_0 \in p_0$, $q_j = q_i$, $\boldsymbol{x}_i \in p_j$, $w'_{|\Sigma} = w_{|\Sigma}$ (where $_{|\Sigma}$ is the projection of timed words on words on Σ^*) and, for any two successive events $w_l = w'_{l'}$ and $w_m = w'_{m'}$ of $w_{|\Sigma}$, $\sum_{l'<k'<m', w'_{k'} \neq \epsilon} w'_{k'} = \sum_{l<k<m} w_k$.*

Forgetting time, i.e., removing the clock, provides a natural abstraction function α from $TH_{\mathfrak{P}}$ to $DH_{\mathfrak{P}}$ which maps an execution $(q_0, p_0) \xrightarrow{l_0} (q_1, p_1)...(q_i, p_i) \xrightarrow{l_i}$ ..., with $l_i \in \Sigma \cup \{\epsilon\} \cup \mathbb{R}_+$, in $TH_{\mathfrak{P}}$ into the execution $(q_0, p_0) \xrightarrow{\sigma_0} (q_1, p_1)...(q_i, p_i) \xrightarrow{\sigma_i}$ ..., with $\sigma_i \in \Sigma \cup \{\epsilon\}$, in $DH_{\mathfrak{P}}$, with $\sigma_i = l_i$ if $l_i \in \Sigma \cup \{\epsilon\}$ and continuous transitions labeled by $l_i = d_i \in \mathbb{R}_+$ are suppressed. We have: $\alpha_{\mathfrak{P}} = \alpha \circ \alpha^t_{\mathfrak{P}}$.

Definition 15 (Timed abstraction observation). *Given a POHA H and $h = (q_0, p_0) \xrightarrow{l_0} (q_1, p_1)...(q_i, p_i) \xrightarrow{l_i}$ ..., with $l_i \in \Sigma \cup \{\epsilon\} \cup \mathbb{R}_+$, an execution in $TH_{\mathfrak{P}}$, i.e., a timed abstract path, the observation of h is defined as $Obs(h) = p^o_0, l^o_0, p^o_1...p^o_i, l^o_i, ...$, where*

- *p^o_i is obtained by projecting p_i on variables in X_o.*
- *$l^o_i = l_i$ if $l_i \in \Sigma_o \cup \mathbb{R}_+$. Otherwise, $l^o_i = \varepsilon$, which is then removed from $Obs(h)$.*

4.4 Computing time bounds

Now we present and discuss some situations for which time bounds can be computed for the continuous evolution of the hybrid system. In the following, CS is a continuous system, and P is a partition of $\mathbb{R}^n$, $p \in P$ and $I_p = [t_{min}, t_{max}]$ an associated region time interval of p.

Proposition 1 (Sojourn bounds). *A sufficient but not necessary condition for the region p to have finite time bounds (t_{max} finite, thus real nonnegative constant) is that $\exists i \; 1 \leq i \leq n \quad \forall \boldsymbol{x} \in \overline{p} \quad \dot{\boldsymbol{x}}_i \neq 0$.*

If the condition in proposition 1 is verified over p for a dimension i then all trajectories should respect finite time bounds for staying in the region p. On the other hand, a trajectory making a finite number of orbital spins once inside p then exiting p does not satisfy this condition while having finite time bounds. One way to look for a finite time bound is to refine the partition with the objective that the regions become small enough for the condition to hold for each of them. If the derivatives along each axis take each a finite number of null values inside the partition, but never all at the same time, there is no problem with refining the partition to have each null derivative point alone in one region. In the other cases, we present now some examples for which a time bound can be nevertheless obtained.

Finite number of equilibrium points. If the derivatives along each coordinate are null in a finite number of points X_{eq} at the same time, we repartition p such that each new region p_i has exactly one

point of X_{eq} on its boundary. In one dimension, suppose the null derivative point is at x_{eq} and the initial value of $x(t)$ is x_0. The time, which we denote by $\mathcal{T}$, taken by the continuous variable $x(t)$ to cross from x_0 to x_{eq} can in some cases be finite. If $\dot{x}$ is of the form x^k where $k \in \mathbb{R}$, then by studying $\mathcal{T}$ in a neighborhood around $k = 1$ we obtain:

$$\mathcal{T} = \int_0^{t_{eq}} \mathrm{d}t = \int_{x_0}^{x_{eq}} \frac{1}{\dot{x}}\,\mathrm{d}x = \int_{x_0}^{0} \frac{1}{x^k}\,\mathrm{d}x = \left[\frac{x^{-k+1}}{(-k+1)}\right]_{x_0}^{0} \tag{1}$$

