Department of Mathematics
The University of Texas
Austin, Texas 78712

VERY ROUGH DRAFT

Some Automatic Proofs in Analysis

by

W. W. Bledsoe

December 1982 ATP-71

This work was supported by NSF Grant MCS-801-1417.

This paper is an expanded version of a talk given in a special session on Automatic Theorem Proving at the Annual Meeting of the American Mathematical Society, Denver, Colorado, January 6, 1983.

See Automated Theorem Proving: After 25 years
Am. Math Soc. Contemporary Math series #29
pp 89-118.

Some Automatic Proofs in Analysis
W. W. Bledsoe

23 December 1982

1. Introduction

The work described in this talk is mainly that of our own group, or that of others that we know best.

During the last few years a limited number of theorems in introductory Analysis and related areas, have been proved by automatic theorem provers. These fall in the following subareas:

Set Theory

Elementary Set Theory

Theorems in set theory requiring Induction

The Limit Theorems of Calculus

Intermediate Analysis

Elementary Topology

We list below a number of these theorems, and describe in a general way some techniques used in their proofs. All of these have been proved by "stand-alone" provers, with no human help.

LIST OF THEOREMS PROVED

1. Set Theory

1.1 Elementary Set Theory

.1 $A \subseteq A$

.2 $A \subseteq A \cup B$

.3 $A \cup (B \cup C) = (A \cup B) \cup C$

.4 $A \cap (B \cap C) = (A \cap B) \cap C$

.5 $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

.6 $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

.7 $Sb(A \cap B) = Sb(A) \cap Sb(B)$ [1]

etc.

1.2 Set Theory Theorems Requiring Induction

.1 $\omega = \sigma \omega^{2}$

.2 $\omega = \omega \cap Sb\,\omega$

1.3 Higher order set theory theorems

.1 $\{x\} = \{y\} \Rightarrow x = y$

.2 (Cantor's theorem) N^{N} is not denumerable. (N here is the set of integers).

2. The Limit Theorems of Calculus

2.1 (Limit of a sum and product)

If $\lim_{x \to a} f(x) = L$ and $\lim_{x \to a} g(x) = K$, then

(+) $\lim_{x \to a} (f+g)(x) = L+K$

and (·) $\lim_{x \to a} (f \cdot g)(x) = L \cdot K$.

---

1. $Sb(X)$ means the family of subsets of X, i.e., $Sb(X) = \{Y: Y \subseteq X\}$.

2. We use here the definitions: 0 = empty set, 1 = $\{0\}$, 2 = $\{0,1\}$, etc., $\omega = \{0,1,2,3, \ldots\}$. And σF is the union of the members of F, i.e., $\sigma F = \bigcup_{A \in F} A$

2.2 The sum of two continuous functions is continuous.

2.3 The product of two continuous functions is continuous.

2.4 The composition of two continuous functions is continuous.

2.5 Differentiable functions are continuous.

2.6 A uniformly continuous function is continuous.

3. Intermediate Analysis (On the Real Numbers).

3.1 If the function f is continuous on the compact set S then f is uniformly continuous on S.

3.2 If the function f is continuous on the compact set S then $f[S]$ is compact.

3.3 If a sequence s_n converges to some limit p, then s_n is a Cauchy sequence.

3.4 If s_n is a Cauchy sequence, then s_n converges to some limit p.

3.5 If a sequence s_n converges simultaneously to a and b, then $a = b$.

3.6 If the function f is continuous at the point a, and

$$\lim_{n\to\infty} s_n = a, \text{ then } \lim_{n\to\infty} f(s_n) = f(a).$$

3.7 (Balzano-Weierstrass Theorem). If S is an infinite, bounded set, then there exists an accumulation point p of S.

3.8 Let f and g be two continuous functions. Then the set of points on which f and g agree is a closed set.

3.9 If, for the sequences x_n and y_n, we have

$$\lim_{n\to\infty} x_n = a \text{ and } \lim_{n\to\infty} y_n = b,$$

then $\lim_{n\to\infty} (x_n + y_n) = a + b$, and $\lim_{n\to\infty} (x_n \cdot y_n) = a \cdot b$.

3.10 If x_n is a sequence and $\lim_{n\to\infty} x_n = a \neq 0$, then

$$\lim_{n\to\infty} 1/x_n = 1/a .$$

3.11 (Explicit examples).

