In Proceedings Workshop Computer Science Logic, Heidelberg 1990, Springer LNCS 533, pp. 248 – 260

Towards an efficient Tableau Proof Procedure for Multiple-Valued Logics*

Reiner Hähnle

Institute for Logic, Complexity and Deduction Systems
University of Karlsruhe, Am Fasanengarten 5
7500 Karlsruhe
Federal Republic of Germany
haehnle@ira.uka.de

January 1990

Abstract

One of the obstacles against the use of tableau-based theorem provers for non-standard logics is the inefficiency of tableau systems in practical applications, though they are highly intuitive and extremely flexible from a proof theoretical point of view. We present a method for increasing the efficiency of tableau systems in the case of multiple-valued logics by introducing a generalized notion of signed formulas and give sound and complete tableau systems for arbitrary propositional finite-valued logics.

Introduction

One of the main advantages of the method of semantic tableaux [Smullyan, 1968, Beth, 1986] is that it yields analytic proof theories for a wide variety of standard and non-standard logics within a single framework. With relatively minor modifications tableau proof systems can be designed for such different logics as temporal, intuitionistic and multiple-valued logics [Wolper, 1981, Fitting, 1983, Schmitt, 1989]. In addition, one could easily obtain tableau proof systems, which combine several non-standard concepts, a feature which seems to be interesting e.g. in circuit validation [Kropf & Wunderlich, 1990], natural language processing [Fenstad *et al.*, 1985] or semantics of logic programs [Sheperdson, 1989]. Also, avoidance of normal forms is necessary for the potential application of high-level heuristics.

But there are two major obstacles against the use of tableau systems in automated theorem proving without further modifications. First, the search process tends to be much more inefficient than in, say, resolution provers, if no extra care is taken. But recent research showed that it is possible to reach a similar performance as with resolution-based provers [Oppacher & Suen, 1986, Oppacher & Suen, 1988]. And [Fitting, 1990] shows that completeness proofs for tableau systems that have been tuned for automated theorem proving are still much more transparent

*This work is supported by IBM Germany and is a collaboration of the University of Karlsruhe and the IWBS at IBM Germany in Heidelberg.

than their resolution counterparts. Second, the modifications of standard tableau proof systems to adapt them to non-standard logics are, though highly intuitive, usually not very efficient when one asks for performance. In this paper we concentrate on the second problem and on propositional multiple-valued logics. Our work is part of the TCG Project involving the construction of a tableau-based automated theorem prover for multiple-valued logics, a prototype of which is currently being implemented [Hähnle, 1990].

It should be mentioned that there exists at least one other approach to automated theorem proving in multiple-valued logics. In a series of papers (see e.g. [Stachniak, 1990]) Stachniak developed resolution style systems for logics with finitely many truth values. While in his systems the underlying logics are specified by consequence relations, we will assume that our logics are given by a tabular semantics (cf. [Wójcicki, 1988]).

The paper is organized as follows: In section 1 we introduce some mathematical concepts and specify syntax and semantics of the class of languages under consideration. In section 2 we present our variant of a tableau-based calculus, in section 3 we give proofs of soundness and completeness for our system and we conclude with section 4, summarizing what has been gained.

1 Preliminaries, Syntax, Semantics

We recall some concepts from universal algebra, e.g. to be found in [Burris & Sankappanavar, 1981].

Definition 1.1 (Abstract Algebra of finite Type, Homomorphism)
A **finite type** $\mathcal{F} = \{f_1, \ldots, f_r\}$ *is an indexed set of symbols, each of them having assigned an arity by a mapping* $m : \mathcal{F} \to Nat$. *Let* $\mathcal{F}_n$ *denote the operators with arity* n. *Constants are treated as* 0*-ary functions.*

An **abstract algebra of type** $\mathcal{F}$ *or* **Ω-algebra** *is a non-empty universe* A *together with a family of mappings such that for all* n *and each member* f *in* $\mathcal{F}_n$ *there is a corresponding* **fundamental operation** $f^A : A^n \to A$. *If convenient, the abstract algebra* $< A, \{f_i^A \mid 1 \leq i \leq r\} >$ *and its universe* A *are denoted with the same symbol.*

Let A, B *be abstract algebras of the same type and* $h : A \to B$ *any mapping. If for all* $f \in \mathcal{F}_n, n \in Nat$ *and* $a_1, \ldots, a_n \in A$

$$h(f^A(a_1, \ldots, a_n)) = f^B(h(a_1), \ldots, h(a_n))$$

holds, then h *is called* **homomorphism from A to B**.

Let $\mathcal{F} = \{F_1, \ldots, F_r\}$ be a set of logical connectives and $L_0 := \{p_i \mid i \in Nat\}$ the set of propositional variables or **atomic formulas**, which has to be disjoint with $\mathcal{F}$. With L we denote the abstract algebra that is freely generated over L_0 in the class of algebras with type $\mathcal{F}$. Thus we have

$$L_{i+1} = L_i \cup \{F_j(X_1, \ldots, X_{m(j)}) \mid X_1, \ldots, X_{m(j)} \in L_i, F_j \in \mathcal{F}\}$$
$$L = \bigcup\{L_i \mid i \in Nat\}$$

as the universe of L.

L_i denotes the **formulas of depth** i. We call L **(propositional) language**, the members of L are called **(propositional) (L-)formulas**.

Let $N = \{0, 1, \ldots, (n-1)\}$ be the finite set of **truth values** and $D \subseteq N$ the set of **designated truth values**. Furthermore let us denote with $n = |N|$ and $d = |D|$ the number of elements in N and D resp. Though all nonnegative values are possible for n and d, we are only interested in the nontrivial cases where $n \geq 2$ and $d \geq 1$.