$\mathcal{T}$ is convergent if $k < 1$. Thus a time bound can be computed for all dynamics of the form $\dot{x} = x^k$ with $k < 1$, for example for the square root $\dot{x} = \sqrt{x}$. In two dimensions, let r be the distance from the equilibrium point $M_{eq}(x_{eq}, y_{eq})$ to a point $M(x, y)$ which is initially in region p. Let $X = x_{eq} - x$ and $Y = y_{eq} - y$. Since $r^2 = X^2 + Y^2$ then $2r\dot{r} = 2X\dot{X} + 2Y\dot{Y}$ and if $\dot{r} \neq 0$ the time to reach M_{eq} is :

$$\mathcal{T} = \int_0^{t_{eq}} \mathrm{d}t = \int_{r_0}^{0} \frac{1}{\dot{r}}\,\mathrm{d}r = \int_{r_0}^{0} \frac{r}{X\dot{X} + Y\dot{Y}}\,\mathrm{d}r \tag{2}$$

For example, consider the two dimensional continuous system $\dot{x} = -x^2, \quad \dot{y} = -y$ where $(x, y) \in \mathbb{R}^+ \times \mathbb{R}^+$. The equilibrium point is $M_{eq}(0, 0)$. In polar coordinates $x = rcos(\theta)$ and $y = rsin(\theta)$, then $r\dot{r} = x\dot{x} + y\dot{y} = -x^3 - y^2 = -x^2 - y^2 + x^2 - x^3 = -r^2 + x^2(1 - x) = r^2(-1 + cos^2(\theta)(1 - rcos(\theta)))$. In a neighborhood around $(0, 0)$:

$$\mathcal{T} = \int_{r_0}^{0} \frac{1}{r(-1 + cos^2(\theta)(1 - rcos(\theta)))}\,\mathrm{d}r \geq \int_{r_0}^{0} \frac{-1}{r}\,\mathrm{d}r$$

Thus $\mathcal{T}$ is infinite, the equilibrium point is never reached. This reasoning can be extended to n dimensions by evaluating $r\dot{r}$ and using spherical (or hyperspherical) coordinates and to polynomials with real exponents.

Infinite number of null derivatives. Studying a case where at least one derivative along an axis takes an infinite number of null values in a connected set can be done by extending the previous method. For the particular class of continuous systems where the dynamics are only allowed multi-affine function form, [14] showed how it is possible to capture time constraints by decomposing the infinite state space $\mathbb{R}^n$ into hypercubes and evaluate the time elapsed between entering and exiting each cube by bounding the dynamics.

5 Hybrid Automata Abstraction Refinement

We will now explain and formalize the refinement process of the previously defined abstraction. For this purpose, we construct a finer couple of discrete and timed automata by defining a more granular decomposition for regions and give the necessary assumptions to compute such refinement. By making the partition more granular in regions of interest, tighter time bounds are also obtained. The refinement is a necessary step for the CEGAR scheme, it is a required step when a proof for the verification of a property could not be made at a given abstraction level.

5.1 State space decomposition refinement

Definition 16 (Partition refinement)**.** *Given two partitions P and P' of $\mathbb{R}^n$, we say that P' is a refining partition of P iff $\forall p' \in P'\ \exists p \in P\ p' \subseteq p$. This implies: $\forall p \in P\ \exists P'_p \subseteq P'\ p = \biguplus_{p' \in P'_p} p'$.*

Definition 17 (Hybrid state space decomposition refinement)**.** *Given two decompositions* $(\mathfrak{P}, d)$ *and* $(\mathfrak{P}', d')$ *of a hybrid automaton* $H = (Q, X, S_0, \Sigma, F, Inv, T)$*, we say that* $\mathfrak{P}'$ *refines* $\mathfrak{P}$*, denoted by* $\mathfrak{P}' \prec \mathfrak{P}$*, if* $\forall q \in Q$ $d'(q)$ *is a refining partition of* $d(q)$ *and* $(\mathfrak{P}', d') \neq (\mathfrak{P}, d)$*.*

We now define the relation between two decomposition-based abstract automata of a hybrid automaton and the decomposition refinement ordering relation.

Definition 18 (Refined abstract automaton)**.** *Given a hybrid automaton* H *and two abstract timeless automata* $DH_{\mathfrak{P}}$ *and* $DH_{\mathfrak{P}'}$ *(resp. two abstract timed automata* $TH_{\mathfrak{P}}$ *and* $TH_{\mathfrak{P}'}$*) of* H *with respect to two decompositions* $(\mathfrak{P}, d)$ *and* $(\mathfrak{P}', d')$ *respectively, we say that* $DH_{\mathfrak{P}'}$ *is a timeless refinement of* $DH_{\mathfrak{P}}$ *(resp.* $TH_{\mathfrak{P}'}$ *is a timed refinement of* $TH_{\mathfrak{P}}$*) abstracting* H *if* $\mathfrak{P}' \prec \mathfrak{P}$*, which we denote by* $DH_{\mathfrak{P}'} \prec DH_{\mathfrak{P}}$ *(resp.* $TH_{\mathfrak{P}'} \prec TH_{\mathfrak{P}}$*). The onto abstraction function* $\alpha_{\mathfrak{P}',\mathfrak{P}} : Q_{DH}^{\mathfrak{P}'} \to Q_{DH}^{\mathfrak{P}}$ *given by* $\alpha_{\mathfrak{P}',\mathfrak{P}}((q, p')) = (q, p)$ *with* $p' \subseteq p$ *defines a trace preserving mapping* $\alpha_{\mathfrak{P}',\mathfrak{P}}$ *from* $DH_{\mathfrak{P}'}$ *to* $DH_{\mathfrak{P}}$ *(and thus the language defined by* $DH_{\mathfrak{P}'}$ *is included in the language defined by* $DH_{\mathfrak{P}}$*) and we have:* $\alpha_{\mathfrak{P}} = \alpha_{\mathfrak{P}',\mathfrak{P}} \circ \alpha_{\mathfrak{P}'}$ *(resp. a trace preserving mapping* $\alpha^t_{\mathfrak{P}',\mathfrak{P}}$ *from* $TH_{\mathfrak{P}'}$ *to* $TH_{\mathfrak{P}}$*, after trace simplification as in Theorem 4 and provided the time bounds used in* $TH_{\mathfrak{P}'}$*, once added for all regions* p' *included in a given region* p*, are at least as tight as the time bounds used in* $TH_{\mathfrak{P}}$ *and we have:* $\alpha^t_{\mathfrak{P}} = \alpha^t_{\mathfrak{P}',\mathfrak{P}} \circ \alpha^t_{\mathfrak{P}'}$*).*

