Journal of Automated Reasoning 6: 389–402, 1990.

A Sequent-Style Model Elimination Strategy and a Positive Refinement*

DAVID A. PLAISTED
Department of Computer Science, CB # 3175, 352 Sitterson Hall, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175, U.S.A.

(Received: 27 May 1988; revised: 3 January 1989)

Abstract. We present a sequent style proof system related to the MESON procedure, which is itself based on the model elimination strategy for mechanical theorem proving. The MESON procedure is attractive because it is a problem reduction format, that is, it has a goal-subgoal structure. The sequent style system based on it shares this advantage, and also has a simple declarative semantics and soundness proof. A refinement of the sequent style system tends to produce shorter sequents and may facilitate the use of the MESON procedure with caching of solutions to subgoals to avoid repeated work on the same subgoal. In the MESON procedure, a goal is marked 'contradicted' and considered to be solved, if an ancestor goal is complementary to it. In the positive refinement, only the positive goals need to be checked for contradiction in this way. This means that if a list of ancestor goals is kept with each goal, it is only necessary to store the negative ancestor goals, since these are the only ones that need to be examined in testing if a positive goal is contradicted. Similar restrictions on the reduction operation of model elimination are possible.

Key words. Theorem proving, model elimination, MESON procedure.

1. Introduction

The model elimination strategy of Loveland [4] has become important in mechanical theorem proving for a least two reasons: (1) It can be viewed as a predecessor of Prolog's depth-first search strategy (Clocksin and Mellish [2]), and (2) a recent implementation by Stickel [10] is extremely fast. However, the soundness of ME (model elimination) is not as obvious as one would like. Also, the length of the chains (sequences of literals) of ME grows rapidly, leading to possible inefficiencies in implementation especially when caching is used. The MESON procedure of Loveland [5] is a derivation of model elimination that has a problem reduction format, that is, it has a goal-subgoal structure similar to Prolog. Instead of the chains of ME, the MESON procedure has lists of literals called 'ancestor lists'. We present a sequent style proof system equivalent to the MESON precedure. The soundness of this system is apparent. We also present a refinement of the sequent

* This research was supported in part by the National Science Foundation under grant DCR-8516243.

style MESON procedure in which the ancestor list grows more slowly, and prove its completeness. This refinement is also of interest because it may make caching of solutions to subgoals (as in Plaisted [88]) feasible. Despite its speed, Stickel's implementation of ME (actually, of the MESON procedure) is sometimes inefficient because its does not remember solutions to subgoals encountered previously, and so it may repeat the same work many times. Don Loveland [6] has recently developed an extension of Prolog to full first order logic which is similar to the MESON procedure in some ways, but treats positive and negative literals differently, unlike the standard MESON procedure. For another such strategy in a sequent style see Plaisted [9]. Both of these strategies do not need to use contrapositives of clauses, as the MESON procedure does. Another difference from the MESON procedure is that the 'modified problem reduction format' of [9] is not always a set of support strategy, although in practice it often behaves as one. On the other hand, the strategy of Plaisted permits subgoal deletion based on semantics, as does the 'simplified problem reduction format' of Plaisted [8], but the MESON procedure does not, since the negation of the theorem may appear anywhere in the proof. Although subgoals cannot be deleted using semantics in the MESON procedure, we do present a semantic version of the MESON procedure below. In this semantic MESON procedure, a model is used but not for the purpose of deleting false subgoals.

We first present the sequent style MESON system and some examples. Then we present the positive refinement, and prove soundness and completeness. We assume that the reader is familiar with standard first-order logic terminology; for an introduction see Chang and Lee [1], Gallier [3], and Loveland [5]. The latter reference also has a description of the MESON procedure. Also, we present propositional versions of all strategies for simplicity, but they all lift to first-order logic (quantifier-free clause form) as usual. However, factoring (merging) is not needed in the first-order versions of these systems, in contrast to resolution.

2. Logical Preliminaries

DEFINITION. A *proposition* (atom) is a predicate constant.

DEFINITION. A *literal* is a proposition or its negation. A *positive* literal is a proposition and a *negative* literal is the negation of a proposition. The literals P and not(P) are called *complementary*. If L is not(P) then $\neg\mathbf{L}$ refers to P.

DEFINITION. A *clause* is a disjunction of literals. A *Horn clause* is a clause with at most one positive literal. A *Horn set* is a set of Horn clauses. We write a clause as a set of literals, denoting the disjunction of the literals in the set. Thus $\{P, \text{not}(Q), R\}$ denotes $\mathbf{P} \vee \mathbf{not(Q)} \vee \mathbf{R}$. Clauses may be written in Prolog style as $\mathbf{L} :\!\!- \mathbf{L}_1, \ldots, \mathbf{L}_n$ where $\mathbf{L}$ and the $\mathbf{L}_i$ are literals. This represents the clause $\mathbf{L}_1 \wedge \cdots \wedge \mathbf{L}_n \supset \mathbf{L}$, that is, $\{\neg\mathbf{L}_1 \vee \cdots \vee \neg\mathbf{L}_n \vee \mathbf{L}\}$. Note that the same clause may be represented in Prolog style in more than one way. Thus $\mathbf{P} \vee \mathbf{Q}$ may be represented

as P :- not(Q) or as Q :- not(P). These different representations of a clause are called *contrapositives* of each other. In Prolog it is customary to use small letters for predicates, while in first order logic it is often customary to use upper case letters for predicates. Thus our notation will vary.

DEFINITION. A set S of clauses is *unsatisfiable* if it has no model, that is, no interpretation making all the clauses in S true.

