4

Advances and Problems in Mechanical Proof Procedures

D. Prawitz
University of Stockholm

Abstract

In this paper, I first give a simplified description of the method proposed in my earlier paper 'An improved proof procedure'. This method is then further developed in order to diminish the work of finding a substitution that makes a given formula inconsistent. Finally, the possibility of developing the method in another direction is discussed.

By a proof procedure (or proof method) I understand in this connection an algorithm $\mathscr{A}$ with the property: for any valid formula F in the predicate calculus, $\mathscr{A}(F)$ (i.e., the result of applying $\mathscr{A}$ to F) is a proof of F. Equivalently, one may consider algorithms $\mathscr{A}$ with the property: if F is not satisfiable, then $\mathscr{A}(F)$ is a refutation of F. Following a tradition, I shall use this latter formulation. (It is of course immaterial which formulation one chooses, and when it comes to programming the procedure, all differences disappear completely.)

1. THE PRIMITIVE METHOD

Introduction

To have a convenient reference for the following discussion, I start by giving a description of the original method developed by Skolem (1928) and the main theorem on which this method is based. The necessary logical apparatus can be kept remarkably simple.

We use a formulation of predicate logic containing individual constants and function symbols. To simplify the description of the method, it is convenient to restrict the formulae F to which the method is applicable. Firstly, it is supposed that F is closed and in prenex normal form. Secondly, it is supposed that all existential quantifiers are eliminated. To see how this can

The main part of this paper was also presented in lectures at the University of Stockholm and the Technische Hochschule of Hanover in the spring of 1967.

be done, consider the formula $\forall x \exists y F(x,y)$. If this formula is true in some model, we may for each value of x choose a particular value of y so that these values satisfy the formula $F(x,y)$; in other words, there exists a so-called Skolem-function that satisfies the formula $\forall x F(x,f(x))$ in the given model. It follows that if $\forall x \exists y F(x,y)$ is satisfiable, then so is $\forall x F(x,f(x))$; the converse of this is, of course, also true.

Given a closed formula F in prenex normal form, I shall say that F^* is a *Skolem-transformation* of F (and that F^* is *Skolem-transformed*), if F^* is obtained from F by eliminating every existential quantifier, and by replacing the corresponding variable in the formula by a term $f^n(x_1,x_2,\ldots,x_n)$, where $x_1,x_2,\ldots,x_n$ are exactly the variables that are quantified by the universal quantifiers which precede the eliminated existential quantifier in question; for different existential quantifiers, different function symbols not occurring in F are to be chosen. A function symbol with zero arguments is the same as an individual constant and will sometimes be denoted by the letter a (a_0 is to denote a particular individual constant). By the same reasoning as in the example above, it is seen that F is satisfiable if and only if the Skolem-transform F^* is also satisfiable.

Further example. Given a formula $\exists x \forall y \exists z \forall v \exists w F(x,y,z,v,w)$, where $F(x,y,z,v,w)$ is a formula without quantifiers, we replace it by a Skolem-transform $\forall y \forall v F(a,y,f(y),v,g(y,v))$.

Let F be a Skolem-transformed formula. We define the so-called *Herbrand-universe* for F, denoted H_F, as the set of individual terms that can be built up using the individual constants and function symbols of F; in case there is no individual constant in F, we may use the constant a_0. In other words, F_F is recursively defined by:

1. All individual constants in F belong to H_F; if there is no such constant, a_0 belongs to H_F.
2. If $t_1,t_2,\ldots,t_n$ belong to H_F and f^n occurs in F, then $f^n(t_1,t_2,\ldots,t_n)$ belongs to H_F.

Main theorem

A formula F of the form $\forall x_1 \forall x_2 \ldots \forall x_n M(x_1,x_2,\ldots,x_n)$ where $M(x_1,x_2,\ldots,x_n)$ contains no quantifier, is satisfiable if and only if every finite subset of the set

$$\{M(t_1,t_2,\ldots,t_n)\colon t_i \in H_F, \text{for every } i \leqslant n\} \qquad (1)$$

is satisfiable.

