Simplifying diagnosis using LSAT: a propositional approach to reasoning from first principles

Andreas Bauer

Institut für Informatik
Technische Universität München
D-85748 Garching b. München, Germany
baueran@in.tum.de

Abstract. In face of the unwieldiness of non-monotonic logic solving engines, or meta interpreters usually authored in Prolog/CLP as they are commonly used for model-based reasoning and diagnosis, this paper proposes a simple, but effective improvement for performing the complex diagnostic task. The chosen approach is twofold: firstly, the problem of contradicting first order system descriptions with a set of observations is reduced to propositional logic using the notion of symptoms, and secondly, the determination of conflict sets and minimal diagnoses is mapped to a problem whose technical solution has experienced a sheer boost over the past years, namely k-satisfiability using state-of-the-art SAT-solvers. Since the involved problems are (mostly) $\mathcal{NP}$-complete, the ideas for additional improvements for a more diagnosis-specific SAT-solver are also sketched and their implementation by means of a non-destructive solver, LSAT, evaluated.

Keywords: model-based reasoning, model-based diagnosis, SAT-solving, system monitoring, formal specification.

1 Introduction

Ever since Reiter's seminal work on diagnosis from first principles [19], the automated reasoning and model-based diagnosis communities have spawned a lot of work on the implementation and improvement of the proposed as well as on related ideas. Amongst these, probably the most influential ones have been Reiter's own HS-algorithm and *default logic* [19, 4], the concept of *abduction* [20], or even McCarthy's notion of *circumscription* [16]. While reference implementations such as the *General Diagnostic Engine* [6] (GDE) realise some of these ideas, many of these are still faced with time and space complexity problems, mainly due to the sheer complexity of the underlying decision problems; see, for instance, [22, 23].

Given a system S, the diagnostic task is to identify those parts or components $c_i \in S$ which are assumed to be faulty in order to explain an observed behaviour of S. If no such set of components can be isolated, then the system is assumed

to work according to its specification, i. e. is correct. According to the logic- and consistency-based approaches to diagnosis, this task is performed by detecting a contradicting behaviour of a system S when compared to its expected behaviour which is captured by a system description $SD \subset S$. Such a contradiction is then expressed in terms of a set of conflicts or diagnoses.

Moreover, in order to determine practically useful sets of conflicts and diagnoses that would allow hinting to specific, faulty parts of a system, all the proposed diagnosis methods involve a subsequent task minimising the solutions which, in itself, is a computationally complex undertaking. In Reiter's case, for instance, this task is not separable from the initial determination of conflicts. However, his method relies on the availability of a suitable first order theorem prover for finding at least a single conflict set to execute an algorithm for finding minimal *hitting sets* of conflicts that constitute the diagnoses.

The hitting set problem, on the other hand, also known as the transversal problem, is one of the key problems in the combinatorics of finite sets and the theory of diagnosis per se. It turns out to be a hard problem which also helps to explain the continuing hesitation of a broader industrial application of model-based diagnosis techniques. Further, partly empirical, results from other authors regarding the wieldiness of implementations for non-monotonic reasoning (i. e. default logic, circumscription, etc.) second this conclusion (see § 5, for further details on non-monotonic reasoning).

Contribution

This paper will show that the recent achievements in solving the k-satisfiability problem with heuristic search algorithms and pruning using state-of-the-art SAT-solvers can also be used for consistency based diagnosis and even for those cases where the task is not related to merely boolean circuits and the likes. More so, empirical results will show that, using a SAT based approach to diagnosis, one can handle several thousand variables (i. e. abstract system components) at ease and — when constrained to an appropriate n-fault assumption (see § 4.1) — even tens and hundreds of thousands. Clearly, this is much more than non-monotonic reasoning engines can currently handle in reasonable time and space.

Therefore the contribution of this paper is to introduce an alternative methodology (based on simple propositional logic) for diagnosing technical systems, regardless as to whether these are software- or hardware-based, or both. Specifically, LSAT is presented which is a prototype SAT solver tailored to perform system's diagnosis.

Outline

After a brief overview over the theory of consistency- and logic-based diagnosis using system models in § 2, the transformation of the (non-monotonic) diagnostic reasoning from first principles to propositional logic is then described in § 3. LSAT, the main implementation vehicle for the concepts presented in this paper, is outlined in greater detail in § 4, and an evaluation of the deployed algorithms

w. r. t. processing large combinatorical benchmarks is given in § 4.2. A section on related work (§ 5) describes other recent results in complexity measures regarding non-monotonic reasoning which seconds an important claim of this paper; that is, diagnosis should be tackled using modern SAT-solvers with specific, problem-oriented heuristics. Finally, § 6 presents some conclusions.