.1 $\lim_{n\to\infty} n/(n + n^2) = 0$.

.2 If $f(x) = 2x^2 + 3x + 1$, then $f'(a) = 4a + 3$.

3.12 (Intermediate Value Theorem). If f is continuous on $[a,b]$, $a \leq b$, $f(a) \leq 0$, and $0 \leq f(b)$, then $f(x) = 0$ for some x in $[a,b]$.

4. Elementary Topology.

4.1 If a set B contains an open neighborhood of each of its points, then B is open.

4.2 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 $\overline{G} \subseteq\subseteq F$. [3]

It should be noted that this list does not represent all the theorems (in analysis) that can be proved automatically, but rather all that have been proved. A particular research group might spend most of its time developing new methods rather than trying to extend the number of theorems that have been proved automatically.

This list of theorems need explanation, because a number of methods (sometimes special heuristics) were used in their proofs. And a casual reader cannot properly understand what has been done until he has a feeling for these methods and their applicability to other theorems.

For example, the method of REDUCTIONS (Rewrite rules or "demodulators") was used to "build-in" elementary set theory into the prover [1], thereby making it easy to prove automatically the theorems of 1.1, and making it possible to prove those of 1.2 by induction. This and algebraic simplificatior and other methods will be discussed in later sections.

Also a special limit Heuristic [2] was used to prove the limit theorems of calculus, 2. Later some of these were proved without this special heuristic, but the limit heuristic made all such proofs easier. A method of variable restriction was also used for these other theorems.

A technique based on Nonstandard Analysis [3] was used to prove the intermediate analysis theorems 3.1-3.11, and the limit theorems 2.1. This interesting method greatly simplifies proofs but has limited applicability.

A method for automatically instantiating set variables [4], along with techniques for handling general inequalities [5], were used to prove 3.12, 4.1-4.2, and a number of others.

Much of real analysis uses inequalities in a fundamental way. Special provers have been developed to handle ground inequalities theorems (those without variables) and general inequality theorems (those with variables to be instantiated). Also special methods have been developed for handling equality.

---

3. $\overline{G}$ denoted the family of the closures of members of G, i.e., $\overline{G} = \{\overline{A}: A \in G\}$, and $(H \subseteq\subseteq F)$ means that H is a refinement of F, i.e., each member of H is a subset of some member of F.

A prover for a particular theory (such as the first order logic) is said to be complete for that theory if it can prove all theorems within the theory. It is a decision procedure if it can also detect non-theorems.

Completeness is important in ATP because a prover is not worth much if it exhibits very little generality. On the other hand, experience so far has shown that complete procedures have tended to be weak, in the sense that they take too long to prove easy theorems (or cannot prove them at all). So various attempts have been made to obtain speed without sacrificing (much) completeness.

There are two basic types of automatic provers in use, Resolution based provers [6,7,8] and Natural Deduction type provers [9,2,8 Ch 6]. These are described briefly below.

Resolution based provers tend to be complete whereas Natural Deduction provers tend not to be, especially when various auxillary procedures are added (for speed).

For both Resolution and Natural Deduction a theorem is first converted to quantifier-free form by a Skolemization process and then proved. (see Section 2.2 below).

The theorems listed above were all proved by Natural deduction or Resolution (or both), using the various procedures and heuristics listed in the following table.

LIST OF METHODS USED

Theorems	Type of Prover	REDUCTIONS and Simplifications?	Special Heuristics and Procedures	Complete?
1. Set Theory 1.1 Elem set Th.	Nat Ded, Resolution	yes	none	no
1.2 Set Th. Thms using induction	Nat Ded, Res	yes	*Induction*	no
1.2 Higher order set th thms	Resolution	no	Higher Order Resolution	no
2. Limit Thms 2.1-2.6	Nat Ded	yes	Limit Heuristic, Variable restrictions	no
3. Intermediate Analysis 3.1-3.11, 2.1-2.6	Nat Ded	yes	Non-standard Anal, Var restrictions	no
3.12(Part)[4] 2.1.1(+)	Nat Ded, Resolution based	yes	Var elimination and Shielding term removal	yes[5]
4. Elementary Topology 4.1-4.2	Nat Ded	yes	Set Variable instantiation	no

---

4. This is an inequality theorem derived from the the Intermediate Value theorem. See [5] and Section 5.2 below.

5. This prover [5] is complete for the first order logic, but the actual implementation which proved this theorem used manditory variable elemination which is not known to be complete.