Typically we prove a formula A is valid (true) by negating it, converting $\neg A$ to clause form S, and testing if S is unsatisfiable, since S is unsatisfiable iff A is valid.

DEFINITION. A *sequent* (for our purposes) is a formula of the form $\Gamma \rightarrow \mathbf{L}$ where Γ is a list (set) of literals and L is a literal. Such a sequent is interpreted to mean that the conjunction of the literals in Γ implies L. If Γ is empty then we may write $\rightarrow \mathbf{L}$ or $\mathbf{L}$ instead of $\Gamma \rightarrow \mathbf{L}$. A *sequent-style proof system* is a proof system in which all rules are of the form

$$\frac{\Gamma_1 \rightarrow \mathbf{L}_1, \ldots, \Gamma_n \rightarrow \mathbf{L}_n}{\Gamma \rightarrow \mathbf{L}}$$

signifying that if the $\Gamma_1 \rightarrow \mathbf{L}_i$ have been shown then one can infer $\Gamma \rightarrow \mathbf{L}$. In terms of search strategy, we may consider the $\Gamma_i \rightarrow \mathbf{L}_i$ as subgoals of $\Gamma \rightarrow \mathbf{L}$. For a discussion of such systems, see for example [3].

3. The MESON Procedure in Sequent Style

For each set S of clauses we have a sequent-style proof system for S. Suppose S is a set of clauses and $\mathbf{L}$:- $\mathbf{L}_1, \ldots, \mathbf{L}_n$ is a Prolog representation of a clause C in S. (Note that there will also be n other Prolog style representations of C.) For this representation we have the following rule in the sequent-style proof system for S, where brackets [. . .] are used to group sequents:

$$\frac{[\Gamma, \neg \mathbf{L}_1 \rightarrow \mathbf{L}_1], \ldots, [\Gamma, \neg \mathbf{L}_n \rightarrow \mathbf{L}_n]}{\Gamma \rightarrow \mathbf{L}}$$

Thus there are $n + 1$ rules for a clause that has $n + 1$ literals. For a single literal clause $\mathbf{L}$ there is the axiom $\Gamma \rightarrow \mathbf{L}$, corresponding to the above rule without hypotheses. One of the clauses in S is chosen as the *support clause*. For this clause $\{\neg \mathbf{L}_1, \ldots, \neg \mathbf{L}_n\}$, we choose the additional Prolog representation as **false** :- $\mathbf{L}_1, \ldots, \mathbf{L}_n$, yielding in addition to rules as above the rule.

$$\frac{[\Gamma, \neg \mathbf{L}_1 \rightarrow \mathbf{L}_1], \ldots, [\Gamma, \neg \mathbf{L}_n \rightarrow \mathbf{L}_n]}{\Gamma \rightarrow \mathbf{false}}$$

Thus, for a support clause having n literals, there are $n + 1$ rules in all. Also, there is the *assumption axiom*

$\Gamma \rightarrow \mathbf{L}$ **if** $\mathbf{L} \in \Gamma$.

We denote by $\vdash s$ the logical system consisting of all such rules for all clauses in S, together with the assumption axiom. To show S is unsatisfiable, we prove **false** using the rules in $\vdash s$.

For example, corresponding to the clause p :- not q, r (equivalently, $\{p, q, \text{not}(r)\}$) we have the rules

$$\frac{[\Gamma, \mathbf{q} \rightarrow \mathbf{not(q)}], [\Gamma, \mathbf{not(r)} \rightarrow \mathbf{r}]}{\Gamma \rightarrow \mathbf{p}}$$

$$\frac{[\Gamma, \mathbf{p} \rightarrow \mathbf{not(p)}], [\Gamma, \mathbf{not(r)} \rightarrow \mathbf{r}]}{\Gamma \rightarrow \mathbf{q}}$$

$$\frac{[\Gamma, \mathbf{q} \rightarrow \mathbf{not(q)}], [\Gamma, \mathbf{p} \rightarrow \mathbf{not(p)}]}{\Gamma \rightarrow \mathbf{not(r)}}$$

The first of these correspond to the following Prolog clause, using arrow (Γ, L) for $\Gamma \rightarrow \mathbf{L}$, and representing lists of literals by Prolog lists:

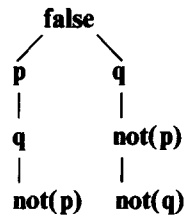
```
arrow(Γ, p) :-
        arrow([q | Γ], not(q)),
        arrow([not(r) | Γ], r).
```

Unit clauses (clauses with only one literal) correspond to axioms, or to Prolog clauses with no literals to the right of :-.

We now discuss the connection of this system with the MESON procedure. The description of the MESON procedure in [10] is particularly simple, and we follow it here. The MESON procedure is essentially Prolog, with contrapositives of clauses used. Also, if a subgoal is complementary to one of its ancestors, the subgoal succeeds; if the subgoal is identical to an ancestor, it fails (the 'failure rule'). In this system, a proof of **false** demonstrates the unsatisfiability of S. Suppose S is $\{p, q\}$, $\{p, \text{not}(q)\}$, $\{\text{not}(p), q\}$, $\{\text{not}(p), \text{not}(q)\}$, and $\{\text{not}(p), \text{not}(q)\}$ is chosen as the support clause. Then the set of Prolog representations of the clauses in S is as follows:

```
p :- not q
q :- not p
p :- q
not q :- not p
not p :- not q
q :- p
not p :- q
not q :- p
false :- p, q
```

