A DEFINITION-DRIVEN THEOREM PROVER

George W. Ernst
Department of Computing and Information Sciences
Case Western Reserve University
Cleveland, Ohio 44106

Abstract

This paper describes a theorem prover, running on a PDP-10-TENEX system, that can prove some theorems whose statements Involve a relatively large number of definitions. Such theorems require special methods because (1) their statements contain a large number of clauses and (2) their proofs are quite long although straight-forward.

A theorem is proven by first subdividing it into "simple" subgoals and then using a standard resolution theorem prover to prove the subgoals. The first part of this process Involves the substitution of definitions for defined quantities and the use of logical simplification. This process which is more similar to a natural deduction system than a resolution system, is shown to be complete when restricted to first-order logic. However, the theorem prover can deal with some interesting higher-order theorems as is shown by an example.

1. Introduction

The motivation for this research stems from our view of how theorem provers will be used in solving real-life problems. We believe that the first few applications of theorem provers will be man-machine oriented because, at the current state of the art, theorem provers can only solve relatively easy problems and 1t will probably be quite a while before they can solve truly difficult problems on their own.

It also seems to us that theorem provers will be faced with a large number of problem-oriented predicates due to diversity of real world data. Some of these predicates will be sufficiently basic that they will of necessity be built into the theorem prover. However, most of them, due to their abundance, will have to be defined 1n terms of the basic predicates and logical connectives.

On the basis of these assumptions, we feel that the role of the machine will be primarily a big data filter, doing the relatively straight-forward part of the theorem proving which might be quite lengthy. For the most part, this amounts to "wading" through definitions and previously proven theorems (and perhaps meta-theorems), many of which are quite long and tedious; proving the easy parts and isolating the non-trivial parts. The man then would select a non-trivial part to work on next and would give the machine "hints" on how to go about proving it. The hints will be of various forms, such as conjecturing induction hypothesis, lemmas, the value of important variables, etc.

This paper describes a theorem prover, TPI, that can prove some lengthy but straightforward theorems whose statement contains defined quantities. Although TPI 1s a totally mechanical, and not a man-machine theorem prover, the types of problems for which 1t is designed are similar to what we feel will be the kind of problems that a machine should do 1n a man-machine, application of theorem proving. Since the approach of TPI 1s somewhat different than a standard resolution theorem prover, 1n the next section we will give an example of how 1t proved a particular theorem. This will motivate Sec. 3 which describes TPI 1n fair detail.

2. An Example of TPI

In this section we give an example of how TPI proved the theorem that the direct product of 2 groups 1s a group. The statement of this theorem 1s*

$$\begin{array}{l}\forall((G\ univ)\ (+\ (FN\ G\ G\ G))\ (-\ (FN\ G\ G))\ (0\ G)\\ (H\ univ)\ (*\ (FN\ H\ H\ H))\ (I\ (FN\ H\ H))\ (E\ H)) \qquad (1)\\ [Gr(G,+,-,0)\ \&\ Gr(H,*,I,E) \rightarrow Gr(GXH,\circ,J,(0,E))]\end{array}$$

The operator and inverse of the product group are ° and J, respectively, and are defined below. In the first line of (1) (G univ) Indicates that G is some element of the universe,and (+(FNG G G)) indicates that + is a binary operator on 6. The remainder of the first line indicates that -:G → G and 0 1s some element of G. The second line of (1) states the same things about H,*,I and E. Thus the first two lines universally quantify 8 variables subject to the restrictions that they must lie 1n certain domains. Gr 1s a defined predicate having 4 arguments, a group, Its operator, its Inverse and Its Identity, respectively. Thus, the last line of (1) states that, If G and H are groups then so is GXH (cartesian product of G and H). The operator ° of the latter 1s

$$\lambda((u\ GXH)(v\ GXH))(1^{st}(u)+1^{st}(v),2^{nd}(u)*2^{nd}(v)) \quad (2)$$

In this expression λ is the usual operator of abstraction. The meaning of (2) 1s that 1t 1s a two-place function. The string (u GXH) Indicates that the first argument which gets substituted for u must be 1n the set GXH. 1 (u) is the first element of the ordered pair u while 2nd(u) is the second element of u. The . 1n (2) 1s the operator which takes two terms and makes an ordered pair out of them. An example of the use of (2) is a°b. This expression 1s defined 1f both a and b are 1n GXH and the term a°b evaluates to the ordered pair $(1^{st}(a)+1^{st}(b),2^{nd}(a)2^{nd}(b))$. Note that G, H, +, * are free variables 1n (2) which are quantified by the quantifiers 1n (1) because ° occurs in (1).

