Deduction in Many-Valued Logics: a Survey*

Reiner Hähnle
Universität Karlsruhe
Institut für Logik, Komplexität und Deduktionssysteme
76128 Karlsruhe, Germany
haehnle@ira.uka.de, http://i12www.ira.uka.de/

Gonzalo Escalada-Imaz
Institut d'Investigació en Intel·ligència Artificial
Campus Universitat Autònoma de Barcelona
08193 Bellaterra, Barcelona, Spain
gonzalo@sinera.iiia.csic.es, http://www.iiia.csic.es

1 Introduction

Until the late 1980s research in many-valued logic (MVL) focussed on theoretical issues in proof theory, algebra, expressivity, axiomatizability and, on the applicative side, discrete function minimization and simplification. The first papers with practical implementation of deduction systems in mind came up in paraconsistent/annotated logic programming [18, 76] and in automated theorem proving [55, 108].

A partial survey of the results up to 1993 is contained in [58]. In the past five years deduction methods for MVL got more and more refined. Recent results can match those in classical theorem proving with respect to depth and attention to detail. They are not confined to mimicking improvements of deduction invented in classical logic, rather, specifically non-classical strategies are started to being pursued. As can be seen from the references list of this article, there is considerable activity in MVL deduction which is why we felt justified in writing this survey.

Needless to say, we cannot give a general introduction to MVL in the present context. For this, we have to refer to general treatments such as [153, 53, 93].

2 A classification of many-valued logics according to their intended application

Many-valued logic (MVL) can be conceived as a set of formal representation languages that have proven to be useful for both real world and computer science applications:

- Since expressiveness of fuzzy logic can be captured by infinite-valued logic [66] and its main phases, namely interpolation and defuzzification rely on functional theories, fuzzy controllers can be modeled by MVL together with

*Part of this work was supported within the Acción Integrada "Algorithms for Manipulating Discrete Functions" by DAAD (Deutscher Akademischer Auslandsdienst) and MEC (Ministerio der Educación).

appropriate logical calculi. Fuzzy control now is a well established area with a huge variety of industrial applications (see Section 7), used to optimize both discrete and continuous industrial processes: chemistry, mechanical and electrical engineering, medicine, scheduling, etc.

- Within computer science there is a broad span of fields relying on MVL; their scope covers a fairly large spectrum going from special purpose through general purpose computer systems. A non-exhaustive list of such fields is: Expert systems [52], error correcting codes [105], optimisation of discrete functions [129, 14], deductive databases [145], logic programming [91], constraint logic programming ([64] and the article of Lu in *this special issue*), automated theorem proving [58], inconsistent reasoning [88], dealing with partial knowledge [74, 114], non-monotonic reasoning [16], multi agent systems [156, 112, 118] and distributed AI [34, 103].

The expressiveness of MVL enables to model various aspects of uncertainty, namely those relying on truth functional theories [153] including fuzziness [66], stochastic signals modelled by functional (Bayesian, Markov) probabilities [116] and imprecise information [114].

MVL languages can also be used in general applications using different kinds of information coming from different real sources, for example, sensors supplying imperfect or uncertain information. Some application areas with these properties are robotics [117], multi agent Systems [156, 112, 118], decision processes [39], modular deductive systems [3, 2].

After this panoramic view of the attractive features inherent to MVL we classify the variety of MVLs according to four parameters: the number of truth values, their mathematical structure, the semantics associated to connectives and to first order quantifiers.

2.1 An abstract framework for many-valued logics

MVL owes its significant differences to classical logic from varying the four parameters indicated above.

Classical propositional logic is defined by two different Boolean algebras.

The **syntax** algebra $\langle \text{Prop}, C \rangle$ gives rise to well-formed (propositional) formulas (WFF) built by structural induction on the connectives $C = \{\neg, \vee, \wedge, \rightarrow\}$.

Meaning is assigned to WFFs by a Boolean **semantics** algebra $\langle \{0,1\}, I, A(C) \rangle$. The function I, called **valuation function** or **interpretation**, assigns a value in $\{0,1\}$ to each proposition in Prop. $A(C) = \{A(\neg), \ldots, A(\rightarrow)\}$ are mappings on $\{0,1\}$ of suitable arity associated with each connective symbol. As usual, I is extended to a homomorphism I from WFFs to $\{0,1\}$.

In the many-valued propositional case, the syntax algebra is then $\langle \text{Prop}, C \rangle$, where $C = \{\theta_1, \theta_2, \ldots, \theta_k\}$ and the semantics algebra is $\langle N, I, A(C) \rangle$, where $A(C) = \{A(\theta_1), A(\theta_2), \ldots, A(\theta_k)\}$. N represents the **truth values**. I is a valuation function that assigns to each proposition an element of N. C are the function (or connective) symbols with corresponding mappings $A(C)$ on N. Each pair of such algebras forms a **many-valued propositional logic** $\mathcal{L}$. Note that the homomorphic extension of I maps WFFs to N

In many applications the definition of the semantics has well-known properties. For example, if $N = \{0,1\}$ then classical propositional logic may be obtained. In other words, MV propositional logics generalize Boolean logic.

The syntax algebra of classical first order logic is a quadruple $\langle F, P, C, Q \rangle$, where F, P, C, and Q are the following sets, respectively: function, predicate, connective and quantifier symbols. Usually, $Q = \{\forall, \exists\}$ and $C \subseteq \{\neg, \vee, \wedge, \rightarrow\}$. Well-formed (first order) terms and formulas are defined inductively as usual.

The semantics algebra is $M = \langle \{0,1\}, \mathbf{T}, I_F, I_P, A(C), A(Q)\rangle$, where $\mathbf{T}$ is a domain or set of semantical terms, I_F associates with each m-ary function symbol a mapping $\mathbf{T}^m \to \mathbf{T}$; I_P associates with an m-ary predicate symbol a mapping $\mathbf{T}^m \to \{0,1\}$; $A(C)$ is as above; given M and a variable assignment $\beta : \mathrm{Var} \to \mathbf{T}$ one defines as usual inductively a valuation function $v_{M,\beta}$ that maps quantifier free formulas into $\{0,1\}$. This is extended to quantified formulas by stipulating that $A(Q)$ maps a quantified formula $\forall x\phi$ ($\exists x\phi$) to 0 iff there are (for all) variable assignments $v_{M,\beta}(\phi) = 0$.

A **many-valued first order logic** can be defined by extending the classical binary case to N and by paying special attention to the connectives and quantifiers. The many-valued syntax algebra again is $\langle F, P, C, Q\rangle$, where C is as in the MVL propositional case and $Q = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$.

The meaning of first order MVL formulas relies on the semantics algebra $M = \langle N, \mathbf{T}, I_F, I_P, A(C), A(Q)\rangle$. N is as in propositional MVL, $\mathbf{T}$ and I_F are as in classical first order logic.

Each connective symbol θ_i has an associated mapping $A(\theta_i) : N^k \to N$, where k is the arity of θ_i and each quantifier symbol λ_j has an associated discrete distribution function $A(\lambda_j) : (2^N - \{\emptyset\})^m \to N$, where m is the arity of the quantifier λ_j. In the present paper we only need $m = 1$, so the quantifier case of $v_{M,\beta}$ is then defined via

$$v_{M,\beta}(\lambda x\phi) = A(\lambda)(\{v_{M,\beta'}(\phi) \mid \beta' \text{ variable assignment}\})$$

Historically, a deduction problem in MVL was represented by a pair $\langle D, \phi\rangle$ where $D \subseteq N$ and ϕ is a WFF. Thus, the deduction problem is to search for interpretations that assign to a given WFF a truth value in D, the set of **designated truth values**, which generalize the rôle of "1" in the Boolean case. Two general deduction problems are distinguished: (1) whether there is at least one such interpretation, or (2) whether all interpretations have the mentioned property.

In the first case, we speak of D**-(un)satisfiability** of a WFF and in the second case, of a D**-tautology**. Other concepts such as D**-logical consequence**, sound inferences of WFFs, D**-logically equivalent** formulas are easily defined analogously to classical logic.

2.2 The cardinality of the truth value set

In this subsection we present a classification of MVLs based on their number of truth elements.

Finite If the cardinality of the set N is very small (say, less than ten but, more typically, three or four), then the functions (connectives) are generally defined by truth tables or algebraically by some standard mathematical structures as for instance lattices, partial orders, etc. [124]. Some well known MVLs of this type are Kleene's three-valued weak and strong logic [80], often defined by truth tables and Belnap's Logic [6], usually defined by a lattice. These MVLs are suitable for applications where deduction must be done in presence of partial knowledge [74, 114], inconsistency [88], or non-monotonicity [37].

Notice yet, that if functions are defined by truth tables, then memory requirements grow exponentially with the arity of the involved connectives. Then, even as truth tables for function definitions are convenient, because its easy to define whole families of functions, the truth table approach becomes infeasible due to this exponential increase.

MVLs with a reasonable limit on the cardinality of the truth value set are often found in applications where the truth values are called "linguistic labels", such as

Very High, *Quite High*, *High*,..., *Very Low*. These logics appear in many important applications such as expert systems [52] and fuzzy control [23].

Sometimes a larger (though finite) number of truth degrees is needed; this is the case for discrete optimisation functions, hardware verification [44], approximation of continuous fuzzy sets.

When the number of finite truth values or the arity of connectives is truly large, their functions need to be calculated without truth tables; one chooses suitable computable functions such as max, min, addition, subtraction, exponential, logarithm, polynomial. These are sufficient to compute functions such as T-norms [54] and certain implications [92].

Infinite When the number of truth values is infinite, two cases arise: either they are countable or not.

Applications with a countably infinite number of truth values appear in real world contexts, where in practice approximations of continuous variables are acceptable by considering discrete variables with infinite domains.

These approximations are suitable for all the real world systems based on discrete variables such as the control processes, where the discrete phenomena emerge due to the inclusion of a computer into the process control loop.