Proof. Since $\forall x_1 \forall x_2 \ldots \forall x_n M(x_1 x_2,\ldots,x_n)$ logically implies $M(t_1,t_2,\ldots,t_n)$ one half of the theorem is trivial. The proof of the other half is conveniently divided into two parts, proving

1. If (1) is satisfiable then so is F.
2. If K is a set of quantifier-free formulae and every finite subset of K is satisfiable, then so is K.

The proof of 1 is easy: given a model of the set (1) we eliminate every individual from the domain of the model that is not a value of a term in H_F. The substructure obtained in this way is a model of F.

The proof of 2 is slightly more complex and seems first to have been given by Skolem (1929). In summary, the argument is: let $K = \{F_1,F_2,\dots\}$ be an infinite set of quantifier-free formulae, and let Γ_n be the set of the truth-value assignments to the atomic formulae occurring in $F_1,F_2,\dots,$ or F_n, that satisfy the set $\{F_1,F_2,\dots, F_n\}$. The following facts are easily seen to be true. (i) Γ_n is not empty (by assumption in 2). (ii) Γ_n is finite. (iii) Each member of Γ_{n+1} is an extension of some member of Γ_n. (i)–(iii) imply that there exists an infinite sequence $T_1,T_2,\dots$ such that $T_n \in \Gamma_n$ and T_{n+1} is an extension of T_n (the argument is the same as in König's Unendlichkeitslemma: an infinite tree where each branching is finite contains an infinite branch). The union of all the assignments in this sequence is a truth-value assignment that satisfies K.

The method

Since one can decide the question whether a finite set of quantifier-free lines is satisfiable or, in other words, consistent, the theorem immediately gives rise to a method of the kind sought: searching through the finite subsets of (1) and testing for consistency, one will sooner or later find an inconsistent set if F is not satisfiable; this inconsistent set can be taken as the refutation of F.

Instead of considering all the subsets of (1), one may fix some order $M_1,M_2,\dots$ among the formulae in (1), henceforth called the *instances* of F, and then restrict attention to the sets $M^m = \{M_1,M_2,\dots, M_m\}$. This is sufficient, since if there is an inconsistent subset of (1), then this subset is contained in some M^m, and, hence, M^m is also inconsistent. The algorithm for showing that F is not satisfiable is then:

For successively larger m, form the set $\{M_1,M_2,\dots, M_m\}$ and test for consistency, until an inconsistent set is found.

This method was first stated by Skolem (1928) and is the core of all known methods that do not simply enumerate all proofs. (Many later methods differ in no essential respect from Skolem's.) The first attempts to program Skolem's method for computers were made independently by Gilmore (1959 and 1960), Prawitz, Prawitz, and Voghera (1959 and 1960), and Wang (1960). These three programs are all rather similar, one difference being that the formulae are not required to be in prenex normal form in the programs proposed by Prawitz *et al.*, and by Wang (cf. the remark in section 4 below).

Problematic features in Skolem's method

There are two main problems involved in Skolem's method. The first concerns the question of deciding whether a quantifier-free formula is consistent. This may of course be done by considering all the 2^n different assignme nt of truth-values to the n atomic formulae in the formula. A less tedious method is to develop the formula into disjunctive normal form and see whether every clause contains a contradiction. This method is the one used in all the three programs mentioned above. (Wang uses the technique of simplifying Gentzen-sequents – developed especially by Kanger (1957) – which essentially only accomplishes the task of transforming a formula to conjunctive normal form. Prawitz, Prawitz, and Voghera use the technique of semantic tableaux developed in Beth (1955), which accomplishes the same task but gives a more economical, tree-formed presentation of the required normal form.) However, this method is still very laborious. If one obtains k conjunctive clauses when transforming M_1 to disjunctive normal form (in the usual way by multiplication), one will obtain k^m clauses when similarly transforming M_1 & M_2 & . . . & M_m.

