Chapter 11
Abstractions Refinement for Hybrid Systems Diagnosability Analysis

Hadi Zaatiti, Lina Ye, Philippe Dague, Jean-Pierre Gallois, and Louise Travé-Massuyès

11.1 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

H. Zaatiti
CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, Gif-sur-Yvette, France

LRI, Univ. Paris-Sud & CNRS, Univ. Paris-Saclay, Gif-sur-Yvette, France
e-mail: hadi.zaatiti@cea.fr; hadi.zaatiti@lri.fr

L. Ye
CentraleSupélec, Univ. Paris-Saclay, Gif-sur-Yvette, France

LRI, Univ. Paris-Sud & CNRS, Univ. Paris-Saclay, Gif-sur-Yvette, France
e-mail: lina.ye@lri.fr

P. Dague (✉)
LRI, Univ. Paris-Sud & CNRS, Univ. Paris-Saclay, Gif-sur-Yvette, France
e-mail: philippe.dague@lri.fr

J. -P. Gallois
CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, Gif-sur-Yvette, France
e-mail: jean-pierre.gallois@cea.fr

L. Travé-Massuyès
LAAS-CNRS, Univ. de Toulouse, Toulouse, France
e-mail: louise@laas.fr

M. Sayed-Mouchaweh (ed.), *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*, https://doi.org/10.1007/978-3-319-74962-4_11

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 automata [21, 35, 62]. 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 [16, 32, 48, 52, 53, 60]. However, most of the existing works are applied on discrete event systems.

In this chapter, we are concerned with abstractions oriented towards hybrid systems diagnosability checking. Our goal is to create discrete abstractions in order to verify, at design stage, if a fault that would occur at runtime could be unambiguously detected in finite time (or within a given finite time bound for bounded diagnosability) by the diagnoser using only the allowed observations. This verification can be done on the abstraction by classical methods developed for discrete event systems, which provides 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, 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 and we define several refinement strategies. Abstractions as timed automata are particularly studied as they allow one to capture qualitative temporal constraints [10, 13]. The chapter is organized as follows. We first present the hybrid automata formalism and define diagnosability for hybrid systems. We then introduce a formal framework for constructing hybrid automata abstractions while defining the refinement relation. Lastly, we detail the counterexample guided abstraction refinement (CEGAR) scheme adapted for diagnosability verification and a case study example illustrating this scheme.

11.2 Hybrid Dynamical Systems

In this section, we start with a brief general description of hybrid systems and then move on to propose a formal representation framework for hybrid automata that is adopted throughout the chapter. Later on, we provide an example of a practical system modeled as a hybrid automaton. Lastly, we introduce various classes of hybrid automata, among which timed automata are our primary interest.

11.2.1 Hybrid Automata Definition

Hybrid systems are dynamical systems that include discrete and continuous behaviors [38]. **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 Automaton (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 $\mathbf{X} \subseteq \mathbb{R}^n$ represents the *continuous* part of H. $S = Q \times \mathbf{X}$ is the state space of H, whose elements, called states, are noted $(q, \mathbf{x})$ with q and $\mathbf{x}$ 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 $(q_0, \mathbf{x}_0)$.
- Σ is a finite set of events.
- $F : S \to 2^{\mathbb{R}^n}$ is a mapping assigning to each state $(q, \mathbf{x}) \in S$ a set $F(q, \mathbf{x}) \subseteq \mathbb{R}^n$ constraining the time derivative $\dot{\mathbf{x}}$ of the continuous part of the mode q by $\dot{\mathbf{x}} \in F(q, \mathbf{x})$. If there is no uncertainty on the derivative, then F is a function $S \to \mathbb{R}^n$ specifying the flow condition $\dot{\mathbf{x}} = F(q, \mathbf{x})$ in each mode q (the dynamics in each mode is thus given by a set of n first-order ordinary differential equations (ODEs)).
- $Inv : Q \to 2^{\mathbf{X}}$ assigns to each mode q an invariant set $Inv(q) \subseteq \mathbf{X}$, 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 $\{\mathbf{x} \mid (q, \mathbf{x}) \in S_0\} \subseteq Inv(q)$.
- $T \subseteq S \times \Sigma \times S$ is a relation capturing discontinuous state changes, i.e., instantaneous discrete transitions from one mode to another one. Precisely, $t = (q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T$ represents a transition whose source and destination states are $(q, \mathbf{x})$ with $\mathbf{x} \in Inv(q)$ and $(q', \mathbf{x}')$ with $\mathbf{x}' \in Inv(q')$, respectively, and labeled by the event σ. It represents a jump from $\mathbf{x}$ in mode q to $\mathbf{x}'$ 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: $\forall (q, \mathbf{x}), (q', \mathbf{x}') \in S$, $\forall \sigma \in \Sigma$, $((q, \mathbf{x}, \sigma, q', \mathbf{x}') \in T \Leftrightarrow \tau = (q, \sigma, q') \in \delta \wedge \mathbf{x} \in G(\tau) \wedge \mathbf{x}' \in R(\tau)(\mathbf{x}))$.

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 $\mathbf{x}' \in R(\tau)(\mathbf{x})$. We will make use equally of both representations.

Guards in any mode q will be assumed non-intersecting: $\forall q \in Q, \forall \tau_1 = (q, \sigma_1, q_1) \in \delta, \forall \tau_2 = (q, \sigma_2, q_2) \in \delta, (\tau_1 \neq \tau_2 \Rightarrow G(\tau_1) \cap G(\tau_2) = \emptyset)$. Thus, at any moment of its continuous evolution in a mode q, the system may jump to at most one another mode and by a unique event. Nevertheless, a HA is generally non-deterministic: the continuous dynamics in each mode may be non-deterministic, the moment where a jump occurs is non-deterministic (as long as $Inv(q)(\mathbf{x})$ and $G(\tau)(\mathbf{x})$ are true, where q is the source mode of the mode transition τ, the system may continue to continuously evolve in q or make the transition τ) and the reset after a jump may be non-deterministic.

11.2.2 Modeling with Hybrid Automata

Hybrid automata represent an intuitive modeling framework. They are used in various domains to model complex hybrid systems. Here is a practical case where a hybrid automaton is used for modeling a system.

Example 1 A simple thermostat system maintains the temperature of an object quasi-constant by turning on and off a heater device. In practice such system contains at least, a temperature sensor, a heater device and logic control electronic circuits. The circuitry decides, given the actual measured temperature of the object, to activate or not the heater. A hybrid automaton $H = (Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$ models the behavior of such system (see graphical representation on Fig. 11.1):

- $Q = \{on, off\}, X = \{x\}, S_0 = (off, [80, 90])$
- $\Sigma = \{B_{on}, B_{off}\}, F(on) = \{\dot{x} = -x + 100\}, F(off) = \{\dot{x} = -x\}$
- $\delta = \{\tau_1 = (off, B_{on}, on), \tau_2 = (on, B_{off}, off)\}, Inv(off) = x \geq 68, Inv(on) = x \leq 82$
- $G(\tau_1) = x \leq 70, G(\tau_2) = x \geq 80, R(\tau_1) = R(\tau_2) = (\mathbf{x} = \mathbf{x}')$

The assigned hybrid automaton H is one dimensional, where x represents the sensed temperature.

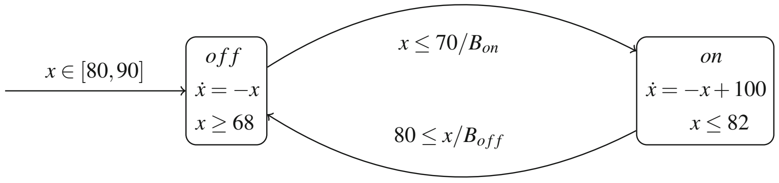


Fig. 11.1 One-dimensional hybrid automaton modeling a thermostat

11.2.3 Hybrid Automata Semantics

We denote by $q_0, q_1, \ldots$ the modes of Q, and by $x_1, x_2, \ldots, x_n$ 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 $L = \Sigma \cup \mathbb{R}_+$: $(q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1) \ldots (q_i, \mathbf{x}_i) \xrightarrow{l_i} \ldots$ such that $(q_0, \mathbf{x}_0) \in S_0$ and, for any two successive states $(q_i, \mathbf{x}_i) \xrightarrow{l_i} (q_{i+1}, \mathbf{x}_{i+1})$ in the sequence, one of the following is true:

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

In the first case, the system executes a discrete transition (also called discrete jump) $\tau_i = (q_i, \sigma_i, q_{i+1})$ from the mode q_i to the destination mode q_{i+1}. Such a transition is possible (enabled) as soon and as long as $\mathbf{x}_i \in G(\tau_i)$. After the jump, the system may follow the new dynamics given by $F(q_{i+1})$, starting from the continuous state $\mathbf{x}_{i+1} \in R(\tau_i)(\mathbf{x}_i)$. Notice that no time elapses during a discrete jump, which is instantaneous. In the second case, the system performs a continuous transition (also called continuous flow) of duration d_i inside the mode q_i, constrained by the dynamics $F(q_i)$ and the invariant set $Inv(q_i)$. The sequence $h = (\mathit{off}, 80) \xrightarrow{0.15} (\mathit{off}, 69) \xrightarrow{B_{on}} (on, 69) \xrightarrow{0.5} (on, 81) \xrightarrow{B_{\mathit{off}}} (\mathit{off}, 81) \ldots$ is valid for the thermostat example (Fig. 11.1), thus $h \in [[H]]$. 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: $time(h) = \sum d_i$.

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 $\rightarrow \subseteq \overline{S} \times L \times \overline{S}$ 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 $S_H^t = (\overline{S}, \overline{S}_0, L, \rightarrow)$, i.e., $[[H]]$ is the set of all paths of S_H^t issued from an initial state. S_H^t, 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_H^t, called the **timeless transition system of** H, is obtained by ignoring also the duration of flows and thus defined as $S_H = (\overline{S}, \overline{S}_0, \Sigma \cup \{\varepsilon\}, \rightarrow)$, obtained from S_H^t 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 $(q_i, \mathbf{x}_i) \xrightarrow{\varepsilon} (q_{i+1}, \mathbf{x}_{i+1})$, that can be considered as a silent transition. It has infinite set of states but finite set of transition 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_H^t (resp. S_H). Conversely, any path in S_H^t (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_H^t (resp. S_H) abstracts a concrete behavior of H. In this latter case, there is thus no spurious abstract behavior in S_H^t (resp. S_H), which expresses faithfully the behavior of H.*

11.2.4 Hybrid Automata Classes and Particular Cases

Definition 3 (Discrete Automaton (DA)) It is the case where there is no continuous part. Thus, a (finite) discrete automaton (DA) is a tuple $D = (Q, Q_0, \Sigma, \delta)$ where Q is a finite set of discrete states (modes), $Q_0 \subseteq Q$ is the set of initial states, Σ is a finite set of events, and $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions of the form $\tau = (q, \sigma, q')$.

