Semantic Proof Search Methods for ALL* - a first approach -

Didier Galmiche & Jean-Yves Marion
CRIN-CNRS & INRIA Lorraine,
Campus Scientifque, BP 239,
F-54506 Vandœuvre-lès-Nancy, France,
e-mail: Didier.Galmiche{Jean-Yves.Marion}@loria.fr

1 Introduction

Recent works have been devoted to proof search in fragments of linear logic (LL) with resolution, sequent calculus [8, 9] or with tableau methods [6]. This paper deals with derivability in the additive fragment of linear logic without constants (called ALL) [10, 16]. The only connectives in formulae and sequents in ALL are the additive ones, i.e. & (conjunction) and $\oplus$ (disjunction), and the linear negation $^{\perp}$. In general, these connectives are interesting for various reasons [11, 14] but involve some problems due to the context dependency and it could be appropriate to integrate specific techniques for ALL into existing procedures for theorems proving in linear logic. Thus, we aim to study efficient (polynomial, if possible) proof search methods based on semantics. A first step in this investigation can consist in analyzing deduction in ALL from a semantical point of view before trying to propose mixed methods.

2 Provability in additive linear logic

ALL sequents are of the following form $E \vdash F$ since we have neither multiplicative connectives nor structural rules. The inference system for ALL is the following:

$$\frac{}{E \vdash E}\ Ax$$

$$\frac{E_i \vdash F}{E_1 \& E_2 \vdash F}\ \&_{L i_{i=0,1}} \qquad \frac{E \vdash F_1 \qquad E \vdash F_2}{E \vdash F_1 \& F_2}\ \&_R$$

$$\frac{E_1 \vdash F \qquad E_2 \vdash F}{E_1 \oplus E_2 \vdash F}\ \oplus_L \qquad \frac{E \vdash F_i}{E \vdash F_1 \oplus F_2}\ \oplus_{R i_{i=0,1}}$$

Here we restrict ourselves, without loss of generality, to sentences where negation is applied only to atoms because the negation can be moved at the atom level by a simple *polynomial-time* procedure. Let us also recall that the cut elimination theorem holds for ALL.

* Proceedings of the 4th Workshop on Theorem Proving with Analytic tableaux and Related methods, 1995, to appear

2.1 A naive procedure for ALL

The provability in ALL can be decided with the following map T from ALL sequents to boolean sentences (without variables). The map integrates a decision strategy for the ALL fragment [4].

$$
\begin{array}{lll}
T(A \vdash A) & \equiv 1 & \textit{if } A \textit{ is a literal} \\
T(A \vdash B) & \equiv 0 & \textit{if } A \neq B \textit{ are literals} \\
T(E \vdash F_1 \& F_2) & \equiv T(E \vdash F_1) \wedge T(E \vdash F_2) & \\
T(E_1 \oplus E_2 \vdash F) & \equiv T(E_1 \vdash F) \wedge T(E_2 \vdash F) & \\
T(E_1 \& E_2 \vdash F_1 \oplus F_2) & \equiv T(E_1 \vdash F_1 \oplus F_2) \vee T(E_2 \vdash F_1 \oplus F_2) \vee & \\
 & \quad T(E_1 \& E_2 \vdash F_1) \vee T(E_1 \& E_2 \vdash F_2) & \\
T(A \vdash F_1 \oplus F_2) & \equiv T(A \vdash F_1) \vee T(A \vdash F_2) & \textit{if } A \textit{ is a literal} \\
T(E_1 \& E_2 \vdash B) & \equiv T(E_1 \vdash B) \vee T(E_2 \vdash B) & \textit{if } B \textit{ is a literal}
\end{array}
$$

Lemma 1. *$T(E \vdash F)$ evaluates to 1 iff $E \vdash F$ is provable in ALL.*

Proof. by induction on the length of the derivations.

This lemma suggests that to solve the decision problem in ALL we only need to build T and then to evaluate the boolean formula. However this approach is not efficient because it induces too many redundancies. For example the sequent $HS_n^m \equiv A_1 \& \ldots \& A_n \vdash B_1 \oplus \ldots \oplus B_m$ is very hard to evaluate and more precisely

Proposition 2. *If we consider the sequent HS_n^m, then the size of $T(HS_n^m)$ is greater than $2^{n.m}$.*

Hence the naive recursive computation, defined by the T translation, runs in exponential time.

