Ex. 12. $(\forall y(y < \ell \rightarrow \exists z(y < z \leq b)) \wedge a \leq \ell \rightarrow a \leq b)$.

1. $z_y \leq b \vee \ell \leq y$
2. $y < z_y \vee \ell \leq y$
3. $a \leq \ell$
4. $b < a$ (top clause)

TY: $[a \leq \ell,\ b < a]$

5.	$y < z_y \vee b < y$	4, 3, 2, CHAIN‡
6.	$y < b \vee \ell \leq y \vee b < y$	5, 1, CHAIN‡
7.	$\ell \leq b \vee b < b$	6, eliminate y
8.	$\ell \leq b$	7, REDUCE
9.	$\square$	8, TY

---

‡Note that this step falls under the RL-CASE of CHAIN (see RL-CASE, Section 3).

Ex. 13. $(\forall \epsilon\ (0 < \epsilon \rightarrow A \leq B + \epsilon) \rightarrow A \leq B)$

1. $A \leq B + \epsilon \vee \epsilon \leq 0$ (ϵ is a variable)

2. $B < A$ (top clause)

TY: $[B < A]$

3. $0 \leq B - A$ 1, eliminate variable ϵ

4. $\square$ 3, TY

In this example, the literal $A \leq B + \epsilon$ is rewritten as $A - B \leq \epsilon$ so that ϵ can be seen as eligible in clause 1, and then eliminated, leaving $0 \leq B - A$.

Ex. 14. $(\forall \epsilon\ (\epsilon > 0 \rightarrow A \leq B + \epsilon) \wedge B \leq C \rightarrow A \leq C)$

1. $A \leq B + \epsilon \vee \epsilon \leq 0$

2. $B \leq C$

3. $C < A$

TY: $[B \leq C,\ C < A]$

4. $0 \leq B - A$ 1, eliminate ϵ

5. $\square$ 4, TY

Ex. 15. $\exists \epsilon[(\epsilon > 0 \rightarrow A \leq B_\epsilon + \epsilon) \wedge B_\epsilon \leq C \rightarrow A \leq C]$

1. $A \leq B_\epsilon + \epsilon \vee \epsilon \leq 0$
2. $B_\epsilon \leq C$
3. $C < A$ (top clause)

TY: $[C < A]$

4. $A \leq C + \epsilon \vee \epsilon \leq 0$ 3, 1, 2, CHAIN

(See Section 3, RL-CASE, Case DIR = T, E $\neq$ 0)

5. $0 \leq C - A$ 4, eliminate ϵ
6. $\square$ 5, TY

Ex. 16. $\exists \epsilon[(\epsilon > 0 \rightarrow A_\epsilon \leq B_\epsilon + \epsilon) \wedge B_\epsilon \leq C \rightarrow A_\epsilon \leq C]$ False.

1. $A_\epsilon \leq B_\epsilon + \epsilon \vee \epsilon \leq 0$
2. $B_\epsilon \leq C$
3. $C < A_\epsilon$

---

4. $A_\epsilon \leq C + \epsilon$ 3, 1, 2, CHAIN

But now we cannot eliminate ϵ as we did in Ex. 15.

Ex. 17. $[\forall \epsilon(\epsilon > 0 \rightarrow f(\ell) \leq f(z_\epsilon) + \epsilon \wedge f(z_\epsilon) \leq f(t_0)) \rightarrow f(\ell) \leq f(t_0)]$

1. $f(\ell) \leq f(z_\epsilon) + \epsilon \vee \epsilon \leq 0$
2. $f(z_\epsilon) \leq f(t_0) \vee \epsilon \leq 0$
3. $f(t_0) < f(\ell)$ (top clause)

TY: $[f(t_0) < f(\ell)]$

4. $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0$ 3, 1, 2, CHAIN
5. $0 \leq f(t_0) - f(\ell)$ 4, Eliminate ϵ
6. $\square$ 5, TY

Ex. 18. $\forall z(P(z,c_1) \vee P(z,c_2)) \rightarrow \exists x \exists y\,(P(x,y) \wedge a(y) < x)$.