In practice, the refined abstract automaton is obtained by performing a finite number of state split operations.

Definition 19 (State split operation)**.** *Given an abstract timeless automaton* $DH_{\mathfrak{P}}$ *(resp. an abstract timed automaton* $TH_{\mathfrak{P}}$*) of a hybrid automaton* H*, a split operation of the state* $(q, p) \in Q_{DH}$ *is defined by a partition* $\{p_1, p_2\}$ *of* p*,* $p = p_1 \biguplus p_2$*, and results in two states* (q, p_1) *and* (q, p_2) *and in the refined abstract timeless automaton* $DH_{\mathfrak{P}'}$ *(resp. the refined abstract timed automaton* $TH'_{\mathfrak{P}}$*) with* $\mathfrak{P}'$ *obtained from* $\mathfrak{P}$ *by replacing* $d(q)$ *by* $d'(q) = d(q) \backslash \{p\} \cup \{p_1, p_2\}$*.*

The construction of $DH_{\mathfrak{P}'}$ from $DH_{\mathfrak{P}}$ after a (q, p) state split is a local operation as only the transitions of δ_{DH} having as source or as destination the state (q, p) have to be recomputed from H. Concerning the refined abstract timed automaton $TH_{\mathfrak{P}'}$ resulting from a split of (q, p) into (q, p_1) and (q, p_2), if $I_{(q,p)} = [t_{min}, t_{max}]$ the region time intervals $I_{(q,p_1)} = I_{(q,p_2)} = [0, t_{max}]$ can be adopted in first approximation as they are safe, but in general new tighter time bounds are recomputed from H for the sojourn duration in the regions p_1 and p_2.

5.2 Refinement guided by reachability analysis

We give here some general mathematical properties, in particular about conservation of the connectivity property of the regions when following the solution flow, that are useful for the refinement process when guided by the dynamics of the hybrid system and reachability conditions.

Reachability from a connected set. Let $CS = (X, S_0, F, Inv)$ be a continuous system with deterministic flow condition $F : \mathbf{X} \to \mathbb{R}^n$ and $\mathbf{K} \subset \mathbb{R}^n$ a set, such that the following hypotheses are verified:

- $\mathbf{K}$ is a connected and bounded closed (i.e., compact) set and $\forall \mathbf{x} \in \mathbf{K}, F(\mathbf{x}) \neq \mathbf{0}$.
- The flow solution function $x(t, \mathbf{x}_0)$ initially starting at $t = 0$ from $\mathbf{x}_0 \in \mathbf{K}$ is continuous with respect to $\mathbf{x}_0 \in \mathbf{K}$ and of class C^1 with respect to the time t (this property is true in the case of polynomial dynamics).

With these hypotheses, trajectories issued from $\mathbf{x}_0$ are continuous with respect to $\mathbf{x}_0$ (proof can be made using the uniform continuity deduced from the continuity as $\mathbf{K}$ is a compact set).
Let $\mathbf{y}$ be a reachable element from $\mathbf{K}$ and $x(t)$ the trajectory reaching $\mathbf{y}$ at time t_0. We define the successor trajectory $post(\mathbf{y})$ and the predecessor trajectory $pre(\mathbf{y})$ by:

$$post(\mathbf{y}) = \{\mathbf{x} \in \mathbf{X} \mid \exists\, t > t_0\ \mathbf{x} = x(t)\} \qquad pre(\mathbf{y}) = \{\mathbf{x} \in \mathbf{X} \mid \exists\, 0 < t < t_0\ \mathbf{x} = x(t)\} \tag{3}$$

We extend this definition to the set $\mathbf{K}$:

$$post(\mathbf{K}) = \bigcup_{k \in \mathbf{K}} post(k) \qquad pre(\mathbf{K}) = \bigcup_{k \in \mathbf{K}} pre(k) \tag{4}$$

With continuity argument from the hypotheses, we have the following result.

