Merge Path Improvements for Minimal Model Hyper Tableaux

Peter Baumgartner, J.D. Horton, and Bruce Spencer

University of New Brunswick Fredericton, New Brunswick E3B 5A3, Canada
{baumgart,jdh,bspencer}@unb.ca

Abstract. We combine techniques originally developed for refutational first-order theorem proving within the clause tree framework with techniques for minimal model computation developed within the hyper tableau framework. This combination generalizes well-known tableaux techniques like complement splitting and folding-up/down. We argue that this combination allows for efficiency improvements over previous, related methods. It is motivated by application to diagnosis tasks; in particular the problem of avoiding redundancies in the diagnoses of electrical circuits with reconvergent fanouts is addressed by the new technique. In the paper we develop as our main contribution in a more general way a sound and complete calculus for propositional circumscriptive reasoning in the presence of minimized and varying predicates.

1 Introduction

Recently clause trees [6], a data structure and calculus for automated theorem proving, introduced a general method to close branches based on so-called *merge paths*. In this paper we bring these merge paths to tableaux for minimal model reasoning (e.g. [4, 11–13]). We use the framework of *hyper tableau* for this, which began with [2].

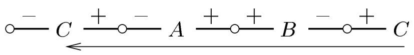
The paper [6] is devoted to *refutational* theorem proving. Merge paths allow branches to close earlier than it would be possible without them or when using merge paths to simulate known instances such as folding-down [8]. Expressed from the viewpoint of complement splitting [9], one advantage is that the splitting of literals can be *deferred*.

In this paper we advocate to use merge paths for *model computation* calculi. In addition to the advantages in the refutational framework, merge path allow one to partially re-use previously computed models instead of computing them again. To achieve this, new inference rules dealing with merge paths for minimal model computation are defined. In contrast to the purely refutational setting, these inference rules have to be applied with care, as termination is no longer a trivial property. Therefore, we give conditions for termination such that the central properties of minimal model soundness and minimal model completeness hold. More precisely, as our main result we develop such a calculus for the more general case of circumscriptive reasoning for minimized and varying predicates (Section 4). The minimal model completeness proof is given by a simulation of merge paths by atomic cuts (cf. Lemma 1 in Section 4). Viewed from this point, our approach can thus be seen as a more and generalized approach for a *controlled* integration of the cut rule for the purpose of minimal model computation.

The rest of this paper is structured as follows: first we briefly give the idea of *merge paths* as defined in [6]. This presentation should be sufficient to explain the subsequent motivation of the new calculus from the viewpoint of a certain problem encountered in diagnosis tasks. In Section 2 we bring merge paths into trees and define an ordering on merge paths. It is employed in Section 3 in the new calculus. In Section 4 we show how merge paths can be simulated by atomic cuts and, based on that, prove soundness and completeness. Section 5 discusses certain aspects of the calculus (memory requirements, atomic cuts vs. merge paths).

Clause trees. Merge paths are introduced and studied in [6] in the context of clause trees. Clause trees are a data structure that represent equivalence classes of resolution derivations. Merge paths are a unified inference rule and generalize the folding up/folding down technique of [8].

Clause trees consist of *clause* nodes and *atom* nodes. Clause nodes are indicated by a $\circ$. Every clause node N corresponds to some input clause $\lambda(N) = L_1 \vee \ldots \vee L_n$ as can be seen from the n emerging edges; these edges are labeled by the signs of the L_i's, and the atom parts of the L_i's can be found in the adjacent atom nodes. Clause trees are built in such a way that from every atom node exactly two edges with opposite sign emerge. This corresponds to a binary resolution inference. Here is an example:

Clause tree: 

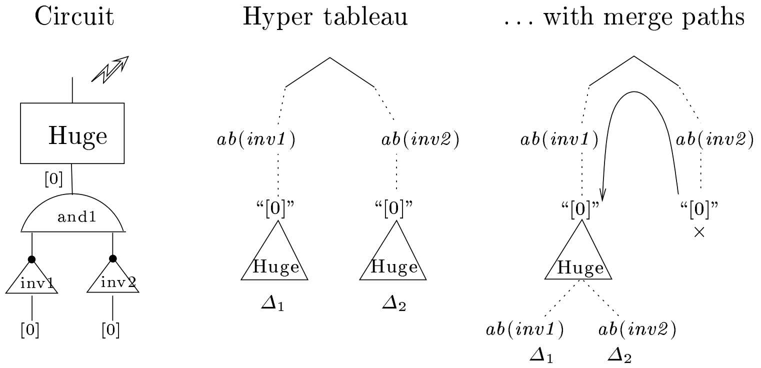
Clause set: $\leftarrow C \qquad C \leftarrow A \qquad A, B \leftarrow \qquad C \leftarrow B$

Now, in addition, *merge paths* can be drawn between equally labeled atom nodes, provided that the first and final edges are also equally labelled. In the preceeding figure, there is a merge path from the right C-node (called the *tail* of the merge path), to the left C-node (called the *head* of the merge path). The idea is "in order to find a proof at the tail of a merge path, look it up (copy it) from the head of the merge path". Thus, tail nodes are considered as proven and need no further extension. Thus a proof is a clause tree where every leaf is proven in this way or is a clause node.

Head nodes can be part of another merge path, and then there is a dependency of the nodes on the path on the head node. In this case the "lookup" of proofs is done recursively. In order to terminate this, cyclic dependencies must be excluded. The absence of cycles in a set of merge paths is referred to by the term "legal". Many of the results in [6] concerning legality and relation notions are derived as general properties of paths in trees. They thus can be readily applied to our case of hyper tableaux as well.

Motivation: A diagnosis application. We consider consistency-based diagnosis according to Reiter [15]. In this scenario, a model of a 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. More precisely, a *diagnosis* Δ is a (minimal) subset of the components of the device, such that the observed behavior is consistent with the assumption that exactly the components in Δ are behaving abnormally. Computing diagnosis can also be formalized as a circumscription problem.

The figure below depicts a hypothetical diagnosis scenario of an electrical circuit where merge paths are useful. The notation [0] in the left picture means that at this point the circuit is logical zero. The [0]'s at the bottom refer to input values of the actually observed behavior. The "Huge" box is meant to stand for a large circuit. The lightning at the output indicates that the predicted output is different from the actual output. Two possible diagnoses are $\Delta_1 = \{inv1\}$ and $\Delta_2 = \{inv2\}$. Notice that with declaring these as "abnormal", it is consistent to have [0] at the output of the *and*-gate.