MVLs with an uncountably infinite number of truth values permit a sharp modelling of any kind of problem requiring continuous variables. For example, this situation is common in control processes based on analog hardware.

Analog hardware may be employed for classical and current fuzzy controllers, too. Indeed, both kinds of control systems are modelled with continuous variables and functions, in other words, with MVLs where variables and functions are continuous.

Thus, given that a computer cannot represent an infinite number of values corresponding to the domains of continuous variables and functions, such applications as diagnostics in analog systems, say analog electronic circuits, need a different approach than the truth tables and computable functions mentioned before.

Still, often even these system models allow a finite representation: figures in its transfer functions stand for thresholds such that beyond a certain threshold the behaviour of the system changes. Then, any system can be controlled in practice by considering merely the numerical values explicitly occurring in the model of a system. Thus, the control system can be modelled with appropriate finite MVLs.

2.3 The structure of the truth value set

Next, we discuss MVLs according to the different mathematical structure of N. More precisely, the input variable domains, the connective functions and the output value must be related to the mathematical structure of N. Vice versa, if we need functions for applications in a certain mathematical structure, then the variables' and functions' domains must be defined according to the nature of N.

Unordered The structure of N is a simple set. For example, this is the case in the definition of MV logics and their deduction problems represented by the pair $\langle D, WFF \rangle$.

In recent years it has been suggested [55, 108] to associate a set $S \subseteq N$ with *each* formula ϕ in an MVL and then consider pairs $\langle S, \phi \rangle$, called **signed formulas** as an atom of Boolean logic. Instead of $\langle S, \phi \rangle$ one writes more compactly $S\,\phi$.

$v_{M,\beta}(S\,\phi)$ holds (that is $S\,p$ is **satisfiable**) iff $v_{M,\beta}(\phi) \in S$. In the case when ϕ is atomic, $S\,\phi$ is called a **signed literal**.

This mixed Boolean and many-valued logic, called **signed logic**, has turned out to be of great practical interest. Indeed, this logic preserves most of the properties of propositional logic and the set S, depending on the context, models constraints, imprecision, inconsistency and other things.

A particular subset of signed MVLs are the **monosigned** MVLs where the sign S of each formula is a singleton.

Another simple and unordered structure used in MVL, and which we encountered already, is the **truth table**. Each connective θ has an associated truth table. If $A(\theta)$ is the connective function of arity 2, the entry at $\langle i, j \rangle$ of the table indicates the value of $\theta(i, j)$. In the general case, the truth table of an n-ary function is an n-dimensional hypercube.

Truth tables have the nice property that all discrete functions can be defined with them, but this feature is of limited value in practice, because the ratio memory/k (k being the maximal arity of any connective) increases exponentially.

Partially ordered In many real world situations, the available information can be imprecise, inconsistent, and partial. Hence, many results concerning knowledge representation and its deductive aspects cannot be employed straightforwardly.

Many problems that feature inconsistent and partial knowledge come from distributed AI [34, 103] and decision theory [39, 95].

Belnap's four-valued lattice and other (three-element) lattices have been envisaged to deal with: partial, inconsistent knowledge in so-called paraconsistent logics [88], non-monotonic reasoning [37].

The lattice order corresponds to degrees of truth, knowledge, or of other parameters. Ginsberg [51] suggested an algebraic structure allowing to combine several parameters.

Ginsberg's ideas were used in [95] which is discussed in Section 7. Another proposal to combine several sources of uncertainty is described in [3, 2]. Here the problem is tied to modular programs where each module is developed relative to a particular MVL. Their difference is chiefly based on the number of truth degrees. Thus, the technical difficulty is due to the need of combining the truth degrees coming from a set of modules. Indeed, the combined truth degree must ensure global soundness with respect to local module soundness. The idea is to define appropriate quasi-morphisms among different MVL algebras which allow to determine mappings over truth degrees of two particular modules. In [2] restrictions concerning the MVLs that can be used in modules are set up.

Totally ordered MVLs with totally ordered sets of truth values have computationally attractive features. They are also natural for graded truth found in many real world applications. For instance, in uncertainty management, the larger a truth value attached to a knowledge unit, the more certain it is. The generic problem can be modeled by a pair $\langle$truth level, $\phi\rangle$, "truth level" being a label indicating that the problem has a solution if there exists an assignment such that the truth value of the WFF ϕ is greater or equal than the "truth level" [42, 62, 87, 106].

Graded truth can be mapped to a signed formula of the form $\langle \phi, S \rangle$, where S is of the form $\{i \in N \mid i \leq i_0\}$ or $\{i \in N \mid i \geq i_1\}$, with respect to a total ordering $<$ of N. In this case S is a **regular sign** [56]. If all literals of a classical formula are signed formulas with regular signs relative to the same total order, we speak of a **regular formula**. In the case of regular signs and formulas, N can without loss of generality, assumed to be a set of equidistant rational numbers of the form $\{0, \frac{1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$. Some aspects concerning the negation function in representations where the graded thuth values are not equidistant are studied in [150].

If signs are taken from the rational or real unit real interval one obtains infinite regular logics [42, 62]. In practice, knowledge bases are finitely bounded by memory. Hence, the number of knowledge units (bases and rules) as well as the number of truth values in each sign taken from the rational or real unit interval, are finite.

2.4 Connectives and Quantifiers

Recall from Section 2.1 that any function $\theta : N^k \to N$ can be considered as a many-valued connective while any function $\lambda : (2^N - \{\emptyset\}) \to N$ can be considered as a many-valued quantifier. Such diversity can lead to high computational cost for deduction in the worst case (Section 5). Fortunately, the many-valued connectives and quantifiers arising from applications are much more structured. In the following we mention some important cases.

For instance, it is quite often the case that connectives θ are **normal** [124] which means there is a designated truth value $1 \in D \subset N$ and a non-designated truth value $0 \in N - D$ such that the restriction of θ to $\{0, 1\}$ yields a classical connective.

As an example we mention the so-called **T-norms**, **S-norms**, and **R-implications**, a family of binary connectives widely used in fuzzy logic [54]. Kleene's connectives based on min and max with respect to a total ordering of N are particular instances of a T-norm and S-norm.

Other many-valued connectives closely related to the classical ones are the so-called **regular connectives** introduced by Hähnle [56]. They are characterized by a monotonicity condition with respect to a total ordering of N plus a restriction to regular signs (Section 2.3). An attractive feature of regular connectives is that nearly classical deductive techniques can be employed for them.

Furthermore, just as in the classical case, many-valued quantifiers often can be gained by infinitely iterating associative and commutative propositional connectives. Thiele [149], for instance, derives **T-quantifiers** and **S-quantifiers** from T-norms and S-norms, respectively. Zabel [157] and Zach [158] showed that quantifiers induced by associative, commutative and idempotent connectives have crisp proof theoretical characterizations. Recently, these results were generalized to quantifiers based on certain types of lattices [61, 128].

As a final example of connectives that relate many-valued to two-valued logic we mention Rosser & Turquette's [125] unary assertion connectives J_i which evaluate to 1 if their argument evaluates to i and and give 0 otherwise. Such connectives allow to express the semantics of other connectives as is witnessed by the fact that any finite-valued logic containing $\wedge = \max$, $\vee = \max$, and J_i for all $i \in N$ is **functionally complete**, i.e., for each $f : N^k \to N$ there is a formula $\phi_f(p_1, \ldots, p_k)$ such that $f = I(\phi_f)$.

2.5 Expressivity

While the additional truth values of many-valued logic do increase expressivity, well-known syntactic classes of classical logic are orthogonal to this and remain unaltered. An important distinction is full logic syntax vs. clause normal form syntax. Most many-valued logics do not possess a subset of their wffs that can be regarded as normal form. In general it is necessary to extend the syntax in order to obtain normal form.

Accordingly, the calculi developed for deduction with many-valued logic formulas in normal form differ substantially from those dealing with full syntax. Also computation of normal form in many-valued logic is a much more difficult process than in classical logic as it has to account for the characteristics of each logic. On the other hand, it turns out that there is a generic, clause-based normal form into which each finite-valued first order logic can be transformed, see Section 4.1. As

a consequence, clause-based reasoning in many-valued logic is independent of the logic from which the clauses were obtained by transformation.

Once clausal normal form has been obtained, one can define a Horn fragment, Krom (2-CNF) fragment, Datalog fragment and so on, see Section 5.

2.6 Combining Many-Valuedness with other Non-Classical Features

It is an interesting question how deduction systems for many-valued logics and for other non-classical logics can be combined. This has been done for many-valued plus higher-order sorted logic [74, 73], many-valued plus modal logic [45, 46] and even many-valued non-monotonic modal logic [47]. An interesting case is [99, 100], where additional truth values were used to simplify a tableau system for intuitionistic and modal logic while, on a similar line, Doherty [38] showed that a particular three-valued logic can ease the proof theoretic treatment of certain non-monotonic logics.

[31] showed that the modal logic $S5$ can be captured by finite-valued propositional logic. A general framework for obtaining deduction systems for combinations of non-classical logics is [49]. It has been applied within the context of MVL in [139].

3 Proof theory of many-valued logics

As is the case for other non-classical logics, proof calculi for deduction in MVL can be roughly divided into two classes:

Internal calculi: the objects constructed during a proof are from the same language as the goal to be proven; a typical example are Hilbert type calculi.

External calculi: the objects occurring during a formal proof are over an *extended* language that may involve elements from the semantics such as designators for truth values, worlds or even non-logical expressions such as constraints; a typical example are signed semantic tableaux.

Proof theorists often only accept calculi of the first kind and regard the second option as a kind of "cheating". On the other hand, if a uniform and computationally efficient treatment of deduction is desired, there seems to be no alternative to external calculi: otherwise, highly indeterministic rules such as cut and weakening are inevitable, even if an internal axiomatization exists.

