SET VARIABLES

W.W. Bledsoe
The University of Texas, Austin 78712

ABSTRACT: A procedure is described that gives values to set variables in automatic theorem proving. The result is that a theorem is thereby reduced to first order logic, which is often much easier to prove. This procedure handles a part of higher order logic, a small but important part. It is not as general as the methods of Huet, Andrews, Pietrzykowski, and Haynes and Henschen, but it seems to be much faster when it applies. It is more in the spirit of J.L. Darlington's F-Matching. This procedure is not domain specific: results have been obtained In intermediate analysis (the intermediate value theorem), topology, logic, and program verification (finding internal assertions). This method is a "maximal method" in that a largest (or maximal) set is usually produced if there is one. A preliminary version has been programmed for the computer and run to prove several theorems.

1. Introduction

For many theorems the main difficulty in the proof is in defining a particular set. Once that is done the proof often proceeds rather easily. For example, in

Theorem. $\exists A\ \forall x(x \in A \rightarrow x \leq 0)$.

If we let $A = \{x: x \leq 0\}$, then we are left with the trivial subgoal:

$$(x \leq 0 \rightarrow x \leq 0).$$

Or, if we are proving the intermediate value theorem,

Theorem. If f is continuous for $a \leq x \leq b$, $f(a) \leq 0$, and $f(b) \geq 0$, then $f(x) = 0$ for some x between a and b. (See Figure 1)

Using the least upper bound axiom,

LUB Axiom. Each non-empty bounded set A of real numbers has a least upper bound.

And if we let

$$A = \{x: f(x) \leq 0 \wedge x \leq b\},$$

then again the proof is rather straightforward (but harder than the last example). The question of course, is how to select A?

*This work was supported by National Science Foundation Grant MCS 76-23760.

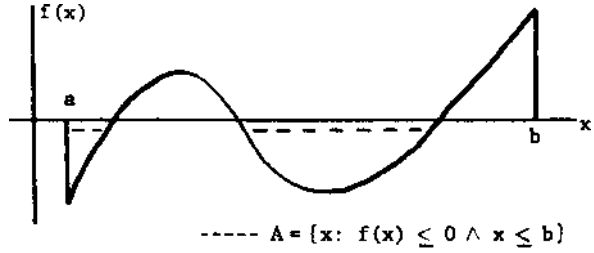


Figure 1

The Intermediate Value Theorem

There are several other theorems in analysis, such as the Heine-Borel Theorem where the chief difficulty lies in defining a particular set. Also a similar situation comes up again and again in other parts of mathematics, and in application areas such as program verification and program synthesis.

The problem of finding a value for a set variable A, is of course equivalent to the problem of giving a value to a one place predicate variable P.

This is a part of higher order logic, and as such can be attacked by the systems and ideas of Huet [3], Pietrzykowski [10], Haynes and Henschen [7], Andrews [11], etc. But these are very slow for many simple proofs. For example, Huet's beautiful system [3] is forced into double splitting on the rather easy theorem given in Example 4 below. (Even a human has trouble applying his procedure to this example.)

In this paper we describe a procedure which attempts to overcome this difficulty. It is less general than those referred to above; it usually applies only to a part of second order logic (but an important part); and it seems to be must faster when it applies. Ours is more in the spirit of J. L. Darlington's "F-Matching", but different in method and scope.

Our methods are not domain specific, not just a collection of heuristics for finding sets in a particular area like analysis. They can be used to prove theorems (such as the intermediate value theorem) in analysis where the set A is a set of real numbers, as well as theorems in topology where the set A is a family of sets, and theorems from program correctness, or from other areas where set variables are to be instantiated.

In Section 2 we give some preliminary examples and in Section 3 we describe our rules for generating the desired set A. They consist of basic rules which apply to simple formulas and combining rules for combining the results from the basic rules.

One of our goals in this work is to avoid indescrimlnant matches (or attempts at matches) between formulas such as (t e A) and P, (where

P is first order), but rather to allow such a match only when (t ∈ A) and P are somehow "connected". In this way the search is drastically reduced. Our basic rules (see Figure 2) are a partial attainment of this goal.

Our methods are "maximal" in that they usually generate the largest set with the desired properties (if there is one). Of course some theorems such as,

$$\exists A[(A \text{ is dense in } R) \wedge ((R-A) \text{ is dense in } R)]$$

has no maximal (nor minimal) solution for A. Also in some cases there is more than one maximal solution (see Example 5, Section 3), even infinitely many solutions. However, we believe there is a wide class of interesting cases that have a unique maximal solution, or at most one or two maximal solutions.

Our procedure utilizes the automatic prover described in [1] as a "control" (see Section 4) for generating the desired sets. But one can follow this presentation without full knowledge of that paper.

In actual practice the prover makes two passes on a theorem: first to define the set A; second to prove the resulting theorem, after the set A has been instantiated. Thus the procedure is sound i.e. there is no danger of it producing a false solution since the solution is always verified.

In Section 5 we give excerpts from some major examples which are described in more detail in [0]. In Section 6 we make several comments.

These methods do not satisfactorily handle the induction axiom. This is discussed in Section 6 of [0]. Some completeness proofs for these methods are given in Appendix II of [0].

