The Inverse Method for Many-Valued Logics*

Laura Kovács[1], Andrei Mantsivoda[2], and Andrei Voronkov[3]

[1] Chalmers University of Technology
[2] Irkutsk State University
[3] The University of Manchester

Abstract. We define an automatic proof procedure for finitely many-valued logics given by truth tables. The proof procedure is based on the inverse method. To define this procedure, we introduce so-called *introduction-based sequent calculi*. By studying proof-theoretic properties of these calculi we derive efficient validity- and satisfiability-checking procedures based on the inverse method. We also show how to translate the validity problem for a formula to unsatisfiability checking of a set of propositional clauses.

1 Introduction

The inverse method of theorem proving was developed in the 1960s [12, 17–19], see [10] for an overview. The method is essentially orthogonal to the tableau method and is based on the bottom-up proof-search in sequent calculi. The inverse method is based on the idea of specialising a sequent calculus by a given goal, using the subformula property of sequent calculi. The inverse method was successfully applied to a number of non-classical logics, including intuitionistic logic and some modal logics [32, 34, 22].

The inverse method can be efficiently implemented for classical logic [31] and some non-classical logics [33]. In this paper we give a presentation of the inverse method for finite many-valued logics. We show how to apply the inverse method to obtain a proof procedure for many-valued logics. Nearly all known methods of automated reasoning have been extended to many-valued logic in [8, 9, 24, 25, 2, 27, 5, 14, 26, 15, 3], but to the best of our knowledge it is the first ever presentation of the inverse method.

This paper is organised as follows. In Section 2 we define many-valued logics, their syntax and semantics. Section 3 defines so-called introduction-based sequent calculi and proves results about their soundness and completeness. The main results of this section are that soundness and completeness are "local" properties, which can easily be checked. We also discuss minimal calculi (which result in more efficient proof procedures) and prove admissibility of the cut rule. In Section 4 we introduce a key definition of *signed subformula*. It is interesting that this definition depends not only on the semantics of a logic, but also on a sequent calculus we choose.

* We acknowledge funding from the Austrian FWF grant S11410-N23 and the Austrian WWTF grant ICT C-050.

Section 5 presents the main technique used by the inverse method: the specialisation of a sequent calculus to prove a particular goal, using the subformula property. In Section 6 we show that the inverse method can be efficiently implemented by translation to propositional satisfiability. Finally, in Section 7 we discuss some related work and future research directions. Detailed proofs of theorems are not included in this paper but can be found in its longer version [4].

2 Many-Valued Logics

This section contains all basic definitions. Many of them are quite standard and included mostly for the sake of unambiguity. Results of this section are not original. We only consider many-valued logics with a finite set of truth values.

DEFINITION 1. A *finite many-valued logic* $\mathcal{L}$ is a triple $(V, Con, arity, val)$ where

1. V is a finite non-empty set of *truth values*;
2. Con is a finite non-empty set of *connectives*;
3. $arity$ is a mapping from Con to the set of non-negative integers, called the *arity mapping*;
4. val is a function from Con to the set of functions on V, called the *valuation function*, such that the arity of $val(c)$ is $arity(c)$, for all $c \in Con$.

If $arity(c) = m$, then we call c an *m-ary connective*. The logic $\mathcal{L}$ is called a *k-valued logic* if k is the number of elements in V.

For the rest of this paper we assume an arbitrary, but fixed k-valued logic $\mathcal{L} = (V, Con, arity, val)$. We also assume to have a countably infinite set $Atom$ of *propositional variables*.

DEFINITION 2. The set of *formulas* of logic $\mathcal{L}$ is defined as follows:

1. Every propositional variable $A \in Atom$ is a formula;
2. If $c \in Con_m$ and $F_1, \ldots, F_m$ are formulas, then $c(F_1, \ldots, F_m)$ is a formula.

In many-valued logics we prove that a formula always has a given truth value or a given set of truth values. We will introduce a notation for expressing that a formula F has a truth value t.

DEFINITION 3. A *signed formula* is a pair (F, t), denoted F^t, where F is a formula and $t \in V$. A *sequent* is a finite set $\{F_1^{t_1}, \ldots, F_m^{t_m}\}$ of signed formulas. For simplicity, we will write such a sequent as a sequence $F_1^{t_1}, \ldots, F_m^{t_m}$.

