A Formal Framework for the Specification, Verification and Synthesis of Diagnosers

Marco Bozzano and **Alessandro Cimatti** and **Marco Gario** and **Stefano Tonetta**
Fondazione Bruno Kessler
Trento, Italy

Abstract

In this work we present a formal approach to the design of Fault Detection and Identification (FDI) components. We define a comprehensive language for the specification of FDI, and discuss how to check whether a given FDI component fulfills its specification. Then, we propose an automatic procedure to synthesize an FDI component that satisfies a given specification. The approach has been implemented and tested in realistic case studies from the aerospace domain.

Introduction

The correct operation of complex critical systems (e.g. trains, satellites, cars, or industrial plants) increasingly relies on the ability to detect when and which faults occur, since an effective diagnostic system can provide vital information to drive the containment of faults and their recovery. This task, known as Fault Detection and Identification (FDI), is typically carried out by an FDI component that triggers a set of predefined alarms based on the sequence of values conveyed by some predefined observables.

Faults can manifest in different ways and they might interact with each other in complex ways. For these reasons, the design of FDI components is an extremely hard task. The state of the practice lacks a structured and effective methodology to design FDI components; moreover, the lack of effective validation tools often results in sub-optimal systems.

In this paper we introduce a tool-supported, formal framework for the design of FDI components. Our work is motivated by the strong interest of the aerospace sector (see for instance (European Space Agency 2010)). We provide domain experts with an expressive framework covering the situations that they want to monitor. Towards this goal, we make three main contributions. First, we propose a *formal language* for the specification of FDI requirements to define the alarms of interests in terms of the unobservable state. Second, we discuss how to verify that a given FDI component satisfies a set of requirements. Finally, we develop an algorithm for the automated synthesis of correct-by-construction FDI components.

Previous work on online diagnoser synthesis has been focusing on rather simple conditions for detection (e.g., unobservable events (Sampath et al. 1995)) and on technical improvements for scalability of the algorithms (Schumann 2004). Instead, we focus on providing an expressive specification language for the diagnoser. This makes the synthesis problem much harder but also provides a more interesting diagnoser that is able, e.g., to raise multiple alarms simultaneously.

We implemented the proposed framework on top of the NuSMV model checker, with Binary Decision Diagrams providing an effective basis to represent belief states. The approach has been experimentally evaluated on industrial case-studies, and the results show that the synthesis of diagnosers is feasible for models of practical interest.

Background

Plants and FDIs are represented with transition systems. A transition system is a tuple $S = \langle V, V_o, W, W_o, I, T\rangle$, where V is the set of state variables, $V_o \subseteq V$ is the set of observable state variables; W is the set of input variables, $W_o \subseteq W$ is the set of observable input variables; I is the initial formula over V; T is the transition formula over V, W, V' (V' being the next'd version of the state variables). A state s (resp. s') is an assignment to the state variables V (V'). An input i is an assignment to the input variables W. The observable part $obs(s)$ of a state s is the projection of s on the subset V_o of observable state variables and, similarly, $obs(i)$ is the projection of i on the subset W_o of observable input variables. A trace of S is a sequence $\pi = s_0, i_1, \ldots, s_n, i_{n+1}, s_{n+1}$ of states and inputs such that s_0 satisfies I and for each k, $0 \leq k \leq n$, $\langle s_k, i_{k+1}, s_{k+1}\rangle$ satisfies T. The observable part of a trace π is $obs(\pi) = obs(s_0), obs(i_1), \ldots, obs(s_n), obs(i_{n+1}), obs(s_{n+1})$.

Given two transition systems $S^1 = \langle V^1, V^1_o, W^1, W^1_o, I^1, T^1\rangle$ and $S^2 = \langle V^2, V^2_o, W^2, W^2_o, I^2, T^2\rangle$ with $(V^1 \setminus V^1_o) \cap (V^2 \setminus V^2_o) = \emptyset$ and $(W^1 \setminus W^1_o) \cap (W^2 \setminus W^2_o) = \emptyset$, we define the synchronous product $S^1 \times S^2$ as the transition system $\langle V^1 \cup V^2, V^1_o \cup V^2_o, W^1 \cup W^2, W^1_o \cup W^2_o, I^1 \wedge I^2, T^1 \wedge T^2\rangle$. Only observable variable can be shared among the two systems and used to perform synchronization. This gives raise to the problem of partial observability: the FDI cannot perfectly track the evolution of the original system. This makes the FDI synthesis problem hard.