Theorem 5. *$post(\boldsymbol{K})$ and $pre(\boldsymbol{K})$ are connected sets.*

This result shows that our connectivity property assumed for all regions is conservative along trajectories (backward and forward) issued from set $\mathbf{K}$.

Reachability from initial state to guard set. Let H be a hybrid automaton with polynomial dynamics and $\mathbf{X}_0 = \{\mathbf{x} \mid (q_0, \mathbf{x}) \in S_0\}$ the set of its initial states in a mode q_0. We suppose we have built an abstract automaton of H with respect to a given partition. By construction of this partition, we consider that for each region p, all incoming trajectories must exit and become outgoing trajectories after having passed a bounded time in p. Then we can define $in(p)$ as the set of all trajectories restricted to p, which is equal to $post(\mathbf{X}_0) \cap p$. We define also the subset $pre(p)$ of the incoming points of $in(p)$ and the subset $post(p)$ of the outgoing points of $in(p)$. It can be demonstrated that $pre(p)$, $in(p)$ and $post(p)$ are unions of a finite number of connected sets. The proof can be made by induction starting from the initial set assumed to be defined by convex linear predicates and using the fact the dynamics is polynomial.
If we consider a guard set $G(\tau)$, assumed to be a convex linear predicate set, the set of outgoing points of the trajectories crossing p and verifying this guard is $post(p) \cap G(\tau)$ and it can be proved that this set is again a union of a finite number of connected sets. So the time passed by a trajectory inside p with outgoing points verifying $G(\tau)$ can be represented by a union of finite number of intervals, as internal trajectories form a union of finite number of connected sets (the set of instants passed in trajectories belonging to a connected set is a time interval if the trajectories are continuous). Outside this union of time intervals for internal trajectories in p, the guard $G(\tau)$ cannot be verified by outgoing points. This analysis will provide important results for the diagnosability verification.

6 CEGAR Adaptation for Hybrid Automata Diagnosability Verification

Note that to verify diagnosability, one should simultaneously analyze two executions to check whether or not they have the same observations while only one contains the considered fault. Time constraints will be used explicitly in our abstraction. When an abstraction refinement is required, tighter time bounds will be obtained over the new regions of the refined decomposition. We will now adapt CEGAR [8], originally proposed to verify safety properties, to diagnosability verification of an hybrid automaton H. The three main steps of this scheme are: 1) *Diagnosability checking* of a timed abstract automaton of H using the twin plant method, which generates a counterexample $C.E$ when diagnosability is not verified; 2) *Validation of the C.E* by checking at the level of H whether it is valid or spurious; 3) *Refinement* of the timed abstract automaton by using a finer hybrid state space decomposition.

6.1 CEGAR for diagnosability verification

Verifying diagnosability of a hybrid automaton by checking it on abstractions of this automaton is justified because if the diagnosability property is verified for an abstraction, then it is verified also for the concrete hybrid system. This can be established by showing that a concrete counterexample of diagnosability lifts up into an abstract counterexample of diagnosability. Actually, given a hybrid automaton $H = (Q, X, S_0, \Sigma, F, Inv, T)$, two executions $h, h' \in [[H]]$, $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1)...(q_i, \mathbf{x}_i) \xrightarrow{l_i} ...$, $h' = (q'_0, \mathbf{x}'_0) \xrightarrow{l'_0} (q'_1, \mathbf{x}'_1)...(q'_i, \mathbf{x}'_i) \xrightarrow{l'_i} ...$ are called a counterexample of diagnosability in H with respect to the fault F if they constitute a critical pair (cf. Definition 7). We will denote each state $(q_i, \mathbf{x}_i)$ by s_i and $(q'_i, \mathbf{x}'_i)$ by s'_i. We assume that the flow condition F is a singleton function (deterministic). Then, from Theorem 4, given a timed abstract automaton $TH_{\mathfrak{P}}$ of H with abstraction function $\alpha^t_{\mathfrak{P}}$, h and h' are mapped by $\alpha^t_{\mathfrak{P}}$ into executions $\hat{h}, \hat{h}' \in [[TH_{\mathfrak{P}}]]$, $\hat{h} = \hat{s}_0 \xrightarrow{\hat{l}_0} \hat{s}_1...\hat{s}_i \xrightarrow{\hat{l}_i} ...$, $\hat{h}' = \hat{s}'_0 \xrightarrow{\hat{l}'_0} \hat{s}'_1...\hat{s}'_i \xrightarrow{\hat{l}'_i} ...$ and, as (h, h') is a critical pair in H with respect to F, so is $(\hat{h}, \hat{h}')$ in $TH_{\mathfrak{P}}$, which establishes thus a counterexample of diagnosability in $TH_{\mathfrak{P}}$: $C.E = (\hat{h}, \hat{h}')$. This proves the following result.

