A Computing Procedure for Quantification Theory*

MARTIN DAVIS

Rensselaer Polytechnic Institute, Hartford Division, East Windsor Hill, Conn.

AND

HILARY PUTNAM

Princeton University, Princeton, New Jersey

The hope that mathematical methods employed in the investigation of formal logic would lead to purely computational methods for obtaining mathematical theorems goes back to Leibniz and has been revived by Peano around the turn of the century and by Hilbert's school in the 1920's. Hilbert, noting that all of classical mathematics could be formalized within quantification theory, declared that the problem of finding an algorithm for determining whether or not a given formula of quantification theory is valid was the central problem of mathematical logic. And indeed, at one time it seemed as if investigations of this "decision" problem were on the verge of success. However, it was shown by Church and by Turing that such an algorithm can not exist. This result led to considerable pessimism regarding the possibility of using modern digital computers in deciding significant mathematical questions. However, recently there has been a revival of interest in the whole question. Specifically, it has been realized that while no *decision procedure* exists for quantification theory there are many proof procedures available—that is, uniform procedures which will ultimately locate a proof for any formula of quantification theory which is valid but which will usually involve seeking "forever" in the case of a formula which is not valid—and that some of these proof procedures could well turn out to be feasible for use with modern computing machinery.

Hao Wang [9] and P. C. Gilmore [3] have each produced working programs which employ proof procedures in quantification theory. Gilmore's program employs a form of a basic theorem of mathematical logic due to Herbrand, and Wang's makes use of a formulation of quantification theory related to those studied by Gentzen. However, both programs encounter decisive difficulties with any but the simplest formulas of quantification theory, in connection with methods of doing propositional calculus. Wang's program, because of its use of Gentzen-like methods, involves exponentiation on the total number of truth-functional connectives, whereas Gilmore's program, using normal forms, involves exponentiation on the number of clauses present. Both methods are superior in many cases to truth table methods which involve exponentiation on the

* Received September, 1959. This research was supported by the United States Air Force through the Air Force Office of Scientific Research of the Air Research and Development Command, under Contract No. AF 49(638)-527. Reproduction in whole or in part is permitted for any purpose of the United States Government.

total number of variables present, and represent important initial contributions, but both run into difficulty with some fairly simple examples.

In the present paper, a uniform proof procedure for quantification theory is given which is feasible for use with some rather complicated formulas and which does not ordinarily lead to exponentiation. The superiority of the present procedure over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by *hand computation* using the present method in 30 minutes. Cf. §6, below.

It should be mentioned that, before it can be hoped to employ proof procedures for quantification theory in obtaining proofs of theorems belonging to "genuine" mathematics, finite axiomatizations, which are "short," must be obtained for various branches of mathematics. This last question will not be pursued further here; cf., however, Davis and Putnam [2], where one solution to this problem is given for elementary number theory.

1. *General Remarks*

We shall describe a computational procedure, or algorithm, which when applied to a logically valid formula written in the notation described below will terminate and yield a proof of the validity of that formula; for formulas which are not logically valid, the computation will continue indefinitely without giving a result.[1]

The symbols of which our formulas are constructed are divided into the classes: punctuation marks, logical symbols, (individual) variables, predicate symbols, and function symbols. The punctuation marks are:

$$, \qquad (\qquad)$$

The logical symbols are:

$$\sim \qquad \& \qquad \vee \qquad \rightarrow \qquad \leftrightarrow \qquad E$$

We shall take as the variables the terms of the following infinite sequence:

$$x_1 \quad x_2 \quad x_3 \quad x_4 \quad \cdots$$

The predicate symbols will be the letters F, G, H, with or without subscripts, and the function symbols[2] will be the terms of the infinite sequence:

$$f_1 \quad f_2 \quad f_3 \quad f_4 \quad \cdots$$

Among all of the expressions (e.g. $\rightarrow \vee Fx_5E$) which can be formed using these symbols, we distinguish three classes: the *terms*, the *atomic formulas*, and the *well-formed formulas* (abbreviated *w.f.f.*).

[1] Since by results of Church and Turing the set of formulas involved is a recursively enumerable set which is not recursive (for terminology, and a proof of this fact, cf. [1] or [6]), this kind of algorithm is the best one can hope to obtain.

[2] We intend to use function symbols not only to stand for functions of one or more arguments but also for individuals. In the latter use they may be thought of as standing for functions of zero argument.

The notion *term* will be defined inductively:

(1) *The expressions f_i and x_i are terms for each $i = 1, 2, 3, \cdots$.*

(2) *If $p_1, p_2, \cdots, p_n$ are terms,*[3] *then so is $f_i(p_1, p_2, \cdots, p_n)$, and $p_1, p_2, \cdots, p_n$ are called the arguments of f_i.*

(3) *The terms consist exactly of the expressions generated by (1) and (2).*

Next:

The expression $p(p_1, p_2, \cdots, p_n)$ is an atomic formula if p is a predicate symbol and $p_1, p_2, \cdots, p_n$ are terms. $p_1, p_2, \cdots, p_n$ are called the arguments of p.

Finally:

(1) *An atomic formula is a w.f.f.*

(2) *If R is a w.f.f., then so are $\sim R$, $(x_i)R$, and $(Ex_i)R$.*

(3) *If R and S are w.f.f.'s, then so are $(R \,\&\, S)$, $(R \vee S)$, $(R \rightarrow S)$, and $(R \leftrightarrow S)$.*

We introduce the following abbreviative conventions:

a stands for f_1.

f stands for f_2.

pq stands for $p(q)$ if p is a function symbol and q is a term.

$\bar{p}(p_1, p_2, \cdots, p_n)$ stands for $\sim p(p_1, \cdots, p_n)$, where p is a predicate symbol and $p_1, \cdots, p_n$ are terms.

An occurrence of x_i in a w.f.f. R is a *bound occurrence* if it is in a w.f. part of R of the form $(x_i)P$ or $(Ex_i)P$. An occurrence of x_i which is not bound is called a *free occurrence*. x_i is *free* in R if it has at least one free occurrence in R.

If $x_{i_1}, x_{i_2}, \cdots, x_{i_n}$ are all of the free variables in R, we sometimes write $R(x_{i_1}, x_{i_2}, \cdots, x_{i_n})$ for R. If $p_1, p_2, \cdots, p_n$ are terms, we write $R(p_1, p_2, \cdots, p_n)$ for the result of replacing x_{i_k} by p_k, $k = 1, 2, \cdots, n$, at all free occurrences of x_{i_k} in R.

