SYMBOLIC SYNTHESIS OF OBSERVABILITY REQUIREMENTS FOR DIAGNOSABILITY

B. Bittner[1], **M. Bozzano**[2], **A. Cimatti**[2], **and X. Olive**[3]

[1]*University of Amsterdam, Science Park 904, 1098XH Amsterdam, The Netherlands; bittner@science.uva.nl.*
[2]*Fondazione Bruno Kessler, Via Sommarive 18 / Povo, 38123 Trento, Italy; {lastname}@fbk.eu.*
[3]*Thales Alenia Space, 100 Boulevard Midi, 06150 Cannes, France; xavier.olive@thalesaleniaspace.com.*

ABSTRACT

Given a partially observable dynamic system and a diagnoser observing its evolution over time, diagnosability analysis formally verifies (at design time) if the diagnosis system will be able to infer (at runtime) the required information on the hidden part of the dynamic state. In this paper, we address a problem that generalizes diagnosability to discovering a set of parameters to be observed that is sufficient to guarantee diagnosability.

We propose a novel approach to the synthesis of observability requirements, with the following characterizing features. First, it fully covers a comprehensive formal framework for diagnosability analysis, and extends it to define cost-optimized configurations of observables. Second, we propose two complementary algorithms for the synthesis of observables. Third, we describe an implementation that takes full advantage of mature symbolic model checking techniques.

The proposed approach is thoroughly evaluated over a comprehensive suite of realistic benchmarks taken from the aerospace domain.

Key words: diagnosability, symbolic model checking, sensor placement.

1. INTRODUCTION

Whenever an engineer designs a system that needs supervision, the important question of what sensors to install arises. In most situations the number of sensors one can use is limited; only a limited number of system parameters are directly observable, and other parameters and properties such as faults have to be inferred by a diagnostic process.

In mission critical systems (e.g. production, power, avionics) it is of prime importance to formally verify such sensor configurations w.r.t. some desired diagnostic model. The question one asks is, "*Can a given diagnostic model be implemented on the basis of the available sensors*?" This problem is commonly referred to as *diagnosability*, and has been studied in a variety of works (e.g. [20], [15], [21], [11], [12], [18], [1]).

The problem we treat in this paper is a generalization of diagnosability, namely the synthesis of observability requirements guaranteeing diagnosability. The problem is also known as sensor placement and sensor selection for diagnosability, terms which can be used interchangeably if one sees a sensor as either a physical or an abstract observable system property, simple or complex. The question we ask is, "*What are the sensor configurations that guarantee diagnosability?*" One can be more specific and ask for one or all sensor configurations that are *diagnosable*, *diagnosable and minimal*, *diagnosable and cardinality-minimum*, or *diagnosable and cost-minimum*.

The synthesis task for diagnosability has already been studied through a variety of frameworks based on explicit-state analysis (e.g. [21], [23], [9], [22], [19], [7], and [14]).

Our work is characterized by the following features: we don't treat hybrid systems, but discrete ones; sensor faults themselves are not handled (by e.g. redundancy); we see diagnosability as a disambiguation problem for sets of states that don't necessarily partition the state space, generalizing the usual fault detection and identification case; we provide exact solutions, not approximations; to each sensor we associate a fixed cost that is a natural number; the cost of a sensor set is the sum of the costs of the individual sensors.

The innovative aspect of our work is that we offer a unified framework for the synthesis of: all diagnosable configurations; all minimal diagnosable configurations; all minimum diagnosable configurations w.r.t. a cost function. For this aim we developed a new algorithm inspired by methods for fault-tree analysis, which as a first step computes all diagnosable sensor placements and then allows to apply various efficient minimization routines. Based on our framework we also developed an algorithm inspired by [14], which aims to save resources by only looking for cost-minimum solutions. Beside the fact that we use a more generic definition of diagnosability, we improve upon [14] by providing symbolic tools for all

operations and by extending its scope to look for all cost-minimum sensor sets.

The paper is structured as follows. In section 2 we define diagnosability following the framework in [11]. Based on this, section 3 defines various synthesis problems for diagnosability, and shows how a sensor configuration can be used to synchronize the observability between two copies of a system plant. Section 4 describes how to build such synchronized plant copies, called twin plant, and how to use them to verify diagnosability. The symbolic encoding for the problem is explained in section 5. Given this framework, we propose in section 6 an algorithm that computes all diagnosable sensor placements and then all minimal or (cost) minimum ones, and in section 7 one that just computes the cost-optimal ones. Section 8 documents our benchmarks on various models including relevant case studies from the aerospace domain. In section 9 we discuss in more detail how our work relates to others, and in section 10 we draw conclusions on what has been done with an outlook to future activities.