Theorem 6. *Given a hybrid automaton H with singleton flow condition, a timed abstract automaton $TH_{\mathfrak{P}}$ of H with abstraction function $\alpha^t_{\mathfrak{P}}$ and a modeled fault F in H, if F is diagnosable in $TH_{\mathfrak{P}}$ then F is diagnosable in H.*

Now, with this result, algorithm 1 illustrates the CEGAR scheme adaptation for hybrid automata diagnosability verification.

Algorithm 1 CEGAR scheme for hybrid automata diagnosability verification

```
Input: hybrid automaton H; considered fault F
TH ← Initial Timed Abstract Automaton of H
C.E ← Diagnosability Check (TH, F)
while C.E ≠ ∅ do
  if Validate(C.E, H) then
    return "F is not diagnosable in H"
  else
    TH ← Refine(TH, C.E, H)
    C.E ← Diagnosability Check(TH, F)
  end if
end while
return "F is diagnosable in H"
```

6.2 Twin plant based diagnosability checking

Diagnosability checking of a discrete event system, modeled as an automaton, based on the twin plant method [13, 21] is polynomial in the number of states (actually it has been proved it is NLOGSPACE-complete [16]). The idea is to construct a non-deterministic automaton, called pre-diagnoser or verifier, that preserves only all observable information and appends to every state the knowledge about past fault occurrence. The twin plant is then obtained by synchronizing the pre-diagnoser with itself based on observable events to get as paths in the twin plant all pairs of executions with the same observations in the original system. Each state of the twin plant is a pair of pre-diagnoser states that provide two possible diagnoses. A twin plant state is called an ambiguous one if the corresponding two pre-diagnoser states give two different diagnoses (presence for one and absence for the other of a past fault occurrence). A

critical path is a path in the twin plant with at least one ambiguous state cycle. It corresponds to a critical pair and it has thus been proved that the existence of a critical path is equivalent to non-diagnosability. The twin plant method has been adapted to be applied to timed automata [19], where a twin plant is constructed in a similar way except that the time constraints of two executions are explicitly taken into account using clock variables. The idea is to verify whether the time constraints can further distinguish two executions by comparing the occurrence time of observable events. The definition of a critical path in the twin plant is analog, except that ambiguous state cycle is replaced by infinite time ambiguous path.

Lemma 1. *A fault is diagnosable in a timed automaton iff its twin plant contains no critical path [19].*

For timed automata, checking diagnosability is PSPACE-complete.

6.3 Counterexample validation or refutation

After applying the twin plant method on a timed abstract automaton $TH_{\mathfrak{P}}$ of H [19], suppose that a critical pair $C.E = (\hat{h}, \hat{h}')$ is returned (if not, it means that $TH_{\mathfrak{P}}$, and thus H, is diagnosable). Whether we find or not two concrete executions $h, h' \in [[H]]$ whose abstractions by $\alpha^t_{\mathfrak{P}}$ are $\hat{h}, \hat{h}'$ and form a critical pair in H decides if $C.E$ is validated or refuted.

Validated counterexample: If it exists $h, h' \in [[H]]$, whose abstractions by $\alpha^t_{\mathfrak{P}}$ (according to Theorem 4) are $\hat{h}, \hat{h}'$ and such that $Obs(h) = Obs(h')$, then (h, h') constitutes a concrete counterexample realizing $C.E$ and proves thus the non-diagnosability of H. If not, the abstract counterexample $C.E$ is said spurious. In practice, this step involves computing reachable sets of states using safe over approximations such as ellipsoids and zonotopes for complex dynamics or hypercubes for simpler ones [1, 5]. Obviously, due to inherent undecidability of reachability problem at the concrete level of the hybrid automaton, it can happen that a concrete critical pair realizing $C.E$ does actually exist but this existence will not be proved and $C.E$ will be declared spurious, with new chance to discover a concrete critical pair at the next refinement loop.

Refuted counterexample: In case of spurious $C.E = (\hat{h}, \hat{h}')$, the idea is to construct longest finite executions $h, h' \in [[H]]$, that abstract by $\alpha^t_{\mathfrak{P}}$ into finite prefixes of $\hat{h}, \hat{h}'$ and such that $Obs(h) = Obs(h')$. The fact they cannot be extended means that $\forall \overline{h} \in [[H]]/h, \overline{h'} \in [[H]]/h'$ one step executions, either (i) $\hat{s}_{|h|+1} \neq \alpha^t_{\mathfrak{P}}(s_{|h|+1})$ (or $\hat{s}'_{|h'|+1} \neq \alpha^t_{\mathfrak{P}}(s'_{|h'|+1})$) or (ii) $\hat{l}_{|h|} \neq l_{|h|}$ (or $\hat{l}'_{|h|} \neq l'_{|h|}$) or (iii) $Obs(\overline{h}) \neq Obs(\overline{h'})$. In this case, $s^{reach}_{|h|+1}$ and $s'^{reach}_{|h'|+1}$ are returned, that represent the two sets of reachable concrete states that are the first ones to disagree with the abstract $C.E$. We summarize below the reasons resulting in the $C.E$ being spurious.