Now, the crucial observation is that the computation of Δ_1 and Δ_2 show considerable redundancies. The hyper tableau based diagnosis approach of [1] would result in the tableau depicted in the middle of the figure. Diagnoses are read off from open branches by collecting the *ab*-literals found there. The triangles stand for sub-tableaux containing diagnoses of the "Huge" part. There are two open branches containing Δ_1 and Δ_2 respectively.

Notice that the "Huge" part has to be diagnosed twice *although for its diagnosis exactly the same situation applies, namely* [0] *at its input*. This is reflected by the nodes "[0]". Clearly, for the diagnosis of "Huge" it is irrelevant what caused the "[0]"-situation. The generalized underlying problem is well-known in the diagnosis community and is referred to as "reconvergent fanouts".

So, the symmetry hidden in this problem was not exploited. In fact, the merge path technique just realizes this. It is indicated in the right part of the figure above: after the diagnosis Δ_1 is computed in the left branch, and the computation reaches the "[0]" node in the right subtree, a merge path is drawn as indicated, and the branch with the right "[0]" node is closed. The price to be paid is that Δ_1 as computed so far is invalid now. Technically, the $ab(inv1)$ literal can be thought of as being removed from the branch (it becomes "invisible" in our terminology). Hence, the computation starts again as indicated below the triangle. Eventually, both Δ_1 and Δ_2 can be found there.

Why is it attractive to use such a "non-monotonic" strategy? The answer is that it is little effort to recompute the initial segment of the diagnosis and better to save recomputing the "huge" part. We do not suggest to use the merge paths in all possible situations. In order to be flexible and allow guidance by heuristics, merge paths are thus always *optional* in the calculus defined below.

Preliminaries. We assume that the reader is familiar with the basic concepts of first-order logic. Throughout this paper, we are concerned with finite ground clause sets. A *clause* is an expression of the form $\mathcal{A} \leftarrow \mathcal{B}$, where $\mathcal{A} = (A_1, \ldots, A_m)$ and $\mathcal{B} = (B_1, \ldots, B_n)$ are finite sequences of atoms ($m, n \geq 0$); $\mathcal{A}$ is called the *head*, and $\mathcal{B}$ is called the *body* of the clause. Whenever convenient, a clause is also identified with the disjunction $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$ of literals.

Quite often, the ordering of atoms does not play a role, and we identify $\mathcal{A}$ and $\mathcal{B}$ with the sets $\{A_1, \ldots, A_m\}$ and $\{B_1, \ldots, B_n\}$, respectively. Thus, set-theoretic operations (such as "$\subset$", "$\cap$" etc.) can be applied meaningfully.

By $\overline{L}$ we denote the complement of a literal L. Two literals L and K are *complementary* if $\overline{L} = K$. In the sequel, the letters K and L always denote literals, A and B always denote atoms, C and D always denote clauses, $\mathcal{S}$ always denotes a finite ground clause set, and Σ denotes its signature, i.e. $\Sigma = \bigcup\{\mathcal{A} \cup \mathcal{B} \mid \mathcal{A} \leftarrow \mathcal{B} \in \mathcal{S}\}$.

As usual, we represent a Σ-interpretation $\mathcal{I}$ by the set of *true* atoms, i.e. $\mathcal{I}(A) = \mathit{true}$ iff $A \in \mathcal{I}$. Define $\mathcal{I} \models \mathcal{A} \leftarrow \mathcal{B}$ iff $\mathcal{B} \subseteq \mathcal{I}$ implies $\mathcal{A} \cap \mathcal{I} \neq \emptyset$. Notice that this is consistent with other usual definitions when clauses are treated as disjunctions of literals. Usual model-theoretical notions of "satisfiability", "validity" etc. of clauses and clause sets are applied without defining them explicitly here.

Minimal models are of central importance in various fields, like (logic) programming language semantics, non-monotonic reasoning (e.g. GCWA, WGCWA) and knowledge representation. Of particular interest are Γ-minimal models, i.e. minimal models only wrt. the Γ-subset of Σ. From a circumscriptive point of view, Γ is thus the set of atoms to be minimized, and $\Sigma \setminus \Gamma$ varies. In the sequel, Γ always denotes some subset of the signature Σ.

Definition 1 (Γ-Minimal Models). *For any atom set M define the* restriction of M to Γ *as $M|\Gamma = M \cap \Gamma$. In order to relate atom sets M_1 and M_2 define $M_1 <_\Gamma M_2$ iff $M_1|\Gamma \subset M_2|\Gamma$, and $M_1 =_\Gamma M_2$ iff $M_1|\Gamma = M_2|\Gamma$. As usual, the relation $M_1 \leq_\Gamma M_2$ is defined as $M_1 <_\Gamma M_2$ or $M_1 =_\Gamma M_2$. We say that a model $\mathcal{I}$ for a clause set M is* Γ-minimal (for M) *iff there is no model $\mathcal{I}'$ for M such that $\mathcal{I}' <_\Gamma \mathcal{I}$*

It is easy to see that $\leq_\Gamma$ is a partial order and that $=_\Gamma$ is an equivalence relation. Notice that the "general" minimal models can simply be expressed by setting $\Gamma = \Sigma$. Henceforth, by a *minimal* model we mean a Σ-minimal one.

An obvious consequence of this definition is that every minimal model of $\mathcal{S}$ is also a Γ-Minimal model of $\mathcal{S}$ (but the converse does not hold in general).

2 Literal Trees and Merge Paths

We consider finite ordered trees T where the nodes, except the root node, are labeled with literals. The labeling function is denoted by λ. A *branch* of T is a sequence $b = (N_0, N_1, \ldots, N_n)$ of nodes of T such that N_0 is the root, N_i is an immediate successor node of N_{i-1} (for $1 \leq i \leq n$) and N_n is a leaf node. The fact that b is a branch of T is also written as $b \in T$.