The last clause false :- p, q is added because $\{\text{not}(p), \text{not}(q)\}$ is chosen as the support clause. Here is an example of a proof of unsatisfiability for S in the MESON procedure, with subgoals being written below their parent goals:



The support clause is regarded as the query *:-p, q* or false :- *p, q*. Thus from the top subgoal **false** we generate *p* and *q* as subgoals. From *p*, the literal *q* is generated as a subgoal using the clause *p* :- *q*. From *q*, the literal not(*p*) is generated as a subgoal using *q* :- not(*p*). The ancestors of not(*p*) are false, *p*, and *q*. Since not(*p*) is complementary to *p*, the subgoal not(*p*) succeeds (it is 'contradicted', in MESON terminology). The other half of the proof is similar, except that the contrapositive not(*p*) :- not(*q*) of the clause *q* :- *p* is used.

The same proof may be done in the sequent style MESON procedure as follows, with subgoals written above goals:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{not(p),\ not(q),\ p \rightarrow not(p)}}}{\mathbf{not(p),\ not(q) \rightarrow q}} \qquad \dfrac{\dfrac{\dfrac{}{\mathbf{not(q),\ p,\ q \rightarrow not(q)}}}{\mathbf{not(q),\ p \rightarrow not(p)}}}{\mathbf{not(q) \rightarrow q}}}{\mathbf{not(p) \rightarrow p}} \qquad \mathbf{false}$$

Notice that the left part of the sequents contains the ancestor list of the subgoals with the signs of the subgoals reversed. Therefore the success condition for the MESON procedure, namely, that a subgoal is complementary to an ancestor, is translated to the condition that the right part of the sequent occurs in the left part of the sequent. Also, the failure rule corresponds to repeated literals in the left part of a sequent, it turns out.

3.1. LIFTING TO FIRST ORDER LOGIC

We briefly comment on how the sequent style MESON procedure may be implemented in first order logic. The implementation is as suggested by the Prolog clause

```
arrow(Γ, p):-
        arrow([q | Γ], not(q)),
        arrow([not(r) | Γ], r).
```

given above as a representation of an inference rule in the sequent style MESON procedure. Thus unification is used to match subgoals with inference rules. Also, the assumption axiom permits a sequent $\Gamma \rightarrow L$ to succeed if L unifies with a literal in Γ. If L is identical to a literal in Γ, then the sequent succeeds, and other possibilities for achieving it need not be explored.

4. A Positive Refinement of the MESON Procedure

We now present another sequent style proof system for sets of clauses which we call the *positive refinement* of the MESON strategy in sequent form. The idea is that it is only necessary to check negative subgoals against (positive) ancestors. The same idea applies also to model elimination; the reduction operation only needs to be done on chains whose rightmost element is a positive B-literal (for those familiar with model elimination terminology), since a negative subgoal corresponds to a positive literal in a chain. The positive refinement of the MESON strategy may be framed in a sequent style notation, as the standard MESON strategy was; the difference from the sequent style system in Section 3 above is that the negative literals in the antecedents need not be retained. These negative literals correspond to positive ancestor subgoals, which are not needed in the positive refinement of the MESON strategy. This new sequent-style system is as above except that corresponding to the Prolog representation $\mathbf{L}$:- $\mathbf{L}_1, \ldots, \mathbf{L}_n$ of a clause C in S we have the rules

$$\frac{[\Gamma_1 \rightarrow \mathbf{L}_1], \ldots, [\Gamma_n \rightarrow \mathbf{L}_n]}{\Gamma \rightarrow \mathbf{L}}$$

where Γ_j is Γ if $\mathbf{L}_j$ is positive and $\Gamma, \mathbf{L}_j$ otherwise. Also, as before we have an extra rule

$$\frac{[\Gamma_1 \rightarrow \mathbf{L}_1], \ldots, [\Gamma_n \rightarrow \mathbf{L}_n]}{\Gamma \rightarrow \mathbf{false}}$$

for the support clause $\{\neg\mathbf{L}_1, \ldots, \neg\mathbf{L}_n\}$, corresponding to the extra Prolog representation **false** :- $\mathbf{L}_1, \ldots, \mathbf{L}_n$ for this clause. Finally, we have the assumption axiom

$$\Gamma \rightarrow \mathbf{L} \text{ if } \mathbf{L} \in \Gamma.$$

For example, corresponding to the clause p :- not q, r we would have the rules

$$\frac{[\Gamma, \mathbf{q} \rightarrow \mathbf{not(q)}], [\Gamma \rightarrow \mathbf{r}]}{\Gamma \rightarrow \mathbf{p}}$$

$$\frac{[\Gamma, \mathbf{p} \rightarrow \mathbf{not(p)}], [\Gamma \rightarrow \mathbf{r}]}{\Gamma \rightarrow \mathbf{q}}$$

$$\frac{[\Gamma, \mathbf{q} \rightarrow \mathbf{not(q)}], [\Gamma, \mathbf{p} \rightarrow \mathbf{not(p)}]}{\Gamma \rightarrow \mathbf{not(r)}}$$

Suppose S is $\{p, q\}$, $\{p, \text{not}(q)\}$, $\{\text{not}(p), q\}$, $\{\text{not}(p), \text{not}(q)\}$, as above, and $\{\text{not}(p), \text{not}(q)\}$ is chosen as the support clause. The Prolog representations of these clauses are as given earlier. In the positive MESON system for S, we can prove false (that is, $\Gamma \rightarrow$ **false** for empty Γ) by the following proof:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\mathbf{q}, \mathbf{p} \rightarrow \mathbf{p}}{\mathbf{q}, \mathbf{p} \rightarrow \mathbf{not(q)}}}{\mathbf{p} \rightarrow \mathbf{not(p)}}}{\mathbf{q}} \qquad \mathbf{p}}{\mathbf{q}}}{\mathbf{false}}$$

Since we have proved **false**, S is unsatisfiable. In the above proof, there are two occurrences of q but the proof is shown for only one of them. This system corresponds to the MESON procedure with only negative literals being put in the ancestor lists. Note that the sequents in this proof are shorter than in the previous one.