The semantics $[[D]]$ of D is given by the set of sequences (called paths) made up of successive states transitions labeled by events and rooted in an initial state. The trace of such a path is the word in $\Sigma^\star$ whose letters are the successive labels of the path.

Definition 4 (Continuous System (CS)) It is the case where there is no discrete part. Thus, an n-dimensional continuous system (CS) is a particular hybrid automaton C with only one mode ($|Q| = 1$) and $\Sigma, T = \emptyset$ (and thus δ, G, R too). It can thus be denoted as $C = (X, S_0, F, Inv)$ with $S_0 \subseteq Inv$.

The semantics $[[C]]$ of C is the set of all time labeled sequences of continuous states, rooted in an initial state, corresponding to the continuous transitions of a hybrid automaton, constrained by the dynamics F and the invariant set Inv.

The form of the dynamics F determines primarily the class of the hybrid automaton. 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 [39]. We will present some classes of hybrid automata starting from the particular to the more general classes.

Timed Automata 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 5 (Timed Automaton (TA)) A timed automaton (TA) is a hybrid automaton $T = (Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$ such that:

- $\mathbf{X} = \mathbb{R}^n_+$.
- $S_0 = Q_0 \times \{\mathbf{0}\}$.
- $\forall q \in Q \ F(q, .) = \mathbf{1}$, 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 $\mathbf{X}$. We require $\mathbf{0} \in Inv(q_0)$.
- $G : \delta \to C(X)$ associates to each discrete transition (q, σ, q') a rectangle guard in $Inv(q)$.
- $\forall \tau \in \delta \exists Y(\tau) \subseteq X \forall \mathbf{x} \in G(\tau) R(\tau)(\mathbf{x}) = \{\mathbf{x}'\}$ with $\mathbf{x}'_i = 0$ if $x_i \in Y(\tau)$ and $\mathbf{x}'_i = \mathbf{x}_i$ 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 $(q_0, \mathbf{x}_0) \xrightarrow{d_1} (q_0, \mathbf{x}_0 + d_1) \xrightarrow{\sigma_1} (q_1, \mathbf{x}_1) \xrightarrow{d_2} \ldots$ whose trace $trace(h)$ is the timed word $d_1 \sigma_1 d_2 \ldots \in \mathbb{R}_+(\Sigma \mathbb{R}_+)^*$ and duration is $time(h) = \sum d_i$.

The class of timed automata is particularly interesting as the reachability and language emptiness problems are decidable for that class [2]. Actually, decidability still holds for the larger class of **rectangular automata (RA)**, where the unique flow condition $F(.,.)$, the same for all modes, is given by a rectangle in $\mathbb{R}^n$ (instead of the singleton $\mathbf{1}$), $Init(q)$ is a bounded rectangle and $R(\tau)(\mathbf{x}) = \{\mathbf{x}' \mid \mathbf{x}'_i \in I_i$ if $x_i \in Y$ and $\mathbf{x}'_i = \mathbf{x}_i$ else$\}$, where the I_i's are bounded intervals depending only on τ [38]. And thus holds for the subclass of **singular automata**, where the flow rectangle is reduced to a singleton, whose timed automata are a particular case.

But decidability does not hold any more if the flow condition is allowed to change from one mode to another one. The simplest example of this is the generalization of timed automata where we allow the presence of stopwatches. A **stopwatch** is a clock which can switch from active (turned on) to inactive (turned off), or vice versa, when transiting between two modes. The generalized flow condition is thus given by: $\forall q \in Q \, \exists \mathbf{c} \in \{0,1\}^n \; F(q,.) = \mathbf{c}$, which means that the dynamics of clocks evolution in each mode q is given by $\dot{x}_i = 1$ for those clocks active in q and $\dot{x}_i = 0$ for those clocks inactive in q. Thus, during inactivity, a stopwatch holds its last valuation when it was active (or 0 in case of reset). It happens that reachability decidability does not hold for a generalized timed automaton if only one clock is allowed to be a stopwatch, i.e., when the flow conditions are not independent of the mode (and thus also for the more general classes of **multisingular automata**, where the flow singletons depend on the mode, and of the **multirectangular automata**, where the flow rectangles depend on the mode). Notice nevertheless that, allowing changes of flow conditions with changes of modes may remain manageable if, e.g., we require a reset of the variables concerned when it occurs. That is how **initialized multi-rectangular automata**, i.e., where for each discrete jump, each variable whose flow interval is changed in this jump has to be reset (reinitialized), can be translated to rectangular automata. Another case where decidability is lost in general for a hybrid system (but not for a timed automaton, for which it does not change the expressivity) is when the set $C(X)$ of constraints is extended to contain primitive constraints of the form $(x_i - x_j) \; op \; c_{ij}$, i.e., if variables are not pairwise independent (and thus also for the class of **triangular automata** which generalize rectangular automata by adding such constraints). **Linear automata** generalize both multirectangular and triangular automata by allowing sets $F(q), Init(q), Inv(q), G(\tau)$ to be any convex polyhedra in $\mathbb{R}^n$ (instead of just rectangles or triangles) and different flows conditions for different modes. And **polynomial automata** generalize linear automata by allowing those sets to be defined no longer by just linear constraints but by polynomial constraints.

11.3 Diagnosability of Hybrid Dynamical Systems

We will now introduce the model of hybrid systems that is used for diagnosability analysis and remind some methods from the literature aimed at verifying this property.

11.3.1 Hybrid System Model for Diagnosability Analysis

Fault diagnosis is a crucial and challenging task in the automatic control of complex systems, whose efficiency depends on the system property called diagnosability. This is a property describing the system ability to determine without ambiguity whether a fault of a given type has effectively occurred based on the observations. Diagnosability analysis has already received considerable attention in the literature over latest decades. However, most of the existing works refer to discrete event systems [16, 32, 34, 48, 52, 53, 60] with stochastic and fuzzy variants [41, 44, 54] or continuous systems [5, 15, 47, 57]. Diagnosability was also studied in the framework of decentralized and distributed architectures [46, 49, 50, 59].

But many modern technological processes exhibit both continuous evolution and discrete transitions whose combination is at the origin of complex behavior and important phenomena of such systems. To the best of our knowledge, very few works handle diagnosability of hybrid systems with satisfactory results.

As a first step, Travé-Massuyès et al. [56] proved that the existing definitions of diagnosability for discrete event systems and for continuous systems can be stated as a property of the system fault signatures, and a unified definition of diagnosability was established. However hybrid system diagnosability was not considered.

Among the contributions concerned with hybrid diagnosability, we can mention [11] that slightly modified the classical necessary and sufficient condition for diagnosability of a discrete event system of [52] and expressed it in terms of reachability. Reference [29] generalized this condition requiring more restrictive hypotheses. Despite the claim that the two above methods deal with hybrid systems, these works do not really account for the hybrid nature of the system as they use only a very high level discrete abstraction and ignore the continuous dynamics. On the other hand, in [19], diagnosability is expressed in terms of mode discernability (also called distinguishability by other authors) and is only based on the continuous dynamics.

Reference [8] was among the early works that coped with actual hybrid systems, introducing the idea to consider a hybrid model as a twofold mathematical object. A hybrid system is modeled as a hybrid automaton whose discrete states represent its operation modes for which the continuous dynamics are specified. The discrete event part (automaton) constrains the possible transitions among modes and is referred to as the *underlying DES*. The restriction of the hybrid system to the continuously-valued part of the model is defined as the *multimode system*.

Considering the analytical redundancy approach to define a set of residuals [33] for every mode, Bayoudh et al. [8] introduced the concept of *mode signature* which refines the classical concept of fault signature. Mode signatures determine mode distinguishability. The key idea of [8] is to abstract the continuous dynamics by defining a set of "diagnosis-aware" events, called *signature-events*, associated to mode signature changes across modes. Signature-events are used to enrich appropriately the underlying DES. The behavior of the abstract system is then modeled by a prefix-closed language over the alphabet enriched by these additional events. The finite state machine generating this language is called the *behavior automaton*. Based on the *abstract language*, the diagnosability analysis of the hybrid system is cast into a discrete event framework and standard methods of this field can be used.

The approach of [8] later consolidated in [6] can be compared to the approach proposed in [22, 23] which uses fault signatures to capture the continuous dynamics. The fault signatures of [22, 23] are based on fault transients and they directly express the expected dynamic behavior of measured variables after the fault abstracted in qualitative terms. The approach of [6, 8] differs in that it uses mode signatures that are specifically built for diagnosis, based on standard analytical redundancy residual methods of the FDI control field [27]. Its originality relies in that it proposes a way to integrate these methods with equally standard methods of the DES diagnosis field [63]. Bayoudh et al. [6, 8] adopt the diagnoser approach [52] because it has the advantage to also support straightforwardly online diagnosis. Diene et al. [25] repeats these ideas differing by the fact that the diagnoser is directly built from the underlying DES and mode distinguishability is used to cluster its state labels. This method leads to a so-called *clustered diagnoser*. Let us note that this method only applies to a restricted class of hybrid systems for which transitions triggered by continuous dynamics are not allowed.

Checking DES diagnosability with methods based on the construction of diagnosers has exponential complexity with the size of the underlying DES automaton. Hence, approaches based on *verifiers*, also known as *twin plant* approaches, are generally preferred. This is because, although a twin plant cannot be used for online diagnosis, it can be constructed in polynomial time. Methods integrating a twin plant approach with mode distinguishability checking for assessing hybrid system diagnosability are recent. The reader can refer to [26] as a first piece of work in this direction. Later, Grastien et al. [35] indicated that mode distinguishability could be complemented by another property of the continuous dynamics named *ephemerality*. Ephemerality states when the system cannot stay forever in a given set of modes. The continuous dynamics are hence abstracted remembering only these two pieces of information. In addition to this, Grastien et al. [35] checks diagnosability in an incremental way. It starts by generating the most abstract DES model of the hybrid system and checking diagnosability of this DES model. A "counterexample" that negates diagnosability is possibly provided based on the twin plant. The model is then refined to try to invalidate the counterexample and the procedure repeats as far as diagnosability is not proved. This approach hence uses just the necessary information about continuous dynamics, in an "on request" manner, hence making the best out of computation.

In the most recent literature concerned with hybrid system diagnosability like [35] and also [24], which characterizes the maximum delay for diagnosing faults given measurement uncertainty, abstraction is key. Abstraction is also at the core of other methods to check other properties of hybrid systems.