Any subsequence $b' = (N_i, \ldots, N_j)$ with $0 \leq i \leq j \leq n$ is called a *partial branch of* b; if $i = 0$ then this subsequence is called *rooted*. Define $last(b') = N_j$. In the sequel the letter b always denotes a branch or a partial branch. The expression (b_1, b_2) denotes the concatenation of partial branches b_1 and b_2; similarly, the expression (b, N) denotes $(N_i, \ldots, N_j, N)$, where b is the partial branch $(N_i, \ldots, N_j)$. For convenience we write "the node L", where L is a literal, instead of the more lengthy "the node N labeled with L", where N is some node given by the context. In the same spirit, we write $(L_1, \ldots, L_n)$ and mean the partial branch $(N_1, \ldots, N_n)$, or even $(N_0, N_1, \ldots, N_n)$ in case N_0 is the root and N_i is an immediate successor node of the root, where N_i is labelled with L_i (for $1 \leq i \leq n$). Further, (b, L) means (b, N), where N is some node labeled with L and b is a partial branch.

A branch b is labeled either as "open", "closed" or with some subset of Γ. In the latter case, b is called a MM-branch, and MM(b) denotes that set, which is called the *minimal model of* b. A tree or subtree is *closed* iff every of its branches is closed, otherwise it is *non-closed*. A tree or subtree is *open* if some of its branches are open.

Definition 2 (Ancestor Path, Merge Path). *Let T be a tree and suppose that T contains a rooted partial branch b of the form $b = (N_0, N_1, \ldots, N_i, \ldots, N_n)$ with N_0 being the root. Any sequence $ancp(b, N_i) := (N_n, N_{n-1}, \ldots, N_i)$, where $n \geq i > 0$, is called an* ancestor path *(of b). The node N_n is called the* tail *and the node N_i is called the* head *of this ancestor path. Now, if it additionally holds that $\lambda(N_i) = A$ and $\lambda(N_n) = \neg A$ (for some atom A) then $ancp(b, N_i)$ is called an* ancestor merge path *(of b).*
Let T contain rooted partial branches $b^T = (N_0, N_1, \ldots, N_i, N_{i+1}, \ldots N_n)$ and $b^H = (N_0, N_1, \ldots, N_i, M_{i+1}, \ldots M_m)$ with N_0 being the root and $m, n > i \geq 0$ and $M_{i+1} \neq N_{i+1}$ and such that $\lambda(N_n) = \lambda(M_m) = A$ for some atom A. Define $p^T = N_n, \ldots, N_{i+1}$, $p^H = M_{i+1}, \ldots, M_m$, and $p = (p^T, p^H)$. Here, p is understood as a concatenation of p^T and p^H. By this definition, nodes on paths are written in order from tail to head. We assume that p can always be decomposed into its constituents p^T and p^H; p is called a non-ancestor merge path of T from b^T to b^H with tail N_n and head M_m. *It is also denoted by $mergep(b^T, b^H)$. The node N_i is called the* turn point *of p. Note that the turn point is not on p.*

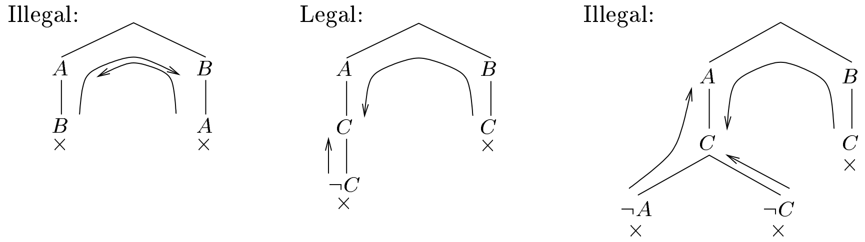
By a merge path *we mean a non-ancestor merge path or an ancestor merge path. The letters p and q are used in the sequel to denote ancestor paths or merge paths, and the letter $\mathcal{P}$ will be used to refer to sets of merge paths.*

A non-ancestor merge path with $m = n = i + 1$ is called *factoring*, the case $m = i + 1$ is called a *hook*, and the case $m > i + 1$ is called a *deep merge path*.

Definition 3 (Ordering on paths). *Suppose the paths $p = (N_1, \ldots, N_n)$ and $q = (M_1, \ldots, M_m)$ as given. Define q* precedes *p, as $q \prec p$ iff $M_m \in \{N_2, \ldots, N_{n-1}\}$. We say that a finite set of paths $\mathcal{P}$ is* legal *iff the $\prec$ relation on $\mathcal{P}$ can be extended to a partial order $\ll$ on $\mathcal{P}$.* Illegal *means not legal.*

Notice that the $\prec$ relation is irreflexive but in general not transitive. One could also define a set of paths to be illegal if it contains a cycle, i.e. if there are paths $p_1, \ldots, p_n \in \mathcal{P}$ such that $p_1 \prec p_2 \prec \cdots \prec p_n \prec p_1$, for some $n > 1$. Avoiding cycles is important to guarantee the soundness of the calculus.

Example 1 (Ordering). The figure below contains examples of trees equipped with merge paths. The underlying clause sets can be left implicit. Merge paths are indicated using arrow notation. For instance, in the right tree, the arrow from the leaf node $\neg A$ to A indicates an (the) ancestor merge path $p_1 = (\neg A, C, A)$ of the branch $(A, C, \neg A)$ with tail $\neg A$ and head A. In the same tree, the arrow from the rightmost node C to the other node C indicates a non-ancestor merge path $p_2 = \mathit{mergep}((B, C), (A, C)) = (C, B, A, C)$ with tail C (the right node) and head C (the other node C) and the root as turn point. In terms of Definition 2 we have $p_2^T = (C, B)$ and $p_2^H = (A, C)$. The path p_2 is an example of a *deep merge path*. The merge path set $\{p_1, p_2\}$ is not legal because both $p_1 \prec p_2$ and $p_2 \prec p_1$ and hence $\prec$ cannot be extended to a partial order. The left tree contains two non-ancestor merge paths and both are "hooks".



The left and right cases are the simplest cases for illegality, as in both cases only two merge paths are involved. These are illegal, because the heads of the merge paths are mutually contained as inner nodes. The left tableau would correspond to an unsound combination of the "folding up" and "folding down" inference rules, usually avoided in implementations by choosing not to combine them at all.

The new calculus to be presented below does not only construct a tableaux T as the derivation proceeds, but also a *legal* set of merge paths $\mathcal{P}$. This guarantees soundness.

In order to achieve minimal model computation, we have to define how interpretations are extracted from open branches.