2. Some Preliminary Examples

We are concerned here only with cases of the form

$$\exists A\ P(A) \quad \text{or} \quad ((\forall A\ P(A)) \rightarrow C)^*$$

where the variable A is to be found (i.e., given a value). Since ((VAP(A)) > C) is equivalent to 3A(P(A) > C), we will usually act as if our theorem is already in the form 3 A P(A). Theorems where A is a constant can usually be handled as first order logic.

We will treat only the case where there is but one such set variable A. However our methods will often work for theorems with many such variables.

Once the set A has been found, the theorem becomes first order, and hopefully not too difficult. That will often be the case. However, it is well known that automatic theorem proving for first order logic is a most challenging and unresolved task.

It is our objective here to automatically define such variables.

Let us look at some examples before giving the rules. These fragments help illuminate the procedure. More substantial theorems will be given later in Section 5.

* Provided that A does not occur in C.

Example 1. $\exists A \forall x (x \in A \rightarrow x \leq 0)$.

Solution: $A = [x: x \leq 0]$.

This is the simple theorem which states that "there is a set A for which, for each x in A, x is non-positive". Clearly, one such A is the set of all non-positive reals, and that is exactly what is returned by our program. When this value is substituted for A the theorem reduces to the trivial subgoal $(x \leq 0 \rightarrow x \leq 0)$, which our program quickly verifies as true.

Example 2. $\exists G \forall A (A \in G \rightarrow \exists B (B \in F \wedge A \subseteq B))$.

Solution: $G = [A: \exists B (B \in F \wedge A \subseteq B)]$.

This example is very much like the previous one, except that G plays the role of A, A plays the role of x, and $\exists B (B \in F \wedge A \subseteq B)$ plays the role of $x \leq 0$. Notice that the variable G is a "family" variable rather than a "set" variable. Thus we have (technically) proved a theorem in third order logic, although it is for all practical purposes a theorem in second order. It is desirable, we believe, to keep formulas like $\exists B (B \in F \wedge A \subseteq B)$ together during the processing and this is what our program usually does.

Next we consider a theorem of a little more substance.

Example 3.

$$(P(a) \rightarrow \exists A [\forall x (x \in A \rightarrow P(x)) \wedge \underset{\text{ODD}}{\exists} y\ (y \in A)]).$$

Solution: $A = [x: P(x)]$.

It is now time to note that $A = \emptyset$ is a perfectly good solution to Example 1. However, $A = \emptyset$ will not work for Example 3; here we must include in A at least the point a. We prefer to put all we can in A, thus getting a maximal solution.

In Example 3 there are two types of occurrences of A: the "EVEN" occurrence $y \in A$, and the "ODD" occurrence in $(x \in A \rightarrow P(x))$[2]. The ODD occurrences are used to determine A (according to the rules in Section 3), and the EVEN occurrences are just checked after A has been defined[3]. Accordingly when the solution given is put in for A we get

[2] Note that $(x \in A \rightarrow P(x))$ is equivalent to $(x \notin A \vee P(x))$. If A does not occur in B (except possibly as a skolem argument -- see footnote 4) then A is in an EVEN position of $A \wedge B$, $A \vee B$, $(B \rightarrow A)$, B, and A itself; and A is in an ODD position of $(A \rightarrow B)$, $(\sim A)$, and B. Also an EVEN position of an EVEN position is EVEN, ODD of ODD is EVEN, ODD of EVEN is ODD, and EVEN of ODD is ODD.

[3] This is equivalent to putting the universal set U for A for the EVEN occurrence and intersecting it with the set $\{x: p(x)\}$ gotten for the ODD occurrence.

$$(P(a) \rightarrow [\forall x\,(p(x) \rightarrow p(x)) \wedge \exists y\,(p(y))])$$

in which the ODD part is now trivial and the EVEN part is yet to be checked (but can be).

The next example will illuminate that point.

Example 4.

$$(a < b < c \rightarrow \exists A\,(a \notin A \wedge b \in A \wedge c \notin A))$$

ODD EVEN ODD

Solution: $\{x: x \neq a \wedge x \neq c\}$.

The two ODD occurrences give respectively $\{x: x \neq a\}$ and $\{x: x \neq c\}$, and the EVEN occurrence gives U (using the rules of Section 3) which are combined (by the "combining rules" of Section 3) into the given solution.

Notice that this is not the only solution. There are many others such as $\{x: a < x < b\}$, $\{x: a < x \leq b\}$, $\{b\}$, but none of these are maximal. Our method gives the maximal solution if there is one.

We could have developed a minimal theory, getting the smallest sets, by using the EVEN occurrences of A instead of the ODD, but we have a slight preference for maximal. Intermediate sets would be difficult to produce automatically. Only when we work against the extremes do we reduce the complexity of the problem.

3. Set Building Rules

3.1. Basic Rules

Our rules for generating maximal sets are of two kinds: (i) basic rules and (ii) combining rules. The basic rules give solutions to certain subformulas of the form: $(x \in A)$, $(x \notin A)$, $(x \in A \rightarrow p(x))$, and the combining rules consolidate these basic solutions into one general solution, depending on the placement of these subformulas in the theorem. For instance, in Example 4 above, the subformulas $(a \notin A)$, $(b \in A)$, and $(c \notin A)$ were connected by "$\wedge$", so the corresponding solutions $\{x: x \neq a\}$, U, and $\{x: x \neq c\}$ were intersected to obtain the general solution $\{x: x \neq a \wedge x \neq c\}$.