Parentheses will be omitted wherever their omission can cause no confusion.

Our next step is to single out from the class of w.f.f.'s those which are *logically valid.* This can be done either by specifying axioms and rules of inference or by referring to "interpretations" of the w.f.f.'s of the system, and by a basic result due to Gödel[4] both of these procedures will lead to the same class of formulas. For our present purposes it is most convenient to use the latter formulation employing "interpretations."

An *interpretation* for a formula R consists of a nonempty set of elements U called a *universe* and an assignment of "values" to each function symbol and predicate symbol as follows:

To each function symbol which occurs in R with n arguments,[5] we assign a function of n variables ranging over U, whose values are in U.[6]

To each predicate symbol which occurs in R with n arguments, we assign a

[3] Note that the symbols p_1, p_2, etc. occur here as "syntactic variables." That is, they stand for expressions made up of our symbols.

[4] The Gödel completeness theorem. Cf. [5], [6], or [7].

[5] Thus, if $n = 0$, f_i is assigned an element of U.

[6] Note that if f_i occurs in R both with m arguments and with n arguments, $m \neq n$, it is assigned different functions in each case. In practice this will not happen in examples considered below. (However, two occurrences in R of f_i with the same number of arguments are, of course, to be assigned the same value.)

function of n variables ranging over U, whose values are the truth values, 0 (*falsehood*) and 1 (*truth*).[7]

Let $R(x_{n_1}, x_{n_2}, \cdots, x_{n_k})$ be a w.f.f. Then, given an interpretation of R over universe U, the value 0 or 1 will be assigned to $R(t_1, t_2, \cdots, t_k)$ for each ordered k-tuplet $(t_1, t_2, \cdots, t_k)$ of elements of U. This value may be obtained simply by interpreting 0 as falsehood and 1 as truth, using the usual truth tables for $\sim$ & $\vee$ $\rightarrow$ and $\leftrightarrow$, interpreting $(x_i)P(x_i)$ as 0 unless $P(t)$ has the value 1 for all t in U, and interpreting $(Ex_i)P(x_i)$ as 1 unless $P(t)$ has the value 0 for all t in U.

A w.f.f. R is called *valid* if under every interpretation and for every set of arguments from U, R is assigned the value 1.

A w.f.f. R is called *consistent* (or *satisfiable*) if there is some interpretation under which R is assigned the value 1, for *some choice* of *arguments from* U. R is *inconsistent* if it is not consistent.

We shall make use of the obvious fact that:

R is valid if and only if $\sim R$ is inconsistent.

That is, to "prove" R it suffices to "refute" $\sim R$, and indeed our *proof procedure* for validity will be couched in the form of a *refutation procedure.*

R is called *logically equivalent* to S if the *w.f.f.* $(R \leftrightarrow S)$ is valid.

A w.f.f. is called *quantifier-free* if it contains no occurrence of (x_i) or (Ex_i). A w.f.f. is a *prenex formula*, or in *prenex normal form*, if it begins with a sequence of quantifiers (x_i) and (Ex_i) in which no variable occurs more than once (called the *prefix*) and if the sequence is followed by a quantifier free w.f.f. (called the *matrix*). An example of a prenex formula is:

$$(x_1)(Ex_3)(x_7)(Ex_2)F(f(x_3), f_3(x_1, x_2), x_5)$$

S is called *a prenex normal form of R* if S is a prenex formula which is logically equivalent to R. There is a simple algorithm (cf. [5], [7]), for obtaining a prenex normal form of a given w.f.f. *Thus, for the purpose of our refutation procedure it suffices to consider prenex formulas.*

The *disjunction of* $R_1, \cdots, R_n$, $n \geqq 1$, is the w.f.f. $R_1 \vee R_2 \vee \cdots \vee R_n$; their *conjunction* is the w.f.f. $R_1 \& R_2 \& \cdots \& R_n$. A *literal* is a w.f.f. which is either an atomic formula or $\sim R$, where R is atomic. A *clause* is a disjunction $R_1 \vee R_2 \vee \cdots \vee R_n$ in which each R_i is a literal and in which no atomic formula occurs twice. (E.g., $F(x_1) \vee \bar{G}(x_2, x_3)$ is a clause, but $F(x_1) \vee \bar{F}(x_1)$ is not.)

A conjunction of clauses is said to be *a formula in conjunctive normal form.*

EXAMPLE: $(p \vee q \vee \bar{r})$ & $(s \vee \bar{t})$ is a formula in conjunctive normal form if p, q, r, s, t are atomic formulas.

If a w.f.f. A is in conjunctive normal form and A is logically equivalent to B, then A is called a *conjunctive normal form of B.*

EXAMPLE: $(p \vee \bar{q})$ & $(q \vee \bar{p})$ is a conjunctive normal form of $p \leftrightarrow q$ if p and q are any atomic formulas.

For further discussion of conjunctive normal form the reader may consult Hilbert and Ackermann [5]. In particular, there is a simple algorithm by which

[7] The comment in footnote 6 regarding function symbols applies also to predicate symbols.

a conjunctive normal form is obtainable for any quantifier-free formula which is not valid; if the formula is valid the same algorithm will establish that fact. (Cf. [5].) Hence, we may assume that the w.f.f. which is offered for refutation is a prenex formula whose matrix is in conjunctive normal form. Later we shall see why this is a useful and practical assumption.

2. *Replacement of Existential Quantifiers by Function Symbols*

The refutation algorithm to be presented will exploit the following idea (which, in essence, goes back to Lowenheim): that existential quantifiers in a prenex formula can be replaced by function symbols without affecting consistency. The notion may be clarified by an example: Suppose the given prenex formula is

$$(x_1)(Ex_2)(Ex_3)(x_4)(Ex_5)R(x_1, x_2, x_3, x_4, x_5), \qquad \text{(i)}$$

where the matrix $R(x_1, x_2, x_3, x_4, x_5)$ is supposed to be quantifier-free and to contain ony the free variables indicated. Then the formula (i) is consistent only if the formula

$$(x_1)(x_4)R(x_1, f_2(x_1), f_3(x_1), x_4, f_5(x_1, x_4)) \qquad \text{(ii)}$$

is, where f_2 and f_3 are one-place function symbols and f_5 is a two-place function symbol. To verify this, observe that (ii) logically implies (i), so if (ii) is consistent, so is (i). On the other hand, if (i) is true in some universe U (under some interpretation of the predicate letters in R), then there are functions[8] f_2, f_3 and f_5 over U such that (ii) is true in U under the same interpretation of the predicate letters in R. Thus if (i) is consistent, so is (ii).

