Diagnosability of Fair Transition Systems

Benjamin Bittner, Marco Bozzano*, Alessandro Cimatti, Marco Gario, Stefano Tonetta, Viktoria Vozarova

Fondazione Bruno Kessler
Via Sommarive 18, 38123 Povo, Trento, Italy

Abstract

The integrity of complex dynamic systems often relies on the ability to detect, during operation, the occurrence of faults, or, in other words, to diagnose the system. The feasibility of this task, also known as *diagnosability*, depends on the nature of the system dynamics, the impact of faults, and the availability of a suitable set of sensors. Standard techniques for analyzing the diagnosability problem rely on a model of the system and on proving the absence of a faulty trace that cannot be distinguished by a non-faulty one (this pair of traces is called *critical pair*).

In this paper, we tackle the problem of verifying diagnosability under the presence of fairness conditions. These extend the expressiveness of the system models enabling the specification of assumptions on the system behavior such as the infinite occurrence of observations and/or faults.

We adopt a comprehensive framework that encompasses fair transition systems, temporally extended fault models, delays between the occurrence of a fault and its detection, and rich operational contexts. We show that in presence of fairness the definition of diagnosability has several interesting variants, and discuss the relative strengths and the mutual relationships. We prove that the existence of critical pairs is not always sufficient to analyze diagnosability, and needs to be generalized to *critical sets*. We define new notions of critical pairs, called *ribbon-shape*, with special looping conditions to represent the critical sets.

Based on these findings, we provide algorithms to prove the diagnosability under fairness. The approach is built on top of the classical twin plant construction, and generalizes it to cover the various forms of diagnosability and find sufficient delays.

The proposed algorithms are implemented within the xSAP platform for safety analysis, leveraging efficient symbolic model checking primitives. An experimental evaluation on a heterogeneous set of realistic benchmarks from various application domains demonstrates the effectiveness of the approach.

Keywords: diagnosis, diagnosability, fair transition systems, symbolic model-checking

*Corresponding author
Email addresses: benjamin.bittner@gmx.it (Benjamin Bittner), bozzano@fbk.eu (Marco Bozzano), cimatti@fbk.eu (Alessandro Cimatti), marco@gario.org (Marco Gario), tonettas@fbk.eu (Stefano Tonetta), vvozarova@fbk.eu (Viktoria Vozarova)

1. Introduction

In many applications, ranging from industrial plants (e.g. production, power) to transportation (e.g. railways, avionics, space), the state of the system is not fully observable. This is a problem when conditions that are important for the correct operation of the system (e.g. faults) are not directly measurable, but instead need to be inferred from the values obtained from the available sensors. The process of inferring such hidden information is often referred to as *diagnosis* [1]. Diagnosis can be carried out "post-mortem", as an off-line analysis from recorded traces, or on-line, in parallel with the control process. In the latter case, the objective is often to detect whether the system under diagnosis, also referred to as plant, is operating correctly (i.e. in nominal conditions) or whether a fault has occurred, and possibly to identify which fault occurred. This process is also known as *fault detection and identification* (FDI). The availability of accurate information provided by the FDI may be crucial to preserve the overall integrity of the system, so that adequate recovery countermeasures (e.g. activation of backup components or reconfiguration) can be applied in a timely manner (see for instance [2]).

The Diagnosability Problem. The feasibility of implementing an accurate and timely FDI solution is commonly known as *diagnosability* [3]. The problem of *verification of diagnosability* amounts to checking whether the available sensors are sufficient to infer – in all relevant operational conditions – some desired information (e.g. the presence of a fault) on the hidden behavior of the system. Diagnosability verification (also known as "diagnosability testing", or "diagnosability checking") may confirm that the choice of sensors is adequate or may pinpoint conditions in which diagnosis will not be possible. Historically, diagnosability checking has been reduced to showing the absence of a *critical pair* [4, 5], i.e. a pair of executions that are observationally indistinguishable and witness the violation of diagnosability: one of them contains a condition that should be detected, while the other one does not. Thus, it is impossibile for a diagnoser to ascertain whether the condition actually occurred.

Diagnosabilty under Fairness Conditions. In this paper, we study diagnosability taking into consideration the fundamental notion of *fairness*, i.e. the property that a certain event could (be required to) occur an infinite number of times in an execution trace. Fairness allows to characterize models at a higher level of abstraction, before the implementation choices have been made, and to express important operational conditions. Fairness can also be used to encode some context expressed in temporal logic, restricting the behavior of the plant to define some boundaries on the diagnosability verification.

Overall, we adopt a rich formal framework for diagnosability [6, 7]. The system under diagnosis is modeled as a finite-state fair transition system, synchronously[1] connected to a diagnoser able to observe a given set of variables.

[1] Most of the discrete-event systems approaches rely on the assumption that the connection

Our characterization of the diagnosability problem takes into account temporally extended diagnosis conditions, the operational context, and various forms (e.g. exact, bounded, finite) of diagnosability delay. We adopt and extend an industrially validated pattern language for the specification of requirements over FDI components [9].

Contributions of the paper. We provide a full formal account of the problem, and we make three main contributions.

1) Critical pairs are not enough. First, we show that, due to the expressiveness of the adopted framework, verification of diagnosability can not be reduced, as often done, to proving the absence of *critical pairs*. Although the presence of a critical pair is sufficient for diagnosability violation, quite surprisingly it turns out not to be necessary. We show that there are cases where failure to diagnosability is witnessed by a generalized version of critical pair, namely a *critical set*. We prove that the existence of a critical set is a necessary and sufficient condition for non-diagnosability. We show that critical sets, in general, may have infinite cardinality, due to the fairness conditions. Nevertheless, we identify finite representations of such critical sets, called *ribbon-shape critical pairs*, which give rise to decision procedures for the diagnosability problem also in the presence of fairness.

2) Algorithms for Verification. Our second contribution is to tackle verification of diagnosability by providing for each pattern a decision procedure based on model checking. The approach follows the well known *twin plant* construction, where two replicas of the plant are composed to encode the space of indistinguishable traces and to produce critical pairs. The existence of a simple critical pair for a given plant is reduced to checking whether a suitable formula in linear temporal logic with past operators [10, 11] holds over the corresponding twin plant. Instead, a ribbon-shape critical pair has a branching structure and its existence is verified with a dedicated algorithm using model checking of computation-tree logic with fairness [10] as subroutine. The basic twin plant construction is adapted to encompass all the features of the proposed framework, including temporally extended diagnosis conditions, operational context and delays. In order to cover the various diagnosability patterns, the proposed twin plant construction is extended to deal with critical sets. The algorithms are shown to be sound and complete.

Moreover, we propose an integrated algorithm that carries out a comprehensive diagnosability analysis for a given diagnosis condition. In particular, it constructs a complete map that identifies which of the various forms of diagnosability hold, and for which delays. This is done by combining different

is asynchronous. Our choice is not a limitation in itself, as it is possible to lift the proposed approach to the asynchronous setting. It is possible to follow an approach similar to what was done in [8], that extends the synchronous setting of [7] to the asynchronous case. See also the discussion in Section 10.

calls to the algorithms for the individual patterns, orchestrated by deductions justified by several theoretical results on how some forms of diagnosability can be inferred by others. The algorithm provides an integrated view of the diagnosability properties holding for a given condition.

3) Implementation and Experimental Evaluation. The third contribution is a practical evaluation of the formal framework. We implemented the proposed diagnosability algorithms in xSAP [12]. xSAP is a platform for the formal safety analysis of critical systems, with specific functions for the design, verification and synthesis of FDI components, and has been applied to numerous industrial sized case studies [13, 14, 15, 16, 17]. xSAP is the core engine of the COMPASS system, jointly developed with the European Space Agency. The baseline of the framework used in this paper has been validated with the involvement of industrial partners within several projects in the space sector [18, 19, 20, 21].

xSAP builds on top of nuXmv [22], a symbolic model checker supporting the representation and the automated analysis of finite-state (and infinite-state) fair transition systems. In this setting, transition systems are symbolically expressed in form of logical formulae, and efficient symbolic model checking algorithms can be reused. In particular, we implemented the new algorithms for ribbon-shape critical pairs on top of standard BDD-based procedures for CTL symbolic model checking [23].

We carried out an evaluation on a comprehensive set of realistic benchmarks from various application domains. The results clearly demonstrate the effectiveness of the approach on benchmarks of industrial size.

Structure of the paper. The paper is structured as follows. In Section 2 we present the framework of symbolic transition systems and temporal logics. In Section 3 we define diagnosability and the problem of diagnosability verification, and we present and illustrate our system of patterns. In Section 4 we introduce critical pairs. In Section 5 we define critical sets and analyze the relationships with critical pairs. In Section 6 we define ribbon-shape critical pairs. In Section 7 we present an integrated approach to diagnosability verification built on top of the twin plant construction and its extensions. In Section 8 we present our integrated algorithm for diagnosability. In Section 9 we describe the implementation, and experimentally evaluate the proposed methods on a set of benchmarks from various domains. In Section 10 we contrast our approach with the relevant literature. In Section 11 we draw some conclusions and outline directions for future activities.

2. Background

We now describe the formal framework used in this paper. We formalize partially observable plants and introduce the temporal logic LTL, which can be used to express properties of such systems, and the temporal logic CTL, which is used in the reduction of the diagnosability problem to model checking.

2.1. Preliminaries

We work in the setting of propositional logic [24]. Atomic propositions are Boolean variables. A formula is either an atomic proposition or the application of a Boolean connective (negation $\neg$ or disjunction $\vee$) to formulae. We use the following standard abbreviations: $\phi \wedge \psi$ for $\neg(\neg\phi \vee \neg\psi)$, $\phi \rightarrow \psi$ for $\neg\phi \vee \psi$, and $\phi \leftrightarrow \psi$ for $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$. With a slight abuse of notation we may sometimes write $P = Q$ instead of $P \leftrightarrow Q$ when P and Q are propositional variables. We use the standard notions of assignment, model, and logical consequence. Given an assignment μ to a set of variables X, and $X_1 \subseteq X$, we denote the projection of μ over X_1 with $\mu_{|X_1}$.

We write $\phi(X)$ to stress that ϕ is a formula over the variables in X. A formula ϕ can be simplified with respect to a (partial) assignment μ, by carrying out a parallel substitution of each of the variables x with the constant $\mu(x)$. Such a simplified formula is referred to as the restriction of ϕ with respect to μ, and is denoted as $\phi_{\downarrow\mu}$. We write $\mu \models \phi$ to indicate that μ satisfies all models of ϕ, i.e. $\phi_{\downarrow\mu} \equiv \top$; we write $\psi \models \phi$ to indicate that all models satisfying ψ also satisfy ϕ; finally, we write $\models \phi$ to indicate that any model satisfies ϕ, i.e. that ϕ is a validity.

2.2. Symbolic Fair Transition Systems

A plant P is represented as a finite-state *symbolic fair transition system* (SFTS). An SFTS is a tuple $\langle V, V_o, I, T, F\rangle$, where V is a finite set of Boolean[2] state variables; $V_o \subseteq V$ is the set of observable state variables; I is a formula over V defining the initial states, T is a formula over V, V' (with V' being the next version of the state variables) defining the transition relation, and F is a set of formulas over V defining the fairness conditions.

We remark that the choice of representing the plant in form of a Fair Transition System does not restrict the generality of the framework. In fact, it is possible to encode labeled transition systems and discrete event systems [8].

A *state* s is an assignment to the state variables V. We denote with s' the corresponding assignment to V'. Given an assignment to a set V of Boolean variables, we also represent the assignment as the set of variables that are assigned to true.

In the following we assume that an SFTS $P \doteq \langle V, V_o, I, T, F\rangle$ is given.

The *observable part* $obs(s)$ of a state s is the projection of s on the subset V_o of observable state variables. Thus, $obs(s) \doteq s_{|V_o}$. Given a sequence of states σ, we denote with σ^k the sequence obtained by repeating σ k times, and σ^ω the sequence obtained by repeating σ an infinite number of times.

Given a state s_0 of P, a *trace of P starting from* s_0 is an infinite sequence $\pi \doteq s_0, s_1, s_2, \ldots$ of states starting from s_0 such that, for each $k \geq 0$, $\langle s_k, s'_{k+1}\rangle$

[2]Without loss of generality, we consider Boolean variables only. In our exposition we sometimes use formulae with variables ranging over finite domains, including theory variables and arithmetic operators over them. Such formulae may be encoded using Boolean variables as described in [25].

satisfies T, and for all $f \in F$, for infinitely many $i \geq 0$, $s_i \models f$. The last condition requires that each fairness condition f is satisfied in infinitely many (but not necessarily all) states of the trace. If s_0 is initial, i.e., it satisfies I, then we say that π is a *trace of* P. We write Π_P for the set of traces of P.

We denote with $\pi[k]$ the $(k+1)$-th state s_k of π. We use $\pi[k,l]$ to denote a subtrace of π from $\pi[k]$ to $\pi[l]$ included. We use $\pi[k,\infty]$ to denote the suffix of π starting from $\pi[k]$. We say that s is *reachable* (in k steps) in P iff there exists a trace $\pi \in \Pi_P$ such that $s = \pi[k]$ for some $k \geq 0$. A state s is fair if there exists a trace starting from s. Note that we consider infinite fair traces only.

Given a trace $\pi \doteq s_0, s_1, s_2, \ldots$ and a subset of variables $W \subseteq V$, we denote by $\pi_{|W} \doteq s_{0|W}, s_{1|W}, s_{2|W}, \ldots$ the projection over the variables in W. The observable part of π is $obs(\pi) \doteq obs(s_0), obs(s_1), obs(s_2), \ldots = \pi_{|V_o}$. Given two traces π_1 and π_2, we denote by $\textsc{ObsEqUpTo}(\pi_1, \pi_2, k)$ the condition saying that, for all i, $0 \leq i \leq k$, $obs(\pi_1[i]) = obs(\pi_2[i])$.

We say that S is observable-deterministic if i) there are no two different initial states s_{0_1} and s_{0_2} s.t. $obs(s_{0_1}) = obs(s_{0_2})$, ii) there are no two different transitions $\langle s, s'_1 \rangle$ and $\langle s, s'_2 \rangle$ from a reachable state s s.t. $obs(s'_1) = obs(s'_2)$.

Let $S^1 = \langle V^1, V^1_o, I^1, T^1, F_1 \rangle$ and $S^2 = \langle V^2, V^2_o, I^2, T^2, F_2 \rangle$ be two SFTSs with $(V^1 \setminus V^1_o) \cap V^2 = \emptyset = V^1 \cap (V^2 \setminus V^2_o)$. The equalities to empty set mean that the internal set of one system does not overlap with the observable set of the other system. We define the *synchronous product* $S^1 \times S^2$ as the SFTS $\langle V^1 \cup V^2, V^1_o \cup V^2_o, I^1 \wedge I^2, T^1 \wedge T^2, F_1 \cup F_2 \rangle$. Every state s of $S^1 \times S^2$ is an assignment to the two sets of state variables V^1 and V^2 such that $s^1 = s_{|V^1}$ is a state of S^1 and $s^2 = s_{|V^2}$ is a state of S^2.

We say that S^1 *does not constrain* S^2 iff i) for every initial state s^2 of S^2, there exists an initial state s^1 of S^1 such that $s^1_{|V^1_o \cap V^2_o} = s^2_{|V^1_o \cap V^2_o}$ and ii) for every reachable state $s^1 \times s^2$ of $S^1 \times S^2$, for every transition $\langle s^2, s'^2 \rangle$ of S^2 to a fair state s'^2, there exists a transition $\langle s^1, s'^1 \rangle$ to a fair state s'^1 such that $s'^1_{|V^1_o \cap V^2_o} = s'^2_{|V^1_o \cap V^2_o}$.

2.3. Temporal Logic

2.3.1. LTL

We use Linear Temporal Logic (LTL) with Past operators [11] to specify diagnosability patterns. Atomic propositions may be built over V, and represent sets of system states.

The syntax of LTL formulae is defined as follows. If ϕ is an atomic proposition, then ϕ is an LTL formula. If ϕ and ψ are LTL formulae, so are $\neg\phi$ (negation), $\phi \vee \psi$ (disjunction), $\psi \mathsf{U} \phi$ (sometime in the future ϕ and, until then, ψ), $\psi \mathsf{S} \phi$ (sometime in the past ϕ and, since then, ψ), $\mathsf{X}\phi$ (in the next state ϕ), $\mathsf{Y}\phi$ and $\mathsf{Z}\phi$ (in the previous state ϕ, respectively default to false and true if there is no previous state). We denote the size of a formula ϕ as $|\phi|$, which represents the number of symbols of the formula. We interpret LTL formulae over infinite traces $\pi = s_0, s_1, s_2, \ldots$. The semantics of LTL is as follows:

- $\pi, i \models p$, where p is an atomic proposition, iff $\pi[i] \models p$.

- $\pi, i \models \phi \vee \psi$ iff $\pi, i \models \phi$ or $\pi, i \models \psi$.
- $\pi, i \models \neg\phi$ iff $\pi, i \not\models \phi$.
- $\pi, i \models \psi \mathsf{U} \phi$ iff there exists $j \geq i$ such that $\pi, j \models \phi$ and, for all $i \leq h < j$, $\pi, h \models \psi$.
- $\pi, i \models \psi \mathsf{S} \phi$ iff there exists $0 \leq j \leq i$ such that $\pi, j \models \phi$ and, for all $j < h \leq i$, $\pi, h \models \psi$.
- $\pi, i \models \mathsf{X}\phi$ iff $\pi, i+1 \models \phi$.
- $\pi, i \models \mathsf{Y}\phi$ iff $i > 0$ and $\pi, i-1 \models \phi$.
- $\pi, i \models \mathsf{Z}\phi$ iff $i = 0$ or $\pi, i-1 \models \phi$.

We use the following standard abbreviations: $\top \doteq p \vee \neg p$ for some proposition p, $\bot \doteq \neg\top$, $\phi \wedge \psi \doteq \neg(\neg\phi \vee \neg\psi)$, $\mathsf{F}\phi \doteq \top \mathsf{U} \phi$ (sometime in the future ϕ), $\mathsf{G}\phi \doteq \neg\mathsf{F}\neg\phi$ (always in the future ϕ), $\mathsf{O}\phi \doteq \top \mathsf{S} \phi$ (once in the past ϕ), $\mathsf{H}\phi \doteq \neg\mathsf{O}\neg\phi$ (historically in the past ϕ), $\mathsf{X}^n\phi \doteq \mathsf{X}\mathsf{X}^{n-1}\phi$ with $\mathsf{X}^0\phi \doteq \phi$, $\mathsf{Y}^n\phi \doteq \mathsf{Y}\mathsf{Y}^{n-1}\phi$ with $\mathsf{Y}^0\phi \doteq \phi$, $\mathsf{F}^{\leq n}\phi \doteq \phi \vee \mathsf{X}\phi \vee \cdots \vee \mathsf{X}^n\phi$, $\mathsf{G}^{\leq n}\phi \doteq \phi \wedge \mathsf{X}\phi \wedge \cdots \wedge \mathsf{X}^n\phi$, $\mathsf{O}^{\leq n}\phi \doteq \phi \vee \mathsf{Y}\phi \vee \cdots \vee \mathsf{Y}^n\phi$, $\mathsf{H}^{\leq n}\phi \doteq \phi \wedge \mathsf{Z}\phi \wedge \cdots \wedge \mathsf{Z}^n\phi$.

A trace π satisfies an LTL formula ϕ (denoted by $\pi \models \phi$) iff ϕ is true in π at step 0 (i.e., $\pi, 0 \models \phi$). We denote by Π_ϕ the set of traces π that satisfy ϕ.

The model-checking problem $P \models \phi$, read "ϕ holds in P", amounts to checking if ϕ holds in all the traces of P.

An LTL formula of the form $\mathsf{G}\phi$ where ϕ does not contain any temporal operators is called an invariant property. ϕ is interpreted on every single state of a trace, and the property is falsified if any such state does not satisfy ϕ.

Given a SFTS P with variables V and an LTL formula ϕ over V, we can build a SFTS P_ϕ with variables $V_\phi \supseteq V$ and a propositional formula $Enc(\phi)$ over V_ϕ such that :

1. for every trace π of P, there exists a trace π_ϕ of P_ϕ such that $\pi_{\phi|V} = \pi$;
2. for every trace π_ϕ of P_ϕ, $\pi_{\phi|V}$ is a trace of P;
3. for every trace π_ϕ of P_ϕ, for all $i \geq 0$, $\pi_\phi[i] \models Enc(\phi)$ iff $\pi_{\phi|V}, i \models \phi$;
4. $|V_\phi \setminus V|$ is linear in the number of temporal operators of ϕ.

We can deduce that $P \models \phi$ iff P_ϕ has no trace π_ϕ such that $\pi_\phi[0] \models \neg Enc(\phi)$. Intuitively, P_ϕ is the product of P with the automaton built from ϕ, but, moreover, it can evaluate ϕ in every state of the trace. Such construction can be easily derived from [26][3].

[3]For the sake of completeness, the construction described in [26] deals only with future operators, but has been extended to past operators and implemented in NuSMV [27].

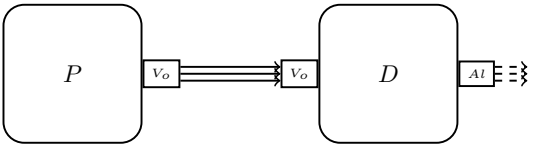

Figure 1: A plant P and its diagnoser D.

2.3.2. CTL

We use Computation-Tree Logic (CTL) to specify branching-time model checking problems. To this purpose, we only need future operators.

The syntax of CTL formulae is defined as follows. If ϕ is an atomic proposition, then ϕ is a CTL formula. If ϕ and ψ are CTL formulae, so are $\neg\phi$ (negation), $\phi \vee \psi$ (disjunction), $\mathsf{E}(\psi \mathsf{U} \phi)$ (there exists a trace where sometime in the future ϕ and, until then, ψ), $\mathsf{EX}\phi$ (there exists a trace where in the next state ϕ), $\mathsf{EG}\phi$ (there exists a trace where ϕ always holds). We interpret CTL formulae over a state s of an SFTS P as follows:

- $P, s \models p$, where p is an atomic proposition, iff $s \models p$.
- $P, s \models \phi \vee \psi$ iff $P, s \models \phi$ or $P, s \models \psi$.
- $P, s \models \neg\phi$ iff $P, s \not\models \phi$.
- $P, s \models \mathsf{EG}\phi$ iff there exists a trace $s_0, s_1, s_2, \ldots$ starting from $s = s_0$ such that, for all $i \geq 0$, $P, s_i \models \phi$.
- $P, s \models \mathsf{E}(\psi \mathsf{U} \phi)$ iff there exists a trace $s_0, s_1, s_2, \ldots$ starting from $s = s_0$ and $j \geq 0$ such that $P, s_j \models \phi$ and, for all $0 \leq h < j$, $P, s_h \models \psi$.
- $P, s \models \mathsf{EX}\phi$ iff there exists a trace $s_0, s_1, s_2, \ldots$ starting from $s = s_0$ such that $P, s_1 \models \phi$.

We use the following standard abbreviations: $\top \doteq p \vee \neg p$ for some proposition p, $\bot \doteq \neg\top$, $\phi \wedge \psi \doteq \neg(\neg\phi \vee \neg\psi)$, $\mathsf{EF}\phi \doteq \mathsf{E}(\top \mathsf{U} \phi)$, $\mathsf{AG}\phi \doteq \neg\mathsf{EF}\neg\phi$.

The model-checking problem $P \models \phi$, read "ϕ holds in P", amounts to checking if ϕ holds in all initial states of P.

3. The Diagnosability Framework

We now define the problem of verification of diagnosability. First, in Section 3.1 we define the framework for diagnoser specification; then, in Section 3.2, we discuss the notion of diagnosability [3], building upon the framework defined in [7].

3.1. Specifying a diagnoser

Diagnosability is best understood in the context of the design of a diagnoser. Consider Figure 1. The left box represents the plant, with the V_o box representing its observable interface. The right box represents the diagnoser (the FDI component). The diagnoser is a module that runs in parallel with the plant, driven by the observable variables V_o, and raises some alarms $Al_0, \ldots, Al_n$ in correspondence to conditions of interest, called diagnosis conditions. The behaviour of the diagnoser is typically specified by a set of requirements, describing the relationship between the alarms to be raised and the occurrence of the corresponding diagnosis conditions. It is in principle possible that the requirements are impossible to satisfy, for example because the available sensors do not convey sufficient information to appropriately raise the alarms in a correct and timely fashion. Verification of diagnosability can be intuitively seen as a form of requirements validation, i.e. checking whether the requirements of the diagnoser can indeed be realized.

3.1.1. Assumptions

The following assumptions are used in the rest of the paper. The plant is an open system, responding to an environment which drives some of its signals. For example, the environment may be an abstraction of a control policy that could be attached to the system.

The diagnoser is an observable-deterministic transition system having access to the observable signals of the plant. Formally, given a set $\mathcal{A}$ of alarms and a plant $P = \langle V^P, V_o^P, I^P, T^P, F^P \rangle$, a diagnoser for P and $\mathcal{A}$ is an observable-deterministic transition system $D = \langle V^D, V_o^D, I^D, T^D, \emptyset \rangle$ which does not constrain P and such that $V_o^P \subseteq V_o^D$ and $\mathcal{A} \subseteq V_o^D$.

The plant and the diagnoser are synchronously connected, denoted $P \times D$. Since, by definition, the diagnoser does not constrain the plant, for each transition of the plant there is a corresponding transition of the diagnoser. In other words, the diagnoser does not influence the behavior of the system. The diagnoser keeps track of the history of the plant, but has finitely many states, and thus a bounded amount of memory.

3.1.2. Diagnosis Conditions

In order to specify FDI requirements, we first need a condition on the plant that must be monitored. We call such condition *diagnosis condition*, denoted with β. In order for the diagnosis condition to be evaluated by a diagnoser, we require that β encodes a property on the past, i.e., given a trace π and a point k, the truth of β in k depends only on the prefix of π up to $\pi[k]$ (i.e. the truth of β is the same for any π' such that, for all i with $0 \leq i \leq k$, $\pi'[i] = \pi[i]$). Formally, a *diagnosis condition* for P is any formula β built according to the following rule: $\beta ::= p \mid \beta_1 \vee \beta_2 \mid \neg\beta \mid \mathsf{Y}\beta \mid \beta_1 \mathsf{S} \beta_2$ where p is an atomic proposition over V.

In this formalism, we can express complex diagnosis conditions. We list some sample conditions in Figure 2, where we use c_i and f_i to denote general and

$$
\begin{array}{lcl}
\beta_1 & \doteq & f_1 \vee \ldots \vee f_n \\
\beta_2 & \doteq & f_3 \\
\beta_3 & \doteq & Q \wedge c_3 \wedge \mathsf{Y}(c_2 \wedge \mathsf{Y}c_1) \\
\beta_4 & \doteq & f_2 \wedge \mathsf{O}^{\leq 3} f_1 \\
\beta_5 & \doteq & c_1 \wedge \mathsf{Y}(\neg c_2 \mathsf{S} c_1)
\end{array}
$$

Figure 2: Examples of Diagnosis Conditions.

fault conditions. First, it is possible to model fault detection (whether any fault has occurred), with $f_1 \vee \ldots \vee f_n$, and fault identification (which of the possible faults has occurred), with f_i for a specific i. We might also want to restrict the detection to a particular sub-system, or identification among two similar faults might not be of interest. Second, we can combine information relative to states and transitions. For example, $Q \wedge c_3 \wedge \mathsf{Y}(c_2 \wedge \mathsf{Y}c_1)$ defines the sequence of conditions $c_1; c_2; c_3$ leading to a state where Q holds. Third, it is possible to express sequences of relevant situations, e.g. f_2 preceded of up to three ticks by (or simultaneous with) f_1, by stating $f_2 \wedge \mathsf{O}^{\leq 3} f_1$. Finally, we can express conditions as in the example proposed in [28] to detect the event of going twice to the coffee-shop (denoted by c_1) without going to the library (denoted by c_2), which can be expressed with a formula like $c_1 \wedge \mathsf{Y}(\neg c_2 \mathsf{S} c_1)$.

