AUTOMATED GENERATION OF MODELS AND COUNTEREXAMPLES AND ITS APPLICATION TO OPEN QUESTIONS IN TERNARY BOOLEAN ALGEBRA

Steve Winker and L. Wos

Prepared for

8th International Symposium on
Multiple-Valued Logic
Chicago, IL
May 24-26, 1978

U of C-AUA-USDOE

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated under Contract W-31-109-Eng-38 for the
U. S. DEPARTMENT OF ENERGY

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Department of Energy, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

AUTOMATED GENERATION OF MODELS AND COUNTEREXAMPLES AND ITS APPLICATION TO OPEN QUESTIONS IN TERNARY BOOLEAN ALGEBRA*

Steve Winker
Research Associate
Northern Illinois University
DeKalb, IL 60115

and

L. Wos
Applied Mathematics Division
Argonne National Laboratory
Argonne, IL 60439

Abstract

The thrust of this paper is: first, to answer certain previously unanswered questions in the field of Ternary Boolean algebra; second, to describe the method, utilizing an automated theorem-proving program as an invaluable aid, by which these answers were obtained; and third, to informally give the characteristics of those problems to which the method can be successfully applied. The approach under study begins with known facts in the form of axioms and lemmas of the field being investigated, finds by means of certain specified inference rules new facts, and continues to reason from the expanding set of facts until the problem at hand is solved or the procedure is interrupted. The solution often takes the form of a finite model or of a counterexample to the underlying conjecture. The model and/or counterexample is generated with the aid of an already existing automated theorem-proving procedure and without any recourse to any additional programming.

1. Introduction

In this paper we give a procedure which in part complements the ongoing effort of using computers to find mathematical proofs. The procedure is used to automatedly generate finite models of various mathematical theories and/or finite counterexamples to conjectures (for more details refer to a paper submitted for publication[8]). The ability to thus generate models and counterexamples when combined with an automated theorem-proving program provides one with the more effective attack on certain open questions in mathematics and logic. We invite the submission for consideration of such open questions as those answered in section 2 and of those informally classified in section 4. Such new problems are of value in the development, extension, and refinement of the approach under discussion, and there is, of course, the carrot that some of them may be thus solved. This in fact was the case for those problems in the field of Ternary Boolean algebra which were considered and treated with the procedure under investigation.

The organization of the paper is as follows. In section 2, Ternary Boolean algebras are defined, the previously open questions are stated, and answers are given including certain key models and lemmas. In section 3, the procedure for generating models is discussed. It is here we give both the language in which the problems must be stated and the underlying inference rules which generate additional facts and permit validation of the models. Finally, in section 4 we discuss the kinds of problems and fields of mathematics which are most amenable to the approach of section 3.

We conclude the section with the following summary. The proofs of the various lemmas and the models and counterexamples given here were obtained with the aid of the computer and, more specifically, with extensive use of an automated theorem-proving program[5,6,9,10]. No additional programming was required to obtain our results, nor would any be necessary to attack similar open questions. Since, in addition to model generation, the approach herein can be used for deductive reasoning, it may be of use in areas other than mathematics.

2. Ternary Boolean Algebras: The Solution to Some Open Questions

A Ternary Boolean algebra[4] is a set S together with a Ternary function f and a unary function g which, for all elements in S, satisfy:

(1) $f(f(v,w,x),y,f(v,w,z)) = f(v,w,f(x,y,z))$
(2) $f(y,x,x) = x$
(3) $f(x,y,g(y)) = x$
(4) $f(x,x,y) = x$ and
(5) $f(g(y),y,x) = x$.

One natural question to be asked is: Of the five axioms defining a Ternary Boolean algebra, which (if any) among them are dependent on the remaining? Chinthayamma[3] announced in 1969 without proof that axioms 4 and 5 were dependent on the subset consisting of 1, 2, and 3. This left open the question of which of the remaining proper subsets of the five axiom set are strong enough to define a Ternary Boolean algebra. Then, for each subset, U, which is too weak, one can ask: what are the more interesting properties of U, and what is the cardinality of the smallest model which satisfies U and simultaneously fails to satisfy the remaining of the five axioms which define a Ternary Boolean algebra. The answer to the foregoing can be thus summarized.

