Semantically Guided Theorem Proving for Diagnosis Applications

Peter Baumgartner
Univ. Koblenz
Inst. f. Informatik
`peter@infko.uni-koblenz.de`

Peter Fröhlich
Universität Hannover
`froehlich@kbs.uni-hannover.de`

Ulrich Furbach
Univ. Koblenz,
Inst. f. Informatik
`uli@infko.uni-koblenz.de`

Wolfgang Nejdl
Universität Hannover
`nejdl@kbs.uni-hannover.de`

Abstract

In this paper we demonstrate how general purpose automated theorem proving techniques can be used to solve realistic model-based diagnosis problems. For this we modify a model generating tableau calculus such that a model of a correctly behaving device can be used to guide the search for minimal diagnoses. Our experiments show that our general approach is competitive with specialized diagnosis systems.

1 Introduction

In this paper we will demonstrate that model generation theorem proving is very well suited for solving consistency-based diagnosis tasks. More precisely, we want to emphasize two aspects:

i) Theorem proving techniques are very well applicable to realistic diagnosis problems, as they are contained in diagnosis benchmark suites.

ii) Semantic information from a specific domain, can be used to significantly improve performance of a theorem prover.

According to Reiter ([10]) a simulation model of the technical device under consideration is constructed and is used to predict its normal behavior. By comparing this prediction with the actual behavior it is possible to derive a diagnosis.

This work was motivated by the study of the diagnosis system DRUM-2 [5; 9]. The basic idea is to start with an initial model of a correctly functioning device. This model is revised, whenever the actual observations differ from the predicted behavior.

We will use a proof procedure, which is an implementation of the hyper tableaux calculus presented in [1]. We adapt the idea from DRUM-2 to this tableaux calculus, which yields *semantic hyper tableaux*. The resulting system approximates the efficiency of the DRUM-2 system.

The use of semantics within theorem proving procedures have been proposed before. There is the well-known concept of semantic resolution ([3]) and, more recently, there are approaches by Plaisted ([4] and Ganzinger ([6]). Plaisted is arguing strongly for the need of giving semantic information for controlling the generation of clauses in his instance-based proof procedures, like hyper-linking. Ganzinger and his coworkers are presenting an approach where orderings are used to construct models of clause sets. Indeed, they even relate their approach to SATCHMO-like theorem proving, which is an instance of the hyper tableau calculus. However, the semantics have to be given by orderings or alternatively, by Horn subsets of the set of clauses. In cases where the initially given semantics is not compatible with orderings or is not expressible by Horn subsets it is unclear how to proceed. We will show, that in the case of diagnosis an initial semantics, which is naturally given by a model of the correct behavior of the device under consideration, can improve performance significantly. Our proof procedure does not impose any restrictions on these initial models.

We assume that the reader is familiar with the basic concepts of propositional logic. *Clauses*, i.e. multisets of literals, are usually written as the disjunction $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$ or as an implication $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ ($m \geq 0,\ n \geq 0$). With $\overline{L}$ we denote the complement of a literal L. Two literals L and K are *complementary* if $\overline{L} = K$.

2 Model–Based Diagnosis

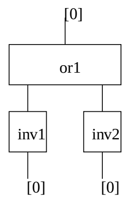
In model-based diagnosis [10] a simulation model of the device under consideration is used to predict its normal behavior, given the observed input parameters. This approach uses an logical description of the device, called the system description (SD), formalized by a set of first–order formulas. The system description consists of a set of axioms characterizing the behavior of system components of certain types. The topology is modeled separately by a set of facts. The diagnostic problem is described by system description SD, a set $COMP$ of components and a set OBS of observations (logical facts). With each component we associate a behavioral mode: $Mode(c, Ok)$ means that component c is behaving correctly, while $Mode(c, Ab)$ (abbreviated by $Ab(c)$) denotes that c is faulty.