As described in Section 2.3.1, any LTL formula φ can be transformed into an automaton and the original plant P can be extended to P_φ such that at every point of a trace we can decide if φ is true or false. Thus, in the rest of the paper we assume that P is already extended with the fault condition and we assume β is propositional.

3.1.3. Alarm Conditions

An *alarm condition* expresses the relation between a diagnosis condition and the raising of an alarm. It is composed of two parts: the diagnosis condition and the delay. The delay relates the time between the occurrence of the diagnosis condition and the corresponding alarm. It may be the case that the occurrence of a fault goes undetected for a certain amount of time. Hence, it is important to specify the maximum length of this interval.

We express alarm conditions using formulae that we call *alarm condition patterns*. We consider four different patterns, which we denote with $\textsc{ExactDel}(Al, \beta, d)$, $\textsc{BoundDel}(Al, \beta, d)$, $\textsc{FiniteDel}(Al, \beta)$ and $\textsc{BoundDel}_{\mathsf{O}}(Al, \beta, d)$. The first three patterns we define are taken from [8], while the fourth captures the definition of diagnosability given in [3]. All patterns have as arguments the name of the alarm being defined and the diagnosis condition; morevoer, all patterns except FiniteDel have the delay as argument. We give the following definition.

Definition 1 (Alarm condition patterns)**.** *Given an alarm Al, a diagnosis condition β and a delay $d \in \mathbb{N}$, alarm condition patterns are defined as follows.*

1. $\textsc{ExactDel}(Al, \beta, d)$ *states that, for any trace π, if Al is true in $\pi[i]$, then*

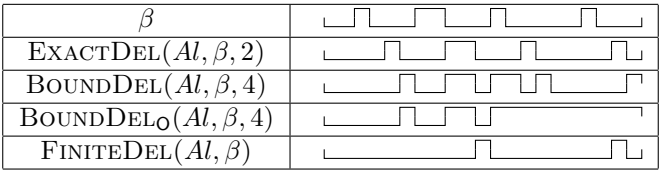

Figure 3: Examples of alarm responses to the diagnosis condition β.

β must be true in $\pi[i-d]$ (correctness); if β is true in $\pi[i]$, then Al is true in $\pi[i+d]$ (completeness).

2. BoundDel(Al, β, d) *states that, for any trace π, if Al is true in $\pi[i]$, then β must be true in $\pi[j']$ for some $i - d \leq j' \leq i$ (correctness); if β is true in $\pi[i]$, then Al is true in $\pi[j]$, for some $i \leq j \leq i + d$ (completeness).*
3. BoundDel$_{\mathsf{O}}(Al, \beta, d)$ *states that, for any trace π, if Al is true in $\pi[i]$, then β must be true in $\pi[j']$ for some $j' \leq i$ (correctness); if β is true in $\pi[i]$, then Al is true in $\pi[j]$ for some $i \leq j \leq i + d$ (completeness).*
4. FiniteDel(Al, β) *states that, for any trace π, if Al is true in $\pi[i]$, then β must be true in $\pi[j']$ for some $j' \leq i$ (correctness); if β is true in $\pi[i]$, then Al is true in $\pi[j]$ for some $j \geq i$ (completeness).*

Intuitively, ExactDel(Al, β, d) specifies that whenever β is true, Al must be triggered exactly d steps later and Al can be triggered only if d steps earlier β was true. BoundDel(Al, β, d) specifies that whenever β is true, Al must be triggered within the next d steps and Al can be triggered only if β was true within the previous d steps. BoundDel$_{\mathsf{O}}(Al, \beta, d)$ specifies that whenever β is true, Al must be triggered within the next d steps and Al can be triggered only if β was true in some previous step. Finally, FiniteDel(Al, β) specifies that whenever β is true, Al must be triggered in a later step and Al can be triggered only if β was true in some previous step.

Figure 3 provides an example of admissible responses for the various alarms to the occurrences of the same diagnosis condition β. In the case of BoundDel$(Al, \beta, 4)$ the alarm must be triggered within the next 4 time-steps after each occurrence of β. This is quite different from having 5 different alarms ExactDel$(Al_0, \beta, 0), \ldots,$ ExactDel$(Al_4, \beta, 4)$ and considering their disjunction $(Al_0 \vee \cdots \vee Al_4)$, since the latter requires that we are always able to exactly pin-point the moment in which the diagnosis condition occurred, while the former provides a degree of flexibility. In the case of BoundDel$_{\mathsf{O}}(Al, \beta, 4)$ the alarm must be triggered within the next 4 time-steps after each occurrence of β, however it is allowed to be triggered at any time provided β occurred in the past (e.g. an alarm which is triggered within 4 time-steps after the first occurrence of β and then stays always true, is compliant with the definition). Finally, FiniteDel(Al, β) is of particular theoretical interest since it captures the idea that, in some systems, the delay might be finite but unbounded. Note that the ExactDel(Al, β, d) pattern may be useful when the delay itself conveys relevant diagnostic information, e.g. to determine in what system mode some

Alarm Condition	LTL Formulation	
	Correctness	Completeness
$\textsc{ExactDel}(Al, \beta, d)$	$\mathsf{G}(Al \rightarrow \mathsf{Y}^d \beta)$	$\mathsf{G}(\beta \rightarrow \mathsf{X}^d Al)$
$\textsc{BoundDel}(Al, \beta, d)$	$\mathsf{G}(Al \rightarrow \mathsf{O}^{\leq d} \beta)$	$\mathsf{G}(\beta \rightarrow \mathsf{F}^{\leq d} Al)$
$\textsc{BoundDel}_{\mathsf{O}}(Al, \beta, d)$	$\mathsf{G}(Al \rightarrow \mathsf{O}\beta)$	$\mathsf{G}(\beta \rightarrow \mathsf{F}^{\leq d} Al)$
$\textsc{FiniteDel}(Al, \beta)$	$\mathsf{G}(Al \rightarrow \mathsf{O}\beta)$	$\mathsf{G}(\beta \rightarrow \mathsf{F} Al)$

Figure 4: Alarm conditions as LTL formulae.

fault occurred which in turn might affect the choice of an appropriate response. This is particularly important when mitigation actions have to be carried out during system operation.

Finally, note that, in presence of fairness, traces are assumed to be fair, hence alarm conditions patterns constrain the occurrence of an alarm on sequences of states satisfying the fairness conditions. Thus, these conditions can be used, e.g., to constrain the operational context under which we want to analyze diagnosability (compare Section 3.1.4). Examples of such use of fairness conditions are illustrated in Section 3.5.

The meaning of the alarm conditions is further clarified in Figure 4, where we associate to each alarm condition type an LTL formalization encoding the concepts of correctness and completeness. The first conjunct expresses *correctness*, and intuitively says that whenever the diagnoser raises an alarm, then the fault must have occurred. *Completeness*, the second conjunct, intuitively encodes that whenever the fault occurs, the alarm will be raised.

We denote *alarm conditions* with Ξ. An alarm condition constrains the behavior of the diagnoser to respond to conditions that occur in the plant. These are typically non-observable conditions, and thus the diagnoser may need to infer their occurrence by only relying on the observable behavior of the plant. Thus, when we say that a diagnoser D for P satisfies an alarm condition Ξ, we mean that the composition $D \times P$ between D and P satisfies the LTL formulae corresponding to Ξ in Figure 4 (we say that the diagnoser is *correct* and *complete*).

All the alarm conditions in Def. 1, but the last one, take a delay d as parameter. We can naturally define the following existential patterns, by quantifying the delay existentially. For instance, $\exists\textsc{BoundDel}(Al, \beta)$ specifies that Al must be triggered if there exists a delay d such that $\textsc{BoundDel}(Al, \beta, d)$.

Definition 2 (Existential alarm condition patterns)**.** *Given an alarm Al and a diagnosis condition β, existential alarm condition patterns are defined as follows.*

1. $\exists\textsc{ExactDel}(Al, \beta)$ *specifies that Al is triggered if and only if there exists a delay $d \in \mathbb{N}$ such that* $\textsc{ExactDel}(Al, \beta, d)$ *holds.*
2. $\exists\textsc{BoundDel}(Al, \beta)$ *specifies that Al is triggered if and only if there exists a delay $d \in \mathbb{N}$ such that* $\textsc{BoundDel}(Al, \beta, d)$ *holds.*
3. $\exists\textsc{BoundDel}_{\mathsf{O}}(Al, \beta)$ *specifies that Al is triggered if and only if there exists a delay $d \in \mathbb{N}$ such that* $\textsc{BoundDel}_{\mathsf{O}}(Al, \beta, d)$ *holds.*

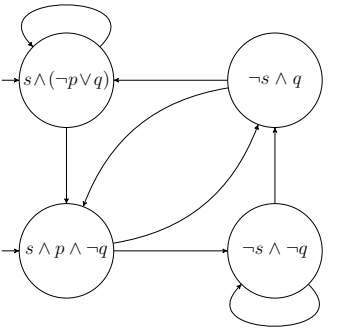

Figure 5: Symbolic fair transition system for the LTL formula $\mathsf{G}(p \rightarrow \mathsf{F}q)$.

3.1.4. Context

A *context* constrains the operational setting of the environment in which we place the plant. For instance, consider the situation in which multiple concurrent faults are extremely unlikely to happen. From the engineering point of view, it might make sense to study the diagnosability of the system under a single-fault assumption. For this reason we introduce the concept of context (denoted by Ψ) that represents a set of traces Π_Ψ that are used to restrict the plant's traces (i.e., the diagnosability definitions quantify over traces in $\Pi_\Psi \cap \Pi_P$). In this paper we assume that the traces in a context can be characterized by an LTL property, similarly to the approach of [5].

Note that, thanks to the fact that the framework supports plants with fairness constraints, the incorporation of the context is straightwforward. In other words, we can replace the plant P by P_Ψ (see definition in Section 2) and consider the diagnosability with no context without loss of generality. Hence, we feel free to omit the context in some theoretical statements and theorems to simplify the notation.

However, in general, it may be useful to analyze the same model with different contexts. Hence, in practice we prefer to consider the context explicitly, rather than modify the model to accommodate different diagnosability analyses.

Assume we have a plant P and the context is the LTL formula $\Psi = \mathsf{G}(p \rightarrow \mathsf{F}q)$. Figure 5 shows an example of an SFTS for the context. The system has three state variables s, p, q, where p, q refer to state variables of P and s is a fresh variable that is not in P. We consider four symbolic states. The top-left state signifies that either there was no occurrence of p or it occurred together with q. The bottom-left state means that there was p and not q. From this state, we move to the right side, either to the top-right state, if q comes in the next step, or to the bottom-right state, where we wait for the occurrence of q. From there, we move back to one of the initial states.

The initial formula is s and the set of fairness conditions is $\{s\}$, i.e., there is a single fairness condition s. The transition formula encodes the transition depicted in the figure. More specifically, it is a conjunction of implications,

where each implication describes transitions for one state:

$$\begin{aligned} \big(s \wedge (\neg p \vee q)\big) &\rightarrow s', \\ (s \wedge p \wedge \neg q) &\rightarrow \neg s', \\ (\neg s \wedge q) &\rightarrow s', \\ (\neg s \wedge \neg q) &\rightarrow \neg s'. \end{aligned}$$

Note that the transitions do not imply the value of variables p', q'. Thus, in the composition P_Ψ of the described system and the plant P, the value of p, q depends only on transitions of P. Besides the fairness conditions of P, P_Ψ has a new fairness condition s that ensures that each trace of the system satisfies the context.

3.2. The Diagnosability Problem

Let an alarm condition Ξ be given. The problem of verification of diagnosability for Ξ amounts to checking whether there exists a diagnoser that satisfies Ξ. In general, the problem of diagnosability amounts to checking the existence of a diagnoser for a diagnoser specification expressed as a set of alarm conditions. In the following we assume that alarm conditions are analyzed individually, and replace the alarm signal *Al* with $\circ$, since it is irrelevant for diagnosability.

Definition 3 (Diagnosability)**.** *Let P be a plant, and $d \in \mathbb{N}$.*

1. *β is* ExactDel diagnosable *with delay d iff for every trace π_1 of P and index $i \geq 0$ such that $\pi_1, i \models \beta$, all traces π_2 of P satisfy:*

$$\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d) \Rightarrow \pi_2, i \models \beta.$$

2. *β is* BoundDel diagnosable *with delay d iff for every trace π_1 of P and index $i \geq 0$ such that $\pi_1, i \models \beta$, there exists $j \in \mathbb{N}$ such that $i \leq j \leq i + d$ and all traces π_2 of P satisfy:*

$$\textsc{ObsEqUpTo}(\pi_1, \pi_2, j) \Rightarrow \exists k \in \mathbb{N}\ j - d \leq k \leq j \cdot \pi_2, k \models \beta.$$

3. *β is* $\textsc{BoundDel}_{\mathsf{O}}$ diagnosable *with delay d iff for every trace π_1 of P and index $i \geq 0$ such that $\pi_1, i \models \beta$, there exists $j \in \mathbb{N}$ such that $i \leq j \leq i + d$ and all traces π_2 of P satisfy:*

$$\textsc{ObsEqUpTo}(\pi_1, \pi_2, j) \Rightarrow \exists k \in \mathbb{N}\ k \leq j \cdot \pi_2, k \models \beta.$$

4. *β is* FiniteDel diagnosable *iff for every trace π_1 of P and index $i \geq 0$ such that $\pi_1, i \models \beta$, there exists $j \in \mathbb{N}$ such that $i \leq j$ and all traces π_2 of P satisfy:*

$$\textsc{ObsEqUpTo}(\pi_1, \pi_2, j) \Rightarrow \exists k \in \mathbb{N}\ k \leq j \cdot \pi_2, k \models \beta.$$

For simplicity, instead of saying "β is Ξ diagnosable (with delay d)", we say that the pattern Ξ is diagnosable. The following theorem shows an equivalent, and simpler formulation of diagnosability for the pattern $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$.

Theorem 1. *Let P be a plant and $d \in \mathbb{N}$ a delay. Then* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ *is* diagnosable *iff for every trace π_1 and index $i \geq 0$ such that $\pi_1, i \models \beta$, all traces π_2 of P satisfy* $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d) \Rightarrow \exists k \in \mathbb{N}\ k \leq i + d \cdot \pi_2, k \models \beta$.

Proof. Assume that the condition stated in Theorem 1 holds. Then, it is enough to take $j = i + d$ to show that $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ is *diagnosable* according to Def. 3.
Vice versa, assume that $\pi_1, i \models \beta$ and take π_2 s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$. Clearly, we have that $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ holds for every $j \leq i + d$. Hence, by Def. 3, we can conclude that there exists k s.t. $k \leq j$ and $\pi_2, k \models \beta$. The thesis follows from the fact that $j \leq i + d$. □

Diagnosability extends to existential patterns, as follows.

Definition 4 (Diagnosability for existential patterns)**.** *We say that* $\exists\textsc{ExactDel}(\circ, \beta)$ *[respectively,* $\exists\textsc{BoundDel}(\circ, \beta)$*,* $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$*] is* diagnosable *iff there exists a delay $d \in \mathbb{N}$ such that* $\textsc{ExactDel}(\circ, \beta, d)$ *[respectively,* $\textsc{BoundDel}(\circ, \beta, d)$*,* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$*] is* diagnosable.

The following theorem links the diagnoser specification to the diagnosability conditions: namely, if the diagnoser satisfies the specification, then the monitored plant must be diagnosable for that specification. As shown in [7], for finite-state plants, the converse is also true, i.e., if the specification is diagnosable then a diagnoser exists.

Theorem 2. *Let P be a plant, Ξ an alarm condition for alarm Al, and D a diagnoser for P and $\{Al\}$. Let us assume that every trace of $P \times D$ satisfies also Ξ. Then Ξ is diagnosable in P.*

Proof. This is a minor extension of the theorem proved in [7, 8], taking into account fairness. As an example, we prove the theorem for the BoundDel pattern. The other cases are similar. Let us assume that $\Xi = \textsc{BoundDel}(\circ, \beta, d)$ is not diagnosable. Then there exists a trace π_1 and an index i such that $\pi_1, i \models \beta$ and $\forall j.i \leq j \leq i + d$ there exists a trace π_2, s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ and $\forall k \in \mathbb{N}.j - d \leq k \leq j$ implies $\pi_2, k \not\models \beta$. Now, since β holds at i in π_1, the diagnoser has to raise an alarm at an index j such that $i \leq j \leq i+d$ on π_1. However, by hypothesis, there exists a trace π_2 which is observationally equivalent to π_1 up to j and that does not satisfy β in the interval $j - d \leq k \leq j$. Since π_2 is observationally equivalent to π_1 and the diagnoser is observable-deterministic, the diagnoser has to raise the alarm on π_2 at j. This leads to a contradiction, since β does not hold on π_2 in the interval $j - d \leq k \leq j$, hence the diagnoser must not raise the alarm. □

In the rest of this section we investigate the dependencies between alarm condition patterns and prove some related results.

3.3. Monotonicity of Alarm Conditions

We have the following result relating diagnosability patterns with the diagnosability delay.

Theorem 3 (Delay Monotonicity)**.** *For all plant P, diagnosis condition β and delay $d \in \mathbb{N}$:*

1. *if* $\textsc{ExactDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\textsc{ExactDel}(\circ, \beta, d')$ *for all* $d' > d$*;*
2. *if* $\textsc{BoundDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\textsc{BoundDel}(\circ, \beta, d')$ *for all* $d' > d$*;*
3. *if* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ *is diagnosable, then so is* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d')$ *for all* $d' > d$*.*

Proof. We rely on Definition 3 and show these properties by case.

1. Pick any π_1 and any $i \in \mathbb{N}$ such that $\pi_1, i \models \beta$. By hypothesis, for any other trace π_2 that has the same observations as π_1 up to $i+d$ it holds that $\pi_2, i \models \beta$. Since $d' > d$, any other trace π_2 that has the same observations as π_1 up to $i + d'$, it also has the same observations as π_1 up to $i + d$, hence $\pi_2, i \models \beta$.
2. Similarly, we can take the j and k witnessing the diagnosability of $\textsc{BoundDel}(\circ, \beta, d)$ to prove the diagnosability of $\textsc{BoundDel}(\circ, \beta, d')$.
3. Similarly to case 2.

□

3.4. Dependencies between Alarm Conditions

The following theorem relates diagnosability of different alarm condition patterns, given the same diagnosis condition and delay.

Theorem 4 (Dependencies between Alarm Conditions)**.** *Let P be a plant, β a diagnosis condition, and $d \in \mathbb{N}$ a delay. Then:*

1. *if* $\textsc{ExactDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\textsc{BoundDel}(\circ, \beta, d)$*;*
2. *if* $\textsc{BoundDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$*;*
3. *if* $\textsc{ExactDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\exists\textsc{ExactDel}(\circ, \beta)$*;*
4. *if* $\textsc{BoundDel}(\circ, \beta, d)$ *is diagnosable, then so is* $\exists\textsc{BoundDel}(\circ, \beta)$*;*
5. *if* $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ *is diagnosable, then so is* $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$*;*
6. *if* $\exists\textsc{ExactDel}(\circ, \beta)$ *is diagnosable, then so is* $\exists\textsc{BoundDel}(\circ, \beta)$*;*
7. *if* $\exists\textsc{BoundDel}(\circ, \beta)$ *is diagnosable, then so is* $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$*;*
8. *if* $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$ *is diagnosable, then so is* $\textsc{FiniteDel}(\circ, \beta)$*.*

Proof. We rely on Definitions 3 and 4, and show these properties case by case.

1. Pick any π_1 and any $i \in \mathbb{N}$ such that $\pi_1, i \models \beta$. Since $\textsc{ExactDel}(\circ, \beta, d)$ is diagnosable, for every trace π_2 such that $obs(\pi_1^i) = obs(\pi_2^i)$, we have that $\pi_2, i \models \beta$. Then $\textsc{BoundDel}(\circ, \beta, d)$ is also diagnosable, by taking $j = i + d$ and $k = i$ in Def. 3 for the $\textsc{BoundDel}$ case.

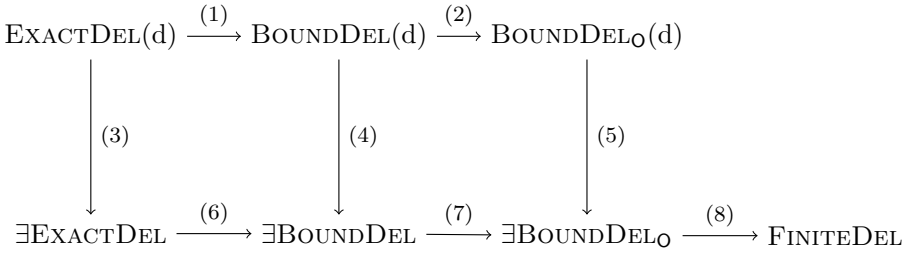

Figure 6: Diagnosability patterns dependencies (β is the same for all patterns; alarm signal and β are omitted for brevity). A directed path from pattern Ξ_1 to pattern Ξ_2 means that diagnosability of Ξ_1 implies diagnosability of Ξ_2. Edges are labeled with the corresponding cases of Theorem 4.

2. Similarly, we can take the j and k witnessing the diagnosability of $\textsc{BoundDel}(\circ, \beta, d)$ to prove the diagnosability of $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$.
3. Immediate by definitions.
4. Immediate by definitions.
5. Immediate by definitions.
6. We can take the delay d witnessing the diagnosability of $\exists\textsc{ExactDel}(\circ, \beta)$ to prove the diagnosability of $\exists\textsc{BoundDel}(\circ, \beta)$ and reduce to case 1.
7. Similarly to case 6.
8. Given a delay d such that $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ is diagnosable, we can take the j and k witnessing the diagnosability of $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ to prove the diagnosability of $\textsc{FiniteDel}(\circ, \beta)$.

□

Dependencies between patterns are pictorially represented in Figure 6. A directed path from pattern Ξ_1 to pattern Ξ_2 signifies that diagnosability of Ξ_1 implies diagnosability of Ξ_2. If there is no such path, then the implication does not hold, i.e. there are situations where Ξ_1 is diagnosable but Ξ_2 is not.

Finally, we prove the following result, that relates $\textsc{BoundDel}_{\mathsf{O}}$ and $\textsc{BoundDel}$ diagnosability. In particular, it shows that β is $\textsc{BoundDel}_{\mathsf{O}}$ diagnosable for a delay d if and only if $\mathsf{O}\beta$ is $\textsc{BoundDel}$ diagnosable for the same delay.

Theorem 5. *Let P be a plant, β a diagnosis condition, and $d \in \mathbb{N}$ a delay. We have that $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ is diagnosable iff $\textsc{BoundDel}(\circ, \mathsf{O}\beta, d)$ is diagnosable.*

Proof. The proof follows by Def. 3.
Let us assume that $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ is diagnosable. Let us take a trace π_1 and an index i such that $\pi_1, i \models \mathsf{O}\beta$. Then, there exists an index $i' \leq i$ such that $\pi_1, i' \models \beta$. Then, by hypothesis we have that there exists $i' \leq j' \leq i' + d$ such that for all traces π_2 that are observationally equivalent to π_1 up to j', there exists $k' \leq j'$ such that $\pi_2, k' \models \beta$. Let us take $j = max(j', i)$. Clearly, all traces

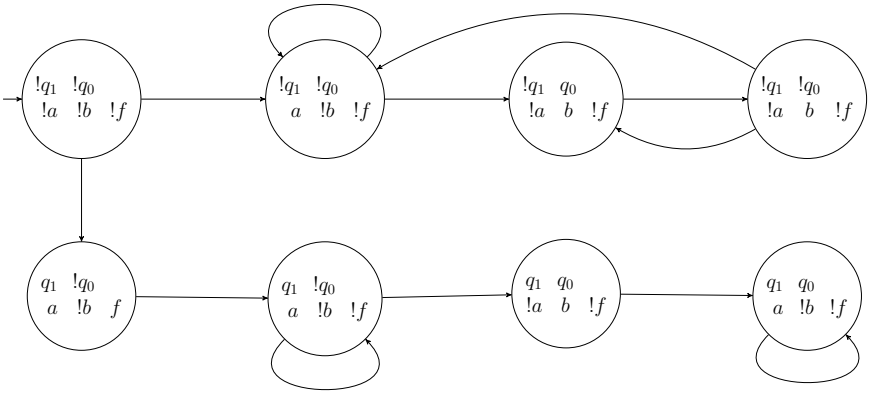

Figure 7: An example which is FINITEDEL diagnosable and not BOUNDDEL$_{\mathsf{O}}$ diagnosable.

π_2 that are observationally equivalent to π_1 up to j are also observationally equivalent to π_1 up to j'. Hence, there exists $k' \leq j' \leq j$ such that $\pi_2, k' \models \beta$. From this, it follows that $\pi_2, j \models \mathsf{O}\beta$. Moreover , we have that $i \leq j \leq i + d$. Hence, we can conclude that BOUNDDEL$(\circ, \mathsf{O}\beta, d)$ is diagnosable.
Vice versa, let us assume that BOUNDDEL$(\circ, \mathsf{O}\beta, d)$ is diagnosable. Let us take a trace π_1 and an index i such that $\pi_1, i \models \beta$. Clearly, we also have that $\pi_1, i \models \mathsf{O}\beta$. Then, by hypothesis we have that there exists $i \leq j \leq i + d$ such that for all traces π_2 that are observationally equivalent to π_1 up to j, there exists $j - d \leq k \leq j$ such that $\pi_2, k \models \mathsf{O}\beta$. Since $\pi_2, k \models \mathsf{O}\beta$ and $k \leq j$, there exists an index $h \leq k \leq j$ such that $\pi_2, h \models \beta$. Hence, we can conclude that BOUNDDEL$_{\mathsf{O}}(\circ, \beta, d)$ is diagnosable. □

3.5. Diagnosability Patterns Exemplified

In this section we show some paradigmatic examples that illustrate the dependencies between diagnosability patterns presented in Section 3.4. We notice the importance of fairness conditions, expressed in the form of contexts, in most of the following examples. We use fairness to disprove pattern dependencies that do not hold, according to Fig. 6.