Let $A =< N, \{f_i \mid 1 \leq i \leq r\} >$ be an algebra of the same type as L. Then we call the pair $\mathcal{A} =< A, D >$ a **structure** for L and the f_i **interpretations** of the F_i. A defines the semantics of the logical operators. We say that $\mathcal{L} =< L, \mathcal{A} >$ is an **n-valued propositional logic with d designated truth-values**.

A **propositional (A-)valuation** of L is a homomorphism v from L to $\mathcal{A}$. A set M of L-formulas is called **(A-)satisfiable**, if there is a valuation v from L to $\mathcal{A}$ such that for any $X \in M$ $v(X) \in D$ holds. In this case v is called **($\mathcal{A}$-)model** for M. If $\{X\}$ is satisfiable for any $\mathcal{A}$-valuation, X is called **tautology**. Due to the universal mapping property (since L was freely generated) it is sufficient to define v on L_0 and then extend it uniquely to L.

Example 1.1 *As the set of logical operators we take* $\mathcal{F} = \{\wedge, \vee, \supset, \neg, \nabla, \sim\}$ *with arities* $m(\wedge) = 2, m(\vee) = 2, m(\supset) = 2, m(\neg) = 1, m(\sim) = 1, m(\nabla) = 1$ *and as truth values* $N = \{0, 1, 2\}, D = \{2\}$. *Their meaning (the abstract algebra A) is given by the following truth tables:*

$\wedge$	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

$\vee$	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

$\supset$	0	1	2
0	2	2	2
1	2	2	2
2	0	1	2

$\neg$	
0	2
1	1
2	0

$\sim$	
0	2
1	2
2	0

∇	
0	0
1	2
2	2

Note that we could have defined disjunction and conjunction alternatively as

$$v(X_1 \vee X_2) = max(v(X_1), v(X_2))$$

$$v(X_1 \wedge X_2) = min(v(X_1), v(X_2))$$

resp. There are many alternatives to our definition of implication, but this is not the issue that interests us here. Let us refer to the logic as defined above with the symbol $\mathcal{L}_3$.

2 Semantic Tableaux

Our goal is to give a tableau proof system for propositional multiple-valued logics with the following features:

- We want a generic proof system, i.e. it should yield a sound and complete set of tableau rules for any logic given to it.
- We do not want to have redundancy in proofs due to the formulation of the tableau rules alone.

The first task was begun by Surma [Surma, 1984] and completed by Carnielli [Carnielli, 1987], who provided a generic tableau proof system as proposed for multiple-valued first-order logics with arbitrary logical connectives and generalized quantifiers. Unfortunately, Carnielli's system does not fulfill the second requirement. To explain this further, let us consider the signed version (see [Smullyan, 1968, Fitting, 1990]) of a tableau proof system for standard logic: A tableau branch may be considered as a set of formulas together with a certain assignment of truth-values. The sign attached to each formula in the branch says that the truth-value of the formula should be the one associated with its sign. The tableau rules provide all significant possibilities to extend a set M of signed formulas preserving consistency. If we can arrive after a number of rule applications at a tableau branch that contains instances of all atomic formulas occurring in M at least once, arbitrarily signed, but non-contradictory, then we are able to construct a model for the formulas on the branch. If this is the case, we say that the branch is *open*. Let us call a tableau *closed* if it is fully expanded and contains no open branches. For the moment, assume that M is a singleton, say $M = \{FX\}$ (where F stands for *false*). Then a closed tableau for $\{FX\}$ represents the fact that there is no way to construct a model where X is false, so X must be a tautology.

Turning to three-valued logics we only need to introduce a third sign, corresponding to the third truth-value (say *undefined*) and define the appropriate rules, but the last step above is no longer valid, since *not false* may be *true* as well as *undefined*. To get a proof of the validity of X we have in fact to construct *two* closed tableaux, namely one with root FX and another one with root UX for the refutation of both non-designated truth-values. In the case of a logic with $(n-d)$ non-designated truth-values this amounts to the construction of $(n-d)$ closed tableaux for the proof of one single theorem. Also, the additional rules tend to be more complicated than the classical ones, as the following example shows:

Example 2.1 *3-valued tableau rules for* $\vee$*:*

$$\frac{F\ X_1 \vee X_2}{\begin{array}{c} F\ X_1 \\ F\ X_2 \end{array}} \qquad \frac{U\ X_1 \vee X_2}{\begin{array}{c|c|c} F\ X_1 & U\ X_1 & U\ X_1 \\ U\ X_2 & U\ X_2 & F\ X_2 \end{array}}$$

On the other hand, inspection of sample proofs shows that there is much redundancy in the proof trees, e.g. in the three-valued case most of the structure and formulas of the tableau for $F\ X$ are also part of the $U\ X$-tableau, even if they contribute nothing to the refutation of UX, and vice versa. We present a systematic way to get rid of this kind of redundancy, resulting in a proof system, where only *one* closed tableau has to be generated to prove the validity of a formula in an arbitrary multiple-valued propositional logic.

One approach to increase efficiency would of course be to perform the steps that are identical in all or in some of the proof trees at the same time (possibly using structure sharing), i.e. to search for the refutation of all non-designated truth-values in parallel. But, as always when one is making algorithms and representations trickier, this leads to a fairly complex proof procedure involving much bookkeeping and hence a cryptical completeness proof. A far more satisfying solution can be achieved on a logical level.

