Association of Automated Reasoning Newsletter No. 3

September 1984

From the President

JAR

The *Journal of Automated Reasoning* is on schedule. It will commence publication early in 1985, on a quarterly basis. The cost will be $78 for institutions, $36 for private non-members of AAR, and $27 for private members of AAR.

We are actively seeking papers in all areas of automated reasoning, including automated theorem proving, logic programming, expert systems, program synthesis and validation, artificial intelligence, computational logic, and robotics. We expect the journal to serve as a forum for exchanging information for those interested purely in theory, those interested primarily in software, and those interested in specific applications. For example, we have already accepted articles on the following topics:

- Mechanical transformation of program executions to derive concurrency, by Christian Lengauer. Concurrency is derived by transforming the sequential execution of a program into an equivalent concurrent execution on the basis of formal transformation rules. The paper contains a proof, using the Boyer-Moore theorem prover, of the equivalence of the executions for an example involving sorting networks.

- Deduction in non-Horn databases, by Larry Henschen and Adnon Yahya. This paper studies the use of theorem-proving techniques in databases containing non-Horn clauses. An improved algorithm is given for certain kinds of negative information. Query answering in non-Horn databases is compared with the Horn case.

- ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system, by James S. Bennett. This paper describes the structure of the ROGET system and demonstrates how it might have assisted with the acquisition of the conceptual structure of the MYCIN system. Use of ROGET for acquiring additional forms of expertise is also discussed.

Questions about manuscript submissions may be addressed to the editor-in-chief

L. Wos
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439

Tel.: (312) 972-7224 (office)
(312) 493-0767 (home)

Successes with Program Verification Systems

Two recent successes with program verification systems deserve mention. Boyer and Moore have verified an NSA encryption algorithm currently in use, and Good has verified a 4200-line GYPSY program that is an encryption packet interface to the ARPANET.

Future Newsletters

So that the future newsletters can keep our members well informed, please send any appropriate announcements to us at Argonne National Laboratory. For example, we shall be pleased to announce current research, new software, experimental results, new books, new problems for testing programs, and descriptions of open questions.

New Books

Several new books have come to our attention:

• *The Computer Modelling of Mathematical Reasoning*, by Alan Bundy, published by Academic Press. This book represents a successful attempt to unify various approaches to the application of artificial intelligence ideas, particularly in the area of deduction, to the process of doing mathematics. Along the way, it gives very clear introductions to several topics in automated reasoning, including basic logic, higher order logics, resolution-based theorem proving, search strategies, and rewrite rules. It gives several examples of systems that apply these ideas to mathematics.

• *Micro-Prolog: Programming in Logic*, by K. L. Clark and F. G. McCabe, published by Prentice-Hall. This is a tutorial introduction to logic programming and Prolog in particular. The syntax used is that of the Prolog system micro-Prolog, which can be obtained for a variety of microcomputers. The book includes applications of Prolog to critical-path analysis, expert systems, games, and problem-solving. It requires no background in logic or programming.

• *Implementations of PROLOG*, edited by J. A. Campbell, and published by Ellis Horwood Limited. This book is a collection of papers on implementation-related issues in Prolog. Most of the papers assume familiarity with Prolog.

• *Automated Reasoning: Introduction and Applications* by Wos, Overbeek, Lusk, and Boyle, published by Prentice-Hall in February 1984. This book is available in both soft cover and hard cover. It assumes no background, discusses various applications, and contains numerous examples and exercises. The applications include circuit design, circuit validation, program verification, and research in mathematics and in formal logic.

New Journal Announced

A new journal, the *Journal of Symbolic Computation*, will begin publication in early 1985. Published by Academic Press, the journal is intended to provide a forum for research in the algorithmic treatment of all types of symbolic objects -- formulae, terms, programs, and algebraic and geometric objects.

Three basic aspects will be emphasized: mathematical foundations, correctness, and complexity of new (sequential and parallel) algorithms; implementation of the algorithms in software systems; and applications of the systems as tools for problem solving in the mathematical and natural sciences.