FINITEDEL *Diagnosability Does Not Imply* $\exists$BOUNDDEL$_{\mathsf{O}}$ *Diagnosability.* Let us consider the example in Figure 7, adapted from [29]. Let $\{q_1, q_0, a, b, f\}$ be the set of state variables, and $\{a, b\}$ the set of observable ones. Let $\beta \doteq f$ and consider a context $\Psi \doteq (\mathsf{GF}(!q_1 \wedge q_0 \wedge b)) \vee (\mathsf{GF}(q_1 \wedge q_0 \wedge a))$. We have that β is not BOUNDDEL$_{\mathsf{O}}$ diagnosable for any d. In fact, given d, there exist a trace π_1 and a trace π_2 such that π_1 satisfies β at time 1 ($\pi_1[1] \models f$), and π_2 does not satisfy β at any time point, π_1 and π_2 satisfy Ψ, and π_1 and π_2 have prefixes of length $d+2$

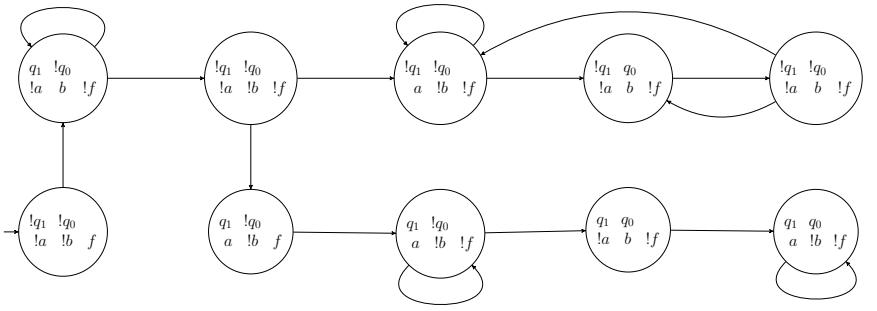

Figure 8: An example which is BoundDel$_{\mathsf{O}}$ diagnosable and not BoundDel diagnosable.

that are observationally equivalent, i.e.[4] $obs(\pi_1^{d+1}) = obs(\pi_2^{d+1}) = \{\}\{a\}^{d+1}$.

However, β is FiniteDel diagnosable. In fact, we can observe that any fault, due to the context, is followed eventually (i.e., after a finite number of steps) by a single occurrence of b followed by a (observations $\{b\}\{a\}$), whereas a trace without faults eventually contains two consecutive b (observations $\{b\}\{b\}$). Hence, a faulty trace can be distinguished from a nominal one. Formally, given a trace π_1 that satisfies β (necessarily at time 1), we can pick a $j = l + 1$ such that $\pi_1[l] \models b$ that satisfies the conditions of Definition 3 for FiniteDel.

$\exists$BoundDel$_{\mathsf{O}}$ *Diagnosability Does Not Imply* $\exists$BoundDel *Diagnosability.* Consider a slight modification of the example in Figure 7, shown in Figure 8, where we have added two states at the beginning of times, feeding the former initial state in Figure 7. The variables, observables, β and context Ψ are the same as for the example in Figure 7. We have that β is trivially BoundDel$_{\mathsf{O}}$ diagnosable, since any trace satisfies β at time 0, i.e., any trace π is such that $\pi[0] \models f$.

However, β is not BoundDel diagnosable. Intuitively, we can diagnose the first occurrence of the fault, but not the second one. To observe this, for any d we can consider a trace π_1 that satisfies Ψ, with an observable prefix $\{\}\{b\}^{d+1}\{\}\{a\}^{d+1}$, and such that $\pi_1[d{+}3] \models f$. Now, for each $d{+}3 \leq j \leq 2d{+}3$, we can construct a trace π_2 that satisfies Ψ, does not satisfy β in the interval $j - d \leq k \leq j$, and has the same observable prefix of length $2d + 4$ as π_1.

$\exists$BoundDel *Diagnosability Does Not Imply* $\exists$ExactDel *Diagnosability.* Let us consider the simple example in Figure 9. Let f be a non-observable state variable, $\beta \doteq f$ and consider a context *true*. The other state variables are also non-observable and are defined such that there are exactly three traces in the model. Note that all the traces of the model are observationally equivalent. We

[4]We remind the reader that a state is an assignment to state variables. Here, we represent an assignment as a set, i.e. the set of state variables that are assigned to true. $\{\}$ represents the assignment corresponding to the empty set $\emptyset$, whereas the superscript k indicates repetition

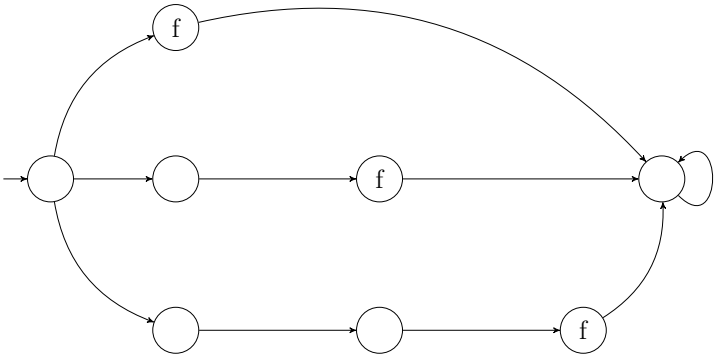

Figure 9: An example which is BoundDel diagnosable and not ExactDel diagnosable.

have that β is BoundDel diagnosable for delay $d = 2$. In fact, it is easy to verify that for any trace π_1 and time point i such that $\pi_1[i] \models f$, we can find a j such that $i \leq j \leq i+2$ such that for any other trace (observationally equivalent to π_1) there exists a k such that $j-2 \leq k \leq j$ such that $\pi_2 \models f$. Intuitively, we can diagnose the fault with a delay of 2 since, regardless of the execution trace, at time point 4 we are guaranteed that f occurred by at most two time steps. On the other hand, it is easy to verify that β is not ExactDel diagnosable for any delay, since we have no way to know the precise time step where f occurs.

An Example which is not FiniteDel *Diagnosable.* We conclude by showing a slight modification of the example in Figure 7 which is not even FiniteDel diagnosable. Consider the system in Figure 10, where we have removed the topmost right state, and reconnected the state to its left. The variables, observables, β and context Ψ are the same as for the example in Figure 7. We can show that β is not FiniteDel diagnosable. In fact, any faulty trace follows the observable regular expression $\{\}\{a\}^k\{b\}\{a\}^\omega$, whereas a nominal one follows $\{\}\{a\}^k(\{b\}\{a\}^l)^\omega$. Intuitively, a nominal trace has still an infinite number of occurrences of b, but we do not force consecutive b, hence there is an arbitrary number of steps before another b occurs. More formally, we can show that there exists a faulty trace π_1 satisfying the context, and an index i such that π_1 satisfies β at i, such that for all $j \geq i$, there exists a nominal trace π_2 satisfying the context that is observationally equivalent to π_1.

4. Critical Pairs

Historically, diagnosability checking is tackled by looking for refutation conditions: diagnosability is concluded when failing to find a counterexample witnessing the violation. A counterexample is in form of a *critical pair*, i.e. a pair

for k times.

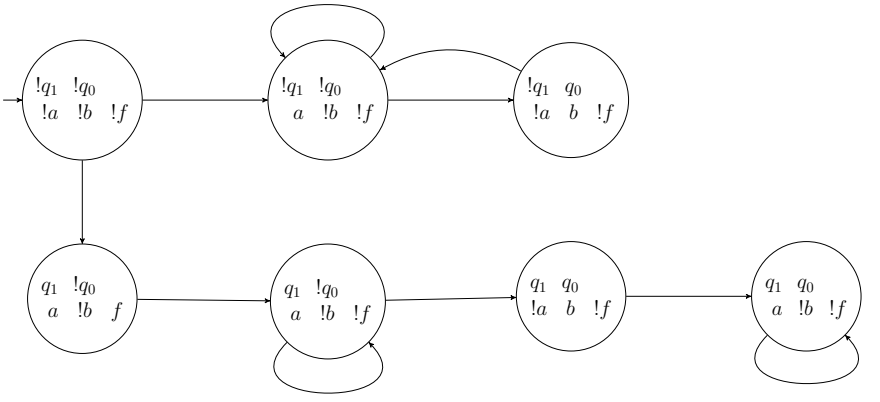

Figure 10: An example which is not FINITEDEL diagnosable.

of traces π_1, π_2 that are indistinguishable from the standpoint of the observer, but differ in the way they exhibit the property to be diagnosed [4, 5].

In this section, we analyze the nature of counterexamples to diagnosability for our framework, extending the classical definition of critical pair to consider fairness and the patterns introduced in Section 3. We show that the existence of a critical pair is a sufficient, but in general not necessary condition for diagnosability violation.

Definition 5 (Critical Pair (CP))**.** *Let P be a plant, Ξ an alarm condition and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a critical pair for an alarm condition Ξ at time i iff $\pi_1, i \models \beta$ and:*

- *if $\Xi = \text{EXACTDEL}(\circ, \beta, d)$, then $\text{OBSEQUPTO}(\pi_1, \pi_2, i+d)$ and $\pi_2, i \not\models \beta$;*
- *if $\Xi = \text{BOUNDDEL}(\circ, \beta, d)$, then $\text{OBSEQUPTO}(\pi_1, \pi_2, i+d)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \in [i-d, i+d]$;*
- *if $\Xi = \text{BOUNDDEL}_\mathsf{O}(\circ, \beta, d)$, then $\text{OBSEQUPTO}(\pi_1, \pi_2, i+d)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \leq i+d$;*
- *if $\Xi = \text{FINITEDEL}(\circ, \beta)$, then $obs(\pi_1) = obs(\pi_2)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$.*

We say that $\pi_1, \pi_2 \in \Pi_P$ are a critical pair for Ξ if it is one for some time point $i \in \mathbb{N}$.

In the next two sections we will show that the existence of a critical pair is a sufficient condition for non-diagnosability, while arguing that it is not always a necessary condition, except in the EXACTDEL and $\text{BOUNDDEL}_\mathsf{O}$ cases.

4.1. Critical Pairs are Sufficient for Diagnosability Violation

We first prove a fundamental property of the critical pair approach. Namely, we establish that when a critical pair is found, the respective alarm condition must indeed not be diagnosable for the given system. Dually, the absence of critical pairs is a necessary condition for diagnosability.

Theorem 6 (Critical Pairs are Sufficient for Diagnosability Violation)**.** *Let P be a plant and Ξ an alarm condition. If there exist π_1, π_2 that are a critical pair for Ξ as per Definition 5, then Ξ is not diagnosable.*

Proof. Assume there exist traces π_1, π_2 such that π_1, π_2 are a critical pair for Ξ, but Ξ is diagnosable. We show for each type of pattern that this leads to a contradiction.

- If $\Xi = \textsc{ExactDel}(\circ, \beta, d)$, from Definition 5 we know that there exists $i \geq 0$ such that $\pi_1, i \models \beta$, $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$, and $\pi_2, i \not\models \beta$. Such a pair of traces is however not possible if Ξ is diagnosable, since, according to Definition 3, $\pi_1, i \models \beta$ and $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$ imply that $\pi_2, i \models \beta$, which is not true on the critical pair.

- If $\Xi = \textsc{BoundDel}(\circ, \beta, d)$, from Definition 5 we know that there exists $i \geq 0$ such that $\pi_1, i \models \beta$, $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i+d)$, and $\forall j \in [i-d, i+d] \cdot \pi_2, j \not\models \beta$. Such a pair of traces is however not possible if Ξ is diagnosable, since, according to Definition 3, $\pi_1, i \models \beta$ implies that $\exists j \in [i, i + d]$ such that either $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ is false, which is not the case for any j (by the fact that $j \leq i + d$ and $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$), or $\exists k \in [j - d, j] \cdot \pi_2, k \models \beta$, which is also not true, as k is by definition within $[i - d, i + d]$, and we know that for none of these time points β holds on π_2.

- If $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$, from Definition 5 we know that there exists $i \geq 0$ such that $\pi_1, i \models \beta$, $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$, and $\forall j \leq i + d \cdot \pi_2, j \not\models \beta$. Such a pair of traces is however not possible if Ξ is diagnosable, since, according to Theor. 1 $\pi_1, i \models \beta$ implies that $\exists k \leq i + d \cdot \pi_2, k \models \beta$, which is not true on the critical pair.

- If $\Xi = \textsc{FiniteDel}(\circ, \beta)$, from Definition 5 we know that there exists $i \geq 0$ such that $\pi_1, i \models \beta$, $obs(\pi_1) = obs(\pi_2)$, and $\forall j \in \mathbb{N} \cdot \pi_2, j \not\models \beta$. Such a pair of traces is however not possible if Ξ is diagnosable, since, according to Definition 3, $\pi_1, i \models \beta$ implies that $\exists j \in \mathbb{N} \cdot i \leq j$ such that either $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ is false, which is not the case for any j (since $obs(\pi_1) = obs(\pi_2)$), or $\exists k \in [0, j] \cdot \pi_2, k \models \beta$, which is also not true for whatever j we choose.

□

4.2. Critical Pairs are not Necessary for Diagnosability Violation

We now show that it is in general not possible to check diagnosability by looking for critical pairs. In particular, we prove that the absence of critical pairs is sufficient for diagnosability in case of ExactDel and BoundDel$_{\mathsf{O}}$, while it is not for BoundDel and FiniteDel.

Critical Pairs for ExactDel *and* BoundDel$_{\mathsf{O}}$

For ExactDel and BoundDel$_{\mathsf{O}}$ the absence of critical pairs is a sufficient condition for diagnosability.

Theorem 7 (Critical Pairs are necessary for ExactDel and BoundDel$_{\mathsf{O}}$ diagnosability violation)**.** *Let P be a plant and $\Xi = \textsc{ExactDel}(\circ, \beta, d)$ or $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ an alarm condition. If Ξ is not diagnosable, then there exists a critical pair for Ξ as per Definition 5.*

Proof. The proof follows directly from the definitions of diagnosability and critical pair.

- If $\Xi = \textsc{ExactDel}(\circ, \beta, d)$, assume that Ξ is not diagnosable. Then, by Definition 3, there exists a trace π_1 and an index $i \in \mathbb{N}$ s.t. $\pi_1, i \models \beta$ and there exists a trace π_2 s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$ and $\pi_2, i \not\models \beta$. Then, by Def. 5, π_1, π_2 is a critical pair for Ξ at time i.
- If $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$, assume that Ξ is not diagnosable. Then, by Theor. 1, there exists a trace π_1 and an index $i \in \mathbb{N}$ such that $\pi_1, i \models \beta$ and $\exists \pi_2$, s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$ and $\forall k \in \mathbb{N}. k \leq j \Rightarrow \pi_2, k \not\models \beta$. Then, by Definition 5, π_1, π_2 are a critical pair for Ξ at time i.

□

Critical Pairs for BoundDel

In general, in case of BoundDel, the absence of a critical pair does not imply that the system is diagnosable. To illustrate this case, we consider an example of a transmitter, shown in Figure 11. The variables *in* and *out* are Booleans representing the presence of respectively the input and the output of the transmitter. The output is generated if and only if the input is present in the current and in the last two states. This is obtained with two history variables, h_1 and h_2, which store the value of i in respectively the last and the second last state. Let us assume that we can observe only the output *out* and that we want to detect when *in* is false ($\beta \doteq !in$), assuming that the *in* is true in the first two states ($\Psi \doteq in \wedge Xin$). Note how the context is used to ignore non-interesting non-diagnosable cases.

Consider the three traces in Figure 12. These traces have the same evaluation of *out* and thus are not distinguishable. The trace π satisfies β in k, the trace π_1 does not satisfy β in positions $k-1$ and k, while the trace π_2 does not satisfy β in positions k and $k+1$. Therefore, according to Def. 3 we can conclude that $\textsc{BoundDel}(\circ, \beta, 1)$ is not diagnosable at k. However, it is not possible to

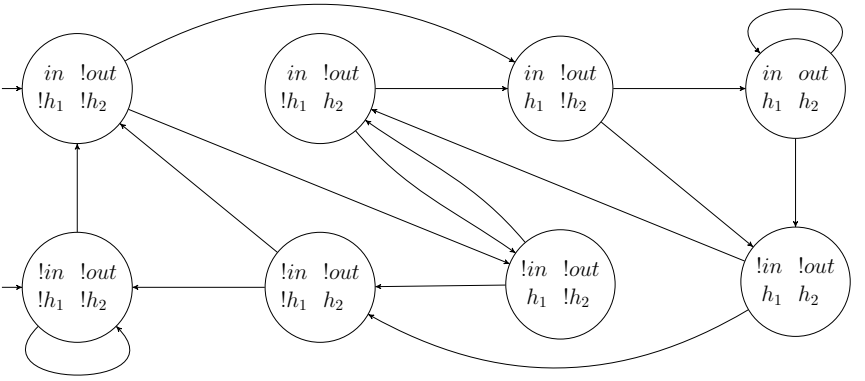

Figure 11: Transmitter example.

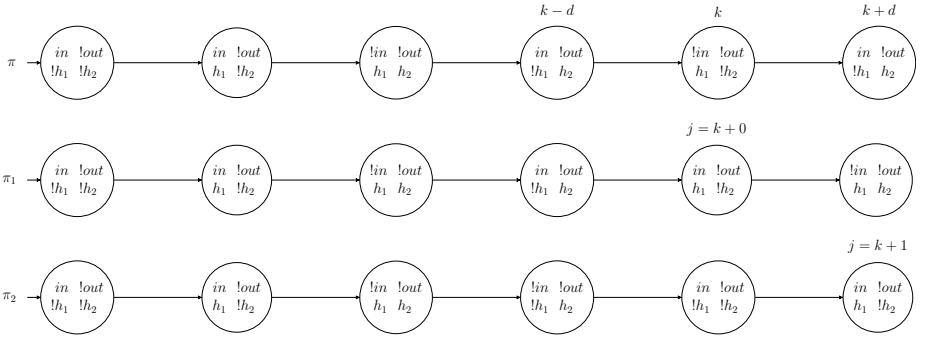

Figure 12: Example showing that BoundDel$(\circ, \beta, 1)$ is not diagnosable at k.

construct a BoundDel critical pair w.r.t. $\pi[k]$ for delay $d = 1$. (Note that π and π_1 are a critical pair for BoundDel$(\circ, \beta, 0)$, instead.)

In general, in this example, no pair of traces constitutes a critical pair for BoundDel$(\circ, \beta, 1)$. It is important to note here that the problem is not in the specific definition of BoundDel critical pairs, but in the fact that falsification is done only considering *pairs* of traces.

Critical Pairs for FiniteDel

Also in the case of FiniteDel the absence of critical pairs does not necessarily prove diagnosability. To illustrate this, we consider the state machine of a light bulb as shown in Figure 13. Consider the observable value OFF/ON, the diagnosis condition $\beta \doteq KO$ and the context $\mathsf{G}(KO \rightarrow \mathsf{F}\ OFF) \wedge \mathsf{G}(OK \rightarrow \mathsf{F}\ ON)$. If the execution reaches KO, it will eventually go into state OFF/KO and remain there forever. If an execution is instead always OK, then it will visit infinitely often the state ON/OK. Therefore, it is impossible to find a critical pair, namely a pair of traces that are forever observationally equivalent,

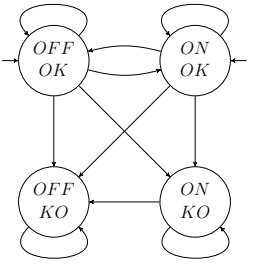

Figure 13: Light bulb example.

but where one satisfies β and the other one does not.

However, for every j, there exists a trace without β that is observationally equivalent up to j to the trace with β. Therefore, β is not FiniteDel diagnosable, according to Def. 3. Notice how we use the context to encode a fairness constraint that causes the observations after a failure to always diverge *eventually*, but that this event can be delayed indefinitely.

4.3. Properties of Critical Pairs

In this section we prove some properties related to the existence of critical pairs, their monotonicity and dependencies.

First, we prove that critical pairs are monotonic w.r.t. the diagnosability delay. Namely we show that critical pairs for ExactDel, BoundDel and BoundDel$_{\mathsf{O}}$ w.r.t. a delay d' are also witnesses for non-diagnosability with a delay $d < d'$.

Theorem 8 (Monotonicity of Critical Pairs)**.** *Let P be a plant, β a diagnosis condition and $d, d' \in \mathbb{N}$ with $d' > d$.*

1. *If π_1, π_2 are a critical pair for* ExactDel$(\circ, \beta, d')$*, then they are also a critical pair for* ExactDel$(\circ, \beta, d)$*.*
2. *If π_1, π_2 are a critical pair for* BoundDel$(\circ, \beta, d')$*, then they are also a critical pair for* BoundDel$(\circ, \beta, d)$*.*
3. *If π_1, π_2 are a critical pair for* BoundDel$_{\mathsf{O}}(\circ, \beta, d')$*, then they are also a critical pair for* BoundDel$_{\mathsf{O}}(\circ, \beta, d)$*.*

Proof. The proof follows directly from Definition 5. For instance, assume that π_1, π_2 are a critical pair at time i for BoundDel$(\circ, \beta, d')$. Then, by definition 5, we have that ObsEqUpTo$(\pi_1, \pi_2, i + d')$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \in [i-d', i+d']$. In particular, since $d < d'$, we have that ObsEqUpTo$(\pi_1, \pi_2, i+d)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \in [i - d, i + d]$. Then, by Definition 5, π_1, π_2 are a critical pair for BoundDel$(\circ, \beta, d)$. The other cases are similar. □

The theorem below relates the existence of critical pairs for different alarm conditions patterns. Namely it shows that critical pairs for BoundDel are also critical pairs for ExactDel, critical pairs for BoundDel$_{\mathsf{O}}$ are also critical

pairs for BoundDel, and critical pairs for FiniteDel are also critical pairs for BoundDel$_{\mathsf{O}}$.

Theorem 9 (Dependencies between Critical Pairs)**.** *Let P be a plant, β a diagnosis condition and $d \in \mathbb{N}$.*

1. *If π_1, π_2 are a critical pair for* BoundDel$(\circ, \beta, d)$*, then they are also a critical pair for* ExactDel$(\circ, \beta, d)$*.*
2. *If π_1, π_2 are a critical pair for* BoundDel$_{\mathsf{O}}(\circ, \beta, d)$*, then they are also a critical pair for* BoundDel$(\circ, \beta, d)$*.*
3. *If π_1, π_2 are a critical pair for* FiniteDel$(\circ, \beta)$*, then they are also a critical pair for* BoundDel$_{\mathsf{O}}(\circ, \beta, d)$ *for any d.*

Proof. The proof follows directly from Definition 5. For instance, assume that π_1, π_2 are a critical pair at time i for BoundDel$(\circ, \beta, d)$. Then, by definition 5, we have that ObsEqUpTo$(\pi_1, \pi_2, i + d)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \in [i - d, i + d]$. In particular, we have that $\pi_2, i \not\models \beta$. Then, by Definition 5, π_1, π_2 are a critical pair for ExactDel$(\circ, \beta, d)$. The other cases are similar. □

The following theorems state conditions where the absence of a critical pair is sufficient to infer diagnosability. We first prove that the absence of BoundDel critical pairs for a delay d is a sufficient condition for diagnosability with a delay of $2d$.

Theorem 10 (No Critical Pairs sufficient for Double-depth BoundDel diagnosability)**.** *Let P be a plant, and Ξ an alarm condition such that $\Xi =$* BoundDel$(\circ, \beta, 2d)$*. If no critical pair π_1, π_2 for $\Xi' =$* BoundDel$(\circ, \beta, d)$ *exists as per Definition 5, then Ξ is diagnosable.*

Proof. Assume no critical pair for Ξ' exists, but Ξ is not diagnosable. The fact that Ξ is not diagnosable implies that $\exists \pi_1$, $\exists i \in \mathbb{N}$ such that $\pi_1, i \models \beta$ and, furthermore, $\forall j \in [i, i + 2d] \cdot \exists \pi_2$ such that ObsEqUpTo(π_1, π_2, j) and $\forall k \in [j - 2d, j] \cdot \pi_2, k \not\models \beta$. In particular, for π_1 and i we pick $j = i + d$, for which we know exists a π_2 such that $\pi_1, i \models \beta$, ObsEqUpTo$(\pi_1, \pi_2, i + d)$, and $\forall k \in [i - d, i + d] \cdot \pi_2, k \not\models \beta$. π_1, π_2 are thus a critical pair for Ξ', which contradicts the initial assumption that no such pair exists.

□

We notice that this bound is tight: without making further assumptions, a lower bound cannot be guaranteed from the absence of critical pairs, even a $d' = 2d - 1$. An example of this is the system of Figure 12.

As a consequence of Theorem 10 we also have the following result.

Theorem 11 (Critical Pairs for BoundDel are necessary for ∃BoundDel diagnosability violation)**.** *Let P be a plant, and Ξ an alarm condition such that $\Xi =$* ∃BoundDel$(\circ, \beta)$*. If Ξ is not diagnosable, then for all d there exists a critical pair for $\Xi' =$* BoundDel$(\circ, \beta, d)$ *as per Definition 5.*

Proof. If $\Xi = \exists\textsc{BoundDel}(\circ, \beta)$ is not diagnosable then $\textsc{BoundDel}(\circ, \beta, d)$ is not diagnosable for all d, in particular it is not diagnosable for $2d$. Then, due to Theorem 10, there exists a critical pair for $\Xi' = \textsc{BoundDel}(\circ, \beta, d)$. □

Now, we show some results for diagnosability where β is persistent. Namely, we consider the case where β is persistent in P with respect to Ψ, i.e. for all i, for all trace π, if β holds at time i, then it also holds at any time $j \geq i$. We remark that the monotonicity/persistency assumption is quite common in the existing literature. In particular, in [3] (and all successive works based on it) the diagnosis condition is defined as the occurrence of a fault event at some point in the past, i.e. $\beta \doteq \mathsf{O}f$ for some proposition f, which has a monotonic behavior. As proven in [8], this can be reduced to a BoundDel problem. More in general, monotonicity is not limited to the past occurrence of fault events. Indeed, all diagnosis conditions of the form $\beta \doteq \mathsf{O}\beta'$, such as the ones described in [28], are monotonic. From a practical point of view, a permanent fault is an example of persistent condition. There are however situations where β is a persistent condition without being a permanent fault. Consider for example that β may be counting whether the count of transient faults has exceeded a given threshold.

When β is persistent, we can prove that BoundDel diagnosability and $\textsc{BoundDel}_{\mathsf{O}}$ diagnosability are equivalent, hence it is possible to search indifferently for a critical pair for either of the two patterns. This is formalized in the following theorem.

Theorem 12 (Persistent β: BoundDel diagnosability and $\textsc{BoundDel}_{\mathsf{O}}$ diagnosability are equivalent)**.** *Let P be a plant, Ψ a context, $d \in \mathbb{N}$ a delay, and let β be persistent in P and Ψ, i.e. for every trace π we have $\pi \models \mathsf{G}(\mathsf{O}\beta \rightarrow \beta)$. Then $\textsc{BoundDel}(\circ, \beta, d)$ is diagnosable iff $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ is diagnosable.*

Proof. If β is persistent, then β holds iff $\mathsf{O}\beta$ does. The conclusion follows from Theorem 5. □

As a corollary, if β is persistent, the absence of critical pairs for $\textsc{BoundDel}(\circ, \beta, d)$ implies diagnosability.

Theorem 13 (Persistent β: Critical Pairs are necessary for BoundDel diagnosability violation)**.** *Let P be a plant, Ψ a context, $d \in \mathbb{N}$ a delay, and Ξ an alarm condition such that $\Xi = \textsc{BoundDel}(\circ, \beta, d)$. Let β be persistent in P and Ψ, i.e. for every trace π we have $\pi \models \mathsf{G}(\mathsf{O}\beta \rightarrow \beta)$. If Ξ is not diagnosable, then a critical pair for Ξ as per Definition 5 exists.*

4.4. Critical Pairs for Existential Patterns

We conclude this section by extending the definition of critical pair to the case of existential patterns. We give the following definition.