The second problem and the main weakness of Skolem's method may be illustrated in the following way. Suppose that F is inconsistent and let $M_1,M_2,\ldots$ be some ordering of the instances of F. Then the least inconsistent set of instances is probably much smaller than the least inconsistent set among the sets $\{M_1,M_2,\ldots, M_m\}$. The set $\{M_5,M_{61},M_{100}\}$ may, for example, be inconsistent, but the first inconsistent set among the sets $\{M_1,M_2,\ldots, M_m\}$ may well be $\{M_1,M_2,\ldots, M_{100}\}$. While it would have been an easy task to refute F if the instances had been generated in the order M_5, M_{61}, M_{100} a machine might be exhausted by trying to decide that M_1 & M_2 & . . . & M_{100} is inconsistent.

The problem thus concerns the order in which the instances of the formula F are generated. One would like to have a method for generating these instances in such an order that a *minimal* inconsistent set of instances is found (i.e., there is to be no smaller inconsistent set of instances).

More efficient procedures for dealing with the first problem were developed by Dunham, Fridsal, and Sward (1959) and by Davis and Putnam (1960). A method for dealing with the second problem was proposed by Prawitz (1960), by which a minimal inconsistent set of instances is always found for every inconsistent formula.[1]

2. THE IMPROVED METHOD

Introduction

In this section, I give a somewhat simplified presentation of the method proposed in Prawitz (1960). The main idea is that instead of generating the instances of the given formula $\forall x_1 \forall x_2 \ldots \forall x_n M(x_1,x_2,\ldots, x_n)$ in some arbitrarily defined order, one should find by calculations the values which substituted for $x_1,x_2,\ldots, x_n$ give an inconsistent set of instances. (One may compare the situation with that of solving ordinary equations like $x+17 = 32$;

[1] For a survey of the development in this area up to 1964, see Cooper (1966).

one may, of course, solve the equation by enumerating the natural numbers, but a more efficient procedure is to use the ordinary subtraction algorithm.) The basic procedure is first to decide whether any substitution for $x_1, x_2, \ldots, x_n$ makes $M(x_1, x_2, \ldots, x_{2n})$ inconsistent. If this is not the case, one asks whether any substitution for $x_1, x_2, \ldots, x_{2n}$ makes $M(x_1, x_2, \ldots, x_n)$ & $M(x_{n+1}, x_{n+2}, \ldots, x_{n+n})$ inconsistent; and so on.

One thus requires some method for calculating whether there is any substitution for the variables that makes the formula

$$M(x_1,x_2,\ldots,x_n) \mathbin{\&} M(x_{n+1},x_{n+2},\ldots,x_{n+n}) \mathbin{\&} \ldots \mathbin{\&} M(x_{kn+1},x_{kn+2},\ldots,x_{kn+n}) \quad (k = 0,1,\ldots) \tag{2}$$

inconsistent (other than simply trying the different possibilities).[1]

Definitions

Instead of dealing with actual substitutions, I shall consider different *substitution conditions,* which will be built up from equations $t = u$, where t and u are terms (not necessarily in the Herbrand-universe), by using conjunction and disjunction. For instance, $f(x) = f(g(y))$ is an (atomic) substitution condition. A substitution of terms from H_F for the variables in t and u are said to satisfy the atomic substitution condition $t = u$, if t and u are transformed into two occurrences of the same term by the substitution. For instance, the substitution of a for y and $g(a)$ for x satisfies the condition $f(x) = f(g(y))$, but there is no substitution that satisfies the condition $x = f(x)$ or $f(x) = g(y)$. A substitution satisfies a conjunction (disjunction) of substitution conditions, if every (some) condition in the conjunction (disjunction) is satisfied by the substitution. *Example:* the condition $f(x) = f(g(y)) \vee y = f(x)$ is satisfied by the substitution considered above, but the condition $f(x) = f(g(y)) \mathbin{\&} y = f(x)$ is not satisfied by any substitution; to satisfy the first equation, we must satisfy $x = g(y)$, and to satisfy both this condition and $y = f(x)$, we must satisfy $x = g(f(x))$, which is impossible.