2 Consistency based diagnosis

In model and consistency based diagnosis[1], the system to be diagnosed, S, is determined by a tuple $(SD, COMP)$, where SD constitutes a finite set of first order sentences comprising a system description, and $COMP$ a finite set of components in S. The set of components can be of almost arbitrary granularity; depending on the properties of the system to be diagnosed, $COMP$ may refer to, say, Java threads, user session objects within a web application, or even physical entities such as sensors, actuators, or entire nodes of a computer network. The overall system behaviour is then defined in terms of the components' behaviours and their causal dependencies, represented as shared variables/predicates.

2.1 Definitions

In this section, let us recall some notions, notations and terminology used later in the paper.

Definition 1 (Observation). *An observation for a system $S = (SD, COMP)$ is a finite set of first order sentences each comprising a mapping of in- and outputs of S to actual/observed values: $input_i, output_i : c \in COMP \rightarrow Num$. The index i denotes the i-th input (resp. output) for component c, whereas Num represents the class of all numerical sorts. An observation OBS for S is denoted by $(SD, COMP, OBS)$.*

Example 1. The example system S depicted in Fig. 1 contains two multiplication components, M_1 and M_2, and two summation components, A_1, A_2. We use the more compact representation of an n-tuple $\langle i_1, \ldots, i_4, m_1, m_2, o_1, o_2 \rangle$ to capture the model's observed in- and output values in $\mathbb{N}$. Hence, $S = (SD, \{M_1, M_2, A_1, A_2\})$, with $OBS = \langle 2, 3, 4, 5, 6, 20, 26, 26 \rangle$. Without going much into further detail at this point, SD basically captures the behaviour and causality of each component. ◊

Diagnosis can be understood as the process of finding and isolating differences between a system's model, i. e. the intended behaviour, and reality, i. e. observed behaviour. Typically, in order to reason about system models, at least one predicate needs to be introduced for the mere purpose of representing "normal" and

[1] From this point forward, the terms consistency-, logic-, and model-based are used synonymously due to the similarities of these approaches and their common problems w. r. t. complexity of their realisations.

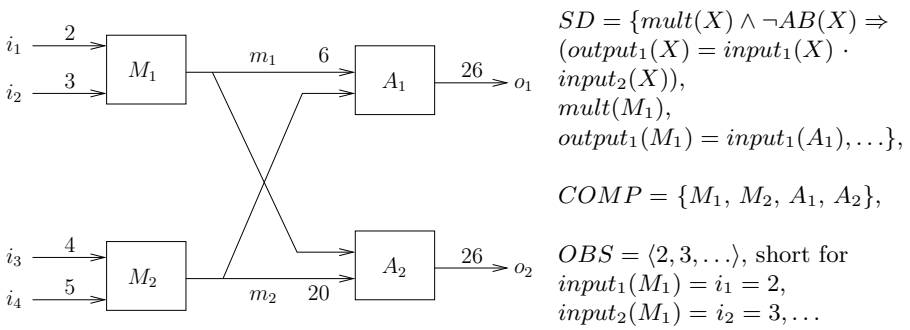

Fig. 1. A simple system description for a multiplier and adder example.

"abnormal" parts of the system: $\neg AB(c)$ denotes a component which works according to its specification, while $AB(c)$ denotes an abnormal component. A diagnosis can now be defined w. r. t. to these predicates which help explain an observed behaviour.

Definition 2 (Diagnosis). *A diagnosis for a system $S = (SD, COMP)$ is a minimal set $\Delta \subseteq COMP$ such that*

$$SD \cup OBS \cup \{AB(c) \mid c \in \Delta\} \cup \{\neg AB(c) \mid c \in COMP \backslash \Delta\}$$

is consistent.

Proposition 1. *$\emptyset$ is a diagnosis (and the only diagnosis) for $(SD, COMP, OBS)$, iff*

$$SD \cup OBS \cup \{\neg AB(c) \mid c \in COMP\}$$

is consistent, i. e. iff the observation does not conflict with what the system should do if all its components were behaving correctly. (For a proof, see [19, § 3].)

Using this definition and continuing with what is presented in Ex. 1, it is self evident that substituting o_1 with anything but 26 will lead to the conclusion $\Delta = \{A_1\}$, i. e. $\neg AB(M_1)$, $\neg AB(M_2)$, $\neg AB(A_2)$, and $AB(A_1)$.

Definition 3 (Conflict Set). *A conflict set for $(SD, COMP, OBS)$ is a set $\{c_i, \ldots, c_j\} \subseteq COMP$ with $1 \leq i \leq j$ such that*

$$SD \cup OBS \cup \{\neg AB(c_i), \ldots, \neg AB(c_j)\}$$

is inconsistent.