2. DIAGNOSABILITY

We now describe the diagnosability framework used in this paper (see [11] for more details). Diagnosability analysis works with partially observable plants (see figure 1). Such plants are connected through sensor and actuator signals to a controller. The internal state of the plant is hidden.

Definition 1. *A (partially observable)* plant *is a structure* $P =< X, U, Y, \delta, \lambda >$*, where X,U,Y are finite sets, respectively called the* state space*,* input space*, and* output space*,* $\delta \subseteq X \times U \times X$ *is the* transition relation*, and* $\lambda \subseteq X \times Y$ *is the* observation relation*. We require that* $\forall x.\exists y.\lambda(x, y)$.

We notice that a state can be associated with more than one observation values. In this way, it is possible to model incomplete and/or imperfect sensing (see for instance [2]).

The sensor and actuator signals are intercepted by the diagnoser, whose task it is to give a state estimation for the plant to the controller. Diagnosis is usually carried out on the observable part of a finite system trace, called *history window*, which is used for diagnosing the hidden state.

Definition 2. *A* system trace *(feasible execution) of such a plant with a discrete number of time steps* t *is described as a sequence* $\pi = \langle x^0, y^0, u^1, x^1, y^1, \ldots, u^t, x^t, y^t \rangle$*. The observable part consists of the input and output signal:* $obs(\pi) = \langle y^0, u^1, y^1, \ldots, u^t, y^t \rangle$.

The diagnosability of a plant is defined w.r.t. a set of diagnosis conditions, where two properties define two sets of states that must always be distinguishable (e.g. $fault \perp \neg fault$, or $fault_a \perp fault_b$).

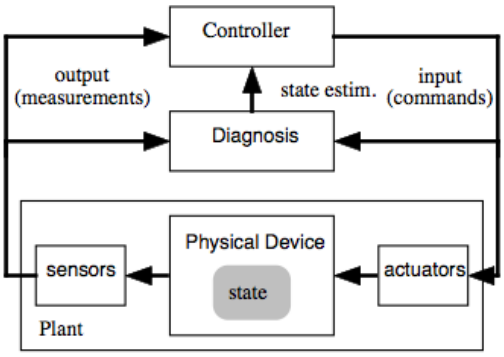

Figure 1. Architecture of a diagnosis system.

Definition 3. *A* diagnosis condition *for a plant P is a pair of nonempty sets of states* $c_1, c_2 \subseteq X$*, with* $c_1 \cap c_2 = \emptyset$*, written* $c_1 \perp c_2$.

A violation to diagnosability is witnessed by a pair of finite traces with equivalent observability, where in the last state of the first trace c_1 holds, and in the last state of the second trace c_2 holds. Such a pair of traces is called a *critical pair*.

Definition 4. *A* critical pair *for diagnosis condition* $c_1 \perp c_2$*, given plant* P*, is a pair of system traces* π_1 *and* π_2*, both of length* t*, with equivalent observability* $obs(\pi_1) = obs(\pi_2)$*, and* $c_1(x^t_{\pi_1}) \wedge c_2(x^t_{\pi_2})$.

3. SYNTHESIS FOR DIAGNOSABILITY

We now define the problem addressed in this paper, i.e. synthesis of observability requirements. This means finding (one or more) suitable configuration of sensors that will ensure diagnosability.

In order to do this, we consider an observation relation λ presented, in structured form, by N observation relations λ_i. We assume that the outputs are Boolean vectors, that is, $Y = \mathbb{B}^N$ (where $\mathbb{B} = \{0, 1\}$). Given N total observation relations $\lambda_i : X \times \mathbb{B}$, the induced observation relation $\lambda : X \times \mathbb{B}$ is defined as $\lambda(x, \mathrm{B})$ iff for all $i \in [1, N].\lambda_i(x, \mathrm{B}[i])$.

A sensor configuration represents a set of observability requirements, i.e. a subset of all the observations that are possible in the current system.

Definition 5 (Sensor Configuration, Plant Restriction)**.** *Let* $P =< X, U, Y, \delta, [\lambda_1, \ldots, \lambda_N] >$ *be a plant. A* sensor configuration *for* P *is a set of indices* $sc \subseteq \{1, \ldots, N\}$*. The* restriction *of* P *to a sensor configuration* sc*, denoted* $P_{\downarrow sc}$*, is the plant* $< X, U, Y, \delta, [\lambda'_1, \ldots, \lambda'_N] >$*, where* $\lambda'_i = \lambda_i$ *if* $i \in sc$*, and* $\lambda'_i = \lambda_\circ : X \times \{0\}$ *otherwise.*

Since $\lambda_\circ$ associates 0 to every state, it conveys no infor-

mation. This models the fact that in $P_{\downarrow sc}$ only the information available from the sensors in sc is available.

In the following we assume that a plant P and a a diagnosability problem for P are given. We want to find "less expensive" sensor configurations, that guarantee diagnosability with respect to the given problem, but at the same time use a reduced number of sensors, and/or minimize some cost measure.