This is why this chapter reviews different ways of abstracting hybrid automata in the next section, then elaborates from the diagnosability procedure proposed in [35] that uses the counterexample guided abstraction refinement (CEGAR) as initially introduced in [3, 28]. The algorithmic basis refers to CEGAR considering that the verification problem for even very simple hybrid systems is undecidable [39]. The method abstracts the hybrid automaton and then refines the abstraction while being guided by a diagnosability counterexample found at this abstract level. A first discrete abstraction of the hybrid system is computed, diagnosability is then verified using classical discrete methods. The verification could either yield the abstraction as diagnosable, which infers the diagnosability property back to the hybrid system, or non-diagnosable with a generated counterexample that validates this decision. The produced counterexample is either present in the original system in which case the hybrid system is not diagnosable, or it is a spurious effect of the abstraction. In the latter case, the counterexample is analyzed and the current abstraction is refined according to this analysis and the diagnosability verification task is iterated.

11.3.2 Observations and Faults

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 the 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 Fig. 11.1, 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 is 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 6 (Partially Observable Hybrid Automaton (POHA)) A partially observable hybrid automaton (POHA) is a 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 7 (Execution (Timed) Observation) Given an execution $h \in [[H]]$ of a POHA H, $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1) \ldots (q_i, \mathbf{x}_i) \xrightarrow{l_i} \ldots$, with $l_i \in \Sigma \cup \mathbb{R}_+$, the (timed) observation of h is defined as $Obs(h) = \mathbf{x}_0^o, l_0^o, \mathbf{x}_1^o \ldots \mathbf{x}_i^o, l_i^o, \ldots$, where:

- $\mathbf{x}_i^o$ is obtained by projecting $\mathbf{x}_i$ 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 $Obs(h)$.

Note that all durations labels $l_i = d_i$ in h are present in $Obs(h)$. Thus, any observable event $l_i = \sigma_i$ in h is present in $Obs(h)$ together with its occurrence time, obtained by adding up all durations d_j in $Obs(h)$ from the origin up to the event σ_i. In the same way, any observable variable x has its value known in $Obs(h)$ at all those instants t obtained as the sums of consecutive durations in $Obs(h)$ from the origin. If t is the occurrence time of an (observable or unobservable) event σ and if x is reset by the discrete transition σ, then the value of x changes instantaneously after this transition and the new value will be noted $x^+(t)$ to distinguish it from the value $x(t)$ before the transition (a reset observable variable may thus identify the presence of an unobservable event). 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. We have thus defined what is the observation of a POHA H at the level of its timed transition system S_H^t. Defining its observation at the level of its timeless transition system S_H is similar, with $l_i \in \Sigma \cup \{\varepsilon\}$ and $l_i^o = l_i$ if $l_i \in \Sigma_o$ and removed otherwise. This means that the timeless observation is obtained from the timed observation $Obs(h)$ above by removing all durations d_i, keeping thus only observable events in Σ_o and values $\mathbf{x}_i^o$ of observable variables at each transition step as an ordered sequence without any occurrence time attached.

11.3.3 System Diagnosability Definition

As we just explained, 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 [52] introduced for discrete event systems (the bounded one and the unbounded one in terms of executions lengths). 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 $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1) \ldots (q_i, \mathbf{x}_i)$, the set of post-executions of h in $[[H]]$ is defined as $[[H]]/h = \{h' = (q_i, \mathbf{x}_i) \xrightarrow{l_i} \ldots \mid h.h' \in [[H]]\}$, where $h.h'$ is obtained by merging the final state of h and the first state of h', both should be the same.

Definition 8 ((Δ-)Faulty Executions) Given a hybrid automaton H and F a fault event, a faulty execution is an execution $h \in [[H]]$ such that $F \in trace(h)$. Thus

$h = h^F h'$ where h^F is the prefix of h whose last label is the first occurrence of F. We denote the *period from* (the first occurrence of) *fault F in h* by $time(h, F) = time(h')$. Given a positive real number $\triangle \in \mathbb{R}_+^*$, we say that at least Δ time units pass after the first occurrence of F in h, or, in short, that h is *Δ-faulty*, if $time(h, F) \geq \Delta$.

Definition 9 (Hybrid Automaton (Time Bounded and Unbounded) Diagnosability) Given $\triangle \in \mathbb{R}_+^*$, a fault F is said $\triangle$-diagnosable in a POHA H iff (we abbreviate $F \in trace(h)$ by $F \in h$)

$$\forall h \in [[H]](h\Delta\text{−faulty}$$
$$\Rightarrow \forall h' \in [[H]](Obs(h') = Obs(h) \Rightarrow F \in h')).$$

i.e.,

$$\forall h^F \in [[H]]\forall h \in [[H]]/h^F(time(h) \geq \triangle$$
$$\Rightarrow \forall h' \in [[H]](Obs(h') = Obs(h^F.h) \Rightarrow F \in h')).$$

A fault F is said diagnosable in H iff

$$\exists \triangle \in \mathbb{R}_+^*(F\triangle\text{-diagnosable in } H).$$

This definition states that F is $\triangle$-diagnosable (resp., diagnosable) iff, for each execution h^F in $[[H]]$, for each post-execution h' of h^F with time at least $\triangle$ (resp., with enough long time, depending only on F), then every execution in $[[H]]$ that is observably equivalent to $h^F.h'$ should contain F. Precisely, the existence of two indistinguishable behaviors, i.e., executions holding the same observations, with exactly one containing F and time long enough after F, i.e., whose time after F is at least $\triangle$ (resp., is arbitrarily long), violates the $\triangle$-diagnosability (resp., diagnosability) property for hybrid automata. Inspired from the framework of discrete event systems, we define critical pairs for partially observable hybrid automata taking into account both continuous and discrete dynamics.

Definition 10 ($\triangle$-Critical Pair) A pair of executions $h, h' \in [[H]]$ is called a $\triangle$-critical pair with respect to F iff: $F \in h$ and $F \notin h'$ and $Obs(h) = Obs(h')$ and $time(h, F) \geq \triangle$.

Theorem 2 *A fault F is $\triangle$-diagnosable in a POHA H iff there is no $\triangle$-critical pair in* $[[H]]$ *with respect to F. F is diagnosable in H iff, for some $\triangle$, there is no $\triangle$-critical pair in* $[[H]]$ *with respect to F (i.e., there is no arbitrarily long time after F critical pair).*

Note that all above definitions (e.g., observable projection, post-executions, diagnosability, critical pairs, etc.) are applicable in a similar way to timed automata, which can be considered as a special type of hybrid automata. The only difference is that the set of continuous variables is the set of clock variables whose derivative is always 1 [7, 9, 20, 21, 35]. And, as for automata the existence of arbitrarily long (in terms of transitions number) after F faulty executions implies the existence of an infinite faulty execution, in the same way it has been proved [58] that for timed automata the existence of arbitrarily long time after F faulty executions implies the existence of a $+\infty$-faulty execution (extending the definition above to $\triangle = +\infty$) and thus that non-diagnosability is witnessed by the existence of a $+\infty$-critical pair and its checking is PSPACE-complete.

Theorem 3 *A fault F is diagnosable in a partially observable timed automaton T iff there is no $+\infty$-critical pair in $[[T]]$ with respect to F. Checking diagnosability of T is PSPACE-complete.*

We will rest on this result as diagnosability checking of a POHA H will be done on a time automaton T abstracting H.

11.4 Abstracting Hybrid Automata

For continuous systems, verifying the most basic properties such as “Is this state reachable?” is not decidable, due in particular to the uncountability of continuous domains [37]. A fortiori, for a hybrid system, a simple computation of the reachable set of states starting from an initial state is not a decidable matter except for few unpractical classes [39]. This is why a common practice is to partition infinite domains into a finite number of subsets, abstracting the system behavior in each of those subsets. The abstraction of the domain into representative sets is usable in computations to possibly reason about the infinite domain. In this section, we thus focus on abstractions that discretize the infinite state space defined by continuous variables into finite sets. We show the targeted class of properties we wish to verify. As our study considers abstractions of complex hybrid dynamical systems for diagnosability analysis, it is therefore crucial to first introduce abstractions for continuous dynamical systems which are a particular case of hybrid dynamical systems. Utilizing these abstractions, we aim at verifying temporal properties and bounding the time for fault detection and isolation. Abstractions retain less but important information regarding a property that we wish to verify about a complex system. Due to the uncountability of the continuous state space domain, verifying simple properties of continuous and more generally hybrid systems via abstraction becomes challenging. The challenging part about abstractions is the choice made to select the representative sets and the criterion for choosing them. This choice relies entirely on the class of the properties one wishes to verify and on the structure of the hybrid system itself. Here, we are interested in hybrid systems abstraction aimed towards diagnosability checking. Abstractions that can be refined if necessary are of our concern, as refinement allows adding more information into the abstracted system while being always guided by the property to check. In this section, we will discuss abstractions in general and focus on those that capture time constraints in particular.

11.4.1 Different Abstraction Strategies

Using Qualitative Principles We first take a look at abstractions using qualitative concepts. Given a set of ODEs, these concepts classically discretize the infinite state space of the continuous variables into finite sets. The discretization of $\mathbb{R}^n$ is often achieved by rectangles, i.e., is built by product from a discretization of $\mathbb{R}$. And this one is obtained by fixing a finite number of (rational) landmarks l_i, resulting in a finite partition in terms of open intervals (l_i, l_{i+1}) (with possibly infinite endpoints) and singleton intervals $[l_i]$, allowing what is called absolute order of magnitude reasoning. The coarsest partition (except $\mathbb{R}$ itself) is obtained from the single landmark 0 and corresponds to the sign partition: $(-\infty, 0)$, $[0]$, $(0, +\infty)$, giving rise to a partition of size 3^n of $\mathbb{R}^n$. It is particularly interesting when applied to the valuations of the variables derivatives, as it corresponds to discretize according to the sign of the derivative, which is constant within each set of the partition, and thus to the change direction of the variable itself (decreasing, constant, increasing) . The variables being continuously differentiable, it is not possible for the sign of the derivative to pass from negative to positive without crossing zero. Exploiting this feature, it is possible to draw a rough scheme of the behavior of the variables called "qualitative simulation" in the literature and to obtain a loose overview about how the system will behave [30, 43, 55].

Example 2 Consider this simple linear continuous system:

$$\begin{aligned} \dot{x} &= 3x \\ \dot{y} &= y - 1 \end{aligned} \tag{11.1}$$

Adopting the partition of the state space given by the signs of the derivatives, the abstract state space of size 9 is thus: $(\dot{x} > 0 \vee \dot{x} < 0 \vee \dot{x} = 0) \wedge (\dot{y} > 0 \vee \dot{y} < 0 \vee \dot{y} = 0)$. The transitions between the abstract states are computed according to the laws of evolution given the signs of the derivatives. The abstract state $(\dot{x} > 0 \wedge \dot{y} > 0)$ corresponds to the region $\{x, y \mid x > 0 \wedge y > 1\}$ in the state space. From this state, no transition is possible to another abstract state. Suppose we wish to verify a basic reachability property: starting from the state $(1, 3)$ is it possible to reach the state $(-5, -4)$? The answer would be no, the proof is given using the previous abstraction method and inferring the property back to the original system. Such abstraction is **sound**: from any initial state (x_0, y_0) the solutions of the differential equation system (11.1) will always satisfy the constraints imposed by the abstract system rules, i.e., the possible transitions.

