MUltlog 1.0: Towards an Expert System for Many-valued Logics* **

Vienna Group for Multiple-valued Logics***

Technische Universität Wien, Austria

Abstract. MUltlog is a system which takes as input the specification of a finitely-valued first-order logic and produces a sequent calculus, a natural deduction system, and a calculus for transforming a many-valued formula to clauses suitable for many-valued resolution. All generated rules are optimized regarding their branching degree. The output is in the form of a scientific paper, written in LaTeX.

1 Introduction

Many-valued logics are a generalization of classical logic introduced in the 1920's independently by Łukasiewicz and Post. For a long time they were mainly of interest to logicians, who among other problems investigated the axiomatization of these logics, concentrating primarily on Hilbert-style calculi. Schröter [12] and Rousseau [10] gave, for the first time, sequent calculi for finitely-valued logics; they showed that the rules can be generated systematically from the truth tables of operators and quantifiers. Similar methods can be used to construct natural deduction systems [3], clause formation rules [1], and tableaux [6] in a fully automatic manner.

Within the last decade many-valued logics have also attracted attention in the computer science community. They were discovered as an appropriate tool for hard- and software verification, artificial intelligence, natural language processing, and several other fields (see [7] for a comprehensive survey). This brought about the necessity for automatizing deduction in these logics. It turned out that the axiomatizations mentioned above were not really practicable, as they produce rules with a maximal branching degree leading to an exploding proof length. Only recently investigations started on minimizing the branching degree [1, 7, 9, 11, 13].

One of the main activities of the Vienna Group for Multiple-valued Logics (VGML) is to exploit the algorithmic content of the work done in many-valued

* Supported by FWF grant P10282-MAT.

** Appeared in: *Proc. 13th Int. Conf. on Automated Deduction (CADE'96)*, LNCS (LNAI) 1104, pp. 226–230. Springer, 1996.

*** Matthias Baaz, Christian G. Fermüller, Gernot Salzer, and Richard Zach. Technische Universität Wien, A-1040 Vienna, Austria. E-mail: vgml@logic.tuwien.ac.at

logics by implementing it in the MUltlog system, with the ultimate goal of creating an expert system for many-valued logics. This paper describes the current state of the MUltlog system as well as future developments.[1]

2 The many faces of MUltlog

A user of MUltlog has the choice between different interfaces. One is written in Tcl/Tk (by Andreas Leitgeb) and runs under Unix and X-Windows.[2] A second one (by Wolfram Nix) is written in C for PCs under DOS. A third one (by Markus Schranz) is written in HTML and Perl, providing access to MUltlog via WWW: the user fills in some HTML forms and gets the output of MUltlog as a Postscript file, obviating the need to install it on her own machine. All three interfaces communicate with MUltlog by an ordinary text file, which can be viewed as a fourth interface. The next section discusses an example for such a text file.

3 Specifying a many-valued logic

A many-valued logic is characterized by the truth functions associated with its propositional operators and quantifiers. More precisely, if W denotes the set of truth values, then a total function $\tilde{\theta}\colon W^n \mapsto W$ is associated with each n-ary operator θ, and a total function $\tilde{\lambda}\colon (2^W - \{\emptyset\}) \mapsto W$ with each quantifier λ.[3]

For finitely-valued logics, $\tilde{\theta}$ and $\tilde{\lambda}$ can be specified by finite tables. The size of quantifier tables, however, grows exponentially with the number of truth values. Fortunately, many operators and quantifiers are defined implicitly as greatest lower or least upper bounds with respect to some (semi-)lattice ordering on the truth values; conjunction and disjunction as well as universal and existential quantification fall into this class. For this reason MUltlog offers several possibilities for specifying operators and quantifiers.

Example 1. Consider the specification of the three-valued Gödel logic in the left part of Fig. 1. Negation ('`neg`') is specified as a mapping, giving for each argument the corresponding value of the function. Implication ('`imp`') can be given as a table since it is binary. The remaining operators and quantifiers refer either

[1] An early prototype of MUltlog was already presented in [2]. Unfortunately, this implementation turned out to be buggy, and never made it to an official release. The program died when the program author left the group. The current version of MUltlog was written from scratch.