A somewhat extreme position of gaining a classical logic approach to deduction in non-classical logic would be to formulate the "external" elements in the second approach as a meta theory in classical logic. For a wide range of logics this is even possible in first order logic. The "meta theory" of finite-valued logic in particular can always be captured without having to move to a higher-order stage.[1] From the viewpoint of efficiency, however, it is definitely better to formulate dedicated calculi for many-valued logics. The situation is analogous to higher programming languages which usually contain built-in functions for, say, standard arithmetical operators instead of having them implemented in the language itself.

3.1 Internal Proof Systems

Hilbert style calculi for all well-known MVLs may be found, for example, in overviews such as [124, 22]. "Internal" Gentzen style calculi for some logics were given by Avron [9] and Hösli [68].

[1] This is not necessarily true for infinite-valued logic, see [134].

An interesting example of an internal proof system constitutes many-valued non-clausal resolution introduced by Stachniak [141], now nicely summarized in [142]. The basic idea is to derive from formulas $\phi(p)$ and $\psi(p)$ (where p is an atom occurring in ϕ and ψ) a new formula[2] $\phi(\rho) \vee \psi(\rho)$ for certain variable-free formulas ρ and then to perform logic-specific simplifications.

3.2 External Proof Systems

As pointed out at the beginning of this section, we now extend the language used in the proof wrt to the language of the formula to be proven. Many notations have been invented, the most flexible being the signed formulas of Section 2.3:

As pointed out, it is possible to view a signed formula $S\,\phi(x_1, \ldots, x_m)$ with free variables $x_1, \ldots, x_m$ itself as an atomic expression and to build classical first order formulas over such atoms, for example, $S\,(p \sqcup q) \vee S'\,r$. Here, $\sqcup$ is a many-valued connective, say $\sqcup = \max$, and $\vee$ is classical disjunction.

Each of the following deduction systems must have means to express a signed formula $S\theta(\phi_1, \ldots, \phi_m)$ (where θ is an m-ary connective) by a classical formula over atoms of the form $S_i\ \phi_i$ for suitable $S_i \subseteq N$, formally:

Theorem 1 *Let $\phi = S\theta(\phi_1, \ldots, \phi_m)$ ($m \geq 1$, $S \subseteq N$, $|N| = n$) be a signed formula from an n-valued logic $\mathcal{L}$. Then there are numbers $M_1, M_2 \leq n^m$, index sets I_1, $\ldots$, I_{M_1}, J_1, $\ldots$, $J_{M_2} \subseteq \{1, \ldots, m\}$, and signs $S_{rs}, S_{kl} \subseteq N$ with $1 \leq r \leq M_1$, $1 \leq k \leq M_2$ and $s \in I_r$, $l \in J_k$ such that*

ϕ is satisfiable iff $\bigvee_{r=1}^{M_1} \bigwedge_{s \in I_r} S_{rs}\,\phi_s$ is satisfiable iff $\bigwedge_{k=1}^{M_2} \bigvee_{l \in J_k} S_{kl}\,\phi_l$ is satisfiable.

In this generality the theorem was first proven by Hähnle [55], other references are given below. A similar result holds which grants elimination of many-valued quantifiers by a generalization of classical Skolemization.

Obviously, by repeated application of Theorem 1 each signed formula can be converted to a *classical* formula over *signed literals*.

By the usual transformation then, a *classical* CNF or DNF formula over signed literals, called **signed CNF/DNF formula**, is obtained. Some MVL deduction systems assume this conversion was done in a preprocessing step: they check consistency of conjunctions of clauses of the form

$$S_1\,p_1 \vee \cdots \vee S_m\,p_m \tag{1}$$

where the $S_i\ p_i$ are signed literals. Other calculi interleave normalization and consistency checking.

3.2.1 Sequents and Tableaux

Recall that in classical logic signed semantic tableaux and sequent systems correspond to each other very closely (see, for example, [48, p. 96f]). This extends to the many-valued case. Indeed, both sequent *and* tableau rules can be derived from Theorem 1, if one keeps in mind that tableau rules correspond to a disjunction of conjunctions, whereas sequent rules correspond to a conjunction of disjunctions. Thus, if S_{rs}, S_{kl} are as in the theorem, then the following sequent, resp., tableau rules are sound and complete for the connective appearing in the premise provided that there is a rule for each occurring sign:

[2] There are workarounds if $\vee = \max$ happens not to be a connective of the given logic.

$$\frac{\Gamma \cup \bigcup_{l \in J_1} S_{1l}\, \phi_l \quad \cdots \quad \Gamma \cup \bigcup_{l \in J_{M_2}} S_{M_2 l}\, \phi_l}{\Gamma \cup \{S\, \theta(\phi_1, \ldots, \phi_m)\}} \qquad \begin{array}{c} S\, \theta(\phi_1, \ldots, \phi_m) \\ \hline \vdots \\ S_{1s}\, \phi_s \\ \vdots \end{array} \;\Big|\; \begin{array}{c} \vdots \\ \cdots \\ \vdots \end{array} \;\Big|\; \begin{array}{c} \vdots \\ S_{M_1 s}\, \phi_s \\ \vdots \end{array} \tag{2}$$

Note that unlike in classical logic sequents are not pairs of formula sets separated by an arrow, but simply sets of signed formulas (recall that a classical sequent $\Gamma \Rightarrow \Delta$ can be expressed with $\{\{1\}\, \gamma |\, \gamma \in \Gamma\} \cup \{\{0\}\, \delta |\, \delta \in \Delta\}$).

Inconsistency (i.e. **closure**) of tableau branches is signalled by one of the following conditions. Either there are formulas $S_1\, \phi, \ldots, S_m\, \phi$ on a branch such that $\bigcap_{1 \leq i \leq m} S_i = \emptyset$ or there is a complex formula $S\, \phi$ such that none of the truth values in S is reached by the leading connective of ϕ. Tautologyhood or, in other words, axiomatic status of sequents is expressed by dual conditions.

All results in this section still hold for monosigned formulas. In this case, there is no need for set brackets and one simply writes $i\, \phi$ for signed formulas. In fact, historically, Theorem 1 was first proven for the singleton case and sequents independently by Rousseau [126] and Takahashi [148]. Other authors who had essentially the same idea, but with certain restrictions, are Schröter [137] and Kirin [78, 79]. Surma [147], Carnielli [33], Baaz & Zach [13], Zabel [157], and Bloesch [21] worked the dual tableau case. Natural deduction systems based on the same idea are described in [12]. A summary of results is contained in [11].

The idea of using truth value sets as signs is due to Hähnle [55] and, independently, to Doherty [38] and Murray & Rosenthal [108, 110]. It occurs in disguise for a special case in [146].

Needless to say, all these authors invented a plethora of notations to denote what boils down to signed formulas. Rousseau [126], for instance, uses n-ary sequents of the form

$$\Gamma_0 \,|\, \Gamma_{\frac{1}{n-1}} \,|\, \cdots \,|\, \Gamma_1$$

in which the i-th slot contains the formulas being asserted truth value i.

3.2.2 Resolution

Many-valued resolution systems work on clauses of the form (1).[3,4] It is assumed that a CNF over such clauses has been achieved somehow, see Section 4.1 below and the discussion following Theorem 1. Note that signed clauses are independent of the many-valued logic they originated from. In fact, signed clauses do not contain *any* many-valued connective and are simply a generic and flexible language for denoting many-valued interpretations.

As in classical logic it is common to identify a signed clause with its set of literals that is sequence and multiplicity of literals is irrelevant.

Recall that classical resolution is based on combining clauses that contain inconsistent literal sets. Many-valued resolution does exactly the same, but, of course, one has to use the many-valued version of inconsistency defined in Section 3.2.1. In contrast to classical logic inconsistency sets in general are required to contain more than two elements. In this case one has a choice whether to combine two clauses at a time or all required clauses at once. Accordingly, most many-valued resolution rules are instances of one of the following schemata:

$$\frac{S_1\, p \vee C_1 \quad \cdots \quad S_m\, p \vee C_m}{C_1 \vee \cdots \vee C_m} \quad \text{if} \bigcap_{1 \leq i \leq m} S_i = \emptyset \tag{3}$$

[3] Non-clausal resolution is mentioned in Section 3.1.

[4] Again, a lot of different notations for signed clauses were invented.

$$\frac{S\ p \vee C \qquad S'\ p \vee C'}{(S \cap S')\ p \vee C \vee C'} \qquad\qquad \frac{\emptyset\ p \vee C}{C} \tag{4}$$

Rule (3) can be simulated by several applications of rules (4). On the other hand, (4) has resolvents which can't be produced with (3). Let us call (3) **(many-valued) hyperresolution** and (4) **(many-valued) binary resolution**. The literal $(S \cap S')\ p$ in binary resolution is called a **residue**, the rule on the right in (4) is called a **reduction rule**.

Similar as for tableau and sequent calculi, historically the monosigned restriction of rule (4) came first [84, 83, 102, 10]. Orłowska [113] and Schmitt [135, 136] implicitly considered truth value sets as signs in a specialized context.

Rule (3) appeared first in [57] and, independently, rule (4) in [108, 110]. All of those papers stipulated a many-valued **merging rule** as well, but in [62] is proven that either of (3) and (4) alone is complete with just merging of identical literals.

It is easy to prove [111, 62] that every CNF formula over signed literals is equivalent to one in which only regular signs occur. Such a formula is called a **regular formula**. For regular formulas refined versions of many-valued resolution were given [62].

Lifting of resolution to first order CNF formulas over signed literals is done exactly as in classical logic and requires no further discussion.

The restriction of binary resolution to the case when one input clause must be a unit **(unit resolution)** is at the heart of the Davis-Putnam-Loveland procedure [35]. Its many-valued version was introduced for regular formulas in [62]. Recently, it has been analyzed and improved by Manyà [94], see also Section 5.