Hence, $\{M_1, M_2, A_1, A_2\}$, would be a conflict set for our example, given $o_1 \neq 26$. Further, a conflict set for $(SD, COMP, OBS)$ is called *minimal*, iff no proper subset of it is a conflict set for $(SD, COMP, OBS)$ at the same time. That is, $\{A_1\}$ is a minimal conflict set. Diagnoses are then minimal conflict sets.

2.2 Reasoning with incomplete information

For those cases where obviously only a single component is at fault, finding minimal conflict sets seems a straightforward task under the gross assumption that OBS contains all relevant in- and outputs of the system, or its respective diagnosis model. However, in practice, one cannot always rely on the availability of complete information, but rather — and more realistically — on a *black* or *grey box* view yielding partial information.

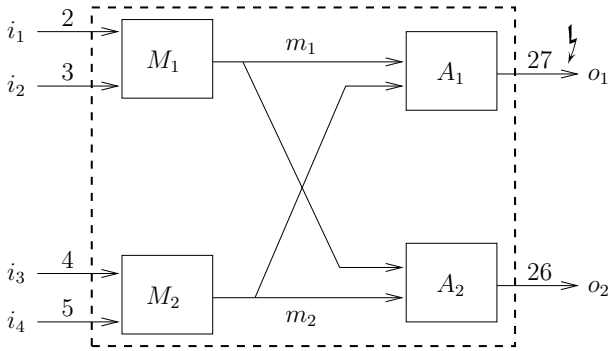

Fig. 2. Example of a black/grey box view where the values for m_1 and m_2 cannot be observed. According to qualitative measures, o_1 is faulty.

Example 2. If the network depicted in Fig. 1 is modified according to Fig. 2, there is already a significantly larger amount of possible conflict sets — implicitly depending on the values of $\{m_1, m_2\} \nsubseteq OBS$:

$$\{M_1, M_2\}, \{M_1, A_1\}, \{M_1\}, \{M_2, A_1\}, \{M_2\}, \{A_1\}, \{M_1, M_2, A_1\}.$$

Obviously, incomplete information creates a lot of diagnosis candidates, but as the above example demonstrates, these are entangled with assumptions regarding the missing elements of OBS. In other words, $\{M_2\}$ is a conflict set, iff the assignment for at least one unobservable connection contradicts the system specification according to the set SD, e. g. $m_2 \neq 20$. ◊

3 A propositional solution

Of course, the traditional diagnosis approaches such as applications of default logic, or abduction, although specifically tailored for dealing with incomplete and inconsistent information, face increasing difficulties the more in- and outputs remain unobservable. Reducing this problem to propositional logic, however, suits this situation perfectly, given a number of prerequisites are fulfilled.

3.1 Combining qualitative and logical measures

Let us assume we have two different models of the system under consideration, *a)* a qualitative model for reasoning about in- and output values of components, and *b)* an abstract model representing only causality; we can then use predicates that indicate whether an observation has been correct, or in error: $ok(m_1)$, for instance, would indicate that a result of M_1 is correct according to the qualitative assertion. Such "micro-evaluations" are realistic in many real-world scenarios, e. g. where sensor values are checked and compared, sometimes even multiple times to rule out tampered results due to jitter.

Qualitative assertions are typically made by dedicated *monitors* which continuously interpret a system component's in- and output values w. r. t. aberrations from the specification. However, monitors are not part of SD themselves, but rather constitute safety properties which ought to be fulfilled by single components $c_i \in COMP$, respectively.

In our case, these monitors are represented as predicates. That is, if the predicate holds, an observed value is assumed correct, otherwise it hints to existing aberrations. For example, the result of the boolean monitor $\beta(output_1(M_1)) = ok(output_1(M_1))$ would indicate conformance of the observed value $output_1(M_1)$ to its specification.

Definition 4 (Symptom). *Let $S = (SD, COMP, OBS)$ be a system under diagnosis. The ok-predicate is then defined over a subset of all in- and outputs, of a component $c \in COMP$. A negative evaluation of $ok(i \in OBS)$ is then called a symptom for an error in S.*

In other words, a negative result of a boolean monitor does not necessarily indicate that the monitored component is at fault. It merely hints to the fact that *some* component is faulty as captured in the following proposition:

$$\neg ok(i \in OBS) \Rightarrow \exists_{c \in COMP} \neg AB(c)^2.$$

Unlike the AB-predicate which is defined only over $c_i \in COMP$, ok is defined w. r. t. to observable system values. This notion of a symptom is then used to contradict the merely causal system specification and in order to distil a finite set of negative AB-predicates; that is, faulty components.

