P. C. Gilmore

# A Proof Method for Quantification Theory: Its Justification and Realization

**Abstract: A program is described which can provide a computer with quick logical facility for syllogisms and moderately more complicated sentences. The program realizes a method for proving that a sentence of quantification theory is logically true. The program, furthermore, provides a decision procedure over a subclass of the sentences of quantification theory. The subclass of sentences for which the program provides a decision procedure includes all syllogisms. Full justification of the method is given.**

**A program for the IBM 704 Data Processing Machine is outlined which realizes the method. Production runs of the program indicate that for a class of moderately complicated sentences the program can produce proofs in intervals ranging up to two minutes.**

## Introduction

Before a rigorous mathematical proof of any theorem can be given, it is necessary that the theorem, and the axioms from which the theorem is to be deduced, be precisely stated in an unambiguous language. The formal language variously called the first-order predicate calculus or logic, the first-order functional calculus or logic, or quantification theory, is adequate for the expression of any theorem or axiom. Further, although the sentences of quantification theory are ambiguous in the sense of being capable of possessing many different meanings, it is an easy matter to remove this ambiguity simultaneously from all sentences of the language by assigning meaning to the primitive symbols from which the sentences are formed. But most important of all, it is possible to give for sentences $T, S_1, S_2, \ldots$ of quantification theory a precise definition of "$T$ is a logical consequence of the sentences $S_1, S_2, \ldots$" to replace the intuitive but vague notion in natural languages that one sentence is a logical consequence of other sentences. Thus quantification theory not only provides a language for mathematics but also permits a description of what constitutes a rigorous mathematical proof.

The primitive symbols of quantification theory are brackets (,), the logical connectives $-$, &, v, $\supset$, $\equiv$, the predicate letters $A, B, C, \ldots$ the individual names, $a, b, c, \ldots$ the individual variables $x, y, z, \ldots$ and the quantifiers $E$ and $A$. Any finite sequence of these primitive symbols is a formula of the theory. From the formulae of the theory are selected some which in some sense are meaningful and are therefore called *well-formed formulae,* or briefly wff. From the wff of the theory will be selected the sentences of the theory.

Inductive definitions of wff and of "the (individual) variable $X$ occurs free in the wff $S$" are given simultaneously:

1. A formula consisting of a predicate letter followed by any number of individual names or variables is an *atomic* wff and a wff and the variables occurring in it occur free.

2. If $S$ is a wff then so is $-S$, and the variables occurring free in $S$ occur free in $-S$.

3. If $S$ and $T$ are wff then so are $(S \,\&\, T)$, $(S \vee T)$, $(S \supset T)$ and $(S \equiv T)$, and the variables occurring free in either $S$ or $T$ occur free in these wff.

4. If $S$ is a wff and $X$ is any variable then $(EX)S$ and $(AX)S$ are wff and the variables other than $X$ occurring free in $S$ occur free in these wff.

A variable $X$ which occurs in a wff but does not occur free is said to be bound in the wff and must, therefore, by part (4) of the definition, occur with quantifier symbols as in $(EX)$ or $(AX)$. A *sentence* of the theory is a wff in which no variable occurs free. Calling such formulae of the theory sentences is reasonable if the primitive symbols of the theory are interpreted as follows:

Atomic sentences (atomic wff which are sentences) can be understood as abbreviations for sentences such as "$1+3=5$", "$3<7$", or "John is tall"; these could be ex-

pressed in quantification theory as $A135$, $B37$, or $Ca$, allowing numerals also to be names in quantification theory. Atomic wff which are not sentences can be understood as abbreviations for sentence forms such as "$1+x=5$", or "$x+y=5$", or "$x+y=z$", or "$3<y$" or "$x<y$" or "$x$ is tall", where if variables in a sentence form are replaced by names the result is a sentence. The logical connectives $-$, &, ∨, ⊃ and ≡ are to be understood as expressing negation, conjunction, disjunction, implication and equivalence respectively in the sense that, for example, $(S \& T)$ is a sentence which is true if and only if both $S$ and $T$ are true, and $(S \supset T)$ is a sentence which is true if and only if $S$ is false or $T$ is true, and $-S$ is a sentence which is true if and only if $S$ is false. When the logical connectives are attached to sentence forms rather than sentences the result is to be understood as a new sentence form with the expected properties. For example, if $S(x)$ is a sentence form with one free variable $x$, then $-S(x)$ is a sentence form with one free variable $x$ and such that for any name $\alpha$, the sentence $-S(\alpha)$ formed from $-S(x)$ by replacing $x$ everywhere by $\alpha$ is true if and only if $S(\alpha)$ is false. Finally a sentence such as $(Ex)S(x)$ can be understood to be a sentence which is true if and only if for some name $\alpha$, $S(\alpha)$ is true, while a sentence $(Ax)S(x)$ can be understood to be a sentence which is true if and only if for every possible name $\alpha$, $S(\alpha)$ is true. For sentence forms $S(x, y)$ with two free variables $x$ and $y$, $(Ey)S(x, y)$ and $(Ay)S(x, y)$ are sentence forms with one free variable $x$ and are related in the obvious manner to the sentence form $S(x, y)$.

A sentence is generally thought to be either true or false. But a sentence of quantification theory may be true or false depending upon the interpretations given to the predicate letters and names which occur in it. Thus "$(Ey)(Bay \& Cy)$" is true when "$Bay$" is understood as "$a$ is the brother of $y$", "$Cy$" is understood as "$y$ is tall" and "$a$" is the name of someone with a tall brother, but can be false under other interpretations.

If atomic sentences or sentences formed from atomic sentences by attaching a negation sign "$-$" are called *ground sentences*, then by an interpretation can be understood a set $I$ of ground sentences of quantification theory with the following properties:

1. No atomic sentence and its negation are members of $I$ ($I$ is *consistent*);

2. For every possible atomic sentence which can be formed from names and predicate letters occurring in ground sentences in $I$, either the sentence or its negation is a member of $I$ ($I$ is *complete*).