Definition 6 (Critical Pairs for Existential Patterns)**.** *Let P be a plant, Ξ an alarm condition and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a critical pair for an alarm condition Ξ at time i iff $\pi_1, i \models \beta$ and:*

- *if* $\Xi = \exists\textsc{ExactDel}(\circ, \beta)$*, then* $obs(\pi_1) = obs(\pi_2)$ *and* $\pi_2, i \not\models \beta$*;*
- *if* $\Xi = \exists\textsc{BoundDel}(\circ, \beta)$*, then* $obs(\pi_1) = obs(\pi_2)$*, and* $\pi_2, j \not\models \beta$ *for all* $j \in \mathbb{N}$*;*
- *if* $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$*, then* $obs(\pi_1) = obs(\pi_2)$*, and* $\pi_2, j \not\models \beta$ *for all* $j \in \mathbb{N}$*;*

We say that $\pi_1, \pi_2 \in \Pi_P$ *are a critical pair for* Ξ *if it is one for some time point* $i \in \mathbb{N}$*.*

Notice that the definition of critical pairs for the $\exists\textsc{BoundDel}$ and $\exists\textsc{BoundDel}_{\mathsf{O}}$ cases collapse to the one for $\textsc{FiniteDel}$ (compare Def. 5).

We can prove an analogous result of Theorem 6, namely we have that critical pairs, in the case of existential patterns, are sufficient (but not necessary) for diagnosability violation.

Theorem 14 (Critical Pairs for existential patterns are sufficient for diagnosability violation)**.** *Let* P *be a plant and* Ξ *an alarm condition such that* $\Xi = \exists\textsc{ExactDel}(\circ, \beta)$ *or* $\Xi = \exists\textsc{BoundDel}(\circ, \beta)$ *or* $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$*. If there exist* π_1, π_2 *that are a critical pair for* Ξ *as per Definition 6, then* Ξ *is not diagnosable.*

Proof. It is easy to see that from a critical pair for $\exists\textsc{ExactDel}(\circ, \beta)$ [respectively, $\exists\textsc{BoundDel}(\circ, \beta)$, $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$] at time i, for all d we can construct a critical pair for $\textsc{ExactDel}(\circ, \beta, d)$ [respectively, $\textsc{BoundDel}(\circ, \beta, d)$, $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$]. The conclusion follows from Def. 4 and Theorem 6. □

5. From Critical Pairs to Critical Sets

In the previous section we have introduced critical pairs, and we have shown that they are not always adequate to represent a violation of diagnosability. In this section, we look for necessary and sufficient conditions for diagnosability. To this aim, we introduce the notion of *critical set*. We show that the existence of a critical set is a necessary and sufficient condition for diagnosability violation.

Definition 7 (Critical Set (CS))**.** *Let* P *be a plant,* Ξ *an alarm condition and* β *a diagnosis condition. We say that a set of traces* $\mathcal{CS} \subseteq \Pi_P$ *is a* critical set *for an alarm condition* Ξ *at time* i *iff there exists a trace* $\pi_1 \in \mathcal{CS}$ *s.t.* $\pi_1, i \models \beta$ *and* $\mathcal{CS}$ *is s.t.:*

- *if* $\Xi = \textsc{ExactDel}(\circ, \beta, d)$*, then* $\exists \pi_2 \in \mathcal{CS}$*, s.t.* $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i+d)$ *and* $\pi_2, i \not\models \beta$*;*
- *if* $\Xi = \textsc{BoundDel}(\circ, \beta, d)$*, then* $\forall j. i \leq j \leq i + d \quad \exists \pi_2 \in \mathcal{CS}$*, s.t.* $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ *and* $\forall k \in \mathbb{N}. j - d \leq k \leq j \Rightarrow \pi_2, k \not\models \beta$*;*
- *if* $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$*, then* $\exists \pi_2 \in \mathcal{CS}$*, s.t.* $\textsc{ObsEqUpTo}(\pi_1, \pi_2, i + d)$ *and* $\forall k \in \mathbb{N}. k \leq i + d \Rightarrow \pi_2, k \not\models \beta$*;*

- *if* $\Xi = \textsc{FiniteDel}(\circ, \beta)$, *then* $\forall j.i \leq j \quad \exists \pi_2 \in \mathcal{CS}$, *s.t.* $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ *and* $\forall k \in \mathbb{N}.k \leq j \Rightarrow \pi_2, k \not\models \beta$.

We say that $\mathcal{CS}$ *is a critical set for* Ξ *if there exists a time point* $i \in \mathbb{N}$ *s.t.* $\mathcal{CS}$ *is a critical set for* Ξ *at time* i.

Critical sets are necessary and sufficient for diagnosability violation, as we prove in the next section.

5.1. Critical Sets are Necessary and Sufficient for Diagnosability Violation

We prove a fundamental property of critical sets, namely that the existence of a critical set is a necessary and sufficient condition for diagnosability violation.

Theorem 15 (Critical sets are necessary and sufficient for diagnosability violation)**.** *Let* P *be a plant and* Ξ *an alarm condition.* Ξ *is diagnosable if and only if there exists no critical set for* Ξ *as per Definition 7.*

Proof. The proof follows directly from the definitions of diagnosability and critical set. The $\textsc{ExactDel}$ and $\textsc{BoundDel}_{\mathsf{O}}$ cases are trivial, since the existence of a critical set in these cases is equivalent to the existence of a critical pair. We focus on the interesting cases.

- if $\Xi = \textsc{BoundDel}(\circ, \beta, d)$, assume there exists a critical set $\mathcal{CS}$ for Ξ. By Definition 7, there exists a trace $\pi_1 \in \mathcal{CS}$ and an index $i \in \mathbb{N}$ $\pi_1, i \models \beta$, and $\forall j.i \leq j \leq i+d \;\; \exists \pi_2 \in \mathcal{CS}$ s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ and $\forall k \in \mathbb{N}.j - d \leq k \leq j \Rightarrow \pi_2, k \not\models \beta$. Then, by Definition 3, Ξ is not diagnosable.
  Vice versa, assume that Ξ is not diagnosable. Then, by Definition 3, there exists a trace π_1 and an index $i \in \mathbb{N}$ s.t. $\pi_1, i \models \beta$ and $\forall j.i \leq j \leq i + d$ $\exists \pi_{2,j}$, s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_{2,j}, j)$ and $\forall k \in \mathbb{N}.j - d \leq k \leq j \Rightarrow \pi_{2,j}, k \not\models \beta$. Then, by Definition 7, $\mathcal{CS} = \{\pi_1\} \cup \{\pi_{2,j} \mid i \leq j \leq i + d\}$ is a critical set for Ξ at time i.

- if $\Xi = \textsc{FiniteDel}(\circ, \beta)$, assume there exists a critical set $\mathcal{CS}$ for Ξ. By Definition 7, there exists a trace $\pi_1 \in \mathcal{CS}$ and an index $i \in \mathbb{N}$ such that $\pi_1, i \models \beta$ and $\forall j.i \leq j \quad \exists \pi_2 \in \mathcal{CS}$, s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, j)$ and $\forall k \in \mathbb{N}.k \leq j \Rightarrow \pi_2, k \not\models \beta$. Then, by Definition 3, Ξ is not diagnosable.
  Vice versa, assume that Ξ is not diagnosable. Then, by Definition 3, there exists a trace π_1 and an index $i \in \mathbb{N}$ such that $\pi_1, i \models \beta$ and $\forall j.i \leq j$ $\exists \pi_{2,j}$, s.t. $\textsc{ObsEqUpTo}(\pi_1, \pi_{2,j}, j)$ and $\forall k \in \mathbb{N}.k \leq j \Rightarrow \pi_{2,j}, k \not\models \beta$. Then, by Definition 7, $\mathcal{CS} = \{\pi_1\} \cup \{\pi_{2,j} \mid i \leq j\}$ is a critical set for Ξ at time i.

□

We have the following result about the cardinality of critical sets.

Theorem 16 (Cardinality of critical sets)**.** *Let* P *be a plant and* Ξ *an alarm condition which is not diagnosable. If* $\Xi = \textsc{ExactDel}(\circ, \beta, d)$ *or* $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$*, then there exists a critical set for* Ξ *with cardinality 2. If* $\Xi = \textsc{BoundDel}(\circ, \beta, d)$ *then there exists a critical set for* Ξ *with cardinality* $d + 2$.

Proof. Since Ξ is not diagnosable, Theorem 15 implies that there exists a critical set for it. The case for the ExactDel$(\circ, \beta, d)$ and BoundDel$_{\mathsf{O}}(\circ, \beta, d)$ patterns is trivial, because the definition of $\mathcal{CS}$ requires only the existence of traces π_1 and π_2. For the remaining case, the proof for Theorem 15 shows that, if $\Xi =$ BoundDel$(\circ, \beta, d)$, then it is possible to construct a critical set $\{\pi_1\} \cup \{\pi_j \mid i \leq j \leq i + d\}$ for Ξ, with cardinality $d + 2$. □

Notice that a critical set of cardinality 2 for the ExactDel and BoundDel$_{\mathsf{O}}$ patterns can be seen equivalently as a critical pair (as per Def. 5). We conclude this section by showing examples of critical sets for the most interesting cases, namely BoundDel and FiniteDel.

Critical Sets for BoundDel

To illustrate critical sets for the BoundDel case, we revisit the example of the transmitter illustrated in Section 4.2 (compare Figure 11). Figure 12 shows an example of a critical set with three traces. It is easy to see that the traces satisfy the conditions of Def. 7.

Note that BoundDel$(\circ, \beta, 2)$ is instead diagnosable. This example can be easily extended to obtain a critical set of $n + 1$ traces by simply using n history variables and forcing o to be true if and only if i is true in the current and past n states.

Critical Sets for FiniteDel

To illustrate critical sets for the FiniteDel case, we revisit the example of the light bulb illustrated in Section 4.2 (compare Figure 13). We have noticed that for every j, there exists a trace without β that is observationally equivalent up to j to the trace with β. Hence we can construct a critical set as per Def. 7. Notice that here a critical set contains an infinite number of traces, one for each j.

5.2. Critical Sets for Existential Patterns

Based on the definition of critical pair given in Def. 5, in this section we generalize the notion of critical set to existential patterns. We give the definition below.

Definition 8 (Critical Set for existential patterns)**.** *Let P be a plant, β a diagnosis condition, and Ξ an alarm condition such that $\Xi =$ $\exists$ExactDel$(\circ, \beta)$ [respectively, $\Xi = \exists$BoundDel$(\circ, \beta)$, $\Xi = \exists$BoundDel$_{\mathsf{O}}(\circ, \beta)$]. We say that a set of traces $\mathcal{CS} \subseteq \Pi_P$ is a* critical set *for an alarm condition Ξ at time i iff for each $d \in \mathbb{N}$ there exist a trace $\pi_{1d} \in \mathcal{CS}$ and a trace $\pi_{2d} \in \mathcal{CS}$ s.t. π_{1d}, π_{2d} are a critical pair for* ExactDel$(\circ, \beta, d)$ *[respectively,* BoundDel$(\circ, \beta, d)$, BoundDel$_{\mathsf{O}}(\circ, \beta, d)$*] at time i. We say that $\mathcal{CS}$ is a critical set for Ξ if there exists a time point $i \in \mathbb{N}$ s.t. $\mathcal{CS}$ is a critical set for Ξ at time i.*

We have the following result, extending the fundamental property of critical sets given in Theorem. 15.

Theorem 17 (Critical Sets for existential patterns are necessary and sufficient for diagnosability violation)**.** *Let P be a plant and Ξ an alarm condition such that $\Xi = \exists\textsc{ExactDel}(\circ, \beta)$ or $\Xi = \exists\textsc{BoundDel}(\circ, \beta)$, or $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$. Ξ is diagnosable iff there exists no critical set for Ξ as per Definition 8.*

Proof. Notice that the existence of a critical set for $\exists\textsc{ExactDel}(\circ, \beta)$ [respectively, $\exists\textsc{BoundDel}(\circ, \beta)$, $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$] is equivalent to the existence, for all $d \in \mathbb{N}$, of a critical pair for $\textsc{ExactDel}(\circ, \beta, d)$ [respectively, $\textsc{BoundDel}(\circ, \beta, d)$, $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$]. Therefore, for the $\exists\textsc{ExactDel}$ and $\exists\textsc{BoundDel}_{\mathsf{O}}$ cases, the proof follows directly from Def. 4, Def. 8 and Theorem 7, since critical pairs are necessary and sufficient for diagnosability violation of $\textsc{ExactDel}$ and $\textsc{BoundDel}_{\mathsf{O}}$. For the $\exists\textsc{BoundDel}$ case, the proof follows from Def. 4, Def. 8, Theorem 6 and Theorem. 11 (critical pairs are sufficient for diagnosability violation of $\textsc{BoundDel}$; moreover, critical pairs are necessary for diagnosability violation of $\exists\textsc{BoundDel}$). □

6. From Critical Sets to Ribbon-shape Critical Pairs

The definition of critical sets for existential patterns does not lend itself to a decision procedure, since checking diagnosability relies on the existence of an infinite set of critical pairs; similarly for the $\textsc{FiniteDel}$ pattern, since in this case diagnosability relies on a critical set with infinite cardinality. In this section we discuss necessary and sufficient conditions for these patterns that are constructive, i.e. that can be decided algorithmically. Specifically, we identify finite representations for such critical sets.

6.1. Ribbon-Shape Critical Pairs

In order to devise constructive conditions, we build upon the notion of so-called *ribbon-shape critical pairs* introduced in [29]. First, we discuss diagnosability in the case of the $\exists\textsc{ExactDel}$, $\exists\textsc{BoundDel}_{\mathsf{O}}$ and $\textsc{FiniteDel}$ cases (the $\exists\textsc{ExactDel}$ case extends the definition for $\exists\textsc{BoundDel}_{\mathsf{O}}$ given in [29]). Figure 14 pictorially illustrates the concept for the $\textsc{BoundDel}_{\mathsf{O}}$ case. The formal definition is as follows.

Definition 9 (Ribbon-Shape Critical Pairs (RCP))**.** *Let P be a plant and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a ribbon-shape critical pair for an alarm condition Ξ iff there are indices k, l such that $k < l$ and:*

- *If $\Xi = \exists\textsc{ExactDel}(\circ, \beta)$, then*
  1. *$\pi_1[k] = \pi_1[l]$ and $\pi_2[k] = \pi_2[l]$;*
  2. *$\textsc{ObsEqUpTo}(\pi_1, \pi_2, l)$;*
  3. *$\pi_1, i \models \beta$ and $\pi_2, i \not\models \beta$ for some i, $0 \leq i \leq l$.*
- *If $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$, then*
  1. *$\pi_1[k] = \pi_1[l]$ and $\pi_2[k] = \pi_2[l]$;*

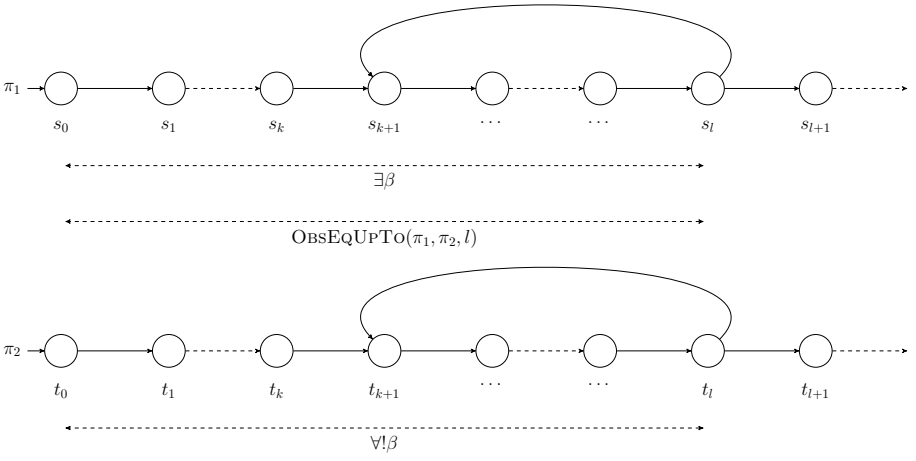

Figure 14: Ribbon-shape critical pair for BoundDel$_{\mathsf{O}}$.

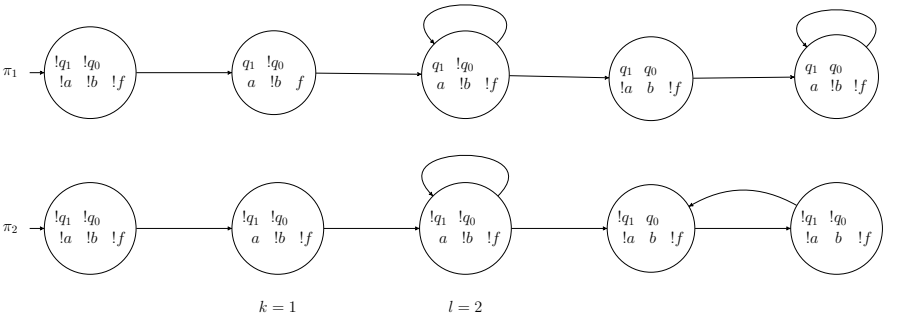

Figure 15: A BoundDel$_{\mathsf{O}}$ ribbon-shape critical pair for the example in Fig 7.

2. ObsEqUpTo(π_1, π_2, l);
3. $\pi_1, i \models \beta$ *for some* i, $0 \le i \le l$, *and* $\pi_2, i \not\models \beta$ *for all* i, $0 \le i \le l$.

- *If* $\Xi =$ FiniteDel$(\circ, \beta)$, *then*

1. $\pi_2[k] = \pi_2[l]$;
2. $obs(\pi_1) = obs(\pi_2')$;
3. $\pi_1, i \models \beta$ *for some* $i \ge 0$, *and* $\pi_2', i \not\models \beta$ *for all* $i \ge 0$;

where $\pi_2' = \pi_2[0, k] \cdot \pi_2[k+1, l]^\omega$.

Fig. 15 shows a ribbon-shape critical pair for the example in Fig. 7. The loop shown in Fig. 14 corresponds to the self loop at index $l = 2$ in Fig. 15. Notice that the two traces π_1 and π_2 are observationally equivalent up to $l = 2$. Moreover, π_1 contains $\beta = f$ at index 1, whereas π_2 does not contain β. Finally,

π_1 is fair since it contains infinitely many instances of the rightmost node, whereas π_2 is fair since it contains infinitely many instances of the fourth node.

Note that, if we restrict to live systems where every state is fair, then there is no need to require the existence of the suffixes $\pi_1[l+1,\infty]$ and $\pi_2[l+1,\infty]$. Thus, in these cases, the ribbon-shape critical pairs degenerate to the simple critical pairs of [30] where the traces may be unfair ($\exists$BoundDel$_{\mathsf{O}}$ case) or the faulty trace must be fair while the nominal trace may be unfair (FiniteDel case).

We can prove that, in the general case, β is not diagnosable if and only if there exists a ribbon-shape critical pair. In other words, ribbon-shape critical pairs are necessary and sufficient for diagnosability violation in the $\exists$ExactDel, $\exists$BoundDel$_{\mathsf{O}}$ and FiniteDel cases. The following theorem is adapted and extended from [29] and can be proved in a similar way.

Theorem 18 (Ribbon-Shape Critical Pairs necessary and sufficient for $\exists$ExactDel, $\exists$BoundDel$_{\mathsf{O}}$ and FiniteDel diagnosability)**.** *Let P be a plant and Ξ an alarm condition such that $\Xi = \exists\text{ExactDel}(\circ,\beta)$ or $\Xi = \exists\text{BoundDel}_{\mathsf{O}}(\circ,\beta)$ or $\Xi = \text{FiniteDel}(\circ,\beta)$. Ξ is not diagnosable iff there exists a ribbon-shape critical pair for Ξ as per Def. 9.*

Proof. The proof can be done similarly as in [29]. We use $\pi_{i,n} = \pi_i[0,k]\cdot\pi_i[k+1,l]^n\cdot\pi_i[l+1,\infty]$ to denote a trace π_i with n unrollings of the loop.

- If $\Xi = \exists\text{ExactDel}(\circ,\beta)$, then:

  Assume there exists a ribbon-shape critical pair π_1,π_2 for Ξ as per Definition 9, and let i, $i \le l$, be such that $\pi_1, i \models \beta$ and $\pi_2, i \not\models \beta$. Then for any $d \ge 0$ there exists n such that such that $i + d < k + n * (l - k)$. The inequality ensures that index $i + d$ is included in the prefix and unrolling of the loop. It is easy to verify that that $\pi_{1,n}$, $\pi_{2,n}$ are a critical pair for delay d at time i as per Definition 5.

  Vice versa, assume Ξ is not $\exists$ExactDel diagnosable. Then for each d we can find a critical pair π_1,π_2 and an index i such that ObsEqUpTo$(\pi_1,\pi_2,i+d)$, $\pi_1, i \models \beta$ and $\pi_2, i \not\models \beta$. Since P is finite, also pair of states of P are finite, hence there exists D large enough such that for some k, $0 \le k < i+D$, we have $\pi_1[i+D] = \pi_1[k]$ and $\pi_2[i+D] = \pi_2[k]$. From this, if we take $l = i + D$, it is easy to construct a ribbon-shape critical pair for Ξ as per Definition 9.

- If $\Xi = \exists\text{BoundDel}_{\mathsf{O}}(\circ,\beta)$, then:

  The proof is the same as for $\Xi = \exists\text{ExactDel}(\circ,\beta)$, only we have $\pi_2, i \not\models \beta$ for all $i \le l$.

- If $\Xi = \text{FiniteDel}(\circ,\beta)$, then:

  Assume there exists a ribbon-shape critical pair π_1,π_2 for Ξ as per Definition 9. For all $j \in \mathbb{N}$ there exists $n \in \mathbb{N}$ such that $j < k + n * (l - k)$ and $\pi_{2,n}$ is a trace of P. Since $obs(\pi_1) = obs(\pi_2')$, it holds that

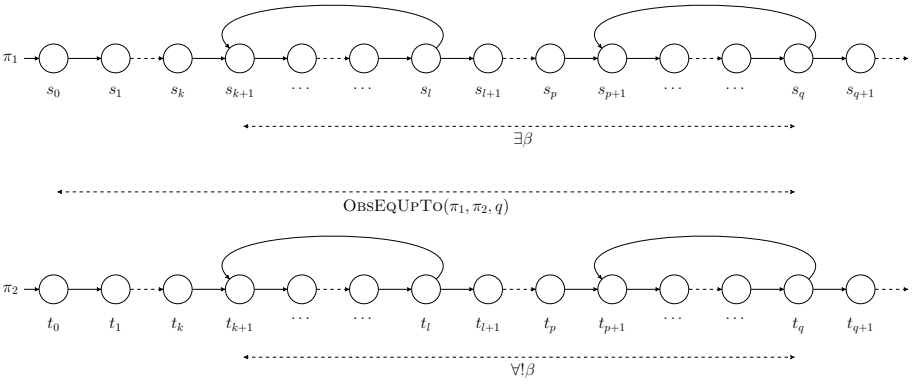

Figure 16: Double-ribbon-shape critical pair for BoundDel.

ObsEqUpTo$(\pi_1, \pi_{2,n}, j)$. Moreover, $\pi_{2,n}, i \not\models \beta$ for all $i \leq j$. Then a set $CS = \{\pi_1\} \cup \{\pi_{2,n}\}_{n\geq 1}$ is a a critical set as per Definition 7, hence Ξ is not FiniteDel diagnosable.

Vice versa, if Ξ is not FiniteDel diagnosable then there exists a critical set $\mathcal{CS}$ and an index i such that $\pi_1 \in \mathcal{CS}$, $\pi_1, i \models \beta$ and $\mathcal{CS}$ is s.t $\forall j.i \leq j$ $\exists \pi_2 \in \mathcal{CS}$, s.t. ObsEqUpTo$(\pi_1, \pi_2, j)$ and $\forall k \in \mathbb{N}.k \leq j \Rightarrow \pi_2, k \not\models \beta$. Since P is finite, there exists l large enough such that for some k, $0 \leq k < l$, we have that $\pi_2[k] = \pi_2[l]$ and $\pi_{2,n}$ a trace of P for all $n \geq 1$, $obs(\pi_1) = obs(\pi_2')$, where $\pi_2' = \pi_2[0, k] \cdot \pi_2[k+1, l]^\omega$ and $\pi_2' \not\models \beta$ for all $i \geq 0$, hence π_1 and π_2 are a ribbon-shape critical pair for Ξ as per Definition 9.

□

In the next section, we discuss a constructive diagnosability condition for the reamining ∃BoundDel case.

6.2. Double-Ribbon-Shape Critical Pairs

In the ∃BoundDel case, we can find a sufficient and necessary condition for diagnosability by extending Definition 9. We define so-called *double-ribbon-shape critical pairs*. Figure 16 pictorially illustrates the concept.

The formal definition is as follows.

Definition 10 (Double-Ribbon-Shape Critical Pairs (DRCP))**.** *Let P be a plant and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a double-ribbon-shape critical pair for an alarm condition $\Xi = \exists$BoundDel$(\circ, \beta)$ iff there are indices k, l, p, q such that $0 \leq k < l < p < q$ and:*

1. $\pi_1[k] = \pi_1[l]$, $\pi_1[p] = \pi_1[q]$, $\pi_2[k] = \pi_2[l]$, $\pi_2[p] = \pi_2[q]$
2. ObsEqUpTo(π_1, π_2, q)
3. $\pi_1, i \models \beta$ *for some* i, $k \leq i \leq q$, *and* $\pi_2, i \not\models \beta$ *for all* i, $k \leq i \leq q$.

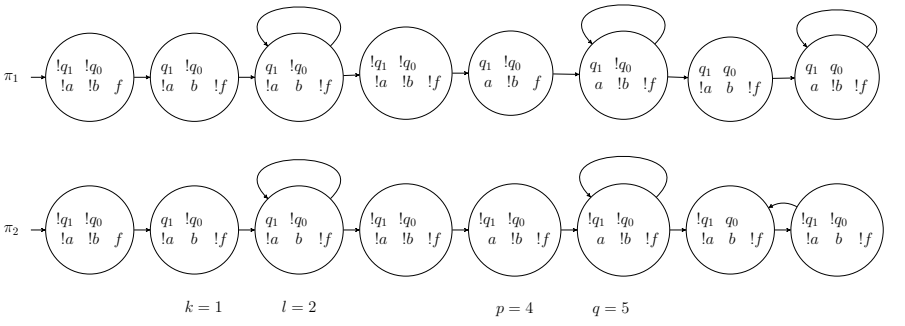

Figure 17: A BoundDel double-ribbon-shape critical pair for the example in Fig 8.

Fig. 17 shows a double-ribbon-shape critical pair for the example in Fig. 8. The two loops shown in Fig. 16 correspond to the two leftmost self loops at indices $l = 2$ and $q = 5$ in Fig. 17. Notice that the two traces π_1 and π_2 are observationally equivalent up to $q = 5$. Moreover, π_1 contains $\beta = f$ between indices k and q, whereas π_2 does not contain β between indices k and q. Finally, π_1 is fair since it contains infinitely many instances of the rightmost node, whereas π_2 is fair since it contains infinitely many instances of the seventh node.

In the $\exists$BoundDel case we can prove that β is not diagnosable if and only if there exists either a ribbon-shape critical pair or a double-ribbon-shape critical pair.