The Inverse J of the product group 1s defined 1n a manner similar to (2) as

$$\lambda((u\ GXH))(-(1^{st}(u)),\ I(2^{nd}(u))). \quad (3)$$

The Identity of the product group 1s given 1n (1) as the ordered pair (0,E).

To prove {1)» TPI must know the definition of group, 1^{st}, etc. TPI has a repertoire of definitions and previously proven theorems which includes these definitions and other definitions, such as definition of homomorphlsm, that are not relevant to proving (1). We will not state the definitions because their content will become clear by their use.

First of all, TPI simplifies (1) which 1n this

•There are 3 differences between (1) and the formula given to TPI: An example of the first Is that Gr(G,+,-,0) 1s given to the machine as (Gr G + - 0). The second 1s that infix notation 1s used 1n (1) and thirdly parentheses are omitted in (1) if their omission causes no ambiguity.

case Involves putting Skolem functions 1n for the universally quantified variables in (1). For notational purposes we will use the same names as the variables and thus (1) becomes

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow Gr(GXH,\circ,J,(0,E)). \quad (4)$$

In this formula G, +, etc. are constants and their domains are known because of the first 2 lines of (1), e.g., + Is a binary operator on G. Next, TPI sees from the definition of group that 1t is sufficient to prove that o 1s associative; J is a right inverse; and (0,E) is a right identity. The resulting formula is immediately simplified and stated as the following three subgoals (or lemmas) which TPI will work on separately:

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow (1^{st}(a)+1^{st}(0,E),\ 2^{nd}(a)*2^{nd}(0,E))=a \quad (5)$$

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow [1^{st}(b)+1^{st}(-(1^{st}(b)),\ I(2^{nd}(b))),\ 2^{nd}(b)*2^{nd}(-(1^{st}(b)),I(2^{nd}(b)))] = (0,E) \quad (6)$$

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow [1^{st}(1^{st}(c)+1^{st}(d),2^{nd}(c)*2^{nd}(d))+1^{st}(e),\ 2^{nd}(1^{st}(c)+1^{st}(d),2^{nd}(c)*2^{nd}(d))*2^{nd}(e)] = [1^{st}(c)+1^{st}(1^{st}(d)+1^{st}(e),2^{nd}(d)*2^{nd}(e)),\ 2^{nd}(c)*2^{nd}(1^{st}(d)+1^{st}(e),2^{nd}(d)*2^{nd}(e))] \quad (7)$$

In these formulae a, b, c, d, e are Skolem functions that are elements of GXH. (5), (6), (7) state respectively that (0,E) is a right identity, J is a right inverse and ° Is associative. The reader will note that (5), (6) and (7) are more complicated than textbook formulae because they contain terms that can obviously be simplified such as lst(0,E) which is just 0. However, these formulae were produced directly from the definition of a group and such simplification will occur later in the proof.

TPI next processes (5) 1n much the same way that it processed (1). It sees from the definition of an ordered pair that two ordered pairs are equal if their first components are equal and if their second components are equal. Using this definition on (5), the following two new subgoals are produced:

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow 1^{st}(a)+1^{st}(0,E)=1^{st}(a) \quad (8)$$

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow 2^{nd}(a)+2^{nd}(0,E)=2^{nd}(a) \quad (9)$$

When TPI processes (8) it notes that (8) cannot be further divided into subgoals, and thus calls on a resolution-based theorem prover called the subgoal solver (SGS), to solve 1t. But first 1t replaces defined quantities by their definitions and then reduces the formula to normal form. Fig. 1 gives the result of processing (8) 1n this way. In Fig.1 the variable x is restricted to be 1n G. This information 1s checked implicitly in TPI whereas a standard resolution theorem prover would append an $x \notin G$ to each clause containing an x, e.g., the first clause in Fig. 1 would be $x \notin G$ V x+0=x. Similarly y 1s restricted to be 1n H while u and v are not restricted.