Definition 4 (Visibility, Branch Semantics). *Let $b = (N_0, N_1 \ldots, N_n)$ be a rooted partial branch in a tree T (not necessarily a hyper tableau) with $n \geq 0$,* *and let $\mathcal{P}$ be a legal set of merge paths in T. The node N_i (where $0 < i \leq n$) that is not the tail of a merge path in $\mathcal{P}$ is said to be* visible from N_n wrt. $\mathcal{P}$ *iff* $\mathcal{P} \cup \{ancp(b, N_i)\}$ *is legal. Define*

$$[\![(N_0, N_1, \ldots, N_n)]\!]_{\mathcal{P}} = \{\lambda(N_i) \mid N_i \textit{ is visible from } N_n \textit{ wrt. } \mathcal{P}, \textit{ for } 0 < i \leq n\} \ .$$

The set $[\![b]\!]_{\mathcal{P}}$ *is called* inconsistent *iff* $\{A, \neg A\} \subseteq [\![b]\!]_{\mathcal{P}}$ *for some atom A;* consistent *means "not inconsistent". We omit "wrt. $\mathcal{P}$" when $\mathcal{P}$ is given by the context.*

The head of a merge path hides nodes that are on the path from nodes beyond the head, i.e. away from the direction that the head points. Those nodes that are not hidden from a node are visible to that node. In the definition of branch semantics an atom A is true in a consistent branch if and only if it is visible from the leaf. For instance, in the middle tableau in Example 1 we have $[\![(A, C, \neg C)]\!]_{\mathcal{P}} = \{\neg C, C\}$ and $[\![(B, C)]\!]_{\mathcal{P}} = \{B, C\}$, where $\mathcal{P}$ consists of the two merge paths drawn there. Notice that the case $n = 0$ is not excluded, and it holds that $[\![(N_0)]\!] = \emptyset$.

3 Hyper Tableaux with Merge Paths

Before defining the new calculus we take one more preliminary step: suppose that $B \in [\![b]\!]_{\mathcal{P}}$ for given open branch b and legal path set $\mathcal{P}$. In the trees constructed in Definition 5, there is a *unique* node N_B in b with $\lambda(N_B) = B$ such that N_B is visible from the leaf of b[1]. Consequently, the ancestor merge path $ancp((b, \neg B), N_B)$ is uniquely defined, and it is denoted by $ancp((b, \neg B))$ alone.

Definition 5 (Hyper tableaux with merge paths). *Let T be a tree, b be a branch in T and let $L_1 \vee \cdots \vee L_n$ be a disjunction of literals. We say that T' is an* extension of T at b with $L_1 \vee \cdots \vee L_n$ *iff T' is obtained from T by attaching to the leaf of b n new successor nodes $N_1, \ldots, N_n$ that are labeled with the literals $L_1, \ldots, L_n$ in this order.*

A selection function *is a total function f that maps an open tree to one of its open branches. If $f(T) = b$ we also say that b* is selected in T by f.

Hyper tableaux T for $\mathcal{S}$ with merge path set $\mathcal{P}$ *– or $(T, \mathcal{P})$ for short – are defined inductively as follows.*

Initialization step: *$(\epsilon, \emptyset)$ is a hyper tableau for $\mathcal{S}$, where ϵ is a tree consisting of a root node only. Its single branch is marked as "open".*

Hyper extension step with C: *If (i) $(T, \mathcal{P})$ is an open hyper tableau for $\mathcal{S}$ with selected branch b, and (ii) $C = A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is a clause from $\mathcal{S}$ (for some $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ and $m, n \geq 0$), and (iii) $\{B_1, \ldots, B_n\} \subseteq [\![b]\!]_{\mathcal{P}}$, and (iv) $\{A_1, \ldots, A_m\} \cap [\![b]\!]_{\mathcal{P}} = \emptyset$ (regularity), then $(T', \mathcal{P}')$ is a hyper tableau for $\mathcal{S}$, where (i) T' is an extension of T at b with $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$, and (ii) every branch $(b, \neg B_1) \ldots, (b, \neg B_n)$ of T' is labeled as closed, and (iii)*

[1] Most proofs are omitted or only sketched for space reasons; the full version [3] contains all proofs.

every branch $(b, A_1) \ldots, (b, A_m)$ *of* T' *is labeled as open, and (iv)* $\mathcal{P}' = \mathcal{P} \cup \{ancp((b, \neg B_1)), \ldots, ancp((b, \neg B_n))\}$. *If conditions (i) – (iv) hold, we say that an "extension step with clause* $\mathcal{A} \leftarrow \mathcal{B}$ *is applicable to* b*".*

Merge path step with p***:*** *If (i)* $(T, \mathcal{P})$ *is an open hyper tableau for* $\mathcal{S}$ *with selected branch* b*, and (ii)* $p = mergep(b, b^H)$ *is a non-ancestor merge path from* b*, for some rooted partial branch* b^H *of* T*, and (iii)* $last(b^H)$ *is not the tail of a merge path in* $\mathcal{P}$*, and (iv)* $\mathcal{P} \cup \{p\}$ *is legal, then* $(T', \mathcal{P}')$ *is a hyper tableau for* $\mathcal{S}$*, where (i)* T' *is the same as* T*, except that* b *is labeled as closed in* T'*, and every* MM-*branch* b' *of* T *with* $[\![b']\!]_{\mathcal{P} \cup \{p\}} | \Gamma \subset \mathrm{MM}(b')$ *is labeled as open in* T'*, and (ii)* $\mathcal{P}' = \mathcal{P} \cup \{p\}$*. If conditions (i) – (iv) hold, we say that a "merge path step with merge path* p *is applicable to* b*".*

Minimal Model Test: *If (i)* $(T, \mathcal{P})$ *is an open hyper tableau for* $\mathcal{S}$ *with selected branch* b*, and (ii)* $[\![b]\!]_{\mathcal{P}}$ *is a* Γ*-minimal model of* $\mathcal{S}$*, then* $(T', \mathcal{P})$ *is a hyper tableau for* $\mathcal{S}$*, where* T' *is the same as* T *except that* b *is labeled in* T' *with* $[\![b]\!]_{\mathcal{P}} | \Gamma$*. If applicability conditions (i) and (ii) hold, we say that the minimal model test inference rule is* applicable *(to* b*).*

A (possibly infinite) sequence $((\epsilon, \emptyset) = (T_0, \mathcal{P}_0)), (T_1, \mathcal{P}_1), \ldots, (T_n, \mathcal{P}_n), \ldots$ *of hyper tableaux for* $\mathcal{S}$ *is called a* derivation*, where* $(T_0, \mathcal{P}_0)$ *is obtained by an initialization step, and for* $i > 0$ *the tableau* $(T_i, \mathcal{P}_i)$ *is obtained from* $(T_{i-1}, \mathcal{P}_{i-1})$ *by a single application of one of the other inference rules. A* derivation of $(T_n, \mathcal{P}_n)$ *is a finite derivation that ends in* $(T_n, \mathcal{P}_n)$*. A* refutation of $\mathcal{S}$ *is a derivation of a closed tableau.*

This definition is an extension of previous ground versions of hyper tableaux (mentioned in the introduction) by bringing in an inference rule for merge paths and explicitly handling Γ-minimal models. The introduction of non-ancestor merge paths requires to explicitly keep track of the ancestor merge paths as well.

