Diagnosis of Discrete-Event Systems Using Satisfiability Algorithms

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 the problem of computing possible behaviors of the system given observations of the actual behavior, and testing whether the behaviors are normal or faulty. We show how the diagnosis problems can be translated into the propositional satisfiability problem (SAT) and solved by algorithms for SAT. Our experiments demonstrate that current SAT algorithms can solve much bigger diagnosis problems than traditional diagnosis algorithms can.

Introduction

Diagnosing a discrete-event system is determining if the behavior of the system is normal given observations about the behavior. This reduces to the question of whether some of the possible behaviors consistent with the observations are normal or faulty. The main difficulty in 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 to an automaton, the diagnoser (Sampath *et al.* 1995), which can be used on-line for efficiently detecting whether an observation sequence corresponds to normal or faulty behaviors. However, the size of the diagnoser can be doubly 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; Cordier & Grastien 2007). Computing all behaviors can be extremely expensive.

The diagnosis task is reducible to finding paths in a graph that represents the possible executions of the system. Similar path finding problems, most notably AI planning (Kautz & Selman 1996) and model-checking (Biere *et al.* 1999), have been very successfully reduced to the propositional satisfiability problem (SAT). Rintanen and Grastien (2007) have shown how the non-diagnosability of a system can be proved with this approach. Instead, in this paper we consider the diagnosis problem of discrete-event systems and show how it can be reduced to SAT and then efficiently solved by SAT solvers. 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.

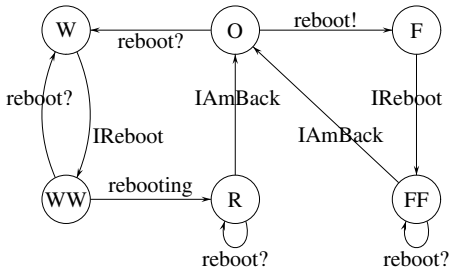


Figure 1: The behavior of one component

The structure of the paper is as follows. First we propose 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. In the experiments' section we present data about the computational properties of current 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.

The method of Lamperti and Zanella (2003) 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

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. In this paper we restrict 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. *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 pairs* $\langle\phi, c\rangle$*, and* s_0 *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 consequences 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. $\{a, \neg a\} \not\subseteq c$ for all $a \in A$). Then define the successor state $s' = succ(s, c)$ of s by

- $s'(a) = 1$ for all $a \in A$ such that $a \in c$,
- $s'(a) = 0$ for all $a \in A$ such that $\neg a \in c$, and
- $s'(a) = s(a)$ for all $a \in A$ that do not occur in c ($\{a, \neg a\} \cap c = \emptyset$).

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 $\forall i \in \{0, \ldots, n-1\}$, $\exists\langle\phi, c\rangle \in \delta(e_i) : s_i \models \phi \wedge s_{i+1} = succ(s_i, c)$. 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*.

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 (non-interfering) actions to be taken simultaneously. This approach has the advantage of making unnecessary to consider all $n!$ total orderings of n independent events because 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

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

Observations

The event sequences in the system are partially observable. The observable events in a system can be observed, but there may be unobservable events between observed events. There may also be uncertainty concerning the observations. For example, the ordering in which observations are made does not have to correspond to the order of the events that caused the observations (Lamperti & Zanella 2003). We formalize this form of observation uncertainty as a function *OBS* that maps a sequence of sets of events $\mathcal{E}$ to $\{$*true*,*false*$\}$ such that *OBS*$(\mathcal{E})$ is *true* iff $\mathcal{E}$ is consistent with the observations.

Diagnosis

The *diagnosis label* of a trajectory is either *normal* or *faulty*. Generally, a sequence of sets of events is called faulty if it contains an event from a predefined set of faulty events $\Sigma_f \subseteq \Sigma_u$. Two trajectories are *f-equivalent* if they have the same diagnosis label.