The limit heuristic used in the proofs of 2.1-2.6 and the non-standard analysis technique used in the proofs of 3.1-3.11, are very powerful agents for theorems in the domain for which they apply, but unfortunately do not seem to have much applicability in other areas.

On the other hand the Reduction and Simplification routines have general applicability throughout all mathematics, and seem to be an essential part of any good prover, especially in analysis. And the Variable restriction and variable elimination techniques seem to have general applicability wherever real inequalities (with variables) are encountered. It is not clear at this time what generality the set variable methods, used in 4.1-4.2, and 3.12, will have.

2. REDUCTIONS, Skolemization, and Induction

2.1 Building-in Elementary Set Theory.

A number of researchers have exercised their provers on theorems like those of 1.1-1.6 from elementary set theory, with varying degrees of success. One difficulty with such experiments has been that the (human) user has had to give to the prover, as additional hypotheses, a number of axioms and definitions from which these simple theorem are derived. This does not appear to be a difficulty until one realizes that when a prover is given too many hypotheses the proof time is drastically increased (because there are many more ways for variables to match). Also when the prover is working on harder theorems, which use these elementary theorems as lemmas, one is obliged to add these theorems themselves as additional hypotheses or forever carry along the needed axioms and definitions. Not surprisingly, such procedures cannot prove difficult theorems.

A similar situation applies to the ordered field axioms, for the real numbers.

In both cases we decided to "build-in" these theories in such a way that such elementary theorems, when encountered in the proof of other theorems, are proved automatically without the need for any additional axioms and definitions.

The heart of the built-in procedure used by our group, is a set of REDUCTIONS or rewrite rules, whereby certain terms are always rewritten as other equivalent terms.

For example, one of our reductions is

$$x \in (A \cap B) \longrightarrow x \in A \ \wedge \ x \in B$$

Thus whenever an expression of the form $x \in (A \cap B)$ appears within any formula being processed by the prover, it is immediately replaced by $(x \in A \ \wedge \ x \in B)$. This a one-way thing, $(x \in A \ \wedge \ x \in B)$ is never replaced by $x \in (A \cap B)$.

The Wos et. al. group at Argonne National Laboratory, also uses such rewrite rules (they call them "demodulators") in their prover, which has achieved a great deal of success.

The prover in [1] employed a REDUCTION table of about thirty entries to prove most theorems of elementary set theory. Table 1 lists some of these, and Table 2 lists some the definitions used in those proofs.

Table 1

Some REDUCTIONS

	INPUT	OUTPUT
1.	$x \in (A \cap B)$	$x \in A \wedge x \in B$
2.	$x \in (A \cup B)$	$x \in A \vee x \in B$
3.	$A \in Sb\ B$	$A \subseteq B$ [6]
4.	$A \subseteq (B \cap C)$	$A \subseteq B \wedge A \subseteq C$
5.	$t \in \{x: P(x)\}$	$P(t)$ [6]
	...	
10.	$\emptyset \subseteq A$	TRUE
11.	$A \subseteq A$	TRUE
12.	$A \cap A$	A
13.	$P \wedge TRUE$	P
	....	

Table 2
Some DEFINITIONS

$A = B$ (set equality)	$A \subseteq B \wedge B \subseteq A$
$A \subseteq B$	$\forall t(t \in A \Rightarrow t \in B)$
...	

6. Rule 3 was actually used with the output $(A \subseteq B \wedge A \in U)$, where U is the universal set. Similarly for Rule 5.

Let us consider two examples to see the effectiveness of these reductions.

EX1. $\forall_A \forall_B (A \subseteq A \cup B)$

(1) $A \subseteq A \cup B$ — The Goal

$t_0 \in A \rightarrow t_0 \in (A \cup B)$ — Defn of $=$ [7]

$t_0 \in A \rightarrow t_0 \in A \vee t_0 \in B$ — REDUCTION Rule 2

TRUE [8]

---

7. Here it instantiates the definition of $\subseteq$. This prover [1], (see Section 3.2 below), does not automatically instantiate every definition where possible, but does so only under limited control. In this case, when it had nothing more that it could do, it instantiated (only) the main connective of the conclusion.