The purpose of the hyper extension step rule is to satisfy a clause that is not satisfied in the selected branch b. An implicit legality check for the ancestor paths added in an extension step is carried out by excluding those atoms from the branch semantics that would cause illegality when drawing an ancestor path to them.

An obvious invariant of the inference rules is that every open or MM-branch b is labeled with positive literals only and hence $[\![b]\!]_{\mathcal{P}}$ is consistent. Thus $[\![b]\!]_{\mathcal{P}}$ conforms to our convention of representing interpretations as the set of atoms being true in it.

The purpose of the minimal model test rule is to remember that a Γ-minimal model is computed and to attach it to the selected branch b. Since usually one is interested only in the Γ-subset of models, we keep only the Γ-atoms. These are thought to be the output of the computation. Notice that for MM-branches, a hyper extension step is not applicable, because MM-branches are not open and only open branches can be selected. For the same reason merge path steps are also not applicable to MM-branches.

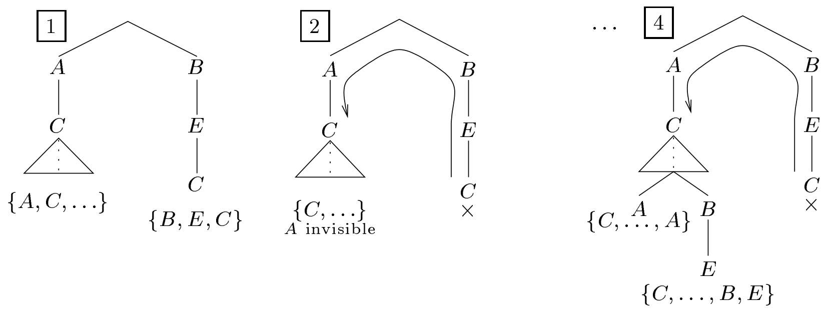
The purpose of the merge path step inference rule is to close branches because a "proof" or a model is to be found in the branch where the drawn merge path is pointing to. But in the course of a derivation, a previously computed Γ-minimal model $\mathrm{MM}(b)$ of a branch b might no longer be the same as $[\![b]\!]_{\mathcal{P}}|\Gamma$, because of a deep merge path step with head node (for instance) in b. Therefore, the label $\mathrm{MM}(b)$ has to be rejected and the branch has to be opened again for further extension. This is expressed in item (i) in the conclusion of the merge path step inference rule (Def. 5). Notice, however, that this happens only if some atom $A \in \Gamma$ in $[\![b]\!]_{\mathcal{P}}$ becomes invisible, not if some other literal from $\Sigma \setminus \Gamma$ becomes invisible. Thus, some deep merge paths can still be drawn without causing recomputation.

3.1 Examples

(1) Consider the figure in Example 1 again. Closed branches are marked with the symbol "$\times$" as closed. Only the tableau in the middle is constructible by the calculus, because the calculus rules forbid the derivation of a tableau with an illegal set of merge paths. In this middle tableau the left branch gets closed by a hyper extension step with the clause $\leftarrow C$, and the right branch is closed by a non-ancestor merge path step as indicated. This application of a non-ancestor merge path step corresponds to a folding-up step in model elimination [8].

The right tableau shows that both ancestor and non-ancestor merge paths have to be taken into account for legality.

(2) The figure below serves as an example to demonstrate the change of branch semantics as the derivation proceeds and the computation of models. We forget about the minimal model test rule for a moment.



Suppose that the hyper tableau $\boxed{1}$ has been constructed. The semantics of the right branch $b = (B, E, C)$ is $[\![b]\!]_{\mathcal{P}} = \{B, E, C\}$. Suppose that this branch can be extended further. Suppose that the left subtree contains an open branch $b_{\ldots}$ that makes A and C true. This is indicated by the set $[\![b_{\ldots}]\!]_{\mathcal{P}} = \{A, C, \ldots\}$. Further suppose that this is a minimal model.

Next, let a merge path step be applied with non-ancestor merge path p to the tableau $\boxed{1}$, yielding the tableau $\boxed{2}$. By this step, b is closed and hence its interpretation is rejected for the time being. A second effect of this step is that the node labeled with A becomes invisible from the leaf of $b_{\dots}$. Thus $[\![b_{\dots}]\!]_{\mathcal{P}\cup\{p\}} = \{C, \dots\}$. Now, this new interpretation has to be "repaired" by bringing in A again. This is done in the next step by extending with $A \vee B$ yielding a tableau $\boxed{3}$ (which is not depicted). Notice that the minimal model $[\![b_{\dots}]\!]_{\mathcal{P}}$ is indeed reconstructed, only in a different order. In order to reconstruct the rejected interpretation $\{B, E, C\}$ from above that was rejected by the merge path step, a hyper extension step below the new B node with E is carried out. This leads to the tableaux $\boxed{4}$. Notice that the new branch with semantics $\{C, \dots, B, E\}$ possibly contains more elements than the corresponding one with semantics $\{B, E, C\}$.

It is worth emphasizing that the re-computation of models happens only in the case of non-ancestor merge paths with their head in *open* branches. Merge paths into *closed* branches are "cheap" in that no re-computation is necessary. Thus, in a sense, refutational theorem proving, which would stop with failure after the first open finished branch (cf. Def. 6 below) is found, is "simpler" than computing models.