Definition 2.1 (Reiter 87) A *Diagnosis* of $(SD, COMP, OBS)$ is a set $\Delta \subseteq COMP$, such that $SD \cup OBS \cup \{Mode(c, Ab) | c \in \Delta\} \cup \{\neg Mode(c, Ab) | c \in COMP - \Delta\}$ is consistent. Δ is called a *Minimal Diagnosis*, iff it is the minimal set (wrt. $\subseteq$) with this property. □

The set of all minimal diagnoses can be large for complex technical devices. Therefore, stronger criteria than minimality are often used to further discriminate among the minimal diagnoses. These criteria are usually based on the probability or cardinality of diagnoses. In the remainder of this paper we will use restrictions on the cardinality of diagnoses. We say that a diagnosis satisfies the *n-fault assumption* iff $|\Delta| \leq n$. Other minimality criteria which prefer e.g. more probable diagnoses are conceivable as well, but are not treated in the present paper.

A widely used example of the n-fault assumption is the 1-fault assumption or *Single Fault Assumption*. Many specialized systems for technical diagnosis have the Single Fault Assumption implicit and are unable to handle multiple faults.

In model–based diagnosis systems the Single Fault Assumption can be activated explicitly in order to provide more discrimination among diagnoses and to speed up the diagnosis process.



As a running example consider the simple digital circuit on the right consisting of an or–gate ($or1$) and two inverters ($inv1$ and $inv2$). The system description SD is given by the following propositional clauses:[1].

OR1: $ab(or1), high(or1, i1), high(or1, i2) \leftarrow high(or1, o)$
$ab(or1), high(or1, o) \leftarrow high(or1, i1)$
$ab(or1), high(or1, o) \leftarrow high(or1, i2)$

INV1: $ab(inv1) \leftarrow high(inv1, o), high(inv1, i)$
$ab(inv1), high(inv1, i), high(inv1, o) \leftarrow$

INV2: $ab(inv2) \leftarrow high(inv2, o), high(inv2, i)$
$ab(inv2), high(inv2, i), high(inv2, o) \leftarrow$

$high(or1, i1) \leftarrow high(inv1, o) \quad high(or1, i2) \leftarrow high(inv2, o)$
$high(inv1, o) \leftarrow high(or1, i1) \quad high(inv2, o) \leftarrow high(or1, i2)$

We observe that both inputs of the circuit have low voltage and the output also has low voltage, i.e. the clause set of OBS is given by $\{\leftarrow high(inv1, i), \leftarrow high(inv2, i), \leftarrow high(or1, o)\}$.

The expected behavior of the circuit given that both inputs are low would be high voltage at the outputs of both inverters and consequently high voltage at the output of the or–gate. This model of the correctly functioning device, namely $I_0 = \{high(inv1, o), high(inv2, o), high(or1, i1), high(or1, i2), high(or1, o)\}$, can be computed very efficiently even for large devices by domain–specific tools, e.g. circuit simulators.

[1]These formulas can be obtained by instantiating a first order description of the gate functions with the structural information. For instance, the clauses for the OR1 gate stem from the formula $\forall OR\_gate : (\neg ab(OR\_gate) \rightarrow (high(OR\_gate, o) \leftrightarrow (high(OR\_gate, i1) \vee high(OR\_gate, i2))))$

3 Hyper Tableaux Calculus

In [1] a variant of clausal normal form tableaux called "hyper tableaux" is introduced. We briefly recall the ground version here.

From now on $\mathcal{S}$ always denotes a finite ground clause set, and Σ denotes its signature, i.e. the set of all predicate symbols occurring in it. We consider finite ordered trees T where the nodes, except the root node, are labeled with literals. In the following we will represent a branch b in T by the sequence $b = L_1, \ldots, L_n$ ($n \geq 0$) of its literal labels, where L_1 labels an immediate successor of the root node, and L_n labels the leaf of b. The branch b is called *regular* iff $L_i \neq L_j$ for $1 \leq i, j \leq n$ and $i \neq j$, otherwise it is called *irregular*. The tree T is *regular* iff every of its branches is regular, otherwise it is *irregular*. The set of *branch literals* of b is $\text{lit}(b) = \{L_1, \ldots L_n\}$. For brevity, we will write expressions like $A \in b$ instead of $A \in \text{lit}(b)$. In order to memorize the fact that a branch contains a contradiction, we allow to label a branch as either *open* or *closed*. A tableau is *closed* if each of its branches is closed, otherwise it is *open*.