3.2 Reduction of SD

An alternative and, foremost, only causal first order system description for Ex. 1 and 2 could be expressed as follows:

$$\begin{aligned} SD = \{ & ok(i_1) \wedge ok(i_2) \wedge \neg AB(M_1) \Rightarrow ok(m_1), \\ & ok(i_3) \wedge ok(i_4) \wedge \neg AB(M_2) \Rightarrow ok(m_2), \\ & ok(m_1) \wedge ok(m_2) \wedge \neg AB(A_1) \Rightarrow ok(o_1), \\ & ok(m_1) \wedge ok(m_2) \wedge \neg AB(A_2) \Rightarrow ok(o_2)\} \end{aligned}$$

[2] Of course, c may be the monitored component, but this reasoning is part of the deductive diagnosis process.

Assuming we can evaluate and thus know when at least some observables w. r. t. *ok* hold, we can rewrite SD in terms of boolean variables and without any predicates:

$$SD' = \{ok\_i_1 \wedge ok\_i_2 \wedge \neg AB\_M_1 \Rightarrow ok\_m_1,$$
$$ok\_i_3 \wedge ok\_i_4 \wedge \neg AB\_M_2 \Rightarrow ok\_m_2, \ldots\}$$

Notice, both system descriptions SD and SD' are now comprising only causal and structural information, contrary to the literature of consistency based diagnosis, where SD always includes the behavioural part which we have "sourced out" in a separate, qualitative model which the monitors are based upon. Notice, monitor generation itself is an active field of research and not directly scope of this paper (see § 5).

Obviously, the mapping, $\Phi : pred \in (PL(\Sigma) \rightarrow \mathbb{B}) \rightarrow var \in \mathbb{B}$, where $PL(\Sigma)$ denotes a first order predicate logic formula defined over a signature Σ, is straightforward: each respective predicate, *pred*, is mapped to exactly one distinctive variable var_i of type $\mathbb{B}$.

3.3 Hitting sets and minimality

Finding conflict sets due to contradictions between expected and observed behaviour is crucial for failure diagnosis. However, in accordance to Definition 2, only the *minimal* diagnoses are of real, practical value. For this purpose, Reiter proposes a hitting set algorithm which constructs a so called *HS-tree* [19] that carries the minimal diagnoses, such that no diagnosis which is already included as a subset of a previously found diagnosis is chosen.

Formally, the problem addressed by Reiter's algorithm is as follows: a collection of non-empty sets $\mathcal{C} = \{C_1, \ldots, C_n\}$ of a set C, representing conflicts, is given. A hitting set (or transversal) of $\mathcal{C}$ is a subset $H \subseteq C$ that meets every set in the collection $\mathcal{C}$. We call a hitting set minimal, if no proper subset of H is a hitting set. More in depth information on the algorithm, specific optimisations, and analyses may also be found in § 5.

Despite well known improvements [8, 19, 14], the minimal hitting set problem remains generally $\mathcal{NP}$-complete, and it is practically undesirable to perform a thorough analysis when applied to diagnosis. Therefore, the chosen approach of this paper is to define a maximum "failure threshold" instead which is reflected in the diagnosis algorithm laid out in § 4.1. A diagnosis is then determined based upon *minimal cardinality* of occurring AB-predicates in the solution set, rather than upon the theory of set inclusion. Essentially, this allows us to improve on the determination of conflicts and still yields practically relevant diagnoses using the same algorithm; no subsequent procedure for minimalisation is required.

4 LSAT

In the previous section we have demonstrated how diagnosis related tasks such as fault isolation and conflict set minimalisation are basically reducible to propositional logic, under the premise that a qualitative model, which can be used to

monitor symptoms, is available. Although the complexity of the involved decision problems has not shrunk to polynomial time, propositional (system) models exhibit the advantage of being manageable by using SAT-solvers.

The purpose of a SAT-solver is to accept a formula, P, in clause normal form (CNF), and to return a variable assignment $\{\alpha(p_1 \in P), \ldots, \alpha(p_n \in P)\}$, such that P evaluates to *true*. If no such assignment can be found, P is not satisfiable.

In recent years, the area of SAT-solving has advanced dramatically: CNF formulas of hundreds of thousands or even millions of literals can now be handled by state-of-the-art solvers, such as (z)Chaff [17], or SATO [24] to name just two of the most popular solvers. These programs more or less are based on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [5] which constructs semantic trees of CNF formulas. Normally, DPLL is a "destructive" algorithm in a sense that it recursively splits the semantic tree — based on some heuristics — and descends until a valid assignment has been found; many SAT-solvers indeed work this way.