The most important facts, obtained with the procedure discussed in section 3, are the following. Let T be the set of the five defining

* Work performed under the auspices of the U.S. Department of Energy and NSF grant MCS77-02703.

axioms, U_1 be obtained from T by the omission of axiom 1, U_2 by the omission of axiom 2, and U_3 by the omission of axiom 3. There are models, exhibited later in this section, showing that each of U_1, U_2, and U_3 is insufficient to define a Ternary Boolean algebra. Equivalently, axioms 1, 2, and 3 are each independent of the remaining four axioms in T. Next, one can show that in U_2, f(g(g(x)),x,y) = g(g(x)) for all x and y. In U_3, f(x,g(x),y) = y for all x and y. Axioms 1 and 4 together imply that f(x,y,x) = x. In T, g(g(x)) = x and finally, from axioms 4 and 5 one can show that, if there exists an element a in the set under consideration such that g(a) = a, then the set consists of just a. Since the establishment of assertions of the type just given reduces the work required to generate the appropriate models, we turn to their proof.

The last of the assertions can be seen from x = f(g(a),a,x) = f(a,a,x) = a for any x. That f(x,y,x) = x in the presence of 1 and 4 can be seen by simply setting v = w in 1.

Next, we see that g(g(x)) = x when all of T is present by setting in axiom 1, w = x, v = g(g(x)), y = z = g(x), and applying 2 and 3. (It should be noted that the program employed by the procedure section 3 does not find this and similar proofs by considering all or various substitutions into the various axioms. Instead, selections from the axioms and derived facts are made and presented to the inference rules which then in turn make a deduction with an emphasis on generality. Then a subprocedure is invoked which compares the "new" result with those already retained. The procedure will purge, for example, a deduction D' in favor of D, regardless of which is newer, when D is more general than D'.)

Returning to the assertions to be proved, we have U_2 implies f(g(g(x)),x,y) = g(g(x)). In axiom 1, just set x = y = g(w), and v = g(g(w)), and apply 3, 4, and 5 to get an alphabetic variant of the desired result.

Finally, the following six-step proof shows that one can prove from U_3, f(x,g(x),y) = y.

Proof. f(f(v,g(v),x),x,v) = f(v,g(v),x) from the substitution into axiom 1, respectively for v,w,x,y,z of v,g(v),x,x,v, and from applications of f(x,y,x) = x and axiom 4. f(f(v,g(v),x),v,x) = f(v,g(v),x) from the substitution into axiom 1 respectively of f(v,g(v),x),x,v,v,x, and applications of the previous step and of axiom 2 and axiom 4. f(f(v,w,g(w)),w,v) = v from substitution into axiom 1 of v,w,g(w),w,v, and applications of f(x,y,x) = x and axiom 5. f(w,g(w),f(v,w,g(w))) = f(v,w,g(w)) from the substitution into axiom 1 of v,w,w,g(w),g(w), and applications of axiom 2. f(f(v,g(v),x),x,f(x,v,g(v))) = f(v,g(v),x) from the substitution into axiom 1 of v,g(v),x,x,f(x,v,g(v)), and applications of axiom 4 and of the previous step. Finally, f(v,g(v),w) = w from the substitution into axiom 1 of f(v,g(v),w),w,f(w,v,g(v)),v,w, and applications of axiom 2 and steps 3, 5, and 2, which completes the proof.

Thus we have completed the proofs of the various assertions given above, and we can now turn to the direct consideration of the question of axiom dependence for Ternary Boolean algebras. As stated earlier, the facts are that axioms 4 and 5 are dependent on the remaining three, but none of axioms 1, 2, or 3 is dependent on the other four. Equivalently, the only subsets of the given set of five axioms defining a Ternary Boolean algebra which are strong enough for the definition are those subsets which contain at least axioms 1, 2, and 3.

First, to see that axiom 1 is independent of the others, merely consider the three-element model, consisting of a, b, and c, with g(g(g(x))) = x, g(a) = b, g(b) = c, f(a,b,a) = b, f(b,c,b) = c, f(c,a,c) = c, and all of the triples satisfying axioms 2 through 5. The substitution into axiom 1 respectively for v,w,x,y,z of a,b,c,c,a respectively yields a = b, which shows that axiom 1 does not hold.

Next we establish the independence of axiom 2. Consider the model consisting of the three elements, a, b, and c such that g(g(g(x))) = x for all x, g(x) not equal x for all x, g(a) = b, f(x,x,y) = x for all x and y, f(g(y),y,x) = x for all x and y, and f(g(g(x)),x,y) = g(g(x)) for all x and y. First note that g(b) = c and g(c) = a. Then, by a tedious examination of all triples, one can verify that axioms 1, 3, 4, and 5 hold. The violations of axiom 2 are three instances of f(g(g(x)),x,y) = g(g(x)), namely, f(a,b,b) = a and f(b,c,c) = b and f(c,a,a) = c.