8. This prover, as are most, is able to detect that such expressions as $(P \rightarrow P \vee Q)$, $(P \wedge Q \rightarrow P \wedge Q)$, etc, are true.

EX2. $\forall_A \forall_B (Sb(A \cap B) = Sb(A) \cap Sb(B))$

(1) $Sb(A \cap B) = Sb(A) \cap Sb(B)$ — The Goal

$[Sb(A \cap B) \subseteq Sb(A) \cap Sb(B)] \wedge [Sb(A) \cap Sb(B) \subseteq Sb(A \cap B)]$ — Defn of $=$ [7]

(1 1) $Sb(A \cap B) \subseteq Sb(A) \cap Sb(B)$ — Subgoal 1

$t \in Sb(A \cap B) \rightarrow t \in [Sb(A) \cap Sb(B)]$ — Defn of $\subseteq$

$t \subseteq A \cap B \rightarrow t \in Sb(A) \wedge t \in Sb(B)$ — REDUCTION Rules 3,1

$t \subseteq A \wedge t \subseteq B \rightarrow t \subseteq A \wedge t \subseteq B$ — REDUCTION Rules 4,3

TRUE

(1 2) $Sb(A) \cap Sb(B) \subseteq Sb(A \cap B)$ — Subgoal 2

Proved similarly.

Pastre [33] has generalized the notion of Reductions in a prover that has proved most of theorems of 1.1 and 1.2, and others.

Notice that the REDUCTION table is a convient place to store unit facts such as $(\emptyset \subseteq A = TRUE)$, $(A \cap A = A)$, $(P \wedge TRUE = P)$, etc. Such rules can be powerful simplifiers when used within a large proof.

Actually the use of reductions rules is just a way of "substituting equals for equals", because each entry in the reduction table is an equality or an equivalence. The real savings in efficiency comes from our insistence that it substitutes only one way. (E.g., it always replaces $x \in (A \cap B)$ by $(x \in A \wedge x \in B)$ but never the other way.) This greatly reduces the combinatorics of the search, whenever such a one-way substitution can be done. When can it be done? When is it possible to replace all equality axioms of a given theory by an equivalent set of reductions without loosing completeness? This question

is addressed by the field of Complete Sets of Reductions [14-16]. For example, ordinary group theory lends itself to such a treatment [14]. See also Lankford's paper in this volume.

Reduce tables, such as the one above, have been generalized to include conditional reductions, whereby a rule is invoked only if some condition is satisfied. For example, the rule

IF P G H ~P H

would rewrite the formula (IF (0 ≤ x) 2x+1 5) as 5, if it has as a hypothesis that x<=-1.

2.2 Skolemization.

As mentioned earlier, most automatic provers require that a formula be quantifier-free when it is presented for proof. For example, the theorem

$$\forall x\ [\ P(x) \rightarrow Q(x)\ ] \wedge P(a) \rightarrow Q(a)$$

is converted to the form

$$[\ P(x) \rightarrow Q(x)\ ] \wedge P(a) \rightarrow Q(a),$$

where a is a constant and x is a variable that can be instantiated during the proof process. (In this example x is bound to the the value a). This process, which is called skolemization, can be automated for any formula in first order logic [7,chap 4; 8,Sect 1.5]. The more complicated example,

$$(\ \forall x\ \exists y\ P(x,y) \rightarrow Q\ )$$

is skolemized to

$$(\ P(x,g(x)) \rightarrow Q\ ).$$

The 'g' is a new function symbol, called a skolem function symbol. The hypothesis asserts that for each x there is a (corresponding) y for which P(x,y) . Since the y depends on x, we write it as g(x).

When instantiating a definition the computer must take account of the position in the theorem of the formula being replaced, in order to insert the proper skolemization. Thus if $(A \subseteq B)$ is replaced by $\forall x\ (x \in A \rightarrow x \in B)$ in the theorem

$$(\ H \rightarrow A \subseteq B)$$

(where H is some hypothesis), we get

$$(\ H \rightarrow (x_0 \in A \rightarrow x_0 \in B)\ ),$$

where x_0 is a new skolem constant, but in

$$(\ A \subseteq B \rightarrow C\ ),$$

we get

$$(\ (x \in A \rightarrow x \in B) \rightarrow C\ ),$$

where x is a variable which can be instantiated during the proof.

