GROUND RESOLUTION USING ANTI-CLAUSES

BY

W. W. Bledsoe

April 1983 ATP-72

Ground Resolution using Anti-Clauses

By

W. W. Bledsoe

Abstract. This note describes a method for deciding whether a finite set S of ground clauses is unsatisfiable. Let A be the set of atoms in S, and F_A be the full set on A (all clauses over A of length $|A|$). An anti-clause of C is any clause in F_A subsummed by C. S is unsatisfiable if the set S_F of anti-clauses of clauses of S, is equal to F_A. A simple algorithm is given for computing S_F from S, and some results of computer runs are provided.

Let A, a list of ground atoms, and F_A be the full set on A. Thus for example if A = {a, b, c}, then F_A is the eight clauses

$$\{ \{ a\ b\ c \} \{ a\ b \sim c \} \{ a \sim b\ c \} \{ a \sim b \sim c \} \{ \sim a\ b\ c \}$$
$$\{ \sim a\ b \sim c \} \{ \sim a \sim b\ c \} \{ \sim a \sim b \sim c \} \}$$

Definition. A clause in F_A which is subsumed by C is called an anti-clause of C.

For example, if A = { a, b, c }, and C = { a c } , then both C_1 = { a b c } and C_2 = { a $\sim$ b c } are anti-clauses of C.

Definition. If S is a set of ground clauses and A is the set of atoms of S, then,

$$S_F = \{ C' : \exists\, C \in S \text{ (} C' \text{ is an anti-clause of } C \text{)} \}$$

Theorem. A finite set S of ground clauses is unsatisfiable if $S_F = F_A$ where A is the set of atoms in S, (i.e., the set of anti-clause of members of S is equal to the full set over A).

Proof. Let C be a clause of S, and S' be the set of anti-clauses of C. Then C implies each member of S' (indeed it subsumes each member of S'), and all the member of S', together, imply C.

This last statement can be shown as follows. Let S" be the full set on

$$A - \{ \text{the set of atoms in } C \},$$

and note that for each C" in S", the clause C $\cup$ C" is an anti-clause of C, and hence

$$S' = \{ C \cup C'' : C'' \in S'' \} .$$

Since S" can be resolved to $\square$, it follows that S' can be resolved to C.

Thus S implies all members of S_F and S_F implies all members of S. But $S_F \subsetneq F_A$ and any subset of the full set is unsatisfiable if it equals the full set. Therefore, S is unsatisfiable if $S_F = F_A$. Q.E.D.

Note. It follows from the theorem that S is unsatisfiable if the length $(S_F) = 2^n$, where n is the number of atoms in S.

The following LISP program has been used to prove various sets of ground clauses. The table gives some computer times (using the DEC 2060) for various examples. It is accessed by a call to (PROVE-G S) where S is a set of ground clauses. The program is preceded by a table listing some computing times for various S's.

Some Computing Times

Exn represents the full set of n letters.

Exn' represents a set of 2" partially deleted clauses from the full set on n.