It is obviously not difficult to set up an algorithm for deciding whether a substitution condition is satisfiable or not.

A pair of atomic formulae $P(t_1, t_2, \ldots, t_n)$ and $\sim P(u_1, u_2, \ldots, u_n)$ such that the substitution condition

$$t_1 = u_1 \mathbin{\&} t_2 = u_2 \mathbin{\&}, \ldots, \mathbin{\&} t_n = u_n \tag{3}$$

is satisfiable is said to be a *possible contradiction.* The condition (3) is said to be the *corresponding substitution condition* to this possible contradiction.

[1] In simple cases, the question whether (2) is inconsistent may be answered by developing the formula into disjunctive normal form and then 'looking' at the formula. This is essentially the procedure used by Kanger (1959 and 1963), which was developed independently of my method.

The method

A systematic method for deciding whether any substitution turns a formula of the form (2) into a contradiction is now as follows:

1. Develop the formula (2) into disjunctive normal form.
2. For each clause i in the disjunction obtained by step 1, form the condition $\alpha_1 \vee \alpha_2 \ldots \vee \alpha_{j_i}$, called C_i, where $\alpha_1, \alpha_2, \ldots, \alpha_{j_i}$ are all the substitution conditions that correspond to possible contradictions among formulae occurring in the clause i.
3. Form the condition C_1 & C_2 & ... & C_m, where $C_1, C_2, \ldots, C_m$ are all the conditions obtained in step 2.
4. Decide whether the condition formed in step 3 is satisfiable.

A necessary and sufficient condition for a substitution to turn the formula (2) into a contradiction is obviously that it satisfies the condition formed in step 3. The method for refuting a formula $\forall x_1 \forall x_2 \ldots \forall x_n M(x_1, x_2, \ldots, x_n)$ is thus to form (2) for successively larger k and go through the steps 1–4 above. The procedure is continued until the question in step 4 is answered by 'yes', which means that the given formula is not satisfiable. To get a set of inconsistent instances of the given formula, one may then take any substitution that satisfies the condition formed in step 3 and carry out this substitution on the last formed formula (2).

3. FURTHER DEVELOPMENTS OF THE IMPROVED METHOD

Introduction

The method described obviously finds a minimal inconsistent set of instances if an inconsistent set of instances exists. As described above, the method uses the technique of developing formulae of propositional logic into disjunctive normal form. Since we now have to deal only with a minimum number of instances of the given formula, this is not a drawback as serious as the one noted in connection with the primitive method, but it is, of course, desirable to develop the method further. One may try, therefore, to combine this method with the methods for deciding whether a formula of propositional logic is inconsistent that were developed by Dunham, Fridsal and Sward and Davis and Putnam, but there is no obvious way of doing that. Attempts to combine the two methods have been made, however, by Davis (1963), and this work has been continued by Chinlund, Davis, Hinman, and McIlroy (1967) and by Loveland (1968). Robinson (1965), whose work has been continued by several authors, has used some of the ideas of the improved method but has developed them in another direction. Davis's method sacrifices some of the strength of the improved method in not finding a minimal inconsistent set of instances (cf. footnote, p. 65). Robinson's method cannot immediately be compared in this respect. It avoids the use of disjunctive normal form but uses instead another method that seems rather inefficient (cf. footnote, p. 67).

Matrices

The form given to the method as described above was intended to facilitate understanding. When carrying out the procedure in practice, it is, for example, not necessary actually to develop the formula (2) into disjunctive normal form, or to form the whole condition mentioned in step 3. To discuss some questions of this kind I shall now reformulate the method.