In [14, 26] signed formulas are signed with *sets of* truth values. For simplicity, we only consider single truth values as signs. However, we can generalize our results to sets of signs as well.

[4] available upon request, due to authors' anonymity

The intuitive meaning of the signed formula F^t is that the formula F has the truth value t. Sequents are needed to define the semantics and the proof theory of many-valued logics. The intuitive meaning of a sequent $F_1^{t_1}, \ldots, F_m^{t_m}$ is that at least one of the F_i's has the truth value t_i.

For sequents S_1, S_2, we will write $S_1 \cup S_2$ as simply S_1, S_2. Likewise, for a sequent S and formula F^t, we will write $S \cup \{F^t\}$ simply as S, F^t, and similarly for multiple sequents and formulas.

The semantics of many-valued logics is defined via the notion of an truth assignment.

DEFINITION 4. A *truth assignment* is any mapping $\alpha : Atom \to V$. The truth value $\alpha(A)$ is the *value* of the variable A under the truth assignment α.

Truth assignments can be extended to arbitrary formulas F of logic $\mathcal{L}$ in the following way. For any $c \in Con$ of arity m, let

$$\alpha(c(F_1, \ldots, F_m)) \stackrel{\text{def}}{=} val(c)(\alpha(F_1), \ldots, \alpha(F_m))$$

We shall also extend evaluations to signed formulas and sequents. Each signed formula and sequent will be either true or false. More precisely, we extend the mapping α to a mapping from signed formulas and sequents to the set of boolean values $\{true, false\}$ as follows:

$$\alpha(F^t) \stackrel{\text{def}}{=} \begin{cases} true, \text{ if } \alpha(F) = t \\ false, \text{ otherwise} \end{cases}$$

$$\alpha(F_1^{t_1}, \ldots, F_n^{t_n}) \stackrel{\text{def}}{=} \begin{cases} true, \text{ if } \alpha(F_i^{t_i}) = true \text{ for some } i = 1, \ldots, n \\ false, \text{ otherwise} \end{cases}$$

DEFINITION 5. A sequent S is called *valid* in a logic $\mathcal{L}$ if for all truth assignments α we have $\alpha(S) = true$.

The semantics of connectives in finitely-valued logics can conveniently be represented via *truth tables*. A truth table for a connective c is a table whose rows are all tuples of the graph of $val(c)$. For example, in a two-valued logic with truth values $\{t, f\}$ the ternary connective c such that $val(c)(t_1, t_2, t_3) = t$ iff $t_1 = t_2 = t$ or $(t_1 = t_3 = t$ and $t_2 = f)$ can be represented by the following truth table:

A_1	A_2	A_3	$c(A_1, A_2, A_3)$
t	t	t	t
t	t	f	t
t	f	t	t
t	f	f	f
f	t	t	f
f	t	f	f
f	f	t	f
f	f	f	f

3 Sequent calculi

The main goal of proof systems for many-valued logics is to prove validity of sequents. There is a technique allowing a sound and complete sequent calculus to be constructed from a given many-valued logic [30, 4]. In this section we define *introduction-based sequent calculi.* A naive way of constructing a sound and complete sequent calculus creates however many redundancies. We try to avoid this by using a *minimal* sequent calculi.

When presenting sequent calculi, we will use the terminology of [6, 10]. Namely, an *inference* has the form

$$\frac{\{S_1, \cdots, S_n\}}{S} \ ,$$

where $n \geq 0$, an *inference rule* is a collection of inferences and a *calculus* is a collection of inference rules. In the inference above, the sequent S is called the *conclusion*, and $S_1, \ldots, S_n$ the *premises* of this inference. For simplicity, we will write such inferences as

$$\frac{S_1 \quad \cdots \quad S_n}{S} \ .$$

As in [6, 10], a *derivation* in a calculus is any tree built from inferences in this calculus. A sequent is *derivable* if it has a derivation.

Definition 6. A calculus G of sequents is called *introduction-based* if

1. G contains the inferences rule

$$\frac{}{A^{t_1}, \ldots, A^{t_k}} \tag{1}$$

   where A is an arbitrary propositional variable and $t_1, \ldots, t_k$ are all truth values of the logic.
2. G contains some inference rules of the form

$$\frac{\{S_j, F_j^{t_j} \mid j \in J\}}{\bigcup_{j \in J} S_j, c(F_1, \ldots, F_n)^t} \tag{2}$$

   where J is a fixed subset of $\{1, \ldots, n\}$, the F_j's are arbitrary formulas and the t_j's and t are fixed truth values. Such an inference rule will be called an *introduction rule* for c.
3. G contains no other rules.

