Symbolic Synthesis of Observability Requirements for Diagnosability

Benjamin Bittner
Universiteit van Amsterdam
benjamin_bittner@gmx.it

Marco Bozzano and **Alessandro Cimatti**
Fondazione Bruno Kessler
lastname@fbk.eu

Xavier Olive
Thales Alenia Space
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. Diagnosability directly depends on the availability of observations, and can be guaranteed by different sets of sensors, possibly associated with different costs. In this paper, we tackle the problem of synthesizing observability requirements, i.e. automatically discovering a set of observations that is sufficient to guarantee diagnosability.

We propose a novel approach with the following characterizing features. First, it fully covers a comprehensive formal framework for diagnosability analysis, and enables ranking configurations of observables in terms of cost, minimality, and diagnosability delay. Second, we propose two complementary algorithms for the synthesis of observables. Third, we describe an efficient implementation that takes full advantage of mature symbolic model checking techniques. The proposed approach is thoroughly evaluated over a comprehensive suite of benchmarks taken from the aerospace domain.

Introduction

When designing 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. (Sampath et al. 1995; Jiang et al. 2001; Travé-Massuyes, Escobet, and Milne 2001; Cimatti, Pecheur, and Cavada 2003; Cordier, Travé-Massuyes, and Pucel 2006; Rintanen and Grastien 2007; Bayoudh, Travé-Massuyes, and Olive 2008)).

The problem we treat in this paper is a generalization of diagnosability, namely the synthesis of observability requirements guaranteeing diagnosability (Travé-Massuyes, Escobet, and Milne 2001; Yassine, Ploix, and Flaus 2008; Cassez, Tripakis, and Altisen 2007; Wang, Lafortune, and Lin 2008; Ru and Hadjicostis 2010; Briones, Lazovik, and Dague 2008; Grastien 2009). 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 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. To this aim, we developed two algorithms: one inspired by (Grastien 2009), which aims at saving resources by only looking for cost-minimum solution, and a new one inspired by methods for fault tree analysis, which as a first step computes all diagnosable sensor placements and then applies various efficient minimization routines. Finally, we generalize the framework to the problem of synthesis with a finite delay.

Diagnosability

We now describe the diagnosability framework used in this paper, adapted from (Cimatti, Pecheur, and Cavada 2003). 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* $P = \langle X, X_0, U, Y, \delta, \lambda \rangle$*, where X,U,Y are finite sets, respectively called the* state space*,* input space*, and* output space*,* $X_0 \subseteq X$ *is the set of* initial states*,* $\delta \subseteq X \times U \times X$ *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 note that a state can be associated with more than one observation value. In this way, it is possible to model incomplete and/or imperfect sensing (see for instance (Bertoli

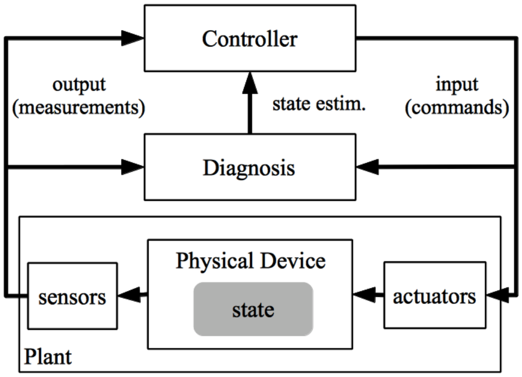

Figure 1: Architecture of a diagnosis system.

et al. 2001)). The sensor and actuator signals are intercepted by the diagnoser, whose task is to give a state estimation 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* $\pi = \langle x^0, y^0, u^1, x^1, y^1, \dots, u^t, x^t, y^t\rangle$ *such that* $x^0 \in X_0$, $\delta(x_{i-1}, u_i, x_i)$ *for* $i = 1, \dots, t$, *and* $\lambda(x_i, y_i)$ *for* $i = 0, \dots, t$. *The observable part consists of the input and output signal:* $obs(\pi) = \langle y^0, u^1, y^1, \dots, u^t, y^t\rangle$.

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. For instance, fault detection and fault identification can be expressed by conditions such as $fault \perp \neg fault$ and $fault_a \perp fault_b$, respectively.