The definition of a group Indicates that + and * are associative. Instead of representing this Information as clauses, the operators are "flagged" and the SGS processes them as n-ary operators.

As the reader can easily see, F1g. 1 gives a contradiction in just two steps. From clauses 9 and 12 in F1g. 1 we get lst(a)+0={a). From this and clause 1 In F1g. 1 we get 1st{a)^ist(a) which 1s contradiction.

TPI processes (9) almost identically to the way in which 1t processed (8). The normal form .of (9) 1s the first 11 clauses In Fig. 1 and 2nd(a)+*2nd(O,E)=2nd(n)* which also has a two-step contradiction.

TPI then proceeds to (6) and uses the definition of ordered pair to divide (6) and Into the following two subgoals:

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow 1^{st}(b)+1^{st}(-(1^{st}(b)),I(2^{nd}(b)))=1^{st}(0,E) \quad (10)$$

$$Gr(G,+,-,0) \;\&\; Gr(H,*,I,E) \rightarrow 2^{nd}(b)*2^{nd}(-(1^{st}(b)),I(2^{nd}(b)))=2^{nd}(0,E) \quad (11)$$

(10) equates the first components of the left and right sides of the = in (6) while (11) equates their 2nd components.

Since (10) cannot be divided into subgoals, it is reduced to normal form, which is just the first 11 clauses in F1g. 1 and

$$1^{st}(b)+1^{st}(-(1^{st}(b)),I(2^{nd}(b)))\neq 1^{st}(0,E) \quad (12)$$

This is passed to the SGS which easily solves 1t in 3 steps. From (12) and clause 9 1n Fig. 1, lst(b)+-(lst(b))=lst(o,E) 1s deduced. This and clause 3 1n F1g. 1 gives O=1st(0,E) which by clause 9 in Fig. 1 implies 0=0 which is a contradiction.

TPI processes (11) in much the same way as (10) proving it also 1n three steps. Proceeding onto (7), TPI breaks it Into 2 subgoals by separately equating the two components of the left and right of the = 1n (7). Since the generation of these subgoals is so similar to the generation of the above subgoals we will not state the subgoals 1n this paper. Also their proofs will be omitted because they are just 3 steps apiece.

This concludes the proof of (1). In summary then, TPI divides (1) into 6 terminal subgoals or lemmas and proves these 6 subgoals independently which constitutes a proof of (1). The purpose (5), (6) and (7), which are intermediate subgoals, is to facilitate the generation of the 6 terminal subgoals; i.e., (5), (6) and (7) are not directly proven by the SGS.

3. Description of TPI

The purpose of this section 1s to describe the internal mechanisms of TPI which is a LISP program currently running on a PDP-10 under TENEX. Due to space limitations, the description will be *very* brief. For this reason the Interested reader 1s referred to ! which is a fairly detailed description of TPI.

TPI operates from a repertoire of definitions not all of which are needed in any particular theorem. Currently, there are over a dozen defined quantities 1n TPI's repertoire. Incorporated into the definitions are also some previously proven theorems. All of this Information Is contained on two lists BD (backward definitions) and FD (forward definitions).

All statements on BD are of the form Q(A+B) while FD statements are of the form q(B=C). B 1s the defined quantity and Q 1s a string of restricted quantifiers. A together with C comprises the definition of B. In addition, certain previously proven theorems about B may also be Incorporated into these statements as described 1n Sec. 4. As an example, consider the definition of a group on BD. In this case B 1s Gr(G,+,-,0) and A 1s

$$\forall((x\ G)(y\ G)\ (z\ G)[x+0=x\ \&\ x+(-x)=0\ \&\ (x+y)+z=x+(y+z)] \qquad (13)$$

and Q is ∀((G univ)(+(FN G G G))(-(FN G G))(0 G)). The intent of these formulae should be clear from the discussion in the previous section. FN and univ are primitives whose semantics are built into TPI. Although the relationship between A and B in this example is 'if and only if', in general, A will only imply B for the statements on BD.