In order to demonstrate the effect of the minimal model test inference rule let now $\Gamma = \{C, E\}$. We start with tableau $\boxed{1}$ again. For the branch $b_{\dots}$ the minimal model $[\![b_{\dots}]\!]_{\mathcal{P}} = \{A, C, \dots\}$ was supposed. Suppose that E is not contained in that set. Then $[\![b_{\dots}]\!]_{\mathcal{P}}|\Gamma = \{C\}$ is a Γ-minimal model, because $[\![b_{\dots}]\!]_{\mathcal{P}}$ is a minimal model. According to the minimal model test inference rule, the branch $b_{\dots}$ can be labeled with $\{C\}$ then.

Now, consider tableau $\boxed{2}$. The merge path p there eliminates the Γ-minimal model candidate in the right branch by closing it. Concerning the left branch $b_{\dots}$, although A has been removed from its previous interpretation $[\![b_{\dots}]\!]_{\mathcal{P}} = \{A, C, \dots\}$, its Γ-minimal model $\{C\}$ has not been changed, i.e. $[\![b_{\dots}]\!]_{\mathcal{P}}|\Gamma = [\![b_{\dots}]\!]_{\mathcal{P}\cup\{p\}}|\Gamma = \{C\}$. Consequently the branch label $\{C\}$ has not to be removed and $b_{\dots}$ has not to be opened again. This is reflected by the result description (i) in the definition of merge path step. If Γ were Σ, the branch $b_{\dots}$ would have to be opened again and the computation could continue as above leading to $\boxed{4}$.

3.2 Finite Derivations

Unfortunately, our calculus does not terminate in general, i.e. there are infinite derivations (for finite clause sets), although we employ the "regularity" test (cf. Def. 5). This is due to deep merge paths – without them, termination is straightforward to prove. For instance, the satisfiable clause set $\{(A, B \leftarrow\ ), (B, C \leftarrow\ ), (A, D \leftarrow\ ), (C \leftarrow A)\}$ admits an infinite derivation (cf. [3]) even under very reasonable assumptions, namely that only hooks are mandatory, and that deep merge paths are carried out only to close branches holding non-minimal models. As a consequence we propose the following technique:

Theorem 1 (Termination Criterion). *A derivation $(T_0, \mathcal{P}_0), \dots, (T_n, \mathcal{P}_n), \dots$ is finite, provided that for every $(T_i, \mathcal{P}_i)$, where $i \geq 0$, an applicable merge path step with merge path p is not carried out if for some open branch b in T_i more* *than an a priori fixed number max of occurrences of some label A is invisible from last*(b) *wrt.* $\mathcal{P}_i \cup \{p\}$.

This criterion avoids infinite derivations by bounding repetitions of the same literal along branches. A trivial instance is $max = 0$. Then no deep merge paths but only hooks are possible. The idea underlying the criterion is that one should not without bound repeat the derivation of an atom that becomes repeatedly invisible on a branch. Due to this criterion we consider from now on only finite derivations.

Definition 6 (Redundancy, Fairness). *Suppose as given some hyper tableau* $(T, \mathcal{P})$ *for* $\mathcal{S}$. *A clause* $\mathcal{A} \leftarrow \mathcal{B}$ *is called* redundant *in an open branch b of T wrt.* $\mathcal{P}$ *iff* $[\![b]\!]_{\mathcal{P}} \models \mathcal{A} \leftarrow \mathcal{B}$ *(iff* $\mathcal{B} \subseteq [\![b]\!]_{\mathcal{P}}$ *implies* $\mathcal{A} \cap [\![b]\!]_{\mathcal{P}} \neq \emptyset$*).*

A branch b of T is called finished (wrt. $\mathcal{P}$) *iff (i) b is closed, or (ii) b is an* MM*-branch, or else (iii) the minimal model test inference rule is not applicable to b and every clause* $\mathcal{A} \leftarrow \mathcal{B} \in \mathcal{S}$ *is redundant in b wrt.* $\mathcal{P}$*. The term* unfinished *means "not finished".*

Now suppose as given a finite derivation $D = (T_0, \mathcal{P}_0), \ldots, (T_n, \mathcal{P}_n)$ *from* $\mathcal{S}$ *with selection function f. D is called* fair *iff (i) D is a refutation, i.e. T_n is closed, or else (ii) $f(T_n)$ is finished wrt.* $\mathcal{P}_n$.

The selection function f is called a model computation selection function *iff f maps a given open hyper tableau* $(T, \mathcal{P})$ *to an unfinished branch wrt.* $\mathcal{P}$*, provided one exists, else f maps T to some other open (finished) branch.*

According to this definition, the only possibility to be unfair is to terminate a derivation with a selected open branch that could be either labeled with a Γ-minimal model or extended further.

The *existence* of fair derivations is straightforward because we insist on *finite* derivations. Notice that any input clause not redundant so far in a branch b can be made redundant by simply carrying out an extension step with that clause.

The idea behind a *model-computation selection function* is that no derivation should stop with an unfinished branch. Since finished open branches constitute Γ-models, with such a selection function every Γ-minimal model is computed.

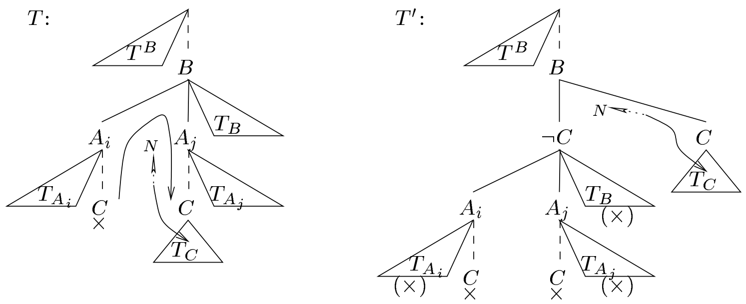
4 Soundness and Completeness

Lemma 1 (Soundness lemma). *Let* $(T, \mathcal{P})$ *be a hyper tableau for satisfiable clause set* $\mathcal{S}$*. Then for every minimal model* $\mathcal{I}$ *of* $\mathcal{S}$ *there is an open branch b of T such that* $[\![b]\!]_{\mathcal{P}} \subseteq \mathcal{I}$.