To be specific, consider the signed $\mathcal{L}_3$-formula $T \sim A$. Application of the corresponding tableau rule from [Carnielli, 1987] or [Surma, 1984] yields two new branches with extensions FA and UA, resp. But encountering this formula during a proof does not give rise to any logical reason to split the proof in two cases "$v(A) = 0$" and "$v(A) = 1$" resp. So our idea is to increase the expressivity of the signs in order to be able to state conditions like "$v(A) = 0$ *or* $v(A) = 1$" or equivalently "$v(A) \neq 2$" within a single signed formula and thus to decrease the number of new branches per rule application significantly. It is noteworthy that neither the

idea of enriching the syntax of signs nor of interpreting them semantically in a different way is new. The first has been used in tableau systems for modal logics for a long time (see e.g. in [Fitting, 1983]); on the other hand, in [Fitting, 1989] Fitting denoted upper and lower bounds in a lattice of truth values with single signs. What we will do is to systematically exploit both ideas at the same time.

Definition 2.1 (Sign, Signed Formula)
Let L be any language and D and N be defined as above. Then we define the **set of signs** *as $\mathcal{S} = \{S_i \mid i \in 2^N\}$. For any logic $\mathcal{L}$ we fix a certain set of signs $\mathcal{S}_\mathcal{L} \subseteq \mathcal{S}$ which satisfies $\{S_{\{0\}}, \ldots, S_{\{n-1\}}\} \subseteq \mathcal{S}_\mathcal{L}$[1]. From now on a logic will be a triple $\mathcal{L} =< L, \mathcal{A}, \mathcal{S}_\mathcal{L} >$. With $I_\mathcal{L} = \{i \mid S_i \in \mathcal{S}_\mathcal{L}\}$ we denote the set of allowed indices of signs. With the same symbol we identify the abstract algebra generated by $I_\mathcal{L}$ that has the same type as A and whose fundamental operations are defined by $f^{I_\mathcal{L}}(i_1, \ldots, i_m) = \bigcup\{f^A(j_1, \ldots, j_m) \mid j_k \in i_k, 1 \leq k \leq m\}$. From the context it will always be clear which is meant.*

If X is an L-formula and $S_i = S_{\{i_0,\ldots,i_r\}}$ a sign, then we call the string $S_i(X)$ **signed (L-)formula**. *L^* is the set of signed formulas in a logic $\mathcal{L}$, i.e. all signed L-formulas with signs from $\mathcal{S}_\mathcal{L}$. The members of L^* will be called* **$I_\mathcal{L}$-signed formulas**.

In the above definition we have deliberately admitted $S_\emptyset$ and S_N as signs. While the following definitions and theorems exclude the former implicitely, the latter would be perfectly right, though it is hard to imagine any meaningful application for it.

Example 2.2 *We define for $\mathcal{L}_3$ the set of signs $\{S_{\{0\}}, S_{\{1\}}, S_{\{2\}}, S_{\{0,1\}}\}$ which for convenience we rewrite as $\{F, U, T, (F|U)\}$.*

The intended interpretation of a signed formula $(F|U)(X)$ then is "$v(X) = 0$ or $v(X) = 1$".

Now we are ready to define the tableau rules. We assume familiarity with trees, a formal treatment of proof trees can be found in [Smullyan, 1968].

Definition 2.2 (Tableau Rule)
Let $X = F(X_1, \ldots, X_m)$ be an L-formula in the logic $\mathcal{L} =< L, \mathcal{A}, \mathcal{S}_\mathcal{L} >$. An **($\mathcal{L}$-)tableau rule** *is a function $\pi_{i,F}$ which assigns to a signed formula $S_i(X) \in L^*$ a tree with root $S_i(F(X_1, \ldots, X_m))$, called* **premise**, *and the linear subtrees*

$$\{S_{j_1}(X_{i_1}) \circ \ldots \circ S_{j_t}(X_{i_t}) \mid j_1, \ldots, j_t \in I_\mathcal{L}, t \leq m \text{ and } H_i(F; j_1, \ldots, j_t) \text{ holds}\},$$

called **extensions**[2].

A collection of extensions satisfying (T0) is called **conclusion** *of a tableau rule.*

(T0) *for any $(z_1, \ldots, z_m) \in f^{-1}(i)$ there is an extension $S_{j_1}(X_{i_1}) \circ \ldots \circ S_{j_t}(X_{i_t})$ with $z_{i_k} \in j_k$ for $1 \leq k \leq t$ and the set of extensions is minimal with respect to this condition*[3].

[1] Otherwise it is not guaranteed that all rules can be properly stated.

[2] Extensions are treated like sets and thus of all subtrees that differ only in the ordering of their signed formulas only one appears as an extension of the rule.

[3] Already in the two-valued case there may be more than one minimal (in our sense) set of extensions for a signed formula, so we need the minimality condition; see [Dueck, 1988, p. 12f] for an example.

The condition $H_i(F; j_1, \ldots, j_t)$ means, there exists a homomorphism $h : L \to I_{\mathcal{L}}$, satisfying (T1)–(T4) below:

(T1) *$h(X_{i_k}) = j_k$ for $1 \leq k \leq t$.*

(T2) *If f is the interpretation of F, then $f(v_1, \ldots, v_m) \in i$ must hold, where $v_{i_k} \in h(X_{i_k})$ for $1 \leq k \leq t$ and all other arguments are arbitrary.*

(T3) *There is no j'_k with $|j'_k| > |j_k|$ for $1 \leq k \leq t$ that satisfies (T1) and (T2).*

(T4) *There is no t' with $t' < t$ that satisfies (T1) and (T2).*

If no such homomorphism exists, no rule for the specific combination of formula and sign is defined.