Throughout the present paper, accordingly, the instruction "replace the existential quantifiers in F by function symbols" (where F is a prenex formula) will have the following meaning: Let the variables in the prefix of F (in order of occurrence) be $x_1, x_2, \cdots, x_N$. Let the *existentially* quantified variables in the prefix be $x_{i_1}, x_{i_2}, \cdots, x_{i_M}$. Then, (1) the quantifier (Ex_{i_j}) (for $j = 1, 2, \cdots, M$) is to be deleted from the prefix, and (2) each occurrence of x_{i_j} in the matrix is to be replaced by an occurrence of the term $f_{i_j}(x_{q_1}, x_{q_2}, \cdots, x_{q_p}$ where $(x_{q_1}), (x_{q_2}), \cdots, (x_{q_p})$ are all the universal quantifiers that precede (Ex_{i_j}) in the prefix of F.

In the above example, following the instruction "replace the existential quantifiers in (i) by function symbols," as just explained, would lead to formula (ii). Finally, (recalling that 0-place function symbols are interpreted simply as individual constants) replacing the existential quantifiers by function symbols in

$$(Ex_1)(x_2)(Ex_3)(x_4)M(x_1, x_2, x_3, x_4)$$

[8] This agreement tacitly employs a nonconstructive principle known as the Axiom of Choice. Alternatively, one can use the theorem that if (*i*) is consistent then (*i*) has a true interpretation in some denumerable universe U (Skolem-Löwenheim theorem; cf. [7], pp. 253–260), and then explicitly define the functions f_2, f_3 and f_5 in terms of some fixed ordering of the elements of U.

would lead to the formula

$$(x_2)(x_4)M(f_1, x_2, f_3(x_2), x_4).$$

3. *The Sequence of Quantifier-Free Lines*

The way our whole refutation-algorithm will "look" may now be indicated in a general way. Suppose the given formula is

$$(x_1)(Ex_2)(x_3)R(x_1, x_2, x_3),$$

where R is quantifier-free and contains only the indicated variables. Then the first step will be to replace the existential quantifier(s) by function symbols, which will lead in this case to

$$(x_1)(x_3)R(x_1, f(x_1), x_3)$$

(recall that "f" abbreviates f_2 and that "a" abbreviates f_1). Next we will form a sequence of *quantifier-free lines* as follow (certain parentheses are omitted for brevity):

$$\begin{array}{l} R(a, fa, a) \\ R(a, fa, fa) \\ R(fa, ffa, a) \\ R(fa, ffa, fa) \\ R(a, fa, ffa) \\ \vdots \end{array} \qquad \text{(Observe that the variables } x_1, x_3 \text{ are replaced in all possible ways with terms from the sequence } a, fa, ffa, \cdots .) \qquad (1)$$

As these quantifier-free lines are generated, we will test the conjunction of the first n lines (for $n = 1, 2, 3, \cdots$) for consistency (by methods described in the next section). If the conjunction of the first n lines is inconsistent, for any n, then the formula $(x_1)(x_3)R(x_1, f(x_1), x_3)$ is inconsistent (since it implies all of the quantifier free lines), and hence the given formula was inconsistent. On the other hand, if the conjunctive of the first n lines is consistent for every n, then the algorithm never terminates, and the given formula was consistent.[9]

We now state the general *rule for forming the sequence of quantifier-free lines.* Let F be the given formula after the existential quantifiers have been replaced by function symbols. Let $f_{i_1}, \cdots, f_{i_M}$ be all the function symbols in F, and let f_{i_k} be an n_k-place function symbol (for $k = 1, 2, \cdots, M$). Let D be the following set: the smallest set containing the individual constant a and having the property that whenever it contains $t_1, \cdots, t_{n_k}$ then it contains the expression $f_{i_k}(t_1, \cdots, t_{n_k})$, for $k = 1, 2, \cdots, M$. Let L be the number of *universal* quantifiers in F, and let S be the sequence of all ordered L-tuplets of members of D,

[9] For the proof of this statement see [7], pp. 253–260. The key point in the proof is that an infinite set of quantifier-free formulas is inconsistent if and only if some *finite* subset is inconsistent.

in lexicographic order.[10] Then the nth quantifier free line (for $n = 1, 2, 3, \cdots$) is the result of substituting[11] t_{n1} for the first universally quantified variable (in F), t_{n2} for the second universally quantified variable, $\cdots$, t_{nL} for the Lth universally quantified variable, where $t_{n1}; \cdots; t_{nL}$ is the nth L-tuplet in the sequence S.

REMARKS:

(A) One may, if one desires, abbreviate the expressions in the set D by numbers according to some convenient scheme. If one adopted this policy, the quantifier-free lines (1) above might look like this:

$$\begin{array}{c} R(1, 2, 1) \\ R(1, 2, 2) \\ R(2, 3, 1) \\ R(2, 3, 2) \\ R(1, 2, 3) \\ \vdots \end{array} \tag{2}$$

Such a scheme of numerical abbreviation is extremely worthwhile from the standpoint of *hand* computation (because it cuts down the length of the formulas). On the other hand, there may be little or no advantage to adopting such a scheme if the algorithm is going to be programmed for a computer.

(B) Instead of testing the conjunction of the first n quantifier-free lines for consistency when $n = 1, 2, 3, \cdots$, one might test "intermittently," e.g., when $n = 10, 20, 30, \cdots$. The relative advantages and disadvantages of such "intermittent" applications of the testing for consistency should be investigated if the algorithm we are describing is to be actually programmed for a computer.

4. *Feasible Methods in the Propositional Calculus*

The idea of a refutation-algorithm, of the sort described in general terms in the preceding section, is not new. In essence, it goes back to Herbrand[12], and formulations of the kind we have given (based on the idea of generating a sequence of quantifier-free lines, and then testing the conjunction of the first n lines for consistency as $n = 1, 2, 3, \cdots$) have been previously given by Quine[13], Gilmore[14], and others. However, the crucial difficulty, to which little attention ap-

[10] For the purposes of defining "lexicographic order," subscripts are to be thought of as if they were written on the line (e.g., $f_{12}(a)$ is to be treated as if it were "$f12(a)$"). Then our alphabet consists of the symbols: ( ) f 0 1 2 3 4 5 6 7 8 9 , ; (the latter symbol being used to separate the members of an L-tuplet thus: "$f2(f1);f6(f1,f2(f1))$"), and the "lexicographic ordering" of the L-tuplets is the ordering in which they are arranged like words in a dictionary.