Definition 3.1 (Hyper tableau) A literal set is called *inconsistent* iff it contains a pair of complementary literals, otherwise it is called *consistent*. *Hyper tableaux* for $\mathcal{S}$ are inductively defined as follows:

Initialization step: The empty tree, consisting of the root node only, is a hyper tableau for $\mathcal{S}$. Its single branch is marked as "open".

Hyper extension step: If (1) T is an open hyper tableau for $\mathcal{S}$ with open branch b, and (2) $C = A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is a clause from $\mathcal{S}$ ($m \geq 0$, $n \geq 0$), called *extending clause* in this context, and (3) $\{B_1, \ldots, B_n\} \subseteq b$ (equivalently, we say that C is *applicable to* b) then the tree T' is a hyper tableau for $\mathcal{S}$, where T' is obtained from T by *extension of b by C*: replace b in T by the *new* branches

$$(b, A_1) \ldots, (b, A_m), (b, \neg B_1) \ldots, (b, \neg B_n)$$

and then mark every inconsistent new branch as "closed", and the other new branches as "open". We say that a branch b is *finished* iff it is either closed, or else whenever C is applicable to b, then extension of b by C yields some irregular new branch. $\square$

The applicability condition of an extension expresses that *all* body literals have to be satisfied by the branch to be extended (like in hyper *resolution*). From now on we consider only regular hyper tableaux. This restriction guarantees that for finite clause sets no branch can be extended infinitely often.

Definition 3.2 (Branch Semantics) As usual, we represent an interpretation I for given domain Σ as the set $\{A \in \Sigma \mid I(A) = \mathit{true}, A \text{ atom}\}$. *Minimality* of interpretations is defined via set-inclusion.

Given a tableau with consistent branch b. The branch b is mapped to the interpretation $[\![b]\!]_\Sigma := \text{lit}(b)^+$, where $\text{lit}(b)^+ = \{A \in \text{lit}(b) \mid A \text{ is a positive literal }\}$. Usually, we write $[\![b]\!]$ instead of $[\![b]\!]_\Sigma$ and let Σ be given by the context. □

Figure 1 contains a hyper tableau for the clause set from our running example. Each open branch for the clause set $SD \cup OBS$ corresponds to a partial model. The highlighted model can be understood as an attempt to construct a model for the whole clause set, without assuming unnecessary ab-predicates. Only for making the clauses from $OR1$ true is it necessary to include $ab(or1)$ into the model. $high(or1, o)$ cannot be assumed, as this contradicts the observation $\leftarrow high(or1, o)$.

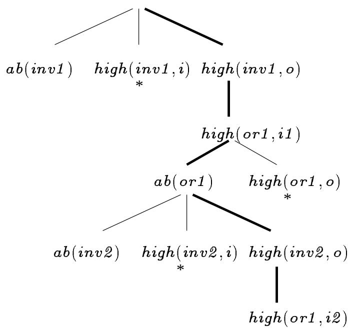


Figure 1: Hyper tableau.

A refutational completeness result for hyper tableaux was given in [1]. For our purposes of computing diagnosis (i.e. models), however, we need a (stronger) model completeness result:

Theorem 3.3 (Model Completeness of Hyper Tableaux.) *Let T be a hyper tableau for $\mathcal{S}$ such that every open branch is finished. Then, for every minimal model I of $\mathcal{S}$ there is an open branch b in T such that $I = [\![b]\!]$.*

This theorem enables us to compute in particular minimal diagnosis by simply collecting all Ab-literals along $[\![b]\!]$, because every minimal diagnosis must be contained in some minimal model of $\mathcal{S}$.

