Computing 12, 43—55 (1974)

An Approach to a Systematic Theorem Proving Procedure in First-Order Logic

By

W. Bibel, München

With 1 Figure

Received May 13, 1973

Abstract — Zusammenfassung

An Approach to a Systematic Theorem Proving Procedure in First-Order Logic. A complexity degree for theorems in first-order logic is introduced which naturally reflects the difficulty of proving them. Relative to that degree it is required that a systematic proof procedure should prove simple theorems faster than harder ones. Such a systematic but relatively inefficient procedure and a semi-systematic but efficient procedure are presented. Both are developed on the basis of the consistency and completeness theorem for the underlying formal system rather than Herbrand's theorem.

Ein Ansatz zu einem systematischen Beweisverfahren für die Prädikatenlogik. Für Theoreme der Prädikatenlogik erster Stufe wird ein Komplexitätsgrad eingeführt, der in natürlicher Weise die Kompliziertheit der zugehörigen Beweise mißt. Im Sinne dieses Grades wird von einer systematischen Beweisprozedur verlangt, daß sie einfache Theoreme schneller beweist als schwierigere. Solch ein systematisches, jedoch relativ ineffizientes Verfahren und ein halb-systematisches, jedoch effizientes Verfahren werden in dieser Arbeit beschrieben. Beide Verfahren stützen sich auf den Konsistenz- und Vollständigkeitssatz des zugrundeliegenden formalen Systems und nicht, wie üblich, auf den Herbrand-Satz.

0. Introduction

In the field of theorem proving in first-order logic almost all work is based on Herbrand's theorem. This is a surprising fact since from a logical point of view the most natural way to prove a theorem (syntactically) is by giving a derivation for it, and this, of course, is primarily not the way suggested by Herbrand's theorem. Therefore a thorough study of how to find a derivation of a theorem might yield results and techniques for theorem proving procedures in a more natural, perhaps easier way.

This motivates why, in this paper, a proof procedure has been presented which uses Prawitz' ideas [3] and is based on the usual consistency and completeness theorem for the underlying formal system rather than Herbrand's theorem.

Hereby, a sort of a complexity degree for theorems will play an important role which reflects in a natural way the difficulty of proving them. For example it

will be proved (see chapter 2) that, relative to this concept, it is advantageous to transform a given formula into an equivalent anti-prenex or miniscope [7] form before trying to prove its validity (and disadvantageous to transform it into prenex form as has been done in quite a few papers).

Moreover this degree suggests another important requirement on theorem proving procedures, namely the natural demand to prove "simple" theorems faster than "hard" ones where these terms are measured in a way which, originally, is independent of the procedure as it is done with the proposed complexity degree. It is not hard to give such a "systematic" procedure, *i.e.* one which satisfies this requirement (see chapter 3). But it turns out to be rather complicated to give one, which is also efficient.

In the way this problem is tackled here one is faced with two major problems. One of these has been solved (see chapter 4) resulting in a "semi-systematic" procedure which fulfills that requirement only for a subclass of formulae (proving all others in later steps than it could be done with a systematic procedure).

1. The Formal System

For simplicity the discussion is restricted to first-order logic without equality. Among the various Gentzen-type systems we choose one developed by Schütte [6], which is briefly described in the next few paragraphs.

Let a, b denote free object variables, x, y bound object variables, f, g n-ary function symbols, and p, q n-ary predicate symbols. (All denotations here and in the following are understood to be used with or without indices.) Addition of the logical symbols $\neg$, $\vee$, V, the comma, and parentheses completes the alphabet, say A. Let $*$ be a special symbol. Then by $\mathbf{F}$, $\mathbf{G}$ we denote words over $A \cup \{*\}$ and call them indicative forms. If W is a word over A and $\mathbf{F}$ an indicative form then $\mathbf{F}[W]$ denotes the word over A obtained from $\mathbf{F}$ by replacing each occurence of $*$ in $\mathbf{F}$ by W.

Terms are the free variables and the words of the form $f(t_1, \ldots, t_n)$ over A where $t_1, \ldots, t_n$ denote terms.

Atomic formulae are of the form $p(t_1, \ldots, t_n)$ and are denoted by P.

Formulae, denoted by A, B, F, G, are defined inductively as follows.

1. Each atomic formula is a formula.
2. With F, G also $\neg F$, $(F \vee G)$ are formulae.
3. If F is a formula of the form $\mathbf{F}[a]$ and does not contain x, then $V x \mathbf{F}[x]$ is also a formula.

Inductive definition of *positive parts* of a formula F.