Problem 1 (Minimal sensor configurations)**.** *Find all the sensor configurations* $sc \subseteq \{1, \ldots, N\}$ *such that* $P_{\downarrow sc}$ *satisfies the problem, and* sc *is minimal, that is, for every* $sc' \subseteq sc$*, if* $P_{\downarrow sc'}$ *satisfies the problem then* $sc = sc'$*.*

Problem 2 (Minimum sensor configurations)**.** *Find all the sensor configurations* $sc \subseteq \{1, \ldots, N\}$ *such that* $P_{\downarrow sc}$ *satisfies the problem, and* sc *is minimum w.r.t. cardinality, that is, for every* $sc' \subseteq \{1, \ldots, N\}$*, if* $P_{\downarrow sc'}$ *satisfies the problem then* $\mid sc \mid \leq \mid sc' \mid$*.*

Minimum sensor configurations are also minimal, following the definition in Problem 2. The notion of minimum configurations can be generalized with respect to a cost function expressed as $cost : 2^{\{1,\ldots,N\}} \rightarrow \mathbb{N}$. We require that $cost(sc_1) \leq cost(sc_2)$ if $sc_1 \subseteq sc_2$, and $cost(sc_1) < cost(sc_2)$ if $sc_1 \subsetneq sc_2$.

Problem 3 (Minimum cost sensor configurations)**.** *Given a diagnosability problem for* P*, find all the sensor configurations* $sc \subseteq \{1, \ldots, N\}$ *such that* $P_{\downarrow sc}$ *satisfies the problem, and* sc *is cost minimum, that is, for every* $sc' \subseteq \{1, \ldots, N\}$*, if* $P_{\downarrow sc'}$ *satisfies the problem then* $cost(sc) \leq cost(sc')$*.*

It is straightforward to see that minimum sensor configurations are a special case of minimum cost sensor configurations, by taking $cost(sc) = \mid sc \mid$. Furthermore, minimum cost sensor configurations themselves are a special case of minimal sensor configurations, as shown and proved below.

Theorem 1. *If a sensor configuration* sc *optimizes cost as defined in Problem 3, it is also minimal as defined in Problem 1.*

Proof. Let $sc' \subseteq sc$ and assume that $P_{\downarrow sc'}$ satisfies the diagnosability problem. Suppose that $sc' \subsetneq sc$. By definition of cost function, $cost(sc') < cost(sc)$, which contradicts the fact that sc is cost minimum. It follows that $sc' = sc$. □

4. TWIN PLANT CONSTRUCTION

In [11], the problem of diagnosability for P is reduced to searching critical pairs of P. This is done with the *coupled twin plant* construction, that consists of two synchronized copies of the original plant, where input and output are equivalent at each moment in time.

Definition 6 (Twin Plant)**.** *The* twin plant *of* P*, denoted* $\textsc{Twin}(P)$*, is the plant* $\langle X \cdot X, U, Y, \delta \cdot \delta, \lambda \cdot \lambda \rangle$*, where* $X \cdot X \in X^2$*, and, for all* $x_1, x_2 \in X$*,* $(x_1, x_2) \in X \cdot X$ *iff there exists* $y \in Y$ *such that* $\lambda(x_1, y)$ *and* $\lambda(x_2, y)$*;* $((x_1, x_2), u, (x_1', x_2')) \in \delta \cdot \delta$ *iff* $(x_1, u, x_1') \in \delta$ *and* $(x_2, u, x_2') \in \delta$*;* $((x_1, x_2), y) \in \lambda \cdot \lambda$ *iff* $\lambda(x_1, y)$ *and* $\lambda(x_2, y)$*.*

There is a one-to-one correspondence between pairs of indistinguishable traces of P, and traces of $\textsc{Twin}(P)$. Thus, the problem of diagnosability for P can be reformulated as a reachability problem for $\textsc{Twin}(P)$:

$$\textsc{Twin}(P) \models \neg \mathbf{F}(c_1(x_1) \wedge c_2(x_2)) \tag{1}$$

Let now a sensor configuration sc be given. The twin plant construction for $P_{\downarrow sc}$ is such that the traces of the left and right twins can only be distinguished using information provided by sensors in sc. This amounts to checking whether the plant obtained by dropping the sensors that are not in sc is still diagnosable. We reformulate problems 1, 2 and 3 as follows.

