Modeling and Solving Diagnosis of Discrete-Event Systems via Satisfiability

Alban Grastien
NICTA &
Australian National University
Canberra, Australia

Anbulagan
NICTA &
Australian National University
Canberra, Australia

Jussi Rintanen
NICTA &
Australian National University
Canberra, Australia

Elena Kelareva
University of Melbourne
Melbourne, Australia

Abstract

The diagnosis of a discrete-event system is finding out whether the behavior of the system is normal or faulty, given observations of this behavior. We show how the diagnosis problems can be translated into the propositional satisfiability problem (SAT) and then solved by the state-of-the-art SAT algorithms. Our experiments demonstrate that the SAT algorithms are able to deal with the problems, which are hard for traditional diagnosis algorithms.

Introduction

The diagnosis of a discrete-event system is finding out whether the behavior of the system is normal or faulty, given observations of this behavior. This computation can be done by checking whether there exist normal/faulty behaviors on the model of the system that are consistent with the observations. The main difficulty of this task is that the behavior of the system is only partially observable.

Two types of approaches for solving this problem have been presented. In the first approach the system model is compiled off-line in a structure which can be used on-line for efficiently detecting whether an observation sequence corresponds to normal or faulty behaviors. The main example is the Sampath *diagnoser* (Sampath *et al.* 1995). However, the size of the diagnoser can be double exponential in the size of the system description in the worst case (Rintanen 2007), which can make the diagnoser practically impossible to generate. The second approach is based on the computation of all behaviors and on checking whether these behaviors are correct (Lamperti & Zanella 2003; Pencolé & Cordier 2005; Su & Wonham 2005; Jiroveanu & Boël 2005; Cordier & Grastien 2007). Computing all behaviors can be extremely expensive.

Basically, the diagnosis task involves finding paths in the model of the system. An efficient approach to solving similar path finding problems most notably in classical AI planning (Kautz & Selman 1996) and model-checking (Biere *et al.* 1999) is to reduce them to the satisfiability (SAT) problem in the classical propositional logic. Rintanen and Grastien (2007) have shown that a system can be proved non-diagnosable with this approach. Diagnosis of static systems has earlier been viewed as a logical satisfiability (consistency) problem and specifically as a SAT problem (Reiter 1987; Veneris 2003). Williams and Nayak (1996) proposed the use of propositional logic to determine the state of the system. Schumann *et al.* (2007) have investigated the use of logic to solve diagnosis problems. In this paper, the diagnosis task is translated to a SAT problem which then can be efficiently solved by the current state-of-the-art SAT solvers.

The structure of the paper is as follows. First we present a simple system that is for earlier approaches to diagnosis too difficult because of the very high number of trajectories and sets of possible current states. Then we present a formalization of the discrete-event system diagnosis problem and its translation into SAT. Extensions for more complex observations and for a more accurate diagnosis are given in the next two sections. In the experiments' section we present data about the computational properties of the state-of-the-art SAT solvers in solving the diagnosis problem and compare the proposed method with existing diagnosis approaches.

Example

We consider a system of 20 identical components in a 5×4 grid such that each component is connected to its 4 neighbors. Corner and border components are neighbors to corner and border elements in the opposite sides. For example, the component in position $(0, 2)$ is connected to components in positions $(0, 1)$, $(0, 3)$, $(1, 2)$ and $(4, 2)$. The behavior of each component is represented by the automaton in Figure 1. All the components are initially in the state O. When a failure occurs in a component, its state changes to F and the component sends the message *reboot!* to its neighbors which receive the message *reboot?*, leading to state W, FF or R depending on their current state.

The goal is to monitor this system. The events IReboot and IAmBack correspond to a component sending an alarm. Given these observations, the monitoring system must determine what happened in the system. Sampath *et al.* (1995) have proposed a method for solving this kind of problems. The method consists of compiling the system description to a finite state machine called a diagnoser which efficiently maps a sequence of observations to abstract representations of sets of possible current states. The main problem with this approach is that the size of the diagnoser can be exponential in the number of states, and for this reason it cannot be computed for many systems with more than a couple of hundred or thousands of states.

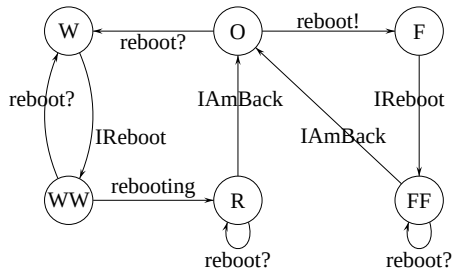

Figure 1: The behavior of one component

The other method involves computing all the behaviors of the system description that are consistent with the observations. However, for this example the number of such behaviors is astronomically large. We propose to solve the diagnosis problem by translating it into a SAT problem and then using state-of-the-art SAT solvers.

Definitions

System

We consider discrete-event systems, i.e. systems whose states can be represented by the assignment of a finite set of variables in finite domains. Here, we are restricted to two-valued (Boolean) state variables but in general variables may have any number of different values. Hence a state $s : A \rightarrow \{0, 1\}$ is a total function from the state variables to the constants 1 (*true*) and 0 (*false*). A *literal* is a state variable or its negation, and the set of all literals is $L = A \cup \{\neg a \mid a \in A\}$. The language $\mathcal{L}$ over A consists of all formulae that can be formed from A and the connectives $\vee$ and $\neg$. We use the standard definitions of further connectives $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$ and $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$.