A predicate letter or name occurring in a member of an interpretation $I$ is said to be a predicate letter or name of $I$.

The members of an interpretation $I$ are the only ground sentences of the language which are true for the interpretation. A ground sentence not in $I$, but formed from a predicate letter and names of $I$, is false for $I$. Hence a sentence in which occurs a predicate letter or name not of an interpretation is neither true nor false for the interpretation. For sentences which are not ground sentences and in which occur only predicate letters and names of $I$, "true for the interpretation" and "false for the interpretation" can be readily defined considering the meanings which have been given to the logical connectives and quantifiers. Thus $(Ey)(Bay \& Cy)$ is true in the interpretation $\{Ca, Cb, Bab, -Bba, -Baa, -Bbb\}$ because $Bab$ and $Cb$ are true in the interpretation and therefore also $(Bab \& Cb)$ and hence the above sentence. But the same sentence is false in the interpretation $\{Ca, -Cb, Bab, -Bba, -Baa, -Bbb\}$ because there is no name $\alpha$ such that both $Ba\alpha$ and $C\alpha$ are true in the interpretation, and hence no name $\alpha$ for which $(Ba\alpha \& C\alpha)$ is true for the interpretation. Clearly any consistent set $I$ of ground sentences can be completed to an interpretation by adding to $I$, for any atomic sentence formed from a predicate letter and names of $I$, either the atomic sentence or its negation, should neither already appear in $I$. Such an interpretation will be called a *completion* of $I$.

This definition of interpretation will appear a little strange because it does not require the specification of any set of objects which is to be the range of the variables $x, y, z, \ldots$, to which the names $a, b, c \ldots$ are assigned or the properties and relations of which are to be assigned to the predicate letters $A, B, C, \ldots$. But since it is possible for every object of a set to be assigned a single name, it is irrelevant whether one uses names to discuss the objects or simply discusses the names. However, this has as a consequence that two apparently quite different interpretations may be, for all practical purposes, the same interpretation. An object given a single name in one interpretation may be given a different name, or many different names in another interpretation. For this reason the notion of a homomorphism for interpretations is convenient.

Let $\psi$ be a single valued mapping of the names of an interpretation $I_1$ onto the names of an interpretation $I_2$, and for any sentence $S$ in which only names of $I_1$ occur, let $\psi(S)$ be the sentence obtained from $S$ by replacing each name $\alpha$ occurring in $S$ by $\psi(\alpha)$. If for any ground sentence $G$ in which only names of $I_1$ occur, $G$ is a member of $I_1$ if and only if $\psi(G)$ is a member of $I_2$, $\psi$ is said to be a homomorphism of $I_1$ onto $I_2$. If there exists a homomorphism of $I_1$ onto $I_2$ then $I_2$ is said to be a homomorphic image of $I_1$. It can be readily seen that if $\psi$ is a homomorphism of $I_1$ onto $I_2$ then for any sentence $S$, $S$ is true or false for $I_1$ if and only if $\psi(S)$ is true or false respectively for $I_2$.

A counterexample for a conjectured theorem $T$ in a mathematical theory with axioms $S_1, S_2, \ldots$ consists in a mathematical structure for which $S_1, S_2, \ldots$ are all true and for which $T$ is false. Assuming that $T, S_1, S_2, \ldots$ are all sentences of quantification theory, this can be more precisely expressed: The counterexample consists in an interpretation for which $S_1, S_2, \ldots$ are all true and $T$ is false. Thus, a conjectured theorem is actually a theorem if it is impossible to find a counterexample for it. Hence one can say that a sentence $T$ is a *logical consequence* of sentences $S_1, S_2, \ldots$ if and only if it is not possible to find an interpretation for which $S_1, S_2, \ldots$ are all true and $T$ is false. If $T$ is a logical consequence of an empty set of

sentences, that is, if it is impossible to find an interpretation in which $T$ is false, then $T$ is *logically* true.

A program for the IBM 704 Data Processing Machine will be described which will, for any logically true sentence $T$, construct a proof that $T$ is logically true. For a special class $M$ of sentences the program will do more; for any member $T$ of $M$ the program can decide whether or not $T$ is logically true, produce a proof of $T$ should it be logically true and provide essentially a counterexample to $T$ should $T$ be not logically true. The class $M$ of sentences is sufficiently wide to include, for example, all syllogisms. Results of production runs will be given. In addition it will be shown how the program can be adapted for producing a proof that a sentence $T$ is a logical consequence of sentences $S_1, S_2, \ldots$ .

The theoretical basis of the program is a process which, for any sentence $S$ of quantification theory, generates at the $n^{\text{th}}$ step of the process a finite number $k_n$, $k_n \geq 0$, of finite sets $I_{n1}, I_{n2}, \ldots, I_{nk_n}$ of ground sentences. Each set formed at the $n+1^{\text{st}}$ step is either one of the sets formed at the $n^{\text{th}}$ step or is obtained from such a set by adding new members to it. Hence if $k_n > 0$ for all $n$ then it is possible to find a function $\phi$ such that $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$ for all $n$. It is then the case that any completion of the set $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$ is an interpretation for which $S$ is true. Conversely if there is any interpretation whatsoever for which $S$ is true, then there is also one which is a completion of $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$ for some $\phi$. It follows that there is an interpretation for which $S$ is true if and only if $k_n > 0$ for all $n$.