1. F itself is a positive part of F.
2. If G is a positive part of F and is of the form $\neg\neg G_0$ or $(G_1 \vee G_2)$ then G_0, or G_1 and G_2, resp., are positive parts of F.

E.g., in $(\neg G_0 \vee (\neg \neg G_1 \vee \neg (G_2 \vee G_3)))$ the positive parts are the whole formula, $\neg G_0$, $(\neg \neg G_1 \vee \neg (G_2 \vee G_3))$, $\neg \neg G_1$, G_1, $\neg (G_2 \vee G_3)$.

A positive part of a formula which does not contain another positive part than itself is called a *minimal part.*

Positive forms, denoted by **P**, are indicative forms satisfying the following two conditions.

1. The $*$ occurs exactly once in **P**.
2. If F is any formula, then $\mathbf{P}[F]$ is a formula with F being positive part in it.

Axioms are all those formulae in which an atomic formula and its negation occur as positive parts.

Rules of inference are

r 1. $\mathbf{P}[\neg F], \mathbf{P}[\neg G] \vdash \mathbf{P}[\neg (F \vee G)]$
r 2. $\mathbf{P}[\neg \mathbf{F}[a]] \vdash \mathbf{P}[\neg \forall x \mathbf{F}[x]]$, where the *eigenvariable* a may not occur in the conclusion
r 3. $\mathbf{P}[\forall x \mathbf{F}[x]] \vee \mathbf{F}[t] \vdash \mathbf{P}[\forall x \mathbf{F}[x]]$; t is called *eigenterm* of the inference.

This concludes the description of the formal system. Without restricting generality, affecting any further definition, or changing efficiency of any of the following algorithms, conjunction and all-quantification have been omitted, since they are easily defined within the system in the usual way. This allows us to deal with only three rules.

In the rest of this section we prepare and state the definition of "degree of a theorem".

Let us call the *main part* of an inference the (unique) positive part in the conclusion which is indicated by the involved positive form (e.g., $\neg (F \vee G)$ in $\mathbf{P}[\neg (F \vee G)]$ in r 1). Each positive part in the conclusion of an inference according r 1 or r 2, except those which contain the main part of this inference, occurs also in each premise. In the case of r 3 this applies even to *all* positive parts in the conclusion without exception. We express this commutative relation on the occurrences of the positive parts in the formulae of an inference also by saying that the positive part in the conclusion *corresponds* to that (unique) positive part in each premise (cf. [6]). This relation provides a commutative relation on the occurrences of the positive parts of the formulae in a derivation by taking the union of the relations of all inferences occuring in the derivation, which we extend to its transitive closure. Still we say related elements *correspond* to each other. Furthermore two inferences r 3 in a derivation are called *corresponding* if their main parts correspond to each other.

Now the *degree of a derivation* is the maximal number of inferences r 3 which correspond to each other and lie in the same branch of the derivation tree.

A theorem T has (complexity) *degree* d iff all derivations of T have degree $\geq d$ and there is a derivation of T whose degree is d.

Consider the theorem $Vx \neg Vy \neg (p(x) \vee \neg p(y))$ (abbreviated by T) and the following derivation of T as an example.

$(T \vee \neg \neg (p(a) \vee \neg p(b))) \vee \neg \neg (p(b) \vee \neg p(c)) \vdash$

$(T \vee \neg \neg (p(a) \vee \neg p(b))) \vee \neg Vy \neg (p(b) \vee \neg p(y)) \vdash$

$(T \vee \neg \neg (p(a) \vee \neg p(b)) \vdash T \vee \neg Vy \neg (p(a) \vee \neg p(y)) \vdash T.$

In this derivation all the T's correspond to each other and T occurs twice as the main part of an inference according to $r3$; therefore the degree of the derivation is 2 by definition. It is easy to see that in this example the number of corresponding inferences according to $r3$ cannot be reduced by transforming the derivation to another one of the same formula. Thus T has degree 2.

In theorem proving the formula is given and the derivation has to be found which means that the rules of inference have to be applied in a backward way. This is a straightforward process as far as the rules $r1$ and $r2$ are concerned; the difficulty lies in the transition from the conclusion to the premise of an inference according to $r3$ since the conclusion gives no hint which choice of the eigenterm to make. Therefore only the inferences according to $r3$ contribute to the degree. Among them only those are added up which correspond to each other and lie in the same branch of the derivation because all others are relatively independent from these and the difficulty of the hardest of the independent subproblems seems to be adequate as a rough measure for the difficulty of the whole problem.