Definition 1 (Model of the system)**.**
The model of the system *is a tuple* $SD = \langle A, \Sigma_u, \Sigma_o, \delta, s_0\rangle$ *where*

- A *is a finite set of* state variables,
- Σ_u *is a finite set of* unobservable events,
- Σ_o *is a finite set of* observable events,
- $\delta \subseteq \Sigma_o \cup \Sigma_u \rightarrow 2^{\mathcal{L}\times 2^L}$ *assigns each event a set of event instances* $\langle\phi, c\rangle$*, and*
- $s_0 : A \rightarrow \{0, 1\}$ *is the initial state.*

An *event instance* of the event e is a pair $\langle\phi, c\rangle \in \delta(e)$ which indicates that the event e can be associated with changes c in states that satisfy the condition ϕ. More formally, an event $e \in \Sigma_o \cup \Sigma_u$ is possible in any state s such that $s \models \phi$ for some $\langle\phi, c\rangle \in \delta(e)$. When e takes place in s, one of the pairs $\langle\phi, c\rangle \in \delta(e)$ satisfying $s \models \phi$ is chosen and the effect of the event is that the literals in c become *true*.

Let s be a state and $c \subset L$ a consistent set of literals (i.e. such that $l \in c \Rightarrow \neg l \notin c$). Then define the successor state $s' = \mathit{succ}(s, c)$ of s by

1. $s'(a) = 1$ for all $a \in A$ such that $a \in c$,
2. $s'(a) = 0$ for all $a \in A$ such that $\neg a \in c$, and
3. $s'(a) = s(a)$ otherwise.

The transition system is initially in the state s_0, and an event sequence $e_0, \ldots, e_{n-1}$ takes the system through a sequence $s_0, s_1, \ldots, s_n$ of states such that $\forall i \in \{0, \ldots, n-1\}$, $\exists\langle\phi, c\rangle \in \delta(e_i) : s_i \models \phi \wedge s_{i+1} = \mathit{succ}(s_i, c)$. Note that a state s_i and an event e_i do not necessarily determine the successor state s_{i+1} uniquely. The sequence of states and events is called a *trajectory* and models a behavior on the system.

This system description is very similar to the one used in planning. An important factor of efficient SAT-based planning (Kautz & Selman 1996) is the notion of parallel or partially ordered plans which allows several independent actions to be taken simultaneously. This approach has the advantage of making unnecessary to consider all $n!$ total orderings of n independent events, since their mutual ordering does not matter. The pairs $\langle\phi_1, c_1\rangle$ and $\langle\phi_2, c_2\rangle$ interfere if there is $a \in A$ that occurs positively/negatively in c_1 and negatively/positively in ϕ_2 or in c_2, or positively/negatively in c_2 and negatively/positively in ϕ_1. Events $e_1, \ldots, e_n$ can take place simultaneously with $o_1 \in \delta(e_1), \ldots, o_n \in \delta(e_n)$ if o and o' do not interfere for any $\{o, o'\} \subseteq \{o_1, \ldots, o_n\}$ such that $o \neq o'$.

We consider sequences $O_0, \ldots, O_{n-1}$ of (possibly empty) sets O_i of event occurrences, corresponding to the sets E_i of events, such that all members of O_i are mutually non-interfering. We define the successor state s' of s with respect to a set O of non-interfering event occurrences by

$$s = \mathit{succ}(s, \bigcup_{\langle\phi,c\rangle\in O} c).$$

Observations

The event sequence that occurs in the system is partially observable. Indeed, each observable event generates an observation emitted by the system. In general, it is considered that the observations received for the diagnosis are identical to the observations emitted by the system. However recent works (Lamperti & Zanella 2003; Grastien, Cordier, & Largouët 2005; Pencolé & Cordier 2005) consider uncertainties to do with the observations: partial order between the occurrence of observable events, uncertainty over which event has occurred, loss of observations, etc. To simplify, we consider in this section and the next one that the observations are a collection of timed observable events. This hypothesis is communly used in time-driven systems. More general cases are presented in a next section.

Definition 2 (Observations)**.**
The observations OBS *is a set of pairs* $\langle e, t\rangle$ *where* $e \in \Sigma_o$ *is an observable event and* $t \in \mathbf{N}^+$ *is a positive integer.*

The semantics is simple: if $\langle e, t\rangle \in OBS$, then the observable event e occurred at time step t. Conversely, if $\langle e, t\rangle \notin OBS$, the observable event e did not occur at time step t. A sequence of sets of events $E_0, \ldots, E_{n-1}$ is consistent with the observations OBS if:

$$\langle e, i\rangle \in OBS \Leftrightarrow (i < n \wedge e \in E_i) \quad \forall i \in \mathbf{N}^+, \forall e \in \Sigma_o.$$

Diagnosis

The *diagnosis label* of a trajectory can be *normal* or *faulty*. In this section, we consider that a trajectory is faulty if it contains a *faulty event*. The set of faulty events is denoted by $\Sigma_f \subseteq \Sigma_u$. More general cases are presented in section *Dealing with faulty behaviors*. Two trajectories are said to be *f-equivalent* if they share the same diagnosis label.

The behavior of the system is often ambigious: given a flow of observations of system behavior, the correctness of the behavior cannot always be proved, particularly if the system is not *diagnosable* (Sampath *et al.* 1995; Rintanen & Grastien 2007). Moreover, there is often a delay between a fault and the observations that enable this fault to be diagnosed. Because of this uncertainty, we consider that it is sufficient to find one normal behavior consistent with the observations to diagnose that the system behavior is correct.