[11] As indicated in the example, a universal quantifier is deleted whenever something is substituted for the variable it contains. This sort of "substitution" is technically known as *universal instantiation* (cf. [7], p. 147).

[12] Cf. [4].

[13] Cf. [8].

[14] Cf. [3].

pears to have been given in this connection, is that of finding a *feasible* technique for testing the conjunction of the first n lines for consistency when n is large. Quine's "uniform proof procedure" is described with hand computation in mind, and thus Quine limits himself to truth-tables as a method in the propositional calculus. However, the number of lines in a truth table, when k propositional variables are involved, is 2^k and so truth-tables quickly become unfeasible for our purposes. Gilmore's procedure is to put the conjunction of the first n lines into disjunctive normal form, but this too leads to exponentiation (on the number of clauses in the matrix of the given formula), and so this method too is unfeasible in general (although fortuitous cancellations may keep the formulas involved down to manageable length in special cases). Still another procedure has been proposed by Wang in [9]. Wang's procedure is less easy to compare with ours because it does not use prenex normal form; however his routine employs a "Gentzentype" formal system in which proofs have a "tree" structure[15] (as opposed to the usual "linear" structure) with "branching" possible at any line. As far as the propositional calculus is concerned, the difficulty with Wang's technique is that the number of branches tends to increase exponentially with the number of logical connectives involved. Thus, none of the three methods just described—truth-tables, disjunctive normal forms, or Gentzen-type systems—is satisfactory as a method for testing the conjunction of the first n lines (in our sequence of quantifier-free lines) for truth-functional consistency when n becomes at all large (e.g., $n > 10$).

By contrast, the method to be described always terminates in at most $2(R-1)$ steps, where R is the number of variables (i.e., the number of steps increases *linearly*, not exponentially, in the number of variables). Moreover, the process will rarely lead to formulas which are much more complicated than those with which one started in examples of the sort likely to arise in practice. Actually it has been found possible to work quite complicated formulas by this method even by *hand* computation.

The method to be described depends on putting the conjunction of the first n lines into *conjunctive* normal form. Since putting a formula into conjunctive normal form does not of itself enable one to tell whether or not the formula is consistent, it is necessary to make one or two remarks explaining our choice of this normal form. Briefly, the reasons are as follows: although normal forms may in certain cases be used as decision-methods (e.g., putting a formula into *disjunctive* normal form automatically reveals whether or not the formula is inconsistent[16]), they have also another function, as the term "normal form" indicates, namely, their use serves to regularize formulas and to cut down structural complexity. For instance, every formula F in conjunctive normal form has the structure $A \ \& \ B \ \& \ R$ where A is the conjunction of the clauses containing a given atomic formula (say, p), B is the conjunction of the clauses containing the negation of that formula (say, $\bar{p}$), and R is the conjunction of the remaining clauses. Moreover, it can be shown that F is inconsistent if and only if $A' \ \& \ R$

[15] For an explanation of "tree structure" cf. [6], pp. 106–107.

[16] Cf. [7], pp. 52–59.

and B' & R are both inconsistent, where A' is obtained from A by deleting occurrences of p, and B' is obtained from B by deleting occurrences of $\bar{p}$. Such regularities are hardly to be hoped for in the case of arbitrary formulas not in normal form.

Our problem, as indicated above, is how to deal with cases in which the number of quantifier-free lines is too large to make it feasible to put the whole system of lines into disjunctive normal form. In such cases there is one normal form we can use: namely, the *conjunctive normal form.*

That the conjunctive normal form can be employed follows from the remark that to put a whole system of formulas into conjunctive normal form we have only to put the individual formulas into conjunctive normal form. Thus, even if a system has hundreds or thousands of formulas, it can be put into conjunctive normal form "piece by piece", without any "multiplying out." This is a feasible (if laborious) task even for *hand* computation: thus no specialization is introduced here beyond supposing that the individual formulas in the system are "manageable" (i.e., short enough to be put into conjunctive normal form by hand) and that the whole system can be written down by a human being.

In the case of our "sequences of quantifier-free lines" (generated according to the rule in the preceding section), the situation is even more pleasant than in the general case of testing some "big" system of formulas for consistency: namely, it suffices to put the *matrix* of the given formula (after the existential quantifiers have been replaced by function symbols) into conjunctive normal form, and then the "quantifier-free lines" will be automatically generated in conjunctive normal form!

In stating our method for testing the conjunction of the first n "quantifier-free lines" for consistency, we shall assume that the matrix of the given formula was in conjunctive normal form (so that the conjunction of the first n lines will likewise automatically be in conjunctive normal form), and we shall speak of the entire conjunction as a single formula F.

Our method consists of the following three rules, in which p, q, r, s are atomic formulas:

I. *Rule for the Elimination of One-Literal Clauses:*

(a) If a formula F in conjunctive normal form contains an atomic formula p as a one-literal clause and also contains $\bar{p}$ as a one-literal clause, then F may be replaced by 0. (I.e., F is self-contradictory).

(b) If case (a) does not apply, and if an atomic formula p appears as a clause in a formula F in conjunctive normal form, then one may modify F by striking out all clauses that contain p affirmatively[17] and deleting all occurrences of $\bar{p}$ from the remaining clauses, thus obtaining a formula F' which is inconsistent if and only if F is.

(c) If case (a) does not apply and $\bar{p}$ appears as a clause in a formula F in conjunctive normal form, then one may modify F by striking out all clauses that con-

[17] An occurrence of p without a negation bar is called an *affirmative* occurrence; one with a negation bar is called a *negative* occurrence.

tain $\bar{p}$ and deleting all occurrences of p from the remaining clauses, again obtaining a formula F' which is inconsistent if and only if F is.

(d) In cases (b) and (c), if F' is empty, then F is consistent.

II. *Affirmative-Negative Rule.* If an atomic formula p occurs in a formula F in conjunctive normal form only affirmatively, or if p occurs only negatively, then all clauses which contain p may be deleted. The resulting formula F' is inconsistent if and only if F is. (If F' is empty, then F is consistent).