1. $P(z,c_1) \vee P(z,c_2)$
2. $\sim P(x,y) \vee a(y) \not< x$ (top clause)

3. $P(z,c_2) \vee a(\ell) \not< x$ 1, 2, z/x, c_1/y
4. $a(c_1) \not< z \vee a(c_2) \not< z$ 3, 2, z/x, c_2/y
5. $\square$ 4, eliminate z

When a new resolvent is computed it is processed and used as the new top clause. It is simplified by REDUCE and duplicate literals are merged, eligible literals are eliminated, and the clause is split if possible. If the result is a ground unit clause, it is resolved with TY and checked for contradiction. Finally, it is sorted, with RL-literals put last, before being used as the top clause in a new call to RESOLVE-CHAIN (see PROCESS-RESOLVENT, Section 3).

In Ex. 19 and later examples, we indicate the literal being resolved upon by outlining it with a rectangle, $\boxed{\text{L}}$. This is usually the first literal of the top clause after it has been processed and sorted.

Ex. 19. $[a \le \ell \le b \wedge a \le t_0 < \ell$

$\wedge \forall \epsilon(\epsilon > 0 \to \exists r(r < \ell \wedge \forall s(r \le s \le \ell \to f(\ell) \le f(s) + \epsilon)))$

$\wedge \forall y(a \le y \le \ell \to \exists z(y < z \le \ell \wedge \forall t(a \le t \le z \to f(z) \le f(t))))$

$\to f(\ell) \le f(t_0)]$

1. $a \le \ell$
2. $\ell \le b$
3. $a \le t_0$
4. $t_0 < \ell$
5. $r_\epsilon < \ell \quad \vee \epsilon \le 0$
6. $f(\ell) \le f(s) + \epsilon \vee \epsilon \le 0 \vee s < r_\epsilon \vee \ell < s$
7. $y < z_y \quad \vee y < a \vee \ell \le y$
8. $z_y < \ell \quad \vee y < a \vee \ell \le y$
9. $f(z_y) \le f(t) \quad \vee y < a \vee \ell \le y \vee t < a \vee z_y \le t$
10. $f(\ell) \not\le f(t_0)$ (top clause)

TY: $[a \le \ell,\ \ell \le b,\ a \le t_0,\ t_0 < \ell,\ f(t_0) < f(\ell)]$

11. $f(\ell) \le f(t_0) + \epsilon \vee \epsilon \le 0 \vee z_y < r_\epsilon \vee \ell < z_y$

$\vee y < a \vee \ell \le y \vee t_0 < a \vee z_y \le t_0$ 10, 6, 9, CHAIN

Clause 11 splits on $(t_0 < a)$ into clauses 12.1 and 12.2.

12.1 $t_0 < a$

13. $\square$ 12.1, TY

12.2 $f(\ell) \le f(t_0) + \epsilon \vee \epsilon \le 0 \vee \boxed{z_y < r_\epsilon} \vee \ell < z_y$

$y < a \vee \ell \le y \vee z_y \le t_0$

13. $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0 \vee y < r_\epsilon \vee \ell < z_y$
$\vee y < a \vee \ell \leq y \vee z_y \leq t_0$ — 12.2, 7, CHAIN

14. $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0 \vee y < r_\epsilon \vee \ell < z_y$
$\vee y < a \vee \ell \leq y \vee y < t_0$ — 13, 7, CHAIN

15. $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0 \vee y < r_\epsilon$
$\vee y < a \vee \ell \leq y \vee y < t_0$ — 14, 8

16. $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0$
$\vee \ell \leq r_\epsilon \vee \ell \leq a \vee \ell \leq t_0$ — 15, eliminate y

Clause 16 splits on $(\ell \leq a)$ and $(\ell \leq t_0)$ into clauses 17.1, 17.2, and 17.3

17.1 $\ell \leq a$

18. $\square$ — 17.1, TY, sub = , $a = \ell = t_0$

17.2 $\ell \leq t_0$

19. $\square$ — 17.2, TY, sub = , $\ell = t_0$

17.3 $f(\ell) \leq f(t_0) + \epsilon \vee \epsilon \leq 0 \vee \ell \leq r_\epsilon$

20. $f(\ell) \leq f(t_0) + \epsilon \quad \epsilon \leq 0$ — 17.3, 5

21. $f(\ell) \leq f(t_0)$ — 20, eliminate ϵ

22. $\square$ — 21, TY

Ex. 20. $\exists a \exists b \forall x \exists u([(x < b \to u \leq a) \wedge x \leq u] \wedge (u \leq a \vee x \neq b))$