Several resolution-based calculi were also given for logic programs based on signed formulas. They are discussed in Section 4.3.

Lehmke [85, 86] gave a resolution system for what he called **weighted bold clauses**. These are signed pairs where the first argument is a multiset of literals of length n and the second is one of $\{0, n-1\}$. Their semantics is given by

$$I\left(\langle\langle l_1, \ldots, l_n\rangle, \delta\rangle\right) = \max\{0, \min\{1, \sum_{i=1}^{n} I(l_i) - \delta\}\} \ .$$

Conceiving clauses as sequences or multisets instead of sets allows to handle many-valued logics (such as Lukasiewicz logic) for whose connectives the law of idempotency does not hold, see Section 4.1 below.

A resolution step is split up into two stages representing combination of clauses and removal of an inconsistent pair of literals:

$$\frac{i\,\langle\langle l_1, \ldots, l_n\rangle, \delta\rangle \qquad\qquad i' : \langle\langle l'_1, \ldots, l'_m\rangle, \delta'\rangle}{\min\{i, i'\}\,\langle\langle l_1, \ldots, l_n, l'_1 \ldots, l'_m\rangle, \delta + \delta' + \max\{i, i'\}\rangle} \qquad \frac{i\,\langle\langle \ldots, l, \ldots, \overline{l}, \ldots\rangle, \delta\rangle}{i\,\langle\langle \ldots\rangle, \delta - 1\rangle} \tag{5}$$

The sign i here has the meaning $\{j \in N \mid j \geq i\}$. Inconsistent signed bold clauses $i\,C$ are those with $I(C) < i$ for all I which can be easily checked.

3.2.3 Decision Diagrams

No account of deduction in many-valued logic would be complete without mentioning the extensive body of work done in the area of Computer Aided Design (CAD) of digital circuits, where many-valued deduction tasks are encountered for quite some time (see Section 7 below).

While early approaches often were of a heuristical nature and did not correspond to formal logical systems [25], in the last decade a family of logical calculi called **decision diagrams** (DD) became the dominant tool. For the binary case, DDs are

often credited to [5], but they appear already in the work of Boole, Shannon and others. Binary DDs (BDD) were popularized in the CAD community by Bryant [26], but they really took off only after efficient implementations became available [24]. Many-valued DDs (MDD) were introduced in [140].

From a logical point of view DDs are closely related to tableau systems. Many of them can be seen as being derived from the DNF characterizations $\bigvee_{r=1}^{M} C_r$ (where $C_r = S_{rs}\ \phi_s$) provided in Theorem 1 with the additional condition that the union of any two conjuncts C_i, C_j is inconsistent in the sense of many-valued tableau branches (Section 3.2.1). This expresses the restriction that the case analysis expressed in the disjunction over the C_i's is exclusive. Even if DD methods are close to tableaux from a purist point of view they add important features: (a) rules are applied in a fixed order determined from a total ordering on atoms, (b) simplification steps on variable-free formulas are performed whenever possible, and (c) identical subformulas are expanded at most once imposing a DAG structure on DDs.

Conditions (a)–(c) together ensure that most variants of (binary and many-valued) DDs are a strong normal form for Boolean, respectively, discrete functions: formulas with identical functions have the same DD. This is an important property for implementation.

The relationship between classical tableaux and BDDs is worked out in [119], aspects of the relationship between many-valued tableau systems and MDDs are discussed in [63].

Surveys and introductions to DD techniques are provided by [27, 144, 101]. Some recent research papers are collected in [133].

We stress that DD methods are essentially confined to the ground case as property (a) is not compatible with applying substitutions and the strong normal form property is lost [120].

3.2.4 Other Calculi

There is a deduction method which, like non-clausal resolution, avoids to compute any normal form altogether: Murray & Rosenthal's **dissolution rule** is available both for classical [109] and finite-valued logics [108, 111].

Many-valued dissolution operates on formulas in **signed negation normal form** (NNF), i.e. formulas built up from $\wedge$, $\vee$ and signed literals. The dissolution rule selects in a signed NNF formula an implicitly conjunctively connected pair of literals $S\ p$, $S'\ p$ and restructures it in such a way that at least one conjoint occurrence of $S\ p$, $S'\ p$ is replaced with $(S \cap S')\ p$. Producing $\emptyset\ p$ leads to obvious simplifications such that any unsatisfiable formula is reduced to the empty formula after a finite number of dissolution steps.

In contrast to dissolution, the so-called TAS method [1] computes a simplified DNF of a given formula in NNF. The input formula is unsatisfiable iff the result is the empty formula i.e. falsity. The power of the method comes from the fact that before each application of the distributive laws unitary models of subformulas are computed and used for simplification. The generalization of the TAS method to signed NNF formulas is found in *this special issue*.

Both, the dissolution and TAS method can principally be lifted to first order logic. This is not the case for the next approach which, on the other hand, is one of the few practical deduction methods that can deal with infinite-valued logics.

Hähnle [59] showed the following: given any formula ϕ of a logic whose connectives are expressable in infinite-valued Łukasiewicz logic, then there is a number of linear inequations with size in $\mathcal{O}(|\phi|)$ with rational variables over $[0, 1]$ and discrete variables over $\{0, 1\}$ having a solution iff ϕ is satisfiable. Problems of the latter kind are known as a *mixed integer programming* problems and there are well-developed tools for solving them.

4 Types of deduction problems

4.1 Normal form computation

Most known normal forms of classical logic have many-valued counterparts [60, 11]. Several deduction problems defined on those normal forms have been already discussed, so efficient approaches to produce normal forms should be discussed as well.

MVL normal forms are mostly adapted directly from classical ones, with the specific nature of an MVL in mind, see Section 2. Thus, CNF and DNF was defined for signed logic and its regular and monosigned subclasses (Section 3.2). As pointed out in Section 3.2.2, [60, 11] give sound and complete resolution-like calculi for signed CNF formulas. These calculi turn out to be particularly significant in light of the result that any unrestricted WFF of any finite-valued first order logic can be transformed into a signed CNF Formula in polynomial time [60], see also Section 5.

(Regular) many-valued Horn formulas are straightforward to define once it is noted that for $j \in N$ regular signed literals of the form $\{i \mid i \geq j\}\ p$ can be considered **positive** while literals of the form $\{i \mid i \leq j\}\ p$ are **negative**. This notion of polarity was introduced in [62].

In [108] signed NNF formulas are defined: these are negation-free Boolean formulas with signed literals or signed formulas as atoms.

Lehmke [85, 86] observed that every formula of infinite-valued Lukasiewicz logic can be expressed in signed NNF provided that Lukasiewicz sum $\oplus$ and product $\otimes$ are used instead of classical disjunction and conjunction.[5] He called this **hierarchical normal form**. Essentially by using Tseitin's [151] trick of introducing new atoms to abbreviate complex expressions he can show that any Lukasiewicz formula can be converted into a CNF over so-called bold clauses as mentioned at the end of Section 3.2.2.

4.2 Satisfiability and related Deduction Problems

In this section, as before, N is a set of truth values with cardinality n in case it is finite.

Here we point out some deduction problems for which neither computational nor complexity analysis have been discussed. In contrast to this, Section 5 is devoted to problems whose computational features have already been established.

In Section 2 we defined the problem of satisfiability of first many-valued logic. Recall for a given WFF and a subset D of N (called the designated truth values) the notions of D-satisfiability, D-validity and D-consequence.

Satisfiability in MVL often can be used to define logical consequence, logical equivalence, etc (that is, whenever consequence is S-consequence for some $S \subseteq N$). On the other hand, consequence relations deviating from this approach can be considerably less [115, 30] or more complex [153] than the satisfiability problem.

In the following we discuss computational properties of some of the MVLs mentioned in Section 2. We divide them into three classes:

1. Signed logic [58, 111, 11].

   One works with Boolean formulas over signed literals that is pairs $\langle S, p\rangle$ (usually written as $S\ p$), where p is an atom (or sometimes a complex formula) and $S \subseteq N$ is called the sign of p. A signed literal is satisfied exactly by those interpretations I such that $I(p) \in S$. In contrast to the Boolean literal case there are n^k different interpretations over k propositional symbols.

[5] This process can blow up a formula exponentially.

As we mentioned, there are two sub-cases of signed logic that deserve to be considered.

(a) In case N is a totally ordered set, we introduced regular formulas, where signs are of the form $\{i \mid i \geq j\}$ or $\{i \mid i \leq j\}$, where $j \in N$. As only the thresholds j instead of the whole set of truth values in each sign are required to perform sound and complete inferences, one often uses notations such as $p \geq j$ or $p \leq j$ for regular signed formulas.

(b) If each sign is a singleton we speak of monosigned formulas. As with regular formulas, no other signs need to be involved in a deduction, and one typically uses a notation such as $i\,p$ instead of $\{i\} : p$.

If one restricts attention to monosigned CNF formulas, then automated deduction in the two latter cases can be improved by pruning techniques, see [43] (in contrast to the general signed logic framework).

2. One may also consider unsigned, non-Boolean functions built from arbitrary connectives θ with meaning $A(\theta) : N^k \to N$. These functions can either have mathematically justified semantics as do T-norms, S-norms, and residuated implications employed in fuzzy environments or, otherwise, their semantics is *ad hoc* and depends on the intuition and experience of experts in a particular application context. N can be finite or countably/uncountably infinite (see Section 2).

3. In case N is a partially ordered set (for example, a lattice) one may proceed similarly as in signed logic: for example, in [75, 88, 95] a literal is of the form $\mu\ p$ (or similar), where μ is an element of a partially ordered set P. An interpretation I satisfies (does not satisfy) $\mu\ p$ iff $I(p) \geq_P \mu$ ($I(p) \not\geq_P \mu$). But in partial orders (and in lattices in particular) one has generally not $P = \{i \mid i \geq_P j\} \cup \{i \mid i \leq_P j\}$. This fact is exploited to handle inconsistency with the partial order determined by a lattice structure [75] as well as to accomodate partial knowledge [97] and some uncertainty paradigms [81].