Though this definition seems to be fairly abstract, for any given logic it essentially boils down to the usual tableau rules plus the extra feature of more general signs. To provide a better understanding of how the tableau rules are generated, we give an informal description of the process:

Remember that the extensions are thought to be disjunctively connected while the formulas within an extension are conjunctively connected.

The conclusion of a tableau rule for a sign i and connective F can be thought of as a minimal generalized sum-of-products representation of the two-valued function that holds the entry *true* in its truth table on each place where the truth table of F holds a member of i and holds *false* otherwise.

Each extension corresponds to a product term in this representation. A geometrical interpretation would associate a partial cover of entries in the hypercube that constitutes the truth table of F with an extension. All extensions taken together are a total cover.

- Condition (T0) ensures that all entries from i are covered in some extension and minimizes the number of extensions.
- Condition (T1) defines the interesting part of h.
- Condition (T2) guarantees soundness.
- Condition (T3) represents the strategy to split the proof tree as late as possible, in other words, to keep the signs as general as possible.
- (T4) minimizes the number of subformulas within the extensions and prevents redundant extensions.

Example 2.3 *Consider the truth table of disjunction in $\mathcal{L}_3$ as defined above. Find the tableau rule for $S_{\{1\}}(X_1 \vee X_2)$. We have to find a minimal set of homomorphisms $h : L \to I_{\mathcal{L}}$ covering all entries equal to 1. Hereby choose the sets $h(X_i)$ maximal.*

First, $X_1 \mapsto \{1\}, X_2 \mapsto \{0,1\}$ defines the partial cover...

x_1/x_2	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

...adding the partial cover that corresponds to $X_1 \mapsto \{0,1\}, X_2 \mapsto \{1\}$ yields

x_1/x_2	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

Obviously all of the conditions (T1)–(T4) are satisfied. And since both partial covers are essential and together represent a total cover, condition (T0) also holds.

Example 2.4 *From the homomorphisms that define the cover of the entries equal to* 1 *we can immediately extract the tableau rule:*

$$\frac{U\ (X_1 \vee X_2)}{\begin{array}{c|c} (F|U)\ X_1 & U\ X_1 \\ U\ X_2 & (F|U)\ X_2 \end{array}}$$

Note that the entry for $X_1 = X_2 = 1$ in the truth table of disjunction is covered by both extensions. The rule is considerably simpler than the one from Example 2.1.

In the Appendix a sound and complete tableau system for $\mathcal{L}_3$ can be found.

Tableaux are by the tableau rules finitely generated trees, their nodes being labeled with signed formulas. A **branch** is a path through a proof tree, beginning with the root and ending with a leaf. Usually we identify a branch with the set of signed formulas that is equal to its label set.

Definition 2.3 (Propositional Tableaux)
Let M be a nonempty finite set of $I_{\mathcal{L}}$-signed formulas. Then a **(propositional) tableau** *for M can be constructed in one of the following ways:*

- *A linear tree, where each formula of M occurs exactly once as a label is a tableau for M.*
- *Let T be a tableau for M and B a branch of T, containing a signed formula $S_i(F(X_1, \ldots, X_m))$. If $\pi_{i,F}$ is defined and has extensions $E_1, \ldots, E_n$, append to T at the end of B n linear subtrees containing the signed formulas in $E_1, \ldots, E_n$, resp. in an arbitrary sequence. The resulting tree is again a tableau for M.*

Definition 2.4 (Open, Closed)
A tableau branch is called **closed** *if one of the following conditions is satisfied:*

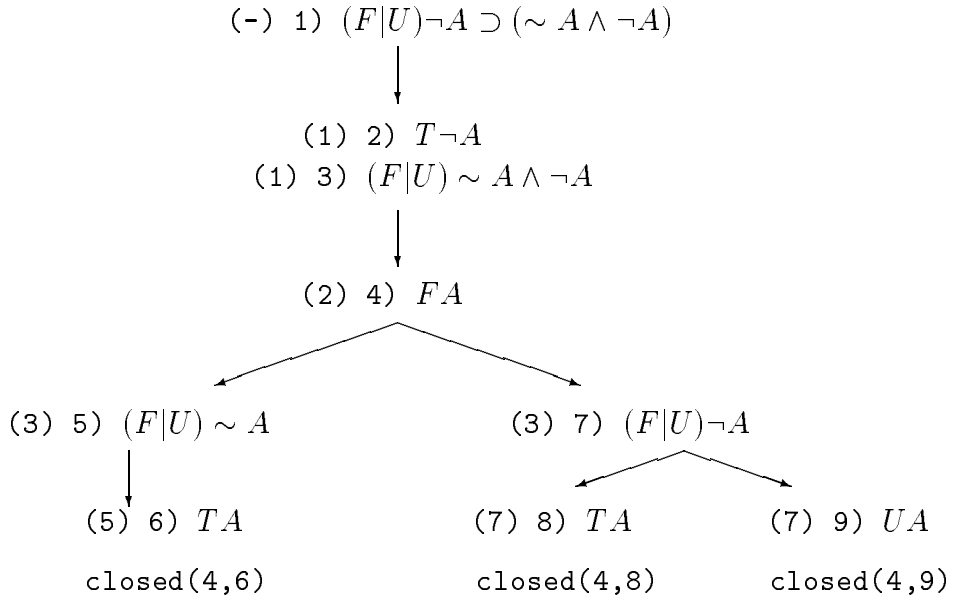
- *It contains a* **complementary atom set**, *i.e. signed atomic formulas $S_{i_1}(p), \ldots, S_{i_n}(p)$ with $\bigcap_{j=1}^{n} i_j = \emptyset$*
- *It contains a non-atomic signed formula for which no rule is defined*[4]*.*