(i) *Spurious state reachability*. There is no concrete execution in H whose abstraction is one of $\hat{h}$ or $\hat{h}'$, as one of the set of states of H whose abstraction is an abstract state $\hat{s}_i$ or $\hat{s}'_i$ is not reachable in H starting from the initial states of H. Note that care will have to be taken when refining $TH_{\mathfrak{P}}$ (see next subsection). E.g., a possible case is that there exist two executions (h, h') reaching (s_1, s'_1) and then (s_2, s'_2) but not reaching (s_3, s'_3) (and none passing by (s_1, s'_1) and (s_2, s'_2) reaches (s_3, s'_3)), and two other executions (u, u') reaching (s_2, s'_2) from $(s, s') \neq (s_1, s'_1)$, with $\alpha^t_{\mathfrak{P}}(s) = \alpha^t_{\mathfrak{P}}(s_1)$ and $\alpha^t_{\mathfrak{P}}(s') = \alpha^t_{\mathfrak{P}}(s'_1)$, and then reaching (s_3, s'_3), all with time periods compatible to those of the abstract executions. If the refined model simply eliminated the transition from $\hat{s}_2$ to $\hat{s}_3$ or from $\hat{s}'_2$ to $\hat{s}'_3$ then it could no longer be considered an abstraction of H, since some concrete execution in H would have no abstract counterpart. Thus the refinement has to apply the split operation as previously described, so that preserving the abstraction while eliminating the spurious counterexample.

(ii) *Spurious time constraints satisfaction*. The abstract critical pair, when considered timeless, owns a concrete critical pair realization in H but none verifying the time bounds imposed by the abstract timed automaton. In this case it is not a spurious state reachability problem but a spurious timed state reachability problem. Actually the time constraints of the abstract critical pair cannot be satisfied by any concrete critical pair realizing it in H.

(iii) *Spurious observation undistinguishability*. The two executions of the abstract critical pair share the same observations (observable events with their occurrence times and snapshots of the values of observable continuous variables at arrival times in each abstract state) but actually any two concrete executions realizing this critical pair in H do distinguish themselves by the observation of some observable continuous variable.

6.4 Refinement of the abstraction

If it reveals that abstract counterexample $C.E = (\hat{h}, \hat{h}')$ is spurious, then one refines the timed abstract automaton $TH_{\mathfrak{P}}$ to get $TH_{\mathfrak{P}'}$, guided by the information from $C.E$. The first step is analyzing $C.E$ to identify the reasons why it is spurious (as classified previously). The idea is to avoid getting relatively close spurious abstract counterexample when applying twin plant method on the refined timed abstract automaton $TH_{\mathfrak{P}'}$. The refinement procedure is described as follows and will be illustrated on our example in the next section.

1. Suppose that $C.E$ is refuted due to an illegal stay, i.e., the corresponding invariant is not respected. The consequence could be $s^{reach}_{|h|+1} = \emptyset$, i.e., an illegal transition. To eliminate such spurious counterexample next time, one can partition the region containing $\hat{s}_{|h|}$ to get a new region representing the legal stay such that the refinement can be done based on this partition. The idea is to eliminate illegal (unobservable) transitions between the new region and others by tightening time constraints. In a similar way, one can handle spurious counterexamples with illegal transitions due to the unsatisfiability of the corresponding guards by the evolution of continuous variables, but with a legal stay this time.

2. Suppose that the refutation of $C.E$ is due to different observations from $s_{|h|+1}$ and $s'_{|h'|+1}$ without reachability problem. The idea is to calculate the exact moment, denoted $t_{spurious}$, before which it is still possible to get the same observations while after it the observations will diverge. With $t_{spurious}$, one can partition $\hat{s}_{|h|+1}$ and $\hat{s}'_{|h'|+1}$ to get a new region whose legal stay is limited by $t_{spurious}$ and transition to another region gives birth to a new refined observation by means of an observable continuous variable if any.

7 Case Study Example

We consider the previously presented thermostat example in Figure 1. The two events B_{on} and B_{off} are assumed to be observable, allowing one to witness mode changes, as is also the continuous variable x. We introduce a fault F, representing a defect in the heater efficiency producing overheat, by a parametric change of a constant (from 100 to 120) in the flow condition in the mode on, as shown in Figure 2. We model the fault occurrence as a transition $\tau_F = (on, F, on^F)$ with unobservable event F allowing mode change from the nominal mode on to the faulty mode on^F.