Let $M(x_1,x_2,\ldots,x_n)$ be written in conjunctive normal form, and let us represent it by the following matrix

$$\begin{array}{l} A_{1,1},A_{1,2},\ldots,A_{1,n_1} \\ A_{2,1},A_{2,2},\ldots,A_{2,n_2} \\ \vdots \\ A_{m,1},A_{m,2},\ldots,A_{m,n_m}, \end{array} \tag{4}$$

where we have left out the disjunction signs between the formulae on the same line and the conjunction signs between the different lines. By a *path* in the matrix, is meant a sequence $A_{1,j_1},A_{2,j_2},\ldots,A_{m,j_m}$; i.e., a path is obtained by taking exactly one literal from each line. The disjunctive normal form of the formula is thus obtained by forming a conjunction of all the literals of a path and then taking the disjunction of all these conjunctions.

We now formulate a necessary and sufficient condition for the existence of a substitution that turns the matrix (4) into a contradiction:

The matrix (4) *is made inconsistent by some substitution for the variables iff there exists a set S of atomic subsitution conditions with these properties:*

1. *Each path of the matrix* (4) *contains at least one possible contradiction such that the atomic parts of the corresponding substitution condition belong to S.*
2. *The substitution conditions in S are simultaneously satisfiable.*

A set S of substitution conditions that satisfies 1 and 2 will be called a *refutation set* for the matrix (4).

Various methods besides the one described in section 2 may be used to find a set S that satisfies 1 and 2. For instance, one may go through the different paths of the matrix in some order, picking out a possible contradiction for each path and forming the corresponding substitution condition. In this way, one successively builds up a set S of substitution conditions. One has to make sure that each new substitution condition is satisfiable simultaneously with the conditions already in S. If one comes to a path where no such substitution condition can be formed, one has to go back to the last path which has some possible contradiction that has not yet been tried.[1] Erasing the substitution

[1] The method developed by Davis *et al.* differs from the one discussed here in that one does not try all different possible contradictions; some choices of possible contradictions are never reconsidered, and instead one enlarges the matrix.

conditions that were formed at that and later paths, one then starts anew from that path. If one comes to a path where no substitution condition can be combined with the already formed conditions and there remains no untried possible contradictions in the preceding paths, the procedure ends with a negative result. One has then to form a new matrix by making an alphabetic change of variables and enlarge the old matrix by writing the new one below. The whole procedure can then be repeated.[1]

Matrix reduction

The method outlined in the preceding paragraph has some advantages as compared with that described in section 2. The outlined procedure will usually decide whether there is a substitution that makes a given matrix inconsistent in a shorter time. The method is also more economical with respect to space. One need not store the development into disjunctive normal form (the development into conjunctive normal form being a much more straightforward matter); in fact, the information that has to be stored at any one moment is not essentially more extensive than that contained in the minimal refutation itself.

However, the method is still rather time-consuming. When there is a substitution that makes a given matrix inconsistent, one must, with the procedure above, consider all the paths of the matrix in order to find the refutation set. The main problem, it would seem, is to find a faster procedure for building up the refutation set. Such a procedure is also possible to construct; in fact, it is not at all necessary to consider all the paths of the matrix when building up the refutation set (a point already made in Prawitz, 1960, p. 118).

It may be noted that a possible contradiction in a path of a matrix usually belongs to several other paths in the matrix as well. Hence, when we pick out a possible contradiction from one path and introduce the corresponding substitution condition in the set S in order to arrange that this path be in accordance with clause 1 in the definition of refutation set, we have then guaranteed that a number of other paths to which the possible contradiction belongs also conform to clause 1. Therefore, these other paths need not be considered, and one would like to have a procedure that automatically disregards paths that already conform to clause 1. A method of accomplishing this is embodied in the following principle:

Principle of matrix reduction

Let M be a matrix of the form (4) *above, let α be the substitution condition that corresponds to a possible contradiction* $(A_{i,j},A_{k,p})$, *where $i \neq k$ and none of*

[1] The account given above (and some of the improvements in the next paragraphs) are on the whole identical with ideas included in a report written by the author in 1960 and presented to Statens Tekniska Forskningsråd, which supported these researches at that time.