In contrast, LSAT[3] is a SAT-solver which uses a non-destructive implementation of DPLL based on mutually linked lists of atoms and clauses (see Fig. 3) where variable assignments are not recursively pushed on the runtime stack, but are encoded in the global data structure itself, similar to [13, 15]. This way, LSAT comes up with more than one truth assignment, if applicable.

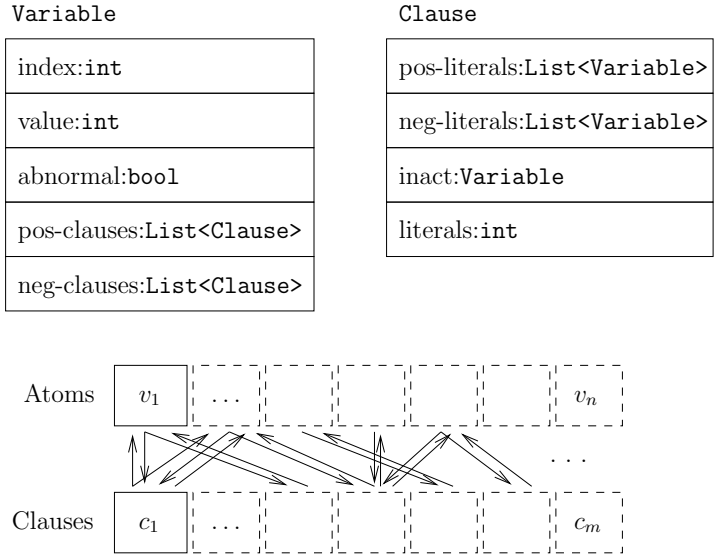

Fig. 3. LSAT's internal CNF representation visualised with v_i:`Variable` and c_i:`Clause`.

[3] LSAT has been released under the GPL open source license and is available in terms of C++ code from the author's home page at `http://home.in.tum.de/~baueran/lsat/`.

In general, other solvers may be modified likewise, e. g. by explicitly marshalling the data on the runtime stack with each truth assignment, but by using globally linked lists, the determination of a single solution set is equal in terms of memory consumption to finding all possible solution sets. Thus, having a single global data structure reduces the overall space complexity of the algorithm: only linear space for the main data structure is required.

The flag 'abnormal' denotes a component $c \in COMP$, and 'inact' contains a pointer to the variable which inactivated the current clause (`NULL` otherwise). Initially all clauses are active, i. e. no truth assignment has been made. 'pos-clauses' (resp. 'neg-clauses') is a list of pointers to clauses where the variable occurs with positive sign. 'pos-literals' (resp. 'neg-literals') is a list of pointers to variables which occur with positive sign in the clause. The rest is self explaining; further details can be found in the implementation.

4.1 Computing single and multiple fault diagnoses

Due to its non-destructive nature and the linear space complexity, LSAT is well suited for performing the diagnosis task based on propositional models. More so, if LSAT is able to conclude one diagnosis for a model, it is able to conclude all possible diagnoses, i. e. there is no need for a subsequent access to a theorem prover in order to determine conflict sets, or the likes.

In contrast, the diagnosis related literature frequently proposes the so called *single fault assumption* for two reasons: *a)* it is often realistic to assume merely single components at fault, rather than a total failure of a whole set of components, and *b)* most operations in non-monotonic reasoning approaches are sufficiently expensive such that the occurrence of a single fault is often used as (premature) exit condition. Naturally, the counterpart of the single fault assumption is the *multiple fault assumption* [6, 12].

Generally speaking, LSAT supports the n-fault assumption (where $n \geq 0$) and uses the 'abnormal' flag in the data type `Variable` in order to determine which diagnoses are useful, i. e. which are of a minimal cardinality w. r. t. the number of faulty components. That is, 'abnormal' determines the set of potential faults that are not symptoms (see Definition 4).

If defined, LSAT adheres to the n-fault assumption by keeping track of the positively assigned "AB-atoms/predicates" and by cutting off the semantic tree iff $AB(c_i)+\ldots+AB(c_j) > n$. Trivially, $n = 1$ selects the single fault assumption, $n > 1$ the multiple fault assumption.