Problem 4 (Find sensor configurations, reformulated)**.** *Find all the sensor configurations* $sc \subseteq \{1, \ldots, N\}$ *such that the twin plant* $\textsc{Twin}(P_{\downarrow sc})$ *satisfies* (1)*, and* sc *is a minimal [minimum, or minimum cost, resp.] configuration.*

5. SYMBOLIC ENCODING

We now present a logical, symbolic encoding that allows us to use symbolic model checking techniques to search for sensor configurations. Following [16], a plant P is modeled symbolically using vectors of logical variables $\overrightarrow{\mathrm{X}}$, $\overrightarrow{\mathrm{U}}$, $\overrightarrow{\mathrm{Y}}$, such that a truth assignment to $\overrightarrow{\mathrm{X}}$ [$\overrightarrow{\mathrm{U}}$, $\overrightarrow{\mathrm{Y}}$, resp.] represents a state [an input, an output, resp.]. The transition relation is represented by a formula $\delta(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{X}}')$. $\overrightarrow{\mathrm{X}}'$ is called vector of next-state variables, and an assignment to $\overrightarrow{\mathrm{X}}'$ represents the state after a transition. Similarly, the observation relation is represented by a formula $\lambda(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{Y}})$. In the symbolic representation, sets of states are characterized by formulae, and set operations have a counterpart in the logical connectives.

The symbolic representation of a twin plant uses four variable vectors $\overrightarrow{\mathrm{X_1}}, \overrightarrow{\mathrm{X_2}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{Y}}$. Any subset of $X \times X \times U \times Y$ can be described with a formula $\phi(\overrightarrow{\mathrm{X_1}}, \overrightarrow{\mathrm{X_2}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{Y}})$. For instance, the formula $c_1(\overrightarrow{\mathrm{X_1}}) \wedge c_2(\overrightarrow{\mathrm{X_2}})$ expresses a state of the twin plant where the first instance is in c_1 and the second in c_2. The transition relation is represented as $\delta \cdot \delta(\overrightarrow{\mathrm{X_1}}, \overrightarrow{\mathrm{X_2}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{X_1}}', \overrightarrow{\mathrm{X_2}}')$, where both "twins" are constrained to respond to the same input $\overrightarrow{\mathrm{U}}$.

The observation relation $\lambda \cdot \lambda$ is parameterized with respect to any possible sensor configuration sc. We introduce a vector of N *activation variables* $\overrightarrow{\mathrm{A}}$, one for each

component λ_i. A truth assignment to $\overrightarrow{\mathrm{A}}$ represents a sensor configuration sc, i.e. $\overrightarrow{\mathrm{A}}[i]$ is true if and only if i is in sc, and thus λ_i is available. $\lambda \cdot \lambda$ is characterized as $\overrightarrow{\mathrm{A}}[i] \rightarrow (\lambda_i(\overrightarrow{\mathrm{X}_1}, \overrightarrow{\mathrm{Y}}) \leftrightarrow \lambda_i(\overrightarrow{\mathrm{X}_2}, \overrightarrow{\mathrm{Y}}))$. The transition relation is extended to constrain activation variables not to change over time.

Given this representation, all the standard symbolic techniques become available, e.g. image computation, reachability analysis, that are based the use of an efficient logical machinery to carry out the manipulation. Traditionally, Ordered Binary Decision Diagrams (OBDDs or BDDs for short) [8] have been extensively used for this purpose. BDDs are a representation for Boolean formulae, which is canonical once an order on the variables has been established. This allows equivalence checking in constant time. The basic set theoretic operations on sets of states (intersection, union, projection) are given by logical operations on BDDs (such as conjunction, disjunction, and quantification). BDDs provide primitives to compute efficiently all these operations.

As an example, we show an encoding of the image operator, which computes the forward image of a set of states S with respect to the transition relation δ:

$$\exists \overrightarrow{\mathrm{X}} \,.\, \exists \overrightarrow{\mathrm{U}} \,.\, (S(\overrightarrow{\mathrm{X}}) \wedge \delta(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{X}}'))$$

6. FTA-BASED SYNTHESIS

The first approach we propose for the synthesis of observables is based on the computation of all diagnosable sensor configurations. The algorithm we use is inspired by methods for FTA (Fault Tree Analysis) to compute minimal cut sets of a Boolean function, see [6]). The FTA problem asks which basic faults cause a given *top level event* (TLE). By duality, here we look for sensor configurations that enable critical pairs, making the system *not* diagnosable. From this, we can compute the configurations that guarantee diagnosability.

The basic faults in FTA are analogous to the activation variables $\overrightarrow{\mathrm{A}}$ of our twin plant. In this context, the top level event corresponds to the occurrence of a critical pair $(c_1(x_1) \wedge c_2(x_2))$ as in Equation 1, whereas a cut set of basic faults causing the TLE corresponds to a non-diagnosable sensor configuration.

The naïve enumerative way to solve this problem, for both FTA and diagnosability, is to first compute all reachable states of the model (here: twin plant). Then one can extract all states where the TLE occurs (here: critical pairs), and infer from the fault history variables for FTA (see [6]) (here: activation variables) under which circumstances the TLE occurs.