2.2 A feasible proof construction method

The aim of this part is to present a polynomial-time proof search procedure for ALL which drastically decreases the proof search complexity. Actually [4] characterized exactly the computational complexity of ALL

Theorem 3. *The decision problem for ALL is $\mathcal{NC}^1$-Complete under many-one, D-log-time reductions.*

The result stems on [2, 3] and is based on a game semantics which could be implemented on parallel machine. (See also [12, 13] for complexity results about another fragments). However, it seems interesting to have a feasible sequential procedure easily implementable as a module of a general theorem prover, to speed up the proof search process.

We now present a *polynomial time* algorithm to build a proof of $E \vdash F$ if there exist one. Firstly, we are defining a directed acyclic graph $G = (V, \mathcal{E})$ (i.e. G is a dag) where the set of nodes V is defined, thus

$$V = \{(U, V) \mid U \ (\textit{resp. } V) \textit{ is a subformula of } E \ (\textit{resp. } F)\}$$

and the the set of edges $\mathcal{E} \subset V \times V$ is such that there is an edge between the nodes (U, V) and (U', V') iff (U', V') is a direct subgoal given by the T translation. For example, there is an edge between the nodes $(U, V_1 \& V_2)$ and (U, V_1). In other words, G represents $T(E, F)$ like a circuit. Concerning such a graph, we have the following lemma

Lemma 4. *The number of nodes of the acyclic graph $G = (V, \mathcal{E})$, defined above, is bounded by a polynomial in the number of literals of E and F.*

Since G is a dag, it is stratified. The terminal nodes which are just pairs of literals are on the first level. Next the level k is provided by the immediate conclusions which come from level $< k$. For instance if (A, C) and (B, C) are nodes of level $< k$ so $(A \oplus B, C)$ is a node of level k if $A \oplus B$ is a subformula of E. Now the graph is evaluated level by level by a bottom-up procedure, like a boolean circuit [15]. First a terminal node is assigned to 1 if it denotes an axiom, i.e. it is a pair like (A, A), otherwise it is assigned to 0. Second values are computed level by level. In fact this procedure is based on dynamic programming and is polynomial in the number of vertices of G. Finally, by combining lemma 4 and this algorithm, we obtain

Theorem 5. *The decision problem for ALL can be solved in polynomial-time in the numbers of literals.*

Compared with the last approach, we do not prove twice the same statement. The proof search is represented by an acyclic graph which allows to share the previous result, see also [6, 17]. Roughly, we can say that a brute recursive method is memoryless whereas here we keep in mind previous established lemma. Let us remark that we can propose some optimizations as to deduce that (U, V) could be provable only by analyzing atoms or to recognize proof templates (modulo instantiation) that allow some reuses during the proof search.

3 A semantic proof search method for ALL

After a direct proof search in ALL we investigate phase semantics as a concrete way to decide whether or not a sequent is provable [6, 16]. Our motivation would be to arithmetize proofs in ALL [1, 14] to propose at least more efficient proof search methods.

3.1 An algebraic semantics for ALL

First of all, let us recall some basic definitions. An additive-structure is a 7-uplet $(\mathcal{D}, \prec, \sqcup, \sqcap, \sim, \mathbf{1}, \mathbf{0})$ satisfying:

1. $(\mathcal{D}, \prec, \sqcup, \sqcap)$ is a complete lattice.
2. $\sim$ is an involution on $\mathcal{D}$, $\sim\sim a = a$
3. $\mathbf{1}$ is the greatest element of $\mathcal{D}$ and $\mathbf{0}$ the least one, $\sim\mathbf{1} = \mathbf{0}$.

4. $\sim(a\sqcup b) = (\sim a)\sqcap(\sim b)$ and $\sim(a\sqcap b) = (\sim a)\sqcup(\sim b)$.

