Long Version of Paper in: Proceedings Interational Symposium on Multiple-Valued Logic, Victoria 1991, IEEE Press, pp. 238 – 245

Uniform Notation of Tableau Rules 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

February 1991

Abstract

Starting from the general tableau-based framework for multiple-valued logics presented in [Hähnle, 1990] we show that the advantages of Smullyan's uniform notation [Smullyan, 1968] for classical logic can be made available to multiple-valued logic as well. The result is a system that serves as theoretical basis for automated theorem proving in multiple-valued logics.

Introduction

We assume that the reader is familiar with the standard reference on tableau calculi, [Smullyan, 1968]. Smullyan introduces so-called uniform notation for tableau rules, which plays a key role in the concise presentation of his calculus as well as in the proofs of the results. Uniform notation in the propositional case exploits the fact that the semantic definitions of some propositional operators can be written as a conjunction (resp. disjunction) of its (possibly negated) arguments. For example, the semantics of propositional implication can be given in the following way:

For any valuation v and formulas x, y:

$$v(x \supset y) = \mathbf{t} \text{ iff } v(x) = \mathbf{f} \text{ or } v(y) = \mathbf{t}$$

The tableau rules of any propositional operator in Table 1 can be represented as instances of only two rule schemes, which are said to be of type α (conjunctive) and type β (disjunctive) resp. (see the two leftmost rule schemes in Table 3).

Thus, in a signed tableau calculus the rule corresponding to a formula with leading operator $\circ$ and sign T or F can be defined by specifying its type α or β and the signs of its direct subformulas. To continue the example from above, the rule corresponding to $x \supset y$ and T is of type β and the signs of its subformulas x and y are F and T resp. A tableau proof of the validity of a formula X may be represented as a tree starting with root FX stating that X is possibly false in some interpretation. Then the tree is expanded by systematically applying the rules first to FX and then to the formulas generated by that rule application and so on. X is proved if finally each branch contains a contradiction, i.e. there is no interpretation that makes X false. On the other hand, a contradiction-free branch immediately yields a counter example.

Rule schemes for the quantifier rules of first-order logic can be devised in a similar way. Uniform notation makes it possible to have logics with a huge variety of built-in operators while at the same time the proofs of soundness and completeness are essentially the same as if one had only one built-in operator (besides negation). On the other hand the possibility of adding certain provisos to a rule scheme provides considerable flexibility, so that uniform notation has been successfully used as a tool e.g. in tableau systems for temporal logics [Emerson & Halpern, 1985, Wolper, 1981], modal and inituitionistic logics [Fitting, 1983] as well as for specifically tuned calculi in automated theorem proving [Fitting, 1990]. The purpose of this paper is to show how uniform notation

*This work is supported by IBM Germany as part of the TCG project, which is a collaboration of the University of Karlsruhe and the IWBS at IBM Germany in Heidelberg.

may be also used for multiple-valued logics to give a tableau system for a wide class of operators in arbitrary logics with finitely many truth values that can serve as well as a theoretical basis for automated theorem proving.

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]).

In Section 1 we present a tableau system for arbitrary propositional logics with finitely many truth values, which will be a signed version of tableau calculus with a generalized notion of signs. This system was presented in [Hähnle, 1990] and will serve as a framework for the following sections. In Section 2 we will give a more or less direct adaption of Smullyan's tableau system to multiple-valued logics, while in Section 3 a slightly more general system for a wide class of multiple-valued operators is presented, but still in uniform notation style. In Section 4 we take a look at functional completeness of the respective logics, in Section 5 we sketch a possible extension of our ideas to quantified multiple-valued logic and summarize what can be achieved with our system in automated theorem proving.

1 A Tableau System for arbitrary Finitely-Valued Propositional Logics

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 from $\mathcal{F}$. By 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 set of **truth values** and $D \subseteq N$ the set of **designated truth values**[1]. 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. X is called **tautology**, if any $\mathcal{A}$-valuation v is also a model for $\{X\}$. 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 semantics 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	1	1	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

Let us refer to this logic with the symbol $\mathcal{L}_3$.

[1] In multiple-valued logics the designated truth values are those that support the validity of a statement. We do not assume any specific structure on the set of truth values nor do we associate any philosophical interpretation with the truth values.

Given any logic of this kind, we would like to have a sound and complete tableau proof system for it. This task was begun by Surma [Surma, 1984] and completed by Carnielli [Carnielli, 1987], who provided a generic signed tableau proof system as proposed for multiple-valued first-order logics with arbitrary logical connectives and generalized quantifiers. Unfortunately, Carnielli's system is not particularly well suited for automated theorem proving (which was not his aim), because it works by simply introducing a sign for each truth value in N and thus extending the analysis from two to multiple truth values. One consequence is that the exclusion of all interpretations of a formula X, which assign a non-designated truth value $\bar{d}_i \in \bar{D} = N - D = \{\bar{d}_1, \ldots, \bar{d}_{n-d}\}$ to X (which proves the validity of X), requires the construction of $n - d$ separate proof trees with roots $\bar{d}_1 X, \ldots, \bar{d}_{n-d} X$ resp. Also the proof trees generate many more branches than for classical logic and the simple α, β schemes are lost due to the case analysis by single truth values. This is a kind of inefficiency that is not inherent in the problem and in [Hähnle, 1990] we presented a method, how it can be overcome. Our idea is to increase the expressivity of signs in order to be able to express conditions like "$v(X) = 0$ *or* $v(X) = 1$ ", or equivalently " $v(X) \neq 2$", with a single signed formula by admitting sets of truth values as signs. To establish the validity of X it is then no longer necessary to build $n - d$ proof trees. Instead, it suffices to build a single proof tree with root $S_{N-D}X$, where the sign S_{N-D} corresponds to the set of non-designated truth values.