Since the relationship is 'if and only if' we could use the same statement on FD, i.e., we could put Q(B≡A) on FD. Note that our use of restricted quantification is somewhat non-standard. The restrictions are processed as if they are conjuncts of the second argument of ≡. For example, ∀((x a)(y b))E≡F is taken to mean ∀xy(E ≡ x∈a & y∈b & F).

TPI attempts to prove theorems by generating an AND/OR tree of subgoals in much the same way as described in [2]. The root of the tree is the theorem to be proven. TPI processes a node in the tree by first applying some logical simplifications such as "λ-conversion," transforming a theorem of the form D→(E→F) into D&E→F, and removing quantifiers at the outermost level by putting in Skolem functions. (Note that, since the theorem has not been negated, the Skolem functions are put in for universally quantified variables instead of existentially quantified variables.) Then, if the simplified theorem is of the form $E\rightarrow(F_1\&F_2\&\ldots)$, the subgoals $E\rightarrow F_1$, $E\rightarrow F_2$... are generated provided that the F_i have no variables in common.

Next for each subgoal of the form $G\rightarrow F_i$, TPI generates the subgoals $G\rightarrow\sigma_{ij}A_{ij}$ where $Q_{ij}(A_{ij}\rightarrow B_{ij})$ are statements on BD and σ_{ij} is the most general unifier of B_{ij} and F_i. These are subnodes of an OR node while the above subgoals come from an AND node. Fig. 2 shows the subgoal tree generated so far. Now the whole process is repeated for the subgoals: logical simplification, generation of AND subnodes, generation of OR subnodes via BD.

Actually this picture is highly simplified. The most notable omission is that TPI checks that the restricted quantification of Q_{ij} is satisfied. Experience has shown that most of the variables are known to be in the right domain (this is always the case in the example in Sec. 2) but for the others separate subgoals must be generated to prove that they are in the correct domain. Examples of such subgoals are given in Sec. 5.

TPI attempts to prove the terminal nodes in the subgoal tree as follows: Each occurrence of a B in the subgoal is replaced by σC where Q(B'≡C) is an element of the FD list and σ is the most general unifier of B and B'. The restrictions on the variables specified by Q are checked and appropriate literals are added to the subgoals when the restrictions are not already known to be satisfied. The replacement of defined quantities, as specified by FD, is a recursive process since the definition of a quantity may contain other defined quantities. In addition, some defined quantities cannot be removed and the definitions of such quantities are added to the subgoal. For example, the definition of ordered pair was added to the subgoals of the example of Sec. 2 because the , operator could not be removed.

The above process is continued until all pertinent definitions have been incorporated into the subgoal. Then it is converted into clause form which is passed to a resolution-based theorem prover, called the subgoal solver (SGS), for proof. Note that the SGS never refers to FD and BD. Instead TPI, selects, via generating the tree of subgoals, the relevant elements of FD and BD to be incorporated into the terminal subgoals before they are passed to the SGS. This is important because most of FD and BD are not used in proving any one theorem.

It is quite easy to see the conditions under which the TPI is complete. Suppose that for each defined quantity B, there is an entry on both BD and FD and one of the entries $Q_i(A_i\rightarrow B)$ on BD has the property that $Q_i(B\rightarrow A_i)$ is also true. Then some subtree of the AND/OR subgoal tree will be equivalent to the original theorem, in the sense that it is valid if and only if all of the terminal nodes in the subtree are valid. The reason is that each OR node has a subnode which is equivalent to it by the above assumption. Since the elements on FD all have ≡ as a main connective, the clause form of a terminal node which is passed to the SGS, is equivalent to the node.

The above is a brief argument showing that TPI is complete if the SGS is complete and the unification algorithm is complete. Both of these conditions are violated by the current version of TPI. The SGS does not process equality in a complete way -- a difficulty which can easily be removed. The difficulty with the unification algorithm can only be removed by removing all higher-order variables because there is no unification algorithm for higher-order logic.[4] Although TPI is not complete, its underlying approach (i.e., an AND-OR tree of subgoals) is complete. We feel it is important that we know in what ways TPI is incomplete and what are the difficulties in the way of making TPI complete.