Figure 2 gives our first set of basic rules. More are added later. In Figure 2 the subformulas shown are expected to be in an EVEN[2] position of the theorem being proved, and the theorem itself is to be in skolemized form[4].

Also it should be noted that these rules cannot handle an expression in A until it is reduced to the form $(x \in A)$. (This includes the cases $x \notin A$ and $(x \in A \rightarrow C)$.) This reduction may require the use of hypotheses, definitions, and lemmas.

[4] See App. 1 of [1] for a complete description of skolemization, or footnote 12 of [2].

	SUBFORMULA	SOLUTION
B1.	$(x \in A \rightarrow P(x))$	$\{z: P(z)\}$
B2.	$(f(x) \in A \rightarrow P(x))$	$\{z: \forall s\,(z = f(s) \rightarrow P(s))\}$
B2'.	$(f(x,y) \in A \rightarrow P(x,y))$	$\{z: \forall r \forall s\,(z = f(r,s) \rightarrow P(r,s))\}$
B3.	$(t \in A \rightarrow P)$	$\{z: z = t \rightarrow P\}$
B4.	$(t \notin A)$	$\{z: z \neq t\}$
B5.	P (does not contain A)	U (universal set) (IGNORED in case of conjunctions)
B6.	$(t \in A)$	U
B7.	E ($t \in A$ is even in E)	U
BQ	If B1-B4 yield $\{z: P(z)\}$, and s is a variable in P(z)	$\{z: \exists s\,P(z)\}$
BE	ELSE	U

Figure 2

BASIC RULES

The rules of Figure 2 operate under the restrictions listed in Table I below:

0. The Rules B1-BE apply to subformulas in EVEN positions of the theorem. A is a set variable to be instantiated. It is the only set variable (indeed the only higher order variable) in the theorem. A occurs only in the form $(t \in A)$, or as a skolem function argument.
1. In B1 x is a skolem function[4] of A, A does not occur otherwise in P(x), x does not occur elsewhere in the theorem, and no other variable occurs in x (i.e., x is a skolem function of no other variable but A).
2. In B2, B2', x and y are skolem functions of A, A does not occur otherwise in P(x) or P(x,y), x and y do not occur elsewhere in the theorem, and no other variable occurs in x, y, f(x), or f(x,y).
3. In B3, A does not occur in t or P.
4. In B4, A does not occur in t
5. In B5, A does not occur in P.
6. In B6, A does not occur in t.

7. In B7, every occurrence of an expression of the form $(t \in A)$ in E, is in an EVEN position of E, A does not occur in t. A cannot occur in E except in one of these subformulas $(t \in A)$.

8. In BQ, s is a variable in P(z), but does not occur elsewhere in the theorem.

9. In BE, the subformulas have the form of B1-B7 or BQ, but the restrictions 1-8 are not satisfied.

Table I

In Rules B1 and B2, "x" is required to be a skolem[4] function of A, and not appear in the rest of the theorem. That is to say the subformula $\forall x\,(x \in A \rightarrow p(x))$ occurs in an EVEN position of the theorem within the scope of A. See Section 7 of [0] for a further discussion of this and some means of easing these restrictions on x. Similarly for x and y in B2'.

Note that when the term "t" is not quantified within the scope of the quantification of A, as in $\exists A\,(t \in A \rightarrow P(t))$ then by Rule B3, we do not derive the solution $\{z: P(z)\}$ as was done in Rule B1, because it is not the maximal solution. Rather the maximal solution is $\{z: z \neq t \rightarrow P(t)\}$ as given by Rule B3. Rule B4 acts similarly.

Rules B5-B7 give the universal set U as a solution for expressions of the form $(t \in A)$ and for EVEN combinations of these. Since $(U \cap A_0) = A_0$, the effect is to ignore expressions that give the solution U whenever they are conjoined (connected by "$\wedge$") to other expressions.

These rules and those in Sections 3.2-3.6 are supported to some degree by the proofs in Appendix II of [0]. Our objective is to find maximal sets for a few basic forms, and to give combining rules that retain that maximality.

3.2. Combining Rules for Conjunctions

We will first give in Figure 3, three combining rules for conjunctions before giving others. These three rules with the basic rules of Figure 2 have been sufficient to prove a number of interesting theorems. Even our extended list of combining rules is by no means complete; we are now in the process of trying to validate and extend both the basic and combining rules.

(i) If A_1 is the (only)[5] maximal solution for P(A),

(ii) and A_2 is the (only) maximal solution for Q(A),

(iii) and both A_1 and A_2 are obtained from the basic rules of Figure 1, or from combining rules C1-C3, (or if (iii)': P(A) and Q(A) have the intermediate property (see below))

C1. then $(A_1 \cap A_2)$ is a maximal solution of $(P(A) \wedge Q(A))$, if it has a solution,

C2. and $(A_1 \cap A_2)$ is a maximal solution of $(H \rightarrow P(A) \wedge Q(A))$, if it has a solution, provided that "A" does not occur in H,

C3. and $(A_1 \cap A_2)$ is a maximal solution of $((R \vee P(A)) \wedge Q(A))$ if it has a solution, provided that "A" does not occur in R.

Figure 3
Three Combining Rules for Conjunctions

Definition. We say that a formula P has the subset property (superset property) if for each B and C, if B is a solution of P and C is a subset (superset) of B then C is a solution of P.

Definition. We say that a formula P has the intermediate property if for each B, C, and D, if B and D are solutions of P and $B \subseteq C \subseteq D$, then C is a solution of P.