(Before turning to the final dependency question and then to the question of minimal counterexamples, we make the following observations which are important to the understanding of the procedure by which the results were obtained.) The procedure of section 3 does the tedious checking of the various triples required to validate the proposed model. One phase of the process is the check for consistency. For example, in the just-given model one could evaluate f(a,a,b) with axiom 4 or with axiom 3. In both choices the same value must be obtained. Also note that the model is not presented to the procedure by means of precisely those equalities given in the previous paragraph, but rather by a set which includes the axioms to be satisfied. The other equalities are derived and used to narrow the search for the desired model or counterexample.

Now to establish the independence of axiom 3, just replace in the previous three-element model the requirement of f(g(g(x)),x,y) = g(g(x)) with the requirement of f(x,g(x),y) = y. Axiom 2 will now hold since, for example, f(a,b,b) = b by the new equality. On the other hand, axiom 3 is now violated for the value of three triples has changed. The three violations are f(a,b,c) = c, f(b,c,a) = a, and f(c,a,b) = b from the new equality.

(For the interest of the reader, axioms 4 and 5 may be proved from axioms 1, 2, and 3 as follows.

The proof that g(g(x)) = x depends only on axioms 1, 2, and 3. Substitute g(g(x)),x,y,g(x),g(x) in axiom 1 and apply 2, 3, and g(g(x)) = x to yield 4. Substitute x,g(y),g(y),g(g(y)),g(g(y)) in 1 and apply 2, 3, and g(g(x)) = x to yield 5.)

The final question, that of minimality for the counterexamples to axiom dependency, can be settled with the following argument. For two-element models, either there exists an x with g(x) = x, or g interchanges a and b. The first possibility is eliminated by an earlier remark, i.e., the presence of axioms 4 and 5 would force then a = b to be true. For the second possibility, axiom 5 forces f(b,a,a) = a and f(a,b,b) = b, which says that axiom 2 holds. Also in this case, axiom 5 forces f(a,b,a) = a and f(b,a,b) = b, while axiom 4 forces f(a,a,b) = a and f(b,b,a) = b, which says that axiom 3 holds. A similar analysis shows that axiom 1 also holds in this case. So the smallest counterexample to the dependence of 1, 2, or 3 consists of three elements.

Returning to the discussion of the models themselves, we illustrate a somewhat different use of the procedure of section 3. The problem for consideration is the generation, if such exists, of an asymmetric three-element model of U_2. The specific objective is that of determining whether or not any three-element models exist satisfying axioms 1, 3, 4, and 5, violating axiom 2, and with g not an onto mapping. There are two such models. In each we have g(g(g(x))) = g(x), g(a) = b, g(b) = c, g(c) = b, f(x,x,y) = x, f(g(y),y,x) = x, and f(x,y,g(y)) = x. In both, from the earlier results, f(x,y,x) = x and f(g(g(x)),x,y) = g(g(x)); also f(a,c,c) = a (substitute a,c,b,a,c in axiom 1 and apply 3, 4, and 5). They differ in that in one, f(a,b,b) = b while in the other f(a,b,b) = a. In both the violations of axiom 2 are f(c,a,a) = c and f(a,c,c) = a. Examination of the various substitutions shows that axiom 1 holds for both models.

When the problem for the previous paragraph is replaced by the corresponding one for U_3, we find that there is but one asymmetric model of 1, 2, 4, and 5 violating axiom 3. The model is quite like that which was just given except that axiom 3 is replaced by axiom 2, f(g(g(x)),x,y) = g(g(x)) is replaced by f(x,g(x),y) = y, f(a,c,b) = b, and f(c,a,b) = b. The last two equalities are, of course, the violations of axiom 3.

The above models are the only three-element models of U_2 and U_3 violating axioms 2 and 3, respectively. First, observe that there are only two possibilities for the effect of g on three elements: the "symmetric" and the "asymmetric" possibilities given above (g(x) cannot be x for any x as noted previously). Secondly, the possibilities for values of f not given above are eliminated by contradictions to axiom 1 which were found by use of the program.