Manuscripts should be sent in triplicate to the editor-in-chief

B. Buchberger
Journal of Symbolic Computation
Johannes-Kepler-Universität
A4040 Linz
Austria
Tel.: Austria (732) 232381-9219
Telex: 2-2323 uni li a

Call for Manuscripts - EUROCAL '85

A call for papers has been issued for the European Conference on Computer Algebra to be held at Linz, Austria, on April 1-3, 1985.

Manuscripts are welcome on topics in computer algebra, including simplification of algebraic and transcendental terms; symbolic integration, summation, solution of differential equations; exact computation of zeros; and computational numbers, group, and ring theory. A special emphasis of EUROCAL '85 will be on the interaction of computer algebra with related areas, for example,

+ Manipulation of abstract data type specifications
+ Computer-aided program verification, synthesis, and transformation
+ Parallel algorithms and hardware for symbolic computation
+ Integration of computer algebra systems into expert systems

The deadline for submission of papers (maximum 12 pages) is November 15, 1984. Extended abstracts (maximum 2 pages) and systems demonstrations are also welcome; these are due January 15, 1985. Four copies of the contributions should be sent to the program chairman B. Buchberger (address given above).

On Efficient Unification without Occur Check
(from David A. Plaisted)

Since the introduction of the unification algorithm by Robinson in 1965, many improvements in efficiency have been made (see, for example, Paterson and Wegman, JCSS 16, pp. 18-167, 1978). We present another improvement that permits the occur check to be omitted in many cases. The occur check is necessary when unifying a variable x with a term t; it is necessary to check if x occurs in t so that, for example, x and $f(x)$ do not unify. Most Prolog implementations do not use true unification, but instead perform unification without occur check for efficiency reasons; this can create terms with loops in them. (For an introduction to Prolog, see Clocksin and Mellish, *Programming in PROLOG*, Springer-Verlag, Berlin, 1981.) Plaisted, in a presentation at the 1984 Symposium on Logic Programming at Atlantic City, showed how true unification can be simulated in Prolog by unification without occur check except in a few cases, in typical Prolog programs. These same ideas apply to any application where unification is necessary, such as theorem proving and pattern matching. However, this application is not made clear by Plaisted's paper. We briefly mention how these ideas can be applied in general, possibly leading to significant speedups in resolution theorem provers. Similar ideas were presented by Stickel at the Atlantic City symposium.

Definition. A literal L is *linear* if it has no repeated variables. Thus $P(f(x),y,g(z))$ is linear but $P(f(x),y,g(x))$ is not.

Proposition. Suppose L and M are literals and at least one of L and M is linear. Suppose L and M have no common variables. Then when unifying L and M, unification without occur check is equivalent to unification with occur check.

This observation may be applied to theorem provers. One can keep a flag with each literal of a clause telling whether it is linear. Then when unifying two literals from different clauses during resolution, if one of the literals is linear, unification without occur check may be performed. This is correct because variables from different clauses are renamed before such a unification is performed. However, it is still necessary to detect if a literal is linear, which may require searching through the entire structure of the literal. This search can also be eliminated in many cases, as we now show.

Proposition. Suppose L_1, L_2, and M_1 are literals and L_2 and M_1 are linear. Suppose M_1 has no common variables with L_1 or L_2. Let σ be the most general unifier of L_1 and M_1. Then $L_2\sigma$ is linear.

Corollary. Suppose C and D are two clauses and L_1 and M_1 are literals of C and D, respectively. Assume C and D have no common variables. Let σ be the most general unifier of L_1 and M_1. Then if M_1 if linear and L_2 is a linear literal in C, $L_2\sigma$ is linear. By symmetry, if M_2 is a linear literal in D and L_1 is linear, then $M_2\sigma$ is linear. Thus it is often possible to know that literals in the resolvent of two clauses are linear without even looking at them.

It is possible to say more if we know that L_1 or M_1 is a ground literal. With C and D as above and σ a most general unifier of L_1 and M_1, if L_1 is a ground literal and L_2 is a linear literal of C, then $L_2\sigma$ is L_2, which is linear.
If L_1 is a ground literal and M_2 is a linear literal of D, then $M_2\sigma$ is linear, since σ replaces variables by ground terms. Symmetrical statements can be made if L_2 is a ground literal.