Initial Abstraction. We consider an initial decomposition $\mathfrak{P} = \{\{p_{off}\}, \{p_{on}\}, \{p_{off^F}\}, \{p_{on^F}\}\}$ of the hybrid state space where each region is $\mathbb{R}$. We suppose that the time interval for each region p is $I_p = [0, +\infty)$, in other words the initial abstraction contains no time constraint. The diagnosability

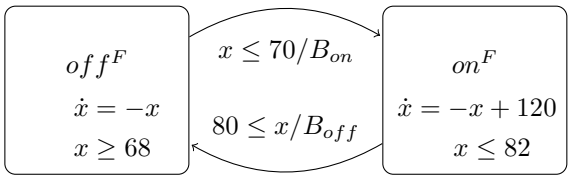

Figure 2: Hybrid automaton modeling faulty modes with a possible overheating fault F

check of this initial abstraction using the twin plant method generates a $C.E = (\hat{h}, \hat{h'})$:

$\hat{h} = (off, p_{off}) \xrightarrow{0.5} (off, p_{off}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.5} (on, p_{on}) \xrightarrow{B_{off}} (off, p_{off})...$
$\hat{h'} = (off, p_{off}) \xrightarrow{0.5} (off, p_{off}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.4} (on, p_{on}) \xrightarrow{F} (on^F, p^F_{on}) \xrightarrow{0.1} (on^F, p^F_{on})$
$\xrightarrow{B_{off}} (off^F, p^F_{off})...$

The attempt to compute a concrete critical pair validating $C.E$ results in the following sets of executions:

$$\{h\} = (off, [80, 90]) \xrightarrow{0.5} (off, [48.5, 54.58] : invalid)$$
$$\{h'\} = (off, [80, 90]) \xrightarrow{0.5} (off, [48.5, 54.58] : invalid)$$

The abstract executions $\hat{h}, \hat{h'}$ are not concretely feasible because if the system stays in mode off for 0.5 time units then the state invariant is no longer true, i.e., if B_{on} is observed then the duration of stay in off has to be smaller than 0.5. To prevent future similar spurious counterexamples, a refinement is applied to the initial abstraction. The refined model considers new regions in mode off: $p_{off_1} = [80, 90]$ and $p_{off_2} = [68, 80)$. The computation of the time intervals relative to each region gives: $I_{(off,[68,80))} = [0, 0.16]$ and $I_{(off,[80,90])} = [0, 0.12]$. The refined abstraction will encode these time constraints and ensure that a set of similar counterexamples (including this one) are eliminated. Regions that are not reachable will be eliminated, such as $[0, 68)$.

Second Abstraction. A second $C.E = (\hat{h}, \hat{h'})$ is generated from the refined timed automaton with:

$\hat{h} = (off, p_{off_1}) \xrightarrow{0.08} (off, p_{off_1}) \xrightarrow{\epsilon} (off, p_{off_2}) \xrightarrow{0.07} (off, p_{off_2}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.5}$
$(on, p_{on}) \xrightarrow{B_{off}} (off, p_{off})...$
$\hat{h'} = (off, p_{off_1}) \xrightarrow{0.08} (off, p_{off_1}) \xrightarrow{\epsilon} (off, p_{off_2}) \xrightarrow{0.07} (off, p_{off_2}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.4}$
$(on, p_{on}) \xrightarrow{F} (on^F, p^F_{on}) \xrightarrow{0.1} (on^F, p^F_{on}) \xrightarrow{B_{off}} (off^F, p^F_{off})...$

Computing corresponding concrete sets of executions of this $C.E$ gives:

$\{h\} = (off, [80, 90]) \xrightarrow{0.15} (off, [69, 77]) \xrightarrow{B_{on}} (on, [69, 70]) \xrightarrow{0.5} (on, [80.5, 81.8])...$
$\{h'\} = (off, [80, 90]) \xrightarrow{0.15} (off, [69, 77]) \xrightarrow{B_{on}} (on, [69, 70]) \xrightarrow{0.4} (on, [79, 79.9]) \xrightarrow{F}$
$(on^F, [79, 79.9]) \xrightarrow{0.1} (on^F, [82.9, 83.7])...$

In this case, the second $C.E$ is also considered as spurious because, given the time constraints, the two concrete executions differ in the observations of x since the last regions are disjoint, i.e., $[80.5, 81.8] \cap$

$[82.9, 83.7] = \emptyset$. The counterexample analysis could identify the time boundary $t_{spurious}$ up to which the observations could be the same for at least two concrete executions, and after which the critical pair becomes spurious. In our example, suppose F occurred at t_F and $x \in [a, b]$ at this time, then $t_{spurious}$ is the time instant from which faulty and nominal sets of executions are disjoint:

$$t_{spurious} = ln\left(\frac{b - a + 20}{20}\right) + t_F$$

For the second spurious $C.E$, $t_F = 0.4$ and $t_{spurious} - t_F = 0.044$. The two concrete nominal and faulty executions originating from $(on, [79, 79.9])$ will be in the following temperature range after 0.044 time units: $x \in [79.9, 80.7]$ in the mode on and $x \in [80.7, 81.6]$ in the mode on^F. Hence, at any future time, the observations are different. By incorporating the time constraint in the refined abstraction, we ensure that counterexamples that are spurious because of disjoint observations including the previous one cannot be generated again.