1. $x_0 < b \vee x_0 \not\leq u \cdot u \not\leq a$
2. $x_0 < b \vee x_0 \not\leq u \vee x_0 = b$
3. $u \not\leq a \vee x_0 \not\leq u \vee u \not\leq a$
4. $u \not\leq a \vee x_0 \not\leq u \vee x_0 = b$

a, b, u are variables, x_0 stands for the expression x_{ab}

5. $x_0 < b \vee x_0 \leq a$ — 1, eliminate u
6. $x_0 < b \vee x_0 = b$ — 2, eliminate u
7. $x_0 \not\leq a$ — 3, eliminate u
8. $x_0 \not\leq a \vee x_0 = b$ — 4, eliminate u

   (subsumed by 7)
9. $b \not\leq a \vee x_0 < b$ — 7, 6, sub =
10. $b \not\leq a$ — 9, 7, CHAIN
11. $\square$ — 10, eliminate a (or b).

Examples 20 and 21 arise in the search for counterexamples in a proof.

Ex. 21. $\exists a \exists b \exists \ell \exists w \forall s \exists u$

$(a \leq b \wedge [u \leq 0 \vee s \neq a] \wedge [0 \leq u \vee s \neq b] \wedge [s \leq w \vee 0 < u] \wedge w \leq \ell)$

...

a, b, ℓ, and w are variables, and s_0 stands for $s_{ab\ell w}$.

1. $b < a \vee 0 < u \vee u < 0 \vee w < s_0 \vee \ell < w$
2. " " " $\vee u \leq 0$ "
3. " " $s_0 = b \vee w < s_0$ "
4. " " " $\vee u \leq 0$ "
5. " $s_0 = a \vee u < 0 \vee w < s_0$ "
6. " " " $\vee u \leq 0$ "
7. " " $s_0 = b \vee w < s_0$ "
8. " " " $\vee u \leq 0$ "

Preprocessing removes clauses 2 and 4 which are tautologies, and eliminates the variable u from the other clauses.

1. $b < a \vee 0 < 0 \vee w < s_0 \vee \ell < w$
3. " $s_0 = b \vee w < s_0$ "
5. " $\vee s_0 = a \vee w < w_0$ "
6. " " "
7. " " $s_0 = b \vee w < s_0$ "
8. " " $s_0 = b$ "

The literal $(0 < 0)$ is removed from clause 1 by REDUCE. Then clause 3 is subsumed by clause 1, and clauses 5, 7, and 8 are subsumed by clause 6.

Thus preprocessing reduces these eight clauses to two.

1. $w < s_0 \vee b < a \vee \ell < w$ (top clause)
6. $s_0 = a \vee b < a \vee \ell < w$

9.	$w < a \vee b < a \vee \ell < w$	1, 6, sub =
10.	$\ell < w$	9, eliminate a
11	$\square$	10, eliminate ℓ

This example shows the power of variable elimination in reducing a messy, but not hard, problem to manageable size.

Ex. 22. $(\forall x(f(x) \geq 0) \wedge \forall s(f(s) \geq 0 \wedge \ell \leq s \rightarrow t \leq s) \rightarrow t \leq \ell)$

1.	$f(x) \geq 0 \vee x \leq \ell$	(top clause)
2.	$f(s) < 0 \vee s < \ell \vee t_0 \leq s$	
3.	$\ell < t_0$	

TY: $[\ell < t_0]$

4.	$s < \ell \vee t_0 \leq s \vee s \leq \ell$	1, 2, s/x
5.	$t_0 \leq \ell$	4, eliminate s
6.	$\square$	5, TY

Notice that in this example clause 3 cannot serve as top clause, although one would naturally choose it because it represents the conclusion of the theorem.

Ex. 23. $f(\ell) < 0 \wedge f(b) \geq 0 \wedge \forall s(0 \leq f(s) \wedge \ell \leq s \rightarrow t \leq s)$

$\wedge \forall x(x \leq b \wedge f(x) \leq 0 \rightarrow x \leq \ell)$

$\wedge \forall y(\forall z(z \leq b \wedge f(z) \leq 0 \rightarrow z \leq y) \rightarrow \ell \leq y) \rightarrow t \leq \ell$

1. $f(\ell) < 0$
2. $0 \leq f(b)$
3. $f(s) < 0 \vee s < \ell \vee t_0 \leq s$ (top clause)
4. $0 < f(x) \vee b < x \vee x \leq \ell$
5. $z_y \leq b \vee \ell \leq y$
6. $f(z_y) \leq 0 \vee \ell \leq y$
7. $y < z_y \vee \ell \leq y$
8. $\ell < t_0$

TY: $[f(\ell) < 0,\ 0 \leq f(b),\ \ell < t_0]$

Note: Clause 8 will not succeed as top clause.

