MUltlog and MUltseq Reanimated and Married *

M. Baaz[1] C.G. Fermüller[1] A. Gil[2] G. Salzer[1] N. Preining[1]

[1]Technische Universität Wien, Vienna, Austria
[2]Universitat Pompeu Fabra, Barcelona, Spain

1 Introduction

MUltlog is a logic engineering tool that produces descriptions of various sound and complete logical calculi for an arbitrary finite-valued first-order logic from a given specification of the semantics of such a logic (see [1]). MUltseq, on the other hand, is a simple, generic, sequent based theorem prover for propositional finite-valued logics (see [6]). From its very beginning, MUltseq was intended to be a 'companion' to MUltlog. So far, however, MUltseq does not directly use the representation of sequent rules as generated by MUltlog. Moreover (due to lack of funding, personnel and time), further development and maintenance of both system has been stalled for some time now. It is the purpose of this abstract to shortly describe the two systems and the current efforts to integrate them.

2 A short description of MUltlog

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.[1]

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 supports several possibilities for specifying operators and quantifiers.

The kernel of MUltlog is written in Prolog. 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

* Partially supported by the Austrian science foundation FWF, project P16539-N04.

[1] 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 $\tilde{\forall}(\{f\}) = \tilde{\forall}(\{t, f\}) = f$.

operators and quantifiers. However, these CNFs are of a maximal branching degree and therefore do not lead to feasible deduction systems. MUltlog computes CNFs that are *optimal* 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 [8]).

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.

Users of MUltlog can choose among different interfaces. One is written in Tcl/Tk and runs under Unix and X-Windows. A second one is written in C for PCs under DOS. A third one 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. Moreover there exists JMUltlog, a Java applet serving roughly the same purpose as the HTML/Perl interface.

3 A short description of MUltseq

In its core, MUltseq is a generic sequent prover for propositional finitely-valued logics. This means that it takes as input the rules of a many-valued sequent calculus as well as a many-sided sequent and searches – automatically or interactively – for a proof of the latter. For the sake of readability, the output of MUltseq is typeset as a LaTeX document.

Though the sequent rules can be entered by hand, MUltseq is primarily intended as a companion for MUltlog. Provided the input sequent calculus is sound and complete for the logic under consideration – which is always the case when the rules were computed by MUltlog – MUltseq serves as a decision procedure for the validity of *sequents* and *formulas*. More interestingly, MUltseq can also be used to decide the *consequence relations* associated with the logic and the sequent calculus. The problem of deciding whether a particular formula ϕ is true in all models satisfying a given set of formulas Δ, i.e., whether ϕ logically follows from Δ, can be reduced to the problem of proving that certain sequent that depends only on ϕ and Δ is true. Similarly, as a consequence of the *Deduction Detachment Theorem for many-valued sequents* [5, 7], the problem of finding a derivation of a sequent σ from hypotheses Σ can be reduced to proving a particular set of sequents.

From the algebraic point of view, it is an interesting problem to determine whether an *equation* or a *quasi-equation* is valid in a finite algebra. If we consider the algebra as a set of truth values and a collection of finitely-valued connectives, and use an appropriate translation of equations and quasi-equations to sequents, the problem again reduces to the provability of many-valued sequents [4].

The decision procedures implemented in `MUltseq` help to get a better intuition and understanding of some theoretical problems. For instance, it is known that each propositional logic between the implication-less fragment of Intuitionistic Propositional Calculus and Classical Propositional Calculus has an algebraic semantics. If we consider the algebraic semantics of all these logics, we obtain a denumerable chain which corresponds to the chain of all subvarieties of the variety of Pseudo-complemented Distributive Lattices [7]. Each of these subvarieties is generated by a finite algebra, so the study of the sequent calculi obtained by `MUltlog` for each of these algebras and the decision procedures in `MUltseq` might help to find algebraizable Gentzen systems for the original logics.

4 Availability

Further information on `MUltlog` as well as the latest version of the system (version 1.10, dated 11/07/2001) is available at

`http://www.logic.at/multlog` .

`MUltseq` is currently is at version 0.6 (dated 13/09/2002). It is available at

`http://www.logic.at/multseq` .