Sometimes proofs in the positive refinement are longer than those in the usual MESON system. This is true of the above example; see Section 3 for a shorter proof. For another example, suppose S consists of the following clauses:

:- P
P :- A
A :- B
B :- not(A)

We then obtain the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{A},\ \mathbf{P} \rightarrow \mathbf{not(P)}}{\mathbf{A} \rightarrow \mathbf{not(A)}}}{\mathbf{B}}}{\mathbf{A}}}{\mathbf{P}}}{\mathbf{false}}$$

Note that since $:\text{-}P$ is an input clause, this clause may also be written as not(P) and thus the sequent $A, P \rightarrow \text{not}(P)$ is proved. The proof in the usual MESON system is

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{not(P),\ not(A),\ not(B),\ A} \rightarrow \mathbf{not(A)}}{\mathbf{not(P),\ not(A),\ not(B)} \rightarrow \mathbf{B}}}{\mathbf{not(P),\ not(A)} \rightarrow \mathbf{A}}}{\mathbf{not(P)} \rightarrow \mathbf{P}}}{\mathbf{false}}$$

For Horn sets, most of the literals in clauses will typically be negative. This means that most of the literals in the Prolog representations of the clauses will be positive. Therefore few literals will typically be added to Γ for Horn sets or near-Horn sets. In fact, for Horn sets Γ will always be empty if an all-negative clause is chosen as the support clause, and so this proof system is very much like pure Prolog. Many mathematical theorems are near Horn sets, for which the positive refinement significantly reduces the size of Γ. This makes the implementation of the MESON procedure more efficient since the sets Γ are smaller and take less time to search.

5. Soundness and Completeness

DEFINITION. We say a logical system is *sound* if it preserves truth. That is, if one can prove A from assumptions S then A is a logical consequence of S. For the sequent style systems, we say they are sound if $\vdash_s A$ implies (A is a logical

consequence of S), where a sequent $\Gamma \rightarrow \mathbf{L}$ is interpreted as $\Gamma \supset \mathbf{L}$ and Γ is interpreted as a conjunction of literals.

DEFINITION. We say a sequent style system is *complete* if for any unsatisfiable set S of clauses, $\vdash_s$ false.

THEOREM 1. *The sequent style MESON procedure is sound and complete.*

Proof. Soundness is fairly straightforward. For a rule of the form

$$\frac{\Gamma_1 \rightarrow \mathbf{L}_1, \ldots, \Gamma_n \rightarrow \mathbf{L}_n}{\Gamma \rightarrow \mathbf{L}}$$

it is necessary to show that $[(\Gamma_1 \supset \mathbf{L}_1) \wedge \cdots \wedge (\Gamma_n \supset \mathbf{L}_n)] \supset (\Gamma \supset \mathbf{L})$, where Γ is interpreted as a conjunction of literals. Now, Γ_i is either Γ or $\Gamma \cup \{\neg(\mathbf{L}_i)\}$, depending on the sign of $\mathbf{L}_i$ and which system we are using. In either case, $\Gamma_i \supset \mathbf{L}_i$ is equivalent to $\Gamma \supset \mathbf{L}_i$ by the rules of Boolean logic, so it suffices to show that $[(\Gamma \supset \mathbf{L}_1) \wedge \cdots \wedge (\Gamma \supset \mathbf{L}_n)] \supset (\Gamma \supset \mathbf{L})$. This is in turn a logical consequence of $[\mathbf{L}_1 \wedge \cdots \wedge \mathbf{L}_n] \supset \mathbf{L}$. But this is equivalent to a clause of S, by the way the rules are constructed. Also, the assumption axiom is a tautology, when $\rightarrow$ is interpreted as logical implication. Hence the systems are sound. By completeness, we mean that if the correct clause is chosen as the support clause, it is possible to derive $\Gamma \rightarrow$ **false** for empty Γ. For this, it suffices to choose a clause C such that $S - \{C\}$ is satisfiable if such exists. If not, then S is not minimal unsatisfiable, but even then if S is unsatisfiable some clause in S will work as a support clause. We will show completeness by a fairly involved argument in Section 8. The ideas for this proof are taken from the Ph.D. thesis of Plaisted [7].

THEOREM 2. *The positive refinement of the sequent style MESON procedure is sound and complete.*

Proof. As above, soundness is easy. We will show completeness in Section 8.

6. Applications

The positive refinement of the sequent style MESON procedure is suitable for caching. The idea of caching is to avoid attempting to solve the same subgoal repeatedly. Instead, solutions are remembered and reused. The reason that this refinement is suitable for caching is that the lists Γ are typically small, so it is more likely that a given sequent will be generated as a subgoal more than once. If all ancestors are included in Γ, then the sequents will be longer and it is less likely that the same one will be generated more than once. Caching avoids repeated work on the same subgoal, at a cost in speed. If a subgoal is rarely generated more than once, then caching does not significantly reduce repeated work, but does incur a penalty in execution speed. Therefore caching is most useful for systems such as the positive refinement of the sequent style MESON procedure in which the same subgoal is likely to be generated repeatedly. For a discussion of a prover using back

chaining together with caching, see [9]. The sequent style systems given here are suitable for back chaining implementations.