It is easily seen that if P has the subset property or the superset property then P has the intermediate property, because $\emptyset$ is a solution of a formula with the subset property (if it has any solution), and U is a solution of a formula with the superset property (if it has any solution).

It is easily shown that the subformulas in Rules B1-B5, and BQ, have the subset property, and those of Rules B5-B7 have the superset property; and that the conjunction of two formulas with the intermediate property also has the intermediate property. Therefore the combining rules of Figure 3 are valid by condition (iii)' as well as by (iii), by Theorems 12-13 of Appendix II of [0].

3.3. Combining Rules for Disjunctions

Often there can be more than one maximal solution or even infinitely many[6]. For example the theorem

$$A\,[\forall x\,(x \in A \rightarrow P(x)) \vee \forall y\,(y \in A \rightarrow Q(y))]$$

has two maximal solutions $\{z: P(z)\}$ and $\{z: Q(z)\}$ each with the subset property. (But $\{z: P(z) \vee Q(z)\}$ is not a solution!)

When there is more than one maximal solution we will indicate the "maximal solution" as a family $\mathcal{F}$ of sets. For example, the family of candidate solutions for

[5] If P(A) or Q(A) has more than one maximal solution (see Example 5 and Figure 5 below), this rule still applies to at least one of the solutions A_1 of P(A) and one of the solutions A_2 of Q(A)t

[6] See Section 3.5.

Example 5.

$$P(a) \rightarrow \exists A\,([\forall x\,(x \in A \rightarrow P(x)) \wedge \exists y\,(y \in A)] \vee [\forall x\,(x \in A \rightarrow Q(x)) \wedge \exists y\,(y \in A)]),$$

is: $\mathcal{F} = \{\{z: P(z)\}, \{z: Q(z)\}\}$.

Then we must verify that some member of $\mathcal{F}$ does indeed satisfy the theorem. In this example only $\{z: P(z)\}$ satisfies.

This brings us to Combining Rules C4-C6 in Figure 4.

3.4. Further Basic and Combining Rules

Each of the subformulas in the basic rules of Figure 2, except Rule BE, have the intermediate property. This allowed us to use the rather simple combining rules of Figure 3 for conjunctions. In Figure 6 of [0] we give rules for some cases which do not have the intermediate property, and use these to prove some theorems from program verification, where internal assertions are not provided by the user but are found by the prover. (See Example 14 below)

If (i), (ii), (iii)′ of Figure 3,

C4. then $\mathcal{F} = \{A_1, A_2\}$ is the maximal solution of $(P(A) \vee Q(A))$.

Also if

(iv) $P(A)$ has a maximal solution $\mathcal{F}_1$,

(v) and $Q(A)$ has a maximal solution $\mathcal{F}_2$,

(vi) where $\mathcal{F}_1$ and $\mathcal{F}_2$ are gotten from C4 - C5, or are gotten from B1-BQ where the output there is treated as a singleton $\{A_0\}$,

C5. then $(\mathcal{F}_1 \cup \mathcal{F}_2)$ is a maximal solution of $(P(A) \vee Q(A))$,

C6. and $(\mathcal{F}_1 \cap\cap \mathcal{F}_2)$[7] includes a maximal solution of $(P(A) \wedge Q(A))$.

Figure 4

Two Combining Rules for Disjunctions

3.5. Infinitely Many Maximal Solutions

We saw in Section 3.3 an example which has two maximal solutions. Others have more, even infinitely many. For example

Example 5A.

$$\exists A \forall x \forall y\,(x \in A \wedge y \in A \rightarrow P(x,y))$$

[7] $(\mathcal{F}_1 \cap\cap \mathcal{F}_2)$ is the family of sets $(D_1 \cap D_2)$ with $D_1 \in \mathcal{F}_1$ and $D_2 \in \mathcal{F}_2$.

If $P(x,y)$ can be "separated", the solution is easy. For example if $P(x,y) = p(x) \wedge q(y)$, then the maximal solution is $\{z: p(z) \wedge q(z)\}$.

However if $P(x,y) \equiv (0 \leq x/2 \leq y \leq 2x)$, then the maximal solutions consist of the infinite family of all the squares of the form

$$\{(x,y): a \leq x \leq 2a \wedge a \leq y \leq 2a\}$$

for $a \geq 0$. (See Figure 5) Notice that these maximal solutions may overlap.

Clearly one could not try all of these solutions, in order, as could be done in Example 5. However, as we shall see shortly, our rules can be very effective indeed even when there are infinitely many solutions.

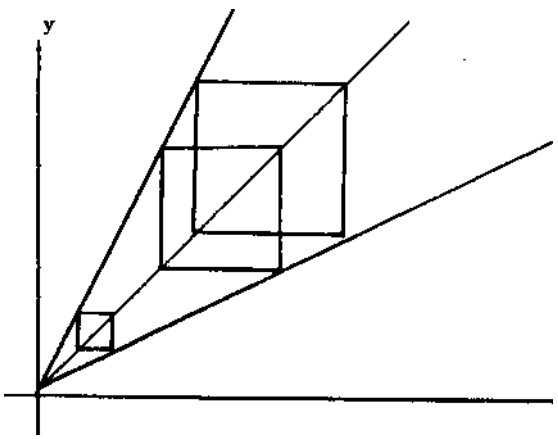


Figure 5

Infinitely many maximal solutions of

$$\exists A \forall x \forall y\,(x \in A \wedge y \in A \rightarrow P(x,y))$$

Example 5B.