Note that in this definition the set of indices J may form a proper subset of $\{1, \ldots, n\}$ and the truth values t_j's should not necessarily be pairwise different. An introduction rule represents an infinite set of inferences, obtained by varying formulas $F_1, \ldots, F_n$.

To illustrate this definition and any other material in this paper we will use the standard two-valued logic $\mathcal{L}_2$ with truth values *true* and *false*, the standard

connectives like the conjunction and a ternary connective *ite* (if-then-else). The valuation function of $\mathcal{L}_2$ for all connectives is considered as standard.

The following are familiar introduction rules for the conjunction $\wedge$:

$$\frac{S, F_1^{false}}{S, (F_1 \wedge F_2)^{false}} \qquad \frac{S, F_2^{false}}{S, (F_1 \wedge F_2)^{false}} \qquad \frac{S_1, F_1^{true} \quad S_2, F_2^{true}}{S_1, S_2, (F_1 \wedge F_2)^{true}}$$

The following is an introduction rule for *ite*:

$$\frac{S_1, F_1^{true} \quad S_2, F_2^{true}}{S_1, S_2, ite(F_1, F_2, F_3)^{true}}$$

In the sequel a calculus means an introduction-based calculus, unless otherwise is clear from the context.

Definition 7. An introduction rule (2) is called *sound* if for every inference of this rule, whenever all premises of the rule are true, the conclusion is also true. A calculus is called *locally sound* if every rule of this calculus is sound. A calculus is called *sound* if every derivable sequent is true.

Theorem 8. A calculus is sound if and only if it is locally sound. Every locally sound calculus is sound, i.e. every sequent provable in a calculus locally sound for $\mathcal{L}$, is valid in $\mathcal{L}$.

Definition 9. A calculus G is called *subset-complete* if for every valid sequent S there exists a derivable sequent S' such that $S' \subseteq S$. A calculus G is called *locally complete* if for every n-ary connective c and truth values $t_1, \ldots, t_n, t$ such that $t = val(c)(t_1, \ldots, t_n)$, there exist a set $J \subseteq \{1, \ldots, n\}$ such that rule (2) belongs to G.

Note the non-standard formulation of completeness caused by the absence of the weakening rule in the calculus. Consider, for example, the logic $\mathcal{L}_2$. The sequent $A^{true}, A^{false}, B^{true}$, where A and B are propositional variables, is valid, but not derivable in *any* introduction-based calculus, since it does not have the form (1) and contains no connectives. However, its proper subset A^{true}, A^{false} is derivable in *every* such calculus, since it is an instance of (1). It is not hard to argue that the following theorem holds.

Theorem 10. A calculus G is subset-complete if and only if it is locally complete.

From Theorems 8 and 10 we obtain the following result.

Theorem 11. An introduction-based calculus is sound and subset-complete if and only if it is both locally sound and locally complete.

Let us introduce the *weakening rule*, or simply *weakening*:

$$\frac{S}{S, F^t} \ .$$

Obviously, weakening preserves validity. We call an *introduction-based calculus with weakening*, or simply *calculus with weakening*, any calculus obtained from an introduction-based calculus by adding the weakening rule.

DEFINITION 12. A calculus is called *complete* if every valid sequent is derivable in this calculus.

THEOREM 13. A calculus with weakening is sound and complete if and only if it is both locally sound and locally complete.

DEFINITION 14. Let $\{t_1, \ldots, t_n\}$ be all truth values of a logic $\mathcal{L}$. The *cut rule* for this logic is defined as follows

$$\frac{S_1, F^{t_1} \qquad \cdots \qquad S_n, F^{t_n}}{S_1, \ldots, S_n}$$

THEOREM 15. The cut rule is admissible in any calculus with weakening which is both locally sound and locally complete, that is, every sequent derivable in this calculus with the use of the cut rule, is also derivable without the cut rule.

Proof. It is enough to note that the cut rule is sound and apply Theorem 13.