Given a set of observations, the correctness of the behavior cannot be determined with certainty, especially 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, observed behavior is considered correct if there is at least one normal behavior consistent with it.

Definition 2. *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 sets of events on* SD *consistent with* OBS *such that* $\mathcal{E}$ *is normal.*

Diagnosis by SAT

Diagnosing a system requires finding out whether there exists a normal path in 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 such as AI planning (Kautz & Selman 1996).

We construct a formula such that its satisfiable valuations correspond to sequences $(s_0, \ldots, s_n)$ of states and sequences $(E_0, \ldots, E_{n-1})$ of sets of events consistent with the observations. The problem of determining the length n of the sequences is discussed later in detail. The propositional variables, with superscript t corresponding to time step t, are

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

Modeling the System

We define the formula Φ_{SD} that represents the system $SD = \langle A, \Sigma_u, \Sigma_o, \delta, s_0\rangle$. The formula $\mathcal{T}(t, t+1)$ for a given t 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 some consequences (2). The value of a state variable changes from *true* to *false* only if there is a reason for the change (*frame axiom*, 3). For the change from *false* to *true* the formulae are defined analogously. An event can occur in only one way, and two events cannot be simultaneous if they interfere (4). The occurrence of events is represented by formula (5).

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

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

$$(a^t \wedge \neg a^{t+1}) \rightarrow (e^t_{1_{o1}} \vee \ldots e^t_{k_{ok}}) \tag{3}$$

for all $a \in A$ where $o_1 = \langle \phi_1, c_1\rangle, \ldots, o_k = \langle \phi_k, c_k\rangle$ are all event occurrences with $\neg a \in c_i$ and $e_1, \ldots, e_k$ are the respective events with $o_i \in \delta(e_i)$.

$$\begin{aligned} &\neg(e^t_o \wedge e^t_{o'}) \text{ for all } e \in \Sigma_o \cup \Sigma_u \text{ and } \{o, o'\} \subseteq \delta(e) \\ &\qquad \text{such that } o \neq o' \\ &\neg(e^t_o \wedge e'^t_{o'}) \text{ for all } \{e, e'\} \subseteq \Sigma_o \cup \Sigma_u \text{ and } o \in \delta(e) \\ &\qquad \text{and } o' \in \delta(e') \text{ such that } o \text{ and } o' \text{ interfere.} \end{aligned} \tag{4}$$

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

The conjunction of all the above formulae for a given t is denoted by $\mathcal{T}(t, t+1)$. 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 possible behaviors over $n+1$ time steps is

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

Modeling the Observations

We use formulae Φ_{OBS} for representing sequences of sets of events $\mathcal{E}$ such that *OBS*$(\mathcal{E})$.

In our experiments we consider three types of observation uncertainty. In the simplest case, each observation has a time tag and hence there is no uncertainty about the time point in which the corresponding event takes place. This is timed observations. In the more difficult cases the observations are totally ordered but the time points of the corresponding events are not known, or the observations are only partially ordered.

We only describe the representation of the simplest form of observations in detail. If the observations are reliable and the time of occurrence of all observable events is known, the observations can be represented as a set $\mathcal{O}$ of timed events and the corresponding formula is simply

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

An important issue is determining the number of time steps n. In some cases, the number of time steps can be known, for example for timed observations. More generally, we would have to consider the sequences of sets of events of all lengths. However, Theorem 1 shows that we can restrict to the trajectories for some n because any trajectory of length $k \neq n$ has an f-equivalent trajectory of length n.

Theorem 1. *The diagnosis is correct when only considering the sequences of sets of events of length n if $\forall k \in \mathbf{N}$, $\forall \mathcal{E}_k = E_0, \ldots, E_{k-1}$ such that OBS$(\mathcal{E}_k)$ is* true*, there exists a sequence of sets of events $\mathcal{E}_n = E_0, \ldots, E_{n-1}$ which is f-equivalent to $\mathcal{E}_k$ and OBS$(\mathcal{E}_n)$ is* true*.*

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 $E_0, \ldots, E_{i-1}, \emptyset, E_i, \ldots, E_{k-1}$. The latter is f-equivalent to the former. This means that if there exists a normal/faulty trajectory of length k, there also exists a normal/faulty trajectory of length $k+1$. Thus it is sufficient to use an upper bound as the number of events. For instance, if the maximum number of unobservable events between two observations is x and the number of observations is p, then an upper bound of the number of events is $(x+1)p$.