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*. We generalize this definition of (Cimatti, Pecheur, and Cavada 2003) to take into account the possibility of diagnosis with a delay $d \in \mathbb{N}$.

Definition 4. *A* critical pair *for diagnosis condition* $c_1 \perp c_2$ *and delay d, given plant P, is a pair of system traces* π_1 *and* π_2, *both of length* $t + d$, *with equivalent observability* $obs(\pi_1) = obs(\pi_2)$, *such that* $c_1(x^t_{\pi_1}) \wedge c_2(x^t_{\pi_2})$ *holds.* [1]

Intuitively, we allow diagnosing using the observations in the next d steps after occurrence of the condition.

Finally, we say that a plant is diagnosable with *any finite delay* if there exists $d \in \mathbb{N}$ such that it is diagnosable with delay d.

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, and without loss of generality, 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 $\lambda_i \subseteq X \times \mathbb{B}$, the induced observation relation $\lambda \subseteq X \times \mathbb{B}^N$ is defined as $\lambda(x, \mathtt{B})$ iff for all $i \in [1, N].\lambda_i(x, \mathtt{B}[i])$.

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* $P = \langle X, X_0, U, Y, \delta, [\lambda_1, \dots, \lambda_N]\rangle$ *be a plant. A* sensor configuration *for P is a set of indices* $sc \subseteq \{1, \dots, N\}$. *The* restriction *of P to a sensor configuration sc, denoted* $P_{\downarrow sc}$, *is the plant* $< X, X_0, U, Y, \delta, [\lambda'_1, \dots, \lambda'_N] >$, *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 information. 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 diagnosability property for P are given. We want to find solutions that guarantee diagnosability and minimize the number of sensors or some cost measure.

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

Problem 2 (Minimum sensor configurations). *Find all the sensor configurations* $sc \subseteq \{1, \dots, N\}$ *such that* $P_{\downarrow sc}$ *satisfies the diagnosability property, and sc is minimum w.r.t. cardinality, that is, for every* $sc' \subseteq \{1, \dots, N\}$, *if* $P_{\downarrow sc'}$ *satisfies the diagnosability property 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 $cost : 2^{\{1,\dots,N\}} \rightarrow \mathbb{N}$. 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 property for P, find all the sensor configurations* $sc \subseteq \{1, \dots, N\}$ *such that* $P_{\downarrow sc}$ *satisfies the diagnosability property, and sc is cost minimum, that is, for every* $sc' \subseteq \{1, \dots, N\}$, *if* $P_{\downarrow sc'}$ *satisfies the diagnosability property then* $cost(sc) \leq cost(sc')$.

It is straightforward to see that minimum sensor configurations (Problem 2) are a special case of minimum cost sensor configurations (Problem 3), 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.

[1] The notation $c(x)$ means that state x is in the set of states c. The notation x^t_π represents the state in trace π at time t.

Lemma 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 property. 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 $sc' = sc$. □

Symbolic Parameterized Twin Plant

In (Jiang et al. 2001), 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* $\langle X \cdot X, X_0 \cdot X_0, U, Y, \delta \cdot \delta, \lambda \cdot \lambda \rangle$*, where* $X \cdot X \subseteq X^2$*, 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* $\lambda(x_2, y)$*;* $X_0 \cdot X_0 = \{(x_1, x_2) \in X \cdot X \mid x_1 \in X_0 \land x_2 \in X_0\}$*;* $((x_1, x_2), u, (x_1', x_2')) \in \delta \cdot \delta$ *iff* $(x_1, u, x_1') \in \delta$ *and* $(x_2, u, x_2') \in \delta$ *and* $(x_1, x_2) \in X \cdot X$ *and* $(x_1', x_2') \in X \cdot X$*;* $((x_1, x_2), y) \in \lambda \cdot \lambda$ *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 $\textsc{Twin}(P)$. Given a sensor configuration sc, 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.

Symbolic Encoding We now describe a symbolic encoding for the twin plant that allows us to use symbolic model checking techniques to search for sensor configurations.