3.1 Lessons from the Specialized Diagnosis System DRUM-2

In order to make the generation of models efficient enough for the large benchmark circuits used in this paper, additional knowledge has to be used to guide this model generation. This is done by starting from a model of the correct behavior of the device SD and revising the model only where necessary. This idea of a "semantically guided " model generation has been introduced first in the DRUM-2 system [5; 9]. The basic idea of DRUM-2 is to start with a model of the correct behavior of the device under consideration, i.e. with an interpretation I_0, such that $I_0 \models SD \cup \{\neg Ab(c) | c \in COMP\}$. Then the system description SD is augmented by an observation of abnormal behavior OBS, such that the assumption that all components are working correctly is no longer valid. Thus, I_0 is no model of $SD \cup OBS$, however it is used to guide the search for models of $SD \cup OBS$.

Note, that the initial model I_0 usually already takes into account a part of the observations which correspond to the inputs of the device under consideration. These input values are needed to simulate the correct behavior of the device. In our example, the initial model reflects the fact that the inputs of the inverters both have low voltage (I_o was given at the end of Section 2).

4 Formalizing the Diagnosis Task with Semantic Hyper Tableaux

In this section we discuss how to incorporate initial interpretations into the calculus. Our first technique by *cuts* should be understood as the semantics of the approach; an efficient implementation by a *compilation technique* is presented afterwards.

4.1 Initial Interpretations via Cuts

The use of an initial interpretation can be approximated in the hyper tableau calculus by the introduction of an additional inference rule, the *atomic cut rule*.

Definition 4.1 The inference rule **Atomic cut (with atom A)** is given by: if

T is an open hyper tableau for $\mathcal{S}$ with open branch b,

then the literal tree T' is a hyper tableau for $\mathcal{S}$, where T' is obtained from T by extension of b by $A \vee \neg A$ (cf. Def. 3.1). □

Note that in regular tableaux it cannot occur that a cut with atom A is applied, if either A or $\neg A$ is contained on the branch. As a consequence it is impossible to use the "same cut" twice on a branch.

We approximate initial interpretations by applying atomic cuts at the beginning of each hyper tableau:

Definition 4.2 An *initial tableau* for an interpretation I_0 is given by a regular tableau which is constructed by a applying atomic cuts with atoms from I_0 as long as possible. □

The branches of an initial tableau for an interpretation I_0 consist obviously of all interpretations with atoms from I_0. A part of the initial tableau for the initial interpretation I_0 given at the end of Section 2 is depicted in Figure 2. Note that the highlighted branch corresponds to the highlighted part in Figure 1 (a negative literal in an initial tableaux, such as $\neg high(or1, o)$, is represented implicitly in a hyper tableau as the *absence* of the complementary positive literal). If this branch is extended in successive hyper extension steps, the diagnosis $ab(or1)$, which was contained in the model from Figure 1 can be derived as well.

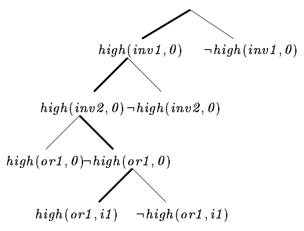


Figure 2: Initial tableau.

Definition 4.3 (Semantic Hyper Tableau – SHT) A *semantic hyper tableau* for I_0 and $\mathcal{S}$ is a hyper tableau which is generated according to Definition 3.1, except that the empty tableau in the initialization step is replaced by an initial tableau for I_0. □

It is easy to derive an open tableau starting from the initial tableau for I_0 in Figure 2, such that it contains the model from Figure 1.

Proposition 4.4 (Model Completeness of SHT) *Let T be a semantic hyper tableau for I_0 and $\mathcal{S}$, such that every open branch is finished. Then, for every minimal model I of $\mathcal{S}$ there is an open branch b in T such that $I = [\![b]\!]$.*

4.2 Initial Interpretations via Renaming

The just defined "semantical" account for initial interpretations via cut is unsuited for realistic examples. This is, because all the $2^n - 1$ possible deviations from the initial interpretation will have to be investigated as well. Hence, in this section we introduce a compilation technique which implements the deviation from the initial interpretation only by need.