Definition 1.1 (Sign, Signed Formula, Satisfiable Formula)
Let L be any language and D and N 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 $\{S_{\{0\}}, \ldots, S_{\{n-1\}}\} \subseteq \mathcal{S}_\mathcal{L}$[2]. 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 $f^{I_\mathcal{L}}(i_1, \ldots, i_m) = \{f^A(j_1, \ldots, j_m) \mid j_k \in i_k, 1 \leq k \leq m\}$.*

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**.

A signed formula $S_i(X)$ is **satisfiable** *in $\mathcal{L}$ iff there is a valuation v, for which $v(X) \in i$ holds. Thus satisfiability of an unsigned formula X is equivalent to satisfiability of $S_D(X)$.*

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 1.2 *For $\mathcal{L}_3$ define $\{S_{\{0\}}, S_{\{1\}}, S_{\{2\}}, S_{\{0,1\}}\}$ as the set of signs, which for convenience we rewrite as $\{F, U, T, (F|U)\}$. The intended interpretation of a signed formula $(F|U)(X)$ is "$v(X) = 0$ or $v(X) = 1$".*

Definition 1.2 (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 linear subtrees (denote with $\circ \ldots \circ$ a linear tree)*

> *$S_{j_1}(X_{i_1}) \circ \ldots \circ S_{j_t}(X_{i_t})$ for which $j_1, \ldots, j_t \in I_\mathcal{L}$, $t \leq m$ and $H_i(F; j_1, \ldots, j_t)$ (see below) holds,*

called **extensions**[3].

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*[4].

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 $1 \leq k \leq t$.*

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

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

[4] 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.

(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.

Example 1.3 *The rules for implication and $(F|U)$, implication and U, ∇ and $(F|U)$ in $\mathcal{L}_3$ are*

$$\frac{(F|U)(A \supset B)}{\begin{array}{c|c} UA & TA \\ (F|U)B & (F|U)B \end{array}} \qquad \frac{U(A \supset B)}{\begin{array}{c|c} UA & TA \\ (F|U)B & UB \end{array}} \qquad \frac{(F|U)\nabla A}{FA}$$

For U and $\sim$ (weak negation) exists no corresponding rule.

A complete tableau for a finite set of $I_{\mathcal{L}}$-signed formulas is now created in the usual manner. Only a minor modification has to be made to the conditions for branch closure:

Definition 1.3 (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*[5].

Example 1.4 *We prove that the formula $(\sim A \supset A) \supset \nabla A$ is a $\mathcal{L}_3$-tautology by constructing a closed tableau (Figure 1) with root $(F|U)(\sim A \supset A) \supset \nabla A$.*

Theorem 1.1 (Soundness, Completeness; [Hähnle, 1990]) *A is a tautology iff there exists a closed tableau with root $S_{N-D}(A)$.*

Although this system yields quite nice tableau rules for most logics that can be found in the literature, it has still some disadvantages. For example, no constructive algorithm[6] is given to determine the tableau rules, and it is very tedious to check out the required homomorphisms by hand. Also, the common structure of the rules is merely that their conclusion is a disjunction of conjunctions, there is nothing like a uniform notation, which would be desirable if one is concerned with automated theorem proving. Finally, the number of signs that is needed to get a useful set of rules may be exponential in the number of truth values and it is difficult to determine a set of signs that yields a compact tableau system, therefore, in the next two sections we present two versions of a tableau system with generalized signs that are very similar to Smullyan's original system for classical logic, thus overcoming the above mentioned disadvantages. A price, of course, will have to be paid: logical connectives are restricted in a certain way, such that "nice" rules result. On the other hand we hope that the resulting class of logics is still large enough to be interesting, and we will try to show this.

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

[6] As pointed out above, one could use existing multiple-valued minimization techniques, but usually these yield only near minimal solutions. The problem of finding a minimal sum-of-product representation for a given function is known to be NP-complete.

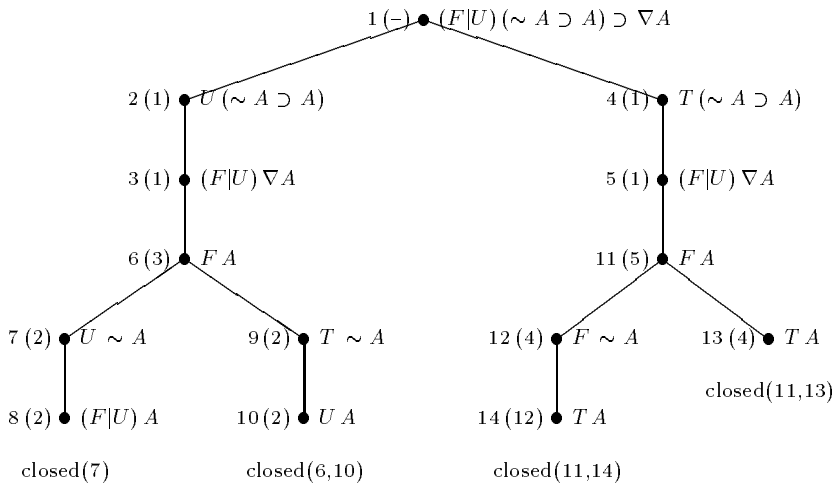


Figure 1: Proof of $(\sim A \supset A) \supset \nabla A$

2 Primary Multiple-Valued Connectives

Consider multiple-valued logics that consist solely of generalized versions of classical primary connectives[7] in the sense of Smullyan [Smullyan, 1968]. For these connectives tableau rules can be supplied that resemble the classical rules for α and β formula components very closely. First we have to state what we mean by generalized versions of classical primary connectives. For convenience in Table 1 we repeat Smullyan's primary connectives.

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

$\downarrow$	0	1
0	1	0
1	0	0

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

$\uparrow$	0	1
0	1	1
1	1	0

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

$\not\supset$	0	1
0	0	0
1	1	0

$\subset$	0	1
0	1	0
1	1	1

$\not\subset$	0	1
0	0	1
1	0	0

Table 1: Primary propositional connectives

Let $N = \{0, \ldots, n-1\}$ be a set of truth values, $D = \{n-d, \ldots, n-1\}$ the set of designated truth values and $\bar{D} = \{0, \ldots, n-d-1\}$ the set of non-designated truth values, i.e. $N = D \cup \bar{D}$. For any truth value $i \in N$ we define its **conjugate** $i^* = n-1-i$. As usual, the conjugate of a set is the set of its conjugates, i.e. if $S \subseteq N$ then $S^* = \{i^* : i \in S\}$. For reasons to be seen later we include the conjugation operator also in our logic as another name for negation.