Following (McMillan 1993), a plant P is modeled 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 $\delta(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{X}}')$, while the initial states are represented by the formula $X_0(\overrightarrow{\mathrm{X}})$. $\overrightarrow{\mathrm{X}}'$ 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 $\lambda(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{Y}})$. Sets of states are symbolically characterized by formulae, and set operations have a counterpart in the logical connectives. The symbolic representation of a twin plant uses 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 $c_1(\overrightarrow{\mathrm{X}_1}) \land c_2(\overrightarrow{\mathrm{X}_2})$ 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, allowing us to symbolically encode a family of plants, one for each sensor configuration. To this aim, 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.

```
function ALLDIAGNOSABLE ($\textsc{Twin}(P), \phi$)
notDiag := $\bot$
do
  maybeDiag := $\neg$ notDiag
  $\pi$ := $check(\textsc{Twin}(P) \models$ maybeDiag $\rightarrow \phi)$
  if ($\pi = \emptyset$) do
    return maybeDiag
  endif
  sc := $Proj(\overrightarrow{\mathrm{A}}, \pi^0)$
  notDiag := notDiag $\bigvee \{$ sc$'$| sc $\rightarrow$ sc$'\}$
while ($\neg$ notDiag)
return $\bot$
```

Figure 2: Compute all diagnosable configurations. ϕ is the LTL-formula expressing the absence of critical pairs.

Given this representation, all the standard symbolic techniques become available, e.g. for image computation and reachability analysis. Traditionally, Ordered Binary Decision Diagrams (OBDDS or BDDs for short) (Bryant 1992) have been extensively used for this purpose. The basic set theoretic operations on sets of states are given by logical operations on BDDs, that provide primitives to compute efficiently all these operations. As an example, the forward image of a set of states S with respect to the transition relation δ can be encoded as $\exists \overrightarrow{\mathrm{X}} \ . \ \exists \overrightarrow{\mathrm{U}} \ . \ (S(\overrightarrow{\mathrm{X}}) \land \delta(\overrightarrow{\mathrm{X}}, \overrightarrow{\mathrm{U}}, \overrightarrow{\mathrm{X}}'))$. More recently, SAT-based verification methods (e.g. bounded model checking (Biere et al. 1999)) have proved to be extremely effective.

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 (Bozzano, Cimatti, and Tapparo 2007). The FTA problem asks which basic faults cause a given *top level event* (TLE). These basic faults are analogous to the observation relations λ_i, modeled by means of the activation variables $\overrightarrow{\mathrm{A}}$ in our symbolic framework, while the TLE corresponds to the diagnosability condition from Definition 3. By duality, the problem we solve is to identify all configurations that can cause a critical pair (Problem 4). By taking the complement of the resulting set one obtains the set of all diagnosable configurations.

Problem 4 (All non-diagnosable configurations). *Find all sensor configurations* $sc \subseteq \{1, \ldots, N\}$ *such that* $P_{\downarrow sc}$ *does* not *satisfy the diagnosability property.*

The simple way to solve the problem is to enumerate all critical pairs, by exploring the reachable state space of the parameterized twin plant. From each critical pair, one can then identify all the corresponding non-diagnosable sensor configurations. Once the set of configurations that admit at least one critical pair is computed, the complement is exactly the set of all the diagnosable configurations. A more efficient way to discover all non-diagnosable configurations is to iteratively identify single critical pairs and the sensor configurations causing them. Additional non-diagnosable configurations can be identified by computing all subsets of the offending configuration. Intuitively, if a certain sensor configuration is not sufficient to rule out a critical pair, then a fortiori the configurations obtained by removing other sensors will not be sufficient either. This is dual to the dynamic pruning technique used to find minimal cut sets in FTA: each superset of a set of basic faults also explains the TLE , and can be pruned from the set of minimal candidates.