A **branch** *that is not closed is called* **open**. *An open branch, for which any rule application yields formulas, that are already on the branch, is called* **exhausted**. *A* **tableau** *is called* **closed** *if each of its branches is closed, and* **open** *otherwise. A tableau is called* **complete** *if each of its branches is either closed or exhausted.*

Example 2.5 *We prove that the formula $\neg A \supset (\sim A \wedge \neg A)$ is a $\mathcal{L}_3$-tautology by constructing a closed tableau with root $(F|U)\neg A \supset (\sim A \wedge \neg A)$. The existence of such a closed tableau tells us that in any possible valuation the truth value of the formula in question can neither be 0 nor 1, so we can conclude that indeed it must be a tautology. In the following tableau the numbers of the formulas are marked*

[4]This case corresponds to closure of branches that contain e.g. $T\ \bot$ in classical logic.

with right brackets, whereas the numbers of the parent formulas are indicated by full bracketed numbers. At the end of each branch the numbers of the complementary formulas are stated. The tableau rules used here refer to the Appendix.



The proof of this theorem in Carnielli's system requires the construction of two trees, one of which is considerably more complex than the one above.

We close this section with an appropriate definition of satisfiability of branches and tableaux, which we shall need for the formulation of the main lemma in the soundness proof.

Definition 2.5 (Satisfiability of Branches)
A set B of **signed formulas** *is called* **satisfiable**, *if there is a valuation v such that for all $S_i(X) \in B$ $v(X) \in i$ holds. In this case we say that v is a* **model** *for B. A* **branch** *is satisfiable iff its label set is. A* **tableau** *is satisfiable iff it contains at least one satisfiable branch.*

3 Soundness and Completeness

3.1 Soundness

Lemma 3.1 (Satisfiability Preservation of Tableau Rules) *Let T be a satisfiable tableau and suppose T' was created by rule application to an arbitrary formula in T. Then T' is also satisfiable.*

Proof: T contains at least one satisfiable branch B. If the formula in the rule application was not in B, B is unchanged and hence still satisfiable.

On the other hand, let $S_i(F(X_1, \ldots, X_m)) \in B$ be the formula that supplied the premise for rule application and let v be a valuation that satisfies B. For such a valuation by definition we always have $v(F(X_1, \ldots, X_m)) \in i$. Since v is a homomorphism,

$$v(F(X_1, \ldots, X_m)) = f(v(X_1), \ldots, v(X_m)) \in i$$

holds.

Let $S_{j_1}(X_{i_1}) \circ \ldots \circ S_{j_t}(X_{i_t})$ be an extension obtained by applying (T0) to $(v(X_1), \ldots, v(X_m)) \in f^{-1}(i)$. Take any i_k: By (T0) we have $v(X_{i_k}) \in j_k$. Together with the assumption that v was a model for B we have the satisfiability of $B \cup \{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$, which concludes the proof. ■

Now soundness follows easily:

Theorem 3.1 (Soundness) *Let A be any L-formula. If there is a closed tableau with root $S_{N-D}(A)$, then A is a tautology.*

Proof: Let T be such a tableau. T cannot be satisfiable. For assume B is an arbitrary branch in T. Since T is closed, B either contains a complementary atom set or a signed formula with no corresponding rule definition. Obviously, no valuation v that satisfies B can exist, since in the first case, it would be no mapping, in the second case it would be only partially defined. Since this holds for arbitrary branches, T is not satisfiable.

The next step is to show by a straightforward induction, using the above lemma, that any tableau with satisfiable root also must be satisfiable.

Together we have that T is not satisfiable, so the root $S_{N-D}(A)$ is not satisfiable, which means by definition for all valuations v that $v(A) \notin N - D$ iff for all valuations v $v(A) \in D$ iff A is a tautology. ■

3.2 Completeness

The completeness proof for our system will be quite straightforward and will closely follow the lines of standard tableau completeness proofs (see e.g. [Fitting, 1990]), but in order to be able to deal with generalized signs we will have to make appropriate modifications of the definitions of Hintikka Set and Analytic Consistency Property. Then we proceed as usual, proving first Hintikka's Lemma, and second a model existence theorem, which in turn yields completeness. For the sake of modularity and flexibility we prefer the formulation with analytic consistency properties over a more direct one. Then it is easy to extend the proofs to first-order formulas or infinite sets of formulas. Also other standard results like strong completeness and compactness may easily be obtained, though we do not include them here.

Definition 3.1 (Hintikka Set)
A set H of $I_{\mathcal{L}}$-signed formulas is called a **Hintikka set** *iff it is atomically consistent and downward saturated, or more precisely, if the following two conditions hold:*

(H1) *For all propositional variables $p \in L_0$: If $S_{i_1}(p), \ldots, S_{i_n}(p) \in H$ then $\bigcap_{j=1}^{n} i_j \neq \emptyset$.*

(H2) *If $S_i(F(X_1, \ldots, X_m)) \in H$ then $\pi_{i,F}$ is defined and at least one of the hereby determined extensions*[5] *$\{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$ is also in H.*

A Hintikka set H' is called **saturated Hintikka set** *or* **model set** *iff in addition to the above stated conditions it is atomically complete and upward saturated, i.e.*

[5] Here and in the following we view extensions as sets.