For further motivation we mention here without proof that this particular degree of theorems coincides in the case of a prenex theorem T with the number of instances of the matrix of T over its Herbrand universe, which are needed to prove T using a procedure based on Herbrand's theorem (see e.g. [5]). This number is widely accepted to determine the number of basic steps of such a procedure. Therefore, as the degree of a theorem turns out as a generalization for all theorems, not only prenex ones, it is proposed here to give it an analoguous role in procedures which are not restricted to prenex theorems.

Apart from this, it is the fact that to any given positive integer n there are theorems of degree higher than n, which makes first-oder logic an undecidable theory. Therefore it seems to be natural to introduce the partition $\mathbf{T} = \bigcup_{i=0}^{\infty} \mathbf{T}_i$, $\mathbf{T}_i$ denoting the class of theorems of degree i, of the class $\mathbf{T}$ of all theorems in first-order logic and regard a theorem $T^{(i)} \in \mathbf{T}_i$ essentially harder to prove than another $T^{(j)} \in \mathbf{T}_j$ iff $j < i$ holds.

2. Formulae in Antiprenex Form

Formulae in antiprenex or miniscope form have been used in proof procedures as early as 1960 by Wang [7] and very recently by Ernst [2]. The main idea of independent subgoals in the latter paper comes out automatically in the procedure presented here.

First, the inductive definition of "formulae in antiprenex form" is given here for the system introduced in the previous chapter (whereby the discrimination between free and bound variables will be neglected for simplicity).

1. Each atomic formula is in antiprenex form.
2. With F, G also $\neg F$ and $(F \vee G)$ are in antiprenex form.
3. If a formula of the form $\neg (F \vee G)$ is in antiprenex form, then $V x \neg (F \vee G)$ is in antiprenex form iff x occurs in F and in G.
4. If a formula of the form $\neg V y F$ is in antiprenex form then $V x \neg V y F$ is in antiprenex form iff x occurs in F.
5. If a formula of the form $V y F$ is in antiprenex form, then $V x V y F$ is in antiprenex form iff $V x F$ is in antiprenex form.
6. If F is (the negation of) an atomic formula then $V x F$ is in antiprenex form iff x occurs in F.

If a formula F is of the form $\neg G$ then let $\bar{F}$ be G else let $\bar{F}$ be $\neg F$. Then the rules according to which a formula can be transformed recursively into antiprenex form are the following, Q denoting a (possibly empty) string of (existential) quantifiers.

1. $\neg \neg A \Rightarrow A$
2. $V x F \Rightarrow F$, x not in F
3. $V x Q (A \vee B) \Rightarrow Q (V x A \vee B)$, x in A, not in B
4. $V x Q \neg (A \vee B) \Rightarrow Q \neg (\neg V x \bar{A} \vee B)$, x in A, not in B
5. $V x Q (A \vee B) \Rightarrow Q (A \vee V x B)$, x in B, not in A
6. $V x Q \neg (A \vee B) \Rightarrow Q \neg (A \vee \neg V x \bar{B})$, x in B, not in A
7. $V x Q (A \vee B) \Rightarrow Q (V x A \vee V x B)$, x in A and B.

As announced in the introduction it will be stated in the following theorem that for theorem proving antiprenex is preferable to prenex form, relative to the concept "degree of a theorem".

Theorem. *a) If F is a formula then there exists a uniquely determined equivalent formula F' in antiprenex form which is obtained from F by applying the transformation rules*1.—7.
b) If F is a theorem then the degree of F' is not greater than that of F and in general it is also not equal.

Part a) follows from well-known equivalences in first-order logic.

The second part of part b) can be shown by an example. In chapter 1 it has already been shown that the theorem $V x \neg V y \neg (p(x) \vee \neg p(y))$ has degree 2. Its antiprenex form is $V x p(x) \vee \neg V y p(y)$ which is of degree 1 because of $(V x p(x) \vee \neg p(a)) \vee p(a) \vdash V x p(x) \vee \neg p(a) \vdash V x p(x) \vee \neg V y p(y)$.