Abstractions for the Verification of Temporal Properties The above abstraction partitions the state space into sets with a constant sign of the derivative. This abstraction is useful to trace the future evolution of the state given the initial one to prove a safety property of avoiding an unwanted state. Nonetheless, for proving a more complex property that involves the notion of time, classically expressed using temporal logic, the above abstraction is not sufficient. One needs to add time as a separate state variable and correlate the variables changes to changes in time.

Example 3 We consider the same continuous system and suppose the initial set of states I such that $I = \{(x, y) \mid 1 < x < 2 \wedge 1 < y < 2 \wedge x < y\}$ and the property $F(x > y)$ where F is the "eventually" linear temporal logic (LTL) operator. Fp, where p is a Boolean proposition, is equivalent to $\exists t_0 \in \mathbb{R}^+, \forall t > t_0, p = \mathit{true}$. It is obvious that the rate at which x is increasing with respect to time is much larger than that of y. Hence, for all the initial states within I the property is true. The previous abstraction method however does not capture the rate at which the derivative of x is changing and is thus useless for establishing the proof. Actually, changing the first equation in (11.1) by $\dot{x} = 0.5x$ would keep the abstract system unchanged and nevertheless change the truth value of the property. In our case, the system can be written as $\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}$ where $\mathbf{x} = (x, y)^T$ and A is the corresponding matrix. We then deduce by computing the eigenvalues of A which are 3 and 1 in our example that the rate at which x increases is larger than the rate at which y increases, which provides a sufficient proof that the above property holds when the system is initiated from I.

The previous example illustrates the simple case of a linear dynamics where the eigenvectors are not rotated by the linear transformation and are thus invariant for the continuous system. Therefore, taking these two vectors into account during the abstraction process is an obvious choice. However, in the more general case of nonlinear dynamics, the invariant takes a more complex form. Some technique encodes the hybrid system, the property to verify and a specific parametric form of the invariant into an SMT (Satisfiability Modulo Theories) based solver and evaluates the unknown parameters of the invariant automatically. Once computed, the invariant is incorporated to make a finer and more representative abstraction [36].

11.4.2 Geometric Decomposition of the State Space

We now introduce finite state space decomposition of a hybrid automaton. We will then present an abstraction based on different decompositions 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.

Definition 11 (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 is that any (finite) continuous path crosses only a finite number of times each region, more precisely, $\forall x : [0,1] \to \mathbb{R}^n$ a continuous function, $\forall p \in P$ 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 the different partitions is guided by the property we wish to verify. For example, consider a continuous system with dynamics F. A coarse but helpful way to obtain a high level reachability mapping would be to identify regions of the state space that conserve the sign of F. E.g., in one dimension, for all elements of the same region the derivative signs would be all either negative or positive or null. Thus, the regions would be the connected components of the three subsets p_1, p_2, p_3 defined by:

$$p_1 = \{\mathbf{x} \in \mathbf{X} | \dot{x} > 0\}, p_2 = \{\mathbf{x} \in \mathbf{X} | \dot{x} < 0\}, p_3 = \{\mathbf{x} \in \mathbf{X} | \dot{x} = 0\} \tag{11.2}$$

For n dimensional systems, the regions would be the connected components of the 3^n subsets E_s of $\mathbb{R}^n$ parametrized by sign vectors $s \in \{-1, 0, +1\}^n$: $E_s = \{\mathbf{x} \in \mathbf{X} \mid \forall i, 1 \leq i \leq n, \dot{\mathbf{x}}^i < 0 \text{ if} s^i = -1, \dot{\mathbf{x}}^i = 0 \text{ if} s^i = 0, \dot{\mathbf{x}}^i > 0 \text{ if} s^i = +1\}$.

Example 4 (The Brusselator) We now illustrate some of the introduced concepts on a mathematical model used for representing chemical reactions: the brusselator whose dynamics is nonlinear. Consider a two-dimensional continuous system $C = (X, S_0, F, Inv)$ such that $X = \{x, y\}$ are two continuously differentiable variables and F given by [4]:

$$\begin{aligned} \dot{x} &= 1 - (b+1)x + ax^2y \\ \dot{y} &= bx - ax^2y \end{aligned} \tag{11.3}$$

where $a, b \in \mathbb{R}$ are two real constants. The stationary point for which $\dot{x} = \dot{y} = 0$ is $M_0(1, \frac{b}{a})$. If $b < a + 1$, then M_0 is an attractor and all trajectories converge towards M_0; if $b > a+1$ then it is a repeller and all trajectories close to M_0 converge towards an orbit. We consider two cases where $b = 1, a = 2$ and $b = 3, a = 1$ illustrated respectively in Fig. 11.2a, b. To characterize the dynamic behavior qualitatively as in Eq. (11.2), consider a partition P yielding nine regions $p_1, \ldots, p_9$ illustrated in the repeller case in Fig. 11.3 [30].

If the considered system is a hybrid automaton, it is practical to allow different partitions in different modes. In the following, we will assume that the sets $Init(q)$, $Inv(q)$, $G(\tau)$ and $R(\tau)(p)$ (for p connected subset of $G(\tau)$) can be expressed as finite unions of connected subsets (if this is not the case, we will over-approximate parts of them). We define thus a decomposition of the hybrid state space as follows.

Definition 12 (Hybrid State Space Decomposition) Given a hybrid automaton $H = (Q, X, S_0, \Sigma, F, \delta, Inv, G, R)$ and a set $\mathfrak{P}$ of partitions of the valuations set of X, $\mathbf{X} \subseteq \mathbb{R}^n$, 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 $d(q) \in \mathfrak{P}$.

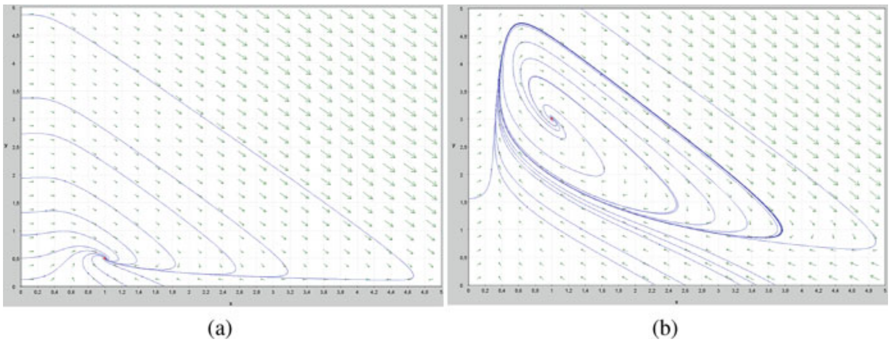
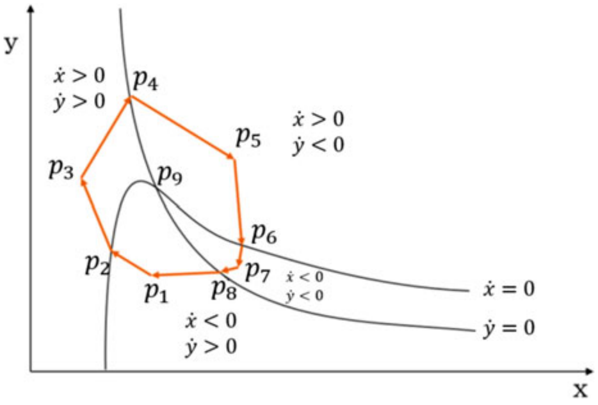


Fig. 11.2 Brusselator phase plane: (**a**) Attractor, (**b**) Repeller

Fig. 11.3 Qualitative partitioning of the state space



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).

11.4.3 Encoding Hybrid Automata Reachability Constraints

We have defined in Sect. 11.2.3, 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 13 (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 14 (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:

- $Q_{DH} = \{(q,p) | q \in Q, p \in d(q)\}$.
- $Q_{0_{DH}} = \{(q, p_{Init}) | q \in Q_0, p_{Init} \in d_{Init}(q)\}$.
- $\Sigma_{DH} = \Sigma \cup \{\varepsilon\}$.
- $((q_i, p_k), \sigma, (q_j, p_l)) \in \delta_{DH}$ 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 $\exists t = (q_i, \mathbf{x}, \sigma, \mathbf{x}', q_j) \in T$ such that $\mathbf{x} \in p_k$ and $\mathbf{x}' \in p_l$.
  - $q_i = q_j$ and $\sigma = \varepsilon$ and $p_k, p_l \in d_{Inv}(q_i)$ are adjacent regions and $\exists d \in \mathbb{R}_+^*$ and $\exists x : [0, d] \to \mathbf{X}$ 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 (11.2.3) is just $off \xrightarrow{B_{on}} on \xrightarrow{B_{off}} off \ldots$

Theorem 4 (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 7 we define the observation of its timeless abstraction.

Definition 15 (Timeless Abstraction Observation) Given a POHA H and $h = (q_0, p_0) \xrightarrow{\sigma_0} (q_1, p_1) \ldots (q_i, p_i) \xrightarrow{\sigma_i} \ldots$, with $\sigma_i \in \Sigma \cup \{\varepsilon\}$, a timeless abstract path in $DH_{\mathfrak{P}}$, the observation of h is defined as $Obs(h) = p_0^o, \sigma_0^o, p_1^o \ldots p_i^o, \sigma_i^o, \ldots$, 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 $Obs(h)$.

Consider an execution $h \in [[H]]$ of the POHA H, $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1) \ldots (q_i, \mathbf{x}_i) \xrightarrow{l_i} \ldots$, with $l_i \in \Sigma \cup \mathbb{R}_+$, its (timed) observation $Obs(h) = \mathbf{x}_0^o, l_0^o, \mathbf{x}_1^o \ldots \mathbf{x}_i^o, l_i^o, \ldots$ as in Definition 7, its timeless abstraction $\alpha_{\mathfrak{P}}(h) = (q_0, p_0) \xrightarrow{\sigma_0} (q_1, p_1) \ldots (q_i, p_i) \xrightarrow{\sigma_i} \ldots$, with $\sigma_i \in \Sigma \cup \{\varepsilon\}$, as in Theorem 4 (assuming F a singleton) and the observation of this last one $Obs(\alpha_{\mathfrak{P}}(h)) = p_0^o, \sigma_0^o, p_1^o \ldots p_i^o, \sigma_i^o, \ldots$ as in Definition 15. We could try to define the timeless abstraction of the observation $Obs(h)$. A natural definition would be $\alpha_{\mathfrak{P}}(Obs(h)) = p_0', \sigma_0^o, p_1' \ldots p_i', \sigma_i^o, \ldots$, with $\sigma_i^o = l_i^o$ if $l_i^o \in \Sigma_o$ (and $= \varepsilon$, which is removed, otherwise), i.e., the same σ_i^o's as in $Obs(\alpha_{\mathfrak{P}}(h))$, and $p_i' = \biguplus_{\{p \mid \mathbf{x}_i^o \in p^o\}} p^o$ the union of the projections on X_o of all regions containing a value whose projection on X_o is equal to $\mathbf{x}_i^o$ (assuming to simplify the same partition for each mode, as the mode may be unknown from observation). So, we notice that $Obs(\alpha_{\mathfrak{P}}(h))$ is more precise than $\alpha_{\mathfrak{P}}(Obs(h))$, as $p_i^o \subseteq p_i'$, which we denote by $Obs(\alpha_{\mathfrak{P}}(h)) \sqsubseteq \alpha_{\mathfrak{P}}(Obs(h))$ to mean that both sequences have common events and there is inclusion of the qualitative space values as subsets of $\mathbf{X}_o$, the valuations set corresponding to the observable variables.