The process can be applied to proving that a sentence $T$ is logically true by taking as input to the process the sentence $-T$. Should $k_n = 0$ for some $n$, then $T$ is true for every interpretation, and therefore logically true, since it is not possible for there to be an interpretation for which $-T$ is true and thus for which $T$ is false. On the other hand, however, the process cannot, in general, show that a sentence $T$ is not logically true; that is, that $-T$ has an interpretation, since it would be necessary to show that $k_n > 0$, for all $n$, while only a finite number of $k_n$ can ever be computed.[1] Nevertheless, for one special class $M$ of sentences of quantification theory the process does actually provide a decision procedure; that is, for any member $T$ of $M$, if $-T$ is used as input for the process, it will be possible to decide after a finite number of steps whether or not $T$ is logically true. For, for any member $T$ of $M$, it is possible to compute a number $N$ such that if $-T$ is used as input to the process then there is an interpretation for which $-T$ is true if and only if $k_n > 0$ for all $n$, $n \leqslant N$. Further if $\phi$ is a function such that $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$ for $n \leqslant N$, then any completion of $\bigcup_{n=1}^{N} I_{n\phi(n)}$ is an interpretation for which $-T$ is true. Thus for $T$ in $M$, if the process is carried out with $-T$ as input and $k_n = 0$ at the $n^{\text{th}}$ step, $n \leqslant N$, then $T$ has been proven to be logically true; while if $k_n > 0$ for each of the first $N$ steps of the process, $T$ is known to be not logically true and a counterexample to $T$ can actually be exhibited.

In the description of the process some well-known concepts and theorems from logic will be used. Two sentences are said to be *logically equivalent* if and only if there exists no interpretation for which one is true and the other is false. A sentence in which no quantifiers appear, and which is in disjunctive normal form, is called a *matrix*. Since a matrix consists of one or more disjunctions of conjunctions of one or more ground sentences, from two matrices $M_1$ and $M_2$ can be formed a product matrix as follows: For each conjunction of $M_1$ and each conjunction of $M_2$, form a conjunction of the product matrix by conjoining the two conjunctions. Contradictory conjunctions, i.e., ones in which both an atomic sentence and its negation appear, can be dropped from the product matrix.

A wff which is not a sentence but is in disjunctive normal form and does not contain any quantifiers is called a *matrix form*. A sentence which is in prenex form, that is, one in which all of the quantifiers occur initially, and which consists of a sequence of quantifiers attached to a wff which is a matrix form, will be said to be in *standard form*. For any sentence of quantification theory, there exists a sentence in standard form which is logically equivalent to the given sentence. A sentence which is in standard form and which is logically equivalent to a given sentence will be said to be a standard form of the given sentence.

The original motivation for the proof method of this paper was the method of semantic tableaux of Beth, although in its present form it is closer to the work of Hintikka.[2] The proof method is related in form, if not entirely in motivation, to the methods of Herbrand and Gentzen.[3]

Although much previous work has been done in proving theorems by machine, the present work is the first working program for quantification theory. The work of Newell, Shaw and Simon did not have as its primary aim the proof of theorems, but did result in a program for the propositional logic.[4]

Gelernter's and Rochester's program[5] for proving theorems in elementary geometry is applied to proving theorems which can be put into the standard form:

$$(AX_1) \ldots (AX_n)(P(X_1, \ldots, X_n) \supset Q(X_1, \ldots, X_n)),$$

where $P(X_1, \ldots, X_n)$ and $Q(X_1, \ldots, X_n)$ are conjunctions of atomic wff, from axioms which can be put into the same standard form. Although there exists a simple decision procedure for theorems in such an axiomatic theory, the Gelernter-Rochester program does not make use of it. Their program, with motivations similar to those of Newell and Simon, instead of exhaustively generating substitutions for the axioms, chooses substitutions which are expected to lead more directly to a proof of the theorem. Dunham, Fridshal and Sward have developed a program which uses an efficient decision procedure for theorems in propositional logic.[6] The work of Wang is most directly related to the present paper.[7] Wang has written two programs and has proposed a third. His first program is a decision procedure for theorems of the

propositional logic, his second a decision procedure for theorems of quantification theory which are members of the Class $M$, while the proposed program is a proof method for theorems of quantification theory which incorporates the decision procedure of the first two programs. Thus the intention of Wang's third program is exactly the same as the program outlined in this paper; the two programs differ, however, in several ways. For example, it is based on a method of proof more directly related to that of Gentzen and Herbrand rather than that of Beth and Hintikka. Also, at the same time, Wang's program is to accomplish more, for it is to accept as input data the sentence to be proven, while the input data for the program of this paper is a standard form of the negation of the sentence to be proven. A further comparison of the two programs will have to wait upon production runs from Wang's program.[8]

The process which has been described as the theoretical basis of the program of this paper is given in detail in the next section. The following section describes the program for the IBM 704 Data Processing Machine for proving sentences to be logically true, and its extension to a program for proving that a given sentence is a logical consequence of other sentences. Results of production runs are given in the subsequent section.

## The process

Given any sentence, choose a standard form $S$ for it. Then $S$ can be written:

$$(Q_1X_1)(Q_2X_2)\ldots(Q_mX_m)M(X_1,\ldots,X_m),$$

where $X_1, X_2, \ldots, X_n$ are all the variables which occur free in the matrix form $M(X_1,\ldots,X_m)$ and each $Q_j$ is either $E$ or $A$. If $Q_j$ is $E(A)$, then $X_j$ is said to be existentially (universally) quantified.

The matrix form $M(X_1,\ldots,X_m)$ consists of the disjunction of one or more conjunctions of one or more basic sentence forms or basic sentences. Two conjunctions are said to be linked by given variables if there exists a finite sequence of conjunctions of the matrix form, beginning with one of the conjunctions and ending with the other, such that for each adjoining pair of conjunctions in the sequence, one of the given variables occurs free in both members of the pair. An existentially quantified variable $X_j$ is said to depend upon all universally quantified variables $X_i$ for which $i<j$ and which occur free in conjunctions where $X_j$ occurs free, or which are linked to the conjunctions in which $X_j$ occurs free by the universally quantified variables $X_k$ for which $j<k$.