(H3) *For all propositional variables $p \in L_0$ there exists an $i \in I_{\mathcal{L}}$ such that $S_i(p) \in H'$.*

(H4) *If $i \in I_{\mathcal{L}}$ then $S_i(F(X_1, \ldots, X_m)) \in H'$, whenever at least one of the extensions $\{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$ determined by $\pi_{i,F}$ is in H'.*

Note that by (H1) and (H2) it is impossible that $S_\emptyset(X)$ for any $X \in L$ ever occurs in a Hintikka set.

Theorem 3.2 (Hintikka's Lemma) *Every Hintikka set H can be extended to a saturated Hintikka set H'.*

Proof: Let H be a Hintikka set and $L_0 := \{p_i \mid i \in Nat\}$ an enumeration of the propositional variables. We extend H to a saturated Hintikka set H' in the following way:

$H_0 = H \cup \{S_{\{0\}}(p_i) \mid i \in Nat \text{ and } S_j(p_i) \in H \text{ for no } j \in I_{\mathcal{L}}\}$
$H_{i+1} = H_i \cup \{S_j(F(X_1, \ldots, X_m)) \mid j \in I_{\mathcal{L}}, \pi_{j,F} \text{ defined and at least one of the extensions } \{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\} \text{ determined by } \pi_{j,F} \text{ is in } H_i\}$
$H' = \bigcup\{H_i \mid i \in Nat\}$

First we extend H such that it assigns a definite truth value (we took 0, but it is arbitrary) to each variable not already occurring in H, then we inductively take all L-formulas into account.

For H' (H1) holds, because let $p \in L_0$, then either there exists a $j \in I_{\mathcal{L}}$ such that $S_j(p) \in H$ and nothing is changed by the construction, so (H1) still holds, or $S_j(p) \in H$ for no $j \in I_{\mathcal{L}}$, then $S_{\{0\}}(p)$ is added and since this is the only occurrence of p (H1) holds trivially. (H3) and (H4) hold by construction of H'. To see that (H2) holds, let $S_j(F(X_1, \ldots, X_m)) \in H'$ and $j \in I_{\mathcal{L}}$. Then either already $S_j(F(X_1, \ldots, X_m)) \in H$ and (H2) is inherited from H, or $S_j(F(X_1, \ldots, X_m))$ was generated during the construction in some $H_i, i > 0$. Then, by definition, at least one of the extensions $\{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$ determined by $\pi_{j,F}$ is in H_{i-1} and (H2) is inherited from H_{i-1}. ■

Definition 3.2 (Analytic Consistency Property)
A family Γ ranging over sets of $I_{\mathcal{L}}$-signed formulas is called an **Analytic Consistency Property (ACP)** *iff for all $K \in \Gamma$ the following conditions hold:*

(F) Γ *is of* finite character, *i.e. K belongs to Γ iff all finite subsets of K belong to Γ.*

(ACP1) *For all propositional variables $p \in L_0$ holds:*
If $S_{i_1}(p), \ldots, S_{i_n}(p) \in K$ then $\bigcap_{j=1}^{n} i_j \neq \emptyset$.

(ACP2) *If $S_i(F(X_1, \ldots, X_m)) \in K$ then $\pi_{i,F}$ is defined and for at least one of the extensions $C = \{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$ $K \cup C \in \Gamma$.*

If $K \in \Gamma$ then K is called **Γ-consistent**. *While Γ has finite character, from $K' \subseteq K$ and $K \in \Gamma$ we always have $K' \in \Gamma$.*

Theorem 3.3 (Model Existence) *Let Γ be an ACP and K a set of $I_{\mathcal{L}}$-signed formulas. If K is Γ-consistent then there exists a valuation v, such that $v(X) \in j$ holds, whenever $S_j(X) \in K$, in other words, v is a model for K.*

Proof: In a first step we will carry out a Lindenbaum-type construction restricted to ACP's in order to find a L^*-maximal element M in Γ (this corresponds to Tukey's lemma in the denumerable case), then we show that M is a Hintikka set, so we can use it to define an appropriate valuation.

Let $\{Z_1, Z_2, \ldots\}$ be an enumeration of all signed formulas in L^* and define C_n for $n \geq 0$ as follows:

$$C_0 = K$$
$$C_{n+1} = \begin{cases} C_n \cup \{Z_i\} & \text{if } C_n \cup \{Z_n\}\Gamma\text{-konsistent} \\ C_n & \text{otherwise} \end{cases}$$

Clearly, all C_n are members of Γ and, ordered by inclusion, are building a chain in Γ. We define

$$M = \bigcup_{n \geq 0} C_i$$

and thus have:

1. M is L^*-maximal in Γ, since
   (a) Let $K \subseteq M$ be arbitrary, but finite. Hence we have some C_n with $K \subseteq C_n$ and while $C_n \in \Gamma$, we have also that $K \in \Gamma$ because of the finite character of Γ. Thus we have $K \in \Gamma$ for all finite $K \subseteq M$ and so $M \in \Gamma$, again because of the finite character of Γ.
   (b) Assume there were $M' \subseteq L^*$ with $M \subset M' \in \Gamma$, $M \neq M'$. So we must have some $Z_n \in M'$ with $Z_n \notin M$. By definition, we have $C_n \subseteq M \subset M'$, hence $C_n \cup \{Z_n\} \subseteq M'$. By the finite character of Γ we know that $C_n \cup \{Z_n\} \in \Gamma$. But then, by definition, $C_{n+1} = C_n \cup \{Z_n\}$ thus yielding $Z_n \in M$, which is a contradiction.
2. (a) and (ACP1) imply (H1), (b) and (ACP2) imply (H2) for M, so M is indeed a Hintikka set. According to Hintikka's lemma we can extend M to a saturated Hintikka set $\bar{M}$.