Definition 3 (Diagnosis)**.**
Let SD be the model of the system, and let OBS be the observations on the system. The behavior of the system is normal *if there exists a sequence $\mathcal{E} = E_0, \ldots, E_{n-1}$ of events on SD consistent with OBS such that $\mathcal{E}$ is normal.*

As we show in section *Dealing with faulty behaviors*, the approach developed in this paper enables to answer *yes* or *no* to nearly any question about the behavior of the system and return a proof if the answer is *yes*.

Diagnosis by SAT

Diagnosing a system requires to find whether there exists a normal path on the model of the system that is consistent with the observations. This test can be formulated as a SAT problem, similarly to other path finding problems in AI planning (Kautz & Selman 1996).

We construct a formula such that satisfiable valuations of this formula correspond to the sequences $(s_0, \ldots, s_n)$ of states and the sequences $(E_0, \ldots, E_{n-1})$ of events consistent with the observations. The number n is the maximum value such that $\langle e, n-1\rangle \in OBS$.

Next, we define the formula Φ_{SD} that represents the system description $SD = \langle A, \Sigma_u, \Sigma_o, \delta, s_0\rangle$. The propositional variables, with superscript t corresponding to time step t, are the following:

- a^t for all $a \in A$ and $t \in \{0, \ldots, n\}$,
- e^t for all $e \in \Sigma_u \cup \Sigma_o$ and $t \in \{0, \ldots, n-1\}$, and
- ω^t for all $e \in \Sigma_u \cup \Sigma_o$, $\omega \in \delta(e)$, and $t \in \{0, \ldots, n-1\}$.

Next we describe $\mathcal{T}(t, t+1)$ for a given t which models the transition from time step t to time step $t+1$. When an event occurs, the event must be possible in the current state (1) and it has also some effects (2):

$$\omega^t \rightarrow \phi^t \quad \text{for every } \omega = \langle \phi, c\rangle \in \delta(e) \tag{1}$$

$$\omega^t \rightarrow \bigwedge_{l \in c} l^{t+1} \quad \text{for every } \omega = \langle \phi, c\rangle \in \delta(e) \tag{2}$$

The value of a state variable changes only if there is a reason for the change (*frame axiom*):

$$(a^t \wedge \neg a^{t+1}) \rightarrow (\omega_1^t \vee \ldots \omega_k^t) \tag{3}$$

for all $a \in A$ where $\omega_1 = \langle \phi_1, c_1\rangle, \ldots, \omega_k = \langle \phi_k, c_k\rangle$ are all event occurrences with $\neg a \in c_i$. For the change from *false* to *true* the formulae are defined similarly by interchanging a and $\neg a$.

An event can occur in only one way, and two events cannot be simultaneous if they interfere:

$$\begin{aligned} &\neg(\omega_1^t \wedge \omega_2^t) \text{ for all } e \in \Sigma_o \cup \Sigma_u \text{ and } \{\omega_1, \omega_2\} \subseteq \delta(e) \\ &\qquad \text{such that } \omega_1 \neq \omega_2 \\ &\neg(\omega_1^t \wedge \omega_2^t) \text{ for all } \{e_1, e_2\} \subseteq \Sigma_o \cup \Sigma_u \text{ and } \omega_1 \in \delta(e_1) \text{ and} \\ &\qquad \omega_2 \in \delta(e_2) \text{ such that } \omega_1 \text{ and } \omega_2 \text{ interfere.} \end{aligned} \tag{4}$$

The occurrence of events is represented by this formula:

$$(\bigvee_{\omega \in \delta(e)} \omega^t) \leftrightarrow e^t \text{ for all } e \in \Sigma_u \cup \Sigma_o \tag{5}$$

The conjunction of all the above formulae for a given t is denoted by $\mathcal{T}(t, t+1)$.[1] A formula for the initial state s_0 is:

$$\mathcal{I}_0 = \bigwedge_{a \in A | s_0(a)=1} a \wedge \bigwedge_{a \in A | s_0(a)=0} \neg a \tag{6}$$

A formula that represents all the behaviors described by the system for the length n is then

$$\Phi_{SD} = \mathcal{T}(0,1) \wedge \cdots \wedge \mathcal{T}(n-1, n) \wedge \mathcal{I}_0 \tag{7}$$

The observations are a set of timed occurrence of observable events: the observable event e occurred at time t iff $\langle e, d\rangle \in OBS$. More complex observations are considered in the next section. The representation of the observations consists of setting the e^t to *true* if this observable event $e \in \Sigma_o$ occurred at time step t, and to *false* if it did not.

$$\bigwedge_{\langle e,t\rangle \in OBS} e^t \wedge \bigwedge_{\langle e,t\rangle \notin OBS} \neg\, e^t \tag{8}$$

Theorem 1.
The solutions to the SAT problem $\Phi_{SD} \wedge \Phi_{OBS}$ represent the set of trajectories on SD consistent with the observations OBS and ending with the last observation of OBS.

To compute the diagnosis, we propose to *filter* the trajectories as proposed in (Torta & Torasso 2004) for static systems with the formula Φ_{Que} that is *true* if no faulty event occurred. Φ_{Que} is called the *query formula*.

$$\Phi_{Que} = \bigwedge_{e \in \Sigma_f, t \in \{0,\ldots,n-1\}} \neg e^t \tag{9}$$

The formula that finds sequences of events on the model that are consistent with the observations and that are normal is defined by: $\Phi_\Delta = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que}$. If this formula is satisfiable, then the diagnosis of the system is normal. Otherwise, it is faulty.

[1] If a single event can occur at any time step t, the following clauses should be added: $\forall e_1, e_2 \in \Sigma_u \cup \Sigma_o$, $\neg e_1^t \vee \neg e_2^t$ or another equivalent representation of this property.