It is not hard to find a locally sound and locally complete calculus. Consider the calculus defined by all inference rules of the form

$$\frac{S_1, F_1^{t_1} \quad \cdots \quad S_n, F_n^{t_n}}{S_1, \ldots, S_n, c(F_1, \ldots, F_n)^t}$$

where $val(c)(t_1, \ldots, t_n) = t$. It is straightforward to show that this calculus is both locally sound and locally complete.

Let us introduce a partial order $\leq$ on inference rules of introduction-based calculi as the smallest order such that

$$\frac{S, F_r^{t_r} \qquad \cdots \qquad S, F_s^{t_s}}{S, c(F_1, \ldots, F_n)^t} \quad \leq \quad \frac{S, F_p^{t_p} \qquad \ldots \qquad S, F_q^{t_q}}{S, c(F_1, \ldots, F_n)^t}$$

whenever $\{r, \ldots, s\} \subseteq \{p, \ldots, q\}$.

EXAMPLE 16. Consider the following three rules for the classical disjunction:

$$\frac{S, F_2^{true}}{S, (F_1 \vee F_2)^{true}} \tag{3}$$

$$\frac{S, F_1^{false} \qquad S, F_2^{true}}{S, (F_1 \vee F_2)^{true}} \tag{4}$$

$$\frac{S_1, F_1^{true} \qquad S_2, F_2^{true}}{S_1, S_2, (F_1 \vee F_2)^{true}} \tag{5}$$

Then (3) $\leq$ (4) and (3) $\leq$ (5), but rules (4) and (5) are incompatible w.r.t. $\leq$.

The relation $\leq$ can be generalised to a partial order $\preceq$ on introduction-based calculi by comparing their sets of rules using the finite multiset extension of $\leq$, see [11] for a precise definition. We are interested in calculi minimal w.r.t. $\preceq$ since smaller calculi have a more compact presentation of inference rules. Consider, for example, introduction rules for *ite* in $\mathcal{L}_2$. It is not hard to argue that the following are all such inference rules minimal w.r.t. $\leq$:

$$\frac{S_1, F_1^{true} \quad S_2, F_2^{true}}{S_1, S_2, ite(F_1, F_2, F_3)^{true}} \qquad \frac{S_1, F_1^{true} \quad S_2, F_2^{false}}{S_1, S_2, ite(F_1, F_2, F_3)^{false}}$$

$$\frac{S_1, F_1^{false} \quad S_3, F_3^{true}}{S_1, S_3, ite(F_1, F_2, F_3)^{true}} \qquad \frac{S_1, F_1^{false} \quad S_3, F_3^{false}}{S_1, S_3, ite(F_1, F_2, F_3)^{false}}$$

$$\frac{S_2, F_2^{true} \quad S_3, F_3^{true}}{S_2, S_3, ite(F_1, F_2, F_3)^{true}} \qquad \frac{S_2, F_2^{false} \quad S_3, F_3^{false}}{S_2, S_3, ite(F_1, F_2, F_3)^{false}}$$

It has the least locally sound and locally complete calculus consisting of the first four of the above rules. This example shows that finding minimal calculi may be non-trivial, since the last two inference rules are minimal, but not occur in the least calculus.

One can also show that, in general, the least calculus may not exist.

4 The Subformula Property

The theorem proving problem for a many-valued logic consist of determining the validity of a sequent. If the formula is valid, one may also want to have a derivation of this formula in some calculus. If it is not valid, one generally requires to find a truth assignment that makes the formula false. To design a proof-search algorithm based on the inverse method, we will use a subformula property. Interestingly, our definition of (signed) subformula will not be given in terms of the logic itself: it will be based on a locally sound and locally complete calculus. In the sequel we assume that such a calculus G is fixed.

Definition 17. Let G contain an inference rule (2). Then we say that each signed formula $F_j^{t_j}$ is the *immediate signed subformula* of $c(F_1, \ldots, F_n)^t$. The notion of a *signed subformula* is the reflexive and transitive closure of the notion of immediate signed subformula. A *signed subformula of a sequent* S is any signed subformula of a signed formula in S.

Suppose that for two calculi G_1 and G_2 we have $G_1 \preceq G_2$. It is interesting that, if a signed formula s_1 is a signed subformula of a signed formula s_2 with respect to G_1, then s_1 is also a signed subformula of s_2 with respect to G_2. This means that smaller calculi give smaller "signed subformula" relations.