Let there be $r$ universally quantified variables $X_{u_1}, X_{u_2}, \ldots, X_{u_r}$. Then a sequence of $r$-tuples $(p_{n1}, p_{n2}, \ldots, p_{nr})$, $n=1, 2, \ldots$, of positive non-zero integers is generated as follows:

(1) For all $j$, $p_{1j}=1$;

(2) Let $\mu_n$ be the maximum of $p_{n1}, p_{n2}, \ldots, p_{nr}$, then (**a**) if for all $j$, $1\leq j\leq r$, $p_{nj}=\mu_n$, then $p_{n+1r}=\mu_n+1$ and $p_{n+1j}=1$ for $j<r$; (**b**) if there is a $k$ such that $p_{nj}=\mu_n$ for $k<j\leq r$ and $p_{nk}=\mu_n-1$, then $p_{n+1j}=p_{nj}$ for $1\leq j<k$, $p_{n+1k}=\mu_n$, and $p_{n+1j}=1$ for $k<j\leq r$; (**c**) if $p_{nr}<\mu_n$, then $p_{n+1r}=p_{nr}+1$ and $p_{n+1j}=p_{nj}$ for $1\leq j<r$; (**d**) if there is a $k$ such that $p_{nj}=\mu_n$ for $k<j\leq r$ and $p_{nk}<\mu_n-1$, then $p_{n+1j}=p_{nj}$ for $1\leq j<k$, $p_{n+1k}=p_{nk}+1$, $p_{n+1,r}=\mu_n$ and $p_{n+1j}=1$ for $k<j<r$.

The sequence of $r$-tuples so generated is without duplications and is such that for any $k$, $1\leqslant k\leqslant r$, and any positive non-zero integer $\mu$, every $k$-tuple of numbers $m$, $1\leqslant m\leqslant\mu$, occurs in the sequence $(p_{n(r-k)+1},\ldots,p_{nr})$, $n=1, 2, \ldots$, before any $k$-tuple with a number $\mu+1$ appears.

A sentence $S$ in standard form is a member of the set $D$ if and only if there is no (existentially quantified) variable of $S$ dependent upon any (universally quantified) variable of $S$. A sentence $T$ is then a member of the set $M$ if and only if the sentence $-T$ has a standard form which is a member of $D$.

Let there occur $s$ names in $S$. Let $S$ not be in $D$. A sequence $(q_{n1}, q_{n2}, \ldots, q_{nm})$, $n=1, 2, \ldots$, of $m$-tuples of positive non-zero integers is generated as follows:

(1) For all $k$ and $n$, if $X_k$ is universally quantified and $k=u_j$, then $q_{nk}=p_{nj}$;

(2) For all $k$, if $X_k$ is existentially quantified and is dependent upon the variables $X_{uj_1},\ldots,X_{uj_t}$, then (**a**) $q_{1k}$ is the maximum of $s+1$, $q_{11}+1,\ldots,q_{1k-1}+1$; (**b**) for all $n>1$, if $h$ is the smallest positive integer for which $(p_{nj_1},\ldots,p_{nj_t})$ is identical with $p_{hj_1},\ldots,p_{hj_t})$, then if $h<n$, $q_{nk}$ is $q_{hk}$; while if $h=n$, $q_{nk}$ is the maximum of $s+1$, $q_{11}+1,\ldots,q_{1m}+1,\ldots,q_{n-11}+1,\ldots,q_{n-1m}+1$, $q_{n1}+1,\ldots,q_{nk-1}+1$; in particular if $t=0$, then $q_{nk}=q_{1k}$.

Should $S$ be in $D$ so that it can be assumed that in the sequence $Q_1, Q_2, \ldots, Q_m$ no $A$ precedes an $E$, then a finite sequence $(q_{n1},\ldots,q_{nm})$, $n=1,2,\ldots,\max(1,(m-r+s)^r)$, of $m$-tuples of positive non-zero integers is generated as follows:

(1) For all $k$ and $n$, if $X_k$ is universally quantified and $k=u_j$, then $q_{nk}=p_{nj}$;

(2) For all $k$ and $n$, if $X_k$ is existentially quantified, then $q_{nk}$ is the maximum of $s+1$, $q_{11}+1,\ldots,q_{1k-1}+1$. In particular, therefore, if either $r=0$ or $r=m$, then only one term is generated.

Let $a_1, a_2, \ldots, a_s, a_{s+1}, \ldots$ be a list of names of quantification theory in which the names $a_1, a_2, \ldots, a_s$ which occur in $S$ are all listed first. Let $P_1$ be the matrix $M(a_{q11}, \ldots, a_{q1m})$, and for $n>1$, let $P_n$ be the product matrix of $P_{n-1}$ with the matrix $M(a_{q_{n1}},\ldots,a_{q_{nm}})$. For $S$ in $D$, $P_n$ is defined for $n\leqslant\max(1,(m-r+s)^r)$, while for $S$ not in $D$, it is defined for all $n\geq 1$. Then for any $n$ and $j$, $I_{nj}$ is the set of ground sentences conjoined together to form the $j^{\text{th}}$ conjunction of $P_n$. Let $k_n$ be the number of conjunctions of $P_n$.

• *Theorem 1*

For $S$ not in $D$ the sets $I_{nj}$ possess the following properties:

(1) For any $n$ for which $k_{n+1}>0$ and any $j$, $1\leq j\leq k_{n+1}$, there exists an $i$ such that $I_{ni}\subseteq I_{n+1j}$;

(2) For any function $\phi$ for which $I_{n\phi(n)}\subseteq I_{n+1\phi(n+1)}$ for

$n \geq 1$, $S$ is true for any completion of $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$;

(3) For any interpretation $I''$ for which $S$ is true there exists an interpretation $I'$, $I' \subseteq I''$, for which $S$ is true and a function $\phi$ for which $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$ for $n \geq 1$, such that $I'$ is the homomorphic image of some completion of $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$.[9]

• *Proof*

That they possess property (1) is immediate. A completion $I$ of a set $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$ is formed from the latter set by adding to it ground sentences consistent with it formed from names and predicate letters occurring in members of the latter set. Hence, all of the sentences $M(a_{q_{n1}}, \ldots, a_{q_{nm}})$, $n=1, 2, \ldots$, are true for $I$ and therefore also $S$ is true for $I$.