One more improvement is possible. When unifying L and M, if L and M have no common variables, an occur check is not necessary when unifying x with t if the variable x has never been seen before during this unification.

It appears that this use of unification without occur check could result in a significantly faster unification algorithm, yielding speedups even greater than those possible using the currently best known algorithms.

A Solution to the Problem from Lewis Carroll
(from E. Lusk)

In the last newsletter, a problem from Lewis Carroll was posed. Two formulations were given. The second formulation included the following clauses:

1. -a | -b | c
2. -e | -k | m
3. -r | s | t
4. -c | -d | k
5. -r | e | n
6. -h | -l | r
7. -c | -m | -e
8. -s | -n | -k
9. -a | b | -h | -r
10. -a | -d | l
11. -c | -t | e
12. a
13. h
14. a | b
15. c | d
16. e | h
17. k | l
18. m | n
19. r | s

The problem was to determine the end of a sentence given by Carroll. One might view this as a challenge to determine the "strongest unit" deducible from the given set of clauses. The answer is "-d", which corresponds to the statement that "all monitors are awake". The deduction is as follows:

20. -b | c from clauses 12 and 1
21. -h | -r | b from clauses 12 and 9
22. -d | l from clauses 12 and 10
23. -l | r from clauses 13 and 6
24. s | t from clauses 19 and 3
25. l | c from clauses 22 and 15
26. c | r from clauses 25 and 23
27. b | c from clauses 21, 13, and 26
28. c from clauses 27 and 20
29. e | s from clauses 28, 11, and 24
30. -d | k from clauses 28 and 4
31. -m | -e from clauses 28 and 7
32. -d | r from clauses 23 and 22
33. -e | -k from clauses 31 and 2
34. -e | n from clauses 31 and 18
35. -k | s from clauses 33 and 29
36. -r | n from clauses 34 and 5
37. -k | -n from clauses 35 and 8
38. -d | n from clauses 36 and 32
39. -n | -d from clauses 37 and 30
40. -d from clauses 39 and 38

Logic Problems
(from C. Morgan)

Charles Morgan from the University of Victoria has sent in some logic problems. These are not open problems, but they do provide some interesting difficulties for theorem provers or reasoning programs in general. The axioms are as follows:

1. P(i(x,i(y,x)))
2. P(i(i(x,i(y,z)),i(i(x,y),i(x,z))))
3. P(i(i(n(x),n(y)),i(y,x)))
4. If P(i(x,y)) & P(x) then P(y)

Roughly speaking, P means "is provable", i means "implies", and n means "not". Thus Axiom 4 prepresents the inference rule of modus ponens.

Problem 1: For all x, P(i(x,n(n(x)))).

Problem 2: For all x, P(i(n(n(x)),x)).

Harder problems arise when Axiom 3 above is replaced by

3'. P(i(i(y,x),i(n(x),n(y))));

So the next problem is:

Problem 3: With axiom 3 replaced by axiom 3', P(i(x,n(n(x)))) for all x.

We have done some experimentation with these problems, which we will report on in the next newsletter.

Open Questions

Progress is being made on one of the goals of AAR, that of compiling a set of open questions for attacking with an automated reasoning program. We have received

contributions from Ed Howorka, a mathematician from the University of Virginia; Curt Lindner, a mathematician from Auburn University; Michael McRobbie, a logician from the Australian National University; and Ward Henson, a mathematician from the University of Illinois. Below is Henson's contribution.

Identities of Real Exponentiation (from C. W. Henson)

Consider terms built up from variables and the constant 1 using the three binary function symbols $+$, $\cdot$, and exp. We interpret these terms in the natural way over the positive integers $\mathbf{N}$ or the positive real numbers $\mathbf{R}^+$. Let ID be the set of equations $t_1 = t_2$ which are valid under these interpretations. (By classical results of G. H. Hardy, the valid identities over $\mathbf{N}$ are exactly the same as over $\mathbf{R}^+$ The set ID is recursive by results from Macintyre (Springer-Verlag *Lecture Notes in Mathematics*, Vol. 890). We are interested in the formal derivability of identities, using rules of inference in equational logic. For example, it is an open problem whether there exists a finite set $E!subsetID$ such that every identity in ID can be derived from E.