Such an upper bound may be too large but in some cases the value x can be tightened. For instance, non-interfering unobservable events occurring between two observable events can be considered to have been occurred simultaneously, lowering the value of x.

Modeling the Diagnosis Query

The basic query that can be answered by a call to a SAT solver is whether behaviors satisfying a given property exist. Typically such a query would be about the *fault level* i of possible behaviors: is the observed behavior compatible with at most i faults occurring. For example, one would be interested in knowing whether the observations are compatible with fault-free behavior, corresponding to fault level 0.

So a basic question in SAT-based diagnosis is to identify diagnoses with a given fault level i. Such diagnoses correspond to satisfying assignments of $\Phi_{SD} \wedge \Phi_{OBS}$ in which

exactly i (or at most i) propositional variables representing a failure event are true.

Bailleux and Boufkhad have shown how to encode this kind of cardinality constraints efficiently as a propositional formula (Bailleux & Boufkhad 2003). We denote the formula with a cardinality constraint i corresponding to fault level i as Φ_{FL_i}. Cardinality of diagnoses has been considered by logic-based methods earlier in the OBDD setting (Torta & Torasso 2004).

Finding a Diagnosis

Define $\Phi_{\Delta_i} = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{FL_i}$. This formula is satisfiable if there exists a sequence of sets of events with fault level i consistent with the observations. For instance, if Φ_{Δ_0} is satisfiable, the system behavior is normal. To diagnose the system, the satisfiability of Φ_{Δ_i} is tested for increasing values of i starting from $i = 0$. Since we are interested in the diagnosis of minimal level, the algorithm is stopped when Φ_{Δ_i} is satisfiable.

Let i be the smallest value such that Φ_{Δ_i} is satisfiable. One event sequence of fault level i can be extracted from the satisfying assignment the SAT solver has found. Since this may not be the only behavior of fault level i that explains the observations, further behaviors can be obtained by calling the SAT solver with an extended formula which excludes the behaviors already found. Note that this is equivalent to the problem of finding all the satisfying assignments of the formula Φ_{Δ_i}.

Empirical Evaluation

We consider the system presented at the beginning of the paper. We solved 360 diagnosis problems with the following parameters: length of the problem, difficulty level of the problem, observability, and whether there is a solution. The *length* of the problem varies from $m = 1$ to $m = 20$ faults: the number of faults determines the number of observations which is about $8m$. The *difficulty level*, *easy*, *medium* or *hard*, is a parameter used to generate the scenario and indicates whether the scenario can be easily reconstructed from the observations. The observability ranges from *timed observations* (the time step when the observable event occurred is known) and *totally ordered observations* (the order of emission between observable events is known) to *partially ordered observations*. Here the ith observation is known to have been emitted before the i'th observation iff $i < i' - 5$. For the non-timed observations in this system, an upper bound on the number of unobservable events between two observable events is $x = 1$. The SAT problem is either satisfiable if $k = m$ faults occurred, or unsatisfiable if $m > k$ faults occurred. Here we use $k = m - 1$.

The smallest problem instance contains 7860 variables, 32560 clauses, and 71840 literals while the largest problem instance contains 366284 variables, 1770145 clauses, and 4124906 literals.