9. $s < \ell \vee t_0 \leq s \vee b < s \vee s \leq \ell$ 3, 4, s/x
10. $b \leq \ell \vee t_0 \leq \ell$ 9, eliminate s

SPLIT 10 into 10.1 and 10.2

10.1 $b \leq \ell$

TY: $[f(\ell) < 0,\ 0 \leq f(b),\ \ell < t_0,\ b \leq \ell]$

11. $\ell < y \vee \ell \leq y$ 10,1, 5, 7, CHAIN
12. $\square$ 11, eliminate y

10.2 $t_0 \leq \ell$

13. $\square$ 10.2, TY

Ex. 24. $a''(z_0) < b''(z_0) \wedge \forall u(a(u) < b(u) \wedge a'(u) < z_0 < b'(u))$

$\rightarrow \exists x \exists y \exists z\ (a'(y) < z < b'(x) \wedge (a(x) < y < b(x) \wedge a''(z) < x < b''(z))$

1. $a'(y) \not< z \vee z \not< b'(y) \vee a(x) \not< y \vee y \not< b(x)$
   $\vee\ a''(z) \not< x \vee x \not< b''(z)$
2. $a''(z_0) < b''(z_0)$
3. $a(u) < b(u)$
4. $a'(u) < z_0$
5. $z_0 < b'(u)$

x, y, z, and u are variables, and clause 1 is the top clause. Notice that no variable is eligible in clause 1.

6. $z_0 \not< b'(y) \vee a(x) \not< y \vee y \not< b(x)$ $\vee\ a''(z_0) \not< x \vee x \not< b''(z_0)$ — 1, 4, y/u, z_0/z
7. $a(x) \not< y \vee y \not< b(x) \vee a''(z_0) \not< x \vee x \not< b''(z_0)$ — 6, 5, y/u

Now y is eligible in 7.

8. $a(x) \not< b(x) \vee a''(z_0) \not< x \vee x \not< b''(z_0)$ — 7, eliminate y
9. $a''(z_0) \not< x \vee x \not< b''(z_0)$ — 8, 3, x/u

Now x is eligible in 9.

10. $a''(z_0) \not< b''(z_0)$ — 9, eliminate x
11. $\square$ — 10, 2.

5. Computer Implementation and Results

These ideas were tested by a program written in LISP and run on the DEC 10 computer at The University of Texas at Austin. In that program we varied somewhat the algorithms described in Section 3. For example, in step 2 of the algorithm RESOLUTION-CHAIN, we allowed L to be only the first literal of the top clause C. This "first literal only" strategy is incomplete as can be seen by rearranging the literals of clause 1 of Ex. 24. However, we feel that examples like Ex. 24 are contrived and unlikely in "normal applications." Much work remains to be done to refine these procedures.

Ex. 1 - Ex. 24 were all proved by this program. We would be interested to know whether other automatic provers are able to prove these examples, especially Ex. 19, 21, 23, 24, and also Ex. 5, 6, 7 of [6].

6. Completeness

Completeness is not claimed for this system. For example, the ability to handle algebraic expressions must be further enhanced if it is to handle complicated polynomial terms.

A series of completeness results might be proved to show that the basic mechanisms of this system are correct (if indeed they are).

For example, the variable elimination rule, Section 2.2, requires that the variable being eliminated be "eligible." Then an interpolation axiom is used (in effect) to eliminate the variable. The underlying philosophy is that one can delay the use of the interpolation axiom until after the variable becomes eligible. To make this more precise, consider the clauses

1. $P(x) \vee a \not< x \vee x \not< b$,

2. $\sim P(x)$,

3. $a < b$

and the interpolation axioms

CI1 $x \not< y \vee x < w(x,y)$

CI2 $x \not< y \vee w(x,y) < y$.

We can proceed to $\square$ in two different ways: by using CI1, CI2 first or last.