2.3 Using Induction.

A number of researchers have used induction in automatic proofs [18,1,19, 10]. The main difficulties in using induction automatically, is in determining

(1) When to use induction,

(2) What variable to induct upon, and

(3) What to use for the induction hypothesis.

For example, in the proof of 1.2.1, $\omega = \sigma\omega$, when the subgoal

(*) $\quad x \in \omega \quad \Rightarrow \quad (t \in x \Rightarrow t \in \omega)$

is encountered, the prover decides to try induction on x with the formula (*) itself as the induction hypothesis. Thus it was required to prove the two subgoals

$$(t \in 0 \Rightarrow t \in \omega)$$

and

$$x \in \omega \wedge \forall s\,(s \in x \Rightarrow s \in \omega) \longrightarrow (t \in \mathrm{scsr}\ x \Rightarrow t \in \omega),$$

which it could do.

In this example it was able to use the subgoal (*) itself as the induction hypothesis. But of course, this sometimes will not work, we often must use a generalized induction hypothesis. To discover such a needed generalized induction hypothesis remains an essentially unsolved problem for automatic provers. Except for the work of Boyer and Moore [10], no automatic prover uses any induction hypothesis other than the subgoal itself.

2.4 Simplification of algebraic expressions.

As mentioned earlier, automatic provers have difficulty coping with the ordered field axioms for the real numbers:

$$x + y = y + x \qquad x \cdot y = y \cdot x$$

$$x + (y + z) = (x + y) + z \qquad x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

when these are given as additional hypotheses of the theorem being proved, because there are so many ways in which the variables x,y,z in the formulas can match the terms of the theorem being proved. (We believe this is the reason that most efforts in Automated Theorem Proving have been in areas other than real analysis)

We have been able to avoid adding such axioms by again using REDUCTIONS, to simplify each algebraic expression to a canonical form. Such simplification is widely used in computer mathematics, especially in such systems as MACYMA, which performs automatic differentiation, integration, equation solving, limit taking, etc. See [20]. Such automatic simplification has found increased use in ATP since about 1970.

3. Types of Provers

3.1 Resolution.

As mentioned earlier there are two basic types of automatic provers in use, Resolution and Natural Deduction. Resolution [6,7,8] is a refutation method whereby a quantifier-free theorem is negated and converted to conjunctive normal form (CNF) and then shown to be unsatisfiable (inconsistent) by performing a series of simple inference steps (resolutions). For example, the theorem

$$\forall x \; (P(x) \rightarrow Q(x)) \wedge P(a) \rightarrow Q(a)$$

first has its quantifier removed

$$(P(x) \rightarrow Q(x)) \wedge P(a) \rightarrow Q(a),$$

(x is now a variable which can be instantiated), and then negated

$$(P(x) \rightarrow Q(x)) \wedge P(a) \wedge \sim Q(a),$$

and placed in CNF,

$$(\overset{1}{\sim P(x)} \vee \overset{1}{Q(x)}) \wedge \overset{2}{P(a)} \wedge \overset{3}{\sim Q(a)} .$$

The disjuncs 1,2,3, are called clauses

1. $\sim P(x) \vee Q(x)$
2. $P(a)$
3. $\sim Q(a)$

Clause 1 is resolved with Clause 2 (with substitution a/x) to obtain

4. $Q(a)$,

which is then resolved with Clause 3 to obtain

5. $\square$,

a contradiction. See the references quoted above for details, and also Reference [11] for a brief elementary introduction to Resolution and ATP. All theorems in first order logic can be proved in this way by Resolution.

3.2 Natural Deduction.

Natural Deduction is a procedure whereby the theorem being proved is manipulated by a set of production rules, in which a goal is converted to one or more subgoals [9,2, 8 Chap 6]. The initial goal is first converted

to quantifier-free form by skolemization, but not negated.