Assume we have an initial interpretation $I_0 = \{a\}$ and a clause set which contains $b \leftarrow$ and $c \leftarrow a \wedge b$. By the only applicable atomic cut we get the initial tableau with two branches, namely $\{a\}$ and $\{\neg a\}$. The first branch can be extended twice by an hyper extension step, yielding $\{a, b, c\}$. The second branch can be extended towards $\{\neg a, b\}$. No more extension step is applicable to this tableau. Let T_{cut} be this tableau.

Let us now transform the clause set with respect to I_0, such that every atom from I_0 occurring in a clause is shifted to the other side of the $\leftarrow$ symbol and complemented. In our example we get the clause $c \vee \neg a \leftarrow b$; the fact $b \leftarrow$ remains, because b is not in I_0. Using $b \leftarrow$ we construct a tableau consisting of the single branch $\{b\}$, which can be extended in an successive hyper step by using the renamed clause. We get a tableau consisting of two branches $\{b, c\}$ and $\{b, \neg a\}$. Let T be that tableau. Now, let us interpret a branch in T as usual, except that we set an atom from I_0 to *true* if its negation is not contained in the branch. Under this interpretation the branch $\{b, c\}$ in T corresponds to the usual interpretation of $\{a, b, c\}$ in T_{cut}. Likewise, the second branch $\{b, \neg a\}$ in T_{cut} corresponds to the second model in T.

Note that by this renaming we get tableaux where atoms from I_0 occur only *negatively* on open branches; such cases just mean deviations from I_0. In contrast to the cut approach, these deviations are now brought into the tableau by need.

The following definition introduces the just described idea formally. Since we want to avoid unnecessary changes to the hyper calculus, a new predicate name neg_A instead of $\neg A$ will be used.

Definition 4.5 (I-transformation) Let $C = L_1 \vee \cdots \vee L_n$ be a clause and I be a set of atoms. The *I-transformation of C* is the clause obtained from C by replacing every positive literal A with $A \in I$ by $\neg neg\_A$, and by replacing every negative literal $\neg A$ with $A \in I$ by neg_A. The *I-transformation of $\mathcal{S}$*, written as $\mathcal{S}^I$, is defined as the I-transformation of every clause in $\mathcal{S}$. □

It is easy to see that every I-transformation preserves the models of a clause set, in the sense that every model for the non-transformed clause set constitutes a model for the transformed clause set by setting neg_A to *true* iff A is *false*, for every $A \in I$, and keeping the truth values for atoms outside of I. More formally we have:

Proposition 4.6 (Model Preservation of I-transformation) *For every interpretation J: $J \models \mathcal{S}$ iff $rename_I(J) \models \mathcal{S}^I$, where $rename_I(J)(neg\_A) = \overline{J(A)}$ iff $A \in I$, and else $rename_I(J)(A) = J(A)$.*

As explained informally above, the branch semantics of tableaux derived from a renamed, i.e. I-transformed clause set, is changed to assign *true* to every atom from I, unless its negation is on the branch. This is a formal definition:

$$\begin{aligned}[\![b]\!]^I = &(I \setminus \{A \mid neg\_A \in \text{lit}(b)\}) \cup \\ &(\text{lit}(b) \setminus \{neg\_A \mid neg\_A \in \text{lit}(b)\})\end{aligned}$$

The connection of semantic hyper tableaux to hyper tableaux and renaming is given by the next theorem.

Theorem 4.7 *Let T be a semantic hyper tableau for $\mathcal{S}$ and I where every open branch is finished; let T^I be a hyper tableau for the I-transformation of $\mathcal{S}$ where every open branch is finished. Then, for every open branch b^I in T^I there is an open branch b in T such $[\![b^I]\!]^I = [\![b]\!]$. The converse does not hold.*

The theorem tells us that with the renamed clause set we compute some deviation of the initial interpretation. The value of the theorem comes from the fact that the converse does not hold in general. That is, not every possible deviation is examined by naive enumeration of all combinations.