Theorem 19 ((Double-)Ribbon-Shape Critical Pairs necessary and sufficient for $\exists$BoundDel diagnosability)**.** *Let P be a plant and Ξ an alarm condition such that $\Xi = \exists\text{BoundDel}(\circ, \beta)$. Ξ is not diagnosable iff there exists either a ribbon-shape critical pair for Ξ as per Def. 9 for the $\exists\text{BoundDel}_{\mathsf{O}}$ case, or a double-ribbon-shape critical pair for Ξ as per Def. 10.*

Proof. If there exists a ribbon-shape critical pair for Ξ as per Definition 9, then Ξ is not $\exists\text{BoundDel}_{\mathsf{O}}$ diagnosable, and due to Theorem 4 it is also not $\exists$BoundDel diagnosable. If there exists a double-ribbon-shape critical pair for Ξ as per Definition 10, then it is easy to see that for each d we can construct a critical pair π_1, π_2 as per Definition 5, by unrolling the two loops a sufficient number of times (notice that, without loss of generality, we can assume that $\pi_1 \models \beta$ for some i, $l < i < p$), hence Ξ is not $\exists$BoundDel diagnosable.
Vice versa, if Ξ is not $\exists$BoundDel diagnosable, then due to Theor. 11 for all d there exists a critical pair π_1, π_2 for $\text{BoundDel}(\circ, \beta, d)$, as per Def. 5, and an index i such that $\pi_1, i \models \beta$, $\text{ObsEqUpTo}(\pi_1, \pi_2, i + d)$, and $\pi_2, j \not\models \beta$ for all $j \in \mathbb{N}$ s.t. $j \in [i - d, i + d]$. Since P is finite, also pair of states of P are finite, hence there exists D large enough such that for some p and q, with $i < p < q < i + D$, we have $\pi_1[q] = \pi_1[p]$ and $\pi_2[q] = \pi_2[p]$, hence we can construct the right loop in Fig. 14. We distinguish two cases. In the first case, we have that $i - D \leq 0$, hence we have constructed a ribbon-shape critical pair

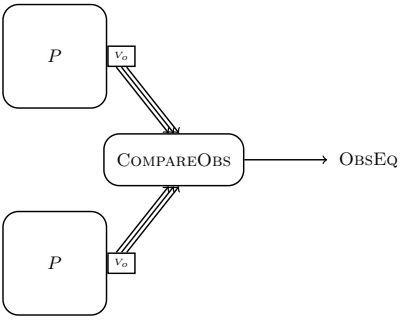

Figure 18: Twin Plant.

for the $\exists$BoundDel$_{\mathsf{O}}$ case as per Def. 9. In the second case, we have that $i - D > 0$. Again, if D is large enough, for some k and l, $i - D \leq k < l < i$, we have $\pi_1[l] = \pi_1[k]$ and $\pi_2[l] = \pi_2[k]$. Moreover, from $\pi_2, j \not\models \beta$ for all $j \in [i - D, i + D]$ it follows that $\pi_2, j \not\models \beta$ for all $j \in [k, q]$. Hence, we have constructed the left loop in Fig. 14, and a double-ribbon-shape critical pair for Ξ as per Definition 10. □

7. Verifying Diagnosability via Symbolic Model Checking

In this section, we analyze the verification of diagnosability via symbolic model checking.

We first remark that the problem in its generality is very hard. In general, diagnosability verification amounts to checking the existence of multiple traces at the same time (like in the cases of hyperproperties [31] and temporal epistemic logic [32, 33]).

In the rest of this section, we do not consider the context explicitly, since we can consider the synchronous product between a plant P and the SFTS S_Ψ corresponding to the context, as remarked in Section 3.

7.1. Finding Critical Pairs via Model Checking

We now reduce the check for existence of critical pairs for a given alarm condition Ξ to the problem of model-checking a suitable temporal formula on a model that is derived from P, called the *twin plant*. Although the general idea dates back to [4, 5], the formalization presented here takes into account the details related to the temporal nature of alarm condition.

7.1.1. The Twin Plant construction

The twin plant construction of a plant P, denoted Twin(P) and originally proposed by [4], is based on two copies of P, such that a trace in the twin plant corresponds to a pair of traces of P. The twin plant construction is pictorially illustrated in Fig. 18. The twin plant can be defined as the synchronous

product of two copies of the SFTS corresponding to the plant. Formally, given a plant $P = \langle V, V_o, I, T, F\rangle$, we denote with $P_L \doteq \langle V_L, V_{Lo}, I_L, T_L, F_L\rangle$ and $P_R \doteq \langle V_R, V_{Ro}, I_R, T_R, F_R\rangle$ the ('left' and 'right') copies of P, obtained by renaming each variable v as v_L or v_R, respectively (i.e., if $\square \in \{L, R\}$, then $V_\square$ stands for the set of variables $\{v_\square \mid v \in V\}$.

Moreover, we define a signal ObsEq stating that the sets of observable variables of the two copies are equal up to a given point (compare block CompareObs in Fig. 18), i.e., ObsEq evaluates to true on a trace π at time point i if and only if ObsEqUpTo(π_L, π_R, i), where π_L and π_R are the left and right copies of π. Later in this section, we will use this formula to identify critical pairs, on which we require that the observations do not diverge up to some point of interest. The twin plant of P is defined as follows.

Definition 7.1 (Twin Plant)**.** *The* twin plant *of* $P = \langle V, V_o, I, T, F\rangle$ *is the SFTS* Twin$(P) \doteq P_L \times P_R$. *Moreover, we define the formula* ObsEq $\doteq \mathsf{H}(\bigwedge_{v \in V_o} v_L = v_R)$.

There is a one-to-one correspondence between $\Pi_P \times \Pi_P$ (pairs of traces of P) and $\Pi_{\text{Twin}(P)}$ (traces of Twin(P)). A trace of Twin(P)

$$\pi \doteq (s_{0,L}, s_{0,R}), (s_{1,L}, s_{1,R}), \ldots$$

can be decomposed into two traces of P:

$$\mathit{Left}(\pi) \doteq s_{0,L}, s_{1,L}, \ldots \qquad \text{and} \qquad \mathit{Right}(\pi) \doteq s_{0,R}, s_{1,R}, \ldots$$

Conversely, given two observationally equivalent traces π_L and π_R in Π_P, there is a corresponding trace in $\Pi_{\text{Twin}(P)}$, denoted by $\pi_L \times \pi_R$.

The following theorem states the main property of ObsEq. We remind that the notation $obs(\pi_L^k) = obs(\pi_R^k)$ indicates that two traces π_L and π_R are indistinguishable up to some $k \geq 0$.

Theorem 20. *Let* $\pi_L, \pi_R \in \Pi_P$ *be two traces;* π_L *and* π_R *are such that* $obs(\pi_L^k) = obs(\pi_R^k)$ *iff the trace* $\pi_L \times \pi_R \in \Pi_{\text{Twin}(P)}$ *is such that* $(\pi_L \times \pi_R)[k] \models$ ObsEq.

Proof. Immediate by definitions. □

7.1.2. Model-Checking on the Twin Plant

A consequence of Theorem 20 is that critical pairs of the plant can be mapped to special traces on the twin plant. Thus, it is possible to reduce the problem of falsification of diagnosability for P to a problem of finding those special traces in Twin(P). This problem can then be solved using model-checking techniques. In particular, for each alarm condition Ξ defined in the previous section, we generate – given the alarm condition – a temporal property $CP(\Xi)$ to be checked on Twin(P), as defined in Table 1. We call such formula the *critical pair formula*, that is satisfied by the critical pairs of Ξ.

Ξ	$CP(\Xi)$						
$\textsc{ExactDel}(\circ, \beta, d)$	$\mathsf{F}(\textsc{ObsEq}$	$\wedge$	$\mathsf{Y}^d \beta_L$	$\wedge$	$\mathsf{Y}^d \neg\beta_R)$	$\wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R$	
$\textsc{BoundDel}(\circ, \beta, d)$	$\mathsf{F}(\textsc{ObsEq}$	$\wedge$	$\mathsf{Y}^d \beta_L$	$\wedge$	$\mathsf{H}^{\leq 2d} \neg\beta_R)$	$\wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R$	
$\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$	$\mathsf{F}(\textsc{ObsEq}$	$\wedge$	$\mathsf{Y}^d \beta_L$	$\wedge$	$(\mathsf{H}\ \neg\beta_R)$	$\wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R$	
$\textsc{FiniteDel}(\circ, \beta)$	$(\mathsf{G}\ \textsc{ObsEq})$	$\wedge$	$(\mathsf{F}\ \beta_L)$	$\wedge$	$(\mathsf{G}\ \neg\beta_R)$	$\wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R$	
$\exists\textsc{ExactDel}(\circ, \beta)$	$(\mathsf{G}\ \textsc{ObsEq})$	$\wedge$	$\mathsf{F}\ (\ \beta_L$	$\wedge$	$\neg\beta_R)$	$\wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R$	
$\exists\textsc{BoundDel}(\circ, \beta)$	$CP(\textsc{FiniteDel}(\circ, \beta))$						
$\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$	$CP(\textsc{FiniteDel}(\circ, \beta))$						

Table 1: Critical Pair: LTL formulae.

In the formulae of Table 1, β_L and β_R are the formulae corresponding to β instantiated for the left and right copies of the plant. I.e, β_L [β_R] is the formula obtained from β by the parallel substitution of every occurrence of v with v_L, respectively v_R, for all $v \in V$. For instance, $\beta_L \wedge \neg\beta_R$ identifies the states of the twin plant where the left state satisfies the diagnosis condition, and the right one does not. Moreover, we denote by *Fairness* the LTL formula $\bigwedge_{f \in F}(\mathsf{GF} f)$, i.e. the conjunction of the formulae defining the fairness conditions, and by $\mathit{Fairness}_L$, $\mathit{Fairness}_R$ the corresponding formulae instantiated for the left and right copies.

Intuitively, the property $CP(\textsc{ExactDel}(\circ, \beta, d))$ states the existence of a pair of traces that are (i) indistinguishable until a given time t, (ii) d steps earlier (i.e. at $t-d$) β holds in the left trace, and (iii) in the same state $(t-d)$ β does not hold on the right one. For the bounded case, $CP(\textsc{BoundDel}(\circ, \beta, d))$ states the existence of a pair of traces that are (i) indistinguishable until a given time t, (ii) d steps earlier (i.e. at $t-d$) β holds in the left trace, and (iii) β does not hold on the right trace in the range from $t-2d$ to t (note that the width of the interval is dictated by Definition 5 for the bounded case). The $CP(\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d))$ case is similar to the previous one, but point (iii) prescribes that β does not hold on the right trace from the origin up to the current time point. In the finite delay case, the property $CP(\textsc{FiniteDel}(\circ, \beta))$ states the existence of a pair of traces that are (i) indistinguishable, (ii) β holds at some unspecified point in the left trace, and (iii) β never holds in the right one. As for the existential exact delay, $CP(\exists\textsc{ExactDel}(\circ, \beta))$ requires the existence of a pair of traces that are (i) indistinguishable, (ii) β holds at some unspecified point in the left trace, and (iii) β does not hold at that point in the right trace. As the definition of critical pairs for $\exists\textsc{BoundDel}$ and $\exists\textsc{BoundDel}_{\mathsf{O}}$ coincides with the one for $\textsc{FiniteDel}$, we reuse the definition of $CP(\textsc{FiniteDel}(\circ, \beta))$ to define $CP(\exists\textsc{BoundDel}(\circ, \beta))$ and $CP(\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta))$.

Theorem 21. *Let P be a plant, Ξ an alarm condition, and $CP(\Xi)$ as defined in Table 1. There exists a critical pair for Ξ iff* $\textsc{Twin}(P) \not\models \neg CP(\Xi)$.

Proof. We reason by cases and use the LTL semantics to prove the theorem. We use Theorem 20 to guarantee that we only consider pairs of traces that are observationally equivalent up to some $k \in \mathbb{N}$.

- Case $\Xi = \textsc{ExactDel}(\circ, \beta, d)$.

  $\textsc{Twin}(P) \not\models \neg(\mathsf{F}(\textsc{ObsEq} \wedge \mathsf{Y}^d \beta_L \wedge \mathsf{Y}^d \neg\beta_R) \wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R)$ iff $\exists \pi \in \Pi_{\textsc{Twin}(P)}$, decomposable in $\pi_L \times \pi_R$, such that (by semantics of LTL) $\pi_L \in \Pi_{P_L}$ and $\pi_R \in \Pi_{P_R}$, $\exists k \in \mathbb{N}$ with $k \geq d$ such that $\pi, k \models \textsc{ObsEq}$, i.e. $obs(\pi_L^k) = obs(\pi_R^k)$, $\pi_L, k-d \models \beta$, and $\pi_R, k-d \not\models \beta$, iff π_L, π_R are a critical pair for Ξ.

- Case $\Xi = \textsc{BoundDel}(\circ, \beta, d)$.

  $\textsc{Twin}(P) \not\models \neg(\mathsf{F}(\textsc{ObsEq} \wedge \mathsf{Y}^d \beta_L \wedge \mathsf{H}^{\leq 2d} \neg\beta_R) \wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R)$ iff $\exists \pi \in \Pi_{\textsc{Twin}(P)}$, decomposable in $\pi_L \times \pi_R$, such that (by semantics of LTL) $\pi_L \in \Pi_{P_L}$ and $\pi_R \in \Pi_{P_R}$, $\exists k \in \mathbb{N}$ with $k \geq d$ such that $\pi, k \models \textsc{ObsEq}$, i.e. $obs(\pi_L^k) = obs(\pi_R^k)$, $\pi_L, k-d \models \beta$, and $\forall j \in [k-2d, k] \cdot \pi_R, j \not\models \beta$, iff π_L, π_R are a critical pair for Ξ.

- Case $\Xi = \textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$.

  $\textsc{Twin}(P) \not\models \neg(\mathsf{F}(\textsc{ObsEq} \wedge \mathsf{Y}^d \beta_L \wedge \mathsf{H} \neg\beta_R) \wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R)$ iff $\exists \pi \in \Pi_{\textsc{Twin}(P)}$, decomposable in $\pi_L \times \pi_R$, such that (by semantics of LTL) $\pi_L \in \Pi_{P_L}$ and $\pi_R \in \Pi_{P_R}$, $\exists k \in \mathbb{N}$ with $k \geq d$ such that $\pi, k \models \textsc{ObsEq}$, i.e. $obs(\pi_L^k) = obs(\pi_R^k)$, $\pi_L, k-d \models \beta$, and $\forall j \leq k \cdot \pi_R, j \not\models \beta$, iff π_L, π_R are a critical pair for Ξ.

- Case $\Xi = \textsc{FiniteDel}(\circ, \beta)$.

  $\textsc{Twin}(P) \not\models \neg((\mathsf{G}\ \textsc{ObsEq}) \wedge (\mathsf{F}\ \beta_L) \wedge (\mathsf{G}\ \neg\beta_R) \wedge \mathit{Fairness}_L \wedge \mathit{Fairness}_R)$ iff $\exists \pi \in \Pi_{\textsc{Twin}(P)}$, decomposable in $\pi_L \times \pi_R$, such that (by semantics of LTL) $\pi_L \in \Pi_{P_L}$ and $\pi_R \in \Pi_{P_R}$, $\forall k \in \mathbb{N} \cdot obs(\pi_L^k) = obs(\pi_R^k)$ and thus $obs(\pi_L) = obs(\pi_R)$, $\pi_L, j \models \beta$ for some $j \in \mathbb{N}$, and $\pi_R, j \not\models \beta$ for any $j \in \mathbb{N}$, iff π_L, π_R are a critical pair for Ξ.

- Case $\Xi = \exists\textsc{ExactDel}(\circ, \beta)$.

  $\textsc{Twin}(P) \not\models \neg((\mathsf{G}\ \textsc{ObsEq}) \wedge \mathsf{F}\ (\beta_L \wedge \neg\beta_R) \wedge \ \mathit{Fairness}_L \wedge \mathit{Fairness}_R)$ iff $\exists \pi \in \Pi_{\textsc{Twin}(P)}$, decomposable in $\pi_L \times \pi_R$, such that (by semantics of LTL) $\pi_L \in \Pi_{P_L}$ and $\pi_R \in \Pi_{P_R}$, $\forall k \in \mathbb{N} \cdot obs(\pi_L^k) = obs(\pi_R^k)$ (thus $obs(\pi_L) = obs(\pi_R)$), and there exists $j \in \mathbb{N}$ such that $\pi_L, j \models \beta$ and $\pi_R, j \not\models \beta$ iff π_L, π_R are a critical pair for Ξ.

- The cases $\Xi = \exists\textsc{BoundDel}(\circ, \beta)$ and $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$ are the same as for $\Xi = \textsc{FiniteDel}(\circ, \beta)$.

□

Complexity. Given an LTL formula ϕ and a symbolic transition system with n states, the model checking problem can be solved in $2^{O(|\phi|)}O(n)$ [34]. The exponential dependency on $|\phi|$ is due to the construction of a Büchi automaton with the same language of ϕ. The complexity can be refined considering that the size of the Büchi automaton is actually exponential in the number of temporal operators in ϕ and the number of state variables [23]. Moreover, the $\mathit{Fairness}_L \wedge$

Ξ	$CS(\Xi)$
$\textsc{BoundDel}(\circ, \beta, d)$	$(\mathsf{F}(\mathsf{Y}^d \beta_1 \wedge \bigwedge_{l=2,\ldots,d+2}(\mathsf{Y}^{d-l+2}(\textsc{ObsEq}_{1,l} \wedge \mathsf{H}^{\leq d}\neg\beta_l)))) \wedge \bigwedge_{l=1,\ldots,d+2} \mathit{Fairness}_l$

Table 2: Critical Set: LTL formula for the BoundDel case.

$\mathit{Fairness}_R$ formula can be encoded into a linear Büchi automaton. Thus, the formula of the critical pair for ExactDel, BoundDel, and $\textsc{BoundDel}_{\mathsf{O}}$ can be converted into an automaton with a number of states that is $O(|F|)\cdot 2^{O(|\beta|+d)}$, while at most $|F|^2 \cdot 2^{|\beta|}$ in the case of FiniteDel. Moreover, note that the twin plant construction uses a linear number of state variables and thus the number of states of the twin plant is exponential in $|V|$. Therefore, applying model checking on the twin plant can be solved in time $O(|F|)2^{O(|\beta|+d+|V|)}$ for ExactDel, BoundDel, and $\textsc{BoundDel}_{\mathsf{O}}$ and $O(|F|)2^{O(|\beta|+|V|)}$ for FiniteDel.

7.2. Finding Critical Sets via Model Checking

The twin plant construction can be generalized to the case of critical sets. In particular, we are interested in looking for critical sets in the BoundDel case, since critical pairs alone are sufficient in the ExactDel case, whereas in the FiniteDel case searching for critical sets is not appealing, since the cardinality of a critical set may be infinite (diagnosability in the FiniteDel case is addressed in Section 7.3 using ribbon-shape critical pairs).

Since critical sets may be arbitrarily large in the BoundDel case, we need to consider N copies of the plant running in parallel, instead of just two. We call this construction *multiple twin plant*, denoted N-Twin(P). Similarly, we can reduce the problem of falsification of diagnosability for P to a model checking problem consisting in searching for traces in N-Twin(P).

Below we sketch this construction and the corresponding results – that can be proved in a similar way as in Section 7.1.

We denote with $P_i \doteq \langle V_i, V_{i_o}, I_i, T_i, F_i\rangle$ the i-th copy of P, obtained by renaming each variable v as v_i. Moreover, we define the formulae $\textsc{ObsEq}_{i,j}$ stating that the sets of observable variables of the i-th and j-th copies are equal. The multiple twin plant of P is defined as follows.

Definition 7.2 (Multiple Twin Plant)**.** *Let $N \in \mathbb{N}$, with $N \geq 2$. The* multiple twin plant *of $P = \langle V, V_o, I, T, F\rangle$ is the SFTS N-Twin(P) $\doteq \times_{i=1,\ldots,N} P_i$. Moreover, we define the formulae* $\textsc{ObsEq}_{i,j} \doteq \mathsf{H}(\bigwedge_{v\in V_o} v_i = v_j)$.

The definition is based on N copies of the plant $P_1, \ldots P_N$. When looking for a critical set of N traces, in P_1 we search for the trace where the diagnosis condition holds (compare trace π_1 in Definition 7), whereas in $P_2 \ldots P_N$ we search for the remaining traces, where the diagnosis condition does not hold. The variables $\textsc{ObsEq}_{1,j}$ keep track of whether the observable state variables have diverged in the past, for pairs of copies of the plant (copy P_1 versus the

generic copy P_i with $i = 2, \ldots, N$). Note that for $N = 2$ this construction reduces to the standard twin plant definition.

Similarly to Section 7.1, we define a temporal property $CS(\Xi)$ that we want to model check on N-Twin(P). We remark that, in the BoundDel case which we are interested in, given a delay d, it is sufficient to consider an N-Twin(P) where $N = d + 2$, according to Theorem 16.

The formula for the BoundDel case is given in Table 2. Here, β_i is the formula corresponding to β instantiated in the i-th copy of the plant. The property $CS(\text{BoundDel}(\circ, \beta, d))$ states the existence of a critical set according to Definition 7, in particular we are looking for a trace $\pi_1 \in \Pi_{P_1}$ and a time point i where β_1 holds and for each index j in the range from i to $i + d$, we have a trace $\pi_l \in \Pi_{P_l}$, with $j = l - 2 + i$, where β_l is observationally equivalent to β_1 up to j and it does not hold for d steps in the past. Moreover, all the traces $\pi_1, \ldots, \pi_{d+2}$ must satisfy the fairness conditions.

We have the following theorem.

Theorem 22. *Let P be a plant, $\Xi = \text{BoundDel}(\circ, \beta, d)$, and $CS(\Xi)$ as defined in Table 2. There exists a critical set for Ξ iff N-Twin(P) $\not\models \neg CS(\Xi)$, with $N = d + 2$.*

Proof. We sketch the proof, which can be carried out in a similar way as for Theorem 21 (an analogous of Theorem 20 can also be stated). Let $N = d + 2$. We have that N-Twin(P) $\not\models \neg(\mathsf{F}(\mathsf{Y}^d\beta_1 \wedge \bigwedge_{l=2,\ldots,d+2}(\mathsf{Y}^{d-l+2}(\text{ObsEq}_{1,l} \wedge \mathsf{H}^{\leq d}\neg\beta_l))))) \wedge \bigwedge_{l=1,\ldots,d+2}$ *Fairness*$_l$ iff $\exists \pi \in \Pi_{N\text{-Twin}(P)}$, which is decomposable in $\pi_1 \times \ldots \times \pi_N$, such that (by semantics of LTL) $\pi_l \in \Pi_{Pl}$ for each $l = 1, \ldots, N$, and there exists an index $i \in \mathbb{N}$ such that $\pi_1, i \models \beta_1$ and for each $l = 2, \ldots, N$, $\pi_l, i + l - 2 \models \text{ObsEq}_{1,l}$, and $\pi_l, k \not\models \beta_l$ for all $i + l - 2 - d \leq k \leq i + l - 2$, i.e., by letting $j = i + l - 2$, $\pi_l, j \models \text{ObsEq}_{1,l}$, and $\pi_l, k \not\models \beta_l$ for all $j - d \leq k \leq j$ iff $\{\pi_1, \ldots, \pi_N\}$ is a critical set for Ξ at time i. □

Complexity. In this case the LTL formula can be converted into a Büchi automaton with at most $O(d|F|) \cdot 2^{O(d(|\beta|+d))}$. The N-Twin(P) has a number of variables linear in $d \cdot |V|$. Thus, similarly to the critical pair case, checking the existence of a critical set for Ξ can be solved in $O(d|F|)2^{O(d(|\beta|+d+|V|))}$.

7.3. Finding Ribbon-Shape Critical Pairs via Model Checking

The existence of ribbon-shape critical pairs for a given alarm condition Ξ can also be reduced to a model checking problem on the twin plant. In this case, we need to use branching properties to look for a possibly non-fair infinite loop, from which there exists a branching trace starting from a state in the loop. Figure 19 shows a procedure deciding the existence of a ribbon-shape critical pair in the given SFTS P and given pattern Ξ. At first, we find all fair states of the original plant using Emerson-Lei algorithm [35] denoted by the procedure compute_fair_states. Then, we create a new twin plant TP, where we consider only fair states on both components of the twin plant Twin(P).

Subscripts L and R refer to all variables in the formula shifted to the left and right part of the twin plant respectively. The resulting TP is an STS, whose traces do not need to satisfy any fairness conditions. This twin plant is used for patterns $\exists\textsc{ExactDel}(\circ,\beta)$ and $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$, where we look for a loop without any requirements on fairness. For pattern $\textsc{FiniteDel}(\circ,\beta)$ we add fairness conditions for the left trace of TP, which are the same as in the original plant. For all cases, we compute the set of states satisfying the provided CTL formulas by using the subprocedure $\mathsf{compute\_sat\_states}$. Finally, there exists a ribbon-shape critical pair iff there is an initial state in TP satisfying the CTL formula.

Theorem 23. *Let P be a plant and Ξ the alarm condition $\exists\textsc{ExactDel}(\circ,\beta)$ or $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$ or $\textsc{FiniteDel}(\circ,\beta)$. There exists a ribbon-shape critical pair for Ξ iff Algorithm $\textsc{RibbonCriticalPairFound}(P,\Xi)$ returns true.*

Proof. We reason by cases.

- Case $\Xi = \exists\textsc{ExactDel}(\circ,\beta)$.
  $\Rightarrow$) Suppose that there exists a ribbon-shape critical pair π_1,π_2 for Ξ. Let TP be defined as in the Algorithm. We can combine and extend π_1 and π_2 to a trace of TP. With abuse of notation, we refer to such trace as $\pi_{1L}\times\pi_{2R}$. From the fact that π_1,π_2 are traces of P, we deduce that the state $\pi_1[l]_L\times\pi_2[l]_R$ of TP satisfy $\mathsf{fair\_states}_L\wedge\mathsf{fair\_states}_R$. From conditions 3 of Definition 9 for the Ξ pattern, and the properties of P_β, we deduce that $\mathsf{TP},\pi_1[i]_L\times\pi_2[i]_R \models \mathsf{beta}_L\wedge\neg\mathsf{beta}_R)$ and, from condition 1 and 3 that $\pi_1[0]_L\times\pi_2[0]_R \in [[\mathsf{EF}\ (\mathsf{beta}_L\wedge\neg\mathsf{beta}_R)]]$. Since $\pi_1[0]_L\times\pi_2[0]_R$ is also an initial state of TP, the Algorithm returns true.