Let $S$ be true for $I''$. Choose a sequence $b_1, b_2, \ldots$ of names of $I''$ as follows:

(1) If names occur in $S$, list these first in the order in which they are listed in $a_1, a_2, \ldots, a_s$;

(2) If no names occur in $S$ and $Q_1$ is $A$, choose any name of $I''$ as $b_1$;

(3) Assume that the sequence of names has been completed for $t$ members ($t$ may be 0 if $s=0$ and $Q_1$ is $E$). Let $n$ be the smallest integer for which $q_{nj}=t+1$ for some $j$. It then follows that for only one $k$ is $q_{nk}=t+1$, so that one can assume that $b_{q_{n1}}, \ldots, b_{q_{nk-1}}$ have already been chosen, and further that $Q_k$ is $E$. Hence, choose $b_{q_{nk}}$ to be such that $(Q_{k+1}X_{k+1}) \ldots (Q_mX_m)M(b_{q_{n1}}, \ldots, b_{q_{nk}}, X_{k+1}, \ldots, X_m)$ is true for $I''$.

For any $n$, $M(b_{q_{n1}}, \ldots, b_{q_{nm}})$ is true for $I''$. Let $P'_1$ be $M(b_{q_{11}}, \ldots, b_{q_{1m}})$ and let $P'_n$ be the product matrix of $P'_{n-1}$ and $M(b_{q_{n1}}, \ldots, b_{q_{nm}})$. Let $I'_{nj}$ be the set of ground sentences in the $j^{\text{th}}$ conjunction of $P'_n$. Then since $P'_n$ is true for $I''$ for any $n$, there must be a $\phi'$ such that the $\phi'(n)^{\text{th}}$ conjunction of $P'_n$ is true for $I''$ and $I'_{n\phi'(n)} \subseteq I'_{n+1\phi'(n+1)}$ for any $n$. Since $I''$ is complete and $\bigcup_{n=1}^{\infty} I'_{n\phi'(n)} \subseteq I''$, there is a unique completion $I'$ of $\bigcup_{n=1}^{\infty} I'_{n\phi'(n)}$ such that $I' \subseteq I''$. Since for any $n$, $M(b_{q_{n1}}, \ldots, b_{q_{nm}})$ is true for $I'$, $S$ also is true for $I'$.

From the manner in which the sets $I'_{n\phi'(n)}$ have been defined, it follows that if $J'_{nj}$ is the set of ground sentences in the $j^{\text{th}}$ conjunction of $M(b_{q_{n1}}, \ldots, b_{q_{nm}})$ then there exists a function $\gamma$ such that $\bigcup_{k=1}^{n} J'_{k\gamma(k)} = I'_{n\phi'(n)}$ for all $n$. Let $J_{nj}$ be the set of ground sentences in the $j^{\text{th}}$ conjunction of $M(a_{q_{n1}}, \ldots, a_{q_{nm}})$ and let $\phi$ be a function such that $\bigcup_{k=1}^{n} J_{k\gamma(k)} = I_{n\phi(n)}$ for all $n$. Let the map $\psi$ be the single-valued map of the names $a_1, a_2, \ldots$ onto the names $b_1, b_2, \ldots$ defined by: for all $i$, $\psi(a_i)=b_i$. Let $I^*$ be the interpretation with ground sentences $G$ as members for which $\psi(G)$ is in $I'$. Then since $\bigcup_{n=1}^{\infty} I_{n\phi(n)} \subseteq I^*$, there exists a unique completion $I$ of $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$ such that $I \subseteq I^*$. $\psi$ is a homomorphism of $I$ onto $I'$.

*Corollary*. For $S$ not in $D$ there exists an interpretation for which $S$ is true if and only if $k_n>0$ for all $n$.

Since by (1) if $k_n>0$ for all $n$ then there exists a function $\phi$ for which $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$ for all $n$, and hence by (2) there exists an interpretation for which $S$ is true. Conversely if there is an interpretation for which $S$ is true then by (3) there is such a $\phi$ and hence $k_n>0$ for all $n$.

• *Theorem 2*

When $S$ is in $D$ and $N=\max(1, (m-r+s)^r)$, the sets $I_{nj}$ possess the following properties:

(1) For any $n$, $1 \leq n < N$, for which $k_{n+1}>0$ and any $j$, $1 \leq j \leq k_{n+1}$, there exists an $i$ such that $I_{ni} \subseteq I_{n+1j}$.

(2) For any function $\phi$ for which $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$ for $1 \leq n < N$, $S$ is true for any completion of $\bigcup_{n=1}^{\infty} I_{n\phi(n)}$.

(3) For any interpretation $I''$ for which $S$ is true there exists an interpretation $I'$, $I' \subseteq I''$, for which $S$ is true and a function $\phi$ for which $I_{n\phi(n)} \subseteq I_{n+1\phi(n+1)}$, for $1 \leq n < N$, such that $I'$ is the homomorphic image of some completion of $\bigcup_{n=1}^{N} I_{n\phi(n)}$.

Properties (1) and (2) follow in exactly the same manner as in Theorem 1. The proof of (3) is similar to that in Theorem 1 except that the sequence $b_1, b_2, \ldots$ of names of $I''$ is finite, the sequence being terminated after all the names in $S$ have been listed and either one member has been chosen for each existentially quantified variable in $S$, should one exist, or otherwise after only one member has been chosen. Then $(b_{q_{n1}}, \ldots, b_{q_{nm}})$ is defined for $1 \leq n \leq N$. Similarly one can prove:

*Corollary*. For $S$ in $D$ there exists an interpretation for which $S$ is true if and only if $k_n>0$ for $n \leq N$.