Dealing with uncertain observations

This section deals with non trivial observations. We first discuss the problem when the number n of time steps is unknown. We then consider different scenarios for the observations: total order, partial order, observations represented by an automaton.

Dealing with an unknown number of time steps

In the previous section, diagnosis is performed with the assumption that the time step of the occurrence of an observable event is known for each observation. It provides several obvious pieces of information (ordering of the observable events, time of occurrence of the observations) but also the number of time steps n in the trajectories consistent with the observations. How to translate the diagnosis problem if n is unknown? In planning by SAT, the goal is to find the shortest plan that enables to reach a goal state: the value of n is incremented until the SAT problem $\Phi(n)$ is satisfiable. In diagnosis, we have to consider *all* the trajectories in the system that are consistent with the observations. We now show that it is possible to restrict ourselves to some value n as equivalent diagnosis result is provided when only considering the trajectories of length n.

Let $\mathcal{E}_k = E_0, \ldots, E_{k-1}$ be a sequence of sets of events of length k on the model of the system, and let $OBS(\mathcal{E}_k)$ be the observations received by the diagnosis system after the occurrence of $\mathcal{E}_k$ ($\mathcal{E}_k$ may not determine $OBS(\mathcal{E}_k)$ uniquely). Let OBS be the observations actually received from the system. The diagnosis using the value n as the number of time steps is always correct if $\forall k \in \mathbf{N}$, $\forall \mathcal{E}_k = E_0, \ldots, E_{k-1}$ such that $OBS(\mathcal{E}_k) = OBS$, there exists a trajectory $\mathcal{E}_n = E_0, \ldots, E_{n-1}$ which is f-equivalent to $\mathcal{E}_k$ and such that $OBS(\mathcal{E}_n) = OBS$. There are different ways to use this result.

Let $\mathcal{E} = E_0, \ldots, E_{k-1}$ be a sequence of sets of events of length k in the system. Another sequence of sets of events of length $k+1$ in the system is $E_0, \ldots, E_{i-1}, \varepsilon, E_i, \ldots, E_{k-1}$. The latter is f-equivalent[2] to the former. This means that if there exists a normal/faulty trajectory of size k, there also exists a normal/faulty trajectory of size $k+1$. Thus, it is sufficient to use an upper bound of the total number of time steps. For instance, if the maximum number of unobservable events between two observable events is x and the number of observations emitted is p, then an upper bound of the number of events is $(x+1) \times p$.

However, such an upper bound may not exist since there may exist loops of unobservable events, or may be too high which leads to a huge number of propositional variables and a huge SAT problem. Fortunately, it is possible to consider a small value for x.

Let $\mathcal{E} = E_0, \ldots, E_{n-1}$ be the sequence of sets of events that occurred in the system and let i, j be two times of occurrence of consecutive observations such that $j - i$ is big. Since most events do not interfere, f-equivalent sequences of sets of events can be found by moving earlier or later some events between E_i and E_j, or by merging consecutive non interfering sets E_l and E_{l+1}. For instance, if E_i and E_{i+1} do not interfere $\mathcal{E}' = E_0, \ldots, E_i \cup E_{i+1}, \ldots, E_{n-1}$ is f-equivalent to $\mathcal{E}$ depending on the definition of a faulty behavior. Thus, it is sometimes possible to use a smaller value for x.

In the example presented in the second section, it is sufficient to consider that $x = 1$. Indeed for any trajectory, there is an equivalent trajectory such that the events between two observable events are all non interfering.

[2]Since the notion of f-equivalence mainly relies on the definition of normal/faulty behavior, one should be careful with this statement in case of non-trivial definitions of a faulty behavior.

Total order on the observations

The observations OBS are received in the order they were emitted, possibly immediately, but the number of unobservable events is unknown. OBS is a set of p pairs $\langle e, i\rangle$ where $e \in \Sigma_o$ and $i \in \{1, \ldots, p\}$ such that $\langle e, i\rangle \in OBS \wedge \langle e', i\rangle \in OBS \Rightarrow e = e'$. Let $o = \langle e, i\rangle \in OBS$ and $o' = \langle e', i'\rangle \in OBS$ be two observations, the event e of o occurred before e' of o' if $i < i'$.

The main issue is to find the occurrence time step of the observations. As seen in the previous subsection, we consider an upper bound of the number of unobservable events. We denote $d(i)$ the time step of occurrence of the ith observable event ($d(i) = (x+1) * i - 1$ if x is constant). The formula Φ_{OBS} is the conjunction of the following clauses:

$$\begin{array}{ll} \neg e^t & e \in \Sigma_o : \forall i,\ d(i) \neq t, \\ \neg e^{d(i)} & i \in \{1, \ldots, p\},\ e \in \Sigma_o : \langle e, i\rangle \notin OBS, \text{ and} \\ e^{d(i)} & i \in \{1, \ldots, p\},\ e \in \Sigma_o : \langle e, i\rangle \in OBS. \end{array}$$

Partial order on the observations

The observations are partially ordered. Let $o = \langle e, i\rangle \in OBS$ and $o' = \langle e', i'\rangle \in OBS$ be two observations. The partial order $\prec$ between the observations has the following semantics: $o \prec o'$ if the observable event e of o surely occurred before e' of o'.

As in the previous subsection, we denote $d(j)$ the time step of occurrence of the jth observable event in the sequence of observable event. Note that the observable event e of the observation $o = \langle e, i\rangle$ is not necessarly the ith event of the sequence. A propositional variable $o^{d(j)}$ is created that indicates that the observable event associated with o occurred at time step $d(j)$ and a variable $\hat{o}^{d(j)}$ that indicates that the observable event associated with o occurred *before* or at the time step $d(j)$.