Let $\mathcal{L}^n_{Sm}$ be the n-valued logic whose language consists of the primary connectives plus negation, and whose semantic is given by the following evaluation rules:[8]

$v(\neg x) = v(x^*) = v^*(x)$
$v(x \vee y) = max\{v(x), v(y)\}$
$v(x \downarrow y) = min\{v^*(x), v^*(y)\}$
$v(x \wedge y) = min\{v(x), v(y)\}$
$v(x \uparrow y) = max\{v^*(x), v^*(y)\}$
$v(x \supset y) = max\{v^*(x), v(y)\}$
$v(x \not\supset y) = min\{v(x), v^*(y)\}$

[7] The primary connectives are exactly those, that can be characterized by an α or β rule.

[8] Note, that in the case when $N = \{0, 1\}$, $D = 1$ the classical primary connectives with the truth tables as shown above will result.

$v(x \subset y) = max\{v(x), v^*(y)\}$
$v(x \not\subset y) = min\{v^*(x), v(y)\}$

α	α_1	α_2
$D\ x \wedge y$	$D\ x$	$D\ y$
$D\ x \downarrow y$	$D^*\ x$	$D^*\ y$
$D\ x \not\supset y$	$D\ x$	$D^*\ y$
$D\ x \not\subset y$	$D^*\ x$	$D\ y$
$\bar{D}\ x \vee y$	$\bar{D}\ x$	$\bar{D}\ y$
$\bar{D}\ x \uparrow y$	$\bar{D}^*\ x$	$\bar{D}^*\ y$
$\bar{D}\ x \subset y$	$\bar{D}\ x$	$\bar{D}^*\ y$
$\bar{D}\ x \supset y$	$\bar{D}^*\ x$	$\bar{D}\ y$
$D^*\ x \vee y$	$D^*\ x$	$D^*\ y$
$D^*\ x \uparrow y$	$D\ x$	$D\ y$
$D^*\ x \subset y$	$D^*\ x$	$D\ y$
$D^*\ x \supset y$	$D\ x$	$D^*\ y$
$\bar{D}^*\ x \wedge y$	$\bar{D}^*\ x$	$\bar{D}^*\ y$
$\bar{D}^*\ x \downarrow y$	$\bar{D}\ x$	$\bar{D}\ y$
$\bar{D}^*\ x \not\supset y$	$\bar{D}^*\ x$	$\bar{D}\ y$
$\bar{D}^*\ x \not\subset y$	$\bar{D}\ x$	$\bar{D}^*\ y$

β	β_1	β_2
$D\ x \vee y$	$D\ x$	$D\ y$
$D\ x \uparrow y$	$D^*\ x$	$D^*\ y$
$D\ x \subset y$	$D\ x$	$D^*\ y$
$D\ x \supset y$	$D^*\ x$	$D\ y$
$\bar{D}\ x \wedge y$	$\bar{D}\ x$	$\bar{D}\ y$
$\bar{D}\ x \downarrow y$	$\bar{D}^*\ x$	$\bar{D}^*\ y$
$\bar{D}\ x \not\supset y$	$\bar{D}\ x$	$\bar{D}^*\ y$
$\bar{D}\ x \not\subset y$	$\bar{D}^*\ x$	$\bar{D}\ y$
$D^*\ x \wedge y$	$D^*\ x$	$D^*\ y$
$D^*\ x \downarrow y$	$D\ x$	$D\ y$
$D^*\ x \not\supset y$	$D^*\ x$	$D\ y$
$D^*\ x \not\subset y$	$D\ x$	$D^*\ y$
$\bar{D}^*\ x \vee y$	$\bar{D}^*\ x$	$\bar{D}^*\ y$
$\bar{D}^*\ x \uparrow y$	$\bar{D}\ x$	$\bar{D}\ y$
$\bar{D}^*\ x \subset y$	$\bar{D}^*\ x$	$\bar{D}\ y$
$\bar{D}^*\ x \supset y$	$\bar{D}\ x$	$\bar{D}^*\ y$

Table 2: α and β formulas and their components

This definition provides a natural generalization of classical primary connectives to the multiple-valued case[9].

Let the set of signs of $\mathcal{L}^n_{Sm}$ be $\{S_D, S_{D^*}, S_{\bar{D}}, S_{\bar{D}^*}\}$ which for convenience we abbreviate with $\{D, D^*, \bar{D}, \bar{D}^*\}$. Table 2 list the components of α and β type formulas, which together with the usual tableau rule schemes and the obvious rules for negation (see Table 3) constitute a sound and complete tableau system for each of the logics $\mathcal{L}^n_{Sm}$. Note that the number of signs does not depend on N or D.

So we have constructed a uniform notation style tableau system for testing validity of formulas in any logic that contains at most negation plus the generalized versions of the classical primary operators. To test validity of a formula, say A, all we have to do is to construct a single closed tableau with root $\bar{D}\ A$.

Automated tableau construction is possible with any tableau based theorem prover that can be modified to handle four instead of two signs. On the other hand, the well developed classical proofs of tableau completeness and soundness (see e.g. in [Fitting, 1990]) can be carried over from classical logic with only minor modifications. This is particularly useful in the case of implementation oriented tableau proof systems, where the completeness proof tends to be awkward.