## The program to produce proofs for logically true sentences

The input for the program is a standard form of the negation of the sentence $T$ to be proven. The input data provides, therefore, a matrix form as well as a list of quantifiers with the dependencies of the existential quantifiers indicated. The matrix form data consists of a list of 36-bit computer words each member in the list being an atomic wff, negated or not, or a dividing word to indicate the occurrence of the connective "v". The negated and unnegated atomic wff are expressed within the computer words as follows: The sign bit is used to indicate the occurrence of the negation sign, a negative word being a negated atomic wff. The next five bits are used for the predicate letter while the remaining 30 bits are used for the variables in a manner dependent upon the number of variables. The 30 bits are broken up, as nearly as possible, into equal fields, the number of which is the same as the

maximum number of variables attached to predicate letters in the sentence to be proven. Variables are expressed as non-zero binary numbers occupying the fields of the 30 bits. The names which replace the variables are also binary numbers and occupy the same field as the variable they replace. Thus positive and negative ground sentences and unnegated and negated atomic wff are represented in the program within a single computer word in exactly the same manner. There results a limitation on the maximum number of names that can be introduced which is determined by the maximum number of variables attached to predicate letters; for example, if this number is 6 then the maximum number of names that can be introduced is $2^5-1$, or 31.

In outline the program is very simple. A substitution generator generates the $m$-tuples $(a_{q_{n1}}, \ldots, a_{q_{nm}})$, $n=1, 2, \ldots$, and as each is produced it is substituted into the matrix form to produce the matrices $M(a_{q_{n1}}, \ldots, a_{q_{nm}})$. As each matrix $M(a_{q_{n1}}, \ldots, a_{q_{nm}})$, $n>1$, is produced it is multiplied by the previous product matrix $P_{n-1}$ and the resulting product matrix $P_n$ is tested to determine whether or not it has any conjunctions. Should $P_n$ have no conjunctions then a proof has been produced for the sentence to be proven. Should the sentence $T$ to be proven be a member of $M$ and should $n=N$ for $T$, then if $P_n$ has a conjunction, $T$ is not provable. Should $n<N$ or should $T$ not be a member of $M$ then if $P_n$ has a conjunction the program goes on to produce $P_{n+1}$.

This outline of the program is not complete, however, since in order to conserve computing time and space, the matrices are coded before being multiplied. A matrix is coded by expressing each of its conjunctions as a pair of 36-bit computer words as follows: As the ground sentences are produced by substitution into the negated and unnegated atomic wff of the matrix form, they receive in turn a code number from 1 to 36, a ground sentence with a negation sign attached receiving the same number as the ground sentence without the negation sign. In the pair of words representing the conjunction of a matrix, a zero in the $i^{\text{th}}$ place of the first word indicates that the negative ground sentence with that code number is a member of the conjunction, and a zero in the $i^{\text{th}}$ place of the second word indicates that the positive ground sentence with that code number is a member of the conjunction.

The limitation of 36 on the number of distinct positive ground sentences that may appear is severe. Therefore, immediate consideration is being given to writing a program in which each conjunction of a matrix is expressed by two, four, six, et cetera, words as needed, allowing the limitation to be raised to 72, 108, 144, et cetera. However, two features in the program for conserving code numbers make the limitation less severe than would first be apparent.

A ground sentence which appears in every conjunction of some $P_n$ must necessarily appear in every conjunction of $P_{n+j}$ for $j>0$. Hence such common factors can be removed from all of the conjunctions of $P_n$ and put onto a special list called the *truth list*. Since a ground sentence on the truth list does not require a code number, the inclusion in the program of a routine to remove common factors results in a saving of code numbers.

If $I_{nj} \subseteq I_{nk}$, for some $j$ and $k$, then it is clear from Theorems 1 and 2 that there is no loss in ignoring the set $I_{nk}$ and considering only further the set $I_{nj}$. For the program this amounts to discarding redundant conjunctions from $P_n$, a redundant conjunction being one for which another conjunction of $P_n$ exists, of which each ground sentence occurs in the redundant conjunction. After the removal of redundant conjunctions, fewer distinct atomic sentences may occur in a product matrix than before the removal so that code numbers may be freed for reuse.

The main steps in the program can now be fully described.

(1) Should $T$ not be in $M$ or should $T$ be in $M$ and $n$ be not greater than $N$ for $T$, then generate a new substitution $(a_{q_{n1}}, \ldots, a_{q_{nm}})$ according to the dependencies of existentially quantified variables on universally quantified variables. Should $T$ be in $M$ and should $n$ be greater than $N$, then $T$ is not logically true and the program stops.

(2) Generate the $j^{\text{th}}$ conjunction of $M(a_{q_{n1}}, \ldots, a_{q_{nm}})$ by substituting $a_{q_{n1}}$ for $X_1, \ldots, a_{q_{nm}}$ for $X_m$ in the $j^{\text{th}}$ conjunction of the matrix form of the input. If a member of the conjunction contradicts any ground sentence on the truth list, or another ground sentence in the conjunction, discard the conjunction and go to (5).

(3) Determine the code numbers that have been previously assigned to the ground sentences of the conjunction, and assign new numbers to those ground sentences which have not previously been assigned numbers. Should a code number larger than 36 be required, the program stops. Express the conjunction as a pair of words.

(4) Form the product of the coded $j^{\text{th}}$ conjunction with the previous product matrix, dropping from the resulting new product any contradictory conjunctions. As each conjunction of the new product matrix is formed, test to see if it is redundant or if it makes an already appearing conjunction of the new product matrix redundant, and store it according to the results of this test.