The SAT Solvers

We exploit the power of the state-of-the-art SAT algorithms, based on either systematic search or stochastic local search (SLS), for solving the diagnosis problems. The best systematic solvers are based on the Davis-Putnam-Logemann-Loveland (DPLL) procedure (Davis, Logemann, & Loveland 1962). We ran the following solvers in our study[1]:

- March_dl: lookahead (LA) enhanced DPLL solver which is one of the best solvers for solving crafted and industrial SAT problems;
- MINISAT v1.14, Siege v4[2] and zChaff v151104: conflict-driven clause learning (CDCL) enhanced DPLL solvers which are the best for solving industrial problems. These three solvers have different performance on some problem domains depending on the decision heuristic and other simplification techniques.
- R+AdaptNovelty$^+$: resolution-enhanced random walk based SLS solver which is the best contemporary solver in the WalkSAT family;
- R+RSAPS: resolution-enhanced clause weighting based SLS solver which is one of the best SLS solvers for dealing with structured problems;

Results

The experiments were conducted on a cluster of 16 AMD Athlon 64 processors running at 2 GHz with 2 GB of RAM.

Figure 2 presents the performance on selected diagnosis problems for various solvers. We do not present the results of R+AdaptNovelty$^+$ as the SLS solver cannot solve any satisfiable problem instance in 20 minutes.

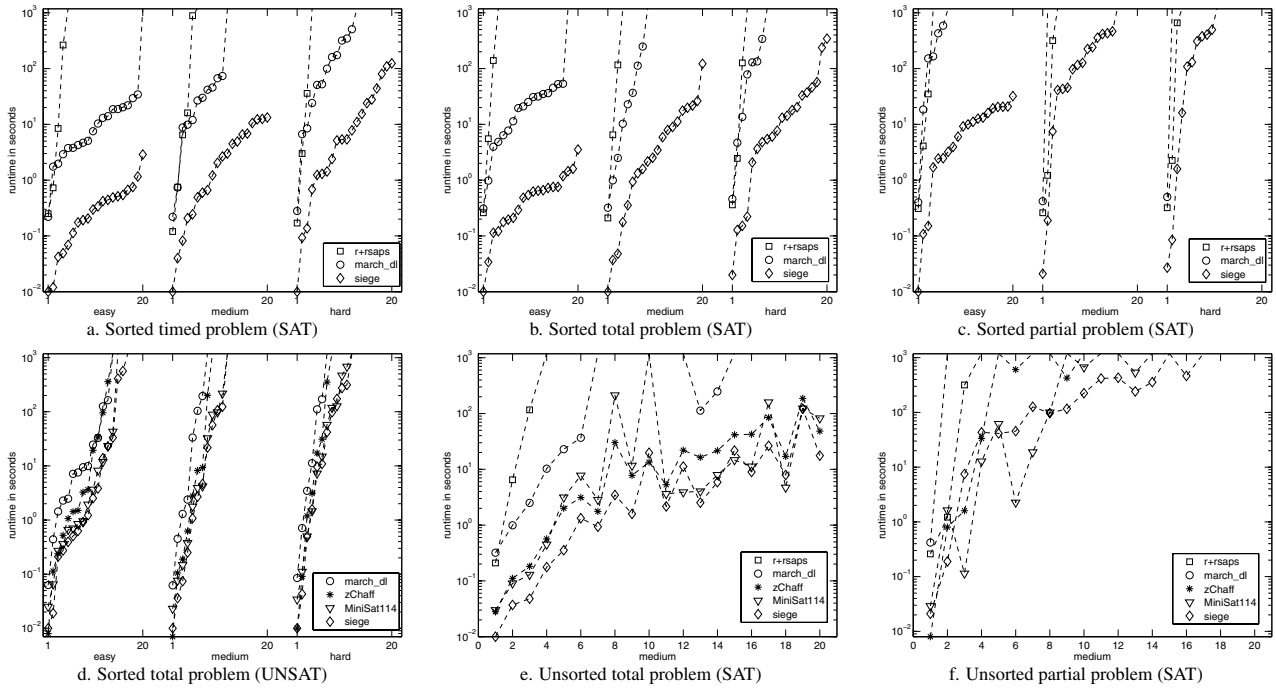
Figures 2a-2c present the results of one SLS, one LA, and one CDCL SAT solvers for the different classes of satisfiable problems, sorted by runtime (to make the curves grow monotonically.) As expected, the runtimes increase with the complexity of the diagnosis problem and the uncertainty about the observations. The runtimes for timed observations and totally ordered observations are similar. The main drawback of totally and partially ordered observations in comparison to the timed case is that the upper bounds for the number of time steps are less tight. Most of the problems were solved by Siege in less than 10 minutes, outperforming the SLS and LA based solvers.