The reason why the conjugates of D and $\bar{D}$ must also be present is, that in general the equality $\bar{D} = D^*$ does not hold, as our example $\mathcal{L}_3$ shows. The signs D and $\bar{D}$ represent designated and non-designated truth values, whereas the signed formula $D^*\ A$ is equivalent to $D\ \neg A$. In classical logic negation and non-designatedness coincide. This is the reason why logically equivalent signed and non-signed versions of classical tableau systems can be formulated without extending the set of signs or the set of logical connectives.

$$\frac{\alpha}{\alpha_1 \\ \alpha_2} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{D\ \neg x}{D^*\ x} \qquad \frac{\bar{D}\ \neg x}{\bar{D}^*\ x} \qquad \frac{D^*\ \neg x}{D\ x} \qquad \frac{\bar{D}^*\ \neg x}{\bar{D}\ x}$$

Table 3: Tableau rules for α and β formulas and for negation

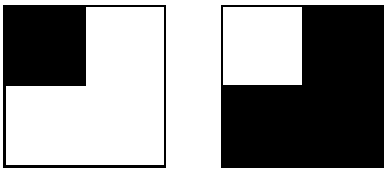
In the next section we give a more concise formulation of the components of α and β formulas for a larger class of logics. The price will be an increase in the number of involved signs.

[9] This is well-known for disjunction and conjunction (see e.g. [Rescher, 1969]), but it applies to the other operators as well. Note, that we can define four of the operators of our earlier example $\mathcal{L}_3$ this way (the remaining operators $\sim$ and ∇ are handled later).

3 A Uniform Notation Style Tableau System

In the following we will define a class of logical operators with a certain kind of regularity resulting in a particularly simple tableau proving system.

First, we observe that the form of the required coverings in the truth tables for the primary connectives is of an extremely simple form, namely one of:



where the pattern is starting from an arbitrary corner. Combining this with the fact that in multiple-valued deduction systems one usually asks whether the truth value of a formula is in D or $\bar{D}$, we are led to the following definition.

Definition 3.1 (Minterm, Circle)
Consider the k-dimensional hypercube of length n that represents the truth table of a k-ary operator $\circ$ in a n-valued logic $\mathcal{L}$. A **minterm**[10] *of $\mathcal{L}$ $\vec{x} = (x_1, \ldots, x_k)$ with $x_i \in N$ for $1 \leq i \leq k$ corresponds to exactly one entry of the cube, namely the one that contains the truth value of $\circ(x_1, \ldots, x_k)$. The* **circle** *with radius r and center $\vec{x}$ is defined as*

$$\begin{aligned} c_{\vec{x},r} \quad = \quad & \{(y_1, \ldots, y_k) \mid \textit{(i) there is a } j \leq k \\ & \textit{such that } \mid y_i - x_i \mid = r \textit{ for } i \neq j \textit{ and} \\ & \mid y_j - x_j \mid \leq j, \textit{and (ii) for all } i: \; y_i \in N\} \end{aligned}$$

We define the set of truth values that occur in the entries of a truth table for $\circ$ on a circle $c_{\vec{x},r}$ with radius r around a minterm $\vec{x}$ as $C_{\circ,\vec{x},r} = \{\circ(\vec{y}) \mid \vec{y} \in c_{\vec{x},r}\}$.

Definition 3.2 (Regular logical operator)
A logical operator $\circ$ is called **regular** *iff there is a minterm $\vec{x} \in \{(y_1, \ldots, y_k) \mid y_i \in \{0, n-1\}$ for all $1 \leq i \leq k\}$ such that:*

1. *for any $r \in N$ $C_{\circ,\vec{x},r}$ is a singleton $\{c_r\}$ and for these*
2. *$c_0 \leq, \ldots, \leq c_{n-1}$ or $c_0 \geq, \ldots, \geq c_{n-1}$ holds.*

Call $\vec{x}$ the **starting point** *of $\circ$.*

The entries in truth tables of regular operators are ordered starting from some corner with the lowest (resp. highest) truth value the operator takes on. On each circle with the starting point as its center all entries contain the same truth value. These truth values are monotically increasing (resp. decreasing) with the radius of the circles. Figure 3 shows a typical pattern, while Table 4 shows an example of a regular operator. Also all operators of our earlier example $\mathcal{L}_3$ are regular.

From now on we concentrate on binary operators and remark that all considerations carry over to arbitrary k-ary operators in a straightforward manner.

First we define parameters that are sufficient to describe all generalized primary connectives.

Definition 3.3 (Direction, Orientation, Threshold)
Let $\circ$ be a regular operator and $\vec{x}$ its starting point. Then $\delta(\circ) = < \delta_1(\circ), \delta_2(\circ) >$, where $\delta_i(\circ) = {}^*$ iff $x_i = n - 1$ and $\delta_i(\circ) = I$ otherwise[11]*, is called the* **direction** *of $\circ$. $\epsilon(\circ) = {}^*$ iff the values on circles around $\vec{x}$ are monotonically increasing with the radius and I otherwise is called the* **orientation** *of $\circ$.*

[10] We adopt here a terminology that is frequently used in minimization of logical functions, see e.g. [Dueck, 1988].
[11] Here * stands for the conjugation defined in the previous section and I stands for the identity function.

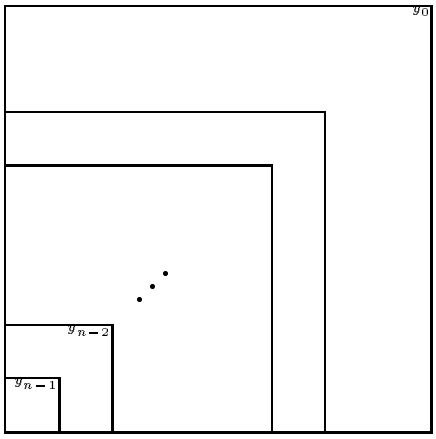


Figure 2: Typical pattern of regular operator

$\odot$	0	1	2	3
0	0	0	0	0
1	**1**	**1**	**1**	0
2	3	3	**1**	0
3	3	3	**1**	0

Table 4: Truth table for operator $\odot$. The marked entries constitute a circle with radius 2 and center $(3, 0)$.