The proof of Lemma 1 is done by simulating non-ancestor merge paths by atomic cuts, i.e. by β-steps applied to disjunctions of the form $A \vee \neg A$, for some atom A. The branch semantics in presence of atomic cuts is given by forgetting about the negative literals, i.e. $[\![b]\!]^{+}_{\mathcal{P}} = \{A \in [\![b]\!]_{\mathcal{P}} \mid \; A \text{ is a positive literal}\}$ for any consistent branch b in a hyper tableau with atomic cuts.

The transformation t defined below takes a hyper tableau with cut $(T, \mathcal{P})$ where $\mathcal{P}$ is legal and contains at least one non-ancestor merge path, and returns a hyper tableau with cut $(T', \mathcal{P}') = t(T, \mathcal{P})$ that contains one less non-ancestor merge path in $\mathcal{P}'$ (which is legal as well). The transformation t preserves the following *invariant*: for every consistent and open branch b' of T' there is a consistent and open branch b of T such that $[\![b]\!]^+_{\mathcal{P}} \subseteq [\![b']\!]^+_{\mathcal{P}'}$. Repeated application of t as long as possible results in a tableau $(T_{\text{cut}}, \mathcal{P}_{\text{cut}})$ with cuts but without non-ancestor merge paths. All literals along all branches are visible there, and hence we have a "standard tableau" with cuts then. The lemma then is proven for this tableau, and using the *invariant* above it can be translated back for the originally given tableau $(T, \mathcal{P})$.

The transformation t itself is depicted in the figure below. The left side displays the most general situation. Dashed lines mean partial branches. For instance, the top leftmost dashed line leading to B means the partial branch p_B from the root to the node (inclusive) labeled with B. Triangles are certain forests. The most appropriate intuition is to think of trees as branch sets. Then the triangle T^B is simply the set of the branches obtained from T by deleting all branches that contain p_B.



Since $\mathcal{P}$ is legal it is extendible to a partial order $\ll$. Let p be a minimal element in this order. This is the one to be transformed away. It is important to use a minimal element in order to prove the invariant.

The solid lines, just like the ones below B indicate a hyper extension step; here, it is supposed that a hyper extension step with clause $C = A_1, \ldots, A_n \leftarrow B$ has been carried out to p_B, and all the literals short of A_i and A_j (for some $i, j \in \{1, \ldots, n\}$ and $i \neq j$) are attached to nodes in the subtree T_B. The assumed non-ancestor merge path p is indicated with tail node C (left) and head node C (right) and turn point B. There might be other non-ancestor merge paths in $\mathcal{P}$, in particular some where the head of p is an inner node. This possibility is indicated in the figure as well, by the arrow pointing into T_C. The tail of this non-ancestor merge path, say p_C is a leaf node N somewhere in T.

Inconsistent or closed branches are marked by "$\times$". The effect of the transformation t is shown on the right. Notice the cut with $C \vee \neg C$ at the turn point B. The transformation is understood to move merge paths as well. For example, the non-ancestor merge path p_C still has the same head and tail node, but they are possibly located in different places in T' now, and also a different turn point might result. After transformation some branches might get closed due to the presence of $\neg C$. This is indicated by "$(\times)$". Notice that the transformation only introduces new *negative* literals into branches, $\neg C$, so that the branch semantics wrt. positive literals does not change, as required in the *invariant*.

The central properties that have to be argued for are (i) that the tableau resulting from the transformation is a hyper tableau (i.e. that all negative leaf nodes can still be closed by legal ancestor paths), and (ii) that the *invariant* holds. This is done by expressing the *invariant* in terms of visibility from leaf nodes and then arguing with the orderings underlying $\mathcal{P}$ and $\mathcal{P}'$.

This lemma is applied in the proof of the next theorem, which is our main result.

Theorem 2 (Soundness and Completeness). *Let f be a model computation selection function and D be a finite, fair derivation from clause set $\mathcal{S}$ of the hyper tableau $(T, \mathcal{P})$. Then $\{\mathrm{MM}(b) \mid b$ is a MM-branch of $T\} = \{\mathcal{I}|\Gamma \mid \mathcal{I}$ is a Γ-minimal model of $\mathcal{S}\}$ Furthermore, if $\mathcal{S}$ is unsatisfiable then T is closed (refutational completeness).*

Proof. Minimal model soundness – the first theorem statement in the "$\subseteq$"-direction – is an immediate consequence of the applicability condition (ii) in the minimal model test inference rule and the result description (i) in the merge path step inference rule. Regarding minimal model completeness – the first theorem statement in the "$\supseteq$"-direction –, suppose to the contrary that for some Γ-minimal model $\mathcal{I}$ of $\mathcal{S}$ there is no MM-branch of T such that $[\![b]\!]_{\mathcal{P}} =_{\Gamma} \mathcal{I}$.

Clearly $\mathcal{I}|\Gamma \subseteq \mathcal{J}$ for some minimal model $\mathcal{J}$ of $\mathcal{S}$. Now, label all MM-branches of T as open and let T' be the resulting tableau. By the soundness lemma (Lemma 1) we know that T' contains an open branch b with $[\![b]\!]_{\mathcal{P}} \subseteq \mathcal{J}$. Suppose that b is a MM-branch of T. The case $[\![b]\!]_{\mathcal{P}} =_{\Gamma} \mathcal{I}$ is impossible by the assumption to the contrary. Hence from $[\![b]\!]_{\mathcal{P}} \neq_{\Gamma} \mathcal{I}$ and $[\![b]\!]_{\mathcal{P}} \subseteq \mathcal{J}$ it follows $[\![b]\!]_{\mathcal{P}} <_{\Gamma} \mathcal{I}$. This, however, is impossible by soundness, as it contradicts the given fact that $\mathcal{I}$ is a Γ-minimal model. Therefore b is not an MM-branch in T. Since it is open in T' it must be open in T as well. We are given that D is fair. Since f is a model computation selection function, this implies that b is finished.