  $\Leftarrow$) Suppose the Algorithm returns true, then there exists an initial state $s_L\times t_R$ of TP such that $\mathsf{TP}, s_L\times t_R \models \mathsf{EF}\ (\mathsf{beta}_L\wedge\neg\mathsf{beta}_R)$. Thus there exists a trace $\pi_{0L}\times\pi_{0R}$ starting from $\pi_{0L}[0]=s_L$, $\pi_{0R}[0]=t_R$, and such that, for some $k\geq 0$, $\mathsf{TP},\pi_{0L}[k]\times\pi_{0R}[k] \models \mathsf{beta}_L\wedge\neg\mathsf{beta}_R$. Since TP has finitely many states, we can assume without loss of generality that $\pi_{0L}\times\pi_{0R}$ is in the form $u_0,u_1,\dots u_k,(u_{k+1},\dots u_l)^\omega$ for some k,l. Note that, since $\pi_{0L}[l]\times\pi_{0R}[l] \models \mathsf{fair\_states}_L\wedge\mathsf{fair\_states}_R$ there exists traces π_{1L},π_{1R} starting from $\pi_{1L}[0]=\pi_{0L}[l]$ and $\pi_{1R}[0]=\pi_{0R}[l]$, respectively. We can now define π_1,π_2 as follows: for i, $0\leq i\leq l$, $\pi_1[i]=\pi_{0L}[i]$ and $\pi_2[i]=\pi_{0R}[i]$; for i, $l\leq i$, $\pi_1[i]=\pi_{1L}[i-l]$ and $\pi_2[i]=\pi_{1R}[i-l]$. Note that π_1 is a trace of P_β and thus for all $i\geq 0$, $\pi_1[i]\models\mathsf{beta}$ iff $\pi_{1|V},i\models\phi$, and the same holds for π_2. Therefore, it is easy to see that $\pi_{1|V},\pi_{2|V}$ satisfy the conditions of Definition 9 for the Ξ pattern and thus form a ribbon-shape pair.

- Case $\Xi = \exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$.

  $\Rightarrow$) Suppose that there exists a ribbon-shape critical pair π_1,π_2 for Ξ. Let TP be defined as set in the Algorithm and let $\pi_1'=\pi_1[0,k-1]\cdot\pi_1[k,l]^\omega$ and $\pi_2'=\pi_2[0,k-1]\cdot\pi_2[k,l]^\omega$ using indices k,l as in Definition 9. We

can combine and extend π'_1 and π'_2 to a trace of TP. With abuse of notation, we refer to such trace as $\pi'_{1L} \times \pi'_{2R}$. From the condition 3 of Definition 9 for the Ξ pattern, $\pi'_{1L} \times \pi'_{2R} \models \mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))$. From the fact that π_1, π_2 are traces of P and the condition 2 of Definition 9 for Ξ, we deduce that all states of $\pi'_{1L} \times \pi'_{2R}$ satisfy the invariant $\mathsf{fair\_states}_L \wedge \mathsf{fair\_states}_R \wedge \mathsf{obs\_eq}$ of TP. Thus, $\pi'_1[0]_L \times \pi'_2[0]_R \in [[\mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))]]$. Since $\pi'_1[0]_L \times \pi'_2[0]_R$ is also an initial state of TP, the Algorithm returns true.

$\Leftarrow$) Suppose the Algorithm returns true, then there exists an initial state $s_L \times t_R$ of TP such that $\mathsf{TP}, s_L \times t_R \models \mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))$. Thus there exists a trace $\pi_{0L} \times \pi_{0R}$ starting from $\pi_{0L}[0] = s_L$, $\pi_{0R}[0] = t_R$, and such that, for some $i \geq 0$, $\mathsf{TP}, \pi_{0L}[i] \times \pi_{0R}[i] \models \mathsf{beta}_L$ and, for all $i \geq 0$, $\mathsf{TP}, \pi_{0L}[i] \times \pi_{0R}[i] \models \neg\mathsf{beta}_R$. Since TP has finitely many states, we can assume without loss of generality that $\pi_{0L} \times \pi_{0R}$ is in the form $u_0, u_1, \ldots u_k, (u_{k+1}, \ldots u_l)^\omega$ for some k, l. Note that, since $\pi_{0L}[l] \times \pi_{0R}[l] \models \mathsf{fair\_states}_L \wedge \mathsf{fair\_states}_R$ there exists traces π_{1L}, π_{1R} starting from $\pi_{1L}[0] = \pi_{0L}[l]$ and $\pi_{1R}[0] = \pi_{0R}[l]$, respectively. We can now define π_1, π_2 as follows: for i, $0 \leq i \leq l$, $\pi_1[i] = \pi_{0L}[i]$ and $\pi_2[i] = \pi_{0R}[i]$; for i, $l \leq i$, $\pi_1[i] = \pi_{1L}[i - l]$ and $\pi_2[i] = \pi_{1R}[i - l]$. Note that π_1 is a trace of P_β and thus for all $i \geq 0$, $\pi_1[i] \models \mathsf{beta}$ iff $\pi_{1|V}, i \models \phi$, and the same holds for π_2. Therefore, it is easy to see that $\pi_{1|V}, \pi_{2|V}$ satisfy the conditions of Definition 9 for the Ξ pattern and thus form a ribbon-shape pair.

- Case $\Xi = \textsc{FiniteDel}(\circ, \beta)$.
  $\Rightarrow$) Suppose that there exists a ribbon-shape critical pair π_1, π_2 for Ξ. Let TP be defined as set in the Algorithm and let π'_2 as in Definition 9. We can combine and extend π_1 and π'_2 to a trace of TP. With abuse of notation, we refer to such trace as $\pi_{1L} \times \pi'_{2R}$. From the condition 3 of Definition 9 for the Ξ pattern, $\pi_{1L} \times \pi'_{2R} \models \mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))$. From the fact that π_1 and π_2 are traces of P and the condition 2 of Definition 9 for Ξ, we deduce that all states of $\pi_{1L} \times \pi'_{2R}$ satisfy the invariant $\mathsf{fair\_states}_L \wedge \mathsf{fair\_states}_R \wedge \mathsf{obs\_eq}$ of TP. Thus, $\pi_1[0]_L \times \pi'_2[0]_R \in [[\mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))]]$. Since $\pi_1[0]_L \times \pi_2[0]_R$ is also an initial state of TP, the Algorithm returns true.

  $\Leftarrow$) Suppose the Algorithm returns true, then there exists an initial state $s_L \times t_R$ of TP such that $\mathsf{TP}, s_L \times t_R \models \mathsf{E}\ (\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{EG}\neg\mathsf{beta}_R))$. Thus there exists a trace $\pi_{0L} \times \pi_{0R}$ starting from $\pi_{0L}[0] = s_L$, $\pi_{0R}[0] = t_R$, and such that, for some $i \geq 0$, $\mathsf{TP}, \pi_{0L}[i] \times \pi_{0R}[i] \models \mathsf{beta}_L$ and, for all $i \geq 0$, $\mathsf{TP}, \pi_{0L}[i] \times \pi_{0R}[i] \models \neg\mathsf{beta}_R$. Since TP has finitely many states, we can assume without loss of generality that $\pi_{0L} \times \pi_{0R}$ is in the form $u_0, u_1, \ldots u_k, (u_{k+1}, \ldots u_l)^\omega$ for some k, l. Note that given the fairness condition of TP, π_{0L} is fair. Moreover, since $\pi_{0R}[l] \in \mathsf{fair\_states}_R$ there exists a trace π_{1R} starting from $\pi_{1R}[0] = \pi_{0R}[l]$. We can now define π_2 as follows: for i, $0 \leq i \leq l$, $\pi_2[i] = \pi_{0R}[i]$; for i, $l \leq i$, $\pi_2[i] = \pi_{1R}[i - l]$. Note that π_1 is a trace of P_β and thus for all $i \geq 0$, $\pi_1[i] \models \mathsf{beta}$ iff $\pi_{1|V}, i \models \phi$,

Algorithm RibbonCriticalPairFound(P, Ξ)

```
1   (let P_β and Enc(β) as defined in Section 2.3.1)
2   beta := Enc(β);
3   fair_states := compute_fair_states(P_β);
4   TP := add_invar(Twin(P_β), fair_states_L ∧ fair_states_R ∧ obs_eq);
5
6   if Ξ = ∃ExactDel(∘, β) then
7     TP := set_fairness_conditions(TP, ∅);
8     sat_states := compute_sat_states(TP, EF (beta_L ∧ ¬beta_R));
9
10  if Ξ = ∃BoundDel_O(∘, β) then
11    TP := set_fairness_conditions(TP, ∅);
12    sat_states := compute_sat_states(TP, E (¬beta_R U (beta_L ∧ EG¬beta_R)));
13
14  if Ξ = FiniteDel(∘, β) then
15    fairness := get_fairness_condition(P);
16    TP := set_fairness_conditions(TP, fairness_L);
17    sat_states := compute_sat_states(TP, E (¬beta_R U (beta_L ∧ EG¬beta_R)));
18
19  return (get_init(TP) ∧ sat_states) ≠ ⊥;
```

Figure 19: Algorithm for deciding the existence of an RCP.

and the same holds for π_2. Therefore, $\pi_{1|V}, \pi_{2|V}$ satisfy the conditions of Definition 9 for the Ξ pattern and thus form a ribbon-shape pair.

□

Complexity. The CTL model checking problem under fairness can be solved in time linear in the number of states and size of the fairness formula [36, 37]. Thus checking the existence of a ribbon-shape critical pair can be solved in $O(|F|)2^{O(|\beta|+|V|)}$ time.

7.4. Finding Double-Ribbon-Shape Critical Pairs via Model Checking

The existence of double-ribbon-shape critical pairs for a given alarm condition Ξ can also be reduced to a model checking problem on the twin plant. Figure 20 shows a procedure deciding the existence of a double-ribbon-shape critical pair in the given fair Kripke structure P and given pattern Ξ. Similarly as for RCP, the twin plant is constrained to fair states on both sides. The double-ribbon-shape fair critical pair formula is rewritten in CTL. Using compute_sat_states we find all states in TP satisfying the CTL formula. There is a double-ribbon-shape critical pair iff there is an initial state satisfying the formula.

Theorem 24. *Let P be a plant and $\Xi = \exists$BoundDel. There exists a double-ribbon-shape critical pair for Ξ iff Algorithm* DoubleRibbonCriticalPairFound(P, Ξ) *returns true.*

Proof. $\Rightarrow$) Suppose there exists a double-ribbon-shape critical pair π_1, π_2 for Ξ. Let TP be defined as set in the Algorithm. We can combine and extend π_1 and π_2 to a trace of TP. With abuse of notation, we refer to such trace as $\pi_{1L} \times \pi_{2R}$. From condition 3 of Definition 10 we have that $\pi_{1L} \times \pi_{2R}, i \models \beta_L$ for some $i, k \leq i \leq q$. This means that the fault occurs in one of the loops or in between them. We assume that the fault occurs in between the loops, that is $l < i < p$. If the fault is only in a loop, we can unroll the loop and create a new critical pair that satisfies the assumption.

From the fact that π_1 and π_2 are traces of P, we deduce that the state $\pi_1[q]_L \times \pi_2[q]_R$ of TP satisfy $\mathsf{fair\_states}_L \wedge \mathsf{fair\_states}_R$. From conditions 1 and 3 of Definition 10, we deduce that $\mathsf{TP}, \pi_1[p]_L \times \pi_2[p]_R \models EG(\neg \mathsf{beta}_R)$ and thus $\pi_1[p]_L \times \pi_2[p]_R \in [[\mathsf{second\_ribbon}]]$ of the Algorithm. From the assumption about the occurrence of β_L, we then deduce that $\mathsf{TP}, \pi_1[l]_L \times \pi_2[l]_R \models \mathsf{E}\,(\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{E}\,(\neg \mathsf{beta}_R\ \mathsf{U}\ \mathsf{second\_ribbon})))$ and thus $\pi_1[l]_L \times \pi_2[l]_R \in [[\mathsf{center}]]$ of the Algorithm. Then, $\mathsf{TP}, \pi_1[k]_L \times \pi_2[k]_R \models \mathsf{EG}\,(\neg \mathsf{beta}_R \wedge \mathsf{center})$ and thus $\pi_1[k]_L \times \pi_2[k]_R \in [[\mathsf{first\_ribbon}]]$. Finally, $\mathsf{TP}, \pi_1[0]_L \times \pi_2[0]_R \models \mathsf{EF}\,(\mathsf{first\_ribbon})$ and since $\pi_1[0]_L \times \pi_2[0]_R$ is also an initial state of TP, the Algorithm returns true.

$\Leftarrow$) Suppose the Algorithm returns true, then there exists an initial state $s_L \times t_R$ of TP such that $\mathsf{TP}, s_L \times t_R \models \mathsf{EF}\,(\mathsf{first\_ribbon})$. Thus, there exists a trace $\pi_{0L} \times \pi_{0R}$ such that $\pi_{0L}[0] = s_L$, $\pi_{0R}[0] = t_R$, and, for some $k \geq 0$, $\mathsf{TP}, \pi_{0L}[k] \times \pi_{0R}[k] \models \mathsf{first\_ribbon} = \mathsf{EG}\,(\neg \mathsf{beta}_L \wedge \neg \mathsf{beta}_R \wedge \mathsf{center})$. Thus, there exists a trace $\pi_{1L} \times \pi_{1R}$ such that $\pi_{1L}[0] = \pi_{0L}[k]$, $\pi_{1R}[0] = \pi_{0R}[0]$, and, for all $i \geq 0$ $\mathsf{TP}, \pi_{1L}[i] \times \pi_{1R}[i] \models \neg \mathsf{beta}_R \wedge \mathsf{center}$. Since TP has finitely many states, we can assume without loss of generality that $\pi_{1L} \times \pi_{1R}$ is in the form $u_0, u_1, \ldots u_{k'}, (u_{k'+1}, \ldots u_l)^\omega$ for some k', l. Note that $\mathsf{center} = \mathsf{E}\,(\neg \mathsf{beta}_R\ \mathsf{U}\ (\mathsf{beta}_L \wedge \mathsf{E}\,(\neg \mathsf{beta}_R\ \mathsf{U}\ \mathsf{second\_ribbon})))$. Thus there exists a trace $\pi_{2L} \times \pi_{2R}$ such that $\pi_{2L}[0] = \pi_{1L}[l]$, $\pi_{0R}[0] = \pi_{1R}[l]$, and, for some $p \geq 0$, $\mathsf{TP}, \pi_{2L}[p] \times \pi_{2R}[p] \models \mathsf{second\_ribbon} = \mathsf{EG}\,(\neg \mathsf{beta}_R)$. Thus, there exists a trace $\pi_{3L} \times \pi_{3R}$ such that $\pi_{3L}[0] = \pi_{2L}[p]$, $\pi_{3R}[0] = \pi_{2R}[p]$ and, for all $i \geq 0$ $\mathsf{TP}, \pi_{1L}[i] \times \pi_{1R}[i] \models \neg \mathsf{beta}_L \wedge \neg \mathsf{beta}_R$. Since P has finitely many states, we can assume without loss of generality that $\pi_{3L} \times \pi_{3R}$ is in the form $u_0, u_1, \ldots u_{p'}, (u_{p'+1}, \ldots u_q)^\omega$ for some p', q. Since $\pi_{3L}[q] \in \mathsf{fair\_states}_L$ there exists a trace π_{4L} such that $\pi_{4L}[0] = \pi_{3L}[q]$ and similarly there exists a trace π_{4R} such that $\pi_{4R}[0] = \pi_{3R}[q]$. We can now construct π_1, π_2 by joining prefixes of π_{0-4L} and π_{0-4R} respectively. We show the formal definition in Table 3. Note that π_1 is a trace of P_β and thus for all $i \geq 0$, $\pi_1[i] \models \mathsf{beta}$ iff $\pi_{1|V}, i \models \phi$, and the same holds for π_2. Therefore, it is easy to see that $\pi_{1|V}, \pi_{2|V}$ satisfy the conditions of Definition 10 and thus form a double-ribbon-shape pair. $\square$

Complexity. As in the previous case, checking the existence of a double-ribbon-shape critical pair can be solved in $O(|F|)2^{O(|\beta|+|V|)}$ time.

Example results. Table 4 shows diagnosability results for the models introduced in the previous sections for the existential patterns and the finite delay pattern. The results are obtained by the procedures in Figure 19 and Figure 20.

i	$\leq k$	$\leq k+l$	$\leq k+l+p$	$\leq k+l+p+q$	$< \infty$
j	i	$i-k$	$i-k-l$	$i-k-l-p$	$i-k-l-p-q$
$\pi_1[i]$	$\pi_{0L}[j]$	$\pi_{1L}[j]$	$\pi_{2L}[j]$	$\pi_{3L}[j]$	$\pi_{4L}[j]$
$\pi_2[i]$	$\pi_{0R}[j]$	$\pi_{1R}[j]$	$\pi_{2R}[j]$	$\pi_{3R}[j]$	$\pi_{4R}[j]$

Table 3: Formal definition of π_1, π_2 proving correctness of DoubleRibbonCriticalPairFound(P, Ξ). For each i, we define $\pi_1[i]$ and $\pi_2[i]$ as $\pi_{0-4L}[j]$ and $\pi_{0-4R}[j]$ respectively. The index j is selected such that we join in π_{1-2} the prefixes of π_{0-4L-R}.

Algorithm DoubleRibbonCriticalPairFound(P, Ξ)
1 (let P_β and $Enc(\beta)$ as defined in Section 2.3.1)
2 beta := $Enc(\beta)$;
3 fair_states := compute_fair_states(P_β);
4 TP := add_invar(Twin(P_β), fair_states$_L$ $\wedge$ fair_states$_R$ $\wedge$ obs_eq);
5 TP := set_fairness_conditions(TP, $\emptyset$);
6 second_ribbon := EG ($\neg$beta$_R$)
7 center := E ($\neg$beta$_R$ U (beta$_L$ $\wedge$ E ($\neg$beta$_R$ U second_ribbon)))
8 first_ribbon := EG ($\neg$beta$_R$ $\wedge$ center)
9 sat_states := compute_sat_states(TP, EF (first_ribbon));
10 **return** (get_init(TP) $\wedge$ sat_states) $\neq \bot$;

Figure 20: Algorithm for deciding the existence of a DRCP.

7.5. Monotonicity and Dependencies of Critical Pairs

We conclude by proving some results about monotonicity and dependencies of critical pairs. In particular, Theorems 25 and 26 follow directly from the analogous Theorems 8 and 9 proved in Section 4.3. They are in some sense dual to Theorems 3 and 4, which describe monotonicity properties and dependencies in cases a specification *is* diagnosable. However, as we have proved in Section 4, the two notions are not equivalent.

Theorem 25 (Monotonicity of Critical Pairs – LTL formulas)**.** *The following conditions hold for all β d and d', with $d' > d$.*

1. $\models CP(\text{ExactDel}(\circ, \beta, d')) \rightarrow CP(\text{ExactDel}(\circ, \beta, d))$.
2. $\models CP(\text{BoundDel}(\circ, \beta, d')) \rightarrow CP(\text{BoundDel}(\circ, \beta, d))$.
3. $\models CP(\text{BoundDel}_{\mathsf{O}}(\circ, \beta, d')) \rightarrow CP(\text{BoundDel}_{\mathsf{O}}(\circ, \beta, d))$.

Proof. The proof follows directly from Theorem 21 and Theorem 8. □

Theorem 26 (Dependencies between Critical Pairs – LTL formulas)**.** *The following conditions hold for all β and d.*

1. $\models CP(\text{BoundDel}(\circ, \beta, d)) \rightarrow CP(\text{ExactDel}(\circ, \beta, d))$.
2. $\models CP(\text{BoundDel}_{\mathsf{O}}(\circ, \beta, d)) \rightarrow CP(\text{BoundDel}(\circ, \beta, d))$.
3. $\models CP(\text{FiniteDel}(\circ, \beta)) \rightarrow CP(\text{BoundDel}_{\mathsf{O}}(\circ, \beta, d))$.

Proof. The proof follows directly from Theorem 21 and Theorem 9. □

	$\exists\text{ED}(\circ,\beta)$	$\exists\text{BD}(\circ,\beta)$	$\exists\text{BD}_{\mathsf{O}}(\circ,\beta)$	$\text{FD}(\circ,\beta)$
Figure 7	N	N	N	Y
Figure 8	N	N	Y	Y
Figure 9	N	Y	Y	Y
Figure 10	N	N	N	N
Figure 11	N	Y	Y	Y
Figure 13	N	N	N	N

Table 4: Diagnosability results for selected patterns. Y means that β is diagnosable in the plant with respect to the given pattern. N means that it is non-diagnosable.

8. Integrated Approach to Diagnosability

In Section 5 we have shown that the verification of diagnosability for an alarm condition Ξ on the plant P, amounts to proving the absence of a critical set, according to Theorem 15. Further, in Section 7 we have formulated verification of diagnosability as a model checking problem, and proved several results about critical sets and critical pairs.

The direct construction of critical sets does not appear to be promising. In Section 6 we have identified constructive conditions to search for critical sets for existential patterns and for the FiniteDel case, namely (double-)ribbon-shape critical pairs. Thus, we approach the problem of verification of diagnosability using a two-layer approach. First, we provide a complete machinery to analyze the presence/absence of ((double-)ribbon-shape) critical pairs. Second, we propose a high-level algorithm that integrates multiple calls to detect the presence/absence of ((double-)ribbon-shape) critical pairs, based on several properties (dependencies conditions) relating different alarm conditions.

The BoundDel case is not covered by our algorithm, since it still requires the construction of critical sets. In theory, this case could be solved by resorting to the search of critical sets. Here, we omit this step as it would require construction of the multiple twin plant, which may be computationally expensive, since it requires to construct the synchronous product of N copies of the plant, where N is equal to $d+2$ in the worst case. Moreover, we will show in Section 9, that this construction is not necessary in the presented industrial case studies.

We now describe an integrated diagnosability algorithm, which attempts to conclude the diagnosability status of a diagnosis condition β in a plant P. The algorithm builds a map of the diagnosability status of the following alarm conditions: $\text{FiniteDel}(\circ,\beta)$, $\exists\text{BoundDel}_{\mathsf{O}}(\circ,\beta)$, $\exists\text{BoundDel}(\circ,\beta)$ and $\exists\text{ExactDel}(\circ,\beta)$; $\text{BoundDel}_{\mathsf{O}}(\circ,\beta,d)$, $\text{BoundDel}(\circ,\beta,d)$ and $\text{ExactDel}(\circ,\beta,d)$ for a delay d ranging from 0 to a given threshold D. The map is stored as three arrays DiagE, DiagB, $\text{DiagB}_{\mathsf{O}}$ from 0 to D and four variables DiagEE DiagEB, $\text{DiagEB}_{\mathsf{O}}$, DiagF. Individual values of the arrays (of the variables) represent the diagnosability status (N for not diagnosable, U for unknown and Y for diagnosable).

The algorithm relies on the formal results proved in Sections 3.2, 4, 5, and 7.

(Thm 4, 3)	$d-1$		d	d		$d+1$
$\textsc{ExactDel}(\circ,\beta,d)$	$\neg Diag$	$\Leftarrow$	$\neg Diag$	$Diag$	$\Rightarrow$	$Diag$
	$\Uparrow$		$\Uparrow$	$\Downarrow$		$\Downarrow$
$\textsc{BoundDel}(\circ,\beta,d)$	$\neg Diag$	$\Leftarrow$	$\neg Diag$	$Diag$	$\Rightarrow$	$Diag$
	$\Uparrow$		$\Uparrow$	$\Downarrow$		$\Downarrow$
$\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta,d)$	$\neg Diag$	$\Leftarrow$	$\neg Diag$	$Diag$	$\Rightarrow$	$Diag$
		$\Uparrow$			$\Downarrow$	
$\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$		$\neg Diag$			$Diag$	
		$\Uparrow$			$\Downarrow$	
$\textsc{FiniteDel}(\circ,\beta)$		$\neg Diag$			$Diag$	
$\textsc{BoundDel}(\circ,\beta,d)$	$\neg Diag$	$\Leftarrow$	$\neg Diag$	$Diag$	$\Rightarrow$	$Diag$
		$\Uparrow$			$\Downarrow$	
$\exists\textsc{BoundDel}(\circ,\beta)$		$\neg Diag$			$Diag$	
		$\Uparrow$			$\Downarrow$	
$\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$		$\neg Diag$			$Diag$	
$\textsc{ExactDel}(\circ,\beta,d)$	$\neg Diag$	$\Leftarrow$	$\neg Diag$	$Diag$	$\Rightarrow$	$Diag$
		$\Uparrow$			$\Downarrow$	
$\exists\textsc{ExactDel}(\circ,\beta)$		$\neg Diag$			$Diag$	
		$\Uparrow$			$\Downarrow$	
$\exists\textsc{BoundDel}(\circ,\beta)$		$\neg Diag$			$Diag$	

Table 5: Diagnosability Dependencies.

Alarm Condition Ξ	Diagnosability criteria	
$\textsc{ExactDel}(\circ,\beta,d)$	$\neg\exists CP(\Xi)$	(Thm 7)
$\textsc{BoundDel}(\circ,\beta,d)$	$\neg\exists CS(\Xi)$	(Thm 15)
	$d=2h$ and $\neg\exists CP(\textsc{BoundDel}(\circ,\beta,h))$	(Thm 10)
	β is persistent and $\neg\exists CP(\Xi)$	(Thm 13)
	β is persistent and $\neg\exists CP(\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta,d))$	(Thm 12)
	[$\neg\exists CP(\Xi)$]	(Thm 6)
$\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta,d)$	$\neg\exists CP(\Xi)$	(Thm 7)
$\exists\textsc{ExactDel}(\circ,\beta)$	$\neg\exists RCP(\Xi)$	(Thm 18)
	[$\neg\exists CP(\Xi)$]	(Thm 14)
$\exists\textsc{BoundDel}(\circ,\beta)$	$\neg\exists RCP(\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)) \wedge \neg\exists DRCP(\Xi)$	(Thm 19)
	[$\neg\exists CP(\Xi)$]	(Thm 14)
$\exists\textsc{BoundDel}_{\mathsf{O}}(\circ,\beta)$	$\neg\exists RCP(\Xi)$	(Thm 18)
	[$\neg\exists CP(\Xi)$]	(Thm 14)
$\textsc{FiniteDel}(\circ,\beta)$	$\neg\exists RCP(\Xi)$	(Thm 18)
	[$\neg\exists CP(\Xi)$]	(Thm 6)

Table 6: Necessary and sufficient conditions for diagnosability (conditions listed in square brackets are only necessary).

We summarize these results in Table 5 and Table 6. The first class of results concerns the dependencies between the diagnosability of various alarm conditions for the same diagnosis condition β, and is summarized in Table 5. The second class of results is about (necessary and) sufficient conditions for diagnosability, and is summarized in Table 6. We use the twin plant construction described in Section 7.1 as a building block to search for critical pairs. Similarly, we can search for (double-)ribbon-shape critical pairs on the twin plant as described in Section 7.3 and Section 7.4. For the finite and existential patterns, we search both for CP and RCP. In some cases, finding CP can be faster than proving non-existence of RCP. In case β is persistent, we use results from Theorem 12 and propagate diagnosability from $\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta, d)$ pattern to $\textsc{BoundDel}(\circ, \beta, d)$.

In the implemented algorithm, we search for all critical pairs in parallel. When the existence of some critical pair, ribbon-shape critical pair or double-ribbon-shape critical pair is decided, the result is propagated to decide the diagnosability of other patterns/delays using the rules in Table 5.

The algorithm is presented in Figure 21. The algorithm uses the subroutines in Figure 22, Figure 23 and Figure 24 to propagate the results of the terminating runs. The only non-trivial case is for deciding diagnosability of DiagEB, where neither RCP for $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$ nor DRCP for $\exists\textsc{BoundDel}(\circ, \beta)$ are found. Here, values ebdo_rcp_not_found and ebd_drcp_not_found are used to denote the temporary status.

For each pattern and delay we run a separate subprocess. The subprocesses are running in the background. We assume each subprocess has its own core and enough resources to run in parallel. When a subprocess terminates, it writes the computed result along with procedure name and used alarm condition to queue unprocessed_results.