In the following we use the word *system* to indicate the composition of the plant and the FDI component; we also use *diagnoser* and *FDI component* interchangeably.

Diagnoser Specification and Verification

Our approach relies on Linear Temporal Logic with past operators (Lichtenstein, Pnueli, and Zuck 1985). A *diagnosis condition* β is a situation of interest described by using LTL restricted to the past: $\beta ::= p \mid \beta \wedge \beta \mid \neg\beta \mid O\beta \mid Y\beta$, with $p \in V \cup W$. We use two abbreviations: $Y^n\phi = YY^{n-1}\phi$ (with $Y^0\phi = \phi$) and $O^{\leq n}\phi = \phi \vee Y\phi \vee \cdots \vee Y^n\phi$.

We now define a language to specify how the occurrence of a diagnosis condition β determines the raising of an alarm A. Interaction with industrial experts led us to identify three patterns of alarms, summarized in the following table:

Template	Formalization	..B...B....B.....B...
EXACTDEL(A, β, n)	$G(\beta \leftrightarrow X^n A)$	....A...A....A.....A.
BOUNDDEL(A, β, n)	$G(\beta \rightarrow F^{\leq n} A \wedge$ $A \rightarrow O^{\leq n}\beta)$	.....A..AA.AA.A.....A
FINITEDEL(A, β)	$G(\beta \rightarrow FA \wedge$ $A \rightarrow O\beta)$	..........A.........A.

EXACTDEL(A, β, n) states that A is triggered exactly n steps after every occurrence of β. BOUNDDEL(A, β, n) allows for a delay between the occurrence of β and the occurrence of A. Finally, finite-delay is of theoretical interest since it captures the idea of diagnosability as defined in previous works (Rintanen and Grastien 2007). The second column presents the LTL formula that defines the relation between the diagnosis condition β and the alarm A. The third column shows an example trace for β (B in the first row) and an admissible response for EXACTDEL$(\mathtt{A}, \beta, 2)$, BOUNDDEL$(\mathtt{A}, \beta, 4)$ and FINITEDEL$(\mathtt{A}, \beta)$.

Each alarm specification can be validated against the given plant in order to understand whether an *ideal* diagnoser would be able to satisfy the specification. This step can be performed by running a diagnosability test (Cimatti, Pecheur, and Cavada 2003) for each alarm specification. We call the subset of alarm specifications that are diagnosable for the plant P the *diagnosable specification* ($\Phi_P \subseteq \Phi$).

Given a plant transition system $P = \langle V^P, V_o^P, W^P, W_o^P, I^P, T^P \rangle$ and a specification Φ_P, we can define a diagnoser D for P as a transition system $D = \langle V^D, V_o^D, W^D, W_o^D, I^D, T^D \rangle$ such that: $V_o^P \subseteq V_o^D$ and $W_o^P \subseteq W_o^D$; for every trace π^P of P, there exists a trace π^D of D such that $obs(\pi^P) \subseteq obs(\pi^D)$; finally, $\{A_\phi\}_{\phi \in \Phi} \subseteq V_o^D$.

Let D be a diagnoser for P. We say that D satisfies a diagnosable specification Φ_P iff for every requirement $\varphi \in \Phi_P$: $D \times P \models_{LTL} \varphi$. This property can be checked with classic model-checking techniques (Clarke, Grumberg, and Peled 2001).

FDI synthesis: Belief Explorer

The idea of the Belief Explorer algorithm is to generate an automaton that (once synchronized with the plant) keeps track of the possible states in which the plant could be after the given observations. This translates into generating the power-set of the states of the plant, and defining a suitable transition relation among the elements of this set. We call the elements in the above power-set defined in this way *belief states*, borrowing from epistemic logic and work on planning under partial observability (Bertoli et al. 2001). Each belief state of the automaton can be annotated with the alarms that are satisfied in all the states of the belief state. We can then encode the automaton with the annotations symbolically, obtaining the diagnoser.