(5) Check to see if the new product matrix is complete; that is check to see if $j$ is the number of conjunctions in the matrix form. If the product matrix is not complete go to (2) to generate the $j+1^{\text{th}}$ conjunction; if it is, go to (6).

(6) Check to see if the product matrix is empty. If it is, the proof is complete and the program stops. If it is not, remove common factors from the product matrix and put them onto the truth list. Determine and record all code numbers that have been freed for reuse.

The only change that would be necessary in order to use the program for proving theorems in an axiomatic theory with axioms $S_1, S_2, \ldots$, would be in (1) and (2). Instead of only generating substitutions for a matrix form obtained from the negation of the theorem $T$ to be proven, the program would also have to generate substitutions for matrix forms obtained from the axioms. The remainder of the program would be unchanged.

*Table 1* **The sentences to be proven.**[11]

(1) $(Ex)(Ay)(Az)\{[((Fy\supset Gy)\equiv Fx)\,\&\,((Fy\supset Hy)\equiv Gx)\,\&\,(((Fy\supset Gy)\supset Hy)\equiv Hx)]\supset(Fz\,\&\,Gz\,\&\,Hz)\}$.

(2) $(Ex)(Ey)(Az)\{[(Fxz\equiv Fzy)\,\&\,(Fzy\equiv Fzz)\,\&\,(Fxy\equiv Fyx)]\supset(Fxy\equiv Fxz)\}$.

(3) $(Ex)(Ay)(Az)\{[((Fyz\supset(Gy\supset Hx))\supset Fxx)\,\&\,((Fzx\supset Gx)\supset Hz)\,\&\,Fxy]\supset Fzz\}$.

(4) $(Ex)(Ey)(Az)\{(Fxy\supset(Fyz\,\&\,Fzz))\,\&\,((Fxy\,\&\,Gxy)\supset(Gxz\,\&\,Gzz))\}$.

(5) $\{[(Ax)(Ey)(Fxy\vee Fyx)\,\&\,(Ax)(Ay)(Fxy\supset Fyy)\supset(Ez)Fzz\}$.

(6) $(Ax)(Ey)(Px\supset(Py\vee Qy))$, where the atomic wff "$Px$" is replaced by: $(Eu)(Av)(Fux\supset(Gvu\,\&\,Gux))$, the atomic wff "$Py$" is replaced by a corresponding wff, and the atomic wff "$Qy$" is replaced by: $(Au)(Av)(Ew)((Gvu\vee Hwyv)\supset Guw)$.

(7) $\{[(Ax)(Kx\supset(Ey)(Ly\,\&\,(Fxy\supset Gxy)))\,\&\,(Ez)(Kz\,\&\,(Au)(Lu\supset Fzu))]\supset(Ev)(Ew)(Kv\,\&\,Lw\,\&\,Gvw)\}$.

(8) (3) in which the atomic wff "$Hx$" is replaced by: $(Au)(Ev)Huvx$, and the atomic wff "$Hz$" is replaced by a corresponding wff.

(9) $(Ax)(Ey)(Az)\{(Pyx\supset(Pxz\supset Pxy))\,\&\,(Pxy\supset(-Pxz\supset(Pyx\,\&\,Pzy)))\}$, where the atomic wff "$Pxy$" is replaced by: $(Au)(Ev)(Fxuv\,\&\,Gyu\,\&\,-Hxy)$, and the other atomic wffs are replaced by corresponding wffs.

*Table 2* **The input.**

| | *Quantifier List* | *No. of Conjunctions* |
|---|---|---|
| (1) | $(Ez)(Ax)(Ey)$ | 18 |
| (2) | $(Ax, y)(Ez)$ | 2 |
| (3) | $(Ax)(Ey, z)$ | 4 |
| (4) | $(Ax, y)(Ez)$ | 4 |
| (5) | $(Ax)(Ey)(Az)$ | 4 |
| (6) | $(Ex)(Ay)(Ez, u, v)(Aw)(Ex')(Ay', z')$ | 8 |
| (7) | $(Ex)(Ay)(Ez)(Av, w)$ | 9 |
| (8) | $(Ax)(Ey, z, u)(Av)(Ew)$ | 6 |
| (9a) | $(Ex)(Ay)(Ez, u)(Av)(Ew)(Ax')(Ey')(Az', u', v')$ | 21 |
| (9b) | $(Ex)(Ay)(Ez, u, v, w)(Ax')(Ey', z')(Au', v', w')$ | 21 |

**Results from production runs**[10]

The sentences for which proofs were attempted by the program are given in Table 1. Information about the quantifier list and matrix function provided by the sentences of Table 1 are given in Table 2. Finally in Table 3 the results of the production runs are given. No names appear in any of the sentences used as inputs. None of the inputs to the program are in the class $D$ although Example (1) is a member of class $M$.

In evaluating the difficulty of Examples (6), (8), and (9), it is important to recognize that although they are described as substitutions into easily proven theorems (the sentence of (9) into which the substitution is made can be proven by the program in less than one-hundredth of a minute), they are themselves only easily proven when

*Table 3* **The results.**

| | *Status* | *Time* | *Proof* | *No. of Subst.* | *No. of Conj.* | *Truth L.* | *Code No.* |
|---|---|---|---|---|---|---|---|
| (1) | yes | 0.01 | yes | 4 | 4 | 10 | 0 |
| (2) | no | 0.01 | no | 11 | 2 | 0 | 36+ |
| (3) | yes | 1.42 | yes | 13 | 590 | 53 | 33 |
| (4) | yes | 21* | no | 7 | 2900+ | 7 | 30 |
| (5) | yes | 0.01 | yes | 3 | 2 | 4 | 5 |
| (6) | yes | 0.12 | yes | 27 | 24 | 16 | 14 |
| (7) | yes | 0.01 | yes | 5 | 6 | 4 | 12 |
| (8) | yes | 0.74 | no | 10 | 150 | 6 | 36+ |
| (9a) | yes | 15.06 | no | 3 | 1900 | 3 | 36+ |
| (9b) | yes | 21* | no | 6 | 1850 | 5 | 35 |

*Status* indicates whether or not the sentence is logically true. *Time* is given in minutes. *Proof* indicates whether or not a proof was produced. *No. of Subst.* is the number of times a new matrix was generated. *No. of Conj.* is the maximum number of conjunctions in any product matrix. *Truth L.* is the number of entries on the truth list. *Code No.* is the maximum number of code numbers used. The two starred examples were manually stopped.

the substitutions are recognized. When the substitutions are disguised, as they are in the input data for the program, the difficulty of the example is considerably increased.