Based on the principles described above, other deduction problems have been tackled. We give examples of deduction problems which have been studied but for which neither experimental nor complexity results were obtained yet. Others may be found in Section 3.2.4.

1. Obtaining Prime Implicants/Implicates in NNF Regular and Post Logics. Ramesh & Murray discuss a novel approach having the advantage that *no intermediate clause normal form* must be computed. As [122] is only an abstract and [123] is not easily available, a full account of this work is contained in *this special issue*.

2. In a similar vein is an effort to simplify signed NNF formulas by removal of so-called anti-links [14], again without retracting to clause form.

3. Determining the relationship between signed logic programing and constraint logic programming [69]. A first step into this direction is the article by Lu in *this special issue*.

4.3 Deductive Databases and Logic Programming

Relational databases correspond to recursion-free, safe Datalog (i.e. without function symbols) programs [152]. Expressiveness of queries resp. conciseness of the data can be improved by allowing recursion or admitting function symbols or non-Horn

rules. All of these drastically increase computational cost (the first two relaxations even imply undecidability). Not so signed formula logic programming (SFLP) and annotated logic programming (ALP).

A signed formula logic program [32] is a collection of signed clauses of the form

$$S\ p \leftarrow S_1\ \phi_1, \ldots, S_n\ \phi_n\ \ , \tag{6}$$

where p is an atom and the ϕ_i are formulas of a finite-valued logic. By using signed CNF representations of the signed formulas $S_i\ \phi_i$ (cf. Section 4.1) one may assume without loss of generality that all ϕ_i are atomic [87].

An **annotated** (sometimes called **paraconsistent**) **logic program** [19, 88, 77] has the same form as (6), but S and S_i are not arbitrary finite sets of truth values; they denote principal filters in complete lattices. Moreover, the ϕ_i are assumed atomic from the start.

Often paraconsistent logics (such as Belnap's logic [17] are based on the lattice $\mathcal{FOUR}$ which contains besides the classical truth values 0 and 1 those for representing missing information ($\bot$) and contradictory information ($\top$) on the truth of a proposition. The partial order employed here is $\bot < 0$, $\bot < 1$, $0 < \top$, $1 < \top$. The logic based on $\mathcal{FOUR}$ has also been represented by Petri Nets [107].

Lu [87] showed that ALP and SFLP can be translated into each other, see also [89, 90]. ALP (and thus SFLP) can also be considered as generalizations of regular formulas (Section 3.2.2), because the truth values in the signs need only be partially ordered.

Deductive tasks in databases differ from automated theorem proving: while consistency of a database is important, it is often guaranteed by non-deductive means and even if not, rarely performed.

More important tasks are query answering, updates, and query optimization. **Query answering** means to decide whether a conjunction of atoms logically follows from a given logic program. It is important for such algorithms that they can take advantage from the fact that a logic program does not change between subsequent queries that is there should be some sort of compilation. This compilation process should be incremental in order to allow database **updates**. As these requirements are fulfilled by standard logic programming techniques one seeks for deductive algorithms which are close those for standard logic programming [82, 98, 95]. Another topic derived from classical database theory is the optimization of queries before they are submitted [81].

Non-monotonic ALP is considered in [16] while [91] is an overview of generalized logic programming.

5 Complexity of deduction in many-valued logics

Although in recent years the number of theoretical papers focussing on MVL was considerable, most of them highlighted on expressiveness and proof theory. Still, some effort was devoted to computational complexity in MVL. We review the main results.

A pioneer result is due to Mundici [104] who proved that satisfiability of formulas in infinite-valued Lukasiewcz logic is NP-complete. A different proof that works via reduction to mixed integer linear optimization was obtained by Hähnle [59].

As to satisfiability of signed, regular and monosigned formulas (See Section 2.3 and 3.2), although it has not been explicitly claimed by any author, it is straightforward to see that their satisfiability problems are all NP-complete and the dual validity problems are co-NP-complete.

More surprising is that for signed CNF formulas already signed 2-SAT (all clauses contain at most two literals) is NP-complete [94] as opposed to the two-valued case, which is linear [7] and even belongs to NC [71].

If one restricts the 2-SAT problem to *regular* formulas, then an $\mathcal{O}(|\Phi|P)$ algorithm ($|\Phi|$ is the number of literals in a signed CNF formula Φ and P denotes the number of positive clauses) can be obtained and if the number of truth values is fixed, then even a linear algorithm is possible [94].

Also MV satisfiability problems derived from the Horn restriction were proved to be tractable, and very efficient algorithms were proposed. The first such result was given in [42], where an algorithm with complexity $\mathcal{O}(|\phi| \log n)$ is described (ϕ is a Horn formula over classical syntax with $\wedge = \min$, $\vee = \max$, $\neg p = 1 - I(p)$, and each clause has a regular sign). In [62] an algorithm with the same complexity is proposed, but for the whole class of signed regular Horn formulas (that is each atom may have a different signs attached to it and $\vee$, $\wedge$, $\neg$ are classical). In [42] a method to obtain minimal models in linear time for a certain subclass of regular CNF formulas is given. For the case when $n = |N|$ is fixed, linear algorithms for regular Horn formulas are in [62, 94].

Recently, a set of on-line (that is: incremental) algorithms for Horn formulas with numerical uncertainty has been proposed [8]. The uncertainty problems are of numerical type. The uncertainty logic may be seen as an MVL as follows: $\wedge$ and $\vee$ connectives are interpreted by the min and max functions, respectively. Implication is the product of the uncertainty factor of the rule and the minimum of the uncertainty values in the premise. For this MVL data structures are studied which admit a linear worst case complexity of the satisfiability problem in formula size, however, it is assumed that multiplication can be done in constant time, whereas the best known algorithm for multiplication is in $\mathcal{O}(k \log k \log^2 k)$ [4]. For infinite-valued logic (that is unbounded number of truth values), an almost cubic algorithm is described.

[60] contains a method to transform any signed formula into a satisfiability equivalent signed CNF formula with worst case complexity in $\mathcal{O}(n^k|\phi|)$, if connectives are at most k-ary. This is at the same time a tight upper bound for the size of the representations given in Theorem 1.

A many-valued propositional backward interpreter for stratified logic programs with linear worst-case complexity is detailed in [43].

Mundici & Olivetti [106] also propose polynomial algorithms to decide regular Horn and 2-CNF problems and for computing minimal models of regular Horn formulas, but with weaker worst-case bounds than [94]. Their main new contribution is the insight that every signed formula of infinite-valued Lukasiewicz logic over one variable can be polyomially translated into regular signed literal

Some authors used a weak many-valued semantics to lower the complexity of computing the consequence relation in knowledge representation tasks, see [115]. An overview of results along this line is [30].

As previously said MDDs represent discrete functions. Space complexity of various kinds of MDDs is discussed, for example, in [130] where further pointers to the literature can be found. In fact every kind of MDD has exponential worst-case space complexity. Indeed, space complexity is sacrificed for efficient computation in practice. In [140] it is empirically argued that exponential growth rarely arises in real problems. The worst-case, best-case and relative space complexity of various kinds of signed DNF representations of discrete functions was recently investigated in [132].

6 Systems

MUltlog The system MUltlog [127, 155] is not a theorem prover, but a tool for developing and analyzing sequent and tableau rules (as displayed in equation (2) above) for any given finite-valued first order logic. Its output can be used to instantiate the systems discussed in the following paragraph.

$_3$TAP & Deep Thought The system $_3TAP$ [15], developed at University of Karlsruhe, is a tableau-based theorem prover for many-valued first order logic with sorts and (two-valued) equality; it is implemented in Prolog.

$_3TAP$ is able to handle full first order logics with any finite number of truth values. Hierarchical (tree-shaped) sorts attached to terms are supported. Efficient processing of the (two-valued) equality predicate is provided. Currently, versions for classical first order logic, for a certain three-valued first order logic [135] and for a seven-valued propositional logic [65] are specified. It is possible either to prove a theorem from a given set of assumptions or to try to check the consistency of an axiom set. Other highlights are methods for handling redundant axiom sets, utilization of pragmatic information contained in axioms to rearrange the search space, and a graphical user interface for control and output visualization.

The system Deep Thought [50] essentially is a re-implementation of $_3TAP$ in the language C. It is considerably faster, but implements only a subset of the former's features.

Milord Milord II [121] essentially consists of a system architecture together with an MVL language developed to facilitate the design of experts systems real world applications [40].

One of its main characteristics is its modularity [52, 3, 2]. A large knowledge base can be split into simpler parts by forming subroutines. From the user's view this is a major advantage, because experts with a minimal knowledge of programming expert systems can more quickly reach the necessary programming skill. From the technical side modularity notably simplifies the validation phase of the input data [154].

Each module contains a knowledge base expressed in a local MVL together with a local inference mechanism controlled by Horn-like meta rules. A mapping determines the global truth degree from the truth degrees computed by each module [52, 3, 2].

MDDs Implementations of many-valued DDs (cf. Section 3.2.3) are reported in [140, 131].

7 Selected applications of MVL Deduction

Some of the following entries are described in greater detail in [58, Chapter 7].

Coding Theory Mundici [105] showed that Lukasiewicz logic is a model of communication over a distorted channel with a bounded number of errors. MVL deduction might be used to optimize such communication.

Formal Verification The description of digital circuits at the switch level can be done naturally with many-valued logic [67]. Formal verification of such specifications leads to MVL deduction problems [28, 65].

In software verification and mathematical applications, several approaches employ three-valued logic to model partiality of programs and mathematical functions [20, 72, 74].