Example 4. Given a system and observations $S = (SD, COMP, OBS)$, as it is depicted in Fig. 2, and an according propositional logic system description similar to Ex. 3, we may use the presented concepts so far to explain $o_1 \in OBS = 27$ as follows:

$$\begin{aligned} SD'' = \{&\neg[1/ok\_i_1] \vee \neg[2/ok\_i_2] \vee [9/AB\_M_1] \vee [5/ok\_m_1], \\ &\neg[3/ok\_i_3] \vee \neg[4/ok\_i_4] \vee [10/AB\_M_2] \vee [6/ok\_m_2], \end{aligned}$$

$$\neg[5/ok\_m_1] \vee \neg[6/ok\_m_2] \vee [11/AB\_A_1] \vee [7/ok\_o_1],$$
$$\neg[5/ok\_m_1] \vee \neg[6/ok\_m_2] \vee [12/AB\_A_2] \vee [8/ok\_o_2], \ldots\}$$

SD'' is obtained by applying $\Phi(SD)$ and performing a subsequent CNF conversion which can be achieved in polynomial time [18]. Hence, SD'' represents the causal dependencies as well as possible states of our system using only natural numbers for each respective variable; 9, 10, 11, and 12 represent components in S. Although of no particular semantic value, the substitutions with natural numbers will be necessary for expressing SD'' in terms of an extended DIMACS format.

2-fault assumption: A snapshot of the semantic tree decision procedure for our example is depicted in Fig. 4. In each "step", the variables of (SD, $COMP$, OBS) are assigned and the set of models expanded. Here, $\alpha(AB(11)) = 1 \equiv \alpha(11) = 1$ violates the 2-fault assumption since $\alpha(9) = 1$ and $\alpha(10) = 1$ on the same branch. The rounded arrow indicates one backtracking step in order to continue the algorithm using an alternative assignment, $\alpha(11) = -1$, i. e. $\neg AB(11)$. In other words, the n-fault assumption is the pruning criterion for the semantic tree procedure.

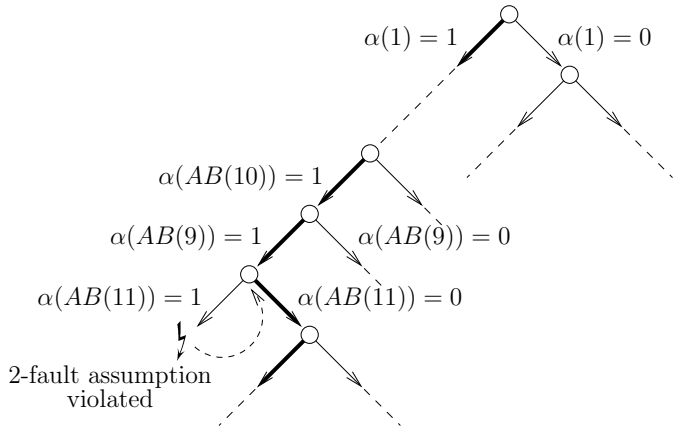

Fig. 4. Semantic tree for the system of Ex. 2: $\alpha(x)$ are the truth assignments; bold arrows indicate a valid assignment path, while the left-most path shows a violation of the 2-fault assumption.

Using LSAT's extended DIMACS format, this example could be encoded and automatically solved as follows:

01	`p cnf 12 18`	Standard DIMACS header.
02	`9 -1 -2 5`	SD: causal dependencies of S.
03	`10 -3 -4 6`	$(10 \vee -3 \vee -4 \vee 6)$
04	`11 -5 -6 7`	$\bigwedge$ $(11 \vee -5 \vee -6 \vee 7)$

```
05       12  -6 -5 8          ⋀  ...
06       -5  -9
07       -6 -10
08       -7 -11
09       -8 -12
10   a    9  10 11 12        COMP: the directive a defines the components in S.
11        1                  OBS: ok(1)
12        2                  ...
13        3
14        4
15       -7                  Symptom: ¬ok(7).  (Notice, {5, 6} ⊈ OBS.)
16        8
17       -9  9               Our hypotheses, i. e. all components may either be
18      -10 10                normal, or abnormal.
19      -11 11
20      -12 12
```

Similarly to the procedure shown in Fig. 4, LSAT is then able to determine all models for S with at most two faulty components; symptoms are underlined, "real" faults framed:

```
01   [9] [10] -11 -12 8 -7 -6 -5 4 3 2 1
02   [9] -10 [11] -12 8 -7 6 -5 4 3 2 1
03   [9] -10 -11 -12 8 -7 6 -5 4 3 2 1
04   -9 [10] [11] -12 8 -7 -6 5 4 3 2 1
05   -9 [10] -11 -12 8 -7 -6 5 4 3 2 1
06   -9 -10 [11] -12 8 -7 6 5 4 3 2 1
```

Each of these six results encodes one valid truth assignment for S, such that the contradicting observation, i. e. $o_1 = 27$ can be explained under the assumption that no more than two components are responsible for the failure. Result #1, for instance, assumes $AB(9) \wedge AB(10) \wedge \neg AB(11) \wedge \neg AB(12)$. ◊

4.2 Evaluation

Clearly, the emphasis of LSAT is on model-based reasoning and diagnosis, rather than trying to outperform programs like SATO or (z)Chaff. However, LSAT does contain a couple of optimisations such as *unit propagation* and an implementation of the *purity rule* in order to deal with far bigger examples than shown in this paper so far. In order to elaborate on the feasibility of the chosen approach, a number of more or less standard benchmarks originating from the area of circuit design are shown in this section. This ISCAS set of benchmark circuits is widely used by the ECAD community for testing several digital design tools. The respective combinatorical tests range from several hundred to ca. 20,000 "components" and ca. 60,000 clauses.