An interpretation consists of $(\mathcal{D},\prec,\sqcup,\sqcap,\sim,\mathbf{1},\mathbf{0})$ and of an assignment to each literal A of an element $\mathbf{A}$ of the domain $\mathcal{D}$. The value of E under an interpretation is noted $[\![E]\!]$ and is defined by

$$\begin{array}{lll} [\![A]\!] & = \mathbf{A} & \textit{if } A \textit{ is a literal} \\ [\![E_1\&E_2]\!] & = [\![E_1]\!]\sqcap[\![E_2]\!] & \\ [\![E_1 \oplus E_2]\!] & = [\![E_1]\!]\sqcup[\![E_2]\!] & \\ [\![E^\perp]\!] & = \sim[\![E]\!] & \end{array}$$

Since $\vdash$ behaves like a multiplicative connector, the interpretation of an additive sequent $E \vdash F$ is sensitive to interpretations of the multiplicative fragment. It bears stressing that we are going to stray from the usual algebraic semantics. Usually the interpretation of $\vdash$ is set on the definition of the binary operation $\bullet$:

$$[\![E \vdash F]\!] = \sim([\![E]\!]\bullet\sim[\![F]\!])$$

where $(\mathcal{D},\bullet)$ is a commutative monoid such that $a\bullet\sim a = \sim a\bullet a = \mathbf{0}$ and $\bullet$ is distributive over $\sqcup$. This interpretation gives no effective way to compute $(a\sqcap b)\bullet(c\sqcap d)$ because $\bullet$ is not distributive over $\sqcap$. However the T translation gives a rule which is $a\sqcap b\bullet c\sqcap d = (a\sqcap b\bullet c)\sqcap(a\sqcap b\bullet d)\sqcap(a\bullet c\sqcap d)\sqcap(b\bullet c\sqcap d)$. By thus setting the operational behaviour of $\bullet$, we integrate proof search rule at a semantic level and we still have

Theorem 6. *$A \vdash B$ is provable iff $\mathbf{1}\prec[\![A \vdash B]\!]$.*

3.2 A Tableaux method for ALL

Although the decision problem for ALL is carried out by former decision procedures, one might still be interested in a tableau or analytic method for ALL. Methods based on tableaux and analytic deduction are of interest to improve efficiency of proof search [6]. In the sequel we come out with such a method which is tied to the previous computational interpretation. The tableau method defines a *proof tableau* by the rules written below in sequent-like notation [5]. Here we assume that the negation is just on the atoms and that Γ, Δ are sets of ALL formulae.

$$\frac{\Gamma \vdash \Delta, F_1\&F_2}{\Gamma \vdash F_1 \quad | \quad \Gamma \vdash F_2} \qquad \frac{\Gamma, E_1 \oplus E_2 \vdash \Delta}{E_1 \vdash \Delta \quad | \quad E_2 \vdash \Delta}$$

$$\frac{\Gamma, E_1\&E_2 \vdash \Delta}{\Gamma, E_1, E_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, F_1 \oplus F_2}{\Gamma \vdash \Delta, F_1, F_2} \qquad \frac{\Gamma, A, \Gamma' \vdash \Delta, A, \Delta'}{\perp}$$

A branch is closed when it ends up with $\perp$. Then a proof tableau is closed when all branches are closed. One of the main point is that the sequent management of the context is performed in an additive way and not in a multiplicative way. Thus, a context is a set which intuitively caveats some weak structural rules, which express, for example, that $\Gamma \vdash E$ is equivalent to $\Gamma \vdash E \oplus E$. In other words, these rules get rid of redundancies in the context. Let us consider the following example

$$
\begin{array}{c|c}
\multicolumn{2}{c}{A\&B \vdash ((A \oplus C)\&(B \oplus C)) \oplus D} \\
\hline
\multicolumn{2}{c}{A, B \vdash ((A \oplus C)\&(B \oplus C)) \oplus D} \\
\hline
\multicolumn{2}{c}{A, B \vdash ((A \oplus C)\&(B \oplus C)), D} \\
\hline
A, B \vdash A \oplus C & A, B \vdash B \oplus C \\
\hline
A, B \vdash A, C & A, B \vdash B, C \\
\hline
\bot & \bot
\end{array}
$$

Finally, all branches are closed, which implies that the root sequent $A\&B \vdash ((A \oplus C)\&(B \oplus C)) \oplus D$ is provable. Roughly, the propositions in Γ and Δ on both sides of the tableau sequent $\Gamma \vdash \Delta$ correspond to non-deterministic choices. The rules, for $\&$ on the right-hand side or for $\oplus$ on the left-hand side, determine a proposition and cancel the others which means they are not invertible. The meaning of these sequents is given by

$$[\![E_1, \ldots, E_2 \vdash F_1, \ldots, F_m]\!] = [\![E_1]\!] \sqcap \ldots \sqcap [\![E_n]\!] \bullet \sim([\![F_1]\!] \sqcup \ldots \sqcup [\![F_m]\!])$$

and a branch is closed whenever a node $\Gamma \vdash \Delta$ verifies $[\![\Gamma \vdash \Delta]\!] \prec \mathbf{0}$.

