MATHEMATICS OF COMPUTATION
VOLUME 37, NUMBER 156
OCTOBER 1981

Semigroups, Antiautomorphisms, and Involutions: A Computer Solution to an Open Problem, I*

By S. K. Winker, L. Wos and E. L. Lusk

Abstract. An *antiautomorphism* H of a semigroup S is a 1-1 mapping of S onto itself such that $H(xy) = H(y)H(x)$ for all x, y in S. An antiautomorphism H is an *involution* if $H^2(x) = x$ for all x in S. In this paper the following question is answered: Does there exist a finite semigroup with antiautomorphism but no involution? This question, suggested by I. Kaplansky, was answered in the affirmative with the aid of an automated theorem-proving program. More precisely, there are exactly four such semigroups of order seven and none of smaller order. The program was a completely general one, and did not calculate the solution directly, but rather rendered invaluable assistance to the mathematicians investigating the question by helping to generate and examine various models. A detailed discussion of the approach is presented, with the intention of demonstrating the usefulness of a theorem prover in carrying out certain types of mathematical research.

1. Introduction. This paper has three objectives. The first is to present an answer to an open question concerning antiautomorphisms of finite semigroups. The second is to describe some functions of a general-purpose automated theorem prover and show how it was used in the investigation of this question. Finally, we wish to solicit other significant questions of a similar nature. Indeed, since no additional programming was required for the attack on this problem, those who understand the techniques presented here should be able to use our theorem-proving system for their own purposes. Since the paper is addressed to groups with widely differing backgrounds, it is written at an elementary level throughout.

1.1. *The Results.* There is a finite semigroup which supports an antiautomorphism but no involution. All necessary definitions are given in Section 2.1, one such semigroup is given in Section 2.2, and a small example (order 7) is given in Section 2.3. The theorem-proving techniques used in the discovery of this example are described in detail in this paper.

There are exactly four semigroups of order 7 having the above property and no smaller semigroups with this property. The techniques used in arriving at this result are very different from those used in the first part of the investigation and will be described in a later paper.

Received June 9, 1980; revised January 28, 1981 and March 13, 1981.

1980 *Mathematics Subject Classification.* Primary 20M05, 20M15; Secondary 03B35.

Key words and phrases. Finite semigroups, involutions, antiautomorphisms, automated theorem-proving.

* This work was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U. S. Department of Energy under Contract W-31-109-Eng-38 (Argonne National Laboratory, Argonne, Illinois 60439) and in part by NSF grant #MCS 79-03870 (Northern Illinois University, DeKalb, Illinois 60115).

0025-5718/81/0000-0171/$04.25

1.2. *The Theorem Prover.* The theorem prover used here is a general-purpose resolution-based program developed over the past eight years at Northern Illinois University and Argonne National Laboratory. Nothing was added to the program to orient it towards this particular problem, nor towards algebra in general. Rather, a collection of nonstandard input statements was prepared which caused the program to generate and check models instead of deriving a contradiction from the input, which is how it has traditionally been used. Particular use was made of the program's capabilities for dealing with equality. Some fundamental theorem-proving concepts are presented in Section 3 for those unfamiliar with automated theorem proving, and the technique used to cause the theorem prover to generate and check models is described in detail in Section 4.

1.3. *Solicitation of Problems.* Theorem-proving technology has advanced slowly over the last fifteen years, but has now reached the state of being able to do more than simple exercises. The authors believe that for a limited class of problems theorem provers can be of real use to mathematicians and other researchers whose calculations are algebraic and logical rather than numerical. The project described here is offered as an example of such use. Several investigations of a similar nature are currently under way, and more problems which might yield to computer-aided attack are solicited. Some success on such problems in a field different from the one described here is described in [6].

2. The Main Result. In this section, we present the relevant definitions, some lemmas, and the technique used to find the desired semigroup. We will describe which parts of the process were assisted by the theorem prover but will defer to a later section the discussion of exactly how it was used.

2.1. *Definitions and Lemmas.*

Definition. A *semigroup* S is a nonempty set together with an associative binary operation $S \times S \to S$.

Definition. An *automorphism* K of a semigroup S is a 1-1 map of S onto S such that for all x, y in S, $K(x)K(y) = K(xy)$. An *antiautomorphism* H of S is a 1-1 map of S onto S such that for all x, y in S, $H(xy) = H(y)H(x)$. An antiautomorphism is *nontrivial* if it is not an automorphism.