$A_{i,j}$ and $A_{k,p}$ stand alone on their lines (i.e., $n_i,n_k > 1$), *and let S be a substitution that satisfies α. Then, S turns M into a contradiction if and only if it turns both M′ and M″ into contradictions, where*

M′ is obtained from M by striking out $A_{k,p}$ and all literals on the line i except $A_{i,j}$, and

M″ is obtained from M by striking out $A_{i,j}$ and all literals on the line k except $A_{k,p}$.

The problem of finding a refutation set for the matrix M is thus reduced to the problem of finding a set containing α that is a refutation set of both M' and M''. In this way, we cut out not only the paths containing the possible contradiction $(A_{i,j},A_{k,p})$ but also all paths in the matrix M^* that are obtained from M by striking out $A_{i,j}$ and $A_{k,p}$. To see the validity of the principle, we note that every path in M that does not contain both $A_{i,j}$ and $A_{k,p}$ is also a path in one of the matrices M', M'' and M^*. But if every path in M' and M'' contains a contradiction, then so does every path in M^*. Hence, it is sufficient to consider the paths in M' and M''. This is equivalent to showing the validity of the equivalence

$$(A \vee B) \,\&\, (\sim A \vee C) \,\&\, D \equiv (A \,\&\, C \,\&\, D) \vee (B \,\&\, \sim A \,\&\, D).$$

Example. Having formed the substitution condition $x = a$, we may replace the matrix M below by the two matrices M' and M'' and try to find a refutation set for them that contains $x = a$:

M	M'	M''
$Qx,f(x), Px$	$Qx,f(x), Px$	$Qx,f(x), P(x)$
$Pa, \sim Qa,f(a), Px$	Pa	$\sim Qa,f(a), Px$
$\sim Px, Qx,y, Qa,f(a)$	$Qx,y, Qa,f(a)$	$\sim Px$
$\sim Pz, \sim Qx,f(a)$	$\sim Pz, \sim Qx,f(a)$	$\sim Pz, \sim Qx,f(x)$

While M contains 36 paths, M' and M'' contain together only 16 paths.

When applying the principle of matrix reduction to a matrix, we shall say that the matrix is *split* into the two simple matrices. We also have the following simpler case of reduction:

Addition to the principle of matrix reductions. *Let M, α, $(A_{i,j},A_{k,p})$, and S be as in the principle above except that either $A_{i,j}$ or $A_{k,p}$ but not both stands alone on its line. Then, S turns M into a contradiction if and only if it turns M′ or M″, respectively, into a contradiction.*

Applications of this additional principle are called *simple reductions.*[1]

Example continued. If we always use a possible contradiction in the leftmost path when applying the principle of matrix reduction, we see that three simple

[1] This method has some similarities to Robinson's method. However, instead of the principle of matrix reduction, Robinson uses a method that he calls the principle of resolution, which seems less efficient. The principle of resolution is based on the fact that $(A \vee B) \,\&\, (\sim A \vee C)$ implies $B \vee C$. By replacing the former formula with the latter one, one looses much information. In contrast, matrix reduction reduces the formula to

reductions of M' turn M'' into a matrix containing a possible contradiction between literals that both stand alone on their lines. Similarly, one splitting of M'' gives two inconsistent matrices.

Trees of matrices

The principle of matrix reduction may be used thus: given a matrix M, we build up a set S of substitution conditions and a tree structure of matrices in successive steps. The given matrix M is placed at the initial node of the tree. A possible contradiction is located in its leftmost path and the corresponding substitution condition is introduced into S. The principle of matrix reduction is now applied to this possible contradiction. If the matrix is split into two matrices M' and M'', the tree branches at the node of M, and at the immediately succeeding nodes we place M' and M''; if we have a simple reduction, the matrix obtained is placed at the unique node that immediately succeeds the node of M. We then apply the same procedure to the matrix (matrices) obtained by the reduction. If we choose a possible contradiction between two literals that are alone on their lines, the corresponding node is to be an end-node. The procedure terminates with S as a refutation set for M when each branch ends with an end-node.