Table 1. Modified ISCAS'89 benchmarks under the n-fault assumption.

				∞-fault		5-fault	
Name:	$\#COMP$:	#Var.:	#Cl.:	#Steps:	CPU:	#Steps:	CPU:
s208.1	66	122	389	84	0.17 sec	60	0.25 sec
s298	75	136	482	27	0.11 sec	58	0.32 sec
s444	119	205	714	20	0.18 sec	105	0.91 sec
s526n	140	218	833	–	timeout	295	0.23 sec
s820	256	312	1,335	–	timeout	562	0.59 sec
s1238	428	540	2,057	38	0.97	262	0.21 sec
s13207	2,573	8,651	27,067	–	timeout	17	0.57 sec
s15850	3,448	10,383	33,189	–	timeout	41	0.17 sec
s35932	12,204	17,828	60,399	2,339	11.16 sec	29	0.21 sec

Table 1 summarises the results of applying a selection of tests to LSAT on a Pentium 4 architecture with 512 MB of RAM using either the 5-fault assumption, or no restriction at all. If a test could not be finished within 60 seconds, it was considered to be a timeout.

Not surprisingly, the numbers substantiate the appropriateness of the presented concepts. Four tests could not be finished using the ∞-fault assumption, while LSAT had no problems solving these tasks when constrained to five faults. The variance between CPU time and the performed number of algorithmic steps can be explained by the heuristic approach and variantly efficient accessor functions in the LSAT tool.

5 Related work

There are recent works which have used SAT-solvers in order to diagnose and debug the design of digital circuits, e. g. [1, 23]. However, research in this area treats the solvers foremost as mere tools, disregarding *a)* the cardinality of the models, and *b)* the adaption of the underlying algorithms and concepts. More so, in the "digital world", there is no need to reduce first order problems to propositional logic, since a circuit is already digestible by the solver as is.

Closer to the ideas presented in this paper is research undertaken by Baumgartner et al. such as [2], for instance. They describe the DRUM-2 system which is based on first order *hypertableaux*, and can be used for model-based reasoning in the above sense. Their system is capable of finding minimal diagnoses based on "abnormal components", similar as it is implemented in LSAT. But the (system) descriptions used by Baumgartner et al. are first order, and the results suggest that the hypertableaux-based algorithms, therefore, do not scale to the same extent as SAT-solving does, especially in light of recent developments in this area [17]. In fact, the benchmarks indicate that the propositional approach taken by LSAT performs quite well even without optimising any heuristics which would help determining the most suitable free variable in a given CNF formula for each computational step (e. g. similar benchmarks for hypertableaux with

only a 2-fault assumption applied are presented in [3] where, regardless of the number of clauses (ca. 1,600–16,000), no solution set could be determined in less than two seconds, and a notably high number of computational steps (between 160–1,500,000)).

Of course, first order systems are far more expressive than LSAT models, but usually less efficient for the diagnosis algorithms. This becomes especially apparent in non-monotonic logic. Recall, a logic is called monotonic if the truth of a proposition does not change when new information, i. e. axioms, are added to the knowledge base. In contrast, a logic is non-monotonic, if the truth of a proposition may change when new information is added to or old information is deleted from the base. Abduction, default logic, and the *closed world assumption* of Prolog and CLP systems are examples for applications of non-monotonic logic.

Recent complexity results for abductive reasoning and default logic as originally considered for diagnosis by McCarthy and Reiter (amongst others) indicate that only specific subsets and "sub-problems" can be dealt with in an efficient manner [7, 4]. Diagnosis based on these prominent concepts remains subject to restrictions which do not exist when using a reduced (but less expressive) propositional model of a diagnosable system along with suitable monitoring mechanisms.

Both diagnosability, i. e. strategic placement of sensors, and generation of monitors — often called observers — are active fields of research today; see, for example, [21], [11], [10], and [9]. Hence, the diagnosis approach shown in this paper should be considered an addition to these activities in order to complement the reasoning about system failure and corresponding causes for it.