Natural Language Representation The idea to use deduction for solving the task of assigning meaning to natural language discourse is extensively studied in computational linguistics [70]. Various ambiguity phenomena in natural language are best modelled with many-valued logic [73].

Databases & Knowledge Representation Some applications of signed formula and annotated logic programming in databases and knowledge representation are [145, 96, 97] others can be found in the papers cited in Section 4.3.

Specifically, in [95, 97] examples are given that illustrate the suitability of lattice structures to model hybrid knowledge originating from different sources (agents) with distinct information. In [97] an annotated logic program is derived from a lattice.

A general framework for using many-valued logic in knowledge representation was given by [36]. An application of many-valued deduction in description logic is [143].

Fuzzy Control Most fuzzy controllers are based on fuzzy rules [23] in which the premise is a conjunction and each conjunct has the form "X_i is A_i", X_i denoting an input or state variable of the process to be controlled and A_i a fuzzy set. In [41] the different possible semantics for such rules is analysed in a lucid style. Many kinds of functions that determine the value of a rule's conclusion were proposed.

Fuzzy Controllers do not model the system to be controlled, but merely focus on the degree of error of a signal depending on which they generate the input signal control aiming at cancelling out the error.

Expert Systems The first expert systems, starting with Mycin [29], modelled uncertainty by Bayesian probabilities [116]. At present, uncertainty knowledge modeled by probabilities or in general, by numerical models is not longer considered. Instead, symbolic approaches based on truth value labels (see Section 2.2), which dramatically outperform numerical models in many aspects, are preferred. They allow both precision in modelling and efficient deduction as is witnessed by the shell Milord II [121], see also Section 6.

References

[1] G. Aguilera, I. P. de Guzmán, and M. Ojeda. Increasing the efficiency of automated theorem proving. *Journal of Applied Non-Classical Logics*, 5(1):9–29, 1995.

[2] J. Agustí-Cullell, F. Esteva, P. García, L. Godó, R. López de Mantaras, and C. Sierra. Local multi-valued logics in modular expert systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:303–321, 1994.

[3] J. Agustí-Cullell, F. Esteva, P. García, L. Godó, and C. Sierra. Combining multiple-valued logics in modular expert systems. In D'Ambrosio, Bruce D., Smets, Philippe, and P. P. Bonissone, editors, *Proc. 7th Conference on Uncertainty in Artificial Intelligence*, pages 17–25, San Mateo/CA, July 1991. Morgan Kaufmann.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, Reading/MA, 1975.

[5] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 27(6):509–516, June 1978.

[6] A. R. Anderson and N. D. Belnap Jr. *Entailment: The Logic of Relevance and Necessity*, volume 1. Princeton University Press, 1975.

[7] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, Mar. 1979.

[8] G. Ausiello and R. Giaccio. On-line algorithms for satisfiability problems with uncertainty. *Theoretical Computer Science*, 171(1–2):3–24, Jan. 1997.

[9] A. Avron. Natural 3-valued logics—characterization and proof theory. *Journal of Symbolic Logic*, 56(1):276–294, Mar. 1991.

[10] M. Baaz and C. G. Fermüller. Resolution for many-valued logics. In A. Voronkov, editor, *Proc. Logic Programming and Automated Reasoning LPAR'92*, LNAI 624, pages 107–118. Springer-Verlag, 1992.

[11] M. Baaz and C. G. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation*, 19(4):353–391, Apr. 1995.

[12] 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, Los Gatos/CA, USA*, pages 208–213. IEEE Press, Los Alamitos, 1993.

[13] M. Baaz and R. Zach. Note on calculi for a three-valued logic for logic programming. *Bulletin of the EATCS*, 48:157–164, Oct. 1992.

[14] B. Beckert, R. Hähnle, and G. Escalada-Imaz. Simplification of many-valued logic formulas using anti-links. Interner Bericht 11/97, Universität Karlsruhe, Fakultät für Informatik, May 1997.

[15] B. Beckert, R. Hähnle, P. Oel, and M. Sulzmann. The tableau-based theorem prover ${}_3T^AP$, version 4.0. In M. McRobbie and J. Slaney, editors, *Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA*, volume 1104 of *LNCS*, pages 303–307. Springer-Verlag, 1996.

[16] C. Bell, A. Nerode, R. Ng, and V. Subrahmanian. Mixed integer programming methods for computing nonmonotonic deductive databases. *Journal of the ACM*, 41(6):1178–1215, Nov. 1994.

[17] N. D. Belnap Jr. A useful four-valued logic. In J. M. Dunn and G. Epstein, editors, *Modern uses of multiple-valued logic*, pages 8–37. Reidel, Dordrecht, 1977.

[18] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. In K. V. Nori, editor, *Proceedings of the 7th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 287 of *LNCS*, pages 340–360, Pune, India, Dec. 1987. Springer-Verlag.

[19] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68(2):135–154, 1989.

[20] A. Blikle. Three-valued predicates for software specification and validation. *Fundamenta Informaticae*, XIV:387–410, 1991.

[21] A. Bloesch. Tableau style proof systems for various many-valued logics. Technical Report 94-18, Software Verification Research Center, Dept. of Computer Science, University of Queensland, Apr. 1994.

[22] L. Bolc and P. Borowik. *Many-Valued Logics. 1: Theoretical Foundations.* Springer-Verlag, 1992.

[23] P. P. Bonissone. Soft computing: the convergence of emerging reasoning technologies. *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, 1(1):6–18, Apr. 1997.

[24] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a BDD package. In *Proc. 27^{th} ACM/IEEE Design Automation Conference*, pages 40–45. IEEE Press, Los Alamitos, 1990.

[25] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer, Boston, 1984.

[26] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[27] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, Sept. 1992.

[28] R. E. Bryant and C.-J. H. Seger. Formal verification of digital circuits using symbolic ternary system models. In E. M. Clarke and R. P. Kurshan, editors, *Computer-Aided Verification: Proc. of the 2nd International Conference CAV'90*, LNCS 531, pages 33–43. Springer-Verlag, 1991.

[29] B. G. Buchanan and E. H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading/MA, 1984.

[30] M. Cadoli and M. Schaerf. On the complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, 18(1):29–50, 1997.

[31] R. Caferra and N. Zabel. An application of many-valued logic to decide propositional $\mathbf{S_5}$ formulae: a strategy designed for a parameterized tableaux-based theorem prover. In *Proc. AIMSA'90, Artificial Intelligence—Methodology Systems Application*, pages 23–32, 1990.

[32] J. Calmet, J. J. Lu, M. Rodriguez, and J. Schü. Signed formula logic programming: Operational semantics and applications. In *Proc of the 9th International Symposium on Methodologies for Intelligent Systems, Zakopane, Poland*, volume 1079 of *LNCS*, pages 202–211. Springer-Verlag, 1996.

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

[34] B. Chaib-Draa, B. Moulin, R. Mandiau, and P. Millot. Trends in distributed artificial intelligence. *Artificial Intelligence Review*, 1(6):35–66, 1992.

[35] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[36] C. G. de Bessonet. *A Many-Valued Approach to Deduction and Reasoning for Artificial Intelligence*. Kluwer Academic Publishers, 1991.

[37] P. Doherty. Preliminary report: NM3 — a three-valued non-monotonic formalism. In Z. Ras, M. Zemankova, and M. Emrich, editors, *Proc. of 5th Int. Symposium on Methodologies for Intelligent Systems, Knoxville, TN*, pages 498–505. North-Holland, 1990.

[38] P. Doherty. A constraint-based approach to proof procedures for multi-valued logics. In *First World Conference on the Fundamentals of Artificial Intelligence WOCFAI-91, Paris*, 1991.

[39] M. Domingo. Towards a knowledge level analysis of classification in biological domains. In *Proceedings of the IMACS International Workshop on Qualitative Reasoning and Decision Technologies QUARDET'93*, pages 535–544. CIMNE, 1993.

[40] M. Domingo. *An Expert System Architecture for Taxonomic Domains. An Application in Porifera: The Development of Spongia*, volume 4 of *Monografies del IIIA*. IIIA–CSIC, Artificial Intelligence Research Institute of the Spanish Scientific Research Council, 1996.

[41] D. Dubois and H. Prade. What are fuzzy rules and how to use them. *Fuzzy Sets and Systems*, 84(2):169–189, Dec. 1996.

[42] G. Escalada-Imaz and F. Manyà. The satisfiability problem for multiple-valued Horn formulæ. In *Proc. International Symposium on Multiple-Valued Logics, ISMVL'94, Boston/MA, USA*, pages 250–256. IEEE Press, Los Alamitos, 1994.

[43] G. Escalada-Imaz and F. Manyà. Efficient interpretation of propositional multivalued logic programs. In B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, editors, *Advances in Intelligent Computing. IPMU '94, 5th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, France*, volume 945 of *LNCS*, pages 428–439. Springer-Verlag, 1995.

[44] H. Eveking. *Verifikation digitaler Systeme: eine Einführung in den Entwurf korrekter digitaler Systeme.* LMI. Teubner, Stuttgart, 1991.

[45] M. C. Fitting. Many-valued modal logics. *Fundamenta Informaticae*, XV:235–254, 1991.

[46] M. C. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, XVII:55–74, 1992.

[47] M. C. Fitting. Many-valued non-monotonic modal logics. In A. Nerode and M. Taitslin, editors, *Proceedings of Logical Foundations of Computer Science, Tver*, volume 620 of *LNCS*, pages 139–150. Springer-Verlag, July 1992.

[48] M. C. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, New York, second edition, 1996.

[49] D. M. Gabbay. *Labelled Deductive Systems*, volume 1. Oxford University Press, 1996.

[50] S. Gerberding. DT—an automated theorem prover for multiple-valued first-order predicate logics. In *Proc. 26th International Symposium on Multiple-Valued Logics, Santiago de Compostela, Spain*, pages 284–289. IEEE Press, Los Alamitos, May 1996.