Based on these ideas, we now describe in detail our algorithm for computing all diagnosable sensor activations, as shown in Figure 2. We use *notDiag* to represent all non-diagnosable activations identified so far. Initially the non-diagnosable activations are set to the empty set (symbolically represented by $\perp$). All available candidates can then be computed through the negation of this formula (line 3 and 10). At each iteration, we verify if all our candidates guarantee diagnosability. In other words, the diagnosability check is used as a termination oracle. The algorithm is parameterized in order to deal with diagnosability with respect to different delays. This is done by setting ϕ to different LTL formulae, depending on the delay of interest. For d=0, we have that ϕ is $\mathbf{G}\neg TLE$, i.e. the diagnosability condition is unreachable. For a non-zero constant delay d (e.g. d=2), ϕ is $\mathbf{G}\neg(TLE \wedge \mathbf{X^2}\top)$, i.e. it is never the case that the diagnosis condition is reached, and then the twin plant can extend the trace for two more steps (without violating the equality of the left and right outputs). For the case of diagnosability with any finite d, we set ϕ to $\mathbf{G}\neg(TLE \wedge \mathbf{G}\top)$, meaning that it is impossible for the twin plant to reach the diagnosis condition, and then infinitely extend the trace. If the model checker proves the property $maybeDiag \rightarrow \phi$ (line 5), then the activations represented by $maybeDiag$ are all diagnosable activations for our problem and we return it. Else, π represents a critical pair for one of the configurations in $maybeDiag$. With the $Proj()$ function, a simple existential quantification routine, we identify the offending activation (line 8). $Proj(A, \pi^0)$ extracts from the first state of the trace (π^0) the assignment to A (the vector of activation variables) that is a symbolic representation of the sensor configuration that caused the counterexample π. Then we update the set $notDiag$ with the activation itself and all of its subsets (line 9). This operation is sound, since the executions allowed on an activation are also allowed by all of its subsets. The corresponding dual in FTA is the computation of widened cut sets (dynamic pruning, see (Bozzano, Cimatti, and Tapparo 2007)) for identifying minimal sets of basic faults.

On the set of all diagnosable configurations obtained by this synthesis algorithm, we can now (optionally) apply minimization requirements as defined previously, or any other kind of selection criteria. For solving Problem 1 we propose using classical procedures for minimization of Boolean functions (e.g., (Coudert and Madre 1993) and (Rauzy 1993)), based on BDD operations. To solve Problem 2 and Problem 3, we can simply traverse the tree representing all diagnosable activations and pruning subtrees with non-minimum cost.

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 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 (Grastien 2009), 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.

```
function ALLMINIMUM (TWIN($P$), $\phi$, costFN)
1    obsReq := $\top$
2    do
3        configs := $getMinConf$(obsReq, costFN)
4        $\pi$ := $check$(TWIN($P$) $\models$ configs $\rightarrow$ $\phi$)
5        if ($\pi = \emptyset$) do
6            return configs
7        endif
8        obsReq := obsReq $\wedge$ $getObsReq(\pi)$
9    while (obsReq)
10   return $\perp$
```

Figure 3: Compute all cost-minimum diagnosable configurations. $getObsReq()$ corresponds to Eq. 1

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, the symbolic representation of those that have minimum cost with respect to the input cost function (line 3); we assume that the function $getMinConf()$, for which we use the same approach as for fta-based synthesis, returns a negative assignment to all activation variables (i.e., the sensor

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: Properties of some benchmark models.

configuration with all sensors deactivated) if $obsReq$ is $\top$. In line 4 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 witnesses the violation to diagnosability. If no counterexample is returned, we have found a set of observability requirements that makes the system diagnosable, and the algorithm returns (line 6). The delay in diagnosability is dealt with by setting ϕ to the appropriate LTL formula, similarly to the algorithm in previous section. 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:

$$\bigvee_i (\overrightarrow{\mathrm{A}}[i]) \quad \text{such that} \quad (\lambda_i(x_1^t) \neq \lambda_i(x_2^t)) \qquad (1)$$

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

Experimental Evaluation

The algorithms proposed in previous sections have been implemented on top of the NuSMV model checker (Cimatti et al. 2002). For the comparison, we used the following benchmarks.