Before introducing a substitution condition into S, we make sure that the condition is satisfiable simultaneously with S. If at some point we come to a matrix where no such substitution condition compatible with S corresponds to a possible contradiction in the leftmost path, we have to go back to a preceding matrix where an untried possible contradiction remains in the leftmost path, and start anew from this matrix (erasing the matrices at succeeding nodes and the corresponding substitution conditions in S). If there is no such preceding matrix, the matrix at the initial node has to be enlarged as before and the whole process started anew.

4. CONCLUDING REMARKS

Individual variables

The more the variables in a matrix are distinct and the more the individual constants and function symbols are similar, the easier it is to find a substitution that makes the matrix inconsistent. The operation of Skolem-

$(A \& B) \vee (\sim A \& C)$. The two methods have also many other differences, which are not so easy to compare. The above procedure has also some features in common with a procedure proposed by Hans Karlgren.

Simple reductions and matrix splitting bear some resemblance to the rules 1 and 3 respectively, in Davis and Putnam (1960). These rules are applied to formulae that are instances of the given formula. The present reduction principles may be said to be an application of these rules to the situation where the formulae still contain free variables, and, in a way, we have thus combined the ideas in Davis and Putnam (1960) and Prawitz (1960); this combination however is quite different from the one in Davis (1963).

transformation was defined for formulae in prenex normal form but the operation can obviously be extended to formulae in general; the details are left to the reader. One can then often diminish the number of arguments of the function symbols that replaced the existential quantifiers. Furthermore, one can arrange for no variable to have occurrences in two different lines of the matrix (4). (It is also possible, but less important, to replace two different function symbols that have the same number of arguments by the same symbol, if they occur in exactly one line of the matrix (4).) These possibilities are useful especially when applying the procedure to axiomatic theories. They amount essentially to the possibility of treating the axioms separately in the steps preparatory to the main procedure. The number of instances of the matrix that are needed to find an inconsistency can often be diminished in this way.

Alphabetic change of individual variables can also be made when splitting a matrix M into two matrices M' and M'': such a change can be made for the occurrences of a variable occurring in M'' in lines not affected by the reduction so that they become distinct from the variable occurrences in M'. (This idea was already incorporated in Prawitz (1960) in the form of the so-called interval index.) That this does not invalidate the procedure is seen by realizing the equivalence between the formula

$$\forall x \forall y \forall z \forall v((Px \vee Qy) \,\&\, (\sim Px \vee Rz) \,\&\, Sv)$$

and the formula

$$\forall x \forall y \forall z \forall v \forall w((Px \,\&\, Rz \,\&\, Sv) \vee (Qy \,\&\, \sim Px \,\&\, Sw)).$$

Strong minimal procedures

Although the various algorithmic procedures that we have considered in sections 2 and 3 never construct more instances of the initial matrix (4) than are needed for finding an inconsistency, it is possible that some of the matrices constructed are only partially needed. In other words, one may find an inconsistent matrix by choosing different numbers of instances of different lines, a situation which is common when proving theorems in axiomatic theories, where different axioms are not usually invoked the same number of times. Davis (1963) has considered the possibility of guessing the proportion between the number of times different lines of the initial matrix have to be used in order to find an inconsistent matrix.

However, one may also contemplate a procedure that finds an inconsistent matrix which is minimal in the strong sense that no matrix with fewer lines is inconsistent. Starting with a given matrix of the form (4), one may try different possibilities of building up an inconsistent matrix by choosing lines from the given matrix. First, one may consider all matrices containing only two different lines. Only lines that contain literals which together form a possible contradiction need to be considered. If one is to prove theorems in an axiomatic theory and if one is not interested in trying to find an inconsistency among the axioms, one may also require that one line comes from a theorem. By applying the principle of matrix reduction, one then investigates whether any one of these matrices can be made inconsistent. If the result is negative, all matrices obtained by different reductions are stored for later use. Instead of forming substitution conditions, however, one may carry out a corresponding substitution. One then considers all possibilities of enlarging the stored matrices by adding a third line containing a literal which forms a possible contradiction together with some literal from the matrix. One then considers all possibilities of reducing these matrices, and so on.