3. The Main Procedure for Generating Models

Since the presentation in subsections 3.1 to 3.6 is brief and somewhat intuitive, we have included an example in 3.7 to aid one's understanding of the basic procedure and of the various underlying concepts. The example illustrates the iterative nature of our procedure. Its development in part parallels the mathematical treatment of section 2, and illustrates the method employed to obtain one of the asymmetric models of U2 given there.

3.1 Overview

The main objective is the development and implementation of a more complete procedure for attacking open questions in mathematics and logic. It is important, in the treatment of such questions, to have a procedure for generating models and counterexamples. Such a procedure, based on an existing automated theorem-proving program, is the focus of attention in this section. For the side of the problem concerned with finding proofs for "true" theorems, there exists computer programs in various stages of development whose objective is that of proof finding. The proofs so obtained are usually ones by contradiction. In general, one begins with a set of statements, some of which correspond to axioms and lemmas while others correspond to the denial of the theorem to be proved, and continually applies rules of inference until the unsatisfiability of the set of statements is established. What is missing from such programs is an automated treatment of the other side of the problem, namely, the establishment with the aid of the computer that a desired result does not hold. Put differently, when the given set of statements is satisfiable, one would like to have a procedure which establishes that fact -- a procedure which generates a counterexample to the purported theorem. Although the fundamental theorems for the first-order predicate calculus prevent us from having a decision procedure, we have been able to establish that certain results are non-theorem and thus answer certain previously open questions as those of section 2.

With the object of automatedly generating models, we turn to a brief description of the various components.

3.2 The Language

The chosen language, employing the clause representation,[2] is one which is closely related to the extended first-order predicate calculus.[2] The equality symbol (predicate) is thus treated as a special symbol with its meaning "built-in". All variables and all given and inferred statements are (implicitly) assumed to be universally quantified. The existentials have been replaced by skolem functions,[2] functions of the appropriate universally quantified variables on which the existentials depend. The ensemble of statements is (implicitly) assumed to be in conjunctive normal form -- there is an implicit "and" between pairs of statements, and implicit "or" between the elements of each statement.

3.3 The Inference Rules

Although the procedure has access to a number of inference rules, the two most frequently employed are respectively generalizations of equality substitution and of modus ponens. The first rule, called paramodulation,[9] takes clauses (statements) in pairs and attempts to substitute from one into the other. A substitution occurs if and only if a common domain of definition can be found for one of the arguments of the "from" clause and for a term in the "into" clause. The "from" clause must be the correspondent of some given statement of equality, while no essential restriction is placed on the term into which the substitution takes place. For further clarification and also to see that we are employing a generalization of equality substitution, note that paramodulation in one inference step takes the pair of statements (clauses), finds when possible a most general replacement of the variables (universal instantiation) in both which will permit a straightforward application of equality substitution, and applies an equality substitution rule to the instantiated pair.

As for the other main rule, called hyperresolution,[7] the following cursory description may suffice. The rule takes a set, Q1,Q2,...,Qn, of positive assertions, an "if-then" statement of the form $Q'_1 \& Q'_2 \& \ldots \& Q'_N \rightarrow R$, finds (where possible) a most general variable replacement to simultaneously apply to the Q_i and Q'_i which permits a modus ponens type inference, and then makes that inference from the thus instantiated clauses. There is the additional requirement that R be positive.

3.4 Usage

No programming is required of the individual who intends to use the procedure as an aid in answering questions. What is required is the preparation of the problem in one of two forms. The procedure itself accepts problems in the clause forms discussed in 3.2. Recall that one is therefore using a conjunctive normal form in which the variables that appear are implicitly universally quantified, the existentials have been replaced by appropriate skolem functions, and an implicit "and" occurs between clauses while an implicit "or" occurs between the literals of a clause. For example, the clause equivalent of that axiom which states that the nonzero elements of a field possess a multiplicative inverse consists of the two clauses, EQUAL(X,0) EQUAL(F(H(X),X),1) and EQUAL(X,0) EQUAL(F(X,H(X)),1), where F denotes product and H inverse and 1 the multiplicative identity. (There are other valid clause encodings of this axiom.) The scope of universal quantification is just that single clause, i.e., an occurrence of the variable "X" in two clauses does not mean the "same" variable. So one can submit the problem directly to the main procedure by encoding it in clause form.