[2] The developer of Tcl/Tk, John Ousterhout, is currently porting Tcl/Tk to the various window systems on PCs, i.e., in the future this interface will also be available on these platforms.

[3] Quantifiers defined this way are called *distribution quantifiers*. The intuitive meaning is that a quantified formula $(\lambda x)A(x)$ takes the value $\tilde{\lambda}(U)$ if the instances $A(d)$ take exactly the elements of U as their values. E.g., the universal quantifier in classical logic can be defined as $\tilde{\forall}(\{t\}) = t$ and $\tilde{\forall}(\{f\}) = \tilde{\forall}(\{t, f\}) = f$.

```
logic "G\""odel".
truth_values{f,u,t}.
designated_truth_values{t}.
operator(neg/1,
    mapping{(t):f, (u):u, (f):t}).
operator(imp/2, table[   f,u,t,
                       f, t,t,t,
                       u, f,t,t,
                       t, f,u,t ]).
ordering(linear, "f < u < t").
operator(and/2, inf(linear)).
operator(or /2, sup(linear)).
quantifier(all, induced_by and/2).
quantifier(ex , induced_by or /2).
```

Calculi for the Gödel Logic
M. Ultlog

Fig. 1. Input and output for the three-valued Gödel logic.

directly or indirectly to an ordering called '`linear`'. The operators '`and`' and '`or`' are the greatest lower and the least upper bound wrt. to this ordering, and the two quantifiers are the natural extensions of these two operators.

4 The MUltlog kernel

The kernel of MUltlog is written in Prolog[4]. Its main task is to compute a certain conjunctive normal form (CNF) for each combination of operators or quantifiers with truth values. Once given the CNF, all calculi can be obtained more or less by syntactic transformations. The problem is not to find any such CNFs: one particular kind can be immediately obtained from the definition of operators and quantifiers. However, as mentioned in the introduction, these CNFs are of a maximal branching degree and therefore do not lead to feasible deduction systems. MUltlog has to *optimize* the CNFs regarding the number of conjuncts. For operators and quantifiers referring to an ordering the matter is easy: provably optimal CNFs are obtained by instantiating a schema. For all other operators and quantifiers more complex computations are needed, which involve resolution and a special inference rule called combination (for a detailed description and correctness proofs of the employed algorithms see [11]).

Example 2. Consider the logic specified in Example 1. The optimal CNF for operator '`imp`' and truth value t is computed as

$$(\{f,u\}A \vee \{t\}B) \wedge (\{f\}A \vee \{u,t\}B)$$

[4] MUltlog has been tested with Paul Tarau's BinProlog and with SICStus Prolog, but should be portable to any standard Prolog.

which can be interpreted as saying that "$\texttt{imp}(A, B)$ takes the truth value t if A takes value f or u or B takes value t, and if A takes value f or B takes value u or t". The optimal CNF for the universal quantifier and truth value u is computed as

$$(\forall x)\{u, t\}A(x) \wedge (\exists x)\{u\}A(x) \ .$$

5 The output of MUltlog

The output consists of a style file containing LaTeX definitions specific to the input logic, which is included by a generic document when compiled with TeX. The style file is generated by DCGs (definite clause grammars) on the basis of the specification read by MUltlog and the minimized CNFs computed by MUltlog.

Example 3. The right part of Fig. 1 is an outline of the paper generated from the input in the left part.[5] As an example, the section on the sequent calculus contains introduction rules for every operator/quantifier and truth value, among them those generated from the CNFs in Example 2:

$$\frac{\Gamma, A^f, A^u, B^t \quad \Gamma, A^f, B^u, B^t}{\Gamma, (\texttt{imp}(A, B))^t}\ \texttt{imp}\colon t \qquad \frac{\Gamma, A(\alpha)^u, A(\alpha)^t \quad \Gamma, A(\tau)^u}{((\texttt{all}x)A(x))^u}\ \texttt{all}\colon u$$

6 Future Developments

The automatic derivation of optimized calculi is only a fraction of what can be automatized in many-valued logics. The ultimate goal is to develop an expert system for many-valued logics covering all of their mechanizable aspects. In particular, future versions of MUltlog will deal with the following topics.