Our main interest in the subformula property can be explained by the following theorem.

Axioms Axioms of G_γ are all sequents of the form

$$A^1, \ldots, A^k$$

such that each A^i is a signed subformula of a signed formula in γ.

Introduction rules These are all rules of G_γ are all rules of the form (2) restricted to signed subformulas of γ.

Fig. 1. The sequent calculus G_γ

THEOREM 18. Let Π be a derivation of a sequent S. Then all signed formulas occurring in Π are signed subformulas of signed formulas in S.

Proof. Straightforward by induction on the depth of Π.

5 The Inverse Method

The inverse method tries to construct derivations from axioms to the goal, using the subformula property. Paper [10] contains a detailed explanation on how one can design the inverse proof search based on sequent calculi having a suitable subformula property. This can be done in two steps: first, given a sequent γ to be proved, build a specialised sequent calculus intended to only prove γ and second, organise proof-search in this specialised calculus. For some modal and other non-classical logics the second step can be highly non-trivial, as, e.g., in [34]. In this paper we show that, for many-valued logics, the second step can be delegated to a propositional SAT solver.

For the rest of this section we assume that γ is a fixed sequent whose validity is to be established. Also, for simplicity of notation, we assume that the set of truth values is the set of integers $\{1, \ldots, k\}$. The specialised calculus G_γ is shown on Figure 1.

It is not hard to argue that G_γ has the following properties:

PROPOSITION 19. The following statements are true about G_γ:

1. Every sequent derivable in G_γ consists of signed subformulas of γ.
2. The set of sequents used in G_γ is finite.

Further, we also have the following property.

THEOREM 20 (SOUNDNESS AND COMPLETENESS OF G_γ). The sequent γ is valid if and only if it is provable in G_γ.

This theorem serves as a foundation of a proof procedure for many-valued logics. In the next section we show how one can use SAT solvers to search for derivations.

6 Inverse Method and Resolution

In this section we generalize to many-valued logics a well-known translation of sequent derivations into hyperresolution derivations. For classical logic it was described in different forms in [18, 16, 23, 10]. In this section we will use the standard terminology in propositional resolution, including (positive and negative) literals and clauses. We will consider a clause both as a set and a disjunction of literals.

Consider the set S_γ of all signed subformulas of γ. We will now treat each signed formula in S_γ as a propositional variable. These propositional variables should not be confused with the propositional variables in $Atom$.

Consider any introduction rule R in G_γ:

$$\frac{\{S_j, F_j^{t_j} \mid j \in J\}}{\bigcup_{j\in J} S_j, c(F_1, \ldots, F_n)^t}\ . \tag{6}$$

Denote by C_R the following propositional clause using variables in S_γ:

$$\bigvee_{j\in J} \neg F_j^{t_j} \vee c(F_1, \ldots, F_n)^t.$$

We recall that positive (propositional) hyperresolution is the following inference rule on propositional clauses:

$$\frac{C_1 \vee p_1 \quad \cdots \quad C_n \vee p_n \quad \neg p_1 \vee \ldots \vee \neg p_n \vee C}{C_1 \vee \ldots \vee C_n \vee C}\ , \tag{7}$$

where $p_1, \ldots, p_n$ are propositional variables and $C_1, \ldots, C_n, C$ are clauses consisting only of positive literals. The conclusion of this rule is called a *hyperresolvent of the clauses* $C_1 \vee p_1$, ... , $C_n \vee p_n$ *against the clause* $\neg p_1 \vee \ldots \vee \neg p_n \vee C$.

The following proposition is straightforward:

Proposition 21. For every instance of (6), the conclusion of this inference is a hyperresolvent of the premises against $C(R)$. Vice versa, every inference of the form (7) against $C(R)$, where all clauses $C_i \vee p_i$ consist of variables in S_γ, is an instance of R.

Suppose that $\gamma = s_1, \ldots, s_m$. Define the set $C(\gamma)$ of clauses consisting of clauses of three kinds:

1. All clauses $A^1 \vee \ldots \vee A^k$, such that $A^1, \ldots, A^k$ are signed subformulas of γ;
2. All clauses $C(R)$, such that R is a rule of G_γ;
3. The (unary) clauses $\neg s_1$, ... , $\neg s_m$.