The formula Φ_{OBS} is the conjunction of the clauses given in Table 1. An observable event occurred before $d(j)$ if it occurred before $d(j-1)$ or at $d(j)$ (10). An observation is emitted only once (11). All the observations were emitted (12). The partial ordering must be defined (13). Finally, an observable event occurs if and only if an observation associated with this event was received (14).

It is possible to reduce the number of propositional variables. If o is such that $o' \prec o$, then o cannot be the first observation emitted. Then, obviously $o^{d(1)} = \hat{o}^{d(1)} = \mathit{false}$ and it is not required to create these variables. If k observations o' are such that $o' \prec o$, then the variables $o^{d(1)}$ to $o^{d(k)}$

$$\hat{o}^{d(j)} \Leftrightarrow \hat{o}^{d(j-1)} \vee o^{d(j)} \quad (10)$$

$$\hat{o}^{d(j-1)} \Rightarrow \neg o^{d(j)} \quad (11)$$

$$\hat{o}^{d(p)} \qquad \forall o \in OBS \quad (12)$$

$$o_2^{d(j)} \Rightarrow \hat{o}_1^{d(j-1)} \qquad \forall o_1, o_2 : \; o_1 \prec o_2 \quad (13)$$

$$e^{d(j)} \Leftrightarrow o_1^{d(j)} \vee \cdots \vee o_k^{d(j)} \quad (14)$$

where $o_1, \ldots, o_k$ represent the observations emitted by the event e.

Table 1: Rules used to model the partial order

are useless. The same reasoning can be used for the successors of o. In the worst case, the number of variables required to represent the observations is quadratic in the number of observations. However, if for each observation o, there are at most k observations o' such that $o \not\prec o'$ and $o' \not\prec o$, then the number of variables is less than $2 \times k \times p$ (where p is the number of observations).

Observations represented as a finite state machine

Recent works on diagnosis consider uncertain observations: in the case of large distributed systems, the delay between the emission of an observation and its reception leads to a partial order on the observations; observations can be noisy; observations can be lost, etc. These uncertainties can be handled by representing the observations using a finite state machine, such as *index space* in (Lamperti & Zanella 2003), *observation automaton* in (Grastien, Cordier, & Largouët 2005), where each path from an initial state to a final state represents a possible sequence of observable events that occurred in the system.

The observations can be translated into a formula Φ_{OBS} in a similar way as for Φ_{SD}. Here, the main issue is to determine the number of time steps n. The modeling of the observations can even be generalised to the observations of some of the state variables.

Dealing with faulty behaviors

The diagnosis result in previous sections only indicates whether a faulty event occurred in the system. However, a more accurate result is generally expected, such as the number of faults that occurred or the identification of these faults. More recently, it has been proposed to consider *supervision patterns*.

Number of faults

We denote by Φ_{Que_0} the formula which indicates that the behavior of the system was correct. Given that $\Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que_0}$ is not satisfiable, we know that at least one faulty event has occurred.

It is possible to build a formula Φ_{Que_i} that is satisfiable iff exactly i faults occurred in the system (Bailleux & Boufkhad 2003). The formula $\Phi_{\Delta_i} = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que_i}$ is satisfiable if a scenario consistent with the observations contains i faults. As we mentioned earlier, faults can occur that cannot be diagnosed because not enough observations were received. For this reason, we are interested in minimal number of faults that possibly occurred on the system.

We propose to test the satisfiability of Φ_{Δ_i} starting from $i = 0$ and increasing the value of i until Φ_{Δ_i} is satisfiable. Then, the minimal number of fault that might have occurred is i.

Let j be the number of faults that occurred in the system. The SAT solver is called for $j + 1$ times. In case j is high, it is possible to multiply i by two at each step and test the satisfiability of $\Phi_{SD} \wedge \Phi_{OBS} \wedge (\Phi_{Que_i} \vee \cdots \vee \Phi_{Que_{i \times 2-1}})$ until a value i is found such that $i \leq j < i \times 2$ and then get the correct value of j by dichotomy. The SAT solver is then called for $2 \times \log_2 j$ times.

Localisation

We also want to identify the exact faults that have occurred. The diagnosis problem now becomes finding the set of all faults that have occurred in the system. Let i be the smallest value such that Φ_{Δ_i} is satisfiable. The solution provided by the SAT solver is an example of a faulty behavior in the system. The set of faulty events $S_1 = \{e_1^{t_1}, \ldots, e_i^{t_i}\}$ can be easily extracted from this solution.

S_1 may not be the only set of faults of size i that is consistent with the observations. If we are interested in obtaining all the minimal cardinality diagnoses, then it is possible to build a formula from Φ_{Δ_i} such that S_1 cannot be a solution:

$$\Phi_{\Delta_i} \wedge \Big(\bigvee_{e^t \in S_1} \neg e^t \Big).$$

If this formula is satisfiable, another diagnosis S_2 can be extracted and a new formula can be built that accepts neither S_1 nor S_2. If there exist k different diagnoses of size i, the SAT solver may be called $i + k + 1$ times.

Extended faulty behaviors

Jéron *et al.* (2006) have recently proposed to consider supervision pattern rather than a unique faulty event. A supervision pattern describes a set of system behaviors. In the previous sections, we considered the set of behaviors that contain a faulty event. In (Jéron *et al.* 2006), a supervision pattern is represented by a finite-state machine that can be easily translated into a formula Φ_{Que} in a similar way as for Φ_{SD} and Φ_{OBS}. More generally, we consider formulae Φ_{Que}, which contain information about the event occurrences and the state variables assignments.