The prover in [9] is given a goal G and a hypothesis H, and is required to find and return a (most general) substitution θ for which $(H\theta \Rightarrow G\theta)$ is a tautology. Some of its rules are:

I4. $(H \Rightarrow A \wedge B)$ "SPLIT"
If $(H \Rightarrow A)$ returns θ
and $(H\theta \Rightarrow B\theta)$ returns λ
then return $\theta\circ\lambda$.
($\theta\circ\lambda$ is the composition of the substitutions θ and λ. See [7,p.76])

H2. $(H \wedge P \Rightarrow C)$ "MATCH"
If $P\theta \equiv C\theta$, return θ

H7. $H \wedge (A \Rightarrow D) \Rightarrow C$ "BACKCHAINING"
If $(D \Rightarrow C)$ returns θ
and $(H\theta \Rightarrow A\theta)$ returns λ,
then return $\theta\circ\lambda$.

...

To prove the example

$$\forall x\ (\ P(x) \Rightarrow Q(x)\ ) \wedge P(a) \Rightarrow Q(a),$$

backchaining (Rule H7) is applied to the original goal

GOAL: $(\ P(x) \Rightarrow Q(x)\ ) \wedge P(a) \Rightarrow Q(a)$

to obtain Subgoals 1 and 2.

SUBGOAL 1: $(\ Q(x) \Rightarrow Q(a)\ )$
returns $\theta = a/x$, by matching (Rule H2),

SUBGOAL 2: $(\ P(x) \Rightarrow Q(x)\ ) \wedge P(a) \Rightarrow P(x)\ (a/x)$
returns TRUE (because $P(x)\ (a/x) = P(a)$).

So a/x is returned for the original GOAL, to complete the proof.

The procedure in [9] is not complete but seems to be adequate to prove a wide variety of theorems. A similar but extended system by Loveland&Stickle [8] is complete (can prove all theorems of first order logic). See [9] for a list of other papers on Natural Deduction type provers. The theorems listed in Section 1 were all proved by Natural Deduction or Resolution type provers. Also some used various heuristics and procedures as indicated in Section 1.

3.3 Provers for Higher Order Logic.

Andrews' prover which is described elsewhere in this volume has successfully proved Cantor's Theorem 2.3.2, and a number of others. See also [21-25].

4. Limit Heuristic

The Natural Deduction prover [2] was used to prove the limit theorems of Calculus 2.1-2.6, employed a limit heuristic and a variable restriction mechanism, as well as reduction and simplification routines.

LIMIT HEURISTIC: when proving a goal of the form

$$|B| < E$$

in the presence of a hypothesis of the form

$$|A| < E'$$

(and other hypotheses H), first try to find a substitution σ, for which $B\sigma$ can be expanded as a non-trivial linear combination of $A\sigma$, i.e.,

$$[\, B = k\,A + L \,]\sigma$$

where σ is a most general substitution, and, if this is possible, then try to prove the following three subgoals:

SG1: $[\, |K| < M \,]\sigma$ for some M, $0 < M < \infty$,

SG2: $[\, |A| < E' \rightarrow |A| < E/(2\cdot M) \,]\sigma\circ\sigma_1$

SG3: $[\, |L| < E/2 \,]\sigma\circ\sigma_1\circ\sigma_2$

where σ, σ_1, σ_2, are returned from subgoals SG1, SG2, and SG3, respectively.

So, for example, in proving that the limit of a product of two functions is the product of their limits (if the two limits exist), the prover encounters the goal

$$|f(x)\cdot g(x) - L_1\cdot L_2| < E$$

in the presence of the hypothesis

$$|f(x_1) - L_1| < E_1$$

(and other hypotheses, H) where E, x, and x_1 are variables. Using the limit heuristic, it expresses $(\, f(x)\cdot g(x) - L_1\cdot L_2 \,)$ as a linear combination of $(\, f(x_1) - L_1 \,)$, as follows:

$$\underbrace{f(x)\cdot g(x) - L_1\cdot L_2}_{B} = \underbrace{g(x)}_{K}\cdot\underbrace{[\, f(x) - L_1 \,]}_{A} + \underbrace{g(x)\cdot L_1 - L_1\cdot L_2}_{L}$$

with substitution $\sigma = (x/x_1)$, and then proceeds to prove the subgoals

Sb1: $|g(x)| < M$

Sb2: $|f(x) - L_1| < E_1 \rightarrow |f(x) - L_1| < E/(2\cdot M)$

Sb3: $|g(x)\cdot L_1 - L_1\cdot L_2| < E/2$,

where x, M, and E, are variables. (See Section 5, in regard to proving these inequalities).