Correctness and completeness proofs. The paper generated by MUltlog will contain proofs for the correctness and completeness of the computed calculi.

Inclusion of further calculi. MUltlog will compute optimized versions of other calculi, like tableau systems or negative variants of sequent calculus. This will require the computation of optimized DNFs. Another extension will consider variants of calculi based on truth value sets as signs [7].

Cut elimination. MUltlog will construct cut elimination algorithms for multiple-valued logics [4] and include a corresponding cut elimination theorem into its output.

Comparison to known logics. MUltlog will be augmented by a database about already known logics and their properties. Each new input logic will be compared against it, and appropriate comments and references will be added to the paper.

[5] The headlines in *italic typeface* are sections to be included in future versions of MUltlog.

Generation of machine-readable output. MUltlog will be linked to an automatic theorem prover, such that the clause formation rules for a particular logic are not just included into the paper, but also result in a theorem prover for that logic. The same holds for the other calculi.

Extension to infinitely-valued logics. In general this extension will not be possible since infinitely-valued logics show a behavior different from finitely-valued ones. However, it has been shown that finitely-valued logics can be used to approximate infinitely-valued logics [5]. Another approach could be to use mixed integer programming for certain logics [8].

7 Availability

MUltlog 1.0 is available via anonymous ftp from host `logic.tuwien.ac.at` in directory `pub/MUltlog`. For further informations see WWW-page

`http://logic.tuwien.ac.at/VGML/MUltlog.html`,

or send an e-mail to `MUltlog@logic.tuwien.ac.at`.

References

1. M. Baaz and C. G. Fermüller. Resolution-based theorem proving for many-valued logics. *J. Symbolic Computation*, 19:353–391, 1995.
2. 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 (LPAR'93)*, LNCS 698 (LNAI), pages 345–347. Springer, 1993.
3. 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 Logic*, pages 208–215. IEEE Computer Society Press, Los Alamitos, May 24–27 1993.
4. M. Baaz, C. G. Fermüller, and R. Zach. Elimination of cuts in first-order finite-valued logics. *J. Inform. Process. Cybernet. (EIK)*, 29(6):333–355, 1994.
5. M. Baaz and R. Zach. Approximatig propositional calculi by finite-valued logics. In *Proc. 24th International Symposium on Multiple-valued Logic*, pages 257–263. IEEE Press, Los Alamitos, May 25–27 1994.
6. W. A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *J. Symbolic Logic*, 52(2):473–493, 1987.
7. R. Hähnle. *Automated Deduction in Multiple-valued Logics.* Clarendon Press, Oxford, 1993.
8. R. Hähnle. Many-valued logic and mixed integer programming. *Annals of Mathematics and Artificial Intelligence*, 12(3,4):231–264, Dec. 1994.
9. R. Hähnle. Commodious axiomatization of quantifiers in multiple-valued logic. In *Proc. 26th International Symposium on Multiple-Valued Logics, Santiago de Compostela, Spain.* IEEE Press, Los Alamitos, May 1996.
10. G. Rousseau. Sequents in many valued logic I. *Fund. Math.*, 60:23–33, 1967.

11. G. Salzer. Optimal axiomatizations for multiple-valued operators and quantifiers based on semi-lattices. In *13th Int. Conf. on Automated Deduction (CADE'96)*, LNCS (LNAI). Springer, 1996.
12. K. Schröter. Methoden zur Axiomatisierung beliebiger Aussagen- und Prädikatenkalküle. *Z. Math. Logik Grundlag. Math.*, 1:241–251, 1955.
13. N. Zabel. *Nouvelles Techniques de Déduction Automatique en Logiques Polyvalentes Finies et Infinies du Premier Ordre*. PhD thesis, Institut National Polytechnique de Grenoble, 1993.