(1) $\exists A \forall L \exists y \forall z\,[(z \in A \rightarrow p(z,y)) \wedge q(L,y)]$.

This is equivalent to (successively):

$$\exists A \exists g \forall L \forall z[(z \in A \rightarrow p(z,g(L))) \wedge q(L,g(L))]$$

$$\exists g \exists A[\forall z\,(z \in A \rightarrow \forall L\, p(z,g(L))) \wedge \forall L\, q(L,g(L))]$$

(2) $\exists g \exists A\,[\forall z\,(z \in A \rightarrow P(z,g)) \wedge Q(g))]$.

Now if g_0 is a solution for $Q(g)$ then $\{z: P(z,g_0)\}$ is a maximal solution (2). At first blush this would seem of little value, since finding such a g_0 is itself a search problem in second order logic, (and since indeed the complete maximal solution of (2) is the possibly infinite family of all such $\{z: P(z,g_0)\}$). However, in many applications the family of maximal solutions is reduced to a family of one by matches during

the proof on other parts of the theorem. This is exactly what happens in the use of the Least Upper Bound Axiom, LUB, to prove theorems like Example 11 of Section 5.

4. Control

4.1. The Prover as a Control

As mentioned above, the set building rules of Section 3 are used to propose a value for a set variable A and then IMPLY, our automatic prover [1] is used to prove the resulting theorem.

It turns out that the prover is also a convenient vehicle for controlling the use of the set building rules. In doing so it proceeds in its normal way to prove the theorem, applying a list of production rules: to manipulate the theorem, to propose subgoals, to manipulate the data base, to match, etc.[8] The set building rules are added to these production rules.

For example, in proving the theorem,

Example 6. $\exists A\,(P(a) \wedge a \neq b \rightarrow \forall x\,(x \in A \rightarrow P(x)) \wedge \exists y\,(y \in A) \wedge b \notin A)$,

It first skolemizes and sets up the goal

$$(\ ) \qquad (P(a) \wedge a \neq b \rightarrow (x_A \in A \rightarrow P(x_A)) \wedge (y \in A) \wedge b \notin A),$$

and then splits it into subgoals (1), (2), (3).

$$(1) \qquad (P(a) \wedge a \neq b \rightarrow (x_A \in A \rightarrow P(x_A))).$$

Rule B1 is applied to this subgoal to yield the solution $\{z: P(z)\}$ for A. This value of A is not substituted for A in the reminder of the theorem, but rather is placed in the data base, to be combined with other values gotten later.

$$(2) \qquad (P(a) \wedge a \neq b \rightarrow y \in A).$$

This subgoal is ignored by Rule B6. (Actually it yields the universal set U, which will be intersected with other sets and therefore "ignored".)

$$(3) \qquad (P(a) \wedge a \neq b \rightarrow b \notin A).$$

Rule B4 is applied to this subgoal to yield the solution $\{z: z \neq b\}$ for A. This too is placed in the data base, and combined with the earlier solution, yielding

$$(*) \qquad \{z: P(z) \wedge z \neq b\}$$

as the current solution for A.

Since (3) is the last subgoal, the final solution (*) is substituted for A in the original theorem, getting the new goal:

$$(\ ) \qquad (P(a) \wedge a \neq b \rightarrow (P(x_0) \wedge x_0 \neq b \rightarrow P(x_0)) \wedge (P(y) \wedge y \neq b) \wedge \sim(P(b) \wedge b \neq b)).$$

Now this, which is a first order theorem, is proved as a series of subgoals.

Our intention here is to emphasize the part of the process that builds the set A using the set building rules, and to deemphasize the proof of the resulting first order theorem.

4.2. Interval Types

Before describing other examples in Section 5 we will describe another part of our data base which is called "Interval types" in [4] and which plays a crucial part in proofs of theorems in real analysis. This is best illustrated with an example.

Example 7. $(P(1) \rightarrow \exists x\,(0 \leq x \leq 2 \wedge P(x)))$.

The skolemized goal is

$$(\ ) \qquad (P(1) \rightarrow (0 \leq x \leq 2 \wedge P(x))).$$

The prover handles the first subgoal

$$(1) \qquad (P(1) \rightarrow 0 \leq x \leq 2)$$

not by giving some particular value to x, such as 0 or 2 or 1, but rather by storing the entry $\{x: 0\ 2\}$ in the data base, indicating that the variable x is now restricted to the interval $0 \leq x \leq 2$. The prover then goes to the second subgoal

$$(2) \qquad (P(1) \rightarrow P(x))$$

with x still a variable. When this goal is solved, with $1/x$, then it must verify that this substitution is consistent with its data base entry, i.e. that $0 \leq 1 \leq 2$.

Such a data base mechanism is used in the proof of the intermediate value theorem in Example 11 below, and other like proofs in real analysis. This concept which was used in [2], has literally made the undoable doable. Otherwise one is involved in the use of the axioms for the real numbers and for inequalities which tend to choke automatic provers not using special mechanisms like this.

The reader is referred to [1] for details and examples. For our purposes here the prover described in [1] has been augmented by a data base for handling our set variables and interval types (see Section 4.2). These data base manipulations are in the spirit of those mentioned in [9] and applied in [5,6] and [8].