Finally, define **thresholds** *for an operator* $\circ$ *and a truth value* $s \in N$ *as:*

$$t_{\circ,s} = \min\{x \mid v(x \circ x) \geq s\} \quad (1)$$
$$\bar{t}_{\circ,s} = \max\{x \mid v(x \circ x) \leq s\} \quad (2)$$

Permuting all combinations of parameters in the two-valued case determines exactly the eight classical primary connectives as the non-trivial regular operators associated with each combination. In the multiple-valued case a regular operator is completely determined by its parameters and the values on the diagonal through its starting point.

Our goal is a tableau proof system with a sufficient number of rules for completely handling queries of the form $D\ A$ and $\bar{D}\ A$. From Figures 2,3 it is obvious that we need only consider signs of the form $S_{\{0,\ldots,s_1-1\}}$ and $S_{\{s_2+1,\ldots,n-1\}}$, where $s_1 = t_{\circ,d}$ and $s_2 = \bar{t}_{\circ,d}$. As a shorthand for the corresponding signs we write[12]

$$E^s = S_{\{0,\ldots,s-1\}} \text{ for } s \in N$$
$${}^sE = S_{\{s+1,\ldots,n-1\}} \text{ for } s \in N$$

Provided, in the premisses of our rules occur only signs of the form E^s and ${}^sE$, we have also in the conclusion only signs of this form, for if $\delta(\circ) =< I, I >, \epsilon(\circ) = I$, obviously the following holds:

$$x \circ y \in E^s \quad iff \quad x \in E^{t_{\circ,s}} \text{ and } y \in E^{t_{\circ,s}}$$
$$x \circ y \in {}^sE \quad iff \quad x \in {}^{\bar{t}_{\circ,s}}E \text{ or } y \in {}^{\bar{t}_{\circ,s}}E$$

[12] If $s = 0$ resp. $s = n - 1$ we define these sets to be the empty set.

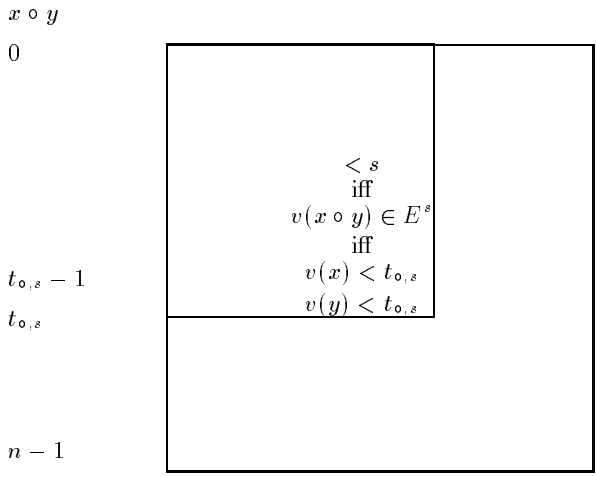


Figure 3: Determining the tableau rules for $E^s\ x \circ y$

Example 3.1 *Let $\odot$ be defined as in Table 4. Then*
$\delta_1(\odot) =^*, \delta_2(\odot) = I, \epsilon(\odot) =^*.$
Consider $\vee$ from Example 1.1:
$t_{\vee,2} = 2, \bar{t}_{\vee,2} = 2.$

Theorem 3.1 *Let $\mathcal{L}$ be any logic containing only regular connectives $\mathcal{F}$, such that for each $\circ \in \mathcal{F}$ $\delta_1 = \delta_2 = \epsilon = I$. Fix $S_{\mathcal{L}} = \{E^s \mid s \in N\} \cup \{{}^sE \mid s \in N\}$. Then the tableau system given by the following α and β component rules is sound and complete.*

α	α_1	α_2
$E^s\ x \circ y$	$E^{t_{\circ,s}}\ x$	$E^{t_{\circ,s}}\ y$

β	β_1	β_2
${}^sE\ x \circ y$	${}^{\bar{t}_{\circ,s}}E\ x$	${}^{\bar{t}_{\circ,s}}E\ y$

Proof: Apply Theorem 1.1. ■

Before we can treat the other cases, when one or more of the parameters is equal to *, we have to state some basic properties of conjugation.

Lemma 3.1

1. $(E^s)^* = {}^{s^*}E$ *for all* $s \in N$
2. ${}^sE^* = E^{(s^*)}$ *for all* $s \in N$
3. $(i^*)^* = i$ *for all* $i \in N$
4. $(X^*)^* = X$ *for all* $X \subseteq N$

Proof: The proof is straightforward and is omitted. ■

To change the rules for proper handling of other values than I for δ_i, observe that $\delta_1 =^*$ indicates that the truth table of f is flipped around its horizontal axis. To compute the threshold function properly we must count from upside down in the first argument of f or, in other words, we must conjugate the first argument before

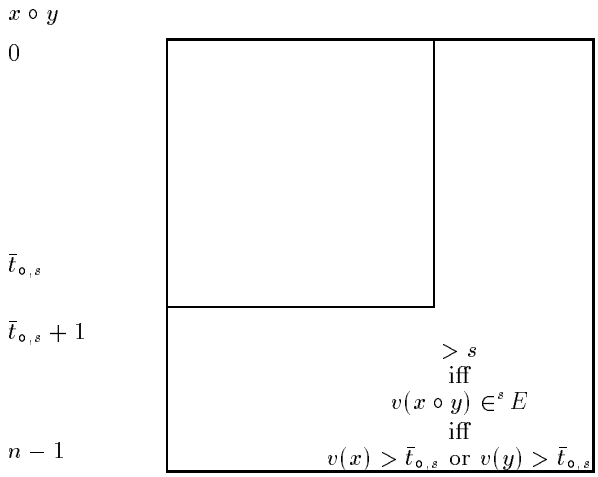


Figure 4: Determining the tableau rules for ${}^sE\ x \circ y$

computing the threshold, however, the result has to be conjugated again. The same considerations also apply, of course, to the second argument of f, thereby arriving at the following definitions for the threshold functions and component rules:

$$t_{\circ,s} = \min\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \geq s\} \quad (3)$$

$$\bar{t}_{\circ,s} = \max\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \leq s\} \quad (4)$$