11.4.4 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. Consider 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 16 (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 the time interval relative to the region p in mode q as $I_{(q,p)}$.

Definition 17 (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 $\sigma = \varepsilon$ 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 $(\mathit{off}, [80, 90])$ and by $(\mathit{off}, [68, 80))$. Then we take as time bounds for $(\mathit{off}, [80, 90])$ $t_{min} = 0$ 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 $Inv_{TH}((\mathit{off}, [80, 90])) = [0, 0.12]$. A beginning of execution of the timed abstract automaton is, for example, $(\mathit{off}, [80, 90]) \xrightarrow{0.08} (\mathit{off}, [68, 80])$.

Theorem 5 (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* $(q_0, \mathbf{x}_0)\xrightarrow{w}_*(q_i, \mathbf{x}_i) \in [[H]]$*, 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* $L' = L \cup \{\epsilon\}$*),* $\mathbf{x}_0 \in p_0$*,* $q_j = q_i$*,* $\mathbf{x}_i \in p_j$*,* $w'_{|\Sigma} = w_{|\Sigma}$ *(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 $(q_0,p_0) \xrightarrow{l_0} (q_1,p_1) \ldots (q_i,p_i) \xrightarrow{l_i} \ldots$, with $l_i \in \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_+$, in $TH_{\mathfrak{P}}$ into the execution $(q_0,p_0) \xrightarrow{\sigma_0} (q_1,p_1) \ldots (q_i,p_i) \xrightarrow{\sigma_i} \ldots$, with $\sigma_i \in \Sigma \cup \{\varepsilon\}$, in $DH_{\mathfrak{P}}$, with $\sigma_i = l_i$ if $l_i \in \Sigma \cup \{\varepsilon\}$ 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 18 (Timed Abstraction Observation) Given a POHA H and $h = (q_0,p_0) \xrightarrow{l_0} (q_1,p_1) \ldots (q_i,p_i) \xrightarrow{l_i} \ldots$, with $l_i \in \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_+$, an execution in $TH_{\mathfrak{P}}$, i.e., a timed abstract path, the observation of h is defined as $Obs(h) = p^o_0, l^o_0, p^o_1 \ldots p^o_i, l^o_i, \ldots$, 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 $Obs(h)$.

As for the timeless case, we can define the timed abstraction of an observation (of an execution $h \in [[H]]$) and we obtain: $Obs(\alpha^t_{\mathfrak{P}}(h)) \sqsubseteq \alpha^t_{\mathfrak{P}}(Obs(h))$.

From another side, the abstraction function α that forgets time maps a timed abstract observation $p^o_0, l^o_0, p^o_1 \ldots p^o_i, l^o_i, \ldots$, with $l^o_i \in \Sigma_o \cup \mathbb{R}_+$, into the timeless abstract observation $p^o_0, \sigma^o_0, p^o_1 \ldots p^o_i, \sigma^o_i, \ldots$, with $\sigma^o_i \in \Sigma_o$, suppressing duration labels $l_i = d_i \in \mathbb{R}_+$. For any concrete execution $h \in [[H]]$, we have: $Obs(\alpha_{\mathfrak{P}}(h)) = \alpha(Obs(\alpha^t_{\mathfrak{P}}(h)))$, i.e., $Obs \circ \alpha_{\mathfrak{P}} = \alpha \circ Obs \circ \alpha^t_{\mathfrak{P}}$.

11.4.5 Computing Time Bounds

In this subsection, 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 $I_p = [t_{min}, t_{max}]$ the 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* $\exists i\ 1 \leq i \leq n \quad \forall \mathbf{x} \in \overline{p} \quad \dot{\mathbf{x}}_i \neq 0$.

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. This idea will be developed in Sect. 11.5. 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 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 $\mathscr{T}$, 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 $\mathscr{T}$ in a neighborhood around $k = 1$ we obtain:

$$\mathscr{T} = \int_0^{t_{eq}} \mathrm{d}t = \int_{x_0}^{x_{eq}} \frac{1}{\dot{x}}\,\mathrm{d}x = \int_{x_0}^{0} \frac{1}{x^k}\,\mathrm{d}x = \left[\frac{x^{-k+1}}{(-k+1)}\right]_{x_0}^{0} \tag{11.4}$$

$\mathscr{T}$ 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 :

$$\mathscr{T} = \int_0^{t_{eq}} \mathrm{d}t = \int_{r_0}^{0} \frac{1}{\dot{r}}\,\mathrm{d}r = \int_{r_0}^{0} \frac{r}{X\dot{X} + Y\dot{Y}}\,\mathrm{d}r \tag{11.5}$$

Example 5 Consider the two dimensional continuous system $\dot{x} = -x^2$ and $\dot{y} = -y$ where $(x, y) \in \mathbb{R}^+ \times \mathbb{R}^+$. The equilibrium point is $M_{eq}(0, 0)$. In polar coordinates $x = r\cos(\theta)$ and $y = r\sin(\theta)$, then $r\dot{r} = x\dot{x} + y\dot{y} = -x^3 - y^2 = -x^2 - y^2 + x^2 - x^3 = -r^2 + x^2(1-x) = r^2(-1 + \cos^2(\theta)(1 - r\cos(\theta)))$. In a neighborhood around $(0, 0)$:

$$\mathscr{T} = \int_{r_0}^{0} \frac{1}{r(-1 + \cos^2(\theta)(1 - r\cos(\theta)))}\,\mathrm{d}r \geq \int_{r_0}^{0} \frac{-1}{r}\,\mathrm{d}r \tag{11.6}$$

Thus $\mathscr{T}$ is infinite, the equilibrium point is never reached. This reasoning can be extended to dimension n by evaluating $r\dot{r}$ and using spherical (or hyperspherical) coordinates and to polynomial with real exponents. We can take an example of square root, for instance $\dot{x} = \sqrt{x}$ and demonstrate the time $\mathscr{T}$ is finite, then the equilibrium is reached.

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, Maler and Batt [45] 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.

Brusselator Time Bounds For our Example 4 of the brusselator dynamics, consider, for the repeller case with $b = 3, a = 1$, a ring set R that excludes M_0 such that $R = \{(x, y) \mid (-1 + x)^2 + (-3 + y)^2 > 0.09 \wedge (-1 + x)^2 + (-3 + y)^2 < 0.5625\}$. Let $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ then $v^2 = (1 - 4x + x^2y)^2 + (3x - x^2y)^2$ and, using a solver, we compute a lower bound $v_{low}^2 = 0.0051$ of v^2. It has been proven that all trajectories initially in $S_0 = \mathbb{R}^2 \setminus \{M_0\}$ converge towards a fixed orbit of the phase plane contained within R. Suppose we split the region R into two connected sets R_1 and R_2 such that $R = R_1 \biguplus R_2$, for each of which the maximal sojourn duration t_{max} is a positive real constant. This is possible since v admits a lower bound v_{low}. This states that all trajectories initiated from S_0 will cross R_1 and R_2 sequentially infinitely often. In practice, the presence of the system in either R_1 or R_2 can correspond to two different visible colors of the chemical reaction.

11.5 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.

Definition 19 (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 20 (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 $\mathfrak{P}' \preceq \mathfrak{P}$, if $\forall q \in Q \; d'(q)$ is a refining partition of $d(q)$.

11.5.1 Refined Timeless Model

Definition 21 (Refined Timeless Abstract Automaton) Given a hybrid automaton H and two abstract timeless automata $DH_{\mathfrak{P}}$ and $DH_{\mathfrak{P}'}$ 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 $DH_{\mathfrak{P}}$ abstracting H if $\mathfrak{P}' \prec \mathfrak{P}$, which we denote by $DH_{\mathfrak{P}'} \prec DH_{\mathfrak{P}}$.

Definition 22 (State Split Operation) Given an abstract timeless automaton $DH_{\mathfrak{P}}$ 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 $DH_{\mathfrak{P}'}$ 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. In practice, the refined model is obtained by performing a finite number of state split operations. After having performed the split operations and in order for the obtained automaton to satisfy Definition 14, it is only required to recompute some of its transitions, while inheriting the rest from $DH_{\mathfrak{P}}$. Let $Q_{split} \subseteq Q_{DH}$ be the set of split states and $post(Q_{split}) = \{q \in Q_{DH} \mid \exists q_s \in Q_{split} \exists (q_s, \sigma, q) \in \delta_{DH}\}$ and $pre(Q_{split}) = \{q \in Q_{DH} \mid \exists q_s \in Q_{split} \exists (q, \sigma, q_s) \in \delta_{DH}\}$. Then to obtain $DH_{\mathfrak{P}'}$ it is sufficient to only recompute transitions (q, σ, q') such that $q, q' \in Q_{split} \cup post(Q_{split}) \cup pre(Q_{split})$.

The onto abstraction function $\alpha_{\mathfrak{P}',\mathfrak{P}} : Q_{DH}^{\mathfrak{P}'} \to Q_{DH}^{\mathfrak{P}}$ defined 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: $\alpha_{\mathfrak{P}} = \alpha_{\mathfrak{P}',\mathfrak{P}} \circ \alpha_{\mathfrak{P}'}$. Defining in a natural way as previously the $\mathfrak{P}$-abstraction $\alpha_{\mathfrak{P}',\mathfrak{P}}$ of the observation of a timeless $\mathfrak{P}'$-abstract execution h, we obtain: $Obs(\alpha_{\mathfrak{P}',\mathfrak{P}}(h)) \sqsubseteq \alpha_{\mathfrak{P}',\mathfrak{P}}(Obs(h))$.

11.5.2 Refined Timed Model

Definition 23 (Refined Timed Abstract Automaton) Given a hybrid automaton H and 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 $TH_{\mathfrak{P}'}$ is a timed refinement of $TH_{\mathfrak{P}}$ abstracting H if $\mathfrak{P}' \prec \mathfrak{P}$. We denote it similarly by $TH_{\mathfrak{P}'} \prec TH_{\mathfrak{P}}$.