For this particular b we show next that $[\![b]\!]_{\mathcal{P}} =_{\Gamma} \mathcal{I}$ holds. For, suppose to the contrary, that $[\![b]\!]_{\mathcal{P}} \neq_{\Gamma} \mathcal{I}$ holds. Again, with $[\![b]\!]_{\mathcal{P}} \subseteq \mathcal{J}$ it follows $[\![b]\!]_{\mathcal{P}} <_{\Gamma} \mathcal{I}$. Since b (of T) is open – i.e. neither closed nor a MM-branch – and finished the minimal model test rule is not applicable and every clause from $\mathcal{S}$ is redundant in b wrt. $\mathcal{P}$. In other words, $[\![b]\!]_{\mathcal{P}} \models \mathcal{S}$. With $[\![b]\!]_{\mathcal{P}} <_{\Gamma} \mathcal{I}$ this is a contradiction to the given Γ-minimality of $\mathcal{I}$. Hence, $[\![b]\!]_{\mathcal{P}} =_{\Gamma} \mathcal{I}$. But then the minimal model test inference rule is applicable to b, because $\mathcal{I}$ is given as a Γ-minimal model, and so $[\![b]\!]_{\mathcal{P}}$ is a Γ-minimal model as well. Hence b is not finished, contradicting the given fairness of D. So the outermost assumption to the contrary must have been wrong, and the theorem follows.

Refutational completeness is proven as follows: suppose that $\mathcal{S}$ is unsatisfiable but T is not closed. By the minimal model soundness result then T must contain an open branch b (because MM-branches are impossible). Since $[\![b]\!]_{\mathcal{P}}$ is an interpretation and $\mathcal{S}$ is unsatisfiable, $[\![b]\!]_{\mathcal{P}}$ falsifies some input clause from $\mathcal{S}$. But then a hyper extension step is applicable to b with this clause. This contradicts the given fact that D is fair. $\square$

5 Further Considerations and Conclusions

Calculi like ours and related calculi need some extra test or device to ensure Γ-minimal model soundness as well. This is due to the inherent complexity of the problem [5]. Fortunately, every Γ-minimal model candidate can be tested in a branch-local way for actual Γ-minimality. More specifically, the approach suggested as the *groundedness test* in [11, 12] is adapted in the full paper. This approach is attractive due to its low (polynomial) memory consumption. Since our approach, when forgetting about merge path steps, is an instance of the method of in [12], low memory assumption can be achieved in our case as well. This does not hold for related methods like MILO-resolution [14], or the minimal-model computation extension of MGTP proposed in [7], or the tableau method of [13], whose worst-case space complexity is exponential.

In the proof of Lemma 1 we indicated how atomic cuts can be used to simulate non-ancestor merge paths. So, the question might arise why not directly use these cuts. The answer is manifold. First, by the mere fact that the simulation exists we get insights how merge paths relate to atomic cuts. Second, the graphical notation might be a helpful metaphor to study the topic. Third, merge paths correspond only to *certain* cuts, much like folding-down [8] or related techniques like complement splitting [9] also correspond only to certain cuts. Fourth, with merge paths, the effect is that they are surgically inserted into the path, and thus in this sense we procrastinate insertion of cuts until useful.

Our approach can be viewed as a Davis-Putnam (DP) procedure. In DP, splitting in a certain order is advantageous for deterministic computation (unit-resulting steps). Our procedure can use the entire set of visible literals to achieve the same determinism, without pre-selecting this splitting order.

It is generally accepted that analytic or even atomic cuts should be applied with care in order not to drown in the search space. This is our viewpoint as well. We emphasize one particular property of the transformation t (cf. the figure in the proof of Lemma 1): in the cut simulation, the subtree T_C is moved to a different place in the tree. By the bare fact that the considered non-ancestor merge path p is legal in the merge path set containing it, we can be *sure* that the destination of T_C (the C node) contains enough ancestor literals so that T_C *remains* a hyper tableau – that all branches with negative leaves remain inconsistent. Clearly, opening branches again would be undesirable as it is unclear if any progress is achieved then. The alternative, forgetting about T_C would cause a lot of recomputation.

Of course, this and other effects and how to avoid them could be formulated as conditions on cuts as well. Non-termination would result if the same atomic cut occurs without bound on a branch.

Conclusions. In this paper we extended previous versions of the hyper tableau calculus by inference rules for merge paths, a device that was originally conceived to speed up refutational theorem proving in the context of clause trees [6]. Our primary goal was to investigate the consequences for model computation purposes. Our main result is therefore a minimal model sound and complete calculus to compute circumscription in the presence of minimized and varying predicates. The motivation was given by the potential to solve a certain problem in diagnosis applications.

We argued that the new calculus generalizes other approaches developed in comparable calculi (folding-up/down, complement splitting). How to apply the new technique *practically*, in particular in the envisaged diagnosis domain, is subject to further investigations. Fortunately, the legality test is $O(|\mathcal{P}|)$ and only negligible overhead is introduced. An algorithm is described in [6].

Acknowledgements. We thank the reviewers for their valuable comments.

References

1. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In *Proc. IJCAI 97*, pages 460–465, Nagoya, 1997.
2. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, LNAI 1126. Springer, 1996.
3. P. Baumgartner, J. Horton, and B. Spencer. Merge Path Improvements for Minimal Model Hyper Tableaux. Fachberichte Informatik 1–99, Universität Koblenz-Landau, Universität Koblenz-Landau, Rheinau 1, D-56075 Koblenz, 1999.
4. F. Bry and A. Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. In Miglioli et al. [10], pages 143–159.
5. T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are π_2^p-complete. *Theoretical Computer Science*, 114:231–245, 1993.
6. J. D. Horton and B. Spencer. Clause trees: a tool for understanding and implementing resolution in automated reasoning. *Artificial Intelligence*, 92:25–89, 1997.
7. K. Inoue, M. Koshimura, and R. Hasegawa. Embedding Negation as Failure into a Model Generation Theorem Prover. In D. Kapur, editor, In *Proc. CADE 11*, LNAI 607, pp. 400–415. Springer, 1992.
8. R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13, 1994.
9. R. Manthey and F. Bry. SATCHMO: a theorem prover implemented in Prolog. In E. Lusk and R. Overbeek, editors, *Proc. CADE 9*, LNCS 310, pp. 415–434. Springer, 1988.
10. P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors. *Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 1071. Springer, 1996.
11. I. Niemelä. A Tableau Calculus for Minimal Model Reasoning. In Miglioli et al. [10].
12. I. Niemelä. Implementing circumscription using a tableau method. In *Proc. ECAI*, pages 80–84, Budapest, 1996. John Wiley.
13. N. Olivetti. A tableaux and sequent calculus for minimal entailment. *Journal of Automated Reasoning*, 9:99–139, 1992.
14. T. Przymusinski. An Algorithm to Compute Circumscription. *Artificial Intelligence*, 38:49–73, 1989.
15. R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, Apr. 1987.