III. *Rule for Eliminating Atomic Formulas.* Let the given formula be put into the form $(A \vee p) \;\&\; (B \vee \bar{p}) \;\&\; R$ where A, B, and R are free of p. (This can be done simply by grouping together the clauses containing p and "factoring out" occurrences of p to obtain A, grouping the clauses containing $\bar{p}$ and "factoring out" $\bar{p}$ to obtain B, and grouping the remaining clauses to obtain R.) Then F is inconsistent if and only if $(A \vee B) \;\&\; R$ is inconsistent.

Justification. For Rule I: The justification of case (a) of the rule is obvious. For case (b), let the formula F be $p \;\&\; A$. Then F is clearly false when $p = 0$; hence F is inconsistent, provided F is false when $p = 1$. Substituting 1 for p in F and simplifying has the following effect: All clauses that contain p affirmatively reduce to 1 and may be deleted. All clauses that contain p negatively reduce to 0 (in case the whole clause was $\bar{p}$) or to $0 \vee B$, where B is the remainder of the clause. But there cannot be any clauses which consist of *just* $\bar{p}$ (otherwise case (a) would apply); and $0 \vee B = B$. Hence the effect of substituting 1 for p in F and simplifying is to strike out all the clauses that contain p affirmatively and delete all occurrences of $\bar{p}$ from the remaining clauses. Thus

$$\begin{aligned} F' \text{ is inconsistent} &\leftrightarrow F \text{ is false whenever } p = 1 \\ &\leftrightarrow F \text{ is inconsistent.} \end{aligned}$$

Case (c) is symmetrical to case (b). Case (d) reduces to the observation that if p occurs in every clause, then $F = 1$ when $p = 1$.

For Rule II: Let p occur in F only affirmatively, and let F be $A \;\&\; R$ where A is the conjunction of all the clauses containing p. Then if F is inconsistent, F is false when $p = 1$. But when $p = 1$ we have $A = 1$, and therefore $(A \;\&\; R) \leftrightarrow R$ when $p = 1$. Hence, if F is inconsistent, so is R. But, since $(A \;\&\; R) \rightarrow R$, if R is inconsistent, so is $(A \;\&\; R)$. (If R is empty, $F = 1$ when $p = 1$, and therefore F is consistent.) The argument is similar when p occurs only negatively, using $p = 0$ instead of $p = 1$.

For Rule III: F is inconsistent if and only if F is false when $p = 0$ and false when $p = 1$. But in the first case, F reduces to $(A \;\&\; R)$ and in the second case to $(B \;\&\; R)$. So F is inconsistent if and only if $(A \;\&\; R)$ and $(B \;\&\; R)$ are both inconsistent, and $(A \;\&\; R) \vee (B \;\&\; R) \leftrightarrow (A \vee B) \;\&\; R$.

Examples. (1) Consider the formula:

$$(p \vee q \vee \bar{r}) \;\&\; (p \vee \bar{q}) \;\&\; \bar{p} \;\&\; r$$

There are two one-literal clauses. Elimination of these leads immediately to $q \;\&\; \bar{q} = 0$.

(2) Consider the formula

$$(p \vee q) \mathbin{\&} \bar{q} \mathbin{\&} (\bar{p} \vee q \vee \bar{r}).$$

Elimination of the one-literal clause yields $p \mathbin{\&} (\bar{p} \vee \bar{r})$, which in turn yields $\bar{r}$. By Rule I or Rule II, this formula is consistent.

(3) The formula

$$(p \vee \bar{q}) \mathbin{\&} (\bar{p} \vee q) \mathbin{\&} (q \vee \bar{r}) \mathbin{\&} (\bar{q} \vee \bar{r})$$

contains r only negatively. By Rule II, it is inconsistent if and only if $(p \vee \bar{q}) \mathbin{\&} (\bar{p} \vee q)$ is. By Rule III (eliminating p), this is inconsistent if and only if $q \vee \bar{q}$ is. But $q \vee \bar{q} = 1$, so this is consistent.

(4) The following example is worked using only Rule III. (Note that it is necessary to put the formula back into conjunctive normal form after each elimination).

$(p \vee r) \mathbin{\&} (p \vee \bar{s}) \mathbin{\&} (\bar{p} \vee s) \mathbin{\&} (\bar{p} \vee \bar{r}) \mathbin{\&} (s \vee \bar{r}) \mathbin{\&} (\bar{s} \vee r)$
$[(r \mathbin{\&} \bar{s}) \vee p] \mathbin{\&} [(\bar{r} \mathbin{\&} s) \vee \bar{p}] \mathbin{\&} (s \vee \bar{r}) \mathbin{\&} (\bar{s} \vee r)$
$(s \vee r) \mathbin{\&} (\bar{s} \vee \bar{r}) \mathbin{\&} (s \vee \bar{r}) \mathbin{\&} (\bar{s} \vee r)$ (p eliminated)
$[(s \mathbin{\&} \bar{s}) \vee r] \mathbin{\&} [(s \mathbin{\&} \bar{s}) \vee \bar{r}]$
$s \mathbin{\&} \bar{s}$ (r eliminated)

To complete the refutation, it suffices to note that $s \mathbin{\&} \bar{s}$ is inconsistent by Rule I.

5. *The Complete Algorithm*

In the preceding sections we have stated the various rules which make up our refutation-algorithm. It remains to "put the pieces together." The following is the complete sequence of steps to be followed in employing the algorithm (we adopt the policy of alluding to rules which have been completely stated in earlier sections of this paper, rather than restating them in full; also we assume the given formula to be prenex, and to have a matrix in conjunctive normal form):

Step 1. Generate one more quantifier-free line (if none have previously been generated, this means: generate a first quantifier-free line). Then test the conjunction of all the so-far-generated quantifier-free lines for consistency by the following steps:

Step 2. Apply the rule for eliminating one-literal clauses (Rule I) to the conjunction obtained at step 1 if it contains any one-literal clauses, and continue applying this rule until the resulting formula has no one-literal clauses. If the empty formula results, the conjunction obtained at step 1 was consistent. If a formula results which is inconsistent by Rule I, the conjunction obtained at step 1 was inconsistent. If a nonempty formula with no one-literal clauses results, go on to—

Step 3. Apply the affirmative-negative rule (Rule II) to the formula obtained at step 2 (or to the conjunction obtained at step 1, if step 2 did not apply) unless that formula had the property that every atomic formula that occurred in it occurred both affirmatively and negatively. Then go back to step 2 if the result contains any one-literal clauses. Otherwise, repeat step 3 if the result contained some literal which occurred only affirmatively or only negatively. If the result is the empty formula, the conjunction obtained at step 1 was consistent. If a non-empty formula with no one-literal clauses and with the property that every atomic formula that occurs in it occurs both affirmatively and negatively results, go on to—

Step 4. Using Rule III, eliminate the first atomic formula from the first clause of minimal length in the formula that has resulted from the preceding steps (or from the conjunction obtained at step 1, if steps 2 and 3 did not apply). If the resulting formula cannot be put back into conjunctive normal form (because every clause would contain an atomic formula both negated and not-negated), the conjunction obtained at step 1 was consistent. Otherwise, put the resulting formula back into conjunctive normal form, and go back to step 2.