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. Thus, the refined timed model is obtained by a finite sequence of the two operations:

- **State split**: similar as before, the state split of (q,p) whose time interval is $I_{(q,p)} = [t_{min}, t_{max}]$ yields (q,p_1) and (q,p_2). The time intervals for the new split regions are set as $I_{(q,p_1)} = I_{(q,p_2)} = [0, t_{max}]$ (t_{max} stays a safe upper bound of the sojourn duration but t_{min} is reset to 0 since the split induces a distance shrink).
- **Time bounds refinement:** in this case, more precise time bounds are obtained for a given region of a discrete state, i.e., if $I_{(q,p)} = [t_{min}, t_{max}]$ then $I'_{(q,p)} = [t'_{min}, t'_{max}]$ with $t'_{min} \geq t_{min}$ and $t'_{max} \leq t_{max}$, at least one of both being a strict inequality.

The onto abstraction function $\alpha_{\mathfrak{P}',\mathfrak{P}}$ defines a trace preserving mapping $\alpha^t_{\mathfrak{P}',\mathfrak{P}}$ from $TH_{\mathfrak{P}'}$ to $TH_{\mathfrak{P}}$, after trace simplification as in Theorem 5 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: $\alpha^t_{\mathfrak{P}} = \alpha^t_{\mathfrak{P}',\mathfrak{P}} \circ \alpha^t_{\mathfrak{P}'}$. Finally, for any timed $\mathfrak{P}'$-abstract execution h, we obtain: $Obs(\alpha^t_{\mathfrak{P}',\mathfrak{P}}(h)) \sqsubseteq \alpha^t_{\mathfrak{P}',\mathfrak{P}}(Obs(h))$.

11.5.3 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:

$$post(\mathbf{y}) = \{\mathbf{x} \in \mathbf{X} \mid \exists t > t_0 \mathbf{x} = x(t)\} pre(\mathbf{y}) = \{\mathbf{x} \in \mathbf{X} \mid \exists 0 < t < t_0 \mathbf{x} = x(t)\} \tag{11.7}$$

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

$$post(\mathbf{K}) = \bigcup_{k \in \mathbf{K}} post(k) \quad pre(\mathbf{K}) = \bigcup_{k \in \mathbf{K}} pre(k) \tag{11.8}$$

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

Theorem 6 *$post(\mathbf{K})$ and $pre(\mathbf{K})$ 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.

Brusselator Example This result applies to the polynomial dynamics of the brusselator, more specifically the two defined regions R_1 and R_2 corresponding to the color change of the system. We consider the repeller case of the brusselator as a discrete hybrid mode. With the previous result, we demonstrate that any inconsistency in observing the color change within the computed maximal time bound could be diagnosed with a change of the current repellor mode. If we model the fault as a discrete jump to the attractor case, the observations in terms of color change are sufficient to diagnose this fault, since in the attractor case all trajectories converge asymptotically to M_0, thus there is no color change.

11.6 CEGAR Adaptation for Diagnosability Verification

Since hybrid systems have an infinite state space due to continuous dynamics, verifying formally their properties often rests on using ordinary model checking together with a finite-state abstraction. Model-checking can be inconclusive, in which case the abstraction must be refined. In this section, we adapt counterexample guided abstraction refinement (CEGAR) that was originally proposed to verify safety properties [3, 28].

Note that to verify safety properties, it is sufficient to check one execution at a time and verify whether the execution can reach an unsafe state. Verifying diagnosability reveals a more complex task as one is required to simultaneously analyze two executions at a time, i.e., to verify whether or not the two executions have the same observations while only one of them contains the considered fault.

In our abstraction method, time constraints are used explicitly. When an abstraction refinement is required, tighter time bounds are obtained over the new regions of the refined decomposition. The proposed abstraction method hence differs from the one proposed in [35]. In [35], the abstraction consists in retaining properties of the continuous dynamics, namely mode distinguishability and ephemerality, which are directly checked on the concrete hybrid system when necessary. On the contrary, in our approach the abstractions refer directly to the continuous state space and the continuous dynamics are interpreted with increasing levels of granularity, which results in finer and finer state space decompositions to which time constraints are associated. These abstractions take the form of timed automata.

The adaptation of CEGAR to verify diagnosability of a hybrid automaton H consists in three steps described as follows and to be detailed in the next subsections:

- **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.
- **Validation of the $C.E$** by checking whether the $C.E$ is valid or spurious.
- **Refinement** of the timed abstract automaton by using a finer hybrid state space decomposition.

11.6.1 CEGAR Scheme for Hybrid Automata 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, $h' \in [[H]]$, $h = (q_0, \mathbf{x}_0) \xrightarrow{l_0} (q_1, \mathbf{x}_1) \ldots (q_i, \mathbf{x}_i) \xrightarrow{l_i} \ldots$, $h' = (q'_0, \mathbf{x}'_0) \xrightarrow{l'_0} (q'_1, \mathbf{x}'_1) \ldots (q'_i, \mathbf{x}'_i) \xrightarrow{l'_i} \ldots$ are called a counterexample of diagnosability in H with respect to the fault F if they satisfy the three conditions defined in Definition 10, i.e., if h and h' constitute a critical pair of H.

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 5, 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 $\hat{h}, \hat{h}' \in [[TH_{\mathfrak{P}}]]$, $\hat{h} = \hat{s}_0 \xrightarrow{\hat{l}_0} \hat{s}_1 \ldots \hat{s}_i \xrightarrow{\hat{l}_i} \ldots$, $\hat{h}' = \hat{s}'_0 \xrightarrow{\hat{l}'_0} \hat{s}'_1 \ldots \hat{s}'_i \xrightarrow{\hat{l}'_i} \ldots$ and, as (h, h') is a critical pair in H with respect to F, so is

Algorithm 1 CEGAR scheme for hybrid automata diagnosability verification

INPUT: hybrid automaton H; considered fault F; constant positive integer *precision*
OUTPUT: *decision* $:=$ H is diagnosable (*true*) $|$ H is not diagnosable (*false*) $|$ precision is reached (*max_reached*)

```
$TH \leftarrow$ Initial Timed Abstract Automaton of $H$
$C.E \leftarrow$ Diagnosability Check $(TH, F)$
$abstraction\_level \leftarrow 0$
while $C.E \neq \emptyset \wedge abstraction\_level < precision$ do
    if Validate$(C.E, H)$ then
        $decision \leftarrow false$ EXIT
    else
        $TH \leftarrow$ Refine$(TH, C.E, H)$
        $C.E \leftarrow$ Diagnosability Check$(TH, F)$
        $abstraction\_level \leftarrow abstraction\_level + 1$
    end if
end while
if $C.E = \emptyset$ then
    $decision \leftarrow true$ EXIT
else
    $decision \leftarrow max\_reached$ EXIT
end if
```

$(\hat{h}, \hat{h}')$ 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 7 *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.

11.6.2 Twin Plant Based Diagnosability Checking

Diagnosability checking of a discrete event system, modeled as an automaton, based on the twin plant method [40, 61] is polynomial in the number of states (actually it has been proved it is NLOGSPACE-complete [51]). 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 [58], 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 [58].*

For timed automata, checking diagnosability is PSPACE-complete.

11.6.3 Counterexample Validation or Refusal

After applying the twin plant method on a timed abstract automaton $TH_{\mathfrak{P}}$ of H as described in [58], 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 $\hat{h}$ and $\hat{h}'$ and form a concrete critical pair decides if $C.E$ is validated or refuted. We detail below both procedures for validation or refusal and the reasons for which, in the latter case, a critical pair can be assumed spurious.

Validated Counterexample If it exists $h, h' \in [[H]]$, whose abstractions by $\alpha^t_{\mathfrak{P}}$ (according to Theorem 5) 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, 12]. 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.

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.

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.

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.

11.6.4 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 $s_{|h|+1}^{reach} = \emptyset$, 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.

11.7 Case Study Example

The CEGAR scheme for diagnosability checking of hybrid automata will be illustrated by the following case study example.

Example 6 (Fault Tolerant Thermostat Model) The two observable events B_{on} and B_{off} allow one to witness mode changes and the continuous variable x is assumed to be observable. The system starts from $x \in [80, 90]$. Two faults are modeled as unobservable events F_1 and $F_2 \in \Sigma_f$ shown in Fig. 11.4. In practice, the fault F_1 models a bad calibration of the temperature sensor. As for fault F_2, it represents a defect in the heater impacting its efficiency and is modeled by a parametric change of a constant in the expression of the first order derivative of x.

11.7.1 CEGAR Scheme for Fault F_1

Initial Abstraction We consider an initial decomposition $\mathfrak{P} = \{P_{off}, P_{on}, P_{off}^{F_1}, P_{on}^{F_1}$ $P_{off}^{F_2}, P_{on}^{F_2}\}$ of the hybrid state space. Each partition $P \in \mathfrak{P}$ is made up of only one region representing the reals $\mathbb{R}$. Hence computing t_{min} and t_{max} for each region p yields $I_p = [0, +\infty)$, in other words the initial abstraction contains no time constraints.

Diagnosability Check The diagnosability check using the twin plant method generates a counterexample $C.E = (\hat{h}, \hat{h}')$ such that:

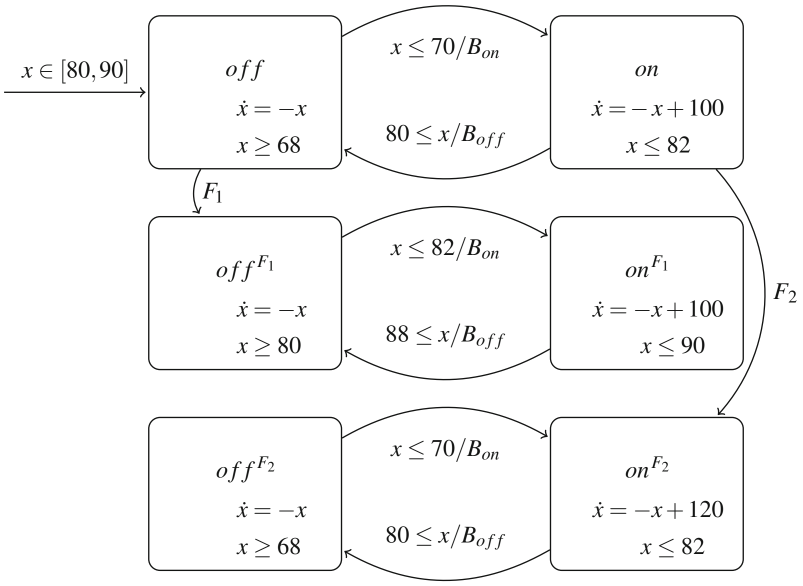


Fig. 11.4 One-dimensional hybrid automaton modeling a faulty thermostat

$$\hat{h} = (\mathit{off}, p_{\mathit{off}}) \xrightarrow{0.15} (\mathit{off}, p_{\mathit{off}}) \xrightarrow{Bon} (on, p_{on}) \xrightarrow{0.3} (on, p_{on}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}, p_{\mathit{off}}) \ldots$$

$$\hat{h}' = (\mathit{off}, p_{\mathit{off}}) \xrightarrow{0.1} (\mathit{off}, p_{\mathit{off}}) \xrightarrow{F_1} (\mathit{off}^{F_1}, p_{\mathit{off}}^{F_1}) \xrightarrow{0.05} (\mathit{off}^{F_1}, p_{\mathit{off}}^{F_1}) \xrightarrow{B_{on}}$$

$$(on^{F_1}, p_{on}^{F_1}) \xrightarrow{0.3} (on^{F_1}, p_{on}^{F_1}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}^{F_1}, p_{\mathit{off}}^{F_1}) \ldots$$

