THEOREM PROVING BY COVERING EXPRESSIONS

L. J. Henschen
W. M. Evangelist
Northwestern University
Evanston, Illinois 60201

Motivation

This paper describes work in progress to develop methods that search directly for unsatisfiable instances of unsatisfiable clause sets (i.e. Herbrand methods) rather than deduce contradictions from such sets. Such approaches attack two serious disadvantages, in the authors' opinion, of resolution. First, resolution is a deduction-oriented rule, and there are generally vastly more inferences that can be deduced from a set of clauses than are used in the refutations produced, even when very restrictive strategies are used; we point to the low penetrance factors cited in the literature. Second, while the separation of variables insures that the general resolvent will subsume families of resolvents of instances, such generality puts the inference in a strictly local context -- aside from possible future resolutions, there is no connection with any other deduction performed in the search so far.

Herbrand methods generally have three parts--1. form some set of replicas of clauses, 2. mate the literals in the replicas, and 3. test for truth-functional unsatisfiability—typically performed in the above order. It is the authors' opinion that the first two of these are the crucial ones; unfortunately, most Herbrand methods reported in the literature have concentrated on the third. For example, Prawitz [3] suggests that when matrix reduction on a set of replicas fails, one should add another replica of each input clause and try again. Gilmore's technique [1] suffers from the additional problem that a disjunctive normal form is calculated before any literals are even mated. This leads to disjunctions that are extremely long. We propose that the three parts above can work together, and that, in particular, <u>the results of the unsatisfiability test can provide significant information about how additional replicas should be chosen</u>. We also propose that it is useful for the three operations to be interspersed, because steps of the unsatisfiability test can often suggest profitable literal matings. This will also reduce the size of the intermediate formulae.

We describe a Herbrand method based on a notation which allows fast, efficient testing for unsatisfiability and easy extraction of information when the test fails. In the ground case each clause yields a cover expression. These expressions are combined to form a cover for the set. Each clause is used <u>once</u>, and the clauses can be taken in <u>any</u> order. There is no wrong order or wrong pair of clauses to use in resolution-based methods. In the general case, we start with some replicas and calculate the cover, then compare this cover with the input clauses to find promising new replicas and literal matings. A more detailed presentation of the method can be found in [2].

Ground Case

We now describe the truth functional test and related notation. Theorems will be stated without proof. The proofs are straightforward. Let S be a set of ground clauses over n atoms, $p_1, \dots, p_n$.

Theorem 1. S is unsatisfiable iff S subsumes each of the 2^n clauses $q_1 \vee \dots \vee q_n$ where each q_i is p_i or $-p_i$.

A <u>cover term</u> is a sequence of n symbols, each of which is +, -, or 0. The symbols + and - are <u>determined</u> symbols. A <u>cover expression</u> is a set of cover terms. A cover term t containing k determined symbols <u>represents</u> the 2^{n-k} clauses whose ith literal has the same sign as position i of t for the positions containing determined symbols and $\pm p_i$ for 0 positions. For example, 0+0- represents the 4 clauses $\pm p_1$ $+p_2$ $\pm p_3$ $-p_4$.

We use terms in a way very similar to Gilmore's disjunctive normal forms. Consider the clause $p_1 \vee p_2 \vee -p_3 = C$. Gilmore would form a DNF of $-C$, $-p_1 \vee p_1 -p_2 \vee p_1 p_2 p_3$; for a set of clauses Gilmore forms such a DNF for each clause and translates the conjunction of those DNFs to DNF, deleting contradictory conjuncts. We formulate a cover expression for each clause; for the above clause, COV(C) consists of the three terms -00, +-0, and +++, analogous to Gilmore's DNF. <u>However, we interpret the terms as disjuncts, i. e. clauses, not as conjuncts</u>. In general, if $C = p_{i_1} \vee \dots \vee p_{i_k}$, then COV(C) has k terms. The jth term has the same symbol in position i_m as the sign of p_{i_m} for m < j, the opposite symbol as the sign of p_{i_j} in position i_j, and 0 everywhere else. The reason for our interpretation as disjunctions is given by

THEOREM 2. COV(C) represents the set of clauses <u>not</u> subsumed by C.

We then "multiply" the covers for the clauses together. If t and s are terms, then t and s are <u>compatible</u> if there is no position in which they contain opposite determined symbols; in this case the <u>product</u>, t*s, is the term whose symbol in position k for any k is given as follows: if either s or t (or both) contain a determined symbol in position k, then the symbol in position k of t*s is that determined symbol, otherwise it is 0 (see example below). The <u>product</u> of two expressions is the union of the products of compatible pairs of terms from the individual expressions.

THEOREM 3. If S is a set of ground clauses, then the product of the covers of clauses in S represents the set of clauses <u>not</u> subsumed by S.