A thorough proof of the first half of b) is rather lengthy. We therefore content ourselves with a short outline. Let the degree of F be d and $\mathbf{H}$ be a derivation of F of degree d. In each step towards a transformation of F into antiprenex form one of the transformation rules is applied; for example assume F contains a part $\forall x(A \vee B)$ and A, B, already in antiprenex form, contain x (case 7). Replacing that part in F by $(\forall x A \vee \forall x B)$ resulting in F_1, $\mathbf{H}$ is to be transformed in a natural way into a derivation $\mathbf{H}_1$ of F_1 with same degree. This is easily done by transferring the change in F to an analogue change in the premise(s) of F and so forth through the whole derivation. In that process the only complication occurs if the main part $\forall x(A \vee B)$ or $\neg \forall x(A \vee B)$ of an inference according to $r3$ or $r2$ changes to $(\forall x A \vee \forall x B)$ or $\neg(\forall x A \vee \forall x B)$, resp. In the first case the inference $\mathbf{P}[\forall x(A \vee B)] \vee (A' \vee B') \vdash \mathbf{P}[\forall x(A \vee B)]$ has to be transformed to $(\mathbf{P}[\forall x A \vee \forall x B) \vee A') \vee B' \vdash \mathbf{P}[\forall x A \vee \forall x B] \vee A' \vdash \mathbf{P}[\forall x A \vee \forall x B]$. Since these two new inferences according to $r3$ do not correspond to each other the degree of $\mathbf{H}_1$ will remain the same as of $\mathbf{H}$. In the second case we have to consider an inference $\mathbf{P}[\neg(A' \vee B')] \vdash \mathbf{P}[\neg \forall x(A \vee B)]$. From the derivation of the premise one gets easily a derivation of $\mathbf{P}[\neg A']$ and another of $\mathbf{P}[\neg B']$ (see [6]); $\mathbf{P}(\neg A'] \vdash \mathbf{P}[\neg \forall x A]$, $\mathbf{P}[\neg B'] \vdash \mathbf{P}[\neg \forall x B]$, from which one gets $\mathbf{P}[\neg(\forall x A \vee \forall x B)]$ by $r1$. — The discussion for the other cases 1.—6. is similar. The process after a finite number of steps results in a derivation $\mathbf{H}'$ of F' in antiprenex form of degree $d' \leq d$, i.e. for the degree d'' of F' the maintained relation $d'' \leq d' \leq d$ holds.

What is called antiprenex form here is a further simplified "simplified miniscope normal form" introduced in [2], because no advantage is taken from the associative laws of the propositional operations (see chapter 5).

3. A Systematic Procedure

Since we now have available a concept, namely the degree of a theorem, with which we can measure the complexity of a theorem, it is quite natural to ask for a procedure that tries to identify a given formula (in antiprenex form) as a theorem of degree 1 in the first step, in case of failure as a theorem of degree 2 in the second step, and so on. Let us refer to such a procedure as a *systematic* procedure. Even if this would result in an algorithm which requires much more combinatorial work to be done in each step it would be worthwhile to pursue this aim because we know from conventional procedures how rapidly the amount of work increases from step to step; i.e., the gain obtained by eventually reducing the number of steps may make up for the increase of work in each step.

As far as I looked through the literature I found only one theorem whose equivalent antiprenex form was of degree higher than 1 (namely of degree 2, see [8], 7.3 example 1). So I have a strong feeling that for "everyday" reasoning one could restrict such a systematic procedure to rather low degrees.

The simplest but admittedly least efficient algorithm of that type is described as follows and uses Prawitz's idea of introducing dummies in the place of terms

(see [3]). It consists of two subroutines. One is called *TRF* and is applied recursively. It transforms at each call a matrix, whose elements are subformulae of the given formula which is to be proved, to another matrix by removing a logical operation in a certain element. Its arguments are the matrix (M), the row number (r), the list of variables and dummies (*VD*), the list of forbidden relations (R), and the step number (s). The process in the i-th step, $i=1, 2, \ldots$, is initiated by calling *TRF* with M being a matrix whose only element is the given formula, $r=1$, *VD*, R being the empty list, $s=i$.

The other subroutine, which is called *CPT*, is activated whenever *TRF* is not applicable anymore and tests whether each row can be made to contain a complementary pair of literals, i.e. an atomic formula and its negation, by replacing dummies by terms while observing the forbidden relations, returning a success or failure message, say "yes" or "no", resp. It is not described in detail here because a version for a special case is found in chapter 4 (subroutine *AX*). If *CPT* fails then the process continues with the next step.

This section concludes with a brief description of *TRF*.

1. Each element in M is replaced by its minimal parts.

2. If all elements in the current row are either literals or labelled with the number s (the current step) then

a) if r is less than the number of rows in M then call *TRF* with M, $r+1$, *VD*, R, s as arguments and return the result; otherwise

b) call *CPT* and return the result; otherwise

3. assume $M_1, \ldots, M_n$ are those elements in row r which are not literals and not labelled with s.