Validation or Refusal The computation of the set of concrete executions $\{h\}$ (resp. $\{h'\}$) whose abstraction is $\hat{h}$ (resp. $\hat{h}'$) yields an approximation as follows:

$$\{h\} = (\mathit{off}, [80, 90]) \xrightarrow{0.15} (\mathit{off}, [69, 77]) \xrightarrow{Bon} (on, [69, 70]) \ldots$$

$$\{h'\} = (\mathit{off}, [80, 90]) \xrightarrow{0.1} (\mathit{off}, [72, 81]) \xrightarrow{F_1} (\mathit{off}^{F_1}, [80, 81]) \xrightarrow{0.05}$$

$$(\mathit{off}^{F_1}, [76, 77] : \mathit{invalid})$$

The concrete state computations show that it is not possible to stay 0.05 time units in mode off^{F_1} as the temperature reached would be [76,77] violating the invariant $x \geq 80$. The *C.E* is thus refuted.

Refinement of the State Space The refinement aims at eliminating the previous spurious *C.E*. From this *C.E*, it is possible to compute the exact time constraint for staying in mode off^{F_1} and then triggering the transition B_{on} and refine the hybrid state space accordingly. Once refined, the new abstraction should not contain similar counterexamples. The validation process reveals that all trajectories entering mode off^{F_1} with $x \in [80, 81]$ cannot stay more than $t_{max} = 0.0124$ time units but it is possible for some trajectories to instantaneously change from *off* to off^{F_1} to on^{F_1} in which case $t_{min} = 0$, thus $I_{(\mathit{off}^{F_1},[80,81])} = [0, 0.0124]$. The refined abstraction would carry this new information by updating the partition of mode off^{F_1}, from $\mathbb{R}$ to $(-\infty, 80) \uplus [80, 81] \uplus (81, +\infty)$, thus ensuring that all future generated counterexamples would satisfy this constraint.

11.7.2 CEGAR Scheme for Fault F_2

Initial Abstraction The same as for F_1.

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

$$\hat{h} = (\mathit{off}, p_{\mathit{off}}) \xrightarrow{0.5} (\mathit{off}, p_{\mathit{off}}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.5} (on, p_{on}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}, p_{\mathit{off}}) \dots$$

$$\hat{h}' = (\mathit{off}, p_{\mathit{off}}) \xrightarrow{0.5} (\mathit{off}, p_{\mathit{off}}) \xrightarrow{B_{on}} (on, p_{on}) \xrightarrow{0.4} (on, p_{on}) \xrightarrow{F_2} (on^{F_2}, p_{on}^{F_2}) \xrightarrow{0.1}$$

$$(on^{F_2}, p_{on}^{F_2}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}^{F_2}, p_{\mathit{off}}^{F_2}) \dots$$

Validation or Refusal The computation of the set of concrete executions $\{h\}$ (resp. $\{h'\}$) whose abstraction is $\hat{h}$ (resp. $\hat{h}'$) yields an approximation as follows:

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

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

This $C.E$ is refuted due to illegal stay in the mode *off* violating the corresponding invariant. In other words the trajectories are not feasible: if the system stays in mode *off* for 0.5 time units, then the state invariant is no longer true. Thus, if B_{on} is observed, then the duration of stay in *off* should be smaller.

Refinement of the State Space To prevent future similar spurious counterexamples, a refinement is applied to the initial abstraction. The refined model considers new regions in mode *off*: $p_{\mathit{off}_1} = [80, 90]$ (initial region) and $p_{\mathit{off}_2} = [68, 80)$ (legal region). The computation of the time intervals relative to each region are: $I_{(\mathit{off},[68,80))} = [0, 0.16]$ and $I_{(\mathit{off},[80,90])} = [0, 0.12]$. 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

Diagnosability Check The second $C.E$ generated from the refined twin plant is:

$$\hat{h} = (\mathit{off}, p_{\mathit{off}_1}) \xrightarrow{0.08} (\mathit{off}, p_{\mathit{off}_1}) \xrightarrow{\varepsilon} (\mathit{off}, p_{\mathit{off}_2}) \xrightarrow{0.07} (\mathit{off}, p_{\mathit{off}_2}) \xrightarrow{B_{on}} (on, p_{on})$$
$$\xrightarrow{0.5} (on, p_{on}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}, p_{\mathit{off}}) \dots$$
$$\hat{h}' = (\mathit{off}, p_{\mathit{off}_1}) \xrightarrow{0.08} (\mathit{off}, p_{\mathit{off}_1}) \xrightarrow{\varepsilon} (\mathit{off}, p_{\mathit{off}_2}) \xrightarrow{0.07} (\mathit{off}, p_{\mathit{off}_2}) \xrightarrow{B_{on}} (on, p_{on})$$
$$\xrightarrow{0.4} (on, p_{on}) \xrightarrow{F_2} (on^{F_2}, p_{on}^{F_2}) \xrightarrow{0.1} (on^{F_2}, p_{on}^{F_2}) \xrightarrow{B_{\mathit{off}}} (\mathit{off}^{F_2}, p_{\mathit{off}}^{F_2}) \dots$$

Validation or Refusal Note that the continuous transitions in the second $C.E$ respect the temporal constraints added during the refinement based on the first $C.E$. The corresponding concrete approximate executions of this $C.E$ are:

$$\{h\} = (\mathit{off}, [80, 90]) \xrightarrow{0.15} (\mathit{off}, [69, 77]) \xrightarrow{B_{on}} (on, [69, 70]) \xrightarrow{0.5} (on, [80.5, 81.8]) \dots$$
$$\{h'\} = (\mathit{off}, [80, 90]) \xrightarrow{0.15} (\mathit{off}, [69, 77]) \xrightarrow{B_{on}} (on, [69, 70]) \xrightarrow{0.4} (on, [79, 79.9])$$
$$\xrightarrow{F_2} (on^{F_2}, [79, 79.9]) \xrightarrow{0.1} (on^{F_2}, [82.9, 83.7]) \dots$$

In this case, this second $C.E$ is also considered as spurious because, given the time constraints, the two trajectories are different in the observations of x in the hybrid system since the last regions are disjoint, i.e., $[80.5, 81.8] \cap [82.9, 83.7] = \emptyset$.

Refinement of the State Space The counterexample analysis could identify the time boundary $t_{spurious}$, up to which the observations could be the same for at least two concrete trajectories, and after which the critical pair becomes spurious. In our example, suppose the fault occurred at t_f where $x \in [a, b]$, then $t_{spurious}$ is the time instant from which faulty and nominal sets of trajectories are disjoint:

$$t_{spurious} = \ln\left(\frac{b - a + 20}{20}\right) + t_f \tag{11.9}$$

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: $x \in [79.90, 80.7]$ in the mode *on* and $x \in [80.7, 81.6]$ in the mode on^{F_2}. 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. For the sake of simplicity, we analyzed the two faults separately. One more sophisticated strategy is to analyze the next fault based on the refined abstraction obtained from the analysis of the precedent fault.

11.8 Conclusion

In this chapter, we were interested in verifying a given formal safety property 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 focused on the diagnosability property, for its importance in safety analysis at design stage and the challenge it gives rise to. 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.

This preliminary work draws many perspectives. First of all, we have to develop refinement strategies by analysis of the counterexample and progress in the (partial) automation of the whole process and the integration of the algorithms for abstract diagnosability checking, for validation of the counterexample and for refinement. We plan in particular to use and extend existing tools for timed automata model checking and for over-approximation reachability at the continuous level. We want to investigate also the usage of SMT solvers [17, 18], in particular with theories including ODEs [31], to deal simultaneously with discrete and continuous variables. Moreover, we plan to apply this approach to other formal safety properties such as predictability, which guarantees to predict in advance the future unavoidable occurrence of a fault given some sequence of observations. With all this, we will be able to tackle real applications and get deeper experimental results.

Another promising aspect is the potential of this approach to deduce minimal concrete sets of observations for which the system is diagnosable. These observations specify a minimal (for set inclusion) needed set of events and continuous variables for which the system is diagnosable. If one element of this set is not considered as observable, the system becomes non-diagnosable. Thus, such a minimal set draws the boundary between the diagnosable and non-diagnosable systems [14] from the point of view of their observability.

Up to now we assume a continuous domain for both the values and the time stamps of the observable variables without taking into account sensor capability, i.e., the minimal interval (of value and of time) that can be captured. This is the reason why our current algorithm may not terminate, due to an infinite refinement process. A fundamental and essential future work perspective is to provide a general algorithm for diagnosability verification with ε-precision [42], 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. In the same spirit, one interesting future work would be to demonstrate a bi-simulation relation between the concrete model and the final refined abstract model when considering this minimal precision imposed by the model and the sensors, where the termination of the refinement can thus be guaranteed. In other words, theoretically, we could always deduce the right verdict, either the system is diagnosable or it is not diagnosable with respect to a given minimal precision.

Acknowledgements The authors would like to thank Alban Grastien for his valuable comments and suggestions.