Example. Let S contain $p_1 p_2$, $p_1 -p_2$, and $-p_1 -p_2$. Then we have

clause	clause cover	cummulative cover
$p_1\ p_2$	-0 +-	-0 +-
$p_1\ {-p_2}$	-0 ++	-0*-0 = -0
$-p_1\ p_2$	+0 --	-0*-- = --

The term -- represents $-p_1\ -p_2$, just the clause needed to make S unsatisfiable. If that clause were present, its cover is +0 -+ and the cumulative cover would be empty.

General Case

THEOREM 4. Let S be an unsatisfiable set of clauses. There exists a set S' of replicas of clauses in S, a partition P of literals in S', and a simultaneous unifier σ of the atom set of the partitions of P such that the cover of $S'\sigma$ is empty.

We rely heavily on the cover of a tentative set of replicas to determine what matings and new replicas to add. In the interactive program now in use, multiplication and various other clerical details are performed automatically. The most important interactive command is SGST, N, M. This causes the program to find instances of input clauses which contain an instance of cover literal N and in which all but M of the literals in the input clause are paired with instances of cover literals. SGST uses only cover literals that have the same sign in all cover terms in the cover, limiting suggested new instances to those that have interaction with all cover terms. SGST rejects suggestions that are tautologies or are subsumed by unit input clauses. We now illustrate with the $x^2 = e$ problem, not because it is a hard problem but because it is a fimiliar one. We use the standard P formulation with demodulators $f(e,x) = x$, $f(x,e) = x$, $f(x,x) = e$, and $f(a,b) = c$. The criterion for chosing new instances involved high weight for interaction with the hypotheses and denial of conclusion, i.e., the set of support and low weight for use of closure. User input is in upper case; because of space limitations, comments and the computer's responses will be abbreviated except for the SGST command.

```
READ   21
computer responds with clauses, literals, etc of
                the problem.
SGST  16, 1     16 is -Pbac.
-Pebb -Pbaf(ba) -Pef(ba)c P bac
2     13                   16<-----   numbers of
-Pcg(a)b -Pg(a)ac -Pcec Pbac       cover literals
         4        1    16<-----             used
-Pcab -Paac -Pcec Pbac
      14     1    16
-Pbg(b)e -Pg(b)ca -Pecc Pbac
 3        2            16
-Pbaf(ba) -Paea -Pf(ba)ec Pbac
 13         1              16
-Pbbe -Pbca -Pecc Pbac
 14         2    16
ADDC -Pcab -Paae -Pccc Pbac    Pick first
ADDC Paae        suggestion that has maximal
ADDC Pcec        interaction with SOS.  Add to set.
SGST  17, 1      Ask for suggestion with Pcab.
-Pecc -Pbaf(ba) -Paf(ba)b Pcab
 2     13                  17
-Pabc -Pbaf(ba) -Paf(ba)b Pcab
 15    13                  17
-Pbg(a)c -Pg(a)ae -Pbeb Pcab
          4        1   17
-Pcg(c)e -Pg(c)ba -Pebb Pcab
 3                 2   17
-Pcaf(ca) -Paea -Pf(ca)eb Pcab
 13        1                17
-Pccc -Pcba -Pebb Pcab
 14          2    17
ADDC -Pccc -Pcba -Pebb Pcab     Again pick sug-
ADDC Pccc                       gestion with max-
ADDC Pebb                       imal SOS literals.
computer responds that the clause
-Pabc -Pbbe -Paea Pcba
 15     14    1    21
an instance of associativity, is represented in
all terms of the remaining cover.
ADDC -Pabc -Pbbe -Paea Pcba
ADDC Pbbe
ADDC Paea
computer responds that the proof is completed.
```

Future Research

While we have not yet experimented extensively, we believe our work so far indicates that covering is potentially useful in both fully automatic and interactive uses. The first step is to experiment with our interactive program until we have developed some heuristics for the method. Some of the parameters under consideration are 1. literal weights, 2. number of instances of an input clause used so far, 3. set of support, and 4. semantic information. The notation seems amenable to most of the standard problem solving approaches because the individual terms of the cover represent subproblems to be solved.

We also want to investigate application areas. The method seems suited to problems where the user can apply semantic information to the suggestions obtained from SGST. Such an area is invariant finding in program verification. Here, suggestions like "if 1 is less than 0 then a = b" can be rejected by the user.

A third area of interest is to determine how equality can be incorporated, perhaps in a way similar to paramodulation. Experiments in which we used equality axioms were not generally successful.

References

1. Gilmore, P. C., "A Proof Method for Quantification Theory: Its Justification and Realization," IBM Journal of Research and Develop., 1960, 28-35.

2. Henschen, L. J., "Theorem Proving by Covering Expressions," Available from Dept. of Comp. Sci. Northwestern Univ., Evanston, Ill., 60201

3. Prawitz, D., "Advances and Problems in Mechanical Proof Procedures," Machine Intelligence 4, 1969.