A more efficient way to discover all fault cut sets (here: diagnosable sensor configurations) is to actively prune the search space during a breadth-first forward search. In

function allDiagnosableSets ($\textsc{Twin}(P), TLE$)
1 $\quad Reach := \mathcal{I}(\textsc{Twin}(P))$
2 $\quad Front := \mathcal{I}(\textsc{Twin}(P))$
3 $\quad SC := \emptyset$
4 $\quad$ **while** $(Front \neq \emptyset)$ **do**
5 $\quad\quad SC := SC \cup Proj(\overrightarrow{\mathrm{A}}, Front \cap TLE)$
6 $\quad\quad SC := SC \cup allSubsets(SC)$
7 $\quad\quad Front := Front \backslash SC$
8 $\quad\quad temp := Reach$
9 $\quad\quad Reach := Reach \cup FwdImage(Front, \textsc{Twin}(P))$
10 $\quad\quad Front := Reach \backslash temp$
11 $\quad$ **end while**
12 $\quad SC := \neg SC$
13 $\quad$ **return** SC

Figure 2. Compute all diagnosable configurations.

FTA, if we know that a certain fault configuration is a cut set, we don't need to consider its proper supersets in further search. Dually, in the context of diagnosability we don't need to consider proper subsets of sensor configurations that cause non-diagnosability.

Based on these ideas, we now describe in detail our algorithm for computing all diagnosable sensor activations as shown in Figure 2. The algorithm makes use of a variable $Reach$ to accumulate the reachable states, a variable $Front$ to keep the *frontier*, i.e. the newly generated states (at each step, the image operator needs to be applied only to the latter set), with both variables being initialized with the initial states. Moreover, a variable SC is used to store the sensor configurations computed so far. The core of the algorithm (line 1 to 11) computes, using forward search on the state space of the twin plant, all sensor configurations that cause a critical pair. Such cut sets are computed by performing a conjunction between the symbolic representation of the TLE and the set of reachable states found so far (line 5).

Dynamic pruning of the search space is achieved by enlarging, at each iteration, the sensor configurations found so far by all their subsets, which by induction also cause critical pairs (line 6); we effectively compute $allSubsets(SC) := \{sc' \mid \exists\, sc \in SC \,.\, sc' \subset sc\}$. This is dual with respect to the widened cut set operation in [6]. The forward search is carried on the new frontier, and stops as soon as a fixpoint is reached. Finally, the set of fault configurations guaranteeing diagnosability is obtained by complementing the set SC computed by the previous routine (line 12).

Based on the results of the algorithm in Figure 2, we can (optionally) compute either all minimal cut sets, or all (cost) minimum cut sets. For solving Problem 1 we propose using classical procedures for minimization of Boolean functions (e.g., [13] and [17]), based on BDD operations, similarly to [6].

To solve Problem 2 and Problem 3, we can compute the cost of each sensor configuration, by computing the prod-

uct between the ADD representing our cost function and the BDD (a 0/1-ADD) representing all diagnosable configurations. We then determine the smallest cost value and keep only the minimum-cost configurations by performing a traversal of the resulting ADD.

7. TRACE-BASED SYNTHESIS

The second algorithm we propose for synthesis of observability requirements directly addresses Problem 2 and Problem 3. The algorithm of the previous section with activation-based pruning doesn't identify necessary observability conditions during search, and thus needs to find all (non-)diagnosable sensor configurations before any minimization can be applied. Necessary observability requirements can however be extracted from counterexamples to diagnosability, which can then be used to prune the activation space, in order to search only for sensor activations that minimize cost (uniform cost search).

The algorithm we describe here is inspired by the approach of [14], which also uses counterexamples for diagnosability in order to gradually refine the observability requirements. We extend this approach by describing a symbolic encoding of the twin plant, by using a symbolic encoding to represent the observability requirements instead of using an explicit enumeration of sensor activations, by using symbolic tools to compute the cost of sensor activations, and by extending the main algorithm to find all minimum configurations instead of only one.

The basic algorithm is shown in figure 3. It uses a variable $obsReq$ to store the set of candidate sensor configurations that make the system diagnosable, which initially contains all possible configurations (line 1). At each iteration we first compute, among all the candidate sensor configurations, those that have minimum cost with respect to the input cost function (line 3); we assume that function $getMinConf()$ returns an empty set (i.e., the sensor configuration with all sensors deactivated) if $obsReq$ is $\top$. In line 4 (where we override $configs$ to denote the logical formula representing the set of minimum-cost configurations) we use LTL model checking to see if the current cost-minimum observability requirements produce a critical pair. If this is the case, model checking returns a trace π for the twin plant that is a counterexample for $\textsc{Twin}(P) \models configs \rightarrow \neg\mathbf{F}\ TLE$. If no counterexample is returned, we have found a set of observability requirements that makes the system diagnosable, and the algorithm returns (line 6). Finally, in line 8 we extract the new observability requirements from the counterexample π. Essentially we compute the set of sensor configurations that may disambiguate counterexample π. This can be obtained as the set of states satisfying the following formula:

$$\bigvee_i (\overrightarrow{\mathrm{A}}[i]) \quad \text{such that} \quad (\lambda_i(x_1^t) \neq \lambda_i(x_2^t))$$

where (x_1^t, x_2^t) represents the state of the π trace at time t, where t is the length of π. If there are no sensors that

function allMinimumSets ($\textsc{Twin}(P), TLE, costFN$)
1 $obsReq := \top$
2 **do**
3 $configs := getMinConf(obsReq, costFN)$
4 $\pi := check(\textsc{Twin}(P) \models configs \rightarrow \neg\mathbf{F}\ TLE)$
5 **if** ($\pi = \emptyset$) **do**
6 **return** ($\top, configs$)
7 **endif**
8 $obsReq := obsReq \cap getObsReq(\pi)$
9 **while** ($obsReq \neq \emptyset$)
10 **return** ($\bot, \emptyset$)

Figure 3. Compute all cost-minimum diagnosable configurations.

can disambiguate the trace pair π, the expression in line 8 will evaluate to the empty set and the algorithm will return that the system is not diagnosable under any configuration (line 10).

8. EXPERIMENTAL EVALUATION

For the experimental evaluation of the presented algorithms we take a comparative approach and focus on problem 2, which asks to find all configurations that minimize cardinality. Therefore all test cases were evaluated using the fta-based approach with cost-based minimization, and the trace-based approach. In all tests, each observable has a cost of 1.

In our evaluation we used the following models.

ORBITER and ROVER are models of an orbiter and of a planetary rover, both develped in the OMCARE project (see [4] and [5]). The models describe the functional level, with various relevant subsystems including failure modes. The diagnosis property used for the benchmarks is whether a working component has failed (fault detection).

CASSINI models is the propulsion system of the Cassini spacecraft (see [6]). It is composed of two engines fed by redundant propellant/gas circuit lines, which contain several valves and pyro-valves. Leakage failures are attached to all components. The diagnosis property of interest is a correct input pressure in at least one of the engines in presence of a correct output pressure from the gas and propellant tanks.

ELEVATOR is a newly created set of models of an elevator controller, parameterized by the number of floors. The modelled properties are cabin and door movement, request and reset operations at each floor, and the controller logic. The property of interest is whether the cabin is moving or not.

The main features of the models are illustrated in Table 1.

The algorithms described in previous sections have been implemented on top of NuSMV [10].

model	orbiter	rover	cassini
# state vars	16	41	129
# input vars	3	2	26
state space	2^{27}	2^{50}	2^{166}
reachable states	2^{19}	2^{46}	2^{41}
% reachable	0.40	6.25	2.35e-36
diameter	33	31	8

model	elevator			
	8 fl.	12 fl.	16 fl.	20 fl.
# state vars	22	30	38	46
# input vars	0	0	0	0
state space	2^{26}	2^{35}	2^{43}	2^{51}
reachable states	2^{14}	2^{19}	2^{23}	2^{27}
% reachable	0.02	2.00e-3	9.54e-5	6.00e-6
diameter	4	4	4	4

Table 1. Some properties of benchmark models.

For both algorithms, we disabled the computation of all reachable states. For the trace-based algorithm, we decided to use backward search as suggested by [14] in order to better evaluate the approach our trace-based algorithm is inspired from. Conjunctive partitioning of the transition relation is the default for NuSMV. The BDD dynamic variable ordering was activated. No other optimizations are employed to allow for a more fair comparison of the approaches.

We ran each experiment with a different invocation of the model checker. The machine used for the experiments is an Intel Xeon 3.0GHz Quadcore with 4GB RAM, running Linux 2.6.18-x86_64. Only one core was used for each test run. The time limit was set to 12 hours (43200 seconds), and 2GB memory limit.

We report the results in Table 2. The labels have the following meaning: obs is the number of observables; c1 and s1 are the compile time and the synthesis time for the fta-based algorithm; s1d is the depth reached by the fta-based algorithm; c2 and s2 are the compile time and synthesis time for the trace-based algorithm; s2t and tl are number of critical pair traces and average trace length used by the trace-based algorithm. In Table 3 the percentage of diagnosable configurations for each model is shown, together with the number of minimal and minimum configurations.