5 The marriage agenda

The input for `MUltseq`, i.e. the description of sequent rules for the introduction of connectives at the sequent-positions corresponding to the truth values, is currently prepared by hand. In principle, such a description could and should be extracted from the output of `MUltlog`. Moreover, the intended use of the systems is to investigate and compare the forms of logical rules that can be computed from truth tables and to check simple logical statements by using these rules. This calls for an explicit integration of `MUltlog` and `MUltseq`. The corresponding agenda is as follows:

1. Write a conversion program that takes the output of `MUltlog`, as described above, as input and generates the corresponding sequent rules in the format used for the input of `MUltseq`.
2. Prepare an integrated distribution package that contains properly updated versions of `MUltlog`, `MUltseq` and the conversion tool just described.
3. Design and maintain a joint internet page, that not only just refers to the already available seperate pages for the two systems, but describes and illustrates the intended use of the integrated system.

6 Future developments

Argueably, a happy marriage should result in common offspring. We list some goals for future developments of `MUltlog` and `MUltseq`; in particular ones that serve the aim of a better integration of the two systems.

- *First order theorem proving:* `MUltseq` should be extended to include the application of rules for distribution quantifiers as computed by `MUltlog`.
- *Model construction:* Augmentation of `MUltseq` with features for the explicit construction of (descriptions of) counter models for non-valid formulas and invalid statements involving different versions of consequence relations.
- *Extension to projective logics:* In [2] the systematic construction of special sequent calculi for projective logics, an extension of the class of finite valued logics, has been described. We plan to integrate these algorithms into `MUltlog` and, correspondingly, to enhance `MUltseq` to allow for the use of the resulting sequent calculi in proof search.
- *Cut elimination:* A future version of `MUltlog` should construct specifications of cut elimination algorithms for finite-valued logics as described in [3]. The corresponding cut-reduction operators should then be integrated into `MUltseq`, together with the possibility to apply appropriate cut rules, at least in an interactive fashion.

References

1. M. Baaz, C. G. Fermüller, G. Salzer, and R. Zach. MUltlog 1.0: Towards an expert system for many-valued logics. In M. A. McRobbie and J. K. Slaney, editors, *13th Int. Conf. on Automated Deduction (CADE'96)*, LNCS 1104 (LNAI), pp. 226–230. Springer-Verlag, 1996.
2. M. Baaz and C. G. Fermüller. Analytic Calculi for Projective Logics. In Neil V. Murray (Ed.), *Automated Reasoning with Analytic Tableaux and Related Methods*, TABLEAUX'99, Saratoga Springs, NY, USA, June 1999, LNAI 1617, Springer-Verlag, 1999, pp. 36–50.
3. M. Baaz, C. G. Fermüller, G. Salzer, and R. Zach. Elimination of Cuts in First-Order Finite-Valued Logics. *Journal of Information Processing and Cybernetics*, EIK 29 (1993) 6, pp. 333-355.
4. A.J. Gil, J. Rebagliato, and V. Verdú. A strong completeness theorem for the Gentzen systems associated with finite algebras. *Journal of Applied non-Classical Logics*, vol. 9-1:9–36, 1999.
5. A.J. Gil, A. Torrens, and V. Verdú. On Gentzen Systems Associated with the Finite Linear MV-algebras. *Journal of Logic and Computation*, 7:1–28, 1997.
6. A.J. Gil, G. Salzer. `MUltseq`: Sequents, Equations, and Beyond. Extended version of an abstract presented at the *Joint conference of the 5th Barcelona Logic Meeting and the 6th Kurt Gödel Colloquium*, June 1999; available at `http://www.logic.at/multseq`
7. J. Rebagliato and V. Verdú. Algebraizable Gentzen systems and the Deduction Theorem for Gentzen systems. Mathematics Preprint Series 175, Universitat de Barcelona, June 1995.

8. G. Salzer. Optimal axiomatizations for multiple-valued operators and quantifiers based on semi-lattices. In M. A. McRobbie and J. K. Slaney, editors, *13th Int. Conf. on Automated Deduction (CADE'96)*, LNCS 1104 (LNAI), pages 688–702. Springer-Verlag, 1996.
9. G. Salzer. Optimal Axiomatizations of Finitely-valued Logics. *Information and Computation*, 162:185–205, 2000.