It remains to show that $\bar{M}$ determines a model for K. For this purpose we fix an arbitrary v for $p \in L_0$ such that:

$$v(p) \in i \text{ iff } S_i(p) \in \bar{M}$$

Since $\bar{M}$ is a saturated Hintikka set, (H1) guarantees that v is welldefined, (H3) that it is totally defined on L_0. We extend v to a homomorphism from L to $\mathcal{A}$ and show by induction on the depth of X that $X \in L$ and $S_j(X) \in \bar{M}$ imply $v(X) \in j$[6].

The case when X is atomic is settled by definition of v.

[6]Note that in the proof we don't make use of (H4). In fact, using (H4) we could show the other direction as well, namely that for any $X \in L$ there exists a $j \in I_{\mathcal{L}}$ such that $v(X) \in j$ implies $S_j(X) \in \bar{M}$.

Suppose that $S_j(X) = S_j(F(X_1, \ldots, X_m)) \in \bar{M}$. According to (H2) there is at least one extension determined by $\pi_{j,F}$ with
$\{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\} \subseteq \bar{M}$. The induction hypothesis yields $v(X_{i_k}) \in j_k$ for $1 \leq k \leq t$. With this we can conclude, using the homomorphism h that defines the extension:

$$\begin{array}{l} v(F(X_1, \ldots, X_m)) \\ = f(v(X_1), \ldots, v(X_{i_1}), \ldots, v(X_{i_t}), \ldots, v(X_m)) \quad (v \text{ hom.}) \\ \in j \text{ (by induction hypothesis, (T1), (T2))} \end{array}$$

So we have indeed constructed a model for $\bar{M}$ and the theorem follows from the fact that $K \subseteq \bar{M}$.

■

Theorem 3.4 (Completeness) *If A is a tautology then there exists a closed tableau with root $S_{N-D}(A)$.*

Proof: Since A is a tautology, for all valuations $v(A) \in D$ must hold. Now suppose no closed tableau with root $S_{N-D}(A)$ exists. It follows that there exists at least one exhausted tableau with root $S_{N-D}(A)$, containing an exhausted, open branch M. Define $\mathcal{B}$ as the set of all finite tableau branches that cannot be closed. For all $B \in \mathcal{B}$ we have:

- For all propositional variables $p \in L_0$ holds:
  If $S_{i_1}(p), \ldots, S_{i_n}(p) \in B$ then $\bigcap_{j=1}^{n} i_j \neq \emptyset$, otherwise B would be closed.
- If $S_i(F(X_1, \ldots, X_m)) \in B$ then $\pi_{i,F}$ is defined and for at least one of the hereby determined extensions
  $C = \{S_{j_1}(X_{i_1}), \ldots, S_{j_t}(X_{i_t})\}$ $B \cup C \in \mathcal{B}$. For assume $\pi_{i,F}$ were not defined, then B were closed and if for no C $B \cup C \in \mathcal{B}$, then B could be closed later on.
- Clearly, $\mathcal{B}$ has finite character.

Putting the facts together, we have that $\mathcal{B}$ is an ACP and $\{S_{N-D}(A)\}$ is $\mathcal{B}$-consistent, since $\{S_{N-D}(A)\} \subseteq M \in \mathcal{B}$. Now, from the model existence theorem we know that there exists a valuation v with $v(A) \in N - D$ and this is the contradiction we have been looking for.

■

4 Conclusion

We presented a generic tableau proof system for propositional multiple-valued logics that is more efficient and elegant than its predecessors. For support of the efficiency claim, consider the $\mathcal{L}_3$-tautology $((\ldots(p_1 \vee p_2) \vee \ldots \vee p_n) \vee \sim p_1)$. It is easy to see that there is a proof of linear size wrt n in our system, while the shortest proof in Carnielli's system is of exponential size wrt n.

The achievement was gained by generalizing signs from truth values to sets of truth values. We emphasize that the improvements were made on a logical rather than on an algorithmical level by enriching the language, so we can use our tableau system for standard tableau

provers with minor modifications. Another advantage of this approach is the compatibility with techniques that are set on the bookkeeping level e.g. indexing schemes or weighting.

The extension of the technique to first-order multiple-valued logics is possible if some restrictions on allowed signs, connectives and quantifiers are imposed. A follow-up to this paper concerned with multiple-valued predicate logic is available [Hähnle, 1991]. To keep the paper short we have excluded the notion of *systematic tableaux* which is needed for mechanizing tableau proofs and which requires no further modifications for the use in our framework.

Acknowledgements

I would like to thank Peter H. Schmitt for many helpful discussions and suggestions during the composition of this paper. I took advantage from the comments of Alan Shepherd and the remarks of an anonymous referee.

Appendix: A Tableau System for $\mathcal{L}_3$

Rules for $\vee$:

$$\frac{T\ X_1 \vee X_2}{T\ X_1 \mid T\ X_2} \qquad \frac{U\ X_1 \vee X_2}{\begin{array}{l|l} U\ X_1 & (F|U)\ X_1 \\ (F|U)\ X_2 & U\ X_2 \end{array}} \qquad \frac{F\ X_1 \vee X_2}{\begin{array}{l} F\ X_1 \\ F\ X_2 \end{array}} \qquad \frac{(F|U)\ X_1 \vee X_2}{\begin{array}{l} (F|U)\ X_1 \\ (F|U)\ X_2 \end{array}}$$