Definition. A (nontrivial) *involution* J of a semigroup S is a (nontrivial) antiautomorphism of S such that $J^2 = 1$, the identity map.

Remark. It is easy to see that if H is a (nontrivial) antiautomorphism, then H^k is an automorphism for even k and a (nontrivial) antiautomorphism for odd k. Since the identity map is an automorphism, the order of a nontrivial antiautomorphism must always be even. Furthermore, if H has order $2n$, then H^n is either a nontrivial involution or an automorphism of order 2 depending on whether n is odd or even. This observation motivates the following question, communicated to us by I. Kaplansky.

Question. Does there exist a finite semigroup which has a nontrivial antiautomorphism but no nontrivial involution?

Note that if such a semigroup exists then by the remark above, the order of the antiautomorphism must be a multiple of four.

2.2. *Techniques for Solution.* Since we are attempting not only to answer the question but also to demonstrate how the theorem prover was used in the

investigation, we present here a relatively detailed account of our approach to investigating the question.

Suppose that the answer to the question is yes, and that H is the antiautomorphism. Since the order of H is a multiple of 4, the semigroup contains distinct elements B, C, D, and E such that $H(B) = C$, $H(C) = D$, and $H(D) = E$. If H has order 4, then such elements exist with $H(E) = B$.

We begin by making a number of assumptions about the desired semigroup S and its antiautomorphism H. We do this not without loss of generality but as a first move. The first assumption is that H has order 4. (Fortunately, a solution to the problem can be found with this condition. If this had not happened, the technique outlined here could still have been used, but more computer time would have been required.) We assume further that S is generated by the elements B, C, D, and E which cycle under H.

Finally, we need to introduce relations which will make the semigroup finite. One particularly straightforward way to do this is to assume that all products of more than a certain number of elements are identical. This has the additional benefit of making it unnecessary for the theorem prover to deal with terms beyond a specified complexity. In this investigation it was assumed that all products of four or more elements were identical. It turned out that this allowed sufficient complexity for the desired example to be found within the limits imposed by this restriction.

The assumptions above provide us with a semigroup of order 85 which supports a nontrivial antiautomorphism.

Unfortunately, involutions abound. For example, one can construct a nontrivial involution by extending the mapping which exchanges two of the generators and leaves the others fixed. The plan of attack on the problem, then, is to adjoin to the current set of relations which define S (those that make all products of four or more elements identical) more relations, so chosen as to block the existence of an involution while preserving the existence of an antiautomorphism.

By adjoining relations to S we are forming a quotient semigroup S', consisting of the equivalence classes of elements of S under the relations. The antiautomorphism H on S can be defined on the quotient semigroup as follows. Let $p\colon S \to S'$ be the natural projection of S onto S'. Define $H'(p(x)) = p(H(x))$. This makes sense as long as it is true that for all x and y in S, $p(x) = p(y)$ implies $p(H(x)) = p(H(y))$.

The computer helps most in examining the consequences of adding a given relation. The first type of consequence, as described in the preceding paragraph, is that the addition of a relation R requires that the relations $H(R)$, $H^2(R)$, and $H^3(R)$ be added as well. (For example, adding the relation $BC = CDD$ requires addition of $DC = EED$, $DE = EBB$, and $BE = CCB$. This is done automatically by the theorem prover, as will be explained below. Secondly, adding a relation may force the addition of yet more relations to keep the operation in S well-defined. For example, adding $BB = DD$ forces us to add $EBB = EDD$, etc.) The theorem prover automatically provided these additional relations. Finally, once a set of relations was chosen, the theorem prover automatically tested the resulting semigroup for the presence of involutions.

The selection of a set of relations can either be automated or left to the investigator. In order to provide some insight into the first solution of the problem,

we present here some observations which led us to the right choice of relations. Even in this phase, the theorem prover provided moral support in that the effects of any choice could always be easily and reliably checked.

Let us consider the original semigroup S of order 85 and what relations to add in order to block the existence of involutions. Each possible involution falls into at least one of the following three classes:

(i) those which interchange at least two of the generators,

(ii) those which interchange a generator with a nongenerator,

(iii) those which fix all four generators.