As mentioned earlier, Stickel has a very efficient implementation of the MESON procedure which does not cache solutions to subgoals. Sometimes this causes his prover to perform much worse than one would expect (though it is fairly fast). Therefore the positive refinement of the sequent style MESON procedure might provide a way for Stickel's implementation to incorporate caching and improve performance on some problems.

Note that the MESON procedure is a set of support strategy, unlike the modified problem reduction format of [9]. This is an advantage, because the set of support restriction focuses attention on deductions that are relevant to the theorem. In practice, the modified problem reduction format is a set of support strategy unless there is an axiom that is a clause containing only negative literals. For many simple problems, only clauses from the negation of the theorem are all-negative, and on these problems the modified problem reduction format behaves as a set of support strategy. However, we do not know of any strategy which is strictly a set of support strategy but does not require contrapositives of clauses, as do the sequent style MESON procedures. Contrapositives may be undesirable because they effectively increase the number of input clauses used in back chaining, and because they lead to a loss of control over the search. However, these disadvantages are often outweighed by the advantages of the set of support property.

7. A Semantic Refinement

Instead of specifying positive and negative literals, we can specify literals that are true or false in a particular interpretation I. Thus for example we can restrict sequents so that only literals true in I are kept in Γ. This generalizes the positive refinement and may be useful in some contexts. Lifted to first order logic, this 'semantic refinement' would restrict Γ to be satisfiable in I.

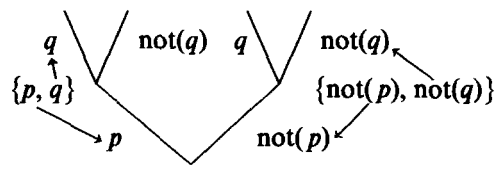
8. Completeness of the Proof Systems

The proof of completeness makes use of semantic trees.

DEFINITION. Given a set S of first-order clauses, a *semantic tree over* S is a finite binary tree T in which each interior node has two sons, a left son and a right son. We view edges as directed from nodes to their sons, and write the edge from M to N as $\langle M, N\rangle$. In addition, each edge e of T is labeled with a literal lit(e). If $N1$ and $N2$ are sons of M then the labels of $\langle M, N1\rangle$ and $\langle M, N2\rangle$ must be complementary literals. A *path* in T is a sequence $\langle M1, M2\rangle, \langle M2, M3\rangle, \ldots$ of edges of T. A *maximal path* in T is a path in T that is not a subsequence of any longer path in T. Thus a maximal path will connect the root of T with a leaf, and there is a one to one correspondence between maximal paths and leaves of T. For each maximal path

P in T, there is a clause $C(P)$ which is an *instance* of a clause in S, and a mapping $\mathbf{f_P}$ from the literals of $C(P)$ into edges of P such that $\text{lit}(\mathbf{f_P}(L)) = L$. We may view $\mathbf{f_P}$ as assigning literals in $C(P)$ to edges of T. The condition that $\text{lit}(\mathbf{f_P}(L)) = L$ means that a literal L must be assigned to an edge labeled L. Furthermore, we require that for every edge e in T there is a maximal path P containing e and a literal L in $C(P)$ such that $\mathbf{f_P}(L) = e$. (Note that there may be several such paths P containing e; not every associated $C(P)$ need have a literal L such that $\mathbf{f_P}(L) = e$.) Thus, every edge must be assigned a literal L from some clause which is an instance of a clause in S.

For example, if S is $\{\{p, q\}\{p\ \text{not}(q)\}\{not(p)\ q\}\{\text{not}(p)\text{not}(q)\}\}$ then a possible tree T is



In this tree, the mappings for only two maximal paths are shown. Note that a clause may appear as $C(P)$ for more than one maximal path P.

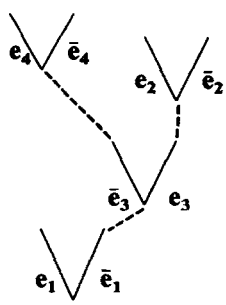
THEOREM 3. *If S is unsatisfiable then there is a (finite) semantic tree T over S.*

Proof. By Herbrand's theorem. As in Chang and Lee [1], there is a finite semantic tree T over S. If T has edges that are not assigned any literals then these edges can be deleted from T to obtain a smaller tree, and this process can be repeated until all edges of T have literals assigned to them.

DEFINITION. Edges $\langle M, N\rangle$ and $\langle M, Q\rangle$ (where N and Q are the sons of M) are called *mates*. If e is an edge then $\bar{e}$ denotes the mate of e. Thus $\overline{\langle \mathbf{M}, \mathbf{N}\rangle}$ is $\langle \mathbf{M}, \mathbf{Q}\rangle$ and $\overline{\langle \mathbf{M}, \mathbf{Q}\rangle}$ is $\langle \mathbf{M}, \mathbf{N}\rangle$.