On the other hand, if one finds such an encoding difficult or inconvenient, one can instead choose to represent the problem in the first-order predicate calculus. There is no requirement, in such a choice, of using prenex form and no restriction on the use of the various boolean connectives. The availability to the user of the first-order representation is due to the existence of a system, called TAMPR,[1] designed and implemented at Argonne National Laboratory. TAMPR, although designed for program transformations of a different type, will take the problem in its first-order form and produce the clauses with appropriate replacement of existentials by functions.

3.5 The Procedure Itself

We choose to slant our description toward the informal and intuitive. The procedure can be said to be divided into a number of phases from which the user can choose any or all. There is the lemma generation or fact finding, the counterexample and/or model building, the validation of the computed counterexample and/or model, the rejection of such, and the proof of the theorem. Thus, despite the bias that may exist when considering a particular open question, the procedure may succeed in proving the corresponding purported theorem or may instead generate a counterexample. It is the second of these alternatives (in the closely related area of generating models for consistent axiom systems) on which we concentrate.

The approach is at present one of iteration in which one begins with a set of statements which may include axioms, lemmas, and various conjectures about the definition or structure of the sought-after counterexample or model. (Throughout the rest of this subsection we make no distinction between "counterexample" and "model.") One then makes a series of computer runs with the intention of appropriately adding to and/or deleting from the original input set. Additions are either lemmas which were not already present or "promising" extensions to the definition of a model. The validity of any of the former can be established by examining its proof tree to show that only already-proven theorems are relied upon, while validity of various of the latter may remain in question until the model is completed because of reliance on other conjectured conditions. (The derivation information is included in the computer output of each run.) Deletions, on the other hand, are usually the result of detecting inconsistency in the conditions defining the model. Inconsistency is signalled by the deduction of "contradiction" by the program. Such deletions often cause a fair amount of back tracking because of the corresponding necessity of making new conjectures about the structure of the model, and this may in turn require duplication of earlier runs but with, of course, the new conjectures. Thus the additions and deletions made by the user in earlier runs in part determine the nature of later runs.

One of the nice features of our procedure is that these computer experiments are accomplished with the aid of an existing automated theorem-proving program and require no additional programming. Each experiment or run is terminated either by exceeding memory or time, or by having made all possible new inferences, or deducing contradiction. The second termination condition occurs either when the model has been both successfully completed and validated or when the model is partially specified but no inconsistencies exist therein. One can differentiate between the two cases by simply examining the computer output. The third termination condition signals model inconsistency or proof of the theorem thus answering the open question under study. Examination of the proof is sufficient to determine which is the case. Finally, the first termination condition can occur with any use in any phase of the procedure.

For each of the phases of the procedure (listed at the beginning of this subsection), the following remarks in general hold. Lemmas are found through use of the inference rule, paramodulation, which is a generalization of equality substitution (see 3.3). New information is checked against that already present, and the more general fact is retained and the less general purged. Both the building and validation of the model are accomplished through application of hyper-resolution, a modus ponens-like rule discussed also in 3.3. The rejection of the model being developed is through a combination of inferences from paramodulation and from hyper-resolution. And finally, the proofs of theorems may be obtained from various inference rules, but paramodulation is often the most successful.

3.6 Applications

The procedure under investigation has been used to generate the multiplication table for various semi groups, for successfully searching for models establishing the correctness of certain conjectures, to generate counterexamples and thereby refute various possible axiom dependencies as in section 2, and for finding proofs for various theorems as also given in section 2. All models and axiom sets considered so far have been of small cardinality.

3.7 An Example of Our Procedure

Some of the automated theorem prover runs made in searching for one model of U2 are listed below to indicate the degree of our reliance on the computer. As might be expected, the search includes tests which appear inconclusive or unnecessary in retrospect.

1. Paramodulation runs proved TBA axioms 4 and 5 from axioms 1, 2, and 3, and incidentally derived $g(g(x)) = x$ from axioms 1, 2, and 3.

2. A paramodulation run attempting to prove axiom 2 from axioms 1, 3, 4, and 5 proved neither axiom 2 nor $g(g(x)) = x$. Incidentally, $f(x,y,x) = x$ was derived.

   The failure to prove axiom 2 motivated the search for a counterexample. The fact that $g(g(x)) = x$ was not derived suggested that a model violating $g(g(x)) = x$ be attempted. Such a model would necessarily violate axiom 2. It could be based on one generator (an a for which $g(g(a)) \neq a$) rather than two (an a and b for which $f(b,a,a) \neq a$), possibly requiring fewer elements, fewer defining relations, and less computer time for verification.