[51] M. L. Ginsberg. Multi-valued logics. *Computational Intelligence*, 4(3), 1988.

[52] L. Godo, R. López de Mántaras, C. Sierra, and A. Verdaguer. MILORD: The architecture and management of linguistically expressed uncertainty. *International Journal of Intelligent Systems*, 4:471–501, 1989.

[53] S. Gottwald. *Mehrwertige Logik. Eine Einführung in Theorie und Anwendungen.* Akademie-Verlag Berlin, 1989.

[54] S. Gottwald. *Fuzzy Sets and Fuzzy Logic.* Vieweg, Braunschweig, 1993.

[55] R. Hähnle. Towards an efficient tableau proof procedure for multiple-valued logics. In E. Börger, H. Kleine Büning, M. M. Richter, and W. Schönfeld, editors, *Selected Papers from Computer Science Logic, CSL'90, Heidelberg, Germany*, volume 533 of *LNCS*, pages 248–260. Springer-Verlag, 1991.

[56] R. Hähnle. Uniform notation of tableaux rules for multiple-valued logics. In *Proc. International Symposium on Multiple-Valued Logic, Victoria*, pages 238–245. IEEE Press, Los Alamitos, 1991.

[57] R. Hähnle. Short normal forms for arbitrary finitely-valued logics. In *Proceedings ISMIS'93, Trondheim, Norway*, volume 689 of *LNCS*, pages 49–58. Springer-Verlag, 1993.

[58] R. Hähnle. *Automated Deduction in Multiple-Valued Logics*, volume 10 of *International Series of Monographs on Computer Science.* Oxford University Press, 1994.

[59] R. Hähnle. Many-valued logic and mixed integer programming. *Annals of Mathematics and Artificial Intelligence*, 12(3,4):231–264, Dec. 1994.

[60] R. Hähnle. Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation*, 4(6):905–927, 1994.

[61] R. Hähnle. Commodious axiomatization of quantifiers in multiple-valued logic. In *Proc. 26th International Symposium on Multiple-Valued Logics, Santiago de Compostela, Spain*, pages 118–123. IEEE Press, Los Alamitos, May 1996.

[62] R. Hähnle. Exploiting data dependencies in many-valued logics. *Journal of Applied Non-Classical Logics*, 6(1):49–69, 1996.

[63] R. Hähnle. Proof theory of many-valued logic—linear optimization—logic design: Connections and interactions. *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, 1997, to appear.

[64] R. Hähnle, R. Hasegawa, and Y. Shirai. Model generation theorem proving with interval constraints. Interner Bericht 45/95, University of Karlsruhe, Dept. of Computer Science, 1995.

[65] R. Hähnle and W. Kernig. Verification of switch level designs with many-valued logic. In A. Voronkov, editor, *Proc. LPAR'93, St. Petersburg, Russia*, volume 698 of *LNCS*, pages 158–169. Springer-Verlag, 1993.

[66] P. Hájek. Fuzzy logic from the logical point of view. In *Proceedings SOFSEM'95*, LNCS. Springer-Verlag, 1995.

[67] J. P. Hayes. Pseudo-Boolean logic circuits. *IEEE Transactions on Computers*, C-35(7):602–612, July 1986.

[68] B. Hösli. *Robuste Logik*. PhD thesis, Eidgenössische Technische Hochschule Zürich, 1993.

[69] J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19 & 20:503–582, May 1994.

[70] H. Kamp and U. Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in linguistics and philosophy*. Kluwer, Dordrecht, 1993.

[71] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM*, 32(4):762–773, Oct. 1985.

[72] M. Kerber and M. Kohlhase. A mechanization of strong Kleene logic for partial functions. In A. Bundy, editor, *Proc. 12th International Conference on Automated Deduction, Nancy/France*, volume 814 of *LNCS*, pages 371–385. Springer-Verlag, 1994.

[73] M. Kerber and M. Kohlhase. A resolution calculus for presuppositions. In *Proc. 12th European Conference on Artificial Intelligence, ECAI-96*, pages 375–379. John Wiley & Sons, 1996.

[74] M. Kerber and M. Kohlhase. A tableau calculus for partial functions. In *Collegium Logicum. Annals of the Kurt-Gödel-Society*, volume 2, pages 21–49. Springer-Verlag, Wien New York, 1996.

[75] M. Kifer and E. L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 9(2):179–215, Oct. 1992.

[76] M. Kifer and E. L. Lozinskij. RI: A logic for reasoning with inconsistency. In *Proc. Logic in Computer Science LICS*, pages 253–262. IEEE Press, Los Alamitos, 1989.

[77] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Jornal of Logic Programming*, 12:335–367, 1992.

[78] V. G. Kirin. On the polynomial representation of operators in the n-valued propositional calculus (in Serbocroatian). *Glasnik Mat.-Fiz. Astronom. Društvo Mat. Fiz. Hravtske Ser. II*, 18:3–12, 1963. Reviewed in MR 29 (1965) p. 420.

[79] V. G. Kirin. Gentzen's method of the many-valued propositional calculi. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 12:317–332, 1966.

[80] S. Kleene. On a notation for ordinal numbers. *Journal of Symbolic Logic*, 3:150–155, 1938.

[81] L. V. Lakshmanan and F. Sadri. Modeling uncertainty in deductive databases. In D. Karagiannis, editor, *Proc. Int. Conf. on Database and Expert Systems Applications, DEXA'94, Athens, Greece*, volume 856 of *LNCS*, pages 724–733, 1994.

[82] S. M. Leach and J. J. Lu. Query processing in annotationed logic programming: Theory and implementation. *Journal of Intelligent Information Systems*, 6(1):33–58, Jan. 1996.

[83] R. C. T. Lee. Fuzzy logic and the resolution principle. *Journal of the ACM*, 19(1):109–119, January 1972.

[84] R. C. T. Lee and C.-L. Chang. Some properties of fuzzy logic. *Information and Control*, 19(5):417–431, December 1971.

[85] S. Lehmke. On resolution-based derivation in 'bold' fuzzy logic with weighted expressions. Forschungsbericht 545, Universität Dortmund, Fachbereich Informatik, Aug. 1994.

[86] S. Lehmke. A resolution-based axiomatisation of 'bold' propositional fuzzy logic. In D. Dubois, E. P. Klement, and H. Prade, editors, *Linz'96: Fuzzy Sets, Logics, and Artificial Intelligence. Abstracts*, pages 115–119, Fuzzy Logic Laboratorium Linz-Hagenberg, Austria, Feb. 1996.

[87] J. J. Lu. Logic programming with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[88] J. J. Lu, L. J. Henschen, V. S. Subrahmanian, and N. C. A. da Costa. Reasoning in paraconsistent logics. In R. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 181–210. Kluwer, 1991.

[89] J. J. Lu, N. V. Murray, and E. Rosenthal. Signed formulas and annotated logics. In *Proc. 23rd International Symposium on Multiple-Valued Logics*, pages 48–53. IEEE Press, Los Alamitos, 1993.

[90] J. J. Lu, N. V. Murray, and E. Rosenthal. A framework for automated reasoning in multiple-valued logics. *Journal of Automated Reasoning*, to appear.

[91] J. J. Lu and E. Rosenthal. Logic-based deductive reasoning in AI systems. In A. B. Tucker, editor, *The Computer Science And Engineering Handbook*, chapter 29, pages 654–657. CRC Press, 1997.

[92] J. Łukasiewicz. Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls. In K. Berka and L. Kreiser, editors, *Logik-Texte. Kommentierte Auswahl zur Geschichte der modernen Logik*, pages 135–150. Akademie-Verlag, Berlin, 1986.

[93] G. Malinowski. *Many-Valued Logics*, volume 25 of *Oxford Logic Guides*. Oxford University Press, 1993.

[94] F. Manyà. *Proof Procedures for Multiple-Valued Propositional Logics*. PhD thesis, Facultat de Ciències, Universitat Autònoma de Barcelona, 1996.

[95] B. Messing. Knowledge representation in many-valued horn clauses. In *Proceedings of the 6th Conference of the Spanish Association for Artificial Intelligence, Alicante*, pages 83–92. Asociacion Española para la Inteligencia Artificial, AEPIA, Sept. 1995.

[96] B. Messing. *Darstellung und Integration von Wissen in verbandsbasierten signierten Logikprogrammen*. PhD thesis, Fakultät für Wirtschaftswissenschaften, Universität Karlsruhe, July 1996. diski 132, infix Verlag, Sankt Augustin.

[97] B. Messing. Combining knowledge with many-valued logics. *Data & Knowledge Engineering*, to appear, 1997. Special Issue on Distributed Expertise.

[98] B. Messing and P. Stackelberg. Regular signed resolution applied to annotated logic programs. Poster abstract. In J. Lloyd, editor, *Proceedings of the International Logic Programming Conference, Portland/OR*, page 268. MIT Press, 1995.

[99] P. Miglioli, U. Moscato, and M. Ornaghi. An improved refutation system for intuitionistic predicate logic. *Journal of Automated Reasoning*, 13(3):361–374, 1994.

[100] P. Miglioli, U. Moscato, and M. Ornaghi. Refutation systems for propositional modal logics. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Proc. 4th Workshop on Deduction with Tableaux and Related Methods, St. Goar, Germany*, volume 918 of *LNCS*, pages 95–105. Springer-Verlag, 1995.

[101] S. Minato. *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer, Norwell/MA, USA, 1996.

[102] C. G. Morgan. A resolution principle for a class of many-valued logics. *Logique et Analyse*, 19(74–75–76):311–339, April 1976.

[103] B. Moulin and B. Chaib-draa. An overview of distributed artificial intelligence. In G. O'Hare and N. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 1. John Wiley and Sons, 1996.

[104] D. Mundici. Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science*, 52:145–153, 1987.