In the main process, results are popped from the queue and are used to update the Diag map. We assume that Diag arrays and values are global variables and can be accessed from all the subroutines. After propagation of the computed values, the subprocesses that are no longer necessary are killed by the main process. The integrated algorithm stops once all subprocesses have terminated or were killed. Since we omitted the procedure for computing critical sets, there may be some U results at the end of the computation.

The algorithm uses the following primitives:

- $\textsc{IsPersistent}(P, \beta)$ returns true iff $P \models \mathsf{G}(\mathsf{O}\beta \to \beta)$.
- $\textsc{CriticalPairFound}(P, \Xi)$ is the primitive that searches for a critical pair based on the twin plant construction, as presented in Section 7.1. Namely, $\textsc{CriticalPairFound}(P, \Xi)$ holds iff $\textsc{Twin}(P) \not\models \neg CP(\Xi)$. Note that the cases of critical pairs for $\exists\textsc{BoundDel}(\circ, \beta)$ and $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, \beta)$ reduce to the $\textsc{FiniteDel}(\circ, \beta)$ case.
- $\textsc{RibbonCriticalPairFound}(P, \Xi)$ is the primitive that searches for a ribbon-shape critical pair based on the twin plant construction, as presented in Section 7.3.

```
function IntegratedDiagnosability(P, β, D)
1   unprocessed_results : Queue;
2   DiagE, DiagB, DiagB_O: array [0..D] of {N, U, Y};
3   DiagEE, DiagEB_O, DiagF: {N, U, Y};
4   DiagEB: {N, U, Y, ebdo_rcp_not_found, ebd_drcp_not_found};
5   DiagE[0..D] := DiagB[0..D] := DiagB_O[0..D] := U;
6   DiagEE := DiagEB := DiagEB_O := DiagF := U;
7   // Start all subprocesses and run them in the background
8   run subprocess CriticalPairFound(P, ExactDel(∘, β, [0..D]));
9   run subprocess CriticalPairFound(P, BoundDel(∘, β, [0..D]));
10  run subprocess CriticalPairFound(P, BoundDel_O(∘, β, [0..D]));
11  run subprocess CriticalPairFound(P, FiniteDel(∘, β));
12  run subprocess CriticalPairFound(P, ∃ExactDel(∘, β));
13  run subprocess RibbonCriticalPairFound(P, ∃ExactDel(∘, β));
14  run subprocess RibbonCriticalPairFound(P, ∃BoundDel_O(∘, β));
15  run subprocess RibbonCriticalPairFound(P, FiniteDel(∘, β));
16  run subprocess DoubleRibbonCriticalPairFound(P, ∃BoundDel(∘, β));
17
18  while any subprocess is running:
19    while unprocessed_results is not empty:
20      procedure, alarm_condition, result := pop(unprocessed_results);
21      switch procedure:
22        case CriticalPairFound:
23          PropagateResult_CriticalPairFound(P, alarm_condition, result);
24        case RibbonCriticalPairFound:
25          PropagateResult_RibbonCriticalPairFound(P, alarm_condition, result);
26        case DoubleRibbonCriticalPairFound:
27          PropagateResult_DoubleRibbonCriticalPairFound(P, alarm_condition, result);
```

Figure 21: An integrated diagnosability algorithm.

```
function PropagateResult_CriticalPairFound(P, alarm_condition, cp_found)
1   switch alarm_condition, cp_found:
2     case ExactDel(∘, β, d), True:
3       DiagE[0..d] := N;
4     case BoundDel(∘, β, d), True:
5       DiagE[0..d] := DiagB[0..d] := N;
6     case BoundDel_O(∘, β, d), True:
7       DiagE[0..d] := DiagB[0..d] := DiagB_O[0..d] := N;
8     case FiniteDel(∘, β), True:
9       DiagEE := DiagE[0..D] := N;
10      DiagEB := DiagB[0..D] := N;
11      DiagEB_O := DiagB_O[0..D] := N;
12      DiagF := N;
13    case ∃ExactDel(∘, β), True:
14      DiagEE := DiagE[0..D] := N;
15
16    case ExactDel(∘, β, d), False:
17      DiagEE := DiagE[d..D] := Y;
18      DiagEB := DiagB[d..D] := Y;
19      DiagEB_O := DiagB_O[d..D] := Y;
20      DiagF := Y;
21    case BoundDel(∘, β, d), False:
22      DiagEB := DiagEB_O := DiagF := Y;
23      if IsPersistent(P, β) then
24        DiagB[d..D] := DiagB_O[d..D] := Y;
25      else
26        DiagB[2d..D] := DiagB_O[2d..D] := Y;
27    case BoundDel_O(∘, β, d), False:
28      DiagEB_O := DiagB_O[d..D] := Y;
29      DiagF := Y;
30      if IsPersistent(P, β) then
31        DiagEB := DiagB[d..D] := Y;
32    case FiniteDel(∘, β), False:
33      // unknown
34    case ∃ExactDel(∘, β), False:
35      // unknown
```

Figure 22: Propagate results for CriticalPairFound procedure.

```
function PropagateResult_RibbonCriticalPairFound(P, alarm_condition, rcp_found)
1   switch alarm_condition, rcp_found:
2     case ∃ExactDel(∘, β), True:
3       DiagEE := DiagE[0..D] := N;
4     case ∃BoundDel_O(∘, β), True:
5       DiagEE := DiagE[0..D] := N;
6       DiagEB := DiagB[0..D] := N;
7       DiagEB_O := DiagB_O[0..D] := N;
8     case FiniteDel(∘, β), True:
9       DiagEE := DiagE[0..D] := N;
10      DiagEB := DiagB[0..D] := N;
11      DiagEB_O := DiagB_O[0..D] := N;
12      DiagF := N;
13
14    case ∃ExactDel(∘, β), False:
15      DiagEE := DiagEB := DiagEB_O := DiagF := Y;
16    case ∃BoundDel_O(∘, β), False:
17      DiagEB_O := DiagF := Y;
18      if DiagEB == ebd_drcp_not_found then
19        DiagEB := Y;
20      else
21        DiagEB := ebdo_rcp_not_found;
22    case FiniteDel(∘, β), False:
23      DiagF := Y;
```

Figure 23: Propagate results for RibbonCriticalPairFound procedure.

```
function PropagateResult_DoubleRibbonCriticalPairFound(P, alarm_condition, drcp_found)
1   // Alarm condition is ∃BoundDel(∘, β)
2   if drcp_found then
3     DiagEE := DiagE[0..D] := DiagEB := DiagB[0..D] := N;
4   else
5     if DiagEB == ebdo_rcp_not_found then
6       DiagEB := Y;
7     else
8       DiagEB := ebd_drcp_not_found;
```

Figure 24: Propagate results for DoubleRibbonCriticalPairFound procedure.

- DoubleRibbonCriticalPairFound(P, Ξ) is the primitive that searches for a double-ribbon-shape critical pair based on the twin plant construction, as presented in Section 7.4.

We now illustrate the algorithm using the transmitter example described in Section 4.2. If we run the algorithm with the inputs P as the plant shown in Figure 11, $\beta \doteq !in$ and with a context $\Psi \doteq in \wedge Xin$, and $D = 5$, we obtain the results shown in Table 7.

d	DiagE	DiagB	DiagB$_\mathsf{O}$	DiagEE	DiagEB	DiagEB$_\mathsf{O}$	DiagF
0	N	N	Y				
1	N	U	Y				
2	N	Y	Y	N	Y	Y	Y
3	N	Y	Y				
4	N	Y	Y				
5	N	Y	Y				

Table 7: Representation of the output of the algorithm in Figure 21.

Assume the subprocesses have finished in the following order. We demonstrate which values were propagated and which subprocesses were killed.

- CriticalPairFound$(P, \text{BoundDel}(\circ, \beta, 0))$ finds CP, then the result N is propagated to DiagE[0] and DiagB[0] and the subprocess CriticalPairFound$(P, \text{ExactDel}(\circ, \beta, 0))$ is killed.

- CriticalPairFound$(P, \text{ExactDel}(\circ, \beta, 3))$ finds CP, then the result N is propagated to DiagE[1..3] and the subprocesses CriticalPairFound$(P, \text{ExactDel}(\circ, \beta, 1..2))$ are killed.

- CriticalPairFound$(P, \text{BoundDel}(\circ, \beta, 2))$ does not find CP. Because β is not persistent, the result Y is propagated to DiagB[4..5], DiagB$_\mathsf{O}$[4..5], DiagEB, DiagEB$_\mathsf{O}$ and DiagF. The following subprocesses are killed:

  - CriticalPairFound$(P, \text{BoundDel}(\circ, \beta, 4..5))$,
  - CriticalPairFound$(P, \text{BoundDel}_\mathsf{O}(\circ, \beta, 4..5))$,
  - CriticalPairFound$(P, \text{FiniteDel}(\circ, \beta))$,
  - RibbonCriticalPairFound$(P, \exists\text{BoundDel}_\mathsf{O}(\circ, \beta))$,
  - RibbonCriticalPairFound$(P, \text{FiniteDel}(\circ, \beta))$
  - and DoubleRibbonCriticalPairFound$(P, \exists\text{BoundDel}(\circ, \beta))$.

- CriticalPairFound$(P, \text{BoundDel}_\mathsf{O}(\circ, \beta, 0))$ does not find CP, then the result Y is propagated to DiagB$_\mathsf{O}$[0..3] and the subprocesses CriticalPairFound$(P, \text{BoundDel}_\mathsf{O}(\circ, \beta, 1..3))$ are killed.

- RibbonCriticalPairFound($P, \exists$ExactDel$(\circ, \beta)$) finds RCP, then the result N is propagated to DiagE[4..5] and DiagEE and the subprocesses CriticalPairFound(P, ExactDel$(\circ, \beta, 4..5)$) and CriticalPairFound($P, \exists$ExactDel$(\circ, \beta)$) are killed.

- CriticalPairFound(P, BoundDel$(\circ, \beta, 1)$) does not find CP. Because β is not persistent, DiagB[1] is set to U and DiagB[2..3] is set to Y. The last subprocesses CriticalPairFound(P, BoundDel$(\circ, \beta, 2..3)$) are killed.

To summarize, there is no CP nor RCP for FiniteDel (thus diagnosable for FiniteDel); there is a CP for BoundDel$(\circ, \beta, 0)$ and no CP for BoundDel$(\circ, \beta, 1)$ (thus, not diagnosable for BoundDel$(\circ, \beta, 0)$ and diagnosable for BoundDel$(\circ, \beta, 2..5)$); finally, there is a CP for ExactDel$(\circ, \beta, 0..5)$ (and thus not diagnosable for ExactDel$(\circ, \beta, 0..5)$). Since β is not persistent, we have an Unknown result for the BoundDel$(\circ, \beta, 1)$ case. Our algorithm does not conclude in this case, since it does not implement the construction of the multiple twin plant; however, in Section 4.2 we have concluded that this case is not diagnosable, by building a critical set by hand (compare Fig. 12).

We remark that the value of the algorithm to the designer is in the production of a map that effectively supports the analysis of the diagnosability requirements. For a given plant, the algorithm constructs a global view of the diagnosability delays that can be achieved relative to the diagnosis condition of interest.

It is easy to prove that the algorithm is sound, namely that if the algorithm provides a conclusive answer (Y or N) for a given alarm condition Ξ, then Ξ is indeed diagnosable (not diagnosable). The proof follows from the soundness of the propagation rules and their correct implementation in the algorithm. Conversely, it is possible to show that the algorithm is complete, i.e. it provides a conclusive answer, in all cases except for the BoundDel pattern (where an Unknown result, for one or more consecutive delays, may be returned).

9. Experimental Evaluation

This section describes the experiments performed to evaluate the feasibility of the proposed techniques in solving the diagnosability problems. We first describe the implementation of the algorithms and their subprocedures, then the system models used for the tests, the experimental set-up, and finally the results. Tools, benchmarks, and extended results are available at `https://es-static.fbk.eu/publications/resources/AIJ-2022-diagnosability.zip`.

9.1. Implementation

The decision procedures we have proposed are implemented in the xSAP toolset [12]. xSAP is a platform for safety analysis, which offers several functionalities including design, verification and synthesis of FDI components. xSAP ar-

chitecture relies on the NUXMV model-checker as a back-end, a symbolic model-checker for finite and infinite-state transition systems [22]. XSAP leverages the algorithms provided by NUXMV, in particular for verification of diagnosability we use the LTL model-checking procedures provided by NUXMV. We use MATHSAT [38] for the SAT-solving steps. XSAP also provides the core of the diagnosability functionalities of the COMPASS toolset [16, 39], a design environment for system-software co-engineering.

In the rest of this section, we assume the plant is described in SMV, the input language of XSAP. We create a copy of this SMV model and introduce in the new model a variable that encodes the semantics of OBSEQ. Based on this, the LTL properties of Table 1 are verified. For the other patterns, procedures described in Figure 19 and Figure 20 are used. For simplicity, we did not implement the algorithm for propagating values and killing subprocesses. Instead, we ran all subprocesses in parallel and then analyzed the results, propagations and runtimes using spreadsheets. In this section, we show results obtained by the analysis which simulate the integrated algorithm.

9.2. Benchmark Set

To investigate the performance of diagnosability verification with the described algorithms we used the following benchmark models. The context definitions, if any, are all expressed by safety properties. The diagnosis conditions are all monotonic (permanent faults).

ACEX and AUTOGEN are hand-crafted models based on a state space derived from partially random graphs. The goal is to diagnose whether certain states have been visited, based on partial observability of the state vector.

CASSINI_DIS is a model describing the propulsion system of the Cassini spacecraft [40, 41]. It is composed of two engines fed by redundant propellant/gas circuit lines, which contain several valves and pyro-valves.

C432 is a Boolean circuit used as a benchmark in the DX Competition [42], whose gates can permanently fail (inverted output). The observables are the inputs and output values for the gates of the circuit. The property is whether a single gate of a certain group of gates is faulty. A context specification is used to limit the analysis to single fault cases.

GUIDANCE is a model of the Space Shuttle engines contingency guidance procedure. Certain actions and intermediate milestones are the observables. The diagnosis condition is passing a specific state of the procedure.

POWERDIST describes a power distribution system consisting of circuit breakers, switches, and power lines. Commands to the system, power availability in various parts and broken-status of switches are the observables. The diagnosis condition here is the failure of a specific power line; we use a context to avoid triggering of masking failures.

Basic indicators on the size of the models are given in Table 8: number of Boolean variables, number of reachable states, diameter of the state space (minimum number of steps to reach any reachable state starting from any initial state), and number of observables.

model	# bool var	# reach	diam	# obs
acex	31	$2^{19.4}$	96	5, 9, 12, 17, 21
autogen	99	$2^{12.0}$	20	4, 8, 12, 16, 20
cassini_dis	176	$2^{44.2}$	8	5, 10, 15, 20, 25, 58
c432	356	N.A.	N.A.	15, 20, 25, 30, 196
guidance	98	$2^{47.5}$	70	5, 10, 15, 20, 25, 62
powerdist	83	$2^{11.0}$	31	5, 10, 15, 20, 25, 41

Table 8: Model Properties.

We write N.A. (Not Available) when it was not possible to compute the set of reachable states/diameter, because the model was too large to complete a BDD-based computation of the reachable states. The subsequent computations are done by way of SAT-based algorithms that are typically able to deal with models having a larger number of variables. Note that these metrics refer to the original plant, not to the (parameterized) twin plant.

In our experiments, we compared results for each model for different sets of observables. We do this to test how does the delay for diagnosable cases decrease with an increased number of observables. We start with a small subset of the full set of variables and then we increasingly add more variables from the full set. Thus, with an increasing number of observables, we are testing a superset of the small set. In practice, this allows us to propagate some results: if β is diagnosable in some Y, than it is diagnosable in any $Y', Y \subseteq Y'$. Note that when we change the observable set, the set of state variables stays the same.

9.3. Experimental Set-Up

With the experimental evaluation we want to investigate feasibility, scalability w.r.t. alarm patterns, delay bounds, and number of observables. To this aim we created several instances for each use case, with variations in alarm pattern, delay bound up to 20, and sets of observables, resulting in a total of 2277 instances. For all the instances we managed to either compute the diagnosability result or to deduce it from a stronger result.

For the experiments we used a cluster where each node has 48GB of RAM and a CPU running at 2.27GHz. Each test was run on a single core, with a new invocation of the model-checker. Timeout was set at 7200 seconds, memory cap to 16GB.

9.4. Results

Table 9 shows summarized results for all used models, number of observable variables and patterns. The results were computed using LTL model checking with IC3 algorithm to find CP for patterns ED, BD, BD_O, $\exists$ED and FD, and by deciding existence of (D)RCP by computing fixpoint of CTL operators on BDD for patterns $\exists$ED, $\exists$BD, $\exists\text{BD}_\text{O}$ and FD. For the existential patterns, only diagnosability result Y or N is reported. For the other patterns, we report either negative results N (*) meaning that we can deduce the model is not diagnosable for any integer delay, or positive results Y (d) meaning that the

model	#obs	DIAG							IA (s)	max (s)
		E	EE	B	EB	B_O	EB_O	F		
acex	5	N (*)	N	Y (5)	Y	Y (5)	Y	Y	211.95	2104.52
acex	9	N (*)	N	Y (5)	Y	Y (5)	Y	Y	197.60	6142.79
acex	13	N (*)	N	Y (5)	Y	Y (5)	Y	Y	228.28	4211.99
acex	17	N (*)	N	Y (5)	Y	Y (5)	Y	Y	158.92	TO
acex	21	N (*)	N	Y (5)	Y	Y (5)	Y	Y	205.24	2557.14
autogen	4	N (*)	N	Y (6)	Y	Y (6)	Y	Y	11.10	786.67
autogen	8	N (*)	N	Y (6)	Y	Y (6)	Y	Y	10.49	TO
autogen	12	N (*)	N	Y (6)	Y	Y (6)	Y	Y	10.20	TO
autogen	16	N (*)	N	Y (6)	Y	Y (6)	Y	Y	11.19	7127.04
autogen	20	N (*)	N	Y (6)	Y	Y (6)	Y	Y	9.95	5645.71
cassini_dis	5	Y (2)	Y	Y (2)	Y	Y (2)	Y	Y	3.96	TO
cassini_dis	10	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	0.76	TO
cassini_dis	15	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	1.14	TO
cassini_dis	20	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	1.02	TO
cassini_dis	25	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	1.09	TO
cassini_dis	58	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	1.30	TO
c432	15	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	14.65	TO
c432	20	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	16.94	TO
c432	25	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	27.36	TO
c432	30	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	18.93	TO
c432	196	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	13.45	TO
guidance	5	Y (3)	Y	Y (3)	Y	Y (3)	Y	Y	8.60	295.94
guidance	10	Y (3)	Y	Y (3)	Y	Y (3)	Y	Y	9.60	1055.37
guidance	15	Y (3)	Y	Y (3)	Y	Y (3)	Y	Y	9.38	3904.56
guidance	20	Y (3)	Y	Y (3)	Y	Y (3)	Y	Y	9.11	3634.71
guidance	25	Y (2)	Y	Y (2)	Y	Y (2)	Y	Y	10.09	TO
guidance	62	Y (2)	Y	Y (2)	Y	Y (2)	Y	Y	9.63	TO
powerdist	5	Y (5)	Y	Y (5)	Y	Y (5)	Y	Y	5.57	16.37
powerdist	10	Y (5)	Y	Y (5)	Y	Y (5)	Y	Y	5.80	21.37
powerdist	15	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	6.05	18.14
powerdist	20	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	5.97	18.17
powerdist	25	Y (1)	Y	Y (1)	Y	Y (1)	Y	Y	5.89	18.65
powerdist	41	Y (0)	Y	Y (0)	Y	Y (0)	Y	Y	1.11	31.12

Table 9: Results for all models/observables/patterns.

model is diagnosable for delays equal and greater than d. Since all diagnosis conditions are persistent, there are no U results.

For each line we report the time needed to compute and propagate results to all cells in the line. We assume that all invocations of xSAP run in parallel and that each result is propagated as described in the integrated diagnosability algorithm. Hence, the first reported time corresponds the execution time of the integrated algorithm (IA) until all DIAG cells are filled with Y or N, i. e. the time of the last subprocess. Moreover, we display the maximum execution time (max) of the worst performing check, which is the time necessary to compute all cells without propagation rules, with timeout 7200 seconds.

In case the integrated algorithm could not finish in the given timeout, we used some simplifications over the model. Namely, for one model (c432) and all observation/pattern combinations we used an overapproximation by disregarding the context.

In the following, we summarize for each model, both with propagation and without propagation, which was the last check, that is the check with the longest execution time.

ACEX With propagation, the last checks were for delay = 4 for patterns BD or BD_O. Without propagation, the last checks were for pattern ED and delays 12, 6, 6, 8, 8 for the respective lines.

AUTOGEN With propagation, for 8 observables the last check was for delay = 5 and pattern BD_O, and for the others the last checks were for ∃ED pattern with CP procedure. Without propagation, for 4 observables the last check was for FD with RCP procedure, for 8 and 12 observables all RCP checks for existential patterns timed out, and for observables 16 and 20 the last checks were for ∃BD pattern with RCP procedure.

CASSINI_DIS With propagation, for 5 observables the last check was for pattern ED and delay = 1, for the others pattern ED and delay = 0. Without propagation, all RCP checks for existential patterns timed out.

C432 With propagation, for 15 and 20 observables the last checks were for pattern BD and delay = 0, for 25, 30 and 196 observables for ED pattern and delay = 0. Without propagation, all RCP checks for existential patterns timed out.

GUIDANCE With propagation, for 5, 25 and 62 observables the last checks were for pattern ED and delay = 2 or delay = 1, for 10 observables the last check was for pattern BD_O and delay = 2, for observables 15 and 20 the last checks were for pattern BD and delay = 2. Without propagation, for 5, 10 and 20 observables the last check was for ∃BD pattern with RCP procedure, for 15 observables the last check was for FD with RCP procedure. For 25 and 62 observables all RCP check for existential patterns timed out.

POWERDIST With propagation, for 5 and 20 observables the last checks were for pattern BD_O and delay = 0, for 10, 15, 25 and 41 observables the last checks were for pattern ED and delay = 4 or delay = 0. Without propagation, for 5, 10, 15 and 20 observables the last check were for pattern ED and delays 17, 20, 19 and 17 respectively, for 25 and 41 observables the last checks were for pattern ∃ED with RCP procedure.

To summarize, in all cases the propagation of results significantly improved the execution time of the integrated algorithm. In most cases, we would need to increase the time limit to obtain results for all patterns. This signifies that the propagation rules presented in this paper are important and useful in practice.

The analysis above clearly shows that algorithms for non-existential patterns (CP) are quicker than those for existential patterns (RCP/DRCP). This is because RCP/DRCP procedures use BDDs, which are in general not efficient, and we need to compute the fair states, which in the worst case means the

exploration of the whole state space.

The experiments do not show any clear correlation of the running time with the number of observables. Observable variables increase the size of the LTL formula but at the same time they reduce the reachable state space, because less states are observably equivalent. This may be a reason why results for some models form the bell curve: with more observables, running time increases, but with even more observables it starts to decrease. However, this trend is not universal for all models. In general, model checkers rely highly on heuristics, thus the exact strategy of the solver is hard to guess.

10. Related Work

10.1. Frameworks for Diagnosability

In this paper we follow [7, 8], where the patterns have been introduced and practically evaluated both in the synchronous setting [7] and in the asynchronous setting [8]. In these works, we studied diagnosis and diagnosability from an epistemic point-of-view and we provided a definition of diagnosability that is more fine-grained compared to previous work, that is, as a property of specific points of system traces. We also show how this can be generalized to whole traces and to sets of traces. Compared to [7, 8], we use a richer notion of diagnosability relative to an operational context Ψ.

The framework adopted in this paper is highly expressive: it encompasses fairness, temporally extended conditions to be diagnosed, various forms of delay, observability, and temporally extended diagnosis contexts to exclude unrealistic scenarios from the analysis. However, it does not consider either infinite-state plants, dense time, or active diagnosability. In this respect, the following works may be considered for future extensions.

With respect to state-space, note that the works in [3, 4, 43, 44, 5, 28, 45, 30, 46, 47, 48, 49, 50] rely on the hypothesis of finite-state plant. Systems with infinite state-spaces are considered in [51, 52, 53, 54, 55, 56, 57, 58], which work with dense-time models, and in [59, 60, 61], which deal with specific classes of infinite-state transition systems. It is worth noting that in some approaches a discrete abstraction is proposed to deal with hybrid automata. In [56], for instance, a hybrid automaton is abstracted to a purely discrete model in a way that preserves certain diagnosability properties of the original automaton. On the resulting abstraction then any algorithm for finite-state systems can be used.

In [62] the notion of diagnosability is generalized to open systems, where some actions are controllable and some uncontrollable. The proposed active diagnosability uses game structure and covers wider range of systems than classic diagnosability discussed in this work. To check the diagnosability, Alternating-Time Temporal Logic is used.

Diagnosability with context. Most of the works on diagnosability refer to the framework of [3], for instance [4, 43, 63, 45, 30, 59, 46, 47, 48, 61, 60, 49]. In these works, diagnosability requires detection of a fault event within a finite

number of steps; a counterexample is a pair of infinite traces where one contains the fault event, the other one does not, and both produce the same observations ad infinitum.

On one hand our frameworks differs in that it uses temporally extended fault models and has various notions of delay. Furthermore, the definition of diagnosability given in [3] might be stronger than necessary: imagine the situation where a few traces are not diagnosable because they contain an unrealistic sequence of inputs, e.g. from the environment. The presence of these traces breaks diagnosability. By using the concept of context, we can easily exclude these traces and better characterize the scenarios where we require diagnosability. Similarly, some unrealistic traces can be also excluded by adding fairness. While we consider fairness conditions on states, the authors in [64] discuss diagnosability on hybrid systems with fair transitions and in [46] both with fair and unfair transitions. We formally compare our framework to [3] in [8]. Essentially, the definition of diagnosability [3] corresponds, in our framework, to asking whether $\exists\textsc{BoundDel}_{\mathsf{O}}(\circ, f)$ is diagnosable, with f being a propositional fault variable.

A related approach is presented in [65], in the context of stochastic discrete event systems. Namely, probability of execution traces is used to estimate the degree of diagnosability and the probability distribution of the detection delay. In the same context, [66] investigates the problem of approximate diagnosability.

Temporally extended diagnosis conditions. Extensions of the framework of [3] that model faults as temporally extended properties instead of simple events are presented in [44, 67, 28, 7]. In the present framework we adopt the notion of diagnosis condition of [7].

In [67] the concept of K-diagnosability is introduced. It is an extension of the classical definition and requires an alarm to be raised after the k-th occurrence of a propositional fault f. In our framework this can be expressed by a corresponding Past-LTL formula counting the occurrences of f. The authors also describe specialized checking procedures.

The framework of [28] is more expressive and uses diagnosis conditions of the form $\mathsf{O}\beta$, where β is a safety property. A diagnosis condition is expressed as a supervision pattern, namely the language defined by a generic labeled transition system which is synchronously connected to the original system. The approach is less general than our fault models which use generic Past-LTL. It can for instance not verify whether a diagnoser will be able to generate the alarm for all occurrences of a temporary condition along any trace, just for fixed number of occurrences. A similar approach is presented in [68], where the authors model both the system and the diagnosis condition as an extended variant of Petri net. A pattern in this context is a word generated by the Petri net language. The approach is limited to finite delay diagnosability.