We can block involutions of the second class by avoiding the addition of relations which allow a generator to be written as a product. To see this, suppose that J is an involution which interchanges, say, B with CD. Then $B = J^2(B) = J(CD) = J(D)J(C)$, and B has been written as a product.

Consider next an involution J of the third class. The presence, for some X and Y, of the relation $BBX = DDY$ requires the addition of the relation $J(X)BB = J(Y)DD$. Thus, if we can find a relation of the form $BBX = DDY$, which does not yield one of the form $ZBB = WDD$ when all required relations are added, this will block involutions of the third class.

Similarly one can examine the consequences of the existence of an involution of the first class in the presence of some proposed relation by considering the various possible interchanges between two generators.

Such considerations can greatly narrow the search for the desired relations. The theorem prover assists by automatically generating the set of relations implied by a given choice, and by checking for involutions. The exact technique by which the theorem prover is made to do this is described in Section 4.

Let R be the relation $BBC = DDE$. Then since $H^2(R) = R$, the only additional relation which must be added is $H(R)$, namely $DCC = BEE$. As noted above, there cannot be any involutions of the second or third class. To see that there cannot be any of the first class, observe that any exchange of generators will introduce more relations. This establishes the existence of a finite semigroup S (of order 83) which has an antiautomorphism (inherited from the antiautomorphism H of S) but no nontrivial involutions. The computer-assisted search for a smaller example will be described in the next section.

2.3. *The Solution of Order* 7. The search for smaller examples of semigroups having the desired property consisted of adding relations to this semigroup of order 83. The theorem prover facilitated experimentation by quickly and accurately calculating the consequence of adding any set of relations. Thus experience was gained which led eventually to the following set of relations:

(1) All products of three or more elements are equal to BD,

(2) $BD = CE = EC = CB$,

(3) $BB = DD = BC$.

Other equalities were derived by the theorem prover from these, in accordance with the considerations described above.

The elements of the resulting semigroup are represented by the equivalence classes of B, C, D, E, BD, BB, BE. If we name these equivalence classes a, b, c, d,

e, f, and g, respectively, the multiplication table of the semigroup is as follows:**

	a	b	c	d	e	f	g
a	f	f	e	g	e	e	e
b	e	g	e	e	e	e	e
c	e	g	f	f	e	e	e
d	e	e	e	g	e	e	e
e	e	e	e	e	e	e	e
f	e	e	e	e	e	e	e
g	e	e	e	e	e	e	e

The antiautomorphism h acts as follows:

$h(a) = b \quad h(b) = c \quad h(c) = d$
$h(d) = a \quad h(e) = e \quad h(f) = g$
$h(g) = f$

3. An Introduction to Automated Theorem Proving. In this section we described in some detail how the theorem prover was used in the investigation described in Section 2. We begin by giving a brief informal overview of standard terminology and techniques, intended for the reader who is totally unfamiliar with automated theorem proving. A more formal and complete treatment may be found in [1]. Then we will show how these techniques and some nonstandard ones were used to extend sets of relations and check for the presence of involutions.

A theorem prover is a program which accepts as input statements in the first-order predicate calculus. Its output consists of statements in the first-order predicate calculus which are derivable from the input statements by means of one or more standard inference rules (such as modus ponens). Several formalisms and inference systems have been developed. The one on which the theorem prover used here is based is called *resolution*.

3.1. *Normal Forms.* For our purposes any statement in first-order predicate calculus can be replaced by a conjunction of disjunctions, in which each disjunction has the property that all variables are universally quantified. For example, the statement $\forall X \exists Y(P(X, Y) \rightarrow Q(X))$ becomes $(\forall X)(\neg P(X, F(X)) \vee Q(X))$. The function F which chooses a Y for each X is called a *Skolem function.* A statement which consists of such a conjunction of disjunctions is said to be in *normal form.* The disjunctions themselves are called *clauses*, and the disjuncts are its *literals*. We consider a list of clauses to represent the conjunction of those clauses, and we omit the universal quantifiers and the disjunction symbols from the representation, since they are redundant. Hence the statement above can be represented as a clause as follows:

$$\text{CL} \quad \neg P(X, F(X)) \quad Q(X).$$

we prefix clauses by "CL" in order to distinguish them from other expressions. In the next section, we describe how new clauses are derived from existing ones.