Given a plant P, the *belief automaton* is defined as $\mathcal{B}(P) = \langle B, B_0, T_b \rangle$ with $B = 2^{2^{V^P}}$, $B_0 \subseteq B$ and $T_b : B \times (2^{W_o^P}, 2^{V_o^P}) \rightarrow B$. B_0 is the set of initial states and it is obtained from partitioning the set of initial states of the plant. Let $S_0|_u$ be the operation restricting the set of states S_0 to only those states compatible with the observation u: $S_0|_u = \{s \mid s \in S_0 \wedge obs(s) = u\}$. We define $B_0 = \{b \mid \exists u \in 2^{V_o}.b = S_0|_u \wedge b \neq \emptyset\}$, meaning that each initial belief state is compatible with one of the possible observations of the system. We implicitly assume that we can initialize the diagnoser by observing the state of the system. The transition relation is defined as $b' = T_b(b, e, u) = \{s' \mid \exists s \in b.T_p(s, i, s') \wedge obs(s') = u \wedge obs(i) = e\}$. Intuitively, the belief state b' is a successor of b iff all the states in b' are compatible with the observations from a state in b.

To obtain the diagnoser, we need to annotate each state of the belief automaton with the corresponding alarms. This can be done by testing each belief state for entailment of the alarm specification φ. To explain this procedure we first consider the simplest case $\varphi = \text{EXACTDEL}(A_\varphi, p, 0)$, where p is a propositional variable. We can explore the belief automaton, and annotate with A_φ all the belief states in which all the plant states satisfy φ: $b \models A_\varphi$ iff $b \models \varphi$ iff $\forall s \in b.s \models \varphi$. The handling of all other alarms specifications can be reduced to the previous case by performing a preprocessing step to the belief automaton construction. For each alarm specification φ we add a history variable $\overline{\varphi}$ in the plant, s.t. the axiom $G(\varphi \leftrightarrow \overline{\varphi})$ holds. All the alarm specifications can thus be redefined as: $\varphi' = \text{EXACTDEL}(A_\varphi, \overline{\varphi}, 0)$.

Implementation and Experiments

We developed full support for the proposed framework on top of the NuSMV model checker (Cimatti et al. 2002), making it possible to specify, synthesize and verify a diagnoser. The Belief Explorer synthesis procedure has been implemented by using BDD's (Bryant 1986) to represent belief states. This enables us to represent each belief state with a single and unique BDD, making it easy to identify already visited belief states. Moreover, the symbolic representation of belief states allows us to check for entailment ($b \models \varphi$) with a single operation.

We experimentally evaluated our work in two different settings: an aerospace case-study, and a set of randomly generated benchmarks. Due to the space limitation, we can not enter in detail about the results (nor discuss future work). Here suffices to say that the synthesis procedure can handle a synthesis problem specified by industrial partners with a significant number of requirements within a runtime of the order of seconds.

References

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observ-

ability via symbolic model checking. In Nebel, B., ed., *IJCAI*, 473–478. Morgan Kaufmann.

Bryant, R. 1986. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on* 100(8):677–691.

Cimatti, A.; Clarke, E. M.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *CAV*, 359–364.

Cimatti, A.; Pecheur, C.; and Cavada, R. 2003. Formal Verification of Diagnosability via Symbolic Model Checking. *International Joint Conference on Artificial Intelligence* 363-369.

Clarke, E. M.; Grumberg, O.; and Peled, D. 2001. *Model checking*. MIT Press.

European Space Agency. 2010. ITT AO/1-6570/10/NL/LvH "Dependability Design Approach for Critical Flight Software". Technical report.

Lichtenstein, O.; Pnueli, A.; and Zuck, L. 1985. The glory of the past. In Parikh, R., ed., *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 196–218.

Rintanen, J., and Grastien, A. 2007. Diagnosability testing with satisfiability algorithms. In Veloso, M. M., ed., *IJCAI*, 532–537.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Tneketzis, D. 1995. Diagnosibility of Discrete-Event Systems. *IEEE Transactions on automatic control* 40(9):1555–1576.

Schumann, A. 2004. Diagnosis of discrete-event systems using binary decision diagrams. *on Principles of Diagnosis (DX'04)* 197–202.