DEFINITION. An *alternating sequence* in a semantic tree T over S is a sequence $(\mathbf{e}_1\ \bar{\mathbf{e}}_1)(\mathbf{e}_2\ \bar{\mathbf{e}}_2)(\mathbf{e}_3\ \bar{\mathbf{e}}_3)\cdots$ of pairs of edges of T such that for all $\mathbf{i}$, $\mathbf{i} \geqslant 1$, $\bar{\mathbf{e}}_\mathbf{i}$ and $\mathbf{e}_{\mathbf{i}+1}$ are in a common maximal path $\mathbf{P_i}$ and there are literals $\mathbf{L_i}$ and $\mathbf{M_i}$ in $\mathbf{C(P_i)}$ such that $\mathbf{f_P(L_i)} = \bar{\mathbf{e}}_\mathbf{i}$ and $\mathbf{f_P(M_i)} = \mathbf{e}_{\mathbf{i}+1}$.

For example, we may have a tree like this:



In this illustration, the alternating sequence is $(\mathbf{e}_1, \bar{\mathbf{e}}_1), (\mathbf{e}_2, \bar{\mathbf{e}}_2), (\mathbf{e}_3, \bar{\mathbf{e}}_3), (\mathbf{e}_4, \bar{\mathbf{e}}_4)$. We are assuming that for some maximal path P containing $\bar{\mathbf{e}}_1$ and $\mathbf{e}_2$, the mapping $\mathbf{f}_P$ assigns literals to both edges $\bar{\mathbf{e}}_1$ and $\mathbf{e}_2$. Similarly, for some other path, a mapping assigns literals to both edges $\bar{\mathbf{e}}_2$ and $\mathbf{e}_3$, etc.

DEFINITION. Suppose T is a semantic tree and n is a vertex of T. Then the *subtree of N in T* is the set of vertices M including N and all vertices contained in paths starting at N. Thus the subtree of N in T includes N and the sons of N and their sons, etc.

THEOREM 4. *Suppose S is an unsatisfiable set of ground clauses and T is a finite semantic tree over S. Suppose A is an infinite alternating sequence in T. Then there is an edge $\mathbf{e}$ in T such that lit($\mathbf{e}$) is a positive literal and such that the pairs $(\mathbf{e}, \bar{\mathbf{e}}), (\bar{\mathbf{e}}, \mathbf{e})$ occur in this order in A.*

Proof. Some of the ideas of this proof are due to Floyd (see [7]). We first make a general comment about alternating sequences. An alternating sequence can only exit a subtree by going below it, that is, by including an edge between the root of the whole tree and the root of the subtree. Let $\mathbf{d}$ and $\bar{\mathbf{d}}$ be the edges of T nearest to the root of T, such that $\mathbf{d}$ and $\bar{\mathbf{d}}$ occur infinitely often in A. It is not difficult to show that this set of edges is unique. Suppose $\mathbf{d}$ is $\langle M, N\rangle$ and $\bar{\mathbf{d}}$ is $\langle M, Q\rangle$. Let $\mathbf{E}_1$ be $\mathbf{d}$ together with all edges in the subtree of N, and let $\mathbf{E}_2$ be $\bar{\mathbf{d}}$ together with all edges in the subtree of Q. After some finite initial portion, the alternating sequence A consists entirely of edges in $\mathbf{E}_1 \mathbf{U} \mathbf{E}_2$. Let $\mathbf{E}_1'$ be $\mathbf{E}_1 - \{\mathbf{d}\}$, and let $\mathbf{E}_2'$ be $\mathbf{E}_2 - \{\bar{\mathbf{d}}\}$. Then $\mathbf{E}_1'$ is also a subtree, and if the alternating sequence exits $\mathbf{E}_1'$, it must go below $\mathbf{E}_1'$. After a finite initial portion, the sequence will not go below $\mathbf{E}_1 \mathbf{U} \mathbf{E}_2$. Thus, in order to exit from $\mathbf{E}_1'$, the sequence must include the edge $\mathbf{d}$. Similarly, to exit from $\mathbf{E}_2'$, the sequence must include the edge $\bar{\mathbf{d}}$. Such exits must occur infinitely often, since $\mathbf{d}$ and $\bar{\mathbf{d}}$ occur infinitely often in this sequence. Thus A must alternate between $\mathbf{E}_1$ and $\mathbf{E}_2$ infinitely often. However, A remains within $\mathbf{E}_1$ until $\mathbf{d}$ occurs, and A remains within $\mathbf{E}_2$ until $\bar{\mathbf{d}}$ occurs. Thus the occurrences of $(\mathbf{d}, \bar{\mathbf{d}})$ and $(\bar{\mathbf{d}}, \mathbf{d})$ alternate in A, and both $(\mathbf{d}, \bar{\mathbf{d}})$ and $(\bar{\mathbf{d}}, \mathbf{d})$ occur infinitely often in A. If lit($\mathbf{d}$) is positive, we can choose $\mathbf{e}$ to be $\mathbf{d}$, else we can choose $\mathbf{e}$ to be $\bar{\mathbf{d}}$ and obtain pairs $(\mathbf{e}, \bar{\mathbf{e}})$ and $(\bar{\mathbf{e}}, \mathbf{e})$ as in the theorem.

DEFINITION. In a sequent style proof, if a rule

$$\frac{\Gamma_1 \rightarrow \mathbf{L}_1, \ldots, \Gamma_n \rightarrow \mathbf{L}_n}{\Gamma \rightarrow \mathbf{L}}$$

is applied, we say that $\Gamma_1 \rightarrow \mathbf{L}_1$ is a *subgoal of* $\Gamma \rightarrow \mathbf{L}$.

DEFINITION. Suppose $\mathbf{e}$ is an edge and P is a path containing $\bar{\mathbf{e}}$. We call the path P a *mate* of $\mathbf{e}$. Note that there may be more than one such path P.