Set $j \leftarrow 1$.

a) If M_j is of the form $\neg (A \vee B)$ then call *TRF* with M', r, R, s where M' results from M by adding a copy of row r to the rows in M and replacing in row r and in that additional row M_j by the minimal parts in $\neg A$ or $\neg B$, respectively.

b) If M_j is of the form $\neg V x \mathbf{F}[x]$ then call *TRF* with M', r, *VD'*, R', s where M' results from M by replacing M_j by the minimal parts in $\neg \mathbf{F}[a]$, a is a new variable, *VD'* is a added to *VD*, R' is R together with the restrictions that a may not be contained in any element of *VD*.

c) If M_j is of the form $V x \mathbf{F}[x]$ and labelled with a number 1 (supposed to be zero if no label) then call *TRF* with M', r, *VD'*, R, s where M' results from M by replacing M_j by $V x \mathbf{F}[x]$ labelled with $l+1$ and all minimal parts in $\mathbf{F}[d]$, where d is a new dummy, and *VD'* is d added to *VD*.

If the answer of the procedure call in one of the cases a)—c) is "yes" then return this result; otherwise if $j<n$ then increase j by one and start in 3. a) again; otherwise return "no" as result, indicating that the formula does not turn out as a theorem in the s-th step.

An example may be helpful. Obviously, the formula $\neg V x\, p(x) \to \Lambda\, x \neg p(x)$ is valid. By definition, in our formal system it reads $\neg \neg V x\, p(x) \vee \neg V x \neg \neg p(x)$, shortly denoted by F. By step 1 it is transformed to the matrix $(V x\, p(x), \neg V x \neg \neg p(x))$. Then, in step 3, j is set equal to 1, to consider one possible premise of F, and, by 3. c, TRF is called again with $M' = (V x\, p(x)^{(1)}, p(d\,1), \neg V x \neg \neg p(x))$, $VD' = (d\,1)$, which results, by 3. b, with $M'' = (V x p(x)^{(1)}, p(d\,1), \neg p(a))$, $r = 1$, $VD' = (d\,1, a)$, $R' =$ ("a not in $d\,1$"). M'' cannot be made an axiom by any suitable substitution of a term for the dummy $d\,1$ because $d\,1 = a$ is forbidden by R'; therefore, after another call of TRF, CPT and TRF return "no" to the original call of TRF, where now j is increased by 1, to try the other possible premise of F, namely $M' = (V x\, p(x), \neg p(a))$, $VD' = (a)$. R' now remains the empty list, so that $M'' = (V x\, p(x)^{(1)}, p(d\,1), \neg p(a))$ now can be made an axiom by setting $d\,1 = a$ yielding the "yes"-result in the first step. (Actually the procedure could now tell us more, namely the whole derivation of F.)

For what follows, it would be helpful to keep in mind that such a little modification as interchanging the two disjunctive components in F (which yields $\neg V x \neg \neg p(x) \vee \neg \neg V x\, p)$) would have avoided the intermediate "no-"-result.

4. An Efficient, Semi-Systematic Procedure

The disadvantages of the procedure in section 3 are obvious. First of all, after each step it forgets everything what it possibly might have learnt in earlier steps, working it out again and again. Moreover, it neglects completely that in the situation as described in step 3 the ordering of the M_i's $i = 1, \ldots, n$ is relevant with respect to efficiency for the following reason: If we assume the case where the loop in step 3 eventually returns "yes" then obviously there is a permutation π and with it the tuple $M_{\pi(1)}, \ldots, M_{\pi(n)}$ which in comparison with all other permutations causes a minimal number of procedure calls and returns the "yes" result while j only has the value 1.

The problem to find out this optimal permutation can naturally be divided in subproblems such that in each of those one has to test whether it is more efficient to treat some element M_i before some other M_j or the other way around (TRF simply tries both ways for all pairs M_i, M_j).

The relevance of this test is dependent on the structure of the formulae M_i and M_j. In the cases where both M_i and M_j are of the form $V x F$, $\neg V x F$, or $\neg (F \vee G)$, resp., and the case where M_i is of the form $\neg V x F$ and M_j of the form $\neg (F \vee G)$ (or vice versa) the choice turns out to be irrelevant in general. This is a consequence of the wellknown fact that in a derivation two inferences according to the same rule $r\,1$, $r\,2$ or $r\,3$, resp., or to $r\,1$ and $r\,2$, which do not correspond (in an extended sense) to each other, can be interchanged without affecting the remaining derivation.