Using Proposition 21 one can prove the following theorem:

Theorem 1. *The sequent γ is valid if and only if the set $C(\gamma)$ is unsatisfiable.*

One can use this theorem not only for establishing validity, but also for finding sequent derivations of (subsets of) γ. To this end, one should find an arbitrary resolution refutation of $C(\gamma)$ and transform it into a hyperresolution refutation. The details are included in a longer version of this paper.

Likewise, for any sequent that is not valid one can find a countermodel, as follows. Take any model M of $C(\gamma)$, for example, found by a SAT solver. We have to find a truth assignment α that makes γ false. Suppose that $A \in Atom$. If all $A^1, \ldots, A^k$ are signed subformulas of γ, then $A^1 \vee \ldots \vee A^k$ is a clause in $C(\gamma)$, hence for at least one j we have $M \models A^j$. In this case we pick any such truth value j and define $\alpha(A) \stackrel{\text{def}}{=} j$. If some of $A^1, \ldots, A^k$ is not a signed subformulas of γ, we pick such A_j and define $\alpha(A) \stackrel{\text{def}}{=} j$.

Note that there is an interesting variation of our translation, where we add clauses $\neg A^i \vee \neg A^j$ for all $i \neq j$. It requires experiments on large and hard formulas to understand whether it improves validity checking. On the one hand, we obtain a large set of clauses. On the other hand, we obtain many binary clauses, which may result in better performance because of improved unit propagation.

7 Conclusion and Related Work

By proof-theoretic investigation of sequent calculi for many-valued logics, we show how to design an inverse method calculus for a given sequent γ. Further, we show that validity checking for γ can be done by a SAT solver by building a set of propositional clauses $C(\gamma)$ such that $C(\gamma)$ is unsatisfiable if and only if γ is valid.

Theorem proving for many-valued logics is an area that attracted attention of many researchers. Some papers were already cited above, let us now mention papers most relevant to this paper. Papers [2, 3] build a resolution calculus for many-valued logics. Our calculus uses the inverse method and our hyperresolution simulation is different since the result of our translation is the set of ordinary propositional clauses. There are several papers providing translation from many-valued clausal logic into SAT, including [1, 7]. On the contrary, we focus on non-clausal formulas. Paper [15] discusses translation of formulas in many-valued logics to short normal forms, however these normal forms are many-valued too. Contrary to all these papers, we design an optimised translation by studying proof-theoretic properties of sequent calculi. For example, our notion of signed subformula is proof-theoretic and in general gives smaller sets of signed subformulas than in other approaches. The proof-theoretic investigation in this paper uses many ideas from [10]. However, [10] does not discuss many-valued logics and the translation to propositional resolution.

In future our results may benefit from results and ideas in the above mentioned papers. For example, using sets of signs instead of single signs is an interesting avenue to exploit. We are also interested in handling quantifiers.

References

1. Carlos Ansótegui, Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Mapping CSP into Many-Valued SAT. In João Marques-Silva and Karem A. Sakallah, editors, *SAT 2007*, volume 4501 of *Lecture Notes in Computer Science*, pages 10–15. Springer, 2007.
2. M. Baaz and C.G. Fermüller. Resolution for many-valued logics. In A. Voronkov, editor, *Logic Programming and Automated Reasoning. International Conference LPAR'92.*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 107–118, St.Petersburg, Russia, July 1992.
3. M. Baaz and C.G. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computations*, 19:353–391, 1995.
4. M. Baaz, C.G. Fermüller, A. Ovrutcki, and R. Zach. MULTLOG: a system for axiomatizing many-valued logics. In A. Voronkov, editor, *Logic Programming and Automated Reasoning. International Conference LPAR'93.*, volume 698 of *Lecture Notes in Artificial Intelligence*, pages 345–347, St.Petersburg, Russia, July 1993.
5. M. Baaz, C.G. Fermüller, and R. Zach. Systematic construction of natural deduction systems for many-valued logics. In *Proc. 23rd International Symposium on Multiple-Valued Logics*, pages 208–213, Los Gatos, CA, 1993. IEEE Computer Society Press.
6. L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.
7. Bernhard Beckert, Reiner Hähnle, and Felip Manyà. Transformations between signed and classical clause logic. In *ISMVL*, pages 248–255, 1999.
8. W.A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *Journal of Symbolic Logic*, 52(2):473–493, 1987.
9. W.A. Carnielli. On sequents and tableaux for many-valued logics. *Journal of Non-Classical Logics*, 8(1):59–76, 1991.
10. A. Degtyarev and A. Voronkov. The inverse method. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 4, pages 179–272. Elsevier Science, 2001.
11. N. Dershowitz and D.A. Plaisted. Rewriting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 9, pages 535–610. Elsevier Science, 2001.
12. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematical Zeitschrift*, 39:176–210, 405–431, 1934. Translated as [13].
13. G. Gentzen. Investigations into logical deduction. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North Holland, Amsterdam, 1969. Reprinted from [12].
14. R. Hähnle. *Automated Deduction in Multiple-Valued Logics*. Clarendon Press, Oxford, 1993.
15. R. Hähnle. Short conjunctive normal forms in finitely valued logics. *Journal of Logic and Computation*, 4(6):905–927, 1994.
16. V. Lifschitz. What is the inverse method? *Journal of Automated Reasoning*, 5(1):1–23, 1989.
17. S.Yu. Maslov. An inverse method for establishing deducibility of nonprenex formulas of the predicate calculus. *Soviet Mathematical Doklady*, 172(1):22–25, 1967. Reprinted as [20].