ORBITER, ROVERS, and ROVERB are models of an orbiter and of a planetary rover, both developed in the OMCARE project (see (Bozzano et al. 2008) and (Bozzano et al. 2011)). 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). CASSINIS and CASSINIB model the propulsion system of the Cassini spacecraft (see (Bozzano, Cimatti, and Tapparo 2007)). 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 models an elevator controller, parameterized by the number of floors. The modeled 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. C432 is a boolean circuit used as a benchmark in the DX Competition (Feldman et al. 2010), whose gates can permanently fail in various ways. The observables are the inputs and output values for the gates of the circuit. The property is whether a single gate is faulty. X34 is a benchmark describing a simplified version of the main propulsion system of a spacecraft (Cimatti, Pecheur, and Cavada 2003). The feature of some of the models are illustrated in Table 1.

model	obs	fta-based			trace-based		
		build	synth	# traces	build	synth	# traces
elev-8	15	161.0	5.3	81 (5)	14.7	2.9	12 (5)
elev-12	15	228.9	6.3	32 (5)	76.2	4.2	11 (5)
elev-16	15	188.2	10.9	80 (5)	310.8	7.1	17 (5)
elev-20	15	590.4	10.5	41 (5)	1013.1	13.1	29 (5)
elev-12	20	1285.6	11.2	131 (5)	592.4	4.4	13 (5)
elev-12	25	1033.4	28.7	373 (5)	1661.5	4.9	14 (5)
elev-12	30	↑	↑	↑	1346.0	5.6	16 (5)
orbiter	15	N.A.	< 1	8 (2)	N.A.	< 1	2 (2)
roverS	20	< 1	15.2	23 (2)	< 1	8.4	11 (2)
roverS	25	< 1	16.2	30 (2)	< 1	8.7	11 (2)
roverS	30	< 1	16.2	14 (2)	< 1	9.7	9 (2)
roverS	35	< 1	18.7	45 (2)	< 1	11.4	12 (2)
roverS	40	< 1	19.0	29 (2)	< 1	11.0	13 (2)
cassiniS	15	N.A.	1.7	17 (2)	N.A.	1.4	13 (2)
cassiniS	20	N.A.	1.3	11 (2)	N.A.	1.5	13 (2)
cassiniB	22	N.A.	5.1	41 (2)	N.A.	3.3	20 (2)
roverB	62	88.2	311.3	109 (2)	78.8	291.3	94 (2)
x34	105	N.A.	6.0	6 (2)	N.A.	12.3	21 (2)
c432	40	↑	↑	↑	N.A.	227.0	18 (3)

Table 2: The results for the two algorithms (timings in seconds). *build* is the time required to enable the BDD-based invariant checking algorithm (N.A.when SAT-based induction succeeds). ↑denotes out of memory.

For the experimental evaluation, we take a comparative approach and focus on Problem 2, which asks to find all configurations that minimize cardinality. In all tests, each observable has a cost of 1. The experimental evaluation was run by disabling the computation of all reachable states. For the trace-based algorithm, we decided to use backward search as suggested by (Grastien 2009) 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. The machine used for the experiments is an Intel Core 2 Duo 2.0GHz with 2GB of RAM, running Linux 2.6.32-37-x86_64. Only one core was used for each test run. The time limit was set to 1 hour (3600 seconds), and the memory limit to 900MB.

We report the results in Table 2. The labels have the following meaning: obs is the number of observables; build is the time required to generate the BDD-based machine, which is used to compute whether a fix point has been reached. Synth is the time required to compute the diagnosable configurations, based on SAT techniques. Sometimes SAT-based is sufficient to conclude that a fix point has been reached. In this case, N.A.is reported in the build column. # traces reports the number of counterexamples needed to solve the synthesis problem and (in parentheses) their average length.

We notice that the cost of the BDD-based induction is often substantial. This is due to two main reasons. First, the results are run with an active variable reordering algorithm that tries to dynamically adjust the order of variables to minimize the size of the BDDs being manipulated. Second, the twin plant contains many comparisons between the two (left and right) copies of the state vectors; these are formulae whose BDDs may easily blow up with the wrong BDD-order, so the variable reordering procedure is activated often. In fact, once the BDD-based representation of the twin plant is built, the cost of the BDD-based reachability turns out to be moderate.