THEOREM 5. *Suppose S is an unsatisfiable set of propositional clauses. Then $\vdash_s \rightarrow$ **false** where $\vdash_s$ is the positive refinement of the sequent style system for the MESON procedure. This means that the MESON system with the positive refinement*

is complete. Note that this result also implies the completeness of the MESON system without the positive refinement since it is less restrictive.

Proof. We will construct a proof of $\rightarrow$**false** in $\vdash_s$ working backwards from the goal $\rightarrow$**false**, that is, from $\Gamma \rightarrow$ **false** with Γ empty. Since S is unsatisfiable, there is a (finite) semantic tree T over S, by Theorem 3. With each occurrence of a sequent $\Gamma \rightarrow \mathbf{L}$ in the proof so constructed, we will have associated on edge **e** of T such that lit(**e**) is $\neg L$. The idea is, that if a sequent $\Gamma \rightarrow \mathbf{L}$ appears, then L is a subgoal, so that some clause in S has a Prolog representation of the form $\mathbf{L}'$:- ... **L** ..., and $\neg\mathbf{L}$ appears in the disjunctive representation of this clause as a set of literals. Thus we associate an edge **e** such that lit(**e**) is $\neg L$, with this sequent.

We now define the proof inductively. Let C be one of the clauses in S. This clause C may be chosen arbitrarily, subject to the condition that S has an unsatisfiable subset $\mathbf{S}_1$ such that $\mathbf{S}_1 - \{C\}$ is satisfiable. Thus C is an essential clause for some proof of unsatisfiability of S. Representing C as a Prolog clause **false** :- $\mathbf{L}_1, \ldots, \mathbf{L}_n$, we give as the initial step a rule of the form

$$\frac{\Gamma_1 \rightarrow \mathbf{L}_1, \ldots, \Gamma_n \rightarrow \mathbf{L}_n}{\Gamma \rightarrow \textbf{false},}$$

where Γ_i is $\neg\mathbf{L}_i$ if $\mathbf{L}_i$ is negative, empty otherwise. Also, let P be some path in T such that $C(P) = C$. With the occurrence of the sequent $\Gamma_i \rightarrow \mathbf{L}_i$ we associate the edge $\mathbf{f}_P(\neg\mathbf{L}_i)$ of P. For the inductive step, suppose $\Gamma \rightarrow \mathbf{L}$ appears in the proof. If L is in Γ then we stop since $\Gamma \rightarrow \mathbf{L}$ is an axiom. Otherwise, suppose **e** is the edge associated with this occurrence of the sequent, so lit(**e**) is $\neg\mathbf{L}$. Let $\mathbf{P}'$ be a maximal path which is a mate of **e** such that some literal of $\mathbf{C}(\mathbf{P}')$ is assigned to $\bar{\mathbf{e}}$. Note that such a path $\mathbf{P}'$ must exist by the definition of a semantic tree. Suppose $\mathbf{C}(\mathbf{P}')$ has a Prolog style representation $\mathbf{L}$:- $\mathbf{M}_1, \ldots, \mathbf{M}_m$. We then apply the rule

$$\frac{\Gamma_1 \rightarrow \mathbf{M}_1, \ldots, \Gamma_n \rightarrow \mathbf{M}_n}{\Gamma \rightarrow \mathbf{L}}$$

where Γ_i is $\Gamma \mathbf{U} \{\neg\mathbf{M}_i\}$ if $\mathbf{M}_i$ is negative, Γ otherwise. With the occurrence of the sequent $\Gamma_i \rightarrow \mathbf{M}_i$ we associate the edge $\mathbf{f}_{P'}(\neg\mathbf{M}_i)$.

Now, if the proof so constructed is infinite, then by Konig's lemma it contains an infinite path, that is, an infinite sequence $\Gamma_1 \rightarrow \mathbf{L}_1, \Gamma_2 \rightarrow \mathbf{L}_2 \ldots$, of occurrences of sequents such that for all **i**, the occurrence of $\Gamma_{i+1} \rightarrow \mathbf{L}_{i+1}$ is a subgoal of the occurrence of $\Gamma_i \rightarrow \mathbf{L}_i$. Let $\mathbf{e}_i$ be the edge associated with $\Gamma_i \rightarrow \mathbf{L}_i$. Then $(\mathbf{e}_1, \bar{\mathbf{e}}_1)$, $(\mathbf{e}_2, \bar{\mathbf{e}}_2), \ldots$ is an infinite alternating sequence. The reason for this is that some inference rule was used to obtain $\Gamma_{i+1} \rightarrow \mathbf{L}_{i+1}$ from $\Gamma_i \rightarrow \mathbf{L}_i$. This implies that some clause of S has a Prolog style representation of $\mathbf{L}_i$:- ... $\mathbf{L}_{i+1}$ ..., by the way the inference rules are designed. The literals $\mathbf{L}_i$ and $\neg\mathbf{L}_{i+1}$ correspond to edges $\bar{\mathbf{e}}_i$ and $\mathbf{e}_{i+1}$ of the semantic tree, which are both assigned literals from the clause $\mathbf{L}_i$:- ... $\mathbf{L}_{i+1}$ ... of S. Thus we have an alternating sequence $(\mathbf{e}_1, \bar{\mathbf{e}}_1), (\mathbf{e}_2, \bar{\mathbf{e}}_2), \ldots$. Hence eventually by Theorem 4 we have $\mathbf{e}_i$ and $\mathbf{e}_j$ such that lit($\mathbf{e}_i$) is the complement of $\mathbf{L}_i$ and is positive. (Thus $\mathbf{L}_i$ is negative.) Also, $\mathbf{e}_j$ is $\bar{\mathbf{e}}_i$. Since $\mathbf{L}_i$ is negative, $\neg\mathbf{L}_i$

is in Γ_i. Since $\Gamma_i \subset \Gamma_{i+1}$ for all i, $\neg \mathbf{L}_i$ is in Γ_j. But $\mathbf{L}_j$ is the complement of $\mathbf{L}_i$, hence $\mathbf{L}_j$ is in Γ_j (because $\neg \mathbf{L}_i$ is in Γ_j). Hence this path would stop at the sequence $\Gamma_j \rightarrow \mathbf{L}_j$ or earlier, since $\Gamma_j \rightarrow \mathbf{L}_j$ is an axiom.