The fundamental difference between the approaches lies in the pruning strategy on the twin plant. The fta-inspired approach performs pruning by expanding a forward breadth-first search only for activations where a proof for non-diagnosability has not been found yet, by example or by deduction. The trace-based approach, on the other hand, uses observability requirements extracted from counterexamples to diagnosability to exclude non-diagnosable activations from search. As a consequence, the fta-inspired approach first computes as a by-product all diagnosable sensor configurations, while the trace-based approach can directly perform a uniform cost search on the solution space.

model	obs	c1	s1	s1d	c2	s2	s2t (tl)
elev-8	15	26.5	77.6	5	26.8	2177.8	11 (4.91)
elev-12	15	109.4	105.4	5	109.5	2587.3	4 (5.50)
elev-16	15	471.6	276.4	5	472.4	5691.3	11 (4.91)
elev-20	15	1028.9	4.5	5	1030.1	1620.4	4 (5.50)
elev-12	20	301.3	1063.4	5	302.1	8824.9	8 (4.75)
elev-12	25	189.6	249.0	5	198.9	32491.3	9 (4.66)
elev-12	30	3356.0	10049.0	5	↑	↑	↑
orbiter	15	1.1	1.9	33	1.2	3.3	2 (2.00)
rover	20	3.9	207.8	33	3.8	16.5	9 (3.00)
rover	25	4.9	198.3	33	5.0	42.1	10 (2.90)
rover	30	7.1	531.8	33	7.2	545.4	9 (3.00)
rover	35	5.1	13876.0	33	5.2	18064.4	11 (3.00)
rover	40	↑	↑	↑	↑	↑	↑
cassini	15	48.8	24257.1	8	48.5	1055.3	6 (2.00)
cassini	20	48.1	16592.3	8	↑	↑	↑

Table 2. Experimental results.

model	diagnosable	minimal	minimum
elevator	30%	3	1
orbiter	27%	2	2
rover	11%	4	2
cassini	5%	3	3

Table 3. Diagnosability statistics valid for all test cases of the respective models.

The results in Table 2 show that the explicit extraction of observability requirements comes with a considerable computational trade-off: even though the fta-inspired approach follows the approach of first computing all diagnosable configurations, it is in almost all cases faster than the trace-based approach, and also seems to be more scalable w.r.t. an increasing number of observables. On the other hand, it seems like the trace-based approach has an advantage in cases with a higher percentage of reachable states.

The performance of the fta-based algorithm, which first synthesizes *all* diagnosable configurations, shows that such an operation is in principle feasible on real-world models. This is important, because having all diagnosable configurations allows one to efficiently extract all minimal configurations, thus getting a better picture of necessary observability constraints. It also allows an engineer to employ selection criteria for sensor configurations that are out of scope of a cost function.

We also ran test cases to obtain all minimal configurations (Problem 1). The performance difference was for all our test cases less than two seconds, therefore the results are not reported explicitly. Also this result is however very important, because minimal configurations can give fundamentally different ways to solve a diagnosability problem.

While the FTA-inspired algorithm gives promising results on a broad range of synthesis problems, the trace-based approach, which focuses only on the sub-problem

of minimum configurations, needs more work. Especially alternative model checking algorithms such as invariant checking or SAT-based bounded model checking need to be evaluated, where the extra effort of extracting counterexamples in order to obtain necessary observability requirements might pay off. Even if the performance of the trace-based algorithm can be improved, it is not certain if that will also enable it to be efficiently extended for solving other synthesis problems. However, a synergic usage of both approaches could ultimately provide a more useful toolbox for synthesis for diagnosability.

9. RELATED WORK

The problem of synthesis of observability requirements for diagnosability in discrete systems has been widely studied with mostly non-symbolic (explicit-state) approaches.

One way of dealing with the problems is using structural analysis, with or without analytical redundancy relations (e.g. [21] and [23], respectively). While providing a way of abstraction for continuous dynamics, these methods don't account for the potential blow-up of the discrete component, and have thus a limited scalability. The reason for this is that the state space is treated enumeratively.

A different solution is provided by [9], based on safety 2-player games and weighted automata. Sensor minimization is done by dynamic activation instead of synthesizing a static observer like in our approach. Furthermore, the cost function this approach optimizes is the mean cost of a run as the run length approaches infinity. Also in [22] optimization is done by dynamic activation, but in terms of the more restrictive but computationally more efficient minimal cardinality of observations (minimal, not minimum).

Regarding dynamic sensor activation, it should be noted that it can be obtained also through our approach: once one has the cost-optimal sensor activation, one can compute for each partition in the observation space the parameters that need to be observed.

In [19] the authors use an explicit-state approach based on Petri nets for obtaining a minimum sensor placement. In order to reduce computational complexity, however, they propose algorithms to compute an approximate solution.

The problem is reformulated in terms of the twin-plant construct in [7]. Given all observationally equivalent pairs of system runs, they give algorithms to increase or decrease the system observability in order to obtain a minimal sensor placement. A way to obtain those pairs of traces was not covered; however, the authors suggest to use an upper limit for trace lengths to reduce the computational complexity.