4. Formulation of BD and FD

This section discusses certain pragmatic considerations in the formulating of FD and BD.

Consider modifying the definition of group in Sec. 3 by "anding" right cancellation to (13). Would this make TPI's job easier or harder? If the definition on BD were modified this way TPI would always have to do more work. The reason is that BD is only used when trying to prove that something is a group, i.e., when Gr(...) is the consequent of some subgoal. It would be ridiculous to try to prove right cancellation in addition to the 3 conjuncts of (13) since the latter are sufficient. Thus, a statement on BD should be "minimal" in the sense that removing any conjunct from its antecedent causes the statement to become invalid.

Adding right cancellation to the definition of group on FD may be quite useful because this statement is used when TPI is "given" that something is a group. Thus, this modification will allow the SGS of TPI to use right cancellation of the given group in proving a subgoal. In general, for any definition Q(B≡C) on FD, a previously proven theorem of the form Q(B→D) can be incorporated into FD by using the new definition Q(B≡C&D) on FD. Note that the logical connective of this new definition is still ≡. Of course, if too many previously proven theorems are incorporated into a definition on BD, TPI may become bogged down. But adding a few carefully selected previously proven theorems can be very beneficial.

Another beneficial way to express things is to remove certain existential quantifiers from definitions. Although quantifiers cannot be "moved across" an ≡ (which statements on FD contain) extra parameters can be added to definitions which allow the existential quantifiers to be removed. For example, suppose we defined group as a predicate with two arguments: set and operator. Then the definiens would contain "there exists an identity and an inverse such that..." Making the identity and the inverse arguments to the predicate, as in Sec. 3, removes these two existential quantifiers. This formulation allows one to give TPI identities and inverses in the statement of theorems as in Sec. 2. But more importantly, it allows other definitions to refer to these quantities. For example, the predicate subgroup can be defined by

$$Sg(S,G,+,-,0) \equiv Gr(G,+,-,0) \;\&\; \forall(x\ S)(y\ S)\ (-x\varepsilon S \;\&\; x+y\varepsilon S) \quad (14)$$

If group were a 2-place predicate this definition would not be possible.

5. An Example with More Complex Higher-Order Terms

Although TPI processes higher-order variables incompletely, it can still handle the example, below, which is to prove that the set of homomorphisms from one abelian group to another is a group. For readability quantifiers and domains of quantified variables will be omittted from the formulae below. Rather we will note in the text the domain of variables and constants whenever this information is important.

The theorem is

$$Ab(G,+,-,0) \;\&\; Ab(H,*,I,E) \rightarrow Gr(homo(G,H,+,*), \alpha,\beta,\gamma) \quad (15)$$

where

$$\alpha \text{ is } \lambda(uv)(\lambda x(u(x)*v(x))) \quad (16)$$

$$\beta \text{ is } \lambda u(\lambda x(I(u(x)))) \quad (17)$$

$$\gamma \text{ is } \lambda uE \quad (18)$$

In the above, u and v are homomorphisms, $x\varepsilon G$, and Ab is the predicate stating its first argument is an abelian group with respect to its other 3 arguments. Similarly Gr is the group predicate (see Section 2).

The definition of homomorphism (on both FD and BD) states

$$f\varepsilon\, homo(G,H,+,*) \equiv \forall((x\ G)(y\ G))f(x+y)=f(x)*f(y) \quad (19)$$

where $f:G \rightarrow H$.

In (19) we have used for arguments of homo, the same names as used in (15) for readability, but in BD and FD the arguments of homo are universally quantified.

The theorem (15) is eventually broken into the 6 terminal subgoals shown in Fig. 3. Space only permits us to describe the highlights of the generation of these subgoals.