5. Some Major Examples

Section 5 of [O] summarizes the proofs of several examples. We list here some of those examples and the derived[9] solution for each.

Example 8. If a set B contains an open neighborhood of each of its points then B is open.[10]

Or symbolically

$$(\forall x\,(x \in B \rightarrow \exists D\,(\text{Open } D \wedge x \in D \wedge D \subseteq B)) \rightarrow \text{Open } B).$$

The following lemma is used:

L: The union of a family of open sets is Open.

I.e.,

$$(\exists G\,(G \subseteq \text{OPEN} \wedge D = (\text{Union } G)) \rightarrow \text{Open } D).$$[10]

The object is to find this family G.

Solution: $\{Z\colon \text{Open } Z \wedge Z \subseteq B\}$.

Example 9.[11] If F is a family of open sets covering the regular topological space X, then there exists a family G of open sets which covers X and for which $G \subseteq\subseteq F$.

The object is to find this family G.

The following definitions are used:

Regular: $\forall A \forall x\,(\text{Open } A \wedge x \in A \rightarrow \exists B\,(\text{Open } B \wedge x \in B \wedge \overline{B} \subseteq A))$

OC F: $F \subseteq \text{OPEN} \wedge \text{Cover } F$

Cover F: $\forall x \exists A\,(A \in F \wedge x \in A)$.

Thus our theorem becomes

$$(\text{Regular} \wedge \text{OC } F \rightarrow \exists G\,(\text{OC } G \wedge \overline{G} \subseteq\subseteq F)),$$

$$(\ ) \quad (\text{Regular} \wedge \text{OC } F_0 \rightarrow (\text{OC } G \wedge \overline{G} \subseteq\subseteq F_0)),$$

in skolemized form.

(1) Solution:

$$\{Z\colon \text{Open } Z \wedge \exists B\,(B \in F_0 \wedge \overline{Z} \subseteq B\} \quad \text{for} \quad G.$$

The next (rather simple) example is given to show the effect of lemmas on the maximal solution. Let R represent the real numbers and Q the rationals.

Example 10. $\exists A\,(A \text{ is dense in } R) \wedge ((R-A) \text{ is dense in } R)$.

This of course has no maximal solution for A. However, if we employ the lemma

L: $\forall B\,(Q \subseteq B \rightarrow B \text{ is dense in } R) \wedge \forall D\,(D \subseteq Q \rightarrow (R-D) \text{ is dense in } R)$

then we do get a maximal solution. (See [O]) $\{z\colon z \in Q\} = Q$.

So a theorem with no maximal solution was given one by the use of the lemma. In other theorems, the maximal solution is often changed (decreased) by the use of lemmas. Indeed that is the case in Example 11 below where a non-maximal (but adequate) solution

$$\{z\colon z \leq b \wedge f(z) \leq 0\}$$

is given instead of the actual maximal solution

$$\{z\colon \exists x\,(x \leq b \wedge f(x) \leq 0 \wedge z \leq x)\}$$

Example 11. (Intermediate Value Theorem)

If f is continuous for $a \leq x \leq b$, $a \leq b$, $f(a) \leq 0$, and $f(b) \geq 0$, then $f(x) = 0$ for some x between a and b. (See Figure 1)

The proof of this requires the least upper bound axiom, LUB:

LUB. Each non-empty bounded set A has a least upper bound.

The object here is to find the set A required by the least upper bound axiom. The definition of the set needed here is not at all obvious (even for humans). We believe that the use of a natural deduction prover, such as ours, as a control, is the key to this kind of problem, whereby the prover explores its various subgoals in a natural way and uses the basic set-building Rules B1-BQ as they become applicable. Only combining Rules C1-C3 are applicable in this example so the basic solutions are intersected to obtain the general solution

$$\{z\colon z \leq b \wedge f(z) \leq 0\}$$

for A. (See [O] for details.)

---

[9] See Section 6 for a discussion of what was actually proved by the computer on these examples.

[10] An open neighborhood of a point x is just an open set containing x. We will use the capitalized OPEN to represent the family, all Open sets. (Union G) is defined to be the set of points contained in some member of G.

[11] We will denote by $\overline{A}$ the "closure" of a set A, and by $\overline{G}$ the family of closures of members of G, i.e. $\overline{G} = \{\overline{A}\colon A \in G\}$. Furthermore $(H \subseteq\subseteq F)$ means that H is a refinement of F, i.e., each member of H is a subset of a member of F, or $(\forall A\,(A \in H \rightarrow \exists B\,(B \in F \wedge A \subseteq B))$. Thus $(\overline{G} \subseteq\subseteq F)$ means $\forall A\,(A \in G \rightarrow \exists B\,(B \in F \wedge \overline{A} \subseteq B))$.

The theorem and axiom (in symbols) are:

Th. $\forall y\,(a \le y \le b \to \text{Cont } f\,y) \wedge a \le b \wedge f(a) \le 0 \wedge f(b) \ge 0 \to \exists x\,(f(x) = 0)$.

LUB. $\forall A\,(\exists u \forall t\,(t \in A \to t \le u) \wedge \exists r\,(r \in A) \to \exists \ell\,[\forall x\,(x \in A \to x \le \ell) \wedge \forall y\,(\forall z\,(z \in A \to z \le y) \to \ell \le y)])$.