3.2. *Resolution and Hyperresolution.* Two literals are said to be *unifiable* if they have a common instance; that is, if there are substitutions for the variables of each which make them identical. The scope of a variable is exactly the clause in which it occurs. Suppose we have two clauses of the form

$$\text{CL} \quad \neg L1 \quad L2$$
$$\text{CL} \quad L3$$

** The three additional semigroups of order seven of the desired type are found in the Appendix.

and suppose that $L1$ and $L3$ are unifiable. Then modus ponens allows us to derive the clause CL $L2$, with the same substitution applied to the variables of $L2$ which was applied to those in $L1$ in order to unify it with $L3$. More generally, any two clauses containing unifiable literals of opposite sign can be used to derive a third clause consisting of all the literals in both clauses except the two matching ones, with the appropriate substitutions applied. Such a derivation is called a *clash* of the two clauses. For example, the clauses

$$\begin{array}{llll} \text{CL} & \neg P(F(X), A) & Q(X), \\ \text{CL} & P(F(B), Y) & R(Y), \end{array}$$

in which X and Y represent variables, can be clashed to derive

$$\text{CL} \quad Q(B) \quad R(A).$$

The inference rule described above (clashing pairs of clauses containing complementary literals) is called *binary resolution*. A different inference rule, which replaces particular sequences of binary resolutions by a single derivation, is called *hyperresolution*. It functions as follows. A clause containing both positive and negative literals is clashed against a set of clauses containing all positive literals until all of its negative literals are gone. For example, from the clauses

$$\begin{array}{llll} \text{CL} & \neg P(X, Y, Z) & \neg Q(X, Y, Z) & S(X, Y, Z), \\ \text{CL} & P(A, B, X) & R(A), & \\ \text{CL} & Q(X, Y, C), & & \end{array}$$

the clause CL $S(A, B, C)$ $R(A)$ is derived.

Hyperresolution becomes a very natural and intuitive inference rule, particularly in the case where the positive clauses contain only one literal, if one notices that the clause

$$\text{CL} \quad \neg L1 \quad \neg L2 \quad L3$$

is equivalent to $(L1 \wedge L2) \rightarrow L3$. Thus hyperresolution is the process of deriving the conclusion of an inference, given all of the hypotheses. For example, the statement that if X and Y are both elements of S then so is their product $F(X, Y)$ can be represented by the clause

$$\text{CL} \quad \neg EL(X, S) \quad \neg EL(Y, S) \quad EL(F(X, Y), S).$$

Then from CL $EL(A, S)$ and CL $EL(B, S)$ hyperresolution derives CL $EL(F(A, B), S)$.

3.3. *Demodulation*. The equality predicate plays a special role in the theorem prover used in this investigation. Whenever a clause of the form CL $EQUAL(T1, T2)$ was derived in this problem, it was added to a list of rewrite rules called *demodulators*. (One has the option with our program of treating none, some, or all new equalities as demodulators.) *Demodulation* is the process of replacing the term $T1$ by the term $T2$ in any derived clause if $EQUAL(T1, T2)$ is a demodulator. Furthermore, if $T1'$ can be obtained from $T1$ by a substitution for some of $T1$'s variables (i.e. $T1'$ is an *instance* of $T1$) and $T2'$ is obtained from $T2$ by applying the same substitution, then we may replace $T1'$ by $T2'$ in any derived clause. Newly derived demodulators are also applied to existing clauses. Multiple demodulators can be applied in a single step. A variety of techniques exist to prevent looping. Finally, all terms are assigned a measure of complexity and

tie-breaking rules exist so that there is a canonical way of ordering the arguments of equalities so that "simpler" terms always occur on the right. As an example, consider the following set of clauses

(1) CL $EQUAL(F(A), B)$ (demodulator),

(2) CL $EQUAL(G(X), F(X))$ (demodulator),

(3) CL $\neg P(Y)$ $Q(Y)$,

(4) CL $P(G(A))$.

The clause CL $Q(B)$, the result of clashing (3) against (4) and applying first (2) and then (1) as demodulators, is derived by the theorem prover in a single step.

3.4. *Subsumption.* When a clause is derived which is a logically weaker statement than a clause which already exists, in most cases the new clause is immediately deleted. For example, if CL $P(V0, V1, V2, V2)$ is an existing clause and CL $P(A, B, C, C)$ is derived, it is immediately deleted, although CL $P(A, B, C, D)$ would not be deleted. The new clause is said to be *subsumed* by the old clause. Old clauses may also be subsumed by new clauses.