Theorem 7. *$E \vdash F$ holds in ALL iff its proof tableau is closed*

Proof. Assume that $E \vdash F$ holds. We prove by structural induction that for all context Γ, Δ, the proof tableau $\Gamma, E \vdash \Delta, F$ is closed. Conversely, we assume that $\Gamma \vdash \Delta$ is a closed tableau. Again we prove by structural induction on the tableau that $C_1 \& \ldots \& C_p \vdash D_1 \oplus \ldots \oplus D_q$ holds where $\Gamma = \{C_1, \ldots, C_p\}$ and $\Delta = \{D_1, \ldots, D_q\}$.

We can also observe that the decision procedure for the sequents HS_n^m (see 2.1) is easily computed by this tableau method. However, an acyclic graph representation of the tableau leads to the result

Theorem 8. *The tableau method is performed in polynomial time.*

4 Conclusion

We have shown that we can speed up the proof search in ALL. Moreover this tedious bookkeeping could be also done on proof template and take care of particular cases which will lead to some optimizations. Surely this technique is easily generalized to another logical fragments. We have also proposed a tableau method for ALL that is polynomial-time computable. It will be interesting to study the mixing of such methods (with refinements) with existing ones for another fragments (with multiplicatives for example) to provide new efficient proof search methods.

References

1. A. Avron, Some Properties of Linear Logic Proved by semantic Methods, *Journal of Logic and Computation*, 4(6):929-938, 1994.
2. S. R. Buss, Algorithms for Boolean Formula Evaluation and for Tree Contraction, 96-115, In *Arithmetic, Proof Theory, and Computational Complexity* ed. by P. Clote and J. Krajicek, Oxford Logic Guides 23, 1993.
3. S. R. Buss, S. Cook, A. Gupta and V. Ramachandran: An Optimal Parallel Algorithm for Formula Evaluation, *SIAM J. On Computing*, 21(4):755-780, August 1992.
4. P. Bradford and J.-Y. Marion and L. Moss, The additive fragment of Linear Logic is NC1-complete, in *Int. Workshop on Logic, Complexity and Computation*, Indianapolis, October, 1994.
5. H. B. Curry, *Foundations of Mathematical Logic*, Dover ed., 1976.
6. M. D'Agostino and M. Mondadori, The taming of the cut. Classical refutations with analytic cut, *Journal of Logic and Computation*, 4(3):285-319, 1994.
7. M. D'Agostino and D.M. Gabbay, A generalization of Analytic Deduction via labelled deductive systems. Part I: Basic Substructural Logics, *Journal of Automated reasoning*, 13:243-281, 1994
8. D. Galmiche and G. Perrier, Foundations of proof search design in linear logic, In *Symposium on Logical Foundations in Computer Science*, LNCS 813, 101-113, St Petersburg, July 1994.
9. D. Galmiche and G. Perrier, On proof normalization in linear logic, *Theoretical Computer Science*, 135(1):67-110, 1994.
10. J-Y. Girard, Linear Logic, *Theoretical Computer Science*, 50:1-102, 1987.
11. J-Y. Girard: Geometry of Interaction III: Accommodating the Additives, manuscript, 1994.
12. M. I. Kanovich, Horn Programming in Linear Logic is NP-Complete, In 7^{th} *Symposium on Logic in Computer Science* , IEEE Press, 200-210, 1992.
13. P. Lincoln, J. Mitchell, A. Scedrov and N. Shankar: Decision Problems for Propositional Linear Logic, *Annals Pure Appl. Logic*, 56:239-311, 1992.
14. P. Lincoln, J. Mitchell, and A. Scedrov, Stochastic Interaction and Linear Logic, manuscript, 1994.
15. C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.
16. A. S. Troelstra, *Lectures on Linear Logic*, CLSI Lecture Notes, No. 29, CSLI, 1992.
17. A. Urquhart, Complexity of proofs in classical propositional logic, In *Logic from Computer Science*, ed. Y. Moschovakis, Springer-Verlag, 596–608, 1992.

This article was processed using the LaTeX macro package with LLNCS style