Continue in this way (i.e., going through the "cycle" steps 2–3–4) until either (a) it has been decided at some application of steps 2, 3, or 4 that the conjunction obtained at step 1 was consistent; or (b) it has been decided that the conjunction obtained at step 1 was inconsistent. (This can only happen at an application of step 2.)

If it is decided that the conjunction obtained at step 1 was inconsistent, then the algorithm terminates, and the given formula was inconsistent (i.e., "refutation" has been accomplished). If it is decided that the conjunction obtained at the preceding application of step 1 was consistent, go back to step 1, and continue.

6. *An Example*

P. C. Gilmore[18] tested his refutation-procedure on a number of formulas, including the following one:

$$(Ex)(Ey)(z)\{(F(x, y) \to (F(y, z) \ \& \ F(z, z))) \ \& \ ((F(x, y) \ \& \ G(x, y)) \to (G(x, z) \ \& \ G(z, z)))\} \tag{1}$$

We have selected this example for purposes of comparison because (a) it is not so long as to make hand computation immediately impractical (e.g., it is already in prenex form, and the matrix can easily be put into conjunctive normal form); yet (b) Gilmore's procedure did *not* lead to a refutation although an IBM 704 was employed for 21 minutes.

Our procedure, on the other hand, *did* lead to a refutation in under a half-hour of *hand* computation! For the purposes of hand computation, one modification was made in the algorithm: instead of testing the conjunction of the first n-lines for consistency when $n = 1, 2, 3, \cdots$, we adopted the scheme of "intermittent" testing alluded to at the end of section 3, and tested at $n = 10, 20, 30$. The conjunction of the first n lines was *consistent* when $n = 10$ and $n = 20$ and inconsistent when $n = 30$. Inspection later revealed that the smallest n for which the conjunction of the first n lines was inconsistent was $n = 25$. That the difficulty

[18] Cf. [3].

with Gilmore's procedure lies in the propositional calculus method employed is confirmed by the fact that in the 21 minutes the IBM 704 was running, only 7 "substitutions" were made; only what amounts to 7 quantifier-free lines were generated. *We adopt the abbreviation, here and below, of omitting the symbol* $\vee$, writing, e.g.,

$$\bar{F}(y, z)\bar{F}(z, z)G(x, y) \quad \text{for} \quad (\bar{F}(y, z) \vee \bar{F}(z, z) \vee G(x, y)).$$

The following is the negation of formula (1) with matrix in conjunctive normal form:

$$(x)(y)(Ez)(F(x, y) \ \& \ \bar{F}(y, z)\bar{F}(z, z)G(x, y) \\ \& \ \bar{F}(y, z)\bar{F}(z, z)\bar{G}(x, z)\bar{G}(z, z)) \tag{2}$$

Replacing the existential quantifier by a function symbol gives:

$$(x)(y)[F(x, y) \ \& \ \bar{F}(y, f(x, y))\bar{F}(f(x, y), f(x, y))G(x, y) \\ \& \ \bar{F}(y, f(x, y))\bar{F}(f(x, y), f(x, y))\bar{G}(x, f(x, y))\bar{G}(f(x, y), f(x, y))]. \tag{3}$$

In writing the first 25 quantifier-free lines generated we have used numbers up to 25 instead of "$f(a, a)$", "$f(f(a, a), a)$", etc, in order to make the formulas shorter and the over-all pattern more clear. Also we have omitted parentheses between predicate symbols and their arguments. The lines are as follows:

Quantifier-Free Lines:

1.	Fa, a & $\bar{F}a, 1$	$\bar{F}1, 1$	Ga, a & $\bar{F}a, 1$	$\bar{F}1, 1$	$\bar{G}a, 1$	$\bar{G}1, 1$
2.	$F1, a$ & $\bar{F}a, 2$	$\bar{F}2, 2$	$G1, a$ & $\bar{F}a, 2$	$\bar{F}2, 2$	$\bar{G}1, 2$	$\bar{G}2, 2$
3.	$F1, 1$ & $\bar{F}1, 3$	$\bar{F}3, 3$	$G1, 1$ & $\bar{F}1, 3$	$\bar{F}3, 3$	$\bar{G}1, 3$	$\bar{G}3, 3$
4.	$Fa, 1$ & $\bar{F}1, 4$	$\bar{F}4, 4$	$Ga, 1$ & $\bar{F}1, 4$	$\bar{F}4, 4$	$\bar{G}a, 4$	$\bar{G}4, 4$
5.	$Fa, 2$ & $\bar{F}2, 5$	$\bar{F}5, 5$	$Ga, 2$ & $\bar{F}2, 5$	$\bar{F}5, 5$	$\bar{G}a, 5$	$\bar{G}5, 5$
6.	$Fa, 3$ & $\bar{F}3, 6$	$\bar{F}6, 6$	$Ga, 3$ & $\bar{F}3, 6$	$\bar{F}6, 6$	$\bar{G}a, 6$	$\bar{G}6, 6$
7.	$Fa, 4$ & $\bar{F}4, 7$	$\bar{F}7, 7$	$Ga, 4$ & $\bar{F}4, 7$	$\bar{F}7, 7$	$\bar{G}a, 7$	$\bar{G}7, 7$
8.	$F1, 2$ & $\bar{F}2, 8$	$\bar{F}8, 8$	$G1, 2$ & $\bar{F}2, 8$	$\bar{F}8, 8$	$\bar{G}1, 8$	$\bar{G}8, 8$
9.	$F1, 3$ & $\bar{F}3, 9$	$\bar{F}9, 9$	$G1, 3$ & $\bar{F}3, 9$	$\bar{F}9, 9$	$\bar{G}1, 9$	$\bar{G}9, 9$
10.	$F1, 4$ & $\bar{F}4, 10$	$\bar{F}10, 10$	$G1, 4$ & $\bar{F}4, 10$	$\bar{F}10, 10$	$\bar{G}1, 10$	$\bar{G}10, 10$
11.	$F2, a$ & $\bar{F}a, 11$	$\bar{F}11, 11$	$G2, a$ & $\bar{F}a, 11$	$\bar{F}11, 11$	$\bar{G}2, 11$	$\bar{G}11, 11$
12.	$F2, 1$ & $\bar{F}1, 12$	$\bar{F}12, 12$	$G2, 1$ & $\bar{F}1, 12$	$\bar{F}12, 12$	$\bar{G}2, 12$	$\bar{G}12, 12$
13.	$F2, 2$ & $\bar{F}2, 13$	$\bar{F}13, 13$	$G2, 2$ & $\bar{F}2, 13$	$\bar{F}13, 13$	$\bar{G}2, 13$	$\bar{G}13, 13$
14.	$F2, 3$ & $\bar{F}3, 14$	$\bar{F}14, 14$	$G2, 3$ & $\bar{F}3, 14$	$\bar{F}14, 14$	$\bar{G}2, 14$	$\bar{G}14, 14$
15.	$F2, 4$ & $\bar{F}4, 15$	$\bar{F}15, 15$	$G2, 4$ & $\bar{F}4, 15$	$\bar{F}15, 15$	$\bar{G}2, 15$	$\bar{G}15, 15$
16.	$F3, a$ & $\bar{F}a, 16$	$\bar{F}16, 16$	$G3, a$ & $\bar{F}a, 16$	$\bar{F}16, 16$	$\bar{G}3, 16$	$\bar{G}16, 16$
17.	$F3, 1$ & $\bar{F}1, 17$	$\bar{F}17, 17$	$G3, 1$ & $\bar{F}1, 17$	$\bar{F}17, 17$	$\bar{G}3, 17$	$\bar{G}17, 17$
18.	$F3, 2$ & $\bar{F}2, 18$	$\bar{F}18, 18$	$G3, 2$ & $\bar{F}2, 18$	$\bar{F}18, 18$	$\bar{G}3, 18$	$\bar{G}18, 18$
19.	$F3, 3$ & $\bar{F}3, 19$	$\bar{F}19, 19$	$G3, 3$ & $\bar{F}3, 19$	$\bar{F}19, 19$	$\bar{G}3, 19$	$\bar{G}19, 19$
20.	$F3, 4$ & $\bar{F}4, 20$	$\bar{F}20, 20$	$G3, 4$ & $\bar{F}4, 20$	$\bar{F}20, 20$	$\bar{G}3, 20$	$\bar{G}20, 20$
21.	$F4, a$ & $\bar{F}a, 21$	$\bar{F}21, 21$	$G4, a$ & $\bar{F}a, 21$	$\bar{F}21, 21$	$\bar{G}4, 21$	$\bar{G}21, 21$
22.	$F4, 1$ & $\bar{F}1, 22$	$\bar{F}22, 22$	$G4, 1$ & $\bar{F}1, 22$	$\bar{F}22, 22$	$\bar{G}4, 22$	$\bar{G}22, 22$
23.	$F4, 2$ & $\bar{F}2, 23$	$\bar{F}23, 23$	$G4, 2$ & $\bar{F}2, 23$	$\bar{F}23, 23$	$\bar{G}4, 23$	$\bar{G}23, 23$
24.	$F4, 3$ & $\bar{F}3, 24$	$\bar{F}24, 24$	$G4, 3$ & $\bar{F}3, 24$	$\bar{F}24, 24$	$\bar{G}4, 24$	$\bar{G}24, 24$
25.	$F4, 4$ & $\bar{F}4, 25$	$\bar{F}25, 25$	$G4, 4$ & $\bar{F}4, 25$	$\bar{F}25, 25$	$\bar{G}4, 25$	$\bar{G}25, 25$

Applying our "one-literal clause rule," we obtain:

Ga, a & $\bar{G}a, 1$ $\bar{G}1, 1$ &
$G1, a$ & $\bar{G}1, 2$ $\bar{G}2, 2$ &
$G1, 1$ & $\bar{G}1, 3$ $\bar{G}3, 3$ &
$Ga, 1$ & $\bar{G}a, 4$ $\bar{G}4, 4$ &
$\bar{F}2, 5$ $\bar{F}5, 5$ $Ga, 2$ & $\bar{F}2, 5$ $\bar{F}5, 5$ $\bar{G}a, 5$ $\bar{G}5, 5$ &
$\bar{F}3, 6$ $\bar{F}6, 6$ $Ga, 3$ & $\bar{F}3, 6$ $\bar{F}6, 6$ $\bar{G}a, 6$ $\bar{G}6, 6$ &
$\bar{F}4, 7$ $\bar{F}7, 7$ $Ga, 4$ & $\bar{F}4, 7$ $\bar{F}7, 7$ $\bar{G}a, 7$ $\bar{G}7, 7$ &
$\bar{F}2, 8$ $\bar{F}8, 8$ $G1, 2$ & $\bar{F}2, 8$ $\bar{F}8, 8$ $\bar{G}1, 8$ $\bar{G}8, 8$ &
$\bar{F}3, 9$ $\bar{F}9, 9$ $G1, 3$ & $\bar{F}3, 9$ $\bar{F}9, 9$ $\bar{G}1, 9$ $\bar{G}9, 9$ &
$\bar{F}4, 10$ $\bar{F}10, 10$ $G1, 4$ & $\bar{F}4, 10$ $\bar{F}10, 10$ $\bar{G}1, 10$ $\bar{G}10, 10$ &
$\bar{F}a, 11$ $\bar{F}11, 11$ $G2, a$ & $\bar{F}a, 11$ $\bar{F}11, 11$ $\bar{G}2, 11$ $\bar{G}11, 11$ &
$\bar{F}1, 12$ $\bar{F}12, 12$ $G2, 1$ & $\bar{F}1, 12$ $\bar{F}12, 12$ $\bar{G}2, 12$ $\bar{G}12, 12$ &
$\bar{F}2, 13$ $\bar{F}13, 13$ $G2, 2$ & $\bar{F}2, 13$ $\bar{F}13, 13$ $\bar{G}2, 13$ $\bar{G}13, 13$ &
$\bar{F}3, 14$ $\bar{F}14, 14$ $G2, 3$ & $\bar{F}3, 14$ $\bar{F}14, 14$ $\bar{G}2, 14$ $\bar{G}14, 14$ &
$\bar{F}4, 15$ $\bar{F}15, 15$ $G2, 4$ & $\bar{F}4, 15$ $\bar{F}15, 15$ $\bar{G}2, 15$ $\bar{G}15, 15$ &
$\bar{F}a, 16$ $\bar{F}16, 16$ $G3, a$ & $\bar{F}a, 16$ $\bar{F}16, 16$ $\bar{G}3, 16$ $\bar{G}16, 16$ &
$\bar{F}1, 17$ $\bar{F}17, 17$ $G3, 1$ & $\bar{F}1, 17$ $\bar{F}17, 17$ $\bar{G}3, 17$ $\bar{G}17, 17$ &
$\bar{F}2, 18$ $\bar{F}18, 18$ $G3, 2$ & $\bar{F}2, 18$ $\bar{F}18, 18$ $\bar{G}3, 18$ $\bar{G}18, 18$ &
$\bar{F}3, 19$ $\bar{F}19, 19$ $G3, 3$ & $\bar{F}3, 19$ $\bar{F}19, 19$ $\bar{G}3, 19$ $\bar{G}19, 19$ &
$\bar{F}4, 20$ $\bar{F}20, 20$ $G3, 4$ & $\bar{F}4, 20$ $\bar{F}20, 20$ $\bar{G}3, 20$ $\bar{G}20, 20$ &
$\bar{F}a, 21$ $\bar{F}21, 21$ $G4, a$ & $\bar{F}a, 21$ $\bar{F}21, 21$ $\bar{G}4, 21$ $\bar{G}21, 21$ &
$\bar{F}1, 22$ $\bar{F}22, 22$ $G4, 1$ & $\bar{F}1, 22$ $\bar{F}22, 22$ $\bar{G}4, 22$ $\bar{G}22, 22$ &
$\bar{F}2, 23$ $\bar{F}23, 23$ $G4, 2$ & $\bar{F}2, 23$ $\bar{F}23, 23$ $\bar{G}4, 23$ $\bar{G}23, 23$ &
$\bar{F}3, 24$ $\bar{F}24, 24$ $G4, 3$ & $\bar{F}3, 24$ $\bar{F}24, 24$ $\bar{G}4, 24$ $\bar{G}24, 24$ &
$\bar{F}4, 25$ $\bar{F}25, 25$ $G4, 4$ & $\bar{F}4, 25$ $\bar{F}25, 25$ $\bar{G}4, 25$ $\bar{G}25, 25$.

Now applying the one-literal clause rule again to eliminate Ga, a, $G1, a$, and $G1, 1$ yields a formula containing $Ga, 1$ and $\bar{G}a, 1$ as clauses, which is inconsistent by Rule I.

The reader may be interested to see how the method works when the conjunction of quantifier-free lines being tested is *not* truth-functionally inconsistent. To illustrate this, let us test the conjunction of the first 10 quantifier-free lines listed above for consistency. Applying the one-literal clause rule yields:

1. [Ga, a &] $\bar{G}a, 1$ $\bar{G}1, 1$
2. $\bar{F}2, 2$ $G1, a$ & $\bar{F}2, 2$ $\bar{G}1, 2$ $\bar{G}3, 3$
3. $\bar{F}3, 3$ $G1, 1$ & $\bar{F}3, 3$ $\bar{G}1, 3$ $\bar{G}4, 4$
4. $\bar{F}4, 4$ $Ga, 1$ & $\bar{F}4, 4$ $\bar{G}a, 4$ $\bar{G}5, 5$

6.–10. } Same as in above list of "quantifier-free lines" except with first clause omitted.

A second application of the one-literal clause rule deletes the clause "Ga, a" (which was bracketted above in anticipation of this deletion). Now all the clauses containing an atomic formula beginning "$\bar{F}$" can be deleted by the affirmative-negative rule, and we obtain $\bar{G}a, 1 \vee \bar{G}1, 1$, which reduces to the empty formula by one more application of the affirmative-negative rule. Thus the conjunction of the first 10 quantifier-free lines was *consistent*. A similar result is obtained on testing the result of the first 20 quantifier-free lines.

NOTE ADDED IN PROOF: The "affirmative-negative rule" has also been employed, independently of our work, for testing propositional-calculus formulas by B. Dunham, R. Fridshal, and G. L. Sward: "A nonheuristic program for proving elementary logical theorems," *Proceedings of the First International Conference on Information Processing, Paris, 1959.*

To the list of reports of working proof procedure programs should be added: Dag Prawitz, Hakan Prawitz, and Neri Vogera, "A mechanical proof procedure and its realization in an electronic computer," *J. Assoc. Comput. Mach.*, *7* (1960), 102–128.

REFERENCES

1. MARTIN DAVIS, *Computability and Unsolvability*, New York, Toronto, and London, McGraw-Hill, 1958, xxv + 210 pp.
2. MARTIN DAVIS AND HILARY PUTNAM, A finitely axiomatizable system for elementary number theory. Submitted to the *Journal of Symbolic Logic*.
3. PAUL C. GILMORE, A proof method for quantification theory. *IBM J. Research Dev.* 4 (1960), 28–35.
4. JACQUES HERBRAND, *Recherches sur la theorie de la demonstration*. Travaux de la Societe des Sciences et des Lettres de Varsovie, Classe III science mathematiques et physiques, no. 33, 128 pp.
5. DAVID HILBERT AND WILHELM ACKERMANN, *Principles of Mathematical Logic*. New York, Chelsea, 1950, xii + 172 pp.
6. STEPHEN C. KLEENE, *Introduction to Metamathematics*. New York and Toronto, D. Van Nostrand, 1952, x + 550 pp.
7. WILLARD V. O. QUINE, *Methods of Logic*. New York, Henry Holt, revised 1959, xx + 272 pp.
8. WILLARD V. O. QUINE, A proof procedure for quantification theory. *J. Symbolic Logic* *20* (1955), 141–149.
9. HAO WANG, Towards mechanical mathematics. *IBM J. Research Dev.* 4 (1960) 2–22.