One run was used to evaluate that part of the program which removes redundant conjunctions from the product matrix, as this portion of the program consumes a large proportion of the running time when the product matrix is large. The result was to prove conclusively the value of this portion of the program.

A number of examples other than those used in production runs were considered but rejected because they are all too easy for the program; that is, the program can produce a proof for them in less than one-hundredth of a minute. Included in such examples are all of the syllogisms. Indeed, as the syllogisms belong to that special class of sentences which are decidable by the program, the program can decide of any syllogism whether or not it is valid within one-hundredth of a minute.

In Table 2, the dependencies indicated by the order of the quantifiers in the quantifier list were the only ones indicated. Two different equivalent quantifier lists for Example (9) were tried. The number of conjunctions is the number in the matrix function. Variables in a group with $E$ or $A$ indicate a sequence of quantifiers.

## Conclusions

Without considering the results of the production runs, it might be concluded that the program had little chance of success. With each multiplication the number of conjunctions in the product matrix can increase rapidly. But the removal of contradictory conjunctions and redundant conjunctions, in some cases at any rate, keeps the number of conjunctions down to a manageable size. The limitation on the number of coding numbers, although quite stringent, is not as serious in practice as would first appear. The results certainly encourage the writing of programs with double or triple the present number of coding numbers. Pessimism regarding the program is confirmed, however, in one respect. If the quantifier list of the input contains $m$ universal quantifiers, then in order to consider all possible substitutions of up to $k$ names into the universally quantified variables it is necessary to generate $k^m$ matrices and form their product. Thus, for example, in (9a) if all possible substitutions of up to only three individuals are to be considered then $3^6$ or 729 matrices must be generated and multiplied, the product matrix of which can have up to $21^{729}$ conjunctions! Considering the number of multiplications performed by the program in production runs, it is clear that success can only be had with such a problem by a program which is more discriminating in the matrices that it generates and multiplies. Example (4) is a good case in point. A very short proof of this sentence can be produced by hand simulation if a very obvious refinement is added to the routine for generating matrices. Nevertheless, the program has succeeded in producing proofs for moderately complicated sentences. For simple logical deductions such as the syllogisms, the program has very fast logical facility.

## References and footnotes


1. Actually, from a theorem of Church in "An Unsolvable Problem of Elementary Number Theory," *American Journal of Mathematics,* **58,** 345-363, it can be concluded that there can exist no effective process for the construction of interpretations for sentences. In particular, therefore, there can exist no effective process for determining, in general, whether or not it is true that $k_n>0$ for all $n$.
2. E. W. Beth, "Semantic Entailment and Formal Derivability," Amsterdam, North Holland Publishing Co., 1955. K. J. J. Hintikka, "Form and Content in Quantification Theory," appearing in "Two Papers on Symbolic Logic," Helsinki, *Acta Philosophica Fennica,* Fasc. VIII, 1955.
3. G. Gentzen, "Untersuchungen über das logische Schliessen," *Mathematische Zeitschrift* **39,** 176-210, 405-431 (1934-5). J. Herbrand, *Recherches sur la théorie de la démonstration,* Travaux de la Société des Sciences et des Lettres de Varsovie, Classe III sciences mathématiques et physiques, No. 33 (1930).
4. A. Newell, J. C. Shaw and H. A. Simon, "Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics," *Proceedings of The Western Joint Computer Conference,* 218-230 (1957). Further references can be obtained from this paper.
5. H. L. Gelernter and N. Rochester, "Intelligent Behavior in Problem-Solving Machines," *IBM Journal,* **2,** 336-345 (1958). See also, H. Gelernter, "Realization of a Geometry Theorem Proving Machine," *Proceedings of the International Conference on Information Processing, Paris,* 1959.
6. B. Dunham, R. Fridshal and G. L. Sward, "A Non-Heuristic Program for Proving Elementary Logical Theorems," *Proceedings of the International Conference on Information Processing, Paris,* 1959.
7. Hao Wang, "Toward Mechanical Mathematics," see p. 2, this journal.
8. There has come to my attention the work of D. Prawitz, H. Prawitz, and Neri Voghera of Stockholm. A brief outline of their work appears in the *Proceedings of the International Conference on Information Processing, Paris,* 1959, in the discussion of the session on theorem proving.
9. A stronger result can be proven but is not needed for the purposes of this paper. For the stronger result $I'$ is such that any sentence is true for $I'$ if and only if it is true for $I''$.
10. First announced in a paper "A Program for the Production of Proofs for Theorems Derivable within the First Order Predicate Calculus from Axioms," *Proceedings of the International Conference on Information Processing, Paris,* 1959.
11. Alonzo Church, *Introduction to Mathematical Logic,* Princeton University Press, 1956, is the source of Example 1 to 4 and 9. They are, respectively, **Ex. 3,** p. 262; **2,** p. 265; **1,** p. 262; **5,** p. 265; and **2,** p. 262. Example 7 appears in Rosser, J. B., *Logic for Mathematicians,* McGraw-Hill Book Co., New York, p. 150 **Ex. (e).** Example 6 and the modification for Example 9 are due to J. D. Rutledge.


*Received April 20, 1959*