3.5. *Standard Use of a Automated Theorem Prover.* The theorem-proving program used in the present investigation [3], [4], [5] was originally designed to find proofs of theorems. This is accomplished by providing as input clauses a set of axioms for a particular domain and a set of clauses representing the negation of the statement of the theorem to be proved. Derivation by the theorem prover of the empty clause then signals a contradiction, since the empty clause results from clashing CL P with CL $\neg P$. The derivation history of the empty clause then provides a proof of the theorem.

One major point of this paper is to show how such a theorem prover can be used, without any new programming, in a completely different way. In investigating the existence of semigroups with certain properties, the theorem prover was used more as an algebraic and logical calculator. This change of use was accomplished merely by using special types of input clauses, so that the required calculations could be accomplished through hyperresolution and demodulation. The following section describes how this was done.

4. Use of the Theorem Prover in Investigating Semigroups. In this section we demonstrate in considerable detail exactly how the standard theorem-proving concepts described in the preceding section were used in the investigation described in Section 2. It is to be emphasized that no new programming specific to this problem was done. An existing theorem-proving program was used, with input clauses specially designed for this investigation. It is hoped that this will serve as an example of nonstandard use of a theorem prover which will inspire other uses of existing theorem-proving programs by researchers outside the theorem-proving community.

Recall from Section 2 that in the investigation described there the theorem prover was used to examine the consequences of adding various relations to a semigroup with four generators, cyclicly permuted by an antiautomorphism H, made finite by the relations equating all products of four or more elements. In the following sections, we describe

(i) the input clauses which describe the basic semigroup S, the antiautomorphism H, and the further relations to be added to define a new semigroup,

(ii) the input clauses which cause new relations forced by the fact that multiplication must be well defined in the new semigroup,

(iii) the input clauses which cause new relations forced by the fact that H must be well defined in the new semigroup,

(iv) the input clauses which cause the theorem prover to check for the presence of antiautomorphisms of order two.

4.1. *Defining the Original Semigroup with Clauses.* In what follows, variables (assumed to be all universally quantified, as explained in Section 3.1) are represented by the symbols $V0$, $V1$, $V2, \ldots$. Multiplication in the semigroup is represented by the function F, so that $F(V0, V1)$ represents the product of $V0$ and $V1$. The generators of the semigroup are represented by B, C, D, and E. The antiautomorphism which permutes the generators is represented by H.

The first clause represents associativity. It becomes a demodulator which canonicalizes products by associating them to the right.

$$1 \quad \text{CL} \quad EQUAL(F(F(V0, V1), V2), F(V0, F(V1, V2))).$$

The next clause says that all products of four or more elements are equal to $BBBB$.

$$2 \quad \text{CL} \quad EQUAL(F(V0, F(V1, F(V2, V3))), F(B, F(B, F(B, B)))).$$

Next we describe the action of H on the generators and extend it to products.

$$3 \quad \text{CL} \quad EQUAL(H(B), C)$$
$$4 \quad \text{CL} \quad EQUAL(H(C), D)$$
$$5 \quad \text{CL} \quad EQUAL(H(D), E)$$
$$6 \quad \text{CL} \quad EQUAL(H(E), B)$$
$$7 \quad \text{CL} \quad EQUAL(H(F(V0, V1)), F(H(V1), H(V0))).$$

All of these become demodulators.

4.2. *Multiplication is Well Defined.* The existence of one relation implies others, since multiplication is well defined. For example, the relation $BB = CC$ implies the relation $BBD = CCD$. This was handled in the following way. All equalities of products were represented by equalities with a variable concatenated to the product on each side of the equality. For example $BB = CC$ is represented by the clause

$$\text{CL} \quad EQUAL(F(B, F(B, V0)), F(C, F(C, V0))).$$

Thus although the relation $BBD = CCD$ is not explicitly derived, if the term BBD occurs in any derived clause, it will immediately be demodulated by the above clause to CCD. Similarly, if DBB occurs, since it will be represented as $F(D, F(B, F(B, V0)))$, the demodulator will apply.