n	number of clauses	Computing Time (seconds) for Exn	Exn'	Exn1
1	2	.01	.003	
2	4	.03	.005	
3	8	.07	.02	
4	16	.19	.15	
5	32	.52	.62	
6	64	1.4	1.9	
7	128	3.8	5.7	
8	256	10.1	19.0	
9	512	28.5	66.6	
6	266	(each clause deleted to one literal)		9.1
6	12	(each clause deleted to one literal, no repetitions)		1.3

```
(DEXPR PROVE-G (S)
  (PROG (ATOMS-S SF TIME-START)
        (SETQ TIME-START (TIME))
        (SETQ ATOMS-S (SORT (ATOMS S)))
        (SETQ SF
              (UNION (MAPCAN '[LAMBDA (C)
                                (ANTI-CLAUSES (SORT C 'COMPARE-G) ATOMS-S)]
                             S)))
        (COND [PRINT-LIGHT (PRINT SF)])
        (PRINT (LIST 'TIME (DIFFERENCE (TIME) TIME-START)))
        (PRINT (LIST 'LENGTH-OF-SF (LENGTH SF)))
        (RETURN (EQUAL (LENGTH SF) (TWO-TO-THE (LENGTH ATOMS-S))))))

(DEXPR ANTI-CLAUSES (C ATMS)
  (PROG (L SFF)
        (RETURN (COND [(NULL ATMS) (LIST NIL)]
                      [(AND C
                            [OR [EQUAL (SETQ L (CAR C)) (CAR ATMS)]
                                [EQUAL L (LIST 'NOT (CAR ATMS))]])
                       (SETQ SFF (ANTI-CLAUSES (CDR C) (CDR ATMS)))
                       (MAPCAR '[LAMBDA (Q) (CONS L Q)] SFF)]
                      [T (SETQ SFF (ANTI-CLAUSES C (CDR ATMS)))
                         (MAPCAN '[LAMBDA (Q)
                                   (LIST (CONS (CAR ATMS) Q)
                                         (CONS (LIST 'NOT (CAR ATMS)) Q))]
                                 SFF)]))))

(DEXPR ATOMS (S)
  (UNION (MAPCAN '[LAMBDA (C)
                   (MAPCAN '[LAMBDA (L)
                             (COND [(ATOM L) (LIST L)]
                                   [(EQ 'NOT (CAR L)) (LIST (CADR L))]
                                   [T (LIST L)])]
                           C)]
                 S)))

(DEXPR COMPARE-G (X Y)
  (LEXORDER (COND [(ATOM X) X] [(EQ 'NOT (CAR X)) (CADR X)] [T X])
            (COND [(ATOM Y) Y] [(EQ 'NOT (CAR Y)) (CADR Y)] [T Y])))

(DEXPR TWO-TO-THE (N)
  (COND [(ZEROP N) 1] [T (TIMES 2 (TWO-TO-THE (SUB1 N)))]))

(DEXPR FULL-SET (N)
  {;; THIS GIVES THE FULL SET ON THE FIRST N LETTERS OF THE ALPHABET -- FOR
      EXAMPLE IF N IS 2 THEN IT RETURNS ((A B) (A (NOT B)) ((NOT A) B)
      ((NOT A) (NOT B))) (FULL-SET-L (FIRST-N N ALPHABET))}
  (FULL-SET-L (FIRST-N N ALPHABET)))

(DEXPR DELETE-P (S M)
  {;; THIS DELETES M ELEMENTS FROM RANDOM FROM EACH CLAUSE OF S}
  (PROG (RANN)
        (SETQ ATOMS-S (ATOMS S))
        (SETQ RANN
              (MAPCAN '[LAMBDA (Q)
                        (COND [(OR [MEMBER Q ATOMS-S]
                                   [AND [NOT (ATOM Q)]
                                        [MEMBER (CADR Q) ATOMS-S]])
                               (LIST Q)])]
```

```
                          RANDOM))
        (RETURN (MAPCAR '[LAMBDA (C) (DELETE-C C M RANN)] S))))

(DEXPR DELETE-C (C M RAN)
  {;; THIS DELETES THE FIRST M ELEMENTS OF RAN FROM C -- IF RAN IS EXAUSTED
      THEN IT IS RESET TO RANN (A PERMANENT LIST OF RANDOM ATOMS FROM S AND
      THEIR NEGATIONS)}
  (COND [(ZEROP M) C]