Therefore only two cases are left where the choice (to handle the part M_i before M_j or M_j before M_i) in general is not irrelevant. This is whenever one is of the

form $V x F$, the other $\neg V y G$ or one is of the form $V x F$, the other $\neg (F_1 \vee F_2)$. Let us refer to them as the *first* and the *second problem.*

How could the choice for these two cases be made in an efficient way? I suggest a solution similar to that one Prawitz proposed in a similar problem (mentioned earlier in this paper), namely when he introduced dummies instead of choosing specific terms: delay the choice as long as possible, that is till the rows of the matrix are tested for complementary pairs. Unfortunately this becomes a very complicated algorithm. Therefore an algorithm is presented here as a first approach which solves the first but neglects the second problem and thus provides only a semi-systematic procedure. In the rest of this section this algorithm will be described.

For simplicity let us restrict the formal system by removing the function symbols from it. Later on the changes, necessary to include them again, will be indicated.

First the given formula is transformed into an antiprenex formula F according to the rules given in chapter 2. The main part then again consists of two subroutines, *ALZ* and *AX*.

ALZ transforms a matrix M of (possibly labelled) formulae into a matrix L of literals and a matrix K of existential formulae and establishes a tree ordering $\prec$ among the introduced variables and dummies. In the first step M initially consists of one element, the formula F.

1. Set $i=0$.

2. If i is less than the number of rows in M, set $i \leftarrow i+1$ and goto 3; otherwise exit to main program.

3. If row i in M is empty, goto 2.; otherwise take the leftmost formula G in it.

3.1 If G is a literal then remove it in M and add it to the i-th row in L.

3.2 If G is of the form $\neg \neg G_0$ then drop the two negation signs.

3.3 If G is of the form $(G_1 \vee G_2)$ and labelled with l then replace it by G_1, G_2 both labelled with l.

3.4 If G is of the form $\neg (G_1 \vee G_2)$ and labelled with l, then add a new row to M, which is a copy of row i, except G is replaced by $\neg G_2$ labelled with l, and replace in row i G by $\neg G_1$ labelled with l. Further copy row i in matrix L and K into a newly created row in L and K, respectively.

3.5 If G is of the form $V x \mathbf{F}[x]$ and labelled with l then let d be a new dummy and replace G by $\mathbf{F}[d]$ labelled with d; further note $l \prec d$, and add G labelled with d to the i-th row of K.

3.6 If G is of the form $\neg V x \mathbf{F}[x]$ and labelled with l then let a be a new variable and replace G by $\neg \mathbf{F}[a]$ labelled with a, and note $l \prec a$.

Then return to 3.

After exit from *ALZ* the subroutine *AX* is called to test whether the dummies can be replaced consistently by terms so that each row of the matrix L contains

a complementary pair. In that process the relation $\prec$ plays now an important role.

To illustrate this *I* mention that each derivation of *F* determines a unique relation $<$ among the eigenvariables and eigenterms, namely $e_1 < e_2$ iff the inference which e_1 belongs to lies below the inference which e_2 belongs to. $\prec$ is roughly spoken a first approach (a sub-relation) to such a relation $<$. Therefore *AX* has also to extend $\prec$ to such a relation $<$ which in case of success reflects part of the structure of the derivation of *F*. In detail the algorithm *AX* reads as follows.

1. Set $i \leftarrow 1$.

2. In row *i* find a pair of an unnegated and a negated literal which has not yet been considered. If there is none such pair, forget that in row *i* any pair has been considered already, cancel all tentative identifications originating from row *i*, and if $i > 1$ then set $i \leftarrow i-1$, otherwise return a failure message; otherwise check for that pair whether the predicate variable and the number of arguments are the same. If no, start again in 2., otherwise compare each pair of corresponding arguments in the following way.

a) If both are variables, but different, cancel all tentative identifications from row *i* and start in 2. again.

b) If one is a variable *a*, the other a dummy *d* and *d* has been identified already with a term different from *a*, or $d \prec a$ holds, then cancel all tentative identifications from row *i* and start over in 2.; otherwise tentatively identify *d* with *a*.

c) If both are dummies d_1 and d_2 with different identifications or d_1 is identified with *a* but for d_2 (or any other dummy identified with d_2) $d_2 \prec a$ or an analogue situation for d_2 holds, then cancel all tentative identifications from row *i* and start again in 2.; otherwise identify one with (the identification of) the other and vice versa.

3. If *i* is less than the number of rows then set $i \leftarrow i+1$ and goto 2.; otherwise return the message that the formula turned out to be valid in the current step of the procedure.

If *AX* returns a failure message the procedure will enter a further step, will call *ALZ* with *M* being initiated by the value of *K* which will transform it into extended matrices *L* and *K*, and will call *AX* again; and so forth.