References

1. M. Althoff, O. Stursberg, M. Buss, Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. Nonlinear Anal. Hybrid Syst. **4**(2), 233–249 (2010)
2. R. Alur, D.L. Dill, A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)
3. R. Alur, T. Dang, F. Ivančić, Counterexample-guided predicate abstraction of hybrid systems. Theor. Comput. Sci. **354**(2), 250–271 (2006)
4. S. Ault, E. Holmgreen, Dynamics of the Brusselator. Academia (2009)
5. M. Basseville, M. Kinnaert, M. Nyberg, On fault detectability and isolability. Eur. J. Control. **7**(6), 625–641 (2001)
6. M. Bayoudh, L. Travé-Massuyès, Diagnosability analysis of hybrid systems cast in a discrete-event framework. Discrete Event Dyn. Syst. **24**(3), 309–338 (2014)
7. M. Bayoudh, L. Travé-Massuyès, X. Olive, Hybrid systems diagnosability by abstracting faulty continuous dynamics, in *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX'06)* (2006), pp. 9–15
8. M. Bayoudh, L. Travé-Massuyès, X. Olive, Coupling continuous and discrete event system techniques for hybrid systems diagnosability analysis, in *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, Patras (2008), pp. 219–223
9. M. Bayoudh, L. Travé-Massuyès, X. Olive, Active diagnosis of hybrid systems guided by diagnosability properties. IFAC Proc. Vol. **42**(8), 1498–1503 (2009)
10. G. Behrmann, A. David, K.G. Larsen, A tutorial on Uppaal, in *Formal Methods for the Design of Real-Time Systems* (Springer, Berlin, 2004), pp. 200–236
11. S. Biswas, D. Sarkar, S. Mukhopadhyay, A. Patra, Diagnosability analysis of real time hybrid systems, in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT'06)*, Mumbai (2006), pp. 104–109
12. O. Botchkarev, S. 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)*. LNCS, vol. 1790 (Springer, Berlin, 2000), pp. 73–88

13. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, S. Yovine, Kronos: a model-checking tool for real-time systems, in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems* (Springer, Berlin, 1998), pp. 298–302
14. L. Brandán Briones, A. Lazovik, P. Dague, Optimizing the system observability level for diagnosability, in *Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA08)*, Chalkidiki, Kassandra (2008)
15. J. Chen, R. Patton, A re-examination of the relationship between parity space and observer-based approaches in fault diagnosis, in *Proceedings of the IFAC Symposium on Fault Detection, Supervision and Safety of Technical Systems Safeprocess'94*, Helsinki (1994), pp. 590–596
16. A. Cimatti, C. Pecheur, R. Cavada, Formal verification of diagnosability via symbolic model checking, in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)* (2003), pp. 363–369
17. A. Cimatti, S. Mover, S. Tonetta, SMT-based scenario verification for hybrid systems. Formal Methods Syst. Des. **42**(1), 46–66 (2013)
18. A. Cimatti, A. Griggio, S. Mover, S. Tonetta, HyComp: an SMT-based model checker for hybrid systems, in *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-2015)*, London (2015), pp. 52–67
19. V. Cocquempot, T.E. Mezyani, M. Staroswiecki, Fault detection and isolation for hybrid systems using structured parity residuals, in *Proceedings of the IEEE/IFAC-ASCC: Asian Control Conference*, vol. 2, Melbourne (2004), pp. 1204–1212
20. M.J. Daigle, A qualitative event-based approach to fault diagnosis of hybrid systems. PhD thesis, Vanderbilt University, 2008
21. M. Daigle, X. Koutsoukos, G. Biswas, An event-based approach to hybrid systems diagnosability, in *Proceedings of the 19th International Workshop on Principles of Diagnosis (DX'08)* (2008), pp. 47–54
22. M.J. Daigle, D. Koutsoukos, G. Biswas, An event-based approach to integrated parametric and discrete fault diagnosis in hybrid systems. Trans. Inst. Meas. Control. (Special Issue on Hybrid and Switched Systems) **32**(5), 487–510 (2010)
23. M.J. Daigle, I. Roychoudhury, G. Biswas, D. Koutsoukos, A. Patterson-Hine, S. Poll, A comprehensive diagnosis methodology for complex hybrid systems: a case study on spacecraft power distribution systems. IEEE Trans. Syst. Man Cybern. Part A (Special Issue on Model-based Diagnosis: Facing Challenges in Real-world Applications) **4**(5), 917–931 (2010)
24. Y. Deng, A. D'Innocenzo, M.D. Di Benedetto, S. Di Gennaro, A.A. Julius, Verification of hybrid automata diagnosability with measurement uncertainty. IEEE Trans. Autom. Control **61**(4), 982–993 (2016)
25. O. Diene, E.R. Silva, M.V. Moreira, Analysis and verification of the diagnosability of hybrid systems, in *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC-14)* (IEEE, New York, 2014), pp. 1–6
26. O. Diene, M.V. Moreira, V.R. Alvarez, E.R. Silva, Computational methods for diagnosability verification of hybrid systems, in *Proceedings of the IEEE Conference on Control Applications (CCA-15)* (IEEE, New York, 2015), pp. 382–387
27. S. Ding, *Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools* (Springer, London, 2008)
28. C. Edmund, F. Ansgar, H. Zhi, K. Bruce, S. Olaf, T. Michael, Verification of hybrid systems based on counterexample-guided abstraction refinement, in *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-2003)*, ed. by H. Garavel, J. Hatcliff. Lecture Notes in Computer Science, vol. 2619 (Springer, Cham, 2003), pp. 192–207
29. G. Fourlas, K. Kyriakopoulos, N. Krikelis, Diagnosability of hybrid systems, in *Proceedings of the 10th Mediterranean Conference on Control and Automation (MED-2002)*, Lisbon (2002), pp. 3994–3999
30. J.-P. Gallois, J.-Y. Pierron, Qualitative simulation and validation of complex hybrid systems, in *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS-2016)*, Toulouse (2016)

31. S. Gao, S. Kong, E. Clarke, Satisfiability modulo ODEs, in *Formal Methods in Computer-Aided Design (FMCAD)* (2013)
32. V. Germanos, S. Haar, V. Khomenko, S. Schwoon, Diagnosability under weak fairness, in *Proceedings of the 14th International Conference on Application of Concurrency to System Design (ACSD'14)*, Tunis (IEEE Computer Society Press, New York, 2014)
33. J. Gertler, *Fault Detection and Diagnosis in Engineering Systems* (Marcel Dekker, New York, 1998)
34. A. Grastien, Symbolic testing of diagnosability, in *Proceedings of the 20th International Workshop on Principles of Diagnosis (DX-09)* (2009), pp. 131–138
35. A. Grastien, L. Travé-Massuyès, V. 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)
36. S. Gulwani, A. Tiwari, Constraint-based approach for analysis of hybrid systems, in *Proceedings of the 20th International Conference on Computer Aided Verification (CAV-2008)* (2008), pp. 190–203
37. E. Hainry, Decidability and undecidability in dynamical systems. Rapport de recherche (CiteSeer, 2009). http://hal.inria.fr/inria-00429965/en/
38. T.A. Henzinger, The theory of hybrid automata, in *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1996), pp. 278–292
39. T.A. Henzinger, P.W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, in *Proceedings of the 27th Annual Symposium on Theory of Computing* (ACM Press, New York, 1995), pp. 373–382
40. S. Jiang, Z. Huang, V. Chandra, R. Kumar, A polynomial algorithm for testing diagnosability of discrete-event systems. IEEE Trans. Autom. Control **46**(8), 1318–1321 (2001)
41. E. Kilic, Diagnosability of fuzzy discrete event systems. Inf. Sci. **178**(3), 858–870 (2008)
42. K.-D. Kim, S. Mitra, P.R. Kumar, Computing bounded epsilon-reach set with finite precision computations for a class of linear hybrid automata, in *Proceedings of the ACM International Conference on Hybrid Systems: Computation and Control* (2011)
43. B. Kuipers, *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge* (MIT Press, Cambridge, MA, 1994)
44. F. Liu, D. Qiu, Safe diagnosability of stochastic discrete-event systems. IEEE Trans. Autom. Control **53**(5), 1291–1296 (2008)
45. O. Maler, G. Batt, Approximating continuous systems by timed automata, in *Formal Methods in Systems Biology* (Springer, Berlin, 2008), pp. 77–89
46. T. Melliti, P. Dague, Generalizing diagnosability definition and checking for open systems: a Game structure approach, in *Proceedings of the 21st International Workshop on Principles of Diagnosis (DX'10)*, Portland, OR (2010), pp. 103–110
47. M. Nyberg, Criterions for detectability and strong detectability of faults in linear systems. Int. J. Control. **75**(7), 490–501 (2002)
48. Y. Pencolé, Diagnosability analysis of distributed discrete event systems, in *Proceedings of the 16th European Conference on Artificial Intelligent (ECAI-04)* (2004), pp. 43–47
49. Y. Pencolé, A. Subias, A chronicle-based diagnosability approach for discrete timed-event systems: application to web-services. J. Universal Comput. Sci. **15**(17), 3246–3272 (2009)
50. P. Ribot, Y. Pencolé, Design requirements for the diagnosability of distributed discrete event systems, in *Proceedings of the 19th International Workshop on Principles of Diagnosis (DX'08)*, Blue Mountains (2008), pp. 347–354
51. J. Rintanen, Diagnosers and diagnosability of succinct transition systems, in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad (2007), pp. 538–544
52. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of discrete event systems. Trans. Autom. Control **40**(9), 1555–1575 (1995)
53. A. Schumann, J. Huang, A scalable jointree algorithm for diagnosability, in *Proceedings of the 23rd American National Conference on Artificial Intelligence (AAAI-08)* (2008), pp. 535–540

54. D. Thorsley, D. Teneketzis, Diagnosability of stochastic discrete-event systems. IEEE Trans. Autom. Control **50**(4), 476–492 (2005)
55. L. Travé-Massuyès, P. Dague, *Modèles et raisonnements qualitatifs* (Hermès, Paris, 2003)
56. L. Travé-Massuyès, M. Cordier, X. Pucel, Comparing diagnosability criterions in continuous systems and discrete events systems, in *Proceedings of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (Safeprocess'06)*, Beijing (2006), pp. 55–60
57. L. Travé-Massuyès, T. Escobet, X. Olive, Diagnosability analysis based on component-supported analytical redundancy relations. IEEE Trans. Syst. Man Cybern. Part A **36**(6), 1146–1160 (2006)
58. S. Tripakis, Fault diagnosis for timed automata, in *Proceedings of International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT-2002)*. Lecture Notes in Computer Science, vol. 2469 (Springer, Berlin, 2002), pp. 205–221
59. Y. Yan, L. Ye, P. Dague, Diagnosability for patterns in distributed discrete event systems, in *Proceedings of the 21st International Workshop on Principles of Diagnosis (DX'10)*, Portland, OR (2010), pp. 345–352
60. L. Ye, P. Dague, Diagnosability analysis of discrete event systems with autonomous components, in *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10)* (2010), pp. 105–110
61. T.-S. Yoo, S. Lafortune, Polynomial-time verification of diagnosability of partially observed discrete-event systems. IEEE Trans. Autom. Control **47**(9), 1491–1495 (2002)
62. M. Yu, D. Wang, M. Luo, D. Zhang, Q. Chen, Fault detection, isolation and identification for hybrid systems with unknown mode changes and fault patterns. Expert Syst. Appl. **39**(11), 9955–9965 (2012)
63. J. Zaytoon, S. Lafortune, Overview of fault diagnosis methods for discrete event systems. Annu. Rev. Control. **37**(2), 308–320 (2013)