α	α_1	α_2
$E^s\ x \circ y$	$(E^{t_{\circ,s}})^{\delta_1(\circ)}\ x$	$(E^{t_{\circ,s}})^{\delta_2(\circ)}\ y$

β	β_1	β_2
${}^sE\ x \circ y$	$({}^{\bar{t}_{\circ,s}}E)^{\delta_1(\circ)}\ x$	$({}^{\bar{t}_{\circ,s}}E)^{\delta_2(\circ)}\ y$

Example 3.2 *Consider the three-valued implication as defined in Section 1. Assume we wanted to compute the tableau rule for the sign $S_{\{1,2\}}$. First $S_{\{1,2\}} = {}^0E$, so our component rules tell us that we have a β type rule with extensions $({}^{\bar{t}_{\supset,0}}E)^{\delta_1(\supset)}\ x$ and $({}^{\bar{t}_{\supset,0}}E)^{\delta_1(\supset)}\ y$. Now, since $\delta(\supset) = <^*, I>$ we have $\bar{t}_{\supset,0} = \max\{x \mid v(x^* \circ x) \leq 0\} = 0$, which yields extensions $({}^0E)^*\ x$ and ${}^0E\ y$. $({}^0E)^* = E^{(0^*)} = E^2 = S_{\{0,1\}}$ and ${}^0E = S_{\{1,2\}}$ finally yield the rule*

$$\frac{S_{\{1,2\}}\ (x \supset y)}{S_{\{0,1\}}\ x \mid S_{\{1,2\}}\ y}$$

We can treat negation as a special case of the above stated rules. If we note that $\alpha_1(\neg) = \alpha_2(\neg)$, $\beta_1(\neg) = \beta_2(\neg)$, $\delta_i(\neg) = {}^*$ and $t_{\neg,s} = \bar{t}_{\neg,s} = s$ for all s, treat extensions as sets and omit identical extensions, we arrive directly at the negation rules stated in Table 5. $\sim$ and ∇ are treated similarly.

We still have to treat the case when $\epsilon(\circ) = {}^*$. Taking into account this possiblity in the component rules and the threshold functions leads to the following definitions:

$$\frac{E^s\ \neg x}{{}^{s^*}E\ x} \qquad \frac{{}^sE\ \neg x}{E^{(s^*)}\ x}$$

Table 5: Negation Rules

$$t_{\circ,s} = \min\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \geq^{\epsilon(\circ)} s\} \quad (5)$$

$$\bar{t}_{\circ,s} = \max\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \leq^{\epsilon(\circ)} s\} \quad (6)$$

where $\leq^*=\geq$ and $\geq^*=\leq$

Example 3.3 *Let $\odot$ be defined as in Table 4.*

$t_{\odot,2} = 2, \bar{t}_{\odot,2} = 2$.

Theorem 3.2 *Let $\mathcal{L}$ be like in Theorem 3.1, but with no restrictions on δ_i, ϵ. Then the tableau system given by the following α and β component rules is sound and complete.*

α	α_1	α_2
$(E^{s^*})^{\epsilon(\circ)} x \circ y$	$(E^{t_{\circ^{\epsilon(\circ)}, s^*}})^{\delta_1(\circ)} x$	$(E^{t_{\circ^{\epsilon(\circ)}, s^*}})^{\delta_2(\circ)} y$

β	β_1	β_2
$({}^{s^*}E)^{\epsilon(\circ)} x \circ y$	$({}^{\bar{t}_{\circ^{\epsilon(\circ)}, s^*}}E)^{\delta_1(\circ)} x$	$({}^{\bar{t}_{\circ^{\epsilon(\circ)}, s^*}}E)^{\delta_2(\circ)} y$

Proof: We proof our claim by reducing it to the case when $\epsilon(\circ) = I$.

If $\epsilon(\circ) = I$, substitute s_0^* for s and apply Lemma 3.1.

If $\epsilon(\circ) = {}^*$, let $\circ^*$ be the **corresponding conjugated operator** for $\circ$, which is defined as

$$x \circ^* y = \neg(\neg x \circ \neg y)$$

.

The following facts are easy to see:

$$x \circ y = \neg(\neg x \circ^* \neg y) \quad (7)$$

$$\delta_i(\circ^*) = \delta_i^*(\circ) \quad (8)$$

$$\epsilon(\circ^*) = \epsilon^*(\circ) \quad (9)$$

From (9) we have $\epsilon(\circ^*) = I$.

So we can conclude from the premise of the β component rule, using Lemma 3.1, (7) and the negation rules:

$$\begin{aligned} ({}^{s^*}E)^{\epsilon(\circ)}\ x \circ y &= ({}^{s^*}E)^*\ \neg(\neg x \circ^* \neg y) \\ &= E^s\ \neg(\neg x \circ^* \neg y) \\ &= {}^{s^*}E\ \neg x \circ^* \neg y \end{aligned}$$

While $\epsilon(\circ^*) = I$ at this point we can use the rules for that case, yielding the extensions

$$({}^{\bar{t}_{\circ^*, s^*}}E)^{\delta_1(\circ^*)}\ x^*$$

$$({}^{\bar{t}_{\circ^*, s^*}}E)^{\delta_2(\circ^*)}\ y^*$$

From here we get with (8), (9), Lemma 3.1 and the negation rules the desired result. The α case is proved in exactly the same way. ■

Before giving another example we want to simplify the rules a bit further. To this end we need the following Lemma:

Lemma 3.2 *For all $s \in N$ and all regular operators $\circ$:*

1. $t_{\circ^*,s^*} = t_{\circ,s}$
2. $\bar{t}_{\circ^*,s^*} = \bar{t}_{\circ,s}$

Proof: We show only part 1.