When it is desired to remove this formal variable (see 4.3 and 4.4), the following clauses are used. The function REMVAR stands for "remove variable."

$$8 \quad \text{CL} \quad EQUAL(REMVAR(F(V0, V1)), V0)$$
$$9 \quad \text{CL} \quad EQUAL(REMVAR(F(V0, F(V1, V2))), F(V0, REMVAR(F(V1, V2)))).$$

The theorem prover prevents clause 8 from demodulating clause 9, and, in situations where both demodulators may apply, applies clause 9 first.

There is also a more subtle way in which a given set of relations may force the addition of more relations because multiplication is well defined. For example, the relations $AB = D$ and $BC = E$ imply both that $ABC = DC$ and $ABC = AE$. The technique described in the preceding paragraph does not add this relation, either implicitly or explicitly. The following clauses are used to add relations which are forced in this way, essentially by transitivity of equality.

$$10 \quad \text{CL} \quad \neg EQUAL(V0, V1) \quad \neg EQUAL(V0, V2) \quad EQUAL(V1, V2)$$

$$11 \quad \text{CL} \quad \neg EQUAL(F(V0, V1), V2) \quad \neg EQUAL(V1, V3) \quad EQUAL(F(V0, V3), V2)$$

$$12 \quad \text{CL} \quad \neg EQUAL(F(V0, F(V1, V2)), V3) \quad \neg EQUAL(V2, V4) \quad EQUAL(F(V0, F(V1, V4)), V3)$$

$$13 \quad \text{CL} \quad \neg EQUAL(F(V0, F(V1, F(V2, V3))), V4) \quad \neg EQUAL(V3, V5) \quad EQUAL(F(V0, F(V1, F(V2, V5))), V4).$$

To see exactly how these clauses work, consider how $AB = D$ and $BC = E$ derive $AE = DC$. The relations $AB = D$ and $BC = E$ will be input as

$$\text{CL} \quad EQUAL(F(A, F(B, V0)), F(D, V0)) \quad \text{and}$$
$$\text{CL} \quad EQUAL(F(B, F(C, V0)), F(E, V0)), \text{ respectively.}$$

Hyperresolution will clash these against the first two literals in clause 11 substituting $F(C, V0)$ for $V0$ in the first of these and with the following substitutions for the variables in clause 11:

$$V0 = A$$
$$V1 = F(B, F(C, V0))$$
$$V2 = F(D, F(C, V0))$$
$$V3 = F(E, V0).$$

The resulting clause is

$$\text{CL} \quad EQUAL(F(A, F(E, V0)), F(D, F(C, V0))),$$

as desired.

4.3. *The Effect of H in Adding Relations.* Recall that we are trying to modify the semigroup S, which has an antiautomorphism H, by adding relations to S in such a way that the semigroup thus formed still has a well-defined antiautomorphism inherited from H. This means that for each relation added, its image under H must also be added. For example, since $H(B) = C$ and $H(C) = D$, the relation $BB = BC$ forces the relation $CC = DC$. In order to accomplish this, only one more clause is needed:

$$14 \quad \text{CL} \quad \neg EQUAL(F(V0, V1), V2) \quad EQUAL(F(H(REMVAR(V1)), F(H(V0), V3))), (F(H(REMVAR(V2)), V3)).$$

Thus from $BB = BC$, which is represented by

$$\text{CL} \quad EQUAL(F(B, F(B, V0)), F(B, F(C, V0)))$$

we get, using clause 14,

$$\text{CL}\quad EQUAL(F(H(REMVAR(F(B, V0)))), F(H(B), V1), F(H(REMVAR(F(B, F(C, V0))))), V1),$$

which demodulates, using clauses 8 and 9, to

$$\text{CL}\quad EQUAL(F(H(B), F(H(B), V1))), F(H(F(B, C)), V1),$$

which demodulates, using clauses 7, 3 and 1, to

$$\text{CL}\quad EQUAL(F(C, F(C, V0)), F(F(D, C), V0)),$$

which demodulates, using clause 1, to

$$\text{CL}\quad EQUAL(F(C, F(C, V0)), F(D, F(C, V0))),$$

as desired.

4.4. *Checking for involutions.* So far we have shown how to represent relations and how the theorem prover automatically adds relations which are implied by these relations. In this section we describe those clauses input to the theorem prover to cause it to check for the presence of an involution in the semigroup that results from adding these relations to the original semigroup S. It is in this situation that subsumption will be used.

In Section 2.2 we discussed how one may narrow the search for involutions by considering only those which permute the generators. Therefore one may check for involutions by checking the permutations of the generators which have order 2. There are nine such permutations.