In [44] a faulty trace is a trace violating an LTL property with future operators. This framework is not comparable to ours. On one hand, LTL with future operators is more expressive than LTL with only past operators. However, the LTL formula is evaluated in [44] only at the first point of the trace

and not at every point of it as in our case. Consider the execution trace π and the LTL formula $G(\beta)$ and suppose β is false at step i, i.e., $\pi, i \not\models \beta$; then, $\pi, 0 \not\models G(\beta)$ independently from what follows after i; in [44], there is no distinction between different violations of β. Instead, our framework evaluates the diagnosis condition in every point of the execution.

An attempt to reason on properties of transient faults is presented in [69]. Here, the focus is on timely detection of a transient fault before it is repaired, and in counting of transient faults.

From Synchronous to Asynchronous Diagnoser Composition. We work in a synchronous setting, where the diagnoser is synchronously connected with the system under diagnosis. An asynchronous composition can be encoded into a synchronous one following the standard approach of inserting stuttering transitions where the plant "does not run", i.e., transitions that do not change the state of the plant. In this way, local transitions that do not change the observable variables cannot be distinguished by stuttering transitions. Thus, our framework can be extended to the asynchronous case; see for example how the synchronous definitions of [7] were extended to the asynchronous case in [8].

In the asynchronous setting, fault conditions can be non-diagnosable due to the absence of a global clock and the possibility to have unobservable loops. For instance, [63] investigates the problem of diagnosability of networked systems, where executions and events can only be partially ordered. It is therefore fundamental to have assumptions on the occurrence of transitions that are observable. This can be easily encoded in our framework with an explicit formalization of the fairness, while in other approaches it is added at a meta level (see, e.g., assumption on absence of unobservable loops in [3]).

10.2. Verification of Diagnosability

Verification of diagnosability has been studied in a variety of works. In [3] the classical definition of diagnosability for DES and an algorithm to check it are given, based on building a diagnoser for the system under observation. The diagnoser contains cycles of ambiguous belief states if and only if the condition is not diagnosable.

Since the diagnoser method is exponential in the number of system states, in [4] the twin plant approach is introduced as a more efficient alternative that is polynomial in the number of states. The key idea is to avoid building the diagnoser and instead only compare pairs of indistinguishable infinite traces by exploring the twin plant. The system is diagnosable iff no such pair exists. In [43] a similar method is described. This twin plant based approach has become the standard way to study diagnosability. Variations of the method have been studied in [28, 70, 30, 59, 48, 61, 60].

In order to avoid using ad-hoc constructions as in the methods above, in [44] the problem of diagnosability is reduced to LTL model-checking. This strategy is also adopted in [5], which furthermore uses symbolic techniques to address the problem of state explosion; this is the general approach that we also adopt in the present paper and validate empirically. Another symbolic approach to

diagnosability to deal with state explosion, based on the twin plant, is also advocated in [45], which proposes a SAT encoding for falsification of diagnosability, similar to bounded model-checking. As an optimization, this work exploits the notion of *succinct transition system*, i.e. it puts some constraints on the representation of formulae and the transition relation, in order to identify actions that are independent and hence can be executed simultaneously.

In [30] two specialized algorithms, based on fix-point computation and forward/backward search are proposed; the backward-directed algorithm starts from an ambiguous state and tries to reach an initial state. The intuition here is that when a fault occurs and it quickly produces observable symptoms that cause divergence of traces, a backward search should reach a fix-point much sooner than a corresponding forward search, thus saving computation time. The same idea is reformulated in [49].

Compositional approaches that aim at dealing with the complexity of bigger systems are described in [71, 47], where the results of local twin plant checks for individual components are integrated to infer the global diagnosability property. In comparison, we adopt a monolithic approach and try to address state-explosion via symbolic model-checking.

In this paper we have shown that when increasing the expressiveness of the diagnosability framework, the standard method based on critical pairs cannot be used in all cases as a verification method (though it always works for falsification). In this respect, we provided novel decision procedures for problems that were not previously solved.

Diagnosability via Temporal-Epistemic Logic. Epistemic logic has been used to describe and reason about knowledge of agents and processes. There are several ways of extending epistemic logic with temporal operators, and in [7] the logic KL_1 [72] is used to reason about an ideal diagnoser. KL_1 extends LTL with the epistemic operator K. The intuitive semantics of $\mathsf{K}\beta$ is that the diagnoser *knows*, by only looking at the observable behavior of the system, that β holds in the current execution. The definition of the semantics of K can take into account multiple aspects such as observability, synchronicity and memory. A semantics for ASL based on Temporal Epistemic Logic (TEL) has been described in [8], providing a sound and complete technique for performing diagnosability testing using Temporal Epistemic logic model-checking. For example, the diagnosability test for $\textsc{ExactDel}(A, \beta, n)$ consists of the KL_1 formula $G(\beta \rightarrow X^n K Y^n \beta)$, stating that whenever β occurs, exactly n steps afterwards, the diagnoser *knows* that n steps before β occurred. Since K is defined on observationally equivalent traces, the only way to falsify the formula would be to have a trace in which β occurs, and another one (observationally equivalent at least for the next n steps) in which β did not occur; but this is in contradiction with the definition of diagnosability as given in Definition 3.

In many practical situations for which the critical pairs approach is complete for diagnosability verification, the twin plant construction presents several advantages over the TEL encoding. First, the level of maturity of tools and techniques for LTL model-checking is significantly higher than for TEL. TEL

model-checking routines are mostly lacking, and when available support only some particular semantics [33], or solve a problem that is as hard as building the diagnoser for the system, thus defeating the purpose of performing the diagnosability test as a validation step. Second, encoding the concept of context in the specification is much more difficult in the TEL framework where the context must be embedded within the plant model, in order to provide a different interpretation of the K operator.

Dynamic and uncertain observations. Instead of operating with a static set of observables, in [73] and [74] it is assumed that sensors can be enabled and disabled during execution of the plant to save operating costs. The goal is to minimize the number of active sensors at any time during the plant's run, and turn on only those that are strictly necessary to preserve diagnosability of the fault event. In the present paper we only consider a static choice of observations (sensors are always switched on).

Diagnosability w.r.t. masks over observable events is a way to model uncertainty in observations and is discussed in various works [75, 73, 76]. In [76] these masks are described as *logically uncertain observations*, i.e. observations that cannot be discriminated. The authors describe also the class of *temporally uncertain observations*, that is when the order in which observable events were generated is not clear.

In our framework the observations are regarded as certain, but to some extent these uncertainties can be modeled also here. To realize *logically uncertain observations* one could introduce new observable signals that are functions over the original observables plus possibly an uncertainty factor such as sensor faults or amount of signal noise. Also *temporally uncertain observations* can in principle be modeled in our framework by using buffers that non-deterministically delay the output signal.

Distinguishing sets of states. The framework in [5] deals with finite-state systems, and expresses the diagnosis condition as a non-temporal pair of conditions to be separated with delay 0; in [77] and [50] it is extended with different notions of delay. This approach is incomparable with the choices made in the present paper. On one hand, by using temporal specifications we can obtain richer fault models, and we also have several ways to express delay requirements; but then again we do not express diagnosability as being able to distinguish a *pair* of properties. The focus here is on the diagnoser that one will eventually build, and thus on the condition for which it will need to generate an alarm. The approach chosen in this paper was found to be simpler and yet adequate in practical cases. The xSAP and COMPASS toolsets for instance are based – as far as diagnosability is concerned – on the theoretical framework expressed in this paper. The same motivation applies also to use an LTL context referring to the traces of the original plant, as opposed to referring directly to twin plant traces as done in [5].

11. Conclusions and Future Work

In this paper we propose a comprehensive approach to the problems of verification of diagnosability under fairness conditions. We adopt an expressive, logic-based framework that takes into account temporally-extended diagnosis conditions, various forms of delay in the diagnosability, and the operating conditions.

We make three key contributions. First, we analyze the relationship between diagnosability and the existence of critical pairs, which underlies the standard approaches to diagnosability verification. We show that, quite surprisingly, the absence of critical pairs is necessary for diagnosability, but not always sufficient. We extend the framework by introducing the notions of critical set and ribbon-shape critical pair.

Second, we propose a comprehensive approach to diagnosability verification, that analyzes the diagnosability status of a given diagnosis condition for different patterns and varying delays. The algorithm orchestrates multiple calls to the twin plant construction and its extensions, according to several theoretical results relating the absence of critical pairs and critical sets to diagnosability.

Third, we implement the approach and carry out a comprehensive experimental evaluation based on realistic models. The results demonstrate the practical applicability of the proposed algorithms to a variety of problems.

As opposed to most related works, we rely on symbolic LTL model-checking as a way to identify critical pairs and prove the absence thereof. In this way, our approach can automatically benefit from any advancement made in this field.

Future work. There are many promising directions for future work.

First, we are investigating ways to strengthen the verification algorithm. We are designing more efficient algorithms for finding ribbon-shaped paths, that do not rely on BDDs and CTL, but reduce the problem to safety verification [78]. We will also consider the adoption of counter-example guided abstraction-refinement [79] to dynamically identify other sufficient conditions for diagnosability (e.g. using a weaker safety context Ψ' such that $\Psi \subset \Psi'$).

Second, we will consider extending the proposed approach beyond the case of finite-state fair transition systems. Relevant extensions include infinite-state transition systems, that are amenable to the use of SMT-based model checking, and timed/hybrid systems, that integrate continuous dynamics [80] and a logic such as HRELTL to express requirements over continuous traces [81]. We will adopt some recent SMT-based techniques, that are becoming relevant for the analysis of networks of hybrid automata [82, 83, 84] and possibly counter-example guided abstraction refinement [85].

Finally, we want to investigate the problem of synthesis for diagnosability, i.e. finding subsets of the available sensors that are sufficient to guarantee diagnosability, and possibly minimize some cost function, see e.g. [86, 75, 30, 77, 87].

Acknowledgements

We are indebted to Charles Pecheur for the initial work on diagnosability. Roberto Cavada implemented the first version of the experiments. The first author was partially supported by the European Space Agency NPI contract No. 4000111815/14/NL/FE.

References

[1] R. Reiter, A Theory of Diagnosis from First Principles, Artificial Intelligence 32 (1987) 57–95.

[2] X. Olive, FDI(R) for Satellites: How to Deal With High Availability and Robustness in the Space Domain?, International Journal of Applied Mathematics and Computer Science 22 (1) (2012) 99–107.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of Discrete-event Systems, IEEE Transactions on Automatic Control 40 (9) (1995) 1555–1575.

[4] S. Jiang, Z. Huang, V. Chandra, R. Kumar, A Polynomial-time Algorithm for Diagnosability of Discrete Event Systems, IEEE Transactions on Automatic Control 46 (8) (2001) 1318–1321.

[5] A. Cimatti, C. Pecheur, R. Cavada, Formal Verification of Diagnosability via Symbolic Model-Checking, in: International Joint Conference on Artificial Intelligence, 2003, pp. 363–369.

[6] M. Bozzano, A. Cimatti, M. Gario, S. Tonetta, A Formal Framework for the Specification, Verification and Synthesis of Diagnosers, in: AAAI (Late-Breaking Developments), 2013.

[7] M. Bozzano, A. Cimatti, M. Gario, S. Tonetta, Formal Design of Fault Detection and Identification Components Using Temporal Epistemic Logic, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2014, pp. 326–340.

[8] M. Bozzano, A. Cimatti, M. Gario, S. Tonetta, Formal Design of Asynchronous Fault Detection and Identification Components using Temporal Epistemic Logic, Logical Methods in Computer Science 11 (4) (2015). doi:10.2168/LMCS-11(4:4)2015.
URL https://doi.org/10.2168/LMCS-11(4:4)2015

[9] B. Bittner, M. Bozzano, A. Cimatti, R. D. Ferluc, M. Gario, A. Guiotto, Y. Yushtein, An integrated process for FDIR design in aerospace, in: F. Ortmeier, A. Rauzy (Eds.), IMBSA, Vol. 8822 of Lecture Notes in Computer Science, Springer, 2014, pp. 82–95. doi:10.1007/978-3-319-12214-4_7.
URL https://doi.org/10.1007/978-3-319-12214-4_7

[10] E. Emerson, Temporal and Modal Logic, Handbook of theoretical computer science 2 (1990) 995–1072.

[11] F. Laroussinie, N. Markey, P. Schnoebelen, Temporal Logic with Forgettable Past, in: Symposium on Logic in Computer Science, 2002, pp. 383–392.

[12] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli, G. Zampedri, The xSAP Safety Analysis Platform, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2016, pp. 533–539.

[13] M. Bozzano, A. Cimatti, A. F. Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, S. Tonetta, Formal Design and Safety Analysis of AIR6110 Wheel Brake System, in: International Conference on Computer Aided Verification, Springer, 2015, pp. 518–535.

[14] C. Mattarei, A. Cimatti, M. Gario, S. Tonetta, K. Y. Rozier, Comparing different functional allocations in automated air traffic control design, in: R. Kaivola, T. Wahl (Eds.), FMCAD, IEEE, 2015, pp. 112–119.

[15] M. Gario, A. Cimatti, C. Mattarei, S. Tonetta, K. Y. Rozier, Model checking at scale: Automated air traffic control design space exploration, in: S. Chaudhuri, A. Farzan (Eds.), CAV, Vol. 9780 of Lecture Notes in Computer Science, Springer, 2016, pp. 3–22. doi:10.1007/978-3-319-41540-6_1.
URL https://doi.org/10.1007/978-3-319-41540-6_1

[16] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, M. Roveri, The COMPASS Approach: Correctness, Modelling and Performability of Aerospace Systems, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2009, pp. 173–186.

[17] M. Bozzano and A. Cimatti and J.-P. Katoen and P. Katsaros and K. Mokos and V.Y. Nguyen and T. Noll and B. Postma and M. Roveri, Spacecraft Early Design Validation using Formal Methods, Reliability Engineering and System Safety 132 (2014) 20–35.

[18] European Space Agency, AO/1-5458/07/NL/JD, "System-Software Co-Engineering: Performance and Verification" (2007).

[19] European Space Agency, ESTEC ITT AO/1-6570/10/NL/LvH "Dependability Design Approach for Critical Flight Software" (2010).

[20] European Space Agency, ESTEC ITT AO/1-6992/11/NL/JK "FDIR Development and Verification & Validation Process" (2011).

[21] European Space Agency, ESTEC ITT AO/1-7263/12/NL/AK "Hardware-Software Dependability for Launchers" (2013).

[22] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, S. Tonetta, The nuXmv Symbolic Model Checker, in: CAV, Vol. 8559 of Lecture Notes in Computer Science, Springer, 2014, pp. 334–342.

[23] E. M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 1999.

[24] D. van Dalen, Logic and structure (3. ed.), Universitext, Springer, 1994.

[25] NuSMV 2.6 User Manual, `http://nusmv.fbk.eu/NuSMV/userman/index-v2.html` (2010).

[26] E. M. Clarke, O. Grumberg, K. Hamaguchi, Another Look at LTL Model Checking, Formal Methods in System Design 10 (1) (1997) 47–71.

[27] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV2: An Open-source Tool for Symbolic Model-checking, in: Computer Aided Verification, Springer, 2002, pp. 241–268.

[28] T. Jéron, H. Marchand, S. Pinchinat, M.-O. Cordier, Supervision Patterns in Discrete Event Systems Diagnosis, in: International Workshop on Discrete Event Systems, IEEE, 2006, pp. 262–268.

[29] M. Bozzano and A. Cimatti and S. Tonetta, Testing Diagnosability of Fair Discrete-Event Systems, in: Proc. International Workshop on Principles of Diagnosis (DX-19), 2019.

[30] A. Grastien, Symbolic testing of diagnosability, in: 20th International Workshop on Principles of Diagnosis (DX-09), 2009, pp. 131–138.

[31] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, C. Sánchez, Temporal Logics for Hyperproperties, in: POST, 2014, pp. 265–284. `doi:10.1007/978-3-642-54792-8\_15`.
URL `https://doi.org/10.1007/978-3-642-54792-8_15`

[32] J. Y. Halpern, M. Y. Vardi, The Complexity of Reasoning about Knowledge and Time. I. Lower Bounds, J. Comput. Syst. Sci. 38 (1) (1989) 195–237.

[33] A. Cimatti, M. Gario, S. Tonetta, A lazy approach to temporal epistemic logic model checking, in: International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1218–1226.

[34] O. Lichtenstein, A. Pnueli, Checking That Finite State Concurrent Programs Satisfy Their Linear Specification, in: POPL, ACM Press, 1985, pp. 97–107.

[35] E. Allen Emerson, C.-L. Lei, Temporal reasoning under generalized fairness constraints, in: B. Monien, G. Vidal-Naquet (Eds.), STACS 86, Springer Berlin Heidelberg, Berlin, Heidelberg, 1986, pp. 21–36.

[36] E. A. Emerson, C. Lei, Modalities for model checking: Branching time logic strikes back, Sci. Comput. Program. 8 (3) (1987) 275–306. doi:10.1016/0167-6423(87)90036-0.
URL https://doi.org/10.1016/0167-6423(87)90036-0

[37] O. Kupferman, M. Y. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, J. ACM 47 (2) (2000) 312–360. doi:10.1145/333979.333987.
URL https://doi.org/10.1145/333979.333987

[38] A. Cimatti, A. Griggio, B. J. Schaafsma, R. Sebastiani, The MathSAT5 SMT Solver, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2013, pp. 93–107.

[39] COMPASS Consortium. COMPASS Project Web Page: www.compass-toolset.org [online]. Accessed: 2017-03-15.

[40] B. C. Williams, P. P. Nayak, A Model-based Approach to Reactive Self-configuring Systems, in: National Conference on Artificial Intelligence, 1996, pp. 971–978.

[41] M. Bozzano, A. Cimatti, F. Tapparo, Symbolic Fault Tree Analysis for Reactive Systems, in: International Symposium on Automated Technology for Verification and Analysis, 2007, pp. 162–176.

[42] A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, J. de Kleer, L. Kuhn, A. van Gemund, Empirical Evaluation of Diagnostic Algorithm Performance Using a Generic Framework, International Journal of Prognostics and Health Management 1 (2010) 24–51.

[43] T.-S. Yoo, S. Lafortune, Polynomial-time Verification of Diagnosability of Partially Observed Discrete Event Systems, IEEE Transactions on Automatic Control 47 (9) (2002) 1491–1495.

[44] S. Jiang, R. Kumar, Failure Diagnosis of Discrete Event Systems with Linear-Time Temporal Logic Fault Specifications, in: American Control Conference, Vol. 1, IEEE, 2002, pp. 128–133.

[45] J. Rintanen, A. Grastien, Diagnosability testing with satisfiability algorithms, in: International Joint Conference on Artificial Intelligence, 2007, pp. 532–537.

[46] S. Biswas, D. Sarkar, S. Mukhopadhyay, A. Patra, Fairness of Transitions in Diagnosability of Discrete Event Systems, Discrete Event Dynamic Systems 20 (3) (2010) 349–376.

[47] L. Ye, P. Dague, An Optimized Algorithm for Diagnosability of Component-based Systems, in: International Workshop on Discrete Event Systems, 2010, pp. 143–148.

[48] A. Madalinski, F. Nouioua, P. Dague, Diagnosability Verification with Petri Net Unfoldings, International Journal of Knowledge-Based and Intelligent Engineering Systems 14 (2) (2010) 49–55.

[49] B. Li, T. Guo, X. Zhu, Z. Li, Reverse Twin Plant for Efficient Diagnosability Testing and Optimizing, Engineering Applications of Artificial Intelligence 38 (2015) 131–137.

[50] A. Boussif, M. Ghazel, Diagnosability Analysis of Input/Output Discrete-Event Systems Using Model-Checking, IFAC-PapersOnLine 48 (7) (2015) 71–78.

[51] S. Tripakis, Fault Diagnosis for Timed Automata, in: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer, 2002, pp. 205–221.

[52] K. Altisen, F. Cassez, S. Tripakis, Monitoring and Fault-Diagnosis with Digital Clocks, in: International Conference on Application of Concurrency to System Design, 2006, pp. 101–110.

[53] S. Xu, S. Jiang, R. Kumar, Diagnosis of Dense-time Systems Under Event and Timing Masks, IEEE Transactions on Automation Science and Engineering 7 (4) (2010) 870–878.

[54] S. Biswas, D. Sarkar, S. Mukhopadhyay, A. Patra, Diagnosability Analysis of Real Time Hybrid Systems, in: IEEE International Conference on Industrial Technology, IEEE, 2006, pp. 104–109.

[55] M. Daigle, X. Koutsoukos, G. Biswas, An Event-based Approach to Integrated Parametric and Discrete Fault Diagnosis in Hybrid Systems, Transactions of the Institute of Measurement and Control, Special Issue on Hybrid and Switched Systems 32 (5) (2010) 487–510.

[56] M. Bayoudh, L. Travé-Massuyès, Diagnosability Analysis of Hybrid Systems Cast in a Discrete Event Framework, Discrete Event Dynamic Systems 24 (3) (2014) 309–338.

[57] D. Bresolin, M. Capiluppi, A Game-theoretic Approach to Fault Diagnosis and Identification of Hybrid Systems, Theoretical Computer Science 493 (2013) 15–29.

[58] M. D. Di Benedetto, S. Di Gennaro, A. D'Innocenzo, Verification of Hybrid Automata Diagnosability by Abstraction, IEEE Transactions on Automatic Control 56 (9) (2011) 2050–2061.

[59] C. Morvan, S. Pinchinat, Diagnosability of Pushdown Systems, in: Haifa Verification Conference, Springer, 2009, pp. 21–33.

[60] S. Chédor, C. Morvan, S. Pinchinat, H. Marchand, Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems, Discrete Event Dynamic Systems 25 (1-2) (2015) 271–294. doi:10.1007/s10626-014-0197-3.
URL https://doi.org/10.1007/s10626-014-0197-3

[61] M. P. Cabasino, A. Giua, S. Lafortune, C. Seatzu, A New Approach for Diagnosability Analysis of Petri Nets Using Verifier Nets, IEEE Transactions on Automatic Control 57 (12) (2012) 3104–3117.

[62] T. Melliti, P. Dague, Generalizing diagnosability definition and checking for open systems: a game structure approach, in: 21st International Workshop on Principles of Diagnosis (DX'10), 2010.

[63] S. Haar, A. Benveniste, E. Fabre, C. Jard, Partial order diagnosability of discrete event systems using petri net unfoldings, in: 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475), Vol. 4, 2003, pp. 3748–3753 vol.4. doi:10.1109/CDC.2003.1271732.

[64] S. Biswas, D. Sarkar, S. Mukhopadhyay, A. Patra, Diagnosability of fair discrete event systems, Asian Journal of Control 10 (2008) 651–665.

[65] H. Bazille, E. Fabre, B. Genest, Diagnosability degree of stochastic discrete event systems, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 2017, pp. 5726–5731. doi:10.1109/CDC.2017.8264524.

[66] N. Bertrand, S. Haddad, E. Lefaucheux, Accurate Approximate Diagnosability of Stochastic Systems, in: Proc. Language and Automata Theory and Applications (LATA 2016), Vol. 9618 of LNCS, Springer, 2016, pp. 549–561.

[67] S. Jiang, R. Kumar, H. E. Garcia, Diagnosis of Repeated/Intermittent Failures in Discrete Event Systems, IEEE Transactions on Robotics and Automation 19 (2) (2003) 310–323.

[68] H. Gougam, Y. Pencolé, A. Subias, Diagnosability analysis of patterns on bounded labeled prioritized petri nets, Discrete Event Dynamic Systems 27 (1) (2017) 143–180. doi:10.1007/s10626-016-0234-5.
URL https://doi.org/10.1007/s10626-016-0234-5

[69] E. Fabre, L. Hélouët, E. Lefaucheux, H. Marchand, Diagnosability of repairable faults, in: 2016 13th International Workshop on Discrete Event Systems (WODES), 2016, pp. 230–236. doi:10.1109/WODES.2016.7497853.

[70] M. P. Cabasino, A. Giua, C. Seatzu, Diagnosability of Bounded Petri Nets, in: IEEE Conference on Decision and Control, IEEE, 2009, pp. 1254–1260.

[71] A. Schumann, Y. Pencolé, Scalable diagnosability checking of event-driven systems, in: International Joint Conference on Artificial Intelligence, 2007, pp. 575–580.

[72] J. Y. Halpern, M. Y. Vardi, The complexity of Reasoning About Knowledge and Time. Lower Bounds, Journal of Computer and System Sciences 38 (1) (1989) 195–237.

[73] F. Cassez, S. Tripakis, K. Altisen, Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis, in: International Conference on Application of Concurrency to System Design, IEEE, 2007, pp. 90–99.

[74] W. Wang, S. Lafortune, F. Lin, Optimal Sensor Activation in Controlled Discrete Event Systems, in: IEEE Conference on Decision and Control, IEEE, 2008, pp. 877–882.

[75] S. Jiang, R. Kumar, H. E. Garcia, Optimal Sensor Selection for Discrete-event Systems with Partial Observation, IEEE Transactions on Automatic Control 48 (3) (2003) 369–381.

[76] X. Su, M. Zanella, A. Grastien, Diagnosability of Discrete-Event Systems with Uncertain Observations, in: International Joint Conference on Artificial Intelligence, 2016, pp. 1265–1271.

[77] B. Bittner, M. Bozzano, A. Cimatti, X. Olive, Symbolic Synthesis of Observability Requirements for Diagnosability, in: AAAI Conference on Artificial Intelligence, 2012, pp. 712–718.

[78] M. Bozzano, A. Cimatti, S. Tonetta, V. Vozarova, Searching for Ribbon-Shaped Paths in Fair Transition Systems, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2022, to appear.

[79] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement for symbolic model checking, J. ACM 50 (5) (2003) 752–794.

[80] T. A. Henzinger, The Theory of Hybrid Automata, in: Verification of Digital and Hybrid Systems, Springer, 2000, pp. 265–292.

[81] A. Cimatti, M. Roveri, S. Tonetta, HRELTL: A Temporal Logic for Hybrid Systems, Information and Computation 245 (2015) 54–71.

[82] A. Cimatti, S. Mover, S. Tonetta, SMT-Based Verification of Hybrid Systems, in: AAAI Conference on Artificial Intelligence, 2012, pp. 2100–2105.

[83] S. Mover, A. Cimatti, A. Tiwari, S. Tonetta, Time-aware Relational Abstractions for Hybrid Systems, in: International Conference on Embedded Software, IEEE, 2013, pp. 1–10.

[84] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, HyComp: An SMT-based Model-checker for Hybrid Systems, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2015, pp. 52–67.

[85] H. Zaatiti, L. Ye, P. Dague, J.-P. Gallois, Counterexample-guided abstraction-refinement for hybrid systems diagnosability analysis, in: M. Zanella, I. Pill, A. Cimatti (Eds.), 28th International Workshop on Principles of Diagnosis (DX'17), Vol. 4 of Kalpa Publications in Computing, EasyChair, 2018, pp. 124–143. doi:10.29007/t8n3.

[86] R. Debouk, S. Lafortune, D. Teneketzis, On an Optimization Problem in Sensor Selection, Discrete Event Dynamic Systems 12 (4) (2002) 417–445.

[87] B. Bittner, M. Bozzano, A. Cimatti, M. Gario, A. Griggio, Towards Pareto-optimal Parameter Synthesis for Monotonic Cost Functions, in: FMCAD, 2014, pp. 23–30.