5. Inequality Provers

One of the most urgent needs for automatic proofs in analysis is an efficient technique for handling general inequalities. Ground inequalities are relatively easy, but those with variables to instantiate are not.

5.1 Variable Restrictions.

One such technique is due to Slagle and Norton [12]. Another is the variable restrictions of Bledsoe [13]. When faced with a subgoal of the form $(1 < x < 3)$ this prover will not instantiate x with a particular value such as 2.5, but instead will give x the "restriction <1 3>", (which simply means that x is between 1 and 3) and store this fact in the data base, leaving x as a variable to be instantiated by later subgoals. Of course any value x_0 of x obtained subsequently must satisfy the restriction $(1 < x_0 < 3)$.

For example in proving the theorem,

$$P(2.5) \Rightarrow \exists x\ (1 < x < 3 \wedge P(x))$$

the prover first encounters the subgoal

(1) $P(2.5) \Rightarrow (1 < x < 3)$

which it satisfies by giving x the restriction <1 3>, but leaving x as a variable; and then encounters the subgoal

(2) $P(2.5) \Rightarrow P(x)$

which it satisfies by giving x the value 2.5. It then finishes the proof by verifying that $1 < 2.5 < 3$, which is immediate.

This concept was used in [2] in proving the limit-of-a-product theorem, mentioned in Section 4. In proving the three subgoals Sb1, Sb2, and Sb3,(see above) the prover encounters the three subgoals

(1) $(0 < D)$,

(2) $(|x - a| < D \Rightarrow |x - a| < D_2)$,

(3) $(|x - a| < D \Rightarrow |x - a| < D_1)$,

(as well as other subgoals) where D is a variable and D_1 and D_2 are constants

The prover satisfies Subgoal (1) by giving D the restriction $<0\ \infty>$, and then updates this restriction to $<0\ D_2>$ to satisfy Subgoal (2). (It contains in its data base the facts that D_1 and D_2 have restrictions $<0\ \infty>$.) Finally, it satisfies Subgoal (3) by further updating the D restriction to $<0, \min(D_1, D_2)>$.

Several other subgoals in this proof are also satisfied by variable restrictions. This technique along with the algebraic simplification that accompanies it, helps avoid the explicit use of the inequalities axioms and the real field axioms, which tend to clutter and degrade an automatic prover.

Other methods for handling inequalities with variables are given in Section 5.2. Methods similar to these and others like Slagle and Norton's [12], have been very useful in proofs that arise in program verification [26,3

5.2 Variable Elimination and Shielding Term Removal.

Another approach to general inequalities (on the reals with variables to be instantiated), is found in [5]. This is a resolution based prover which uses

Inequality Chaining

Variable Elimination

Shielding Term Removal

These will be described shortly. The resulting prover is complete for the first order logic [27].

First a theorem is negated and converted to clausal form (see Sect. 3.1) If the resulting clauses are ground, i.e., have no variables to be instantiated, the proof is usually easy. There are a number of fast decision procedures for ground inequalities [28,29].

We will first describe variable elimination (VE) which plays a central role in the prover. Consider the clause

(1) $\quad x < a \ \vee \ b < x \ , \qquad$ (x is a variable)

which was derived by negation from the formula,

(2) $\quad \exists x \ (a \leq x \leq b)$

to be proved. Since (2) is equivalent (on the real numbers) to

(2') $\quad a \leq b$

it follows that (1) is equivalent to

(1') $\quad b < a \ .$

So we can replace (1) by (1'), thereby eliminating the variable x.

Similarly, the clauses,

$$x \leq a \ \vee \ b < a \ ,$$

$$x < a \ ,$$

$$x \leq a \ \vee \ x \leq b \ \vee \ c < x \ ,$$

can be replaced by $(b \leq a)$, $\square$, and $(c \leq a \ \vee \ c \leq b)$, respectively.