Figure 2d presents the runtimes for the unsatisfiable problems when the observations are totally ordered (other unsatisfiable problems show the same tendency). As expected for unsatisfiable problems, the runtimes are much higher. For hard problems, Siege is able to prove that the number of faults is not $m - 1$ only for $m \leq 11$. We believe that the difficulty comes from the high value of m and not from the number of observations. Thus, we don't consider this a serious limitation as the number of faults in typical diagnosis problems is usually much smaller than 11.

Figures 2e and 2f show that the runtimes generally increase when the number of observations increases but the curves are not very smooth because of the high variation

[1] All SAT solvers used in our empirical study can be downloaded from www.satcompetition.org website.

[2] www.cs.sfu.ca/∼cl/software/siege/



a. Sorted timed problem (SAT) b. Sorted total problem (SAT) c. Sorted partial problem (SAT)

d. Sorted total problem (UNSAT) e. Unsorted total problem (SAT) f. Unsorted partial problem (SAT)

Figure 2: SAT solvers' performance on selected diagnosis problems

of SAT algorithm runtimes, especially for satisfiable formulae (Gomes *et al.* 2000). Here we focus on the totally ordered and the partially ordered observations for problems with medium difficulty level.

An important observation from the results is that there is no one SAT solver that is the best for all problems. Siege performs well for most of the problems but MINISAT and zChaff sometimes outperform it, especially when the number of partially ordered observations is small. Thus, an interesting open question is to determine diagnosis-specific heuristics.

Finally, the SAT solvers scale up quite well for problems with totally ordered observations while problems partially ordered observations become difficult much earlier.

Table 1 shows the numbers of problem instances solved by the SAT solvers. The columns Tmed and Tmean contain the median and mean runtimes respectively, while #Sol contains the numbers of instances solved by a given SAT solver when it is allowed to use 1200 seconds per instance. For R+RSAPS and unsatisfiable problems no data is given as the solver cannot determine the unsatisfiability.

CDCL solvers can solve most of the problems: Siege solves 71 per cent of the problems, MINISAT 64 and zChaff 60, while the other solvers do not perform as well. Moreover, when the observations are timed or totally ordered, Siege finds diagnosis in less than 30 seconds, zChaff similarly performs quite well, but MINISAT less so.

SAT Algorithm vs. Diagnosis Algorithms

Diagnosis with diagnosers (Sampath *et al.* 1995) can be 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 runtimes in the worst case may grow exponentially in the length of the observation sequence. However, in this approach the off-line phase preceding diagnosis, 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 runtimes 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 of this approach 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.

Conclusion and Future Work

We have presented a translation of the diagnosis of discrete-event systems into the SAT problem and then solved it by using SAT algorithms. Empirical results show that SAT al-