        [(NULL RAN) (DELETE-C C M RANN)]
        [T (DELETE-C (REMOVE (CAR RAN) C) (SUB1 M) (CDR RAN))]))

(DEFV PRINT-LIGHT NIL)

(DEFV RANDOM (A (NOT C) (NOT E) D H (NOT I) (NOT A) C G (NOT F) (NOT B) I
              (NOT H) (NOT D) (NOT G) E F B))

(DEXPR PROVE-FULL-SETS (N)
  {;; PROVE ALL FULL SETS UP TO THE FULL SET ON N}
  (PROG (M)
        (SETQ M 1)
    TOP (PRINT (LIST 'N '= M))
        (COND [(LESSP N M) (GO BOTTOM)]
              [T (PRINT (PROVE-G (FULL-SET (FIRST-N M ALPHABET))))
                 (SETQ M (ADD1 M))
                 (GO TOP)])
 BOTTOM (RETURN T)))

(DEFV ALPHABET (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z))

(DEXPR FIRST-N (N A)
  {;; THIS RETURNS THE FIRST N MEMBERS OF A}
  (COND [(ZEROP N) NIL] [T (CONS (CAR A) (FIRST-N (SUB1 N) (CDR A)))]))

(DEXPR PROVE-DELETED-SETS (N)
  {;; THIS TAKES THE FULL SETS UP TO THE FULL SET ON N AND DELETES ABOUT HAL
      OF THE ELEMENTS (MORE OR LESS RANDOMLY) FROM EACH CLAUSE AND PROVES TH
      RESULTING SET}
  (PROG (K)
        (SETQ K 1)
    TOP (PRINT (LIST 'N '= K))
        (COND [(LESSP N K) (GO BOTTOM)]
              [T (PRINT (PROVE-G (DELETE-P (FULL-SET (FIRST-N K ALPHABET))
                                           (QUOTIENT N 2))))
                 (SETQ K (ADD1 K))
                 (GO TOP)])
 BOTTOM (RETURN 'END)))

(DEXPR FULL-SET-L (L)
  {;; THIS GIVE THE FULL-SET ON THE LETTERS OF L -- FOR EXAMPLE IF L IS
      (A B) THEN IT RETURNS ((A B) (A (NOT B)) ((NOT A) B) ((NOT A)
      (NOT B)))}
  (PROG (L1 L2 L3)
        (COND [(NULL L) (RETURN NIL)]
              [(NULL (CDR L))
               (RETURN (LIST (LIST (CAR L))
                             (LIST (COND [(NUMBERP (CAR L)) (MINUS (CAR L))]
                                         [T (LIST 'NOT (CAR L))]))))])
        (SETQ L1 (FULL-SET-L (CDR L)))
        (SETQ L2 (MAPCAR '[LAMBDA (Q) (CONS (CAR L) Q)] L1))
        (SETQ L3
```

```
            (MAPCAR '[LAMBDA (Q)
                      (CONS (COND [(NUMBERP (CAR L)) (MINUS (CAR L))]
                                  [T (LIST 'NOT (CAR L))]])
                            Q)]
                    L1))
        (RETURN (APPEND L2 L3))))

(DEXPR SUBSUME-P (S)
  {;; THIS SELECTS A MINIMAL SUBSET S' OF S WHICH SUBSUMES S}
  (COND [(NULL S) NIL]
        [(SOME '(LAMBDA (C) (SUBSET-N C (CAR S))) (CDR S)) (SUBSUME-P (CDR S]
        [T (CONS (CAR S)
                 (SUBSUME-P (SUBSET '(LAMBDA (C) (NOT (SUBSET-N (CAR S) C)))
                                    (CDR S))))]))

(DEXPR SUBSET-N (A B)
  {;; THIS IS THE ORDINARY SUBSET PREDICATE}
  (EVERY '(LAMBDA (X) (MEMBER X B)) A))

(DEXPR SUBSUME-N (S1 S2)
  {;; THIS SELECTS THOSE MEMBERS OF S2 NOT SUBSUMED BY A MEMBER OF S1}
  (MAPCAN '[LAMBDA (C')
             (COND [(SOME '(LAMBDA (C) (SUBSET-N C C')) S1) NIL] [T (LIST C').
           S2))

(NOCOMPILE
(DEFV GPVRFNS (PROVE-G ANTI-CLAUSES ATOMS COMPARE-G TWO-TO-THE FULL-SET
               DELETE-P DELETE-C PRINT-LIGHT RANDOM PROVE-FULL-SETS ALPHABET
               FIRST-N PROVE-DELETED-SETS FULL-SET-L SUBSUME-P SUBSET-N
               SUBSUME-N))
)
```