In order to see that the converse does not hold, take e.g. $\mathcal{S} = \{a \leftarrow\}$ and $I_0 = \{b\}$. There is only one semantic hyper tableau of the stated form, namely the one with the two branches $\{b, a\}$ and $\{\neg b, a\}$. On the other side, the I_0-transformation leaves $\mathcal{S}$ untouched, and thus the sole hyper tableau for $\mathcal{S}$ consists of the single branch $\{a\}$ with semantics $[\![\{a\}]\!]^{I_0} = \{a, b\}$. However, the semantics of the branch $\{\neg b, a\}$ in the former tableau is different. The set of models which are computed in T^I can be characterized by an ordering, $\geq_{I_0}$, which takes into account the deviation from the initial interpretation. If b_1 and b_2 are branches in a semantic hyper tableau for $\mathcal{S}$ and I, $b_1 \geq_{I_0} b_2$ iff $[\![b_1]\!] \cap I \supseteq [\![b_2]\!] \cap I$ and $[\![b_1]\!] \setminus I = [\![b_2]\!] \setminus I$.

Theorem 4.8 *Let T and T^I be given as in Theorem 4.7. Then, for every open $\geq_{I_0}$-maximal branch b of T there is an open branch b^I in T^I, such that $[\![b^I]\!]^I = [\![b]\!]$.*

To conclude, the semantic hyper tableau approach serves us as a tool for *understanding* the effects of initial interpretations. For *efficient computing* we rely on the next theorem:

Theorem 4.9 (Minimal Diagnosis Completeness) *Let I_0 be an interpretation such that $I_0 \cap \{Ab(c) \mid c \in COMP\} = \emptyset$, and T be a hyper tableau for $\mathcal{S}^{I_0}$ such that every open branch is finished. Then, for every minimal diagnosis $\Delta \subseteq COMP$ there is an open branch b in T such that*

$$\begin{aligned}\Delta &= \{c \in COMP \mid rename_{I_0}([\![b]\!]) \models Ab(c)\}\\ &= \{c \in COMP \mid Ab(c) \in b\} \ .\end{aligned}$$

Proof.(Sketch) We need Theorem 3.3 and Proposition 4.6: suppose Δ is a minimal diagnosis. Consider all atom sets M such that $M \cup \{Ab(c) \mid c \in \Delta\} \models \mathcal{S}$. As a consequence of the model correspondence expressed in Proposition 4.6, thereby using the facts that I_0 does not contain Ab-literals and that Δ is a minimal diagnosis, we can find an M such that $J := rename_{I_0}(M \cup \{Ab(c) \mid c \in \Delta\})$ is a minimal model for $\mathcal{S}^{I_0}$. Hence, by Theorem 3.3 this model, and in particular this diagnosis Δ, will be computed along some finished open branch b. Q.E.D.

5 Implementation and Experiments

We have implemented a proof procedure for the hyper tableaux calculus of [1], modified it slightly for our diagnosis task, and applied it to some benchmark examples from the diagnosis literature.

The Basic Proof Procedure. A basic proof procedure for the plain hyper tableaux calculus for the propositional case is very simple, and coincides with e.g. SATCHMO [8]. Initially let T be a tableau consisting of the root node only. Let T be the tableau constructed so far. *Main loop:* if T is closed, stop with "unsatisfiable". Otherwise select an open branch b from T (branch selection) which is not labeled as "finished" and select a clause $H \leftarrow B$ (extension clause selection) from the input clause set such that $B \subseteq b$ (applicability) and $H \cap b = \{\}$ (regularity check). If no such clause exists, b is labeled as "finished" and $[\![b]\!]$ is a model for the input clause set. In particular, the set of literals on b with predicate symbol Ab (simply called Ab-literals) constitutes a (not necessarily minimal) diagnosis. If every open branch is labeled as "finished" then stop, otherwise enter the main loop again.