Let QUE be a set of faulty patterns. Let $\{QUE_1, \ldots, QUE_k\}$ be a partition of QUE such that $\Phi \in QUE_i$ means that the *fault level* of the pattern Φ is i. If two patterns $\Phi \in QUE_i$ and $\Phi' \in QUE_j$ with $i < j$ are possible explanations of the observations, then the diagnosis should return the pattern Φ rather than Φ'. Define QUE_0 as the set of patterns which accept the behaviors not represented in QUE (i.e. the normal behaviors).

This is a generalisation of the previous subsection: the set of faulty patterns QUE_i contains all the faulty patterns where i faults occurred; moreover it is considered that there is a trajectory consistent with the observations and that contains i faults, then $j > i$ faults should not be diagnosed even

if there exists another trajectory consistent with the observations and that contains j faults.

The formula that is satisfiable if a faulty pattern of level i is consistent with the observation can be defined by:

$$\Phi_{\Delta_i} = \Phi_{SD} \wedge \Phi_{OBS} \wedge (\bigvee_{\Phi \in QUE_i} \Phi).$$

The satisfiability of Φ_{Δ_i} can be tested starting from $i = 0$ until Φ_{Δ_i} is satisfiable. Then, it is easy to extract the pattern $\Phi \in QUE_i$ that has been recognised. Equal level patterns can be found by testing the satisfiability of formula $\Phi_{\Delta_i} \wedge \neg\Phi_1 \wedge \cdots \wedge \neg\Phi_k$, where $\{\Phi_1, \ldots, \Phi_k\} \subseteq QUE_i$ are the patterns that have already been recognized, and extracting a new pattern Φ_{k+1} if the formula is satisfiable.

Experiments

Tests were performed on the system described at the beginning of the paper. The experiments were conducted on a 1.73 GHz Pentium 4 computer using state-of-the-art SAT solvers. We found that the best SAT solvers for this kind of SAT problems are the ones based on the Davis-Putnam-Logemann-Loveland (DPLL) procedure (Davis, Logemann, & Loveland 1962) using the conflict-driven clause learning (CDCL) enhancement, such as Siege v4 (Ryan 2003), zChaff v151104, MINISAT v1.13, MINISAT v1.14, and MINISAT v2.0.[3] MINISAT 2.0 runs complete simplification procedure on input formula, in order to simplify the formula. Siege uses a seed, for the random number generator, taken from system time, so its performance is different for the same problem in different time. Thus, we run Siege 10 times on each problem for getting the minimum (MIN), maximum (MAX) and average (AVG) runtimes.

The first experiments were performed on a scenario with an increasing number of faults (from 1 to 20). The number of observations is about 10 times the number of faults. The observations are totally ordered and the goal is to determine the number of faults. The number of unobservable events between two observable events was set to $x = 1$. No incremental computation was used. Table 2 shows the runtime of the SAT solvers on the satisfiable formula $\Phi_{\Delta_k} = \Phi_{SD} \wedge \Phi_{OBS_k} \wedge \Phi_{Que_k}$; the results are also described graphically in Figure 2. For siege v4, we plot the average runtime of siege (siegeAVG) and describe its MIN and MAX runtimes by the grey region in the figure. The entries `#var` and `#cls` in the tables represent the number of variables and clauses in the CNF formula.

The table shows that SAT solvers solve the diagnosis problems very efficiently. Note that solving these problems by computing all consistent trajectories of the model using classical diagnosis approaches are hard. The Sampath's diagnoser approach would also fail because the diagnoser cannot be computed.

An important result is that the complexity of the computation does not necessarily increase with the size of the diagnosis window. For instance, MINISAT 2.0 takes 6.8s to solve

[3]These SAT solvers are available from http://www.satcompetition.org/.

	CNF properties		Runtime (s)			
k	#var	#cls	siege	zChaff	M 1.14	M 2.0
1	10 679	44 017	$0.01\,^{0.01}_{\mathbf{0.01}}$	0.03	0.03	0.03
2	21 718	90 091	$0.03\,^{0.03}_{\mathbf{0.03}}$	0.06	0.06	0.06
3	38 337	160 343	$0.09\,^{0.09}_{\mathbf{0.1}}$	0.37	0.26	1.11
4	51 636	217 593	$0.29\,^{0.13}_{0.53}$	**0.25**	0.42	0.73
5	70 415	299 361	$0.49\,^{0.22}_{0.98}$	**0.67**	3.1	6.8
6	80 094	343 427	$0.54\,^{0.26}_{\mathbf{1.2}}$	1.7	2	1.4
7	95 483	412 861	$2.4\,^{1.1}_{5.2}$	7.1	33	**4.1**
8	111 032	484 053	$3.2\,^{1.3}_{6.5}$	21	**3.8**	11
9	124 521	547 703	$2.6\,^{1.2}_{\mathbf{6.0}}$	23	138	8.9
10	139 970	621 071	$6.7\,^{1.1}_{30}$	9.6	**2.6**	27
11	153 604	687 472	$4.8\,^{1.8}_{\mathbf{11}}$	15	21	18
12	171 108	772 361	$6.5\,^{3}_{19}$	21	**7.2**	8.8
13	184 892	841 608	$5\,^{2.5}_{\mathbf{7.0}}$	11	14	9.6
14	200 656	920 953	$4.7\,^{2.8}_{6.4}$	18	**3**	14
15	214 585	992 951	$6.1\,^{3.6}_{8.2}$	18	**5.4**	8.1
16	226 664	1 057 317	$6\,^{4.2}_{8.9}$	16	**7**	7.7
17	244 261	1 149 035	$6.6\,^{3.7}_{9.9}$	24	**4.8**	78
18	259 978	1 233 191	$9.1\,^{5.2}_{18}$	32	7.7	**5.9**
19	275 735	1 318 745	$15\,^{8.5}_{23}$	32	18	**8.1**
20	287 672	1 387 077	$10\,^{4.3}_{23}$	20	11	**8.6**