8.1. A REFINEMENT AND ITS COMPLETENESS

We now show that a rule analogous to the 'failure rule' of the MESON procedure may be included without loss of completeness. The failure rule is to disallow sequents $\Gamma \rightarrow \mathbf{L}$ in which Γ contains repeated literals. For this to be meaningful, we consider Γ as a list rather than as a set of literals. Then a repeated literal in Γ implies that there is a sequent $\Lambda \rightarrow \mathbf{L}$ which is a descendent of some sequent of the form $\Lambda' \rightarrow \mathbf{L}$, that is, the subgoal L appears twice on the same proof path. This corresponds to L being identical to one of its ancestors in the MESON procedure, which causes a failure. Then the literal $\neg L$ is added to Λ twice and so Λ has two occurrences of $\neg L$. For the positive refinement, this can only happen if $\neg L$ is positive, that is, if L is negative. In lifting the sequent style systems to first order logic, a sequent $\Gamma \rightarrow \mathbf{L}$ fails only if Γ contains identical (as opposed to unifiable) literals.

THEOREM 6. *The sequent style MESON procedure with the failure rule added is complete, and this is also true of the positive refinement.*

Proof. Suppose a sequent $\Gamma \rightarrow \mathbf{M}$ with $\mathbf{L}$ in Γ is generated. We can then essentially view $\mathbf{L}$ as a clause added to S, for the part of the proof tree descending from this sequent. This is because a subgoal $\Gamma \rightarrow \mathbf{L}$ will be solved by the assumption axiom, which has the same effect as $\mathbf{L}$ being in S for all descendents of the subgoal $\Gamma \rightarrow \mathbf{M}$. Also, any descendent of the sequent $\Gamma \rightarrow \mathbf{M}$ will be of the form $\Gamma' \rightarrow \mathbf{M}'$ for $\Gamma \subset \Gamma'$, so $\mathbf{L}$ is in Γ' also. Let S' be a minimal unsatisfiable subset of $\mathbf{SU}\{\mathbf{L}\}$, and suppose that $\mathbf{S}'$ contains the cause $\{\mathbf{L}\}$. Note that some such minimal unsatisfiable subset $\mathbf{S}'$ must exist, since $\mathbf{L}$ appears in some clause of $\mathbf{S}$. Also, such a set $\mathbf{S}'$ must contain at least one clause containing $\neg \mathbf{L}$. There is then a finite semantic tree T' over S', as before. Note that T' is smaller than T and the only clause in T' containing $\mathbf{L}$ is the unit clause $\{\mathbf{L}\}$. We can then construct an alternating sequence in T' instead of in T. In this way a proof may be constructed as before. Uses of the unit clause $\mathbf{L}$ in the proof for S' correspond to applications of the assumption axiom. However, a proof constructed in this way will not contain any sequents $\Gamma' \rightarrow \mathbf{M}'$ in which Γ' contains repeated literals. The literal $\mathbf{L}$ for example can never be added to Γ again, since that could only happen if some clause C containing $\mathbf{L}$ were in S', But S' contains $\mathbf{L}$ as a unit clause and is minimal unsatisfiable, so no other clause of S' contains $\mathbf{L}$.

The same argument works for the positive refinement, except that Γ contains only positive literals so the failure rule corresponds to a negative subgoal $\mathbf{L}$ being repeated on some proof path. For the positive refinement, we cannot apply the failure rule if there is some sequent of the form $\Gamma \rightarrow \mathbf{L}$ and some other sequent of the form $\Lambda \rightarrow \mathbf{L}$ in the same proof path, for positive $\mathbf{L}$.

References

1. Chang, C. and Lee, R., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.
2. Clocksin, W. F. and Mellish, C. S., *Programming in Prolog*, Springer-Verlag, New York, 1981.
3. Gallier, J., *Logic for Computer Science: Foundations of Automatic Theorem Proving*, Harper and Row, Philadelphia, 1986.
4. Loveland, D. W., 'A simplified format for the model elimination procedure', *JACM* **16** (1969) 349–363.
5. Loveland, D., *Automated Theorem Proving: A Logical Basis*, North-Holland, New York, 1978.
6. Loveland, D. W., 'Near-Horn prolog', *Proceedings of the Fourth International Conference on Logic Programming*, Melbourne, Australia, 1987, pp. 456–469.
7. Plaisted, D., Theorem Proving and Semantic Trees, Ph.D. thesis, Stanford University, 1976.
8. Plaisted, D., 'A simplified problem reduction format', *Artificial Intelligence* **18** (1982) 227–261.
9. Plaisted D., 'Non-Horn clause logic programming without contrapositives', *J. Automated Reasoning* **4** (1988) 287–325.
10. Stickel, M. E., 'A PROLOG technology theorem prover: implementation by an extended PROLOG compiler', *J. Automated Reasoning* **4** (1988) 353–380.