6 Conclusions

This work has presented an efficient approach to model-based diagnosis based on k-satisfiability and models of minimal cardinality. The beauty of the proposed solution lies in its simplicity, because it combines the advantages of state-of-the-art SAT-solving and deals with large designs by pruning the search space according to user defined criteria, i. e. n-fault assumption on AB-predicates.

The evaluation in § 4.2 hints to the scalability of this approach and its potential impact on system diagnosis and monitoring. More so, the algorithms used require at most polynomial space and can be examined and tested in detail using the freely available implementation of the presented LSAT program (see p. VIII).

Unlike many other SAT-solvers, LSAT is able to come up with *all* models (under the n-fault assumption) yielding sensible conflict sets, hence diagnoses. Although this notion of minimality is not correlated to Reiter's original HS-tree [19], thus set theory, it provides for technically useful diagnoses.

Depending on the system under consideration, the generation of monitors, however, may vary dramatically. Possible realisations may be purely in software, e. g. monitoring threads and middleware, or mostly in hardware, e. g. intelligent sensors as increasingly used in the automotive domain, for instance. Although this paper has focussed mainly on aspects of diagnosis, the combination with

monitoring techniques provides potential for a broader application of these techniques, especially in scenarios where quality and safety properties are increasingly important, e. g. embedded systems.

Acknowledgements

The author thanks his colleagues Gernot Stenz and Reinhold Letz for insightful discussions regarding SAT-solving and for their comments on the technicalities of the implementation. Martin Leucker, Martin Wildmoser, and the anonymous referees provided valuable feedback on an earlier version of this paper.

References

1. M. Ali, A. Veneris, S. Safarpour, M. Abadir, R. Drechsler, and A. Smith. Debugging Sequential Circuits Using Boolean Satisfiability. In *5th International Workshop on Microprocessor Test and Verification (MTV'04)*, Austin, 2004.
2. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Tableaux for Diagnosis Applications. Technical Report 23–96, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 1996.
3. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In M. E. Pollack, editor, *15th International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 460–465, Nagoya, 1997. Morgan Kaufmann.
4. R. Ben-Eliyahu-Zohary. Yet some more complexity results for default logic. *Artificial Intelligence*, 139(1):1–20, 2002.
5. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
6. J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.
7. T. Eiter and T. Lukasiewicz. Complexity results for explanations in the structural-model approach. *Artif. Intell.*, 154(1-2):145–198, 2004.
8. R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41:79–88, 1989.
9. J. Håkansson, B. Jonsson, and O. Lundqvist. Generating online test oracles from temporal logic specifications. *STTT*, 4(4):456–471, 2003.
10. K. Havelund and G. Rosu. Monitoring Java Programs with Java PathExplorer. *Electronic Notes Theoretical Computer Science*, 55(2), 2001.
11. K. Havelund and G. Rosu. Synthesizing Monitors for Safety Properties. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 342–356, 2002.
12. Y. C. Kim, K. K. Saluja, and V. D. Agrawal. Multiple faults: Modeling, simulation and test. In *Proceedings of the 2002 conference on Asia South Pacific design automation/VLSI Design*, page 592. IEEE Computer Society, 2002.
13. C. M. Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In G. Smolka, editor, *CP*, volume 1330 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 1997.
14. L. Li and Y. F. Jiang. Computing minimal hitting sets with genetic algorithms. *Algorithmica*, 32(1):95–106, 2002.

15. I. Lynce and J. P. Marques-Silva. Efficient data structures for backtrack search SAT solvers. In *Proceedings of the 5th International Symposium on the Theory and Applications of Satisfiability Testing (SAT)*, May 2002.
16. J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.
17. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, 2001.
18. A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science B.V., 2001.
19. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
20. P. Torasso, L. Console, L. Portinale, and D. T. Dupré. On the role of abduction. *ACM Comput. Surv.*, 27(3):353–355, 1995.
21. Y. L. Traon, F. Ouabdesselam, C. Robach, and B. Baudry. From diagnosis to diagnosability: axiomatization, measurement and application. *J. Syst. Softw.*, 65(1):31–50, 2003.
22. F. Vatan. The complexity of the diagnosis problem. Technical Support Package (TSP) NPO-30315, NASA Jet Propulsion Laboratory, Apr. 2002.
23. A. Veneris. Fault Diagnosis and Logic Debugging Using Boolean Satisfiability. In *Fourth International Workshop on Microprocessor Test and Verification Common Challenges and Solutions*, Austin, Texas, May 2003.
24. H. Zhang. SATO: an efficient propositional prover. In *Proceedings of the International Conference on Automated Deduction (CADE'97), volume 1249 of LNAI*, pages 272–275, 1997.