$(BC)(D)(E)$	$(B)(CD)(E)$	$(BC)(DE)$
$(BD)(C)(E)$	$(B)(CE)(D)$	$(BD)(CE)$
$(BE)(C)(D)$	$(B)(C)(DE)$	$(BE)(CD)$

These are checked, in order, in a single theorem-proving run, but it will suffice to demonstrate what clauses are input to check for the presence of one involution.

To input the involution to be checked, the clause

$$\text{CL}\quad PERM(X, Y, Z, W)$$

is input, where X, Y, Z, and W represent the images of the generators B, C, D, and E under the involution. For example, to check for the presence of the involution $(BC)(D)(E)$, we use the clause

$$\text{CL}\quad PERM(C, B, D, E).$$

In order to cause the input of a *PERM* clause to automatically generate a collection of demodulators corresponding to specifying the values of the involution J on the generators, we introduce the following clauses.

15 $\text{CL}\quad \neg PERM(V0, V1, V2, V3)\ \ \neg PERMSEL(V0, V1, V2, V3, V4, V5)\ EQUAL(J(V4), V5)$

16 $\text{CL}\quad PERMSEL(V0, V1, V2, V3, B, V0)$

17 $\text{CL}\quad PERMSEL(V0, V1, V2, V3, C, V1)$

18 $\text{CL}\quad PERMSEL(V0, V1, V2, V3, D, V2)$

19 $\text{CL}\quad PERMSEL(V0, V1, V2, V3, E, V3).$

Thus, for example, input of the clause

$$\text{CL} \quad PERM(C, B, D, E)$$

will cause generation of the clauses

$$\text{CL} \quad EQUAL(J(B), C)$$
$$\text{CL} \quad EQUAL(J(C), B)$$
$$\text{CL} \quad EQUAL(J(D), D)$$
$$\text{CL} \quad EQUAL(J(E), E).$$

The clause which says J is an antiautomorphism is

$$20 \quad \text{CL} \quad EQUAL(J(F(V0, V1)), F(J(V1), J(V0))).$$

In order for permutation of the generators of the original semigroup S to give rise to an involution on the new semigroup, it must remain well defined in the presence of all the relations being added. That is, if $XY = Z$ then $J(Y)J(X) = J(Z)$. This statement is represented by the clause

$$21 \quad \text{CL} \quad \neg EQUAL(F(V0, V1), V2)$$
$$EQUAL(F(F(J(REMVAR(V1)), J(V0)), V3),$$
$$F(J(REMVAR(V2)), V3)).$$

The various relations are clashed against the first literal in clause 21. If an involution is inheritable to the new semigroup, the two arguments of the second literal must be identical, after all demodulators are implied. If this happens, the resulting clause will be subsumed by the input clause

$$22 \quad EQUAL(V0, V0).$$

Therefore we can tell whether the new semigroup supports an involution inherited from S by seeing whether any clauses are generated from clause 21. If at least one clause is generated (and not subsumed) from clause 21 for each of the nine input *PERM* clauses, then the resulting semigroup has no involutions. If a clause is generated by clause 21 and kept, then the generated clause describes the contradiction which blocks the existence of the proposed involution.

The reader may recognize at this point that difficulties associated with the word problem need to be dealt with, since we are relying on the generated set of demodulators to reduce equal expressions to identical expressions. In fact, our set of demodulators forms a complete set of reductions [2] and so all equal expressions will indeed be reduced to identical ones. The demodulators have the finite termination property since the universe of terms for this problem was well ordered by our program and, when a demodulator is applied, the resulting term is less than the original term in the well ordering. They have the unique termination property since they apply to one another. If $EQUAL(A, B)$ and $EQUAL(A, C)$ both occur, then the first demodulates the second to $EQUAL(B, C)$, deleting $EQUAL(A, C)$ but maintaining its effect, since the term A will now demodulate first to B and then to C. Therefore it is impossible for a term to demodulate to two different terms.

For example, consider the example of order 7 which was found. The input relations were: $BC = BB$, $DD = BB$, and all products of three or more elements equal to BD, as well as to CE, EC, and CB.