A procedure of this kind requires the storage of much more information, but may have other advantages. By successively enlarging the matrices obtained by different previous applications of the principle of matrix reduction, one utilizes the information from previous attempts to find an inconsistent matrix. Furthermore, two matrices that have been obtained by splitting may now be continued in different ways, that is, by the addition of different lines – and this is an obvious advantage.

It seems difficult to decide which one of these two procedures is the best – the one considered in section 3, or the one now outlined – if this question can be made at all precise. In any case, it seems worthwhile to work out the outlined procedure in more detail and then try both methods.

REFERENCES

Beth, E.W. (1955) Semantical entailment and formal derivability. *Med. der Kon. Nederl. Akad. Van Wetensch.*, **18**, no. 13. Amsterdam.

Chinlund, T., Davis, M., Hinman, P. & McIlroy (1967) Theorem-proving by matching. *Communications of the ACM*, to appear.

Cooper, D.C. (1966) Theorem proving in computers. *Advances in programming and non-numerical computation*, pp. 155–82 (ed. Fox, L.). Oxford: Pergamon Press.

Davis, Martin (1963), Eliminating the irrelevant from mechanical proofs. *Proceedings of Symposia in Appl. Math.*, **15**, pp. 15–30.

Davis, M., & Putnam, H. (1960) A computing procedure for quantification theory. *J. Ass. comput. Mach.* **7**, 201–15.

Dunham, B., Fridsal, R., & Sward, G. (1959) A non-heuristic program for proving elementary logical theorems. *Proceedings of the international conference on information processing*, pp. 282–7. Paris: UNESCO.

Gilmore, P.C. (1959) A program for the production from axioms of proofs for theorems derivable within the first order predicate calculus. *Proceedings of the international conference on information processing*, pp. 265–73. Paris: UNESCO

Gilmore, P.C. (1960) A proof method for quantification theory; its justification and realization. *I.B.M. Jl. Res. Devl.*, **4**, 28–35.

Kanger, Stig (1957) *Provability in logic.* Studies in Philosophy, **1**. Stockholm: Almqvist and Wiksell.

Kanger, Stig (1959) *Handbok i logik* (mimeographed). Stockholm.

Kanger, Stig (1963) A simplified proof method for elementary logic, *Computer programming and formal systems*, pp. 87–94 (ed. Braffort, P. & Hirschberg, D.). Amsterdam.

Loveland, D.W. (1968) Mechanical theorem proving by model elimination. *J. Ass. comput. Mach.*, **15**, 236-51.

Prawitz, Dag (1960) An improved proof procedure. *Theoria*, **26**, 102–39.
Prawitz, D., Prawitz, H. & Voghera, N. (1959) Discussion. *Proceedings of the international conference on information processing*, p. 273. Paris: UNESCO.
Prawitz, D., Prawitz, H., & Voghera, N. (1960) A mechanical proof procedure and its realization in an electronic computer. *J. Ass. comput. Mach.*, **7**, 102–28.
Robinson, J. A. (1965) A machine oriented logic based on the resolution principle. *J. Ass. comput. Mach.*, **12**, 23–44.
Skolem, T. (1928) Über die mathematische Logik. *Norsk matematisk Tidskrift*, **10**, 125–42.
Skolem, T. (1929) Über einige Grundlagenfragen der Mathematik. *Skrifter utgitt av det Norske Videnskaps-Akademie i Oslo*, I. Mat.-naturv. klasse, no. 4, Oslo.
Wang, Hao (1960) Towards mechanical mathematics. *IBM Jl Res. Dev.*, **4**, 2–22.