[105] D. Mundici. The complexity of adaptive error-correcting codes. In *Proceedings Workshop Computer Science Logic 90, Heidelberg*, LNCS 533, pages 300–307. Springer-Verlag, 1990.

[106] D. Mundici and N. Olivetti. Resolution and model building in the infinite-valued calculus of łukasiewicz. Unpublished Draft, Dec. 1996.

[107] T. Murata, V. S. Subrahmanian, and T. Wakayama. A Petri net model for reasoning in the presence of inconsistency. *IEEE Transactions on Knowledge and Data Engineering*, 3(3):281–292, Sept. 1991.

[108] N. V. Murray and E. Rosenthal. Resolution and path-dissolution in multiple-valued logics. In *Proceedings International Symposium on Methodologies for Intelligent Systems, Charlotte*, LNAI. Springer-Verlag, 1991.

[109] N. V. Murray and E. Rosenthal. Dissolution: Making paths vanish. *Journal of the ACM*, 3(40):504–535, 1993.

[110] N. V. Murray and E. Rosenthal. Signed formulas: A liftable meta logic for multiple-valued logics. In *Proceedings ISMIS'93, Trondheim, Norway*, LNCS 689, pages 275–284. Springer-Verlag, 1993.

[111] N. V. Murray and E. Rosenthal. Adapting classical inference techniques to multiple-valued logics using signed formulas. *Fundamenta Informaticae*, 21(3):237–253, 1994.

[112] H. Nowana. Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244, 1995.

[113] E. Orłowska. The resolution principle for ω^+-valued logic. *Fundamenta Informaticae*, II(1):1–15, 1978.

[114] E. Orłowska. Many-valuedness and uncertainty. In *Proc. 27th International Symposium on Multiple-Valued Logic, Nova Scotia, Canada*, pages 153–160. IEEE Press, Los Alamitos, May 1997.

[115] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. *Journal of Automated Reasoning*, 6:361–388, 1990.

[116] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, revised second edition, 1994.

[117] J. Pfalzgraf. On mathematical modelling in robotics. In J. Calmet and J. A. Campbell, editors, *Proc. International Conference on AI and Symbolic Mathematical Computation 1992, Karlsruhe*, number 737 in LNCS, pages 116–132. Springer-Verlag, 1993.

[118] J. Pfalzgraf, U. C. Sigmund, and K. Stokkermans. Towards a general approach for modeling actions and change in cooperating agents scenarios. *Journal of the Interest Group in Pure and Applied Logics*, 4(3):445–472, 1996.

[119] J. Posegga. *Deduktion mit Shannongraphen für Prädikatenlogik erster Stufe*. PhD thesis, University of Karlsruhe, 1993. diski 51, infix Verlag.

[120] J. Posegga and P. H. Schmitt. Deduction with first-order Shannon graphs. *Journal of Logic and Computation*, 5(6):697–729, 1995.

[121] J. Puyol-Gruart. *MILORD II: A Language for Knowledge-Based Systems*, volume 1 of *Monografies del IIIA*. IIIA–CSIC, Artificial Intelligence Research Institute of the Spanish Scientific Research Council, 1996.

[122] A. Ramesh and N. Murray. Computing prime implicants/implicates for regular logics. In *Proc. 24th International Symposium on Multiple-Valued Logic, Boston/MA, USA*, pages 115–123. IEEE Press, Los Alamitos, May 1994.

[123] A. G. Ramesh. *Some Applications of Non-Clausal Deduction*. PhD thesis, Department of Computer Science, State University of New York at Albany, 1995.

[124] N. Rescher. *Many-Valued Logic*. McGraw-Hill, New York, 1969.

[125] J. B. Rosser and A. R. Turquette. *Many-Valued Logics*. North-Holland, Amsterdam, 1952.

[126] G. Rousseau. Sequents in many valued logic I. *Fundamenta Mathematicæ*, LX:23–33, 1967.

[127] G. Salzer. MUltlog: an expert system for multiple-valued logics. In *Collegium Logicum. Annals of the Kurt-Gödel-Society*, volume 2. Springer-Verlag, Wien New York, 1996.

[128] G. Salzer. Optimal axiomatizations for multiple-valued operators and quantifiers based on semilattices. In M. McRobbie and J. Slaney, editors, *Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA*, volume 1104 of *LNCS*, pages 688–702. Springer-Verlag, 1996.

[129] T. Sasao, editor. *Logic Synthesis and Optimization.* Kluwer, Norwell/MA, USA, 1993.

[130] T. Sasao. Logic synthesis with EXOR gates. In T. Sasao, editor, *Logic Synthesis and Optimization*, chapter 12, pages 259–286. Kluwer, Norwell/MA, USA, 1993.

[131] T. Sasao. Ternary decision diagrams and their applications. In T. Sasao and M. Fujita, editors, *Representations of Discrete Functions*, chapter 12, pages 269–292. Kluwer, Norwell/MA, USA, 1996.

[132] T. Sasao and J. T. Butler. Comparison of the worst and best sum-of-products expressions for multiple-valued functions. In *Proc. 27th International Symposium on Multiple-Valued Logic, Nova Scotia, Canada*, pages 55–60. IEEE Press, Los Alamitos, May 1997.

[133] T. Sasao and M. Fujita, editors. *Representations of Discrete Functions.* Kluwer Academic Publishers, Boston, 1996.

[134] B. Scarpellini. Die Nichtaxiomatisierbarkeit des unendlichwertigen Prädikatenkalküls von Łukasiewicz. *Journal of Symbolic Logic*, 27(2):159–170, June 1962.

[135] P. H. Schmitt. Computational aspects of three-valued logic. In J. H. Siekmann, editor, *Proc. 8th International Conference on Automated Deduction*, LNCS, pages 190–198. Springer-Verlag, 1986.

[136] P. H. Schmitt. Perspectives in multi-valued logic. In R. Studer, editor, *Proceedings International Scientific Symposium on Natural Language and Logic, Hamburg*, LNCS 459, pages 206–220. Springer-Verlag, 1989.

[137] K. Schröter. Methoden zur Axiomatisierung beliebiger Aussagen- und Prädikatenkalküle. *Zeitschrift für math. Logik und Grundlagen der Mathematik*, 1:241–251, 1955.

[138] J. Siekmann and G. Wrightson, editors. *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970*, volume 2. Springer-Verlag, 1983.

[139] V. Sofronie-Stokkermans. *Fibered Structures and Applications to Automated Theorem Proving in Certain Classes of Finitely-Valued Logics and to Modeling Interacting Systems.* PhD thesis, Johannes Kepler Universität Linz, Forschungsinstitut für symbolisches Rechnen, Mar. 1997.

[140] A. Srinivasan, T. Kam, S. Malik, and R. E. Brayton. Algorithms for discrete function manipulation. In *Proc. IEEE International Conference on CAD, Santa Clara/CA, USA*, pages 92–95. IEEE Press, Los Alamitos, Nov. 1990.

[141] Z. Stachniak. The resolution rule: An algebraic perspective. In *Proc. of Algebraic Logic and Universal Algebra in Computer Science Conf.*, pages 227–242. Springer LNCS 425, Heidelberg, 1988.

[142] Z. Stachniak. *Resolution Proof Systems: an Algebraic Theory.* Kluwer, Dordecht, 1996.

[143] U. Straccia. A sequent calculus for reasoning in four-valued description logics. In D. Galmiche, editor, *Proc. International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Pont-à-Mousson, France*, volume 1227 of *LNCS*, pages 343–357. Springer-Verlag, 1997.

[144] J. Strother Moore. Introduction to the OBDD algorithm for the ATP community. *Journal of Automated Reasoning*, 12(1):33–45, 1994.

[145] V. S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database Systems*, 19(2):291–331, June 1994.

[146] W. Suchoń. La méthode de Smullyan de construire le calcul n-valent de Łukasiewicz avec implication et négation. *Reports on Mathematical Logic, Universities of Cracow and Katowice*, 2:37–42, 1974.

[147] S. J. Surma. An algorithm for axiomatizing every finite logic. In D. C. Rine, editor, *Computer Science and Multiple-Valued Logics*, pages 143–149. North-Holland, Amsterdam, second edition, 1984. Selected Papers from the International Symposium on Multiple-Valued Logics 1974.

[148] M. Takahashi. Many-valued logics of extended Gentzen style I. *Science Reports of the Tokyo Kyoiku Daigaku, Section A*, 9(231):95–116, 1967.

[149] H. Thiele. On T-quantifiers and S-quantifiers. In *Proc. 24th International Symposium on Multiple-Valued Logics, Boston, USA*, pages 264–269. IEEE Press, Los Alamitos, 1994.

[150] V. Torra. Negation functions based semantics for ordered linguistic labels. *International Journal of Intelligent Systems*, 11(11):975–988, Nov. 1996.

[151] G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 1970. Reprinted in [138].

[152] J. D. Ullman. *Principles of Database and Knowledge-Bade Systems. Volume I: Classical Database Systems.* Computer Science Press, 1988.

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

[154] A. Verdaguer, A. Patak, J. Sancho, C. Sierra, and F. Sanz. Validation of the medical expert system PNEUMON–IA. *Computers and Biomedical Research*, 25(6):511–526, 1992.

[155] Vienna Group for Multiple Valued Logics. MUltlog 1.0: Towards an expert system for many-valued logics. In M. McRobbie and J. Slaney, editors, *Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA*, volume 1104 of *LNCS*, pages 226–230. Springer-Verlag, 1996.

[156] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[157] N. Zabel. *Nouvelles Techniques de Déduction Automatique en Logiques Polyvalentes Finies et Infinies du Premier Ordre.* PhD thesis, Institut National Polytechnique de Grenoble, Apr. 1993.

[158] R. Zach. Proof theory of finite-valued logics. Master's thesis, Institut für Algebra und Diskrete Mathematik, TU Wien, Sept. 1993. Available as Technical Report TUW-E185.2-Z.1-93.