4.	$P(w(a,b)) \vee a \not< b$	1, CI1, CI2
5.	$a \not< b$	4, 2
6.	$\square$	5, 3

or

4.	$a \not< x \vee x \not< b$	1, 2
5.	$a \not< b$	4, CI1, CI2
6.	$\square$	5, 3

Our contention is that we can always delay using CI1, CI2 until last, as we did here, or at least delay until x becomes eligible. The statement of this contention is given (but not proved) as Theorem 1.

Theorem 1. Suppose that S is an unsatisfiable set of clauses which contains the interpolation clauses CI1 and CI2 and which does not otherwise contain the symbol w. Then there is a deduction of $\square$ from S in which the function symbol "w" does not occur in a resolvent, except in cases where it is introduced in one step by resolving on one of CI1 or CI2 and eliminated in the next step by resolving on the other.

Of course, if Clauses 2 and 3 were replaced by

2. $\sim P(c)$,

3. $a < c$,

4. $c < b$,

then CI1, CI2 could not be (profitably) used at all and then delaying is even more desirable.

Another argument (perhaps the main argument) for delaying the use of the interpolation axioms is that a premature use of CI1, CI2 might block their further use when more than one use is required. Consider the following example.

1. $P(x) \vee a \not< x \vee x \not< b$

2. $\sim P(s) \vee c \not< s$

3. $a < b$

4. $c < b$

Since x is not eligible in Clause 1, we prefer to first resolve it, upon P(x), with Clause 2, getting

5. $c \not< x \vee a \not< x \vee x \not< b$

and then eliminate it

6. $c \not< b \vee a \not< b$ 5

7. $\square$ 6, 3, 4

If we had tried to use CI1, CI2 first we would get

5. $P(w(a,b)) \vee a \not< b$ 1, CI1, CI2

6. $P(w(a,b))$ 5, 3

7. $c \not< w(a,b)$ 6, 2

But this is blocked.

As a general principle, if we delay using the interpolation axioms until the variable being interpolated becomes <u>eligible</u>, then we collect together <u>all</u> the requirements on x, and eliminate them all at once. If in this process x is instantiated by a non-variable term, then the interpolation axiom could not have been used anyway.

7. Comments

The Special Clause TY

TY is used simply to collect together all ground inequality literals. (Because of splitting, ground literals can only occur in unit clauses.) One could get the same effect by not using TY at all but instead collecting together all ground inequality literals each time a new ground inequality is produced as a resolvent and checking for a contradiction. We prefer the TY arrangement because it lets us use the sup-inf procedures of [11] to speedily process ground inequalities. A similar speed advantage can be obtained by the use of the ground inequality packages of [4] and [5].

In any of these methods, a set of ground equality units might be inferred by TY, and these are applied to TY and S by an equality substitution mechanism.

If one did not use splitting, then a method could be devised whereby special TY-literals (a conjunction of ground inequality literals) would occur in clauses.

Chaining

Inequality chaining (see CHAINER, Section 3) is required for many of the examples given in Section 4 (unless one is to resort to using the transitivity axioms again), but it tends to enlarge the search space when it is used. We envision a system that uses chaining but in a sparing way under the control of heuristics.

The chaining we employ is, of course, similar to that used by Slagle and Norton [2, 12], except that we do not chain across variables (see p. 17), and also we do not retain the intermediate links in the chain as they do. For instance, in the example

$$f(\ell) \leq f(z_y) \vee D(y)$$

$$f(z_a) \leq L$$

$$L < f(\ell) \vee E$$

We do not retain the (intermediate) resolvents

$$f(\ell) \leq L \vee D(a)$$

and

$$f(z_a) < f(\ell) \vee E$$

but only

$$D(a) \vee E.$$

In this way it is somewhat like hyper-resolution.

Fast Implementation

Our intention here was not to present the best and fastest Resolution system but rather to present a few concepts that could make an existing Resolution prover faster for inequality proofs. We have not used here but would highly recommend such things as

- a pointer system which lets relevant clauses be accessed quickly [13],
- mechanisms that resolve more than one clause at once, and avoid the (explicit) use of substitutions [13, 14, 22],
- fast unifiers [15],
- interconnectivity graphs [16-20], which precompute various unification chains.

Some of these concepts are embodied in the provers of Overbeek-Wos-Lusk-Winker. Of course, other powerful resolution procedures such as Model Elimination [22] and SL-Resolution [23] might be used.