Building on the idea of [7], [14] proposes a algorithm to find the cost-optimal sensor placement. The algorithm is based on the conflict-hitting set method, and gradually refines the sensor placement by finding a new diagnosability counterexample for the current placement and obtaining new observability requirements from it. The counterexamples themselves are computed using model-checking and the twin plant approach, as proposed in [11]. However, no symbolic encoding of the synthesis algorithm itself is provided, and only the problem of cost-minimum configurations is addressed.

To the best of our knowledge, our approach is the only one describing a method to symbolically compute all sensor placements that optimize the cost. Almost all methods mentioned above use an explicit-state approach to sensor synthesis, and are thus much more prone to the state-space explosion problem which symbolical methods specifically address. Furthermore, we offer a common framework for finding all diagnosable configurations and for solving all minimization problems, whereas other works on exact solutions focus only on individual problems.

10. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a broad range of algorithmic strategies covering a variety of sensor configuration synthesis problems for diagnosability. In particular, we proposed, implemented, and compared two main approaches: one inspired by fault tree analysis, and another one based on the extraction of necessary observability requirements from counterexamples to diagnosability. The experimental evaluation showed the principal feasability of computing all diagnosable configurations in real-world models, allowing for more diverse selection criteria of sensor configurations.

Future work will include the evaluation of alternative model-checking techniques, especially w.r.t. to the trace-based approach; this will show how useful the extraction of diagnosability counterexamples to obtain necessary observability requirements can be. Also a combination of both approaches seems possible and might provide significant performance increases.

As to alternative model checking techniques, special attention will be given to SAT-based bounded model checking techniques [3]. As opposed to BDDs, that work by saturating sets of states, these techniques are typically used to find single traces of bounded length. The challenge is to make these techniques complete; a possible solution could be the generalization of induction techniques. Furthermore, we also want to investigate a "hybrid" approach combining BDD-based and SAT-based techniques into the same routine.

REFERENCES

[1] M. Bayoudh, L. Travé-Massuyes, and X. Olive. Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. *PAIS 2008*, page 219, 2008.

[2] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proc. IJCAI 2001*, 2001.

[3] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without bdds. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 193–207, 1999.

[4] M. Bozzano, A. Cimatti, A. Guiotto, A. Martelli, M. Roveri, A. Tchaltsev, and Y. Yushtein. Onboard autonomy via symbolic model-based reasoning. 2008.

[5] M. Bozzano, A. Cimatti, M. Roveri, and A. Tchaltsev. A comprehensive approach to on-board autonomy verification and validation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'11)*, 2011.

[6] M. Bozzano, A. Cimatti, and F. Tapparo. Symbolic fault tree analysis for reactive systems. *Automated Technology for Verification and Analysis*, pages 162–176, 2007.

[7] L.B. Briones, A. Lazovik, and P. Dague. Optimal observability for diagnosability. In *Nineteenth International Workshop on Principles of Diagnosis (DX-08)*, 2008.

[8] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[9] F. Cassez, S. Tripakis, and K. Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. In *Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on*, pages 90–99. IEEE, 2007.

[10] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 241–268. Springer, 2002.

[11] A. Cimatti, C. Pecheur, and R. Cavada. Formal verification of diagnosability via symbolic model checking. In *International Joint Conference on Artificial Intelligence*, 2003.

[12] M.O. Cordier, L. Travé-Massuyes, and X. Pucel. Comparing diagnosability in continuous and discrete-event systems. In *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06)*, pages 55–60, 2006.

[13] O. Coudert and J.C. Madre. Fault tree analysis: 1020 prime implicants and beyond. In *Reliability and Maintainability Symposium, 1993. Proceedings., Annual*, pages 240–245. IEEE, 1993.

[14] A. Grastien. Symbolic testing of diagnosability. In *Twentieth International Workshop on Principles of Diagnosis (DX-09)*, 2009.

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

[16] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[17] A. Rauzy. New algorithms for fault trees analysis. *Reliability Engineering & System Safety*, 40(3):203–211, 1993.

[18] J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[19] Y. Ru and C.N. Hadjicostis. Sensor selection for structural observability in discrete event systems modeled by petri nets. *Automatic Control, IEEE Transactions on*, 55(8):1751–1764, 2010.

[20] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans on Automatic Control*, 40:1555–1575, 1995.

[21] L. Travé-Massuyes, T. Escobet, and R. Milne. Model-based diagnosability and sensor placement. In *12th Intl. Work. on Principles of Diagnosis*, 2001.

[22] W. Wang, S. Lafortune, and F. Lin. Optimal sensor activation in controlled discrete event systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 877–882. IEEE, 2008.

[23] A.A. Yassine, S.P. Ploix, and J.M. Flaus. A method for sensor placement taking into account diagnosability criteria. *International Journal of Applied Mathematics and Computer Science*, 18(4):497–512, 2008.