Problem	R+RSAPS			March_dl			MINISAT			Siege			zChaff		
	#Sol	Tmed	Tmean	#Sol	Tmed	Tmean	#Sol	Tmed	Tmean	#Sol	Tmed	Tmean	#Sol	Tmed	Tmean
timed-easy-*-s	4	1200	973.68	19	8.93	70.85	20	0.37	**0.44**	20	**0.32**	0.47	20	1.16	1.18
timed-medium-*-s	5	1200	945.60	11	70.35	555.83	20	2.88	6.82	20	**2.35**	**4.14**	20	3.39	8.38
timed-hard-*-s	3	1200	1021.96	12	332.29	567.48	16	23.28	256.94	20	**5.37**	**23.35**	20	20.17	43.01
timed-easy-*-u	–	–	–	15	7.11	305.20	16	**3.65**	242.97	17	3.67	**198.35**	15	14.64	309.93
timed-medium-*-u	–	–	–	8	1200	735.09	13	130.15	462.98	13	**79.26**	**443.75**	8	1200	721.77
timed-hard-*-u	–	–	–	4	1200	960.45	11	167.64	566.95	11	**150.55**	**564.77**	9	1200	727.74
total-easy-*-s	3	1200	1027.28	17	28.02	199.32	20	0.70	0.75	20	**0.58**	**0.71**	20	1.67	1.68
total-medium-*-s	3	1200	1026.15	8	1200	741.79	19	6.22	92.66	20	**2.99**	**12.70**	20	14.95	27.28
total-hard-*-s	3	1200	1026.47	7	1200	815.01	14	134.10	413.47	20	**10.42**	**42.59**	20	27.84	53.19
total-easy-*-u	–	–	–	13	28.74	439.41	14	5.93	364.78	16	**3.18**	**292.77**	13	26.74	446.08
total-medium-*-u	–	–	–	7	1200	796.83	11	157.21	562.30	11	**115.18**	**555.89**	8	1200	731.13
total-hard-*-u	–	–	–	6	1200	854.84	11	574.88	613.82	11	**293.93**	**586.18**	7	1200	800.26
partial-easy-*-s	3	1200	1021.97	6	1200	907.68	20	**4.35**	**7.43**	20	9.59	10.29	20	53.45	70.82
partial-medium-*-s	3	1200	1035.92	1	1200	1140.02	10	933.31	669.98	15	**232.96**	**430.59**	6	1200	893.23
partial-hard-*-s	3	1200	1053.20	1	1200	1140.03	6	1200	842.21	10	**848.89**	**692.46**	4	1200	971.05
partial-easy-*-u	–	–	–	2	1200	1080.03	4	1200	973.01	4	1200	**968.90**	3	1200	1020.84
partial-medium-*-u	–	–	–	2	1200	1080.03	4	1200	994.38	4	1200	**973.41**	3	1200	1021.36
partial-hard-*-u	–	–	–	2	1200	1080.04	3	1200	1020.74	4	1200	**987.59**	3	1200	1021.15
Total #Sol	30/180			141/360			232/360			**256/360**			219/360		

Table 1: Solvers performance on diagnosis problems.

gorithms outperform the current diagnosis algorithms. We hope that the SAT algorithms can inspire the diagnostic community in developing more efficient algorithms.

From the point of view of SAT algorithms, our results show that algorithms based on clause learning generally outperform algorithms based on lookahead or on stochastic search. An interesting question is that of incremental computation: given new observations, how to efficiently reuse the previous diagnosis in finding a diagnosis for the new closely related problem.

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 *Ninth International Conference on Principles and Practice of Constraint Programming (CP'03)*, 108–122. Springer-Verlag.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In Cleaveland, W. R., ed., *Fifth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-99)*, 193–207. Springer-Verlag.

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

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

Gomes, C. P.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning* 24(1–2):67–100.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, 1194–1201. AAAI Press.

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

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

Rintanen, J., and Grastien, A. 2007. Diagnosability testing with satisfiability algorithms. In Veloso, M., ed., *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 532–537. AAAI Press.

Rintanen, J. 2007. Diagnosers and diagnosability of succinct transition systems. In Veloso, M., ed., *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 538–544. AAAI Press.

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.

Torta, G., and Torasso, P. 2004. The role of OBDDs in controlling the complexity of model based diagnosis. In *Fifteenth international workshop on principles of diagnosis (DX'04)*.

Veneris, A. 2003. Fault diagnosis and logic debugging using Boolean satisfiability. In *Fourth 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 Minker, J., ed., *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 971–978. AAAI Press.