Alfred Tarski raised the question whether every identity in ID can be derived from the familiar set of "high school algebra identities":

$$(HS)\quad \begin{cases} 1^x=1, x^1=1x=x1=x \\ x+y=y+x,\ xy=yx \\ x+(y+z)=(x+y)+z,\ x(yz)=(xy)z \\ x(y+z)=xy+xz \\ x^{y+z}=x^y\times x^z \\ x^{y+z}=x^y\times x^z \\ (xy)^z=x^z\times y^z \\ (x^y)^z=x^{yz} \end{cases}$$

Alec Wilkie answered the question negatively by showing that the identity

$$(W)\quad ((x+1)^x+(x^2+x+1)^x)^y((x^3+1)spy+(x^4+x^2+1)^y))^x$$
$$((x+1)^y+(x^2+x+1)^y)^x\ ((x^3+1)^x+(x^4+x^2+1)^x)^y$$

cannot be derived from (HS). (The fact that it is valid over $\mathbf{R}^+$ can easily be shown by considering the factor $(x^2-x+1)^{xy}$; note that this is *not* a term in the formal language being considered here.)

Wilkie's argument used proof theory. More recently R. Gurevič presented a 59-element model in which Tarski's high school identities are true and Wilkie's identity is false. The same kind of result can also be proved when y in (W) is replaced by a sufficiently complicated term in x alone, for example, x^x.

Given this background, consider the following test problems for automated theorem-proving testing systems:

(A) Verify that (W) cannot be derived from (HS).

(B) Find a finite model for (HS) in which (W) is false. How small can such a model be?

(C) Do (A) and (B) for identities obtained from (W) by replacing y by x^x or other terms in x alone. (Especially interesting is $y=x^2$.)

(D) Find "simpler" identities than (W) that are valid over $\mathbf{R}^+$, but that cannot be derived from (HS).

Problem Sets

Another of the goals of AAR, the formulation of a set of basic problems to test ideas with, is moving forward. Overbeek of Argonne and Siekmann of the University of

Kaiserslautern are putting together such a set. In particular, Overbeek and Lusk (also of Argonne) have assembled a set of graduated problems for testing paramodulation, an inference rule that "builds in" equality substitution. This set of problems is included below.

Graduated Problems for Testing Equality Reasoning (from R. Overbeek and E. Lusk)

Below are six problems that we have used as benchmarks during the development of our theorem-proving systems. We have arranged the problems into an order that reflects their relative difficulty. They are as follows:

Problem 1: In a group, if $x^2 = e$ for all x in the group, then the group is Abelian (for all x and y, $xy = yx$)

Problem 2: In a group, $(x^{-1})^{-1} = x$ for all x in the group.

Problem 3: In a ring, if $x^2 = x$ for all x in the ring, then $xy = yx$ for all x,y in the ring.

Problem 4: In a group, if $x^3 = e$ for all x in the group, then the commutator $h(h(x,y),y) = e$ for all x and y. The commutator $h(x,y)$ is defined as $xyx^{-1}y^{-1}$.

Problem 5: In a ternary Boolean algebra with the third axiom removed, it is true that $f(x,g(x),y) = y$ for all x and y.

Problem 6: In a ring, if $x^3 = x$ for all x in the ring, then $xy = yx$ for all x and y in the ring.

The first problem is a classic in the thecrem proving literature. It is normally used as an initial test to verify that an equality reasoning component is functioning properly.

The second problem is also quite simple, and should be easily solved by any system that includes substitution and simplification capabilities.

The third problem introduces the axioms for a ring. It is of moderate difficulty.