Recall the example of section 3 to compare the operations of *TRF* and *CPT* with those of *ALZ* and *AX*; then it can be seen that now the unnecessary intermediate "no"-result disappears completely, due to the additional information given in the relation $\prec$ which after execution of *ALZ* with *F* still is empty. Therefore in *AX* 2. b the identification $d\,1 = a$ can be made immediately.

A more interesting example would be

$$\Lambda x \Lambda y \Lambda z \left(p(x, y) \rightarrow \neg p(y, z)\right) \rightarrow \mathrm{V} y \Lambda x \neg p(x, y).$$

The reader is invited to apply to it both procedures and find out that it is a theorem (in antiprenex form) of degree 2.

To conclude this section let us briefly indicate how the algorithm AX has to be changed if function symbols are allowed. a) and b) under 2. remain unchanged. In c) one has to include also that e. g. d_1 is identified already with a quasi-term (eventually still containing dummies) in which a variable a occurs for which $d_2 \prec a$ holds in which case also no identification can be performed. Three more cases have to be considered.

d) Variable — quasi-term, starting with a function symbol (fails always).

e) Both quasi-terms, starting with a function symbol (function symbols have to be the same; each argument has to be tested).

f) Dummy — quasi-term, starting with a function symbol (identify, whenever no conflict with relation $\prec$ arises).

5. Concluding Remarks

The algorithm described in chapter 4 has been coded as a SNOBOL4 program [1]. Figure 1 shows the output of a run of it for the formula

$$\Lambda x \Lambda y(((V z h(x,z) \rightarrow \Lambda z g(x,z)) \wedge \Lambda z(g(z,z) \rightarrow h(z,y))) \rightarrow (h(x,y) \leftrightarrow \Lambda z g(x,z)))$$

whose antiprenex form

$$\neg V x V y \neg (((\neg(\neg V z h(x,z) \vee \neg V z \neg g(x,z)) \vee V z \neg(\neg g(z,z) \vee h(z,y))) \vee \neg(\neg(\neg h(x,y) \vee \neg V z \neg g(x,z)) \vee \neg(V z \neg g(x,z) \vee h(x,y))))$$

is of degree 1 but which here is proved in the second step since the procedure is only semi-systematic, not systematic. (LIST OF CLAUSES is the resulting matrix after ALZ has been applied — only the labelled existential formulae are not printed; VARIABLES WITH INCOMPATIBLE DUMMIES indicates the relation $\prec$ and has to be read: $V1 \prec V2$, $V2 \prec V3$, $V1 \prec V3$, ...)

The test results are encouraging. But comparison of the size of the two matrices make obvious the advantage of a systematic procedure because it would avoid running into the second step. Therefore as mentioned before *I* actually regard the procedure of chapter 4 as a stop en route to a systematic and efficient procedure which would also take into account the second problem from chapter 4 using the same philosophy: postpone any choice until the subroutine AX where it can be made in an optimal way.

The procedure could then be further improved by taking advantage of the associative and distributive laws of the propositional operations. Since each of these problems has to be solved in *connection* with all others, the desired algorithm undoubtedly will be a very complicated but also a very efficient one. Moreover this method is qualified for extending the procedure such that after eventually identifying a formula as a theorem, it also provides an actual derivation of it.

---

[1] I wish to thank the Computer Science Section of the Mathematics Department of Wayne State University, Mich., for offering its facilities to complete this work.

For certain applications the degree d of a theorem might be too rough a measure since it does not discriminate among theorems of the same degree. But this could easily be improved for example by taking into account the length l of the formula, e.g. taking $\omega \cdot d + l$.

INPUT FORMULA:

AXAY(((EZ(H,X,Z)SAZ(G,X,Z))CAZ((G,Z,Z)S(H,Z,Y)))S((H,X,Y)BAZ(G,X,Z)))

EQUIVALENT TRANSFORMATION:
NEXEYND(D(ND(NEZ(H,X,Z),NEZN(G,X,Z)),EZND(N(G,Z,Z),(H,Z,Y))),ND(ND(N(H,X,Y),
NEZN(G,X,Z)),ND(EZN(G,X,Z),(H,X,Y))))

LIST OF CLAUSES:
(H,V1,D1)(G,D2,D2)N(H,V1,V2)(G,V1,V3)
N(G,V1,D3)(G,D4,D4)N(H,V1,V2)(G,V1,V4)
(H,V1,D1)N(H,D2,V2)N(H,V1,V2)(G,V1,V5)
(H,V1,D1)(G,D2,D2)N(G,V1,D5)(H,V1,V2)
N(G,V1,D3)N(H,D4,V2)N(H,V1,V2)(G,V1,V6)
N(G,V1,D3)(G,D4,D4)N(G,V1,D6)(H,V1,V2)
(H,V1,D1)N(H,D2,V2)N(G,V1,D7)(H,V1,V2)
N(G,V1,D3)N(H,D4,V2)N(G,V1,D8)(H,V1,V2)