18. S.Yu. Maslov. Relationship between tactics of the inverse method and the resolution method (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 16, 1969. Reprinted as [21].
19. S.Yu. Maslov. Proof-search strategies for methods of the resolution type. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 77–90. American Elsevier, 1971.
20. S.Yu. Maslov. An inverse method for establishing deducibility of nonprenex formulas of the predicate calculus. In J.Siekmann and G.Wrightson, editors, *Automation of Reasoning (Classical papers on Computational Logic)*, volume 2, pages 48–54. Springer Verlag, 1983. Reprinted from [17].
21. S.Yu. Maslov. Relationship between tactics of the inverse method and the resolution method. In J.Siekmann and G.Wrightson, editors, *Automation of Reasoning (Classical papers on Computational Logic)*, volume 2, pages 264–272. Springer Verlag, 1983. Reprinted from [18].
22. Sean McLaughlin and Frank Pfenning. Efficient intuitionistic theorem proving with the polarized inverse method. In Renate A. Schmidt, editor, *CADE-22*, volume 5663 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2009.
23. G. Mints. Gentzen-type systems and resolution rules. Part I. Propositional logic. In P. Martin-Löf and G. Mints, editors, *COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer Verlag, 1990.
24. N.V. Murray and E. Rosenthal. Improving tableau deductions in multiple-valued logics. In *Proceedings of the 21st International Symposium on Multiple-Valued logics*, pages 230–237, Los Alamitos, 1991. IEEE Computer Society Press.
25. N.V. Murray and E. Rosenthal. Resolution and path dissolution in multiple-valued logics. In *Proceedings International Symposium on Methodologies for Intelligent Systems*, Charlotte, 1991.
26. N.V. Murray and E. Rosenthal. Signed formulas: a liftable meta-logic for multiple-valued logics. In *Proceedings ISMIS'93*, Lecture Notes in Computer Science, pages 230–237, Trondheim, Norway, 1993. Springer Verlag.
27. P. O'Hearn and Z. Stachniak. A resolution framework for finitely-valued first-order logics. *Journal of Symbolic Computations*, 13:235–254, 1992.
28. J.A. Robinson. Automatic deduction with hyper-resolution. *International Journal of Computer Mathematics*, 1:227–234, 1965. Reprinted as [29].
29. J.A. Robinson. Automatic deduction with hyperresolution. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 416–423. Springer, 1983. Originally appeared as [28].
30. G. Rousseau. Sequents in many-valued logic 1. *Fundamenta Mathematikae*, LX:23–33, 1967.
31. A. Voronkov. LISS — the logic inference search system. In M. Stickel, editor, *Proc. 10th Int. Conf. on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 677–678, Kaiserslautern, Germany, 1990. Springer Verlag.
32. A. Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 648–662, Saratoga Springs, NY, USA, June 1992. Springer Verlag.
33. A. Voronkov. Deciding K using kk. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 198–209, 2000.
34. A. Voronkov. How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi. *ACM Transactions on Computational Logic*, 2(2):182–215, 2001.