The fourth problem, also a classic, is substantially more difficult than the first two problems. It was included in one of the papers that introduced paramodulation (see Wos and Robinson, *Machine Intelligence*, Edinburgh U. Press, 19969, pp. 135-150).

The fifth problem involves ternary Boolean algebras, a rather obscure area in mathematics. Our system attained a proof of this problem by using "noncomplexifying paramodulation". In this restriction of paramodulation, variables that occur both in the into term and outside the into term can be instantiated only to other variables or to constants (variables in the from term can be arbitrarily instantiated). We have found noncomplexifying paramodulation useful in other proofs, as well; however, no comprehensive study has been made of its general utility.

The sixth problem is truly difficult for existing systems. Two researchers have reported on approaches that resulted in proofs (Stickel, *Lecture Notes in Computer Science*, Vol. 170, 1984, pp. 248-258; and Veroff, ANL 81-6, February 1981).

1. Problem 1

```
1   f(e,x) = x          e is a left identity
2   f(x,e) = x          e is a right identity
3   f(g(x),x) = e       there exists a left inverse
4   f(x,g(x)) = e       which is also a right inverse
```

```
5   f(f(x,y),z) = f(x,f(y,z))        associativity
6   x = x                            reflexivity of equality
7   f(x,x) = e                       x^2 = e (special hypothesis)
8   -(f(a,b) = f(b,a))               denial of the theorem

9   f(x,f(y,f(x,y))) = e             7 5 5 5 5
10  x = f(y,f(y,x))                  7 5 1
11  f(x,f(y,x)) = y                  9 10 2
12  f(x,y) = f(y,x)                  11 10
13  null                             12 8
```

2. Problem 2

```
1   f(e,x) = x                       e is a left identity
2   f(x,e) = x                       e is a right identity
3   f(g(x),x) = e                    there is a left inverse
4   f(x,g(x)) = e                    which is also a right inverse
5   f(f(x,y),z) = f(x,f(y,z))        associativity
6   x = x                            reflexivity of equality
7   -(g(g(a)) = a)                   denial of the theorem

8   z = f(x,f(g(x),z))               5 4 1
9   g(g(x)) = x                      8 4 2
10  null                             9 7
```

3. Problem 3

```
1   j(0,x) = x                       0 is a left identity for sum
2   j(x,0) = x                       and a right identity for sum
3   j(g(x),x) = 0                    there is a left inverse for sum
4   j(x,g(x)) = 0                    which is a right inverse for sum
5   j(j(x,y),z) = j(x,j(y,z))        associativity of addition
6   x = x                            reflexivity of equality
7   j(x,y) = j(y,x)                  commutativity of addition
8   f(f(x,y),z) = f(x,f(y,z))        associativity of multiplication
9   f(x,j(y,z)) = j(f(x,y),f(x,z))   distributivity axioms
10  f(j(y,z),x) = j(f(y,x),f(z,x))
11  f(x,x) = x                       x^3 = x (special hypothesis)
12  -(f(a,b) = f(b,a))               denial of the theorem

13  f(x,j(x,y)) = j(x,f(x,y))                          11 9
14  f(x,j(x,x)) = j(x,x)                               11 13
15  j(x,y) = j(f(x,j(x,y)),f(y,j(x,y)))                11 10
16  j(x,x) = j(j(x,x),j(x,x))                          14 15 14
17  j(j(x,x),y) = j(j(x,x),j(j(x,x),y))                16 5
18  0 = j(x,x)                                         17 4 4 2
19  j(x,j(y,z)) = j(y,j(x,z))                          7 5 5
20  j(x,y) = j(x,j(y,j(f(x,y),f(y,x))))                15 9 11 9 11 7 5 19
21  y = j(g(x),j(x,y))                                 3 5 1
22  j(x,j(f(y,x),f(x,y))) = x                          21 20 21
23  j(f(x,y),f(y,x)) = 0                               21 22 3
24  x = j(y,j(y,x))                                    5 18 1
25  f(x,y) = f(y,x)                                    24 23 2
26  null                                               25 12
```

