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 subsummed 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,

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

**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 $A$, 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 $S_F \subseteq F_A$ 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 $(S_p) = 2^n$, 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.

<table>
<thead>
<tr>
<th>n</th>
<th>number of clauses</th>
<th>Computing Time (seconds) for</th>
<th>Exn</th>
<th>Exn'</th>
<th>Exn1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.01</td>
<td>.003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>.03</td>
<td>.005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>.07</td>
<td>.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>.19</td>
<td>.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>.52</td>
<td>.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>1.4</td>
<td>1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>3.8</td>
<td>5.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>10.1</td>
<td>19.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>28.5</td>
<td>66.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>266</td>
<td>(each clause deleted to one literal)</td>
<td>9.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>(each clause deleted to one literal, no repetitions)</td>
<td>1.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(DEXPRESS PROVE-G (S))
(DEXPRESS ANTI-CLAUSES (C ATMS))
(DEXPRESS ATOMS (S))
(DEXPRESS COMPARE-G (X Y))
(DEXPRESS TWO-TO-THE (N))
(DEXPRESS FULL-SET (N))
(DEXPRESS DELETE-P (S M))
(DEFEXPR DELETE-C (C M RAN)
  (COND ((NULL RAN) (DELETE-C C M RANN))
    [(T (DELETE-C (REMOVE (CAR RAN) C) (SUB1 M) (CDR RAN))))]
  ))

(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))

(DEFEXPR PROVE-FULL-SETS (N)
  (PROG (M)
    (SETQ M 1)
    TOP (PRINT (LIST "\* = 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))

(DEFEXPR FIRST-N (N A)
  (COND ((ZEROP N) NIL) [(T (CONS (CAR A) (FIRST-N (SUB1 N) (CDR A))))])
  ))

(DEFEXPR PROVE-DELETED-SETS (N)
  (COND [(ZEROP N) NIL]
    [(T (CONS (CAR A) (FIRST-N (SUB1 N) (CDR A))))]
    [(T (PRINT (PROVE-G (FULL-SET (FIRST-N K ALPHABET))
          (QUOTIENT N 2)))]
    (SETQ K (ADD1 K))
    (GO TOP)])
  )
)

(DEFEXPR FULL-SET-L (L)
  (COND [(NOT 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))))

(DEEXPR 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)))]))]]))

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

(DEEXPR 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)))

)