Table 2: Runtime of SAT solvers on satisfiable Φ_{Δ_k} with totally ordered observations

Φ_{Δ_5} but only 1.4s to solve Φ_{Δ_6}. This relies on the property of the SAT solvers which do not perform a forward or a backward search but choose the variables in a dynamic order to get the smallest search tree: the behavior of Δ_k can be difficult to find, while it is obvious when new observations are provided in Δ_{k+1}. Interestingly enough, not all SAT solvers consider the same problem hard: Φ_{Δ_9} is harder than $\Phi_{\Delta_{10}}$ for MINISAT 1.14 ($138\text{s} > 2.6\text{s}$), while it is easier for MINISAT 2.0 ($8.9\text{s} < 27\text{s}$). The same tendancy can be found for Δ_9 and Δ_{17}. The issue of determining which algorithm is the more efficient and which heuristic should be used for this kind of SAT problem is an interesting open problem, particularly since the very similar MINISAT solvers give such different results.

In the diagnosis algorithm, Φ_{Δ_k} is computed by increasing k from $k = 0$ until Φ_{Δ_k} is satisfiable. In the planning domain, determining that there is no solution for $i = k - 1$ is often more expensive than finding a solution for $i = k$. Thus, we conduct a batch of experiments with $\Phi_{\Delta_k} = \Phi_{SD} \wedge \Phi_{OBS_k} \wedge \Phi_{Que_{k-1}}$ which is unsatisfiable. The runtime is limited to 600 seconds. The results are shown in Table 3.

The computation for these problems is much more expensive. The difficulty of these problems can be explained by the complexity associated with the computation of the num-

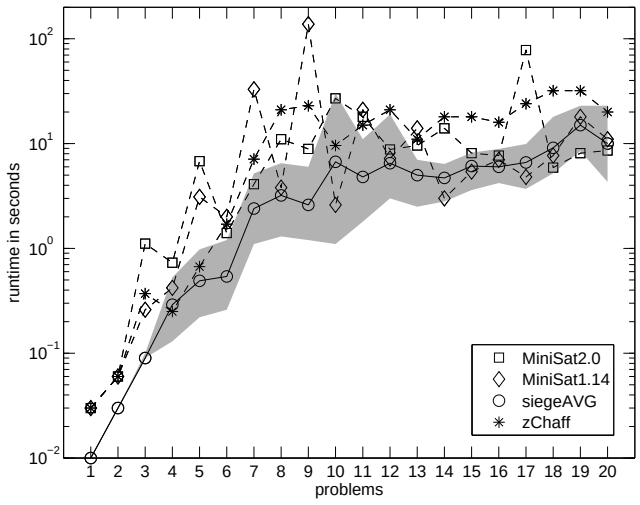

Figure 2: Runtime of SAT solvers on satisfiable Φ_{Δ_k} with totally ordered observations

	CNF properties		Runtime (s)			
k	#var	#cls	siege	zChaff	M 1.14	M 2.0
1	10 440	43 300	$0.01^{0.01}_{0.01}$	0.01	0.02	0.01
2	22 999	94 977	$0.02^{0.02}_{0.02}$	0.09	0.07	0.07
3	36 118	149 931	$0.1^{0.1}_{0.1}$	0.25	0.24	0.25
4	51 077	213 683	$0.54^{0.48}_{0.59}$	1.3	0.49	0.43
5	64 516	271 913	$1.4^{1.2}_{1.6}$	3	1.5	2
6	79 665	338 711	$7.4^{6.2}_{8.3}$	29	6	12
7	94 974	407 267	17^{12}_{23}	105	24	25
8	108 573	469 491	21^{16}_{28}	90	17	27
9	126 072	549 653	18^{13}_{25}	116	27	22
10	143 371	630 653	35^{23}_{47}	>600	62	41
11	155 090	688 191	39^{23}_{68}	>600	19	66
12	170 659	763 837	76^{68}_{84}	>600	52	128
13+	186 308	841 001	$>600^{>600}_{>600}$	>600	>600	>600

Table 3: Runtime of SAT solvers on unsatisfiable Φ_{Δ_k} with totally ordered observations

ber of faulty events. For instance, proving $\Phi_{SD} \wedge \Phi_{OBS_k} \wedge \Phi_{Que_0}$ is unsatisfiable only takes less than one second for any SAT solver used in our study for any given k.

The last test compares the runtimes when the observations become uncertain. We consider the following three cases: the observations are timed, totally ordered or partially ordered. When the observations are uncertain, an observation o_i was surely emitted before o_j, i.e $o_i \prec o_j$, iff $i + 5 < j$. The most efficient SAT solver for the partial order case being MINISAT 1.13, we present the result of this SAT solver in Table 4.