4. Problem 4

```
1   f(e,x) = x                          e is a left identity
2   f(x,e) = x                          and a right identity
3   f(g(x),x) = e                       there is a left inverse
4   f(x,g(x)) = e                       which is also a right inverse
5   f(f(x,y),z) = f(x,f(y,z))           asscociativity
6   x = x                               reflexivity of equality
7   h(x,y) = f(x,f(y,f(g(x),g(y))))     definition of commutator
8   f(x,f(x,x)) = e                     x^3 = e (special hypothesis)
9   -(h(h(a,b),b) = e)                  denial of the theorem

10  g(e) = e                                                    3 2
11  -(f(a,f(b,f(g(a),f(g(b),f(b,f(g(f(a,f(b,f(g(a),g(b))))),g(b)))))))
     = e)                                                       9 7 7 5 5 5
12  x = f(y,f(g(y),x))                                         5 4 1
13  x = f(g(y),f(y,x))                                         5 3 1
14  e = f(x,f(y,g(f(x,y))))                                    5 4
15  -(f(a,f(b,f(g(a),f(g(f(a,f(b,f(g(a),g(b))))),g(b))))) = e)
                                                               13 11
16  g(g(x)) = x                                                12 4 2
17  f(g(x),g(x)) = x                                           12 8 2
18  f(x,x) = g(x)                                              17 16 16
19  f(x,f(y,f(x,y))) = g(f(x,y))                               18 5
20  f(g(x),y) = f(x,f(x,y))                                    18 5
21  f(x,f(g(y),x)) = f(y,g(f(g(y),x)))                         19 12
22  f(x,f(y,x)) = f(g(y),g(f(y,x)))                            19 13
23  f(x,f(g(y),f(x,z))) = f(y,f(g(f(g(y),x)),z))                   21 5 5 5
24  f(x,f(y,f(x,z))) = f(g(y),f(g(f(y,x)),z))                      22 5 5 5
25  -(f(a,f(g(b),f(g(a),f(g(b),f(a,f(b,f(g(a),b))))))) = e)
                                                               24 15 5 5 18 16 5 5 20
26  f(x,g(f(y,x))) = g(y)                                      14 22 10 2 14 2
27  g(f(x,y)) = f(g(y),g(x))                                   26 13
28  f(x,f(y,f(x,y))) = f(g(y),g(x))                            27 19
29  f(x,f(g(y),f(x,z))) = f(y,f(g(x),f(y,z)))                      27 23 16 5
30  -(e = e)                                                   29 25 20 28 16 16 12 8
31  null                                                       30 6
```

5. Problem 5

Ternary Boolean algebras were defined in by A. A. Grau in 1947 ("Ternary Boolean algebra", Bulletin of American Math. Soc. 53, 6, June 1947, pp. 567-572). Later Chinthayamma published work on independent axioms for ternary Boolean algebras (see "Sets of independent axioms for a ternary Boolean algebra", Notices Amer. Math. Soc. 16, 4, June 1969, p. 654).

The function f can be thought of as a three-place product, and the function g may be thought of as inverse.

```
1   f(f(v,w,x),y,f(v,w,z)) = f(v,w,f(x,y,z))
                             ax. 1 of a ternary boolean algebra
2   f(y,x,x) = x             ax. 2 of a ternary boolean algebra
3   f(x,y,g(y)) = x          ax. 3 of a ternary boolean algebra
4   f(x,x,y) = x             ax. 4 of a ternary boolean algebra
5   f(g(y),y,x) = x          ax. 5 of a ternary boolean algebra
6   x = x                    reflexivity of equality
```

Now remove axiom 3 and add the following lemma:

```
7   f(x,y,x) = x                    lemma provable from 1, 2, 5, and 6
```

The denial of the theorem is as follows:

```
8   -(f(a,g(a),b) = b)              denial of the theorem
```

```
9   f(f(v,w,x),x,v) = f(v,w,x)                  1 7 4
10  f(f(y,w1,z),y,z) = f(y,w1,z)                1 9 2 4 9
11  f(f(v,w,g(y)),y,v) = v                      1 7 5 7
12  f(x,y,f(v,x,y)) = f(v,x,y)                  1 2 2
13  f(f(v,w,x),x,f(v1,v,w)) = f(v,w,x)          1 12 4
14  f(y,g(y),z) = z                             1 13 2 10 11 2
15  null                                        8 15
```