8 Conclusion

We were interested in verifying diagnosability, an important property in safety analysis at design stage, on a hybrid system, based on discrete abstractions of this system, for which checking this property is decidable and which guarantee that the property is satisfied at the concrete hybrid level if it is satisfied at the abstract level. We presented elements from the literature regarding hybrid automata abstractions, however few works handle diagnosability verification, as this property deals with a pair of trajectories and partial observations of the system and is thus more complex to check than reachability. In order to handle time constraints at the abstract level, we chose abstractions of the hybrid automaton as timed automata, related to a decomposition of the state space into geometric regions, the abstract time constraints coming from the estimation of the sojourn time of trajectories in each region. Thus the abstractions over-approximate the regions of interest to which are added time constraints obtained from the dynamics of the concrete system. We adapted a CEGAR scheme for hybrid systems diagnosability verification, based on the counterexample provided at the abstract level by the twin plant based diagnosability checking when diagnosability is proved to be unsatisfied. We presented situations for which the produced counterexample is spurious and a refinement in finer regions and tighter time constraints is then required.

We plan in future work to automate the process of analyzing an abstract counterexample and choosing a decomposition-based abstraction refinement to eliminate it when it is spurious and to test the whole method on real examples. A fundamental and essential future work perspective is to provide a general algorithm for diagnosability verification with ϵ-precision, for ϵ arbitrary small (for a given metric to be defined). This will ensure theoretical termination of the algorithm, as the number of refinement steps to reach the precision will then be finite. And this is actually justified in practice because both model parameters and observations cannot be infinitely accurate, thus the value ϵ for the precision would come from the precision of the model parameters and of the measurements, in space and time.

References

[1] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear analysis: hybrid systems*, 4(2):233–249, 2010.

[2] Mehdi Bayoudh, Louise Travé-Massuyès, and Xavier Olive. Hybrid systems diagnosability by abstracting faulty continuous dynamics. In *Proceedings of the 17th International Workshop on Principles of Diagnosis DX'06*, pages 9–15, 2006.

[3] Mehdi Bayoudh, Louise Travé-Massuyès, and Xavier Olive. Active diagnosis of hybrid systems guided by diagnosability properties. In *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, volume 42, pages 1498–1503. Elsevier, 2009.

[4] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.

[5] Oleg Botchkarev and Stavros Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Proceedings of the 3rd International Workshop on Hybrid Systems: Computation and Control HSCC'00*, volume 1790 of *LNCS*, pages 73–88. Springer, 2000.

[6] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *Proceedings of the International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems FTRTFT'98*, LNCS, pages 298–302. Springer, 1998.

[7] A. Cimatti, C. Pecheur, and R. Cavada. Formal Verification of Diagnosability via Symbolic Model Checking. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence IJCAI-03*, pages 363–369, 2003.

[8] Clarke Edmund, Fehnker Ansgar, Han Zhi, Krogh Bruce, Stursberg Olaf, and Theobald Michael. Verification of hybrid systems based on counterexample-guided abstraction refinement. In H. Garavel and J. Hatcliff, editors, *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS-2003*, volume 2619 of *LNCS*, pages 192–207. Springer, 2003.

[9] V. Germanos, S. Haar, V. Khomenko, and S. Schwoon. Diagnosability under weak fairness. In *Proceedings of the 14th International Conference on Application of Concurrency to System Design ACSD'14*, Tunis, Tunisia, june 2014.

[10] Alban Grastien, Louise Travé-Massuyès, and Vincenc Puig. Solving diagnosability of hybrid systems via abstraction and discrete event techniques. In *Proceedings of the 27th International Workshop on Principles of Diagnosis DX'16*, 2016.

[11] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pages 278–292. IEEE Computer Society Press, 1996.

[12] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.

[13] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.

[14] Oded Maler and Grégory Batt. Approximating continuous systems by timed automata. In *Proceedings of the 1st International Workshop on Formal Methods in Systems Biology FSMB-08*, volume 5054 of *LNCS*, pages 77–89. Springer, 2008.

[15] Y. Pencolé. Diagnosability Analysis of Distributed Discrete Event Systems. In *Proceedings of the 16th European Conference on Articifial Intelligence ECAI-04*, pages 43–47. Nieuwe Hemweg: IOS Press., 2004.

[16] Jussi Rintanen. Diagnosers and diagnosability of succinct transition systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 538–544, Hyderabad, India, 2007.

[17] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of Discrete Event System. *Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[18] A. Schumann and J. Huang. A Scalable Jointree Algorithm for Diagnosability. In *Proceedings of the 23rd American National Conference on Artificial Intelligence AAAI-08*, pages 535–540. Menlo Park, Calif.: AAAI Press., 2008.

[19] Stavros Tripakis. Fault diagnosis for timed automata. In *Proceedings of the International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems FTRTFT-02*, volume 2469 of *LNCS*, pages 205–221. Springer, 2002.

[20] L. Ye and P. Dague. Diagnosability Analysis of Discrete Event Systems with Autonomous Components. In *Proceedings of the 19th European Conference on Artificial Intelligence ECAI-10*, pages 105–110. Nieuwe Hemweg: IOS Press., 2010.

[21] T.-S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event

systems. *IEEE Transactions on Automatic Control (TAC)*, 47(9):1491–1495, 2002.