Rules for $\wedge$:

$$\frac{T\ X_1 \wedge X_2}{\begin{array}{l} T\ X_1 \\ T\ X_2 \end{array}} \qquad \frac{U\ X_1 \wedge X_2}{\begin{array}{l|l|l} U\ X_1 & U\ X_1 & T\ X_1 \\ T\ X_2 & U\ X_2 & U\ X_2 \end{array}} \qquad \frac{F\ X_1 \wedge X_2}{F\ X_1 \mid F\ X_2} \qquad \frac{(F|U)\ X_1 \wedge X_2}{(F|U)\ X_1 \mid (F|U)\ X_2}$$

Rules for $\supset$:

$$\frac{T\ X_1 \supset X_2}{(F|U)\ X_1 \mid T\ X_2} \qquad \frac{U\ X_1 \supset X_2}{\begin{array}{l} T\ X_1 \\ U\ X_2 \end{array}} \qquad \frac{F\ X_1 \supset X_2}{\begin{array}{l} T\ X_1 \\ F\ X_2 \end{array}} \qquad \frac{(F|U)\ X_1 \supset X_2}{\begin{array}{l} T\ X_1 \\ (F|U)\ X_2 \end{array}}$$

Rules for $\neg$:

$$\frac{T\ \neg X}{F\ X} \qquad \frac{U\ \neg X}{U\ X} \qquad \frac{F\ \neg X}{T\ X} \qquad \frac{(F|U)\ \neg X}{U\ X \mid T\ X}$$

Rules for $\sim$:

$$\frac{T\ \sim X}{(F|U)\ X} \qquad \text{(no rule defined for } U\ \sim X\text{)} \qquad \frac{F\ \sim X}{T\ X} \qquad \frac{(F|U)\ \sim X}{T\ X}$$

Rules for ∇:

$$\frac{T\ \neg X}{U\ X \mid T\ X} \quad \text{(no rule defined for } U\ \nabla X\text{)} \quad \frac{F\ \nabla X}{F\ X} \quad \frac{(F|U)\ \nabla X}{F\ X}$$

References

[Beth, 1986] E. W. Beth. Semantic entailment and formal derivability. In Karel Berka & Lothar Kreiser, editors, *Logik-Texte. Kommentierte Auswahl zur Geschichte der modernen Logik*, pages 262–266. Akademie–Verlag, Berlin, 1986.

[Burris & Sankappanavar, 1981] Stanley Burris & H.P. Sankappanavar. *A Course in Universal Algebra*, volume 78 of *Graduate Texts in Mathematics*. Springer, New York, 1981.

[Carnielli, 1987] Walter A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *Journal of Symbolic Logic*, 52(2):473–493, June 1987.

[Dueck, 1988] Gerhard W. Dueck. *Algorithms for the Minimization of Binary and Multiple–Valued Logic Functions*. PhD thesis, University of Manitoba, Winnipeg, 1988.

[Fenstad *et al.*, 1985] Jens Erik Fenstad, Per-Kristian Halvorsen, Tore Langholm, & Johan von Benthem. Equations, schemata and situations: A framework for linguistic semantics. Technical Report CSLI–85–29, Center for the Studies of Language and Information Stanford, 1985.

[Fitting, 1983] Melvin C. Fitting. *Proof Methods for Modal and Intutionistic Logics*. Reidel, Dordrecht, 1983.

[Fitting, 1989] Melvin C. Fitting. Negation as refutation. In *LICS 1989 Proceedings*, 1989.

[Fitting, 1990] Melvin C. Fitting. *First–Order Logic and Automated Theorem Proving*. Springer, New York, 1990.

[Hähnle, 1990] Reiner Hähnle. Spezifikation eines Theorembeweisers für dreiwertige First–Order Logik. IWBS Report 136, Wissenschaftliches Zentrum, IWBS, IBM Deutschland, September 1990.

[Hähnle, 1991] Reiner Hähnle. Uniform notation of tableaux rules for multiple–valued logics. In *Submitted for International Symposium on Multiple–Valued Logic, Victoria*, 1991.

[Kropf & Wunderlich, 1990] T. Kropf & H.-J. Wunderlich. Hierarchische Testmustergenerierung für sequentielle Schaltungen mit Hilfe von Temporaler Logik. II. ITG/GI Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen, 1990.

[Oppacher & Suen, 1986] F. Oppacher & E. Suen. Controlling deduction with proof condensation and heuristics. In Jörg H. Siekmann, editor, *Proc. 8th International Conference on Automated Deduction*, pages 384–393, 1986.

[Oppacher & Suen, 1988] F. Oppacher & E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69 – 100, 1988.

[Schmitt, 1989] Peter H. Schmitt. Perspectives in multi–valued logic. Proceedings International Scientific Symposium on Natural Language and Logic, Hamburg, 1989.

[Sheperdson, 1989] John C. Sheperdson. A sound and complete semantics for a version of negation as failure. *Theoretical Computer Science*, 65:343–371, 1989.

[Smullyan, 1968] Raymond Smullyan. *First-Order Logic.* Springer, New York, second edition, 1968.

[Stachniak, 1990] Z. Stachniak. Note on resolution approximation of many-valued logics. In *20th International Symposium on Multiple-Valued Logic, Charlotte*, pages 204–209, May 1990.

[Surma, 1984] Stanisław J. Surma. An algorithm for axiomatizing every finite logic. In David C. Rine, editor, *Computer Science and Multiple–Valued Logics*, pages 143–149. North–Holland, Amsterdam, 1984.

[Wójcicki, 1988] Ryszard Wójcicki. *Theory of Logical Calculi.* Reidel, Dordrecht, 1988.

[Wolper, 1981] Pierre Wolper. Temporal logic can be more expressive. In *Proceedings 22nd Annual Symposium on Foundations of Computer Science*, pages 340 – 348, 1981.