6. Problem 6

```
1   j(0,x) = x                      0 is a left identity for sum
2   j(x,0) = x                      and a right identity for sum
3   j(g(x),x) = 0                   there is a left inverse for sum
4   j(x,g(x)) = 0                   which is a right inverse for sum
5   j(j(x,y),z) = j(x,j(y,z))       associativity of addition
6   x = x                           reflexivity of equality
7   j(x,y) = j(y,x)                 commutativity of addition
8   f(f(x,y),z) = f(x,f(y,z))       associativity of multiplication
9   f(x,j(y,z)) = j(f(x,y),f(x,z))  distributivity axioms
10  f(j(y,z),x) = j(f(y,x),f(z,x))
11  f(x,f(x,x)) = x                 x^3 = x  (special hypothesis)
12  -(f(a,b) = f(b,a))              denial of the theorem
```

The proof of this theorem is complex enough to prohibit the type of presentation that we have used for the preceding theorems. We include a proof in the form that a human mathematician might write it. Since the problem is a standard one for graduate courses in algebra, there are commonly available proofs. However, the one that we supply seems somewhat unusual. It was given to us by Steve Winker.

Proof:

(1) Note that $(x^2)^2 = x^2$, since $x^3 = x$.

(2) First, we prove that $x^2y^2x^2 = y^2x^2y^2$:
If we multiply out $(x^2-y^2)^3$ and simplify the result, we find that

$$(x^2-y^2)^3 = x^2 - x^2y^2x^2 + y^2x^2y^2 - y^2$$

However, since $(x^2-y^2)^3 = (x^2-y^2)$ by the hypothesis of the theorem, $x^2y^2x^2 = y^2x^2y^2$.

(3) Now we can show that squares commute; that is, for any x,y

$$x^2y^2 = y^2x^2.$$

Start with the equation

$$x^2y^2x^2 = y^2x^2y^2.$$

If we right multiply each side with $(x^2y^2x^2y^2)$ and simplify the result, we derive

$$x^2y^2 = y^2x^2y^2.$$

On the other hand, if we left multiply each side by $(y^2x^2y^2x^2)$ and simplify, we derive

$$y^2x^2 = y^2x^2y^2.$$

By transitivity, we arrive at the desired lemma: $x^2y^2 = y^2x^2$

(4) Now we can show that a square will commute with anything; that is, for any x and y, $xy^2 = y^2x$:

$$(xy^2)^3 = x(y^2xy^2x)y^2 = xy^2(y^2xy^2x) = xy^2xy^2x \text{ (since squares commute)}$$
$$(y^2x)^3 = y^2(xy^2xy^2)x = (xy^2xy^2)y^2x = xy^2xy^2x$$

Thus, $xy^2 = (xy^2)^3 = xy^2xy^2x = (y^2x)^3 = y^2x$

(5) Now we can finish the proof of the theorem: for any x and y, $xy = yx$.

$$xy = xyxyxy = xyxy^3xy = xy(xy)y^2xy = xyy^2xyxy =$$
$$xyyy(xy)^2 = xyy(xy)^2y = xy^2(xy)^2y = x(xy)^2y^2y =$$
$$xxyxyyyy = x^2yxy^2 = y^2yxx^2 = yx$$

The set of six problems represents a wide range of difficulty. The first two problems are relatively trivial. The third and fourth are fairly difficult, although there are several existing theorem-proving systems capable of deriving proofs in fairly short time periods. We have found the fifth problem quite challenging, although the proof is not terribly long. The sixth problem represents the most complex problem in equality for which a proof has been derived by an automated system.

The reader should note that the proofs given are not always exactly those generated by automated reasoning systems. In some cases, we have shortened proofs derived by our system. We include proofs only as aids for those who wish to study exactly why their system might be failing to reach a complete proof.