By definition we have:

$$t_{\circ^*,s^*} = \min\{x \mid v(x^{\delta_1(\circ^*)} \circ^* x^{\delta_2(\circ^*)}) \geq^{\epsilon(\circ^*)} s^*\}$$

By definition of $\circ^*$, v, (8) and Lemma 3.1:

$$= \min\{x \mid v^*(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \geq^{\epsilon(\circ^*)} s^*\}$$

Because of $i^* \leq j^*$ iff $i \geq j$:

$$= \min\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \leq^{\epsilon(\circ^*)} s\}$$

And finally from (9) and definition of $\leq^*$:

$$= \min\{x \mid v(x^{\delta_1(\circ)} \circ x^{\delta_2(\circ)}) \geq^{\epsilon(\circ)} s\}$$
$$= t_{\circ,s}$$

■

Using Lemma 3.1 and 3.2 we can simplify the component rules for the case $\epsilon(\circ) = {}^*$ to:

α	α_1	α_2
${}^sE\ x \circ y$	$(E^{t_{\circ,s}})^{\delta_1(\circ)}\ x$	$(E^{t_{\circ,s}})^{\delta_2(\circ)}\ y$

β	β_1	β_2
$E^s\ x \circ y$	$(\bar{t}_{\circ,s}E)^{\delta_1(\circ)}\ x$	$(\bar{t}_{\circ,s}E)^{\delta_2(\circ)}\ y$

From this and the rules for the case $\epsilon(\circ) = I$ we can extract the final simplified component rule format for the general case:

α	α_1	α_2
$(E^{s^{\epsilon(\circ)}})^{\epsilon(\circ)}\ x \circ y$	$(E^{t_{\circ,s}})^{\delta_1(\circ)}\ x$	$(E^{t_{\circ,s}})^{\delta_2(\circ)}\ y$

β	β_1	β_2
$({}^{s^{\epsilon(\circ)}}E)^{\epsilon(\circ)}\ x \circ y$	$(\bar{t}_{\circ,s}E)^{\delta_1(\circ)}\ x$	$(\bar{t}_{\circ,s}E)^{\delta_2(\circ)}\ y$

Example 3.4 *We illustrate the rules with the operator $\odot$ defined in Table 4. Suppose we were interested in the tableau rule for sign $S_{\{3\}}$. From $S_{\{3\}} = {}^2E = (E^{2^*})^*$ and Examples 3.1, 3.3 we see that our generic α rule may be instantiated, yielding the extensions: $(E^2)^*\ ; x = {}^1E\ x = S_{\{2,3\}}\ x$ and $E^2\ y = S_{\{0,1\}}\ y$.*

4 Functional Completeness

One may argue that the class of regular connectives is too small and many multiple-valued logics fall outside of it. In this Section we show that for each n there is a functionally complete n-valued logic with only regular connectives. Thus nothing essential has been lost for one can always replace nonregular connectives by regular ones.

We start from a well-known family of multiple-valued logics that is known to be functionally complete[13], namely the logics $\mathcal{P}_n$ of Post. Each $\mathcal{P}_n$ contains exactly two connectives $\vee$ and σ with arities 2 and 1 resp. The semantics of $\vee$ is as above, while σ may be defined by

$$v(\sigma(X)) = (v(X) - 1) \bmod n$$

where $-$ and mod are interpreted as usual, treating truth values as natural numbers. Since $\vee$ is regular, all we have to show is that σ can be composed by regular connectives alone for each n.

Consider the unary connectives σ' and J_0 defined by

$$v(\sigma'(X)) = \max\{0, v(X) - 1\}$$

$$v(J_0(X)) = \begin{cases} n-1 & \text{if } X = 0 \\ 0 & \text{otherwise} \end{cases}$$

As can be seen from Table 6, both are regular, and obviously $\sigma(X) \equiv \sigma'(X) \vee J_0(X)$.

X	$\sigma(X)$	$\sigma'(X)$	$J_0(X)$
0	$n-1$	0	$n-1$
1	0	0	0
2	1	1	0
...	...	...	...
k	$k-1$	$k-1$	0
...	...	...	...
$n-1$	$n-2$	$n-2$	0

Table 6: Truth table of σ, σ' and J_0.

5 First-Order Multiple-Valued Logic

In this Section we sketch how the idea of using sets of truth values as signs may be extended to First-Order Logic. Moreover, it will turn out that our restriction to signs of the form E^s and ${}^sE$ is sufficient to preserve the form of γ and δ rule schemes entirely.

Let $L^1 = L^1(R, Par)$ be a **first-order language** defined in the usual way, where R is a nonempty, countable set of **predicate symbols** (each with an associated arity) and Par an infinite, countable set of **parameters**[14]. Additionally, we have a set of **individual variables** IV. **Formulas** are constructed from those in the usual manner, using a set of connectives $\mathcal{F}$ and the **quantifier symbols** $\forall$, $\exists$.

We assume familiarity with the notions of **substitution** (denoted with $\{X/p\}$), **free variable occurrence, sentence** from classical logic, which we use in the standard way. Denote the set of sentences from L^1 with $\bar{L}^1$.

Let L be the propositional sublanguage of L^1 consisting of its quantifier free sentences. Let L' be the set of sentences from L with no leading quantifier and let $\mathcal{L} =< \mathcal{A}, L >$ be a propositional logic based on L as defined in Section 1.

Now we are able to define the first-order extension of a propositional valuation $v : L' \to N$ to $\bar{v} : \bar{L}^1 \to N$ by

[13] For a proof see e.g. [Urquhart, 1986].

[14] Of course it is possible to include function symbols as well, but this requires considerably more work (and space) in the following definitions.

(V1) $\bar{v}(Y) = v(Y)$ if $X \in L'$

(V2) $\bar{v}((\forall x)Y) = \min\{\bar{v}(Y\{x/p\}) \mid p \in Par\}$

(V3) $\bar{v}((\exists x)Y) = \max\{\bar{v}(Y\{x/p\}) \mid p \in Par\}$

Analog to the propositional case it is sufficient to define v on L and then extend it uniquely to $\bar{L}$[1].