Thus in proving the simple theorem

$$a \leq b \longrightarrow \exists x \ (a \leq x \leq b),$$

it is first converted to clausal form,

1. $a \leq b$

2. $x < a \vee b < x$,

and then VE is used to obtain

3. $b < a$ 2, VE x

and then Clauses 1 and 3 are resolved to obtain

4. $\square$, 1, 3

to complete the proof.

A similar proof does not work for the three clauses

1. $g(y) \leq b \vee 1 \leq y$

2. $y' < g(y')$ $1 \leq y'$

3. $b < 1$,

because the variable y or y' cannot be eliminated from either 1 or 2. But if we first "chain" 1 and 2 on g(y) we get,

4. $y < b$ $1 \leq y$ 1,2, removing g(y)

and then VE can be applied to 4 getting,

5. $1 \leq b$, 4, VE y

6. $\square$ 3,5

The term g(y) in the above is called a "shielding term", because it shields the variable y; once all shielding terms (of a particular variable) have been removed that variable becomes eligible for elimination.

Inequality chaining [12] is simply the concept of applying the transitivity axioms

$$x \leq y \wedge y \leq z \longrightarrow x \leq z$$

$$x \leq y \quad y < z \longrightarrow x < z$$

etc

to two literals, as was done in the above examples to the literals, $g(y) \leq b$ and $y < g(y)$ to get $y < b$. Also unification is permitted, e.g., $y < g(y)$ and $g(c) \leq b$ result in $c < b$.

It is also possible to chain on the variable y' in Clause 2, but note that it can match in three ways. In fact there are 12 different chain resolvents from Clauses 1,2,3, if we allow chaining on variables, and only three if we do not. We avoid this proliferation of clauses by forbidding chaining on variables. Without chaining on variables the prover is still complete.

Random chaining on terms tends to greatly enlarge the search space, even without chaining on variables, so we have attempted to devise an

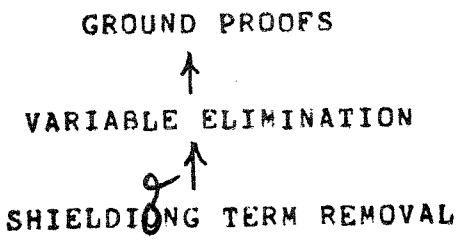
overall strategy which will guide the search. Three facts motivate this strategy

- Ground proofs are easy
- VE removes variables from clauses (makes them "more ground")
- Shielding term removal tends to make variables eligible for VE

So our strategy is as follows:

- If a ground proof is possible, do it
- If not try to eliminate a variable by VE
- If no variable is eligible for VE then try to remove a shielding term by chaining (but don't chain on variables)

This can be depicted this way



Algebraic simplification and reductions are also used in this prover.

Such a system has been shown to be complete for first order logic [27]. (Recall that any first order literal $P(x_1, x_2, \ldots, x_n)$ can be converted to an equivalent inequality literal $q(x_1, x_2, \ldots, x_n) \leq 0$.) Our implementation described in [5] uses mandatory VE, whereby VE must be applied to any clause with a variable which is eligible for VE, and the parent clause discarded (a highly desirable act since we seek to remove all variables). The completeness proof given in [27] does not allow manditory VE: it is still an open question whether the prover is complete with manditory VE.

We now give a few other examples theorems, followed by the corresponding clauses, and proofs (or parts of proofs).

@ $(\forall \varepsilon (0 < \varepsilon \rightarrow A \leq B + \varepsilon) \longrightarrow A \leq B)$

Clauses

1. $\varepsilon \leq 0 \vee A \leq B + \varepsilon$
2. $B < A$

---

3. $A \leq B$ — 1, VE ε
4. $\square$ — 2,3

@ $\forall \varepsilon [(0 < \varepsilon \rightarrow A \leq B(\varepsilon) + \varepsilon) \wedge B(\varepsilon) \leq C] \longrightarrow A \leq C$

(ε is a variable, $B(\varepsilon)$ is a function of ε).

Clauses

1. $\varepsilon \leq 0 \vee A \leq B(\dot{\varepsilon}) + \varepsilon$
2. $B(\varepsilon) \leq C$
3. $C < A$

---

4. $\varepsilon \leq 0 \vee A \leq C + \varepsilon$ 1,2
5. $A \leq C$ 4, VE ε
6. $\square$ 3,5

@ The sum of two continuous functions is continuous.