Instead of the definition of continuity the following two lemmas are used:

L1. $(\text{Cont } f\,x \to \forall a\,(a < f(x) \to \exists t\,(t < x \wedge \forall s\,(t < s \le x \to a < f(s)))))$

L2. $(\text{Cont } f\,x \to \forall a\,(f(x) < a \to \exists t\,(x < t \wedge \forall s\,(x \le s < t \to f(s) < a))))$.

The "interval typing" described in Section 4.2 played a crucial role in this example. See [0].

Example 12. $\forall F\,([x] \in F \to [y] \in F) \to \forall A\,(x \in A \to y \in A)$.

This example which was suggested by Peter Andrews[12] is just the theorem

$$([x] = [y] \to x = y)$$

where $(\alpha = \beta)$ has been characterized by $\forall D\,(\alpha \in D \to \beta \in D)$, (with the proper typing on D).

Solution: $[z: z = [y] \to y \in A]$ for F, using Rule B3

The next two examples use some additional Rules from Reference [0]. These are found in Figures 6 and 7 of [0].

In the following example we use Rules B1 and BC1 of [0] to obtain the preliminary solution, which is then simplified (by the author) to the solution.

Example 13. $\exists A\,[\forall x\,(x \in A \to x \ge 0) \wedge \forall x\,(x \in A \wedge x \ne 0 \to x-2 \in A)]$.

Preliminary Solution:

$[z: z \ge 0 \wedge [\forall n\,(n \in \omega \to z-2n \ge 0) \vee \exists N\,(N \in \omega \wedge \forall n\,(n \in \omega \wedge n \le N \to z-2n \ge 0 \wedge z-2N = 0))])]$.

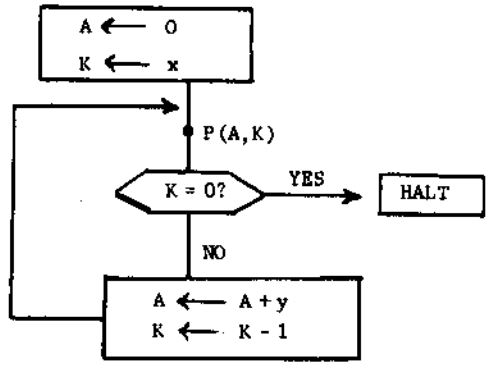
Solution:

$[z: z \ge 0 \wedge \exists N\,(z = 2 \cdot N)]$ = the set of non-negative even integers.

[12] Private communication.

Example 14. $\exists P \forall A \forall K\,[P(0,x) \wedge (P(A,0) \to A = x \cdot \wedge (P(A,K) \wedge K \ne 0 \to P(A+y,K-1))]$

This theorem arises from the field of program verification, in a case where the internal assertion P is not given but must be found by the prover. The theorem represents the verification condition for a simple program which multiplies integers x and y. (See Figure 6)



Input assertion: $x \ge 0$, $y \ge 0$, integers.

Output assertion: $A = x \cdot y$.

Internal assertion at ϕ: P(A,K), to be determined.

Figure 6

Flow Chart for a Simple Multiply Program

In Example 14 we replace the two-place predicate P by a set B of ordered pairs, and skolemize to obtain

$$[(0,x_0) \in B \wedge ((A_B,0) \in B \to A_B = x_0 \cdot y_0) \wedge ((A_B,K_B) \in B \wedge K_B \ne 0 \to (A_B+y_0,K_B-1) \in B)].$$

Then Rules B6, B2' and BC2' are used to obtain,

$$\text{Solution: } [z: \forall s \forall t\,(z = (s,0) \to s = x_0 \cdot y_0) \wedge \forall n\,[n \in \omega \wedge \text{2nd term } (\lambda st(s+y_0, t-1)^n(z)) \ne 0) \wedge \forall s_1\,(\lambda st(s+y_0,t-1)^{n+1}(z) = (s_1,0) \to s_1 = x_0 \cdot y_0)]].$$

This solution is then simplified (by the author) to

$$[(A,K): K \in \omega \wedge A = (x_0-K) \cdot y_0)$$

which corresponds to the predicate P, where

$$P(A,K) \equiv K \in \omega \wedge A = (x_0-K) \cdot y_0$$

which is the usual interval assertion given by humans for the program dipicted in Figure 6. See [0] for more details.

6. Comments

Delaying

In our procedure we have employed a concept of delaying, whereby we delay the final determination of a set A until all parts of the theorem have been processed. Early subgoals place restrictions of the form {z: P(z)} on A, but leave A itself as a variable to be further considered later. Later subgoals may further restrict A, or may force A to take a particular value A0 (e.g., by matching). In this last eventuality the program must check that Ao is consistent with the earlier restriction (z: P(z)}. This kind of delaying has the marked advantage of not closing off the determination of A by assuming early values for it; but rather keeping it "as general as possible", putting on restrictions only as they are forced. Thus we see that the notions of "maximality" and "delaying" are somewhat analogous.

This concept of delaying is an important one in other parts of automatic theorem proving. Huet's constrained Resolution 13] is an example of it where he delays the higher order unifications until resolution matches have been made. The most general unifier [16] is another example, in that it lets resolution (or whatever prover that uses it) delay as long as possible the assignment of constant values to variables.

Also our use of delaying for set variables is entirely analogous to the concept of interval types [2,5,6] explained in Section 4.2, when a variable x is restricted to an interval [a < x < b] to satisfy an earlier subgoal, but left a variable to be instantiated or further restricted later. This technique has greatly simplified our proofs in analysis, and we expect other such "delaying" methods to be developed.