The semantical definitions of the quantifiers as given here, can be read as a generalized conjunction and disjunction resp. Another way to look at the quantifiers would be to ask which truth values are taken on by the quantified formula while the variable is running through the substitutions. If we denote this distribution of truth values by a subset of N, we obtain alternative characterizations of the quantifiers by giving a mapping from distributions to truth values. In the case of $n = 3$ e.g. $(\forall x)Y$ has truth value 1 for the distributions $\{1\}, \{1, 2\}$. This may also be stated as "$\bar{v}((\forall x)Y) = 1$ iff either $\bar{v}(\{Y/p\}) = 1$ for all $p \in Par$ or $\bar{v}(\{Y/p\}) = 1$ for some p and $\bar{v}(\{Y/p\}) = 2$ for some other p".

Rephrasing this observation in the language of tableau rules in [Carnielli, 1987] leads to rules like:

$$\frac{S_1\ (\forall x)Y}{\begin{array}{l|l} S_1\ Y\{x/p\} & S_1\ Y\{x/p_1\} \\ & S_2\ Y\{x/p_2\} \end{array}}$$

With the proviso that for p in the left extension an arbitrary $p \in Par$ may be substituted and in the right extension $p_1 \neq p_2$ must be new parameters on the tableau. Each extension corresponds to one possible distribution of truth values.

Obviously, we could adapt this approach to our system with generalized signs by simply taking into the conclusion all distributions that are associated with a truth value occurring in a sign. If we asked e.g. for the tableau rule for $S_{\{0,2\}}\ (\forall x)Y$ we would arrive at a rule that has as its conclusion all extensions from the rules for $S_0\ (\forall x)Y$ and for $S_2\ (\forall x)Y$. But then in the extensions only signs corresponding to singleton sets would occur and it is hard to see how one can take any advantage from the notion of generalized signs here.

The situation becomes quite different if we restrict as before the set of signs to be $S_{\mathcal{L}} = \{E^s \mid s \in N\} \cup \{{}^sE \mid s \in N\}$. In this case we arrive at the following, surprisingly simple uniform notation schemes for quantified formulas:

γ	$\gamma(p)$
${}^sE\ (\forall x)Y$	${}^sE\ Y\{x/p\}$
$E^s\ (\exists x)Y$	$E^s\ Y\{x/p\}$

δ	$\delta(p)$
$E^s\ (\forall x)Y$	$E^s\ Y\{x/p\}$
${}^sE\ (\exists x)Y$	${}^sE\ Y\{x/p\}$

The corresponding expansion rules in Table 7 are restricted by the usual provisos for γ and δ rules, namely that p is arbitrary from Par in the γ rules and it has to be new on the tableau in the δ rules. It can be proved that together with the propositional tableau rules from either Section 1–3 they result in a sound and complete system for the respective logic as long as the set of signs of the whole system can be restricted as above.

$$\frac{\gamma}{\gamma(p)} \qquad \frac{\delta}{\delta(p)}$$

Table 7: Tableau rules for γ and δ formulas

Due to space constraints, instead of a formal proof here we give only a partially informal justification of the quantifier rules.

Let us concentrate on the γ rule for $\forall$. The meaning of the premise is:

$$\begin{array}{l} \text{There is an } i \in \{s+1, \ldots, n-1\} \text{ such that} \\ \min\{\bar{v}(Y\{x/p\}) \mid p \in Par\} = i \end{array} \tag{10}$$

Obviously, this i must be unique, say it is i_0. Since i_0 is the minimum of all $\bar{v}(Y\{x/p\})$, p ranging over Par, all other values of $\bar{v}(Y\{x/p\})$ lie between i_0 and $n-1$ and we may conclude that

$$\begin{aligned} &\text{For each } p \in Par \text{ there is an } i_1 \in \{s+1, \ldots, n-1\} \\ &\text{such that } \bar{v}(Y\{x/p\}) = i_1 \end{aligned} \tag{11}$$

But this is exactly what the conclusion of the rule says. Note that it is essential for the argument to hold that the set of truth values under consideration has no gaps between $s+1$ and $n-1$. If we substituted an arbitrary S_j for ${}^sE$ we could not be sure whether the required i_1's are contained in it.

The other direction is easily seen to hold as well, in fact without imposing any restriction on the set of signs. Just let i_0 be the minimum ranging over all i_1's. Thus we have proved the equivalence of (10) and (11) which is sufficient to prove tableau soundness and completeness.

By using the first part of the equivalence one can easily proof the δ rule for $\forall$ also. Justification of the rules for $\exists$ is reduced to the $\forall$ case by the duality properties that carry over from classical logic.

Finally, we remark that fairly nice rules may be also obtained for signs of the form $S_{\{j\}}$, where $j \in N$, if it is desired.

Conclusion

We have presented a framework for axiomatizing arbitrary finitely valued logics with minimal overhead when compared to the classical case. The main idea was to work with tableaux using generalized signs, which enabled us to express complex assertions regarding the possible truth values of a formula.

We have introduced the class of regular logical connectives which, together with a suitable restriction on queries (i.e. allowed signs) to the system, allow a uniform notation style presentation of multiple-valued propositional and first-order logics. It has been demonstrated that various systems differing in their allowed classes of connectives and complexity of rules may be formulated.

This allows the use of tools and methods that are close to the ones used in classical logic, both on the theoretical (uniform notation in definitions and proofs) and practical side (use of classical theorem provers with few modifications).

References

[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.

[Emerson & Halpern, 1985] E. Allen Emerson & Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

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

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

[Hähnle, 1990] Reiner Hähnle. Towards an efficient tableau proof procedure for multiple–valued logics. In Wolfgang Schönfeld, editor, *Proceedings Workshop on Computer Science Logic, Heidelberg*, Springer LNCS, 1990.

[Rescher, 1969] Nicholas Rescher. *Many–Valued Logic.* McGraw–Hill, New York, 1969.

[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] Stanislaw 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.

[Urquhart, 1986] Alasdair Urquhart. Many–valued logic. In D. Gabbay & F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. III: Alternatives in Classical Logic*, chapter 2, pages 71–116. Reidel, Dordrecht, 1986.

[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.