The computation is very simple in the case of timed observations. The runtime only increases slightly with new observations. However, the computation becomes very hard when the observations are partially ordered.

k	Timed observations	Total order	Partial order
1	0.02	0.03	0.04
2	0.07	0.08	0.09
3	0.16	0.26	0.14
4	0.36	0.3	3.6
5	2.6	1.9	9.7
6	3.5	16	5.6
7	4.8	9.1	350
8	4.1	24	290
9	4.3	61	63
10	4.2	25	>600
11	2.1	3.9	>600
12	2.7	6.3	>600
13	4.8	8.7	>600
14	4.3	16	>600
15	3.7	18.7	>600
16	11.8	9.6	>600
17	7.7	30	>600
18	9	13	>600
19	30	66	>600
20	12	16	>600

Table 4: Runtime of MINISAT 1.13 on satisfiable Φ_{Δ_k}

Discussion

Diagnosis with diagnosers (Sampath *et al.* 1995) is very efficient because processing an observation sequence can be done in linear time in the length of the sequence. This is in contrast to the SAT approach in which runtime in the worst case may grow exponentially in the length of the observation sequence. However, the construction of the diagnoser may be extremely expensive because the diagnoser may have a size that is exponential in the number of states in the system. In the SAT approach there is no similar direct dependency between the runtime and the number of states in the system. These two approaches represent a different trade-off between on-line and off-line computation and between dependency of the number of states and the length of event sequences.

The other traditional approach computes the set of all trajectories and determines whether these trajectories are normal. A disadvantage is that the number of trajectories is often extremely high and it is not practical to compute all of them. Obviously, for diagnosing one given observation sequence most of the possible trajectories are not needed (and this is taken advantage of in the SAT approach) but, once the set of all trajectories has been computed, it can be used several times for different diagnosis queries. Recent works on this approach (Cordier & Grastien 2007) use decentralized or factored representations to represent the set of all trajectories more compactly without enumerating all of them.

The complexity of the SAT problem is exponential in the number of propositional variables. The number of propositional variables is linear with the number of state variables and rules, and linear in the number of timestep. This makes

the diagnosis by SAT a problem harder in general than the classical algorithms. However, as in planning by SAT, the structure in the SAT problem enables the SAT solvers to solve the problem efficiently.

Conclusion

We translated the diagnosis of discrete-event systems into a SAT problem and then used the state-of-the-art SAT solvers to solve this problem. Our results show that diagnosis problems, which are unreachable by the traditional diagnosis approaches, can be solved efficiently by SAT solvers. However, the diagnosis is still challenging when the number of observations increases. An answer to this problem would be incremental diagnosis approach, and this would be an interesting prospect for diagnosis by SAT. The difficulty is then to ensure that the diagnosis of a temporal window will be consistent with the next window.

The results also show that the existing SAT solvers have performance varying with no single best solver for any given formula. Moreover, the difference in runtime between solvers can be huge. Determining the best SAT algorithm for diagnosis problems is still an open question.

SAT algorithms efficiently explore the search space. We claim that traditional diagnosis algorithms can inspire from these algorithms to improve their efficiency.

Acknowledgements

This research was supported by NICTA in the framework of the SuperCom and the G12 projects. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

References

Bailleux, O., and Boufkhad, Y. 2003. Efficient CNF encoding of Boolean cardinality constraints. In *Proc. of 9th International Conference on Principles and Practice of Constraint Programming (CP-03)*, 108–122.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In *Proc. of 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-99)*, 193–207.

Cordier, M.-O., and Grastien, A. 2007. Exploiting independence in a decentralised and incremental approach of diagnosis. In *Proc. of 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 292–297.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem proving. *Communication of ACM* 5:394–397.

Grastien, A.; Cordier, M.-O.; and Largouët, Ch. 2005. Incremental diagnosis of discrete-event systems. In *Proc. of 16th International Workshop on Principles of Diagnosis (DX-05)*, 119–124.

Jéron, T.; Marchand, H.; Pinchinat, S.; and Cordier, M.-O. 2006. Supervision patterns in discrete-event systems diagnosis. In *Proc. of 17th International Workshop on Principles of Diagnosis (DX-06)*, 117–124.

Jiroveanu, G., and Boël, R. 2005. Petri net model-based distributed diagnosis for large interacting systems. In *Proc. of 16th International Workshop on Principles of Diagnosis (DX-05)*, 25–30.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proc. of 13th National Conference on Artificial Intelligence (AAAI-96)*, 1194–1201.

Lamperti, G., and Zanella, M. 2003. *Diagnosis of Active Systems*. Kluwer Academic Publishers.

Pencolé, Y., and Cordier, M.-O. 2005. A formal framework for the decentralised diagnosis of large scale discrete-event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)* 164:121–170.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence (AIJ)* 32:57–95.

Rintanen, J., and Grastien, A. 2007. Diagnosability testing with satisfiability algorithms. In *Proc. of 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 532–537.

Rintanen, J. 2007. Diagnosers and diagnosability of succinct transition systems. In *Proc. of 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 538–544.

Ryan, L. 2003. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1995. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control* 40(9):1555–1575.

Schumann, A.; Pencolé, Y.; and Thiébaux, S. 2007. A spectrum of symbolic on-line diagnosis approaches. In *Proc. of 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*.

Su, R., and Wonham, W. M. 2005. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Transactions on Automatic Control* 50(12):1923–1935.

Torta, G., and Torasso, P. 2004. The role of OBDDs in controlling the complexity of model based diagnosis. In *Proc. of 15th International Workshop on Principles of Diagnosis (DX-04)*.

Veneris, A. 2003. Fault diagnosis and logic debugging using boolean satisfiability. In *Proc. of 4th International Workshop on Microprocessor Test and Verification: Common Challenges and Solutions*, 60–65.

Williams, B., and Nayak, P. 1996. A model-based approach to reactive self-configuring systems. In *Proc. of 13th National Conference on Artificial Intelligence (AAAI-96)*, 971–978.