VARIABLES WITH INCOMPATIBLE DUMMIES:
V1
V2 V1
V3 V2
V4 V2
V5 V2
V6 V2

AFTER 1TH STEP OF PROCEDURE NO DECISION ABOUT CURRENT FORMULA

LIST OF CLAUSES:
(H,V1,D1)(G,D2,D2)N(H,V1,V2)(G,V1,V3)(H,V1,D9)(G,D10,D10)
N(G,V1,D3)(G,D4,D4)N(H,V1,V2)(G,V1,V4)N(G,V1,D11)(G,D12,D12)
(H,V1,D1)N(H,D2,V2)N(H,V1,V2)(G,V1,V5)(H,V1,D13)(G,D14,D14)
(H,V1,D1)(G,D2,D2)N(G,V1,D5)(H,V1,V2)(H,V1,D15)(G,D16,D16)N(G,V1,D17)
N(G,V1,D3)N(H,D4,V2)N(H,V1,V2)(G,V1,V6)N(G,V1,D18)(G,D19,D19)
N(G,V1,D3)(G,D4,D4)N(G,V1,D6)(H,V1,V2)N(G,V1,D20)(G,D21,D21)N(G,V1,D22)
(H,V1,D1)N(H,D2,V2)N(G,V1,D7)(H,V1,V2)(H,V1,D23)(G,D24,D24)N(G,V1,D25)
N(G,V1,D3)N(H,D4,V2)N(G,V1,D8)(H,V1,V2)N(G,V1,D26)(G,D27,D27)N(G,V1,D28)
(H,V1,D1)(G,D2,D2)N(H,V1,V2)(G,V1,V3)(H,V1,D9)N(H,D10,V2)
N(G,V1,D3)(G,D4,D4)N(H,V1,V2)(G,V1,V4)N(G,V1,D11)N(H,D12,V2)
(H,V1,D1)N(H,D2,V2)N(H,V1,V2)(G,V1,V5)(H,V1,D13)N(H,D14,V2)
(H,V1,D1)(G,D2,D2)N(G,V1,D5)(H,V1,V2)(H,V1,D15)N(H,D16,V2)N(G,V1,D29)
N(G,V1,D3)N(H,D4,V2)N(H,V1,V2)(G,V1,V6)N(G,V1,D18)N(H,D19,V2)
N(G,V1,D3)(G,D4,D4)N(G,V1,D6)(H,V1,V2)N(G,V1,D20)N(H,D21,V2)N(G,V1,D30)
(H,V1,D1)N(H,D2,V2)N(G,V1,D7)(H,V1,V2)(H,V1,D23)N(H,D24,V2)N(G,V1,D31)
N(G,V1,D3)N(H,D4,V2)N(G,V1,D8)(H,V1,V2)N(G,V1,D26)N(H,D27,V2)N(G,V1,D32)

VARIABLES WITH INCOMPATIBLE DUMMIES:
IN 2TH STEP OF PROCEDURE CURRENT FORMULA TURNS OUT TO BE VALID

Fig. 1

References

[1] Bibel, W.: Schnittelimination in einem Teilsystem der einfachen Typenlogik. Archiv f. Math. Logik **12,** 159—178 (1969).
[2] Ernst, G. W.: The utility of independent subgoals in theorem proving. Inform. and Contr. **18,** 237—252 (1971).
[3] Prawitz, D.: An improved proof procedure. Theoria **26,** 102—139 (1960).
[4] Prawitz, D.: A proof procedure with matrix reduction. In: Symposium on Atomic Demonstration, 1968. (Lect. notes in Mathem., vol. 125.) Berlin-Heidelberg-New York: Springer. 1970.
[5] Robinson, J. A.: A review of automatic theorem-proving. Proc. Symp. Appl. Math., Amer. Math. Soc. **19,** 1966.
[6] Schütte, K.: Beweistheorie. Berlin: 1960.
[7] Wang, H.: Toward mechanical mathematics. IBM Journal **1960,** 2—22.
[8] Nilsson, N. J.: Problem-solving methods in artificial intelligence. New York: 1971.

Dr. Wolfgang Bibel
Mathematisches Institut
der Technischen Universität
Arcisstraße 21
D-8000 München 2
Bundesrepublik Deutschland