Hyper-resolution and Counterexamples

Unfortunately, ordinary hyper-resolution is not compatible with our chaining and variable-elimination techniques. For instance, the clauses

1. $z_y \leq t_0$

2. $y \leq z_y$

resolve to $\square$ as follows

3. $y \leq t_0$	1, 2
4. $\square$	3, eliminate y

and yet there are no negative clauses at all. (Of course, if we included the interpolation axioms we would have at least one negative clause.)

A similar problem arises in trying to use models (counterexamples)) because that is just an extended form of hyper-resolution [21]. We expect to overcome this problem to some extent in a later paper.

Acknowledgement

This work was supported by NSF Grant MCS 77-20701. Thanks to Mike Ballantyne for his ideas and help.

References

1. J. A.Robinson. A Machine-oriented Logic Based on the Resolution Principle. J. ACM 12 (1965), 23-41.

2. J. R. Slagle. Automatic Theorem Proving with Built-in Theories Including Equality, Partial Ordering, and Sets. J. ACM 19 (1972), 120-135.

3. J. C. King. A Program Verifier. Ph.D. thesis, Carnegie-Mellon University, 1969.

4. Greg Nelson and Derek Oppen. A Simplifier Based on Efficient Decision Algorithms. Proc. 5th ACM Symp. on Principles of Programming Languages, 1978. (Also: Simplification by Cooperating Decision Procedures, Stanford A. I. Memo.)

5. Robert Shostak. A Practical Decision Procedure for Arithmetic with Function Symbols. J. ACM, April 1979.

6. W. W. Bledsoe, Peter Bruell, and Robert Shostak. A Prover for General Inequalities. The Univ. of Texas Math. Dept. Memo ATP 40A, Feb. 1979. Also IJCAI-79, Tokyo, Japan, Aug. 1979.

7. Louis Hodes. Solving Programs by Formula Manipulation in Logic and Linear Inequality. Proc. IJCAI-71, London (1971), pp. 553-559.

8. W. W. Bledsoe. A Maximal Method for Set Variables in Automatic Theorem Proving. Machine Intelligence 9 (eds. J. E. Hayes, D. Michie, L. I. Mikvlich), Chichester: Ellis Horwood Ltd. and New York: John Wiley. July 1979, 53-100.

9. L. Wos and G. A. Robinson. Paramodulation and Set of Support. Proc. Symp. Automatic Demonstration, Versailles, France. Springer-Verlag, New York (1968), 276-310.

10. W.W. Bledsoe. Splitting and Reduction Heuristics in Automatic Theorem Proving. A.E. Jour. 2 (1971), 55-77.

11. W. W. Bledsoe and Mabry Tyson. Typing and Proof by Cases in Program Verification. Machine Intelligence 8 (eds. D. Michie and E. W. Elcock), Chichester: Ellis Horwood Limited. 1977, 30-51.

12. J. R. Stagle and L. Norton. Experiments with an Automatic Theorem Prover Having Partial Ordering Rules. Commun. ACU 16 (1973), 682-688.

13. R. A. Overbeek. An Implementation of Hyper-resolution, Computer Math. Appl. 1 (1975), 201-214.

14. R. S. Boyer and J S. Moore. The Sharing of Structure in Theorem-proving Programs. Machine Intelligence 7 (eds. B. Meltzer and D. Michie), Edinburgh Univ. Press, Edinburgh (1972), 101-116.

15. J. A. Robinson.

16. R. Kowalski. A Proof Procedure Using Connection Graphs. J. ACM 22 (1975), 572-595.

17. S. Sickel. A Search Technique for Clause Interconnectivity Graphs. IEEE Trans. on Computers C-25 (1976), 823-835.

18. P. B. Andrews. Refutations by Matings. IEEE Trans. on Computers C-25 (1976), 801-807.

19. R. E.Shostak. Refutation Graphs. Artificial Intelligence 7 (1976), 51-64.

20. P. T. Cox. Representational Economy in a Mechanical Theorem-prover. Proc. Fourth Workshop on Automatic Deduction, Austin, Texas, Feb. 1979, 122-128.

21. J. R.Slagle. Automatic Theorem Proving with Renamable and Semantic Resolution. J. ACM 14 (1967), 687-697.

22. Donald W. Loveland. Automated Theorem Proving: A Logical Basis. North Holland, 1978. Mechanical Theorem Proving by Model Elimination, J. ACM 15 (1968), 236-251.

23. Robert Kowalski and Donald Kuehner. Linear Resolution with Selector Function. AI Jour. 2 (1971), 227-260.