The generation of 2 in Fig. 3 is typical of the generation of the first 3 subgoals. From the definition of group we get the AND subnode whose consequent is

$$\alpha(f,\beta(f))=\gamma \quad (20)$$

where u is any homomorphism. This just states that β is the inverse, and by "λ-conversion" immediately simplifies to

$$\lambda x(f(x)*I(f(x)))=\gamma \quad (21)$$

Both the left and the right of (21) are functions from G to H; i.e., their domain are (FN G H). As mentioned in Sec. 3, FN is a primitive built into TPI. To prove that two functions are equal TPI invokes functional extensionality,

$$(\forall x f(x)=g(x)) \rightarrow f=g, \quad (22)$$

an axiom built into TPI. This produces the subgoal

$$(\lambda x(f(x)*I(f(x))))(a)=\gamma(a) \quad (23)$$

where a is a constant in G. (23) immediately simplifies to $f(a)*I(f(a))=E$.

Subgoals 4, 5, and 6 in Fig. 3 are generated because the terms substituted for the variables in the definition of a group are not known to be in the required domain. We will describe the generation of subgoal 5, which is typical of the other two. The domain of β is required to be a function from homomorphisms to homomorphisms according to the definition of a group on BD. However, according to (17), TPI only knows that the domain of β is (FN(FN G H)(FN G H).Thus the subgoal $\beta\varepsilon$(FN homo(G,H,+,*)homo(G,H,+,*)) is generated. TPI uses the built-in axiom, $\forall x(x\varepsilon A \rightarrow f(x)\varepsilon B) \rightarrow f\varepsilon(FN\ A\ B)$, to generate the subgoal,

$$\beta(f)\varepsilon homo(G,H,+,*), \quad (24)$$

where f is a constant homomorphism. Simplifying (24) with "λ-conversion" we get

$$\lambda x(I(f(x)))\varepsilon homo(G,H,+,*) \quad (25)$$

Replacing homo by its definition and simplifying we get subgoal 5 in Fig. 3.

Currently, TPI cannot prove (15) because the commutativity axioms give rise to too many resolvants at each step. Although this is no problem for the first 4 subgoals in Fig. 3, it prevents the SGS from proving 5 and 6. We have included this example because the emphasis in this work has been on the generation of simple subgoals which TPI does in this example. We believe that existing resolution theorem provers can prove 5 and 6 in Fig. 3 and thus if TPI had a more sophisticated SGS, it could prove (15).

6. Conclusion

Most contemporary theorem provers have difficulty proving a theorem whose proof is relatively long or whose statement is relatively long (i.e., large number of clauses). The goal of our research is to develop a theorem prover which is somewhat insensitive to the length of the proof and the length of the statement of the theorem.

The three main methods In TPI for achieving this goal are: (1) dividing the theorem into independent subgoals, (2) the special treatment of definitions, and (3) Implicit use of restricted quantification. The purpose of (1) is to make TPI sensitive to the length of the proof of the "longest" subgoal as opposed to the length of the total theorem. Empirical results Indicate that (1) achieves this purpose quite well.

The main purpose of (2) is to disregard those definitions that are not necessary for the proving theorem. Currently, TPI is given 14 definitions and no theorem has used more than 4. Thus, the extraneous definitions are considerably longer than the theorem and the pertinent definitions. The processing of definitions also Interfaces *very* nicely with (1), allowing TPI to subdivide subgoals into Independent subgoals.

The purpose of (3) is to give high priority to inferences which prove that variables are in the correct *domain.* This heuristic allows TPI to discover longer proofs because such Inferences are done Implicitly. Of course, the price paid for this ability is an Increase in the amount of time for each explicit Inference.

The example 1n Sec. 2 shows that TPI has achieved, to some extent, Its goal. The statement of the theorem given to TPI (including extraneous definitions) is 48 clauses, and the shortest proof that we know of is 27 binary resolutions.

The methods of TPI form a complete system of first-order logic. Although TPI cannot process higher-order terms 1n a complete way, its methods are sufficient for some interesting theorems containing higher-terms. Initial results indicate that the use of lambda notation and λ-normal-form are very powerful theorem-proving techniques.