Relation to the Work of Others

Darlington's program [12,14] has proved Example 8 and other examples using his F-matching. Our procedure has a similarity to F-matching and was partially inspired by talks with Darlington. But it is different especially in its use of the maximal!ty concept which is an outgrowth of the ideas in [15, Sec. 10], and in other ways.

This work is of course related to Behmann's decision procedure for monadic first and second order logic [13][13]. A cursory look at [13] indicates that our solutions are often the same as Behmann's. His methods might be extended to also handle a number of non-monadic cases (as ours do). So it seems that an extensive study of papers on monadic logic is very much in order.

[13]J.A. Robinson first pointed this out to me.

The procedures of [3,4,7,10,11] are more general than ours, and their research provides a necessary base for this type research; we only feel that our work can be more effective on a limited, but important part, of higher order logic.

Completeness

All of our rules are sound because no matter what value we get for the set variable A, we always verify it with another pass through the prover. The only question, then, is one of completeness.

See Section 7.5 and Appendix II of [0] for a discussion of, and some proofs on, completeness of these rules.

Implementation

An augumented version of the prover described in |1], which operates in Machine-only Mode (i.e., not man-machine), has been used to prove some example theorems. But not all of Examples 1-15 of this paper were actually proved. Examples 1-4, 6-12, were proved outright. Some others could have been proved by minor changes in the program which we are in the process of making; and some require more extensive changes.

Acknowledgment

This work has benefited from conversations with Mike Ballantyne, Mabry Tyson, Peter Bruell, Allan Robinson, Ernie Sibert, Peter Andrews, and Gerard Huet. Also, Peter Bruell has helped develop and implement the computer program.

References

0. W.W. Bledsoe. A Maximal Method for Set Variables in Automatic Theorem Proving. Univ. of Texas Math. Dept. Memo ATP-33, Feb. 1977.

1. W.W. Bledsoe and Mabry Tyson. The UT Interactive Theorem Prover. The Univ. of Texas at Austin, Math. Dept. Memo ATP-17, May 1975.

2. W.W. Bledsoe, Robert S. Boyer and William H. Henneman. Computer Proofs of Limit Theorems. A.I. Jour., 3(1972), 27-60.

3. G.P. Huet. Constrained Resolution: A Complete Method for Higher Order Logic. Ph.D. thesis, Case Western Reserve Univ., Jennings Computer Center Report 1117.

4. G.P. Huet. A Unification Algorithm for Typed ^-Calculus. Theoretical Computer Science, 1(1975), 27-57.

5. D.I. Good, R.L. London and W.W. Bledsoe. An Interactive Verification System. Proc. of the 1975 International Conf. on Reliable Software, Los Angeles, Ca., April 1975, 482-492, and IEEE Trans, on Software Engineering. 1(1975), 59-67.

6. W.W. Bledsoe and Mabry Tyson. Typing and Proof by Cases in Program Verification. Univ. of Texas Math. Dept. Memo ATP-15, May 1975. To appear in Machine Intelligence 8, Donald Michie and E.W. Elcock (Eds.), Reidel Publishing Co., 1977.

7. G.A. Haynes and L.J. Henschen. A Refutation Procedure for Omega-order Logic. Northwestern Univ., Aug. 1976. Informal memo.

8. A. Michael Ballantyne and W.W. Bledsoe. Automatic Proofs of Theorems in Analysis Using non-standard Techniques. The Univ. of Texas Math. Dept. Memo ATP-23, July 1975. To appear in JACM.

9. W.W. Bledsoe. Non-resolution Theorem Proving. Univ. of Texas Math. Dept. Memo ATP-29, Sept. 1975. To appear in the A.I. Jour.

10. T. Pietrzykowski. A Complete Mechanization of second Order Type Theory. JACM (1973), 333-364.

11. Peter B. Andrews. Resolution in Type Theory. Jour, of Symbolic Logic, 36(1971), 414-432.

12. J.L. Darlington. Deductive Plan Formation in Higher Order Logic. Machine Intelligence 7, pp. 129-137.

13. Heinrich Behmann. Beitrage Zin Algebra Der Logik: Insbesondere Zum Entscheidongsproblem Mathematische Annalen, 86(1922), 163-229.

14. J.L. Darlington. Talk at Oberwalfach Conference on Automatic Theorem Proving. Oberwolfach, Germany, Jan. 1976.

15. W.W. Bledsoe. Some Ideas on Automatic Theorem Proving. Univ. of Texas Math. Dept. Memo ATP-9, May 14, 1973.

16. J.A. Robinson. A Machine-oriented Logic Based on the Resolution Principle. JACM, 12(1965), 23-41.

17. R.S. Boyer and J S. Moore. Proving Theorems about LISP Functions. JACM, 22(1975), 129-144.

18. J.L. Darlington. Automatic Theorem Proving with Equality Substitution and Mathematical Induction. Machine Intelligence 3, Edinburgh Univ. Press (1968), 113-127.

19. C. Chang and R.C. Lee. Symbolic Logic and Mechanical Theorem Proving. Academic Press. 1973. Section 11.10.

20. W.W. Bledsoe. Splitting and Reduction Heuristics in Automatic Theorem Proving. A.I. Jour. (2), (1971), 55-77.