This semigroup is described by entering the following clauses:

CL $EQUAL(F(B, F(C, V0)), F(B, F(B, V0)))$
CL $EQUAL(F(D, F(D, V0)), F(B, F(B, V0)))$
CL $EQUAL(F(C, F(E, V0)), F(B, F(D, V0)))$
CL $EQUAL(F(E, F(C, V0)), F(B, F(D, V0)))$
CL $EQUAL(F(C, F(B, V0)), F(B, F(D, V0)))$
CL $EQUAL(F(V0, F(V1, F(V2, V3))), F(B, F(D, V3)))$.

The methods of Sections 4.2 and 4.3 cause many other relations (demodulators) to be generated, among them

101 CL $EQUAL(F(C, F(D, V0)), F(B, F(D, V0)))$.

The involution $(BC)(D)(E)$ is considered by entering the clause

CL $PERM(C, B, D, E)$,

which produces a set of demodulators as described above. When clause 101 is clashed against clause 21 the result is

CL $EQUAL(F(F(J(REMVAR(F(D, V0)))), J(C)), V3),$
$F(J(REMVAR(F(B, F(D, V0)))), V3)$.

This is then immediately simplified, using all available demodulators, to

126 CL $EQUAL(F(B, F(D, V0)), F(B, F(E, V0)))$.

This clause is a new relation, and hence represents a contradiction. The exact nature of the contradiction is as follows. The relation $CD = BD$ (CL 101) is a consequence of the input relations. Applying J to both sides give us $BD = BE$ (CL 126), which is false in the semigroup being considered. The other possible involutions are eliminated similarly. In fact, clauses can be constructed which cause the theorem prover to loop through the possible involutions in one run.

Conclusion. The mathematical results contained here are of interest in their own right. Moreover, we have attempted to show by example how one might use a general purpose theorem-proving program as a tool in mathematical research.

Appendix. There are exactly four semigroups of order seven which support an antiautomorphism but no involution. One is given in Section 2.3 above. The remaining are:

	a	b	c	d	e	f	g
a	e	f	f	g	e	e	e
b	e	e	e	g	e	e	e
c	f	g	e	f	e	e	e
d	e	g	e	e	e	e	e
e	e	e	e	e	e	e	e
f	e	e	e	e	e	e	e
g	e	e	e	e	e	e	e

	a	b	c	d	e	f	g
a	f	f	f	g	e	e	e
b	e	g	e	g	e	e	e
c	f	g	f	f	e	e	e
d	e	g	e	g	e	e	e
e	e	e	e	e	e	e	e
f	e	e	e	e	e	e	e
g	e	e	e	e	e	e	e

	a	b	c	d	e	f	g
a	f	f	g	g	e	e	e
b	e	g	e	f	e	e	e
c	g	g	f	f	e	e	e
d	e	f	e	g	e	e	e
e	e	e	e	e	e	e	e
f	e	e	e	e	e	e	e
g	e	e	e	e	e	e	e

For all three semigroups, the antiautomorphism acts as follows:

$$h(a) = b \quad h(b) = c \quad h(c) = d \quad h(d) = a$$
$$h(e) = e \quad h(f) = g \quad h(g) = f.$$

Applied Mathematics Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439

Applied Mathematics Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439

Computer Science Division
Northern Illinois University
DeKalb, Illinois 60115

1. C. L. CHANG & R. C. T. LEE, *Symbol Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

2. D. E. KNUTH & P. B. BENDIX, "Simple word problems in universal algebras," *Computational Problems in Abstract Algebras* (J. Leech, Ed.), Pergamon Press, New York, 1970, pp. 263–297.

3. J. D. MCCHAREN, R. A. OVERBEEK & L. WOS, "Complexity and related enchancements for automated theorem-proving programs," *Comput. Math. Appl.*, v. 2, 1976, pp. 1–16.

4. J. D. MCCHAREN, R. A. OVERBEEK & L. WOS, "Problems and experiments for and with automated theorem-proving programs," *IEEE Trans. Comput.*, v. C-25, No. 8, 1976, pp. 773–782.

5. R. A. OVERBEEK, "An implementation of hyper-resolution," *Comput. Math. Appl.*, v. 1, 1975, pp. 201–214.

6. S. WINKER & L. WOS, *Automated Generation of Models and Counterexamples and Its Application to Open Questions in Ternary Boolean Algebra*, Proc. 8th Internat. Sympos. on Multiple-Valued Logic, Rosemont, Illinois, May 1978, IEEE, New York and ACM, New York, pp. 251–256.