3. Paramodulation run seeking consequences of axioms 1, 3, 4, and 5, in conjunction with $g(g(g(x))) = g(x)$, $g(f(x,y,z)) = f(g(x),g(y),g(z))$, and $f(x,y,z) = f(x,z,y)$. Axiom 2 was not proved, but the last equality with axioms 3 and 5 yielded $g(g(x)) = x$.

   Because a model with $g(g(a)) \neq a$ was being sought, the last equality was not used for subsequent models. The possibility that $g(g(g(x))) = g(x)$ might imply $g(g(x)) = x$ was not tested at this time.

4. A paramodulation run deriving consequences of axioms 1, 3, 4, and 5, in conjunction with $g(f(x,y,z)) = f(g(x),g(y),g(z))$ and $f(a,x,g(g(a))) = a$, derived no undesirable consequences.

   The latter equality was suggested by an examination of the proof of $g(g(a)) = a$ from axioms 1, 2, and 3. This proof used the instance of axiom 2, $f(a,g(g(a)),g(g(a))) = g(g(a))$; if this term had the value a instead, $g(g(a)) = a$ would not be proven. $f(a,x,g(g(a))) = a$ generalizes the second argument of f.

5. Partial-model run in which values for $f(a,c,c)$, $f(c,a,a)$, and $f(a,b,b)$ had not yet been determined. (Here b and c refer not to generators but to $g(a)$ and $g(g(a))$.)

6. Partial-model run in which the value for $f(a,c,c)$ had not yet been determined.

7. Model validation run verifying the first asymmetric model of section 2.

4. Requirements for New Problems

The submission of problems for consideration by the procedure described in this paper is most welcome. To be admissible, such problems must be representable in the first-order predicate calculus. If the object is the generation of a counterexample or model, there is the assumption that one of small finite cardinality will suffice. On the other hand, if the object is the finding of a proof for a purported theorem, we only require a statement of the theorem and a set of axioms characterizing the field from which the theorem is taken. In addition we find value in having a list of the important lemmas of the field.

Among those areas whose open questions may be most amenable to attack with the automated procedure are the theory of semi groups, elementary group theory, ring theory, the theory of Ternary Boolean algebras, Boolean algebra, and Tarskian geometry. For the type of question, one might consider, for example, questions concerned with the equivalence of axiom systems, questions about the existence of certain mappings, and questions of axiom dependency. On the other hand, phrases such as "for all integers n" and "for all functions f" strongly suggest the lack of an appropriate mechanism to handle the question.

We conclude by remarking that consideration of open questions and problems of the type just discussed should, when subjected to treatment by computer programs of the type underlying this paper, lead to the alternation of the solution of some problems followed by the development of more successful automated procedures followed by the solution to others... .

References

1. Boyle, J. and Matz, M., Automating Multiple Program Realization, *Proceedings of the Symposium on Computer Software Engineering*, Polytechnic Press, New York, 1976, pp. 421-456.

2. Chang, C. L. and Lee, R. C. T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

3. Chinthayamma, Sets of independent axioms for a ternary Boolean algebra (preliminary report), *Notices of the Amer. Math. Soc*, Vol. 16, No. 4, June 1969, p. 654.

4. Grau, A. A., Ternary Boolean Algebra, *Bulletin of Amer. Math. Soc.*, Vol. 53, No. 6, June 1947, pp. 567-572.

5. McCharen, J. D., Overbeek, R. A., and Wos, L., Problems and Experiments for and with Automated Theorem-Proving Programs, *IEEE Transactions on Computers*, Vol. C-25, No. 8, August 1976, pp. 773-782.

6. Overbeek, R. A., An implementation of hyper-resolution, *Comput. Math. Appl.*, Vol. 1, 1975, pp. 201-214.

7. Robinson, J. A., Automatic Deduction with Hyper-resolution, *International Journal of Computer Math.*, Vol. 1, 1965, pp. 227-234.

8. Winker, Steve, Generation and Validation of Finite Models and Counterexamples using an Automated Theorem Prover, submitted for publication.

9. Wos, L. and Robinson, G. A., Paramodulation and Set of Support, *Proc. IRIA Symposium on Automatic Demonstration*, Versailles, France (1968), Springer-Verlag, 1970, pp. 276-310.

10. Wos, L., Robinson, G. A., Carson, D. F., and Shalla, L., The concept of demodulation in Theorem proving, *J. Assn. Comput. Mach.*, Vol. 14, 1967, pp. 698-709.