The results show that the two algorithms have comparable performance, both able to deal effective with models of large size, with some leverage for the trace based. The interesting fact is that the performance of the FTA-based algorithm, which first synthesizes *all* diagnosable configurations, shows that such an operation is 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 very similar, therefore the results are not reported explicitly. This result is also very important, because minimal configurations can give fundamentally different ways to solve a diagnosability problem.

Below we report the percentage of diagnosable configurations for some of the models, together with the number of minimal and minimum sensor configurations.

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

Finally, in order to analyze the scalability with respect to the delay in diagnosability, we analyzed the CASSINIB example. The results, reported in following table, show that the introduction of a non-zero delay yields a degrade in performance.

	FTA-based		Trace-based	
delay	runtime (s)	# traces	runtime (s)	# traces
0	5.18	43	4.69	41
5	47.77	41	39.12	35
10	36.58	36	30.69	30
20	26.56	26	37.48	35
any	42.96	43	28.37	26

However, the degrade does not appear to be directly related to the delay. Runtime includes building the BDD model and performing synthesis, except the 0-delay case, where SAT-based induction succeeds. In the case of d=0, both algorithms identified 3 activations, and 27 activations in the case of all other delays. The maximum tracelength for d=0 was 2 steps, and for the other runs 7 steps.

Related work

The problem of synthesis of observability requirements for diagnosability in discrete systems has been widely studied mostly with non-symbolic (explicit-state) approaches. For instance, structural analysis is used in (Travé-Massuyes, Escobet, and Milne 2001; Yassine, Ploix, and Flaus 2008). 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. A solution based on safety 2-player games and weighted automata is described in (Cassez, Tripakis, and Altisen 2007). Here sensor minimization is done by dynamic activation, and the cost function is the mean cost of a run as the run length approaches infinity. In (Wang, Lafortune, and Lin 2008) optimization is also done by dynamic activation, but in terms of the more restrictive but computationally more efficient minimal (not minimum) cardinality of observations. In (Ru and Hadjicostis 2010) the authors use an explicit-state approach based on Petri nets for obtaining a minimum sensor placement and propose algorithms to compute an approximate solution.

The problem is reformulated in terms of the twin-plant construct in (Briones, Lazovik, and Dague 2008). 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 is not covered; however, the authors suggest to use an upper limit for trace lengths to reduce the computational complexity.

Building on these ideas, (Grastien 2009) proposes an 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 (Cimatti, Pecheur, and Cavada 2003). However, no symbolic encoding of the synthesis algorithm itself is provided, and only the problem of cost-minimum configurations is addressed. With respect to (Grastien 2009), we give a more general definition of 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. Moreover, we provide a fully symbolic implementation of their algorithm and extend its scope to look for all cost-minimum sensor sets. Finally, we provide a new algorithm inspired by methods for fault tree analysis.

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 either produce approximate solutions or focus only on individual problems.

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 observability requirements from counterexamples to diagnosability. The experimental evaluation showed the feasibility of computing all diagnosable configurations in real-world models, allowing for different selection criteria of sensor configurations. The key insight is a novel encoding for the twin plant construction, that allows to symbolically explore sets of sensor configurations. We propose an implementation based on state-of-the-art for formal verification technologies, including a combination of BDD-based and SAT-based model checking.

As part of our future work, we plan to investigate a combination of the FTA-based and trace-based approaches. Moreover, we plan to generalize the definition of the cost function to include more general cost measures, and to explore the use of abstraction techniques to enable a faster convergence. Concerning the problem of delayed diagnosability, we want to consider specialized induction techniques, and to generalize the framework to the problem of synthesizing the minimum delay sufficient to guarantee diagnosability.

References

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

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

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

Bozzano, M.; Cimatti, A.; Guiotto, A.; Martelli, A.; Roveri, M.; Tchaltsev, A.; and Yushtein, Y. 2008. On-board autonomy via symbolic model-based reasoning.

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

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

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

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

Cassez, F.; Tripakis, S.; and Altisen, K. 2007. 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*, 90–99. IEEE.

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

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

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

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

Feldman, A.; Kurtoglu, T.; Narasimhan, S.; Poll, S.; Garcia, D.; de Kleer, J.; Kuhn, L.; and van Gemund, A. 2010. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*.

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

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

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

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

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

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

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

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

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

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