In the diagnosis task it is often demanded to compute every (minimal) diagnosis. Hence the proof procedure does not stop after the first open branch is found, but only marks it as "finished" and enters the main loop again.

Adaption for the Diagnosis Task. While incorporation of the initial interpretation is treated by renaming predicates in the input clause set (Section 4), and thus requires no modification of the prover, implementing the minimality restriction is dealt with by the following new inference rule: "any branch containing $n+1$ (due to regularity necessarily pairwise different) Ab-literals is closed immediately".

Notice that this inference rule has the same effect as if the $\binom{|COMP|}{n+1}$ clauses $\leftarrow ab(C_1), \ldots, ab(C_{n+1})$ (for $C_i \in COMP$, $C_i \neq C_j$, where $1 \leq i,j \leq n+1$ and $i \neq j$) specifying the n-faults assumption would be added to the input clause set. Since even for the smallest example (c499) and the 1-fault assumption the clause set would blow up from 1600 to 60000 clauses, the inference rule solution is mandatory.

Computing Minimal Diagnoses From Theorem 4.9 we know that for each minimal diagnosis Δ the hyper tableau contains an open finished branch. However, there are also branches corresponding to non–minimal extensions of Ab. Since the number of non–minimal diagnoses is exponential in the number of components of the device, our goal is to avoid extension of branches which correspond to non–minimal diagnoses. This can be achieved by a combined iterative deepening/lemma technique. It can be basically realized by a simple outer loop around the just described proof procedure. The outer loop includes a counter $N = 0, 1, 2, \ldots$, which stands for the cardinality of the diagnosis computed in the inner loop. The *invariant* for the outer loop is the following: *all minimal diagnosis with cardinality $\leq N-1$ have been computed, say it is the set Δ_{N-1}, and every such diagnosis $\{c_1, \ldots, c_n\} \in \Delta_{N-1}$ has been added to the input clause set as a* lemma clause $\neg Ab(c_1) \vee \cdots \vee \neg Ab(c_n)$. Before entering the proof procedure in the inner loop we set $\Delta_N := \Delta_{N-1}$, and the proof procedure is slightly modified according to the following rule: whenever a finished open branch b is derived, a new diagnosis $\Delta_b = \{c \mid Ab(c) \in b\}$ is found. Hence we set $\Delta_N := \Delta_N \cup \{\Delta_b\}$ and add Δ_b as a lemma clause to the input clause set, as just described. No more modifications are necessary.

Notice that the lemma clauses are purely negative clauses and hence can be used to close a branch. Since we give preference to negative clauses, no diagnosis will be computed more than once, because as soon as a diagnosis is computed the first time, it is turned into a (negative) lemma clause which will be used to immediately close all branches containing the same diagnosis. Furthermore, we compute only *minimal* diagnosis as an immediate consequence of the iterative deepening over N: any branch containing a non-minimal diagnosis would have been closed by a lemma clause stemming from a diagnosis with strictly smaller cardinality, which must be contained in the input clause set due to the invariant for some value $< N$. Thus, the invariant holds for all N.

Although this procedure computing minimal diagnosis under the n-fault assumption is so simple, it has some nice prop-

Name	*# Gates*	*# Clauses*	*# Diagnosis*	*Time (sec.)*	*# Steps*	*All?*
C499	202	1685	2	5	3015	no
2-fault:			67	50	27323	yes
C880	383	2776	19	2	161	yes
C1355	546	3839	5	47	24699	no
2-fault:			5	2948	1284454	no
C2670	1193	8260	31	6	533	yes
C3540	1669	10658	3	10853	1473572	yes
C5315	2307	16122	5	13	3071	yes

Figure 3: ISCAS'85 Circuits and runtime results.

erties. First, it can be implemented easily by slight modifications to the basic hyper tableau proof procedure. Second, in the hyper tableau proof procedure we look at one branch at a time, which gives a polynomial upper bound for the memory requirement for the tableau currently constructed. Admittedly, there are possibly exponentially many minimal diagnosis wrt. $|COMP|$, but we assume that those will be kept anyway. Further, it is important to notice that we compute minimal models only wrt. the extension of the Ab-predicate, but not of minimal models wrt. all predicates. Since $|COMP|$ will be usually much smaller than the number of atoms in the translation $(SD, COMP, OBS)$ to propositional logic, computing minimal models wrt. all predicates would usually require considerably more memory (such an approach to minimal model reasoning was proposed in [2]).