Other theorem provers have used methods which are quite similar to those of TPI. Subdividing a theorem into parts has been used by Bledsoe,[1] Nivens,[5] Norton,[6] and Slagle and Koniver,[7] and probably others. Norton'

uses definitions In a way which 1s very similar to TPI and Bledsoe'sl use of definitions 1s somewhat similar. Although the overall characteristics of all these theorem provers (Including TPI) are quite different, they all proved some relatively impressive theorems. In particular, each can prove theorems that the others cannot prove. However, this author believes that 1n each case much of the power stems from the same few basic principles even though the methods of these theorem provers were independently developed along quite different lines.

Acknowledgments

This research was supported by the National Science Foundation under grant GJ-1135, and the Air Force Office of Scientific Research under grant AFOSR-71-211OC.

References

1. Bledsoe, W. W., "Splitting and Reduction Heuristics in Automatic Theorem proving," Artificial Intelligence 2, 1971, pp. 55-77.

2. Ernst, G. W., "The Utility of Independent Subgoals in Theorem Proving," Information and Control, April, 1971.

3. Ernst, G. W, "A Definition-Driven Theorem Prover," Report No. 1124, Computing and Information Sciences, Case Western Reserve University, 1973.

4. Huet, G. P., "The Undecidabillty of the Existence of a Unifying Substitution in the Simple Theory of Types," Report 1120, Jennings Computing Center, Case Western Reserve University, 1972.

5. Nevins, A, J., "A Human Oriented Logic for Automatic Theorem Proving," Al Memo No. 268, Mass. Inst, of Tech., Cambridge, Mass., 1972.

6. Norton, L. M., "Adept; A Heuristic Program for Proving Theorems of Group Theory," Report MAC-TR-33, Mass. Inst, of Tech., Cambridge, Mass., 1966.

7. Slagle, J. R., and Konlver, D., "Finding Resolution Graphs and Using Duplicate Goals 1n AND/OR Trees," Information Sciences, Vol. 3, No. 4, Oct., 197., pp. 315-342.

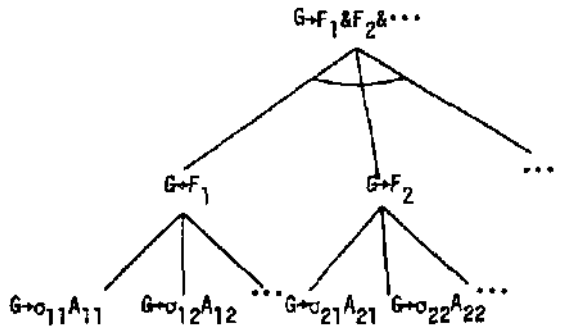

Figure 2 The General Form of the AND/OR Tree Generated by TPI

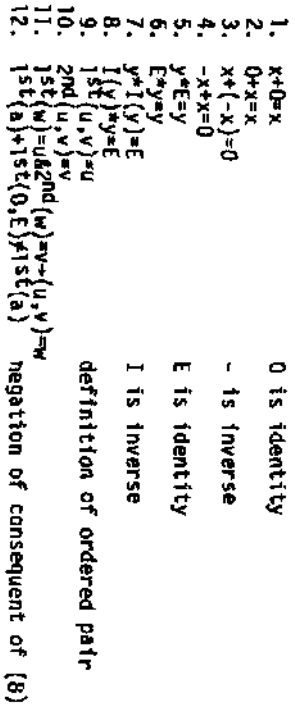

Figure 1 Subgoal (8) after negation and reduction to normal form. The domain of x is G and the domain of y is H whereas u, v and w are not restricted.

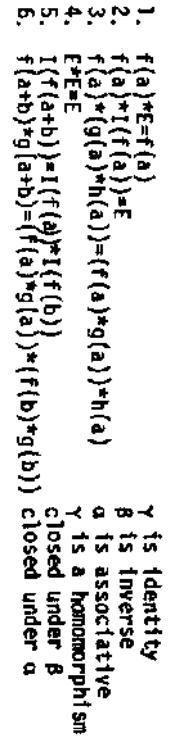

Figure 3 The consequents of 6 terminal sugboals whose conjunction 1s equivalent to (15). All of the symbols 1n this figure are constants, f, g and h are homomorphlsms while a and b are elements of G.