Experiments. We implemented a prover according to the proof procedure as outlined above. It is a prover (written in SCHEME) for first-order logic, and thus carries some significant overhead for the propositional case. For our experiments we ran parts of the ISCAS-85 benchmarks [7] from the diagnosis literature. This benchmark suite includes combinatorial circuits from 160 to 3512 components. Table 3 describes the characteristics of the circuits we tested. The observations which were used can be obtained from the authors.

The results are summarized in Table 3. *# Clauses* is the number of input clauses stemming from the problem description. We ran our prover in 1-fault and 2-fault assumption settings. *Time* denotes proof time proper in seconds, and thus excludes time for reading in and setup (which is less than about 10 seconds in any case). The times are taken on a SparcStation 20. *# Steps* denotes the number of hyper extension steps to obtain the final tableau, and *# Diag* denotes the number of diagnosis. When two rows for a circuit are given, the upper one is for the 1-fault assumption, and the lower one is for the 2-fault assumption (recall that the 2-fault diagnosis *include* the 1-fault diagnosis). We emphasize that the results refer to the clause sets with renamed predicates according to Section 4. Without renaming, and thus taking advantage of the initial interpretation, in the 1-fault assumption only c499 was solvable (in 174 seconds); all other examples could not be solved within 2 hours, whatever flag settings/heuristic we tried!

In the “All?” column a “yes” entry means that there are no diagnosis with cardinality > 1, resp. > 2 for the 2-fault rows. That is, only for c1355 there are possible diagnosis with cardinality > 2 which we did not compute, for all other examples all minimal diagnosis were computed. We can detect if all minimal diagnoses were found by checking if some tableau branch was closed due to the n-fault assumption. To our impression, increasing n usually drastically increases the time to compute diagnosis.

6 Conclusions

In this paper we analyzed the relationship between logic-based diagnostic reasoning and tableaux based theorem proving. We showed how to implement diagnostic reasoning efficiently using a hyper tableaux based theorem prover. We identified the use of an initial model as the main optimization technique in the diagnostic reasoning engine DRUM-2 and showed how to apply this technique within a hyper tableaux calculus. We slightly modified an implementation of the basic hyper tableau calculus by augmenting it with a combined iterative deepening/lemma technique. This guarantees completeness for minimal diagnosis computation.

References

1. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *JELIA 96*. European Workshop on Logic in AI, Springer, LNCS, 1996.
2. F. Bry and A. Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. In P. Moscato, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 143–159. Springer, 1996.
3. C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
4. H. Chu and D. Plaisted. Semantically Guided First-Order Theorem Proving using Hyper-Linking. In A. Bundy, editor, *Automated Deduction — CADE 12*, LNAI 814, pages 192–206, Nancy, France, June 1994. Springer-Verlag.
5. P. Fröhlich and W. Nejdl. A model–based reasoning approach to circumscription. In *Proceedings of the 12th European Conference on Artificial Intelligence*, 1996.
6. H. Ganzinger, C. Meyer, and C. Weidenbach. Soft Typing for Ordered Resolution. Unpublished, 1996.
7. The ISCAS-85 Benchmarks. `http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.html`, 1985.
8. R. Manthey and F. Bry. SATCHMO: a theorem prover implemented in Prolog. In *Proc. 9th CADE*. Argonnee, Illinois, Springer LNCS, 1988.
9. W. Nejdl and P. Fröhlich. Minimal model semantics for diagnosis – techniques and first benchmarks. In *Seventh International Workshop on Principles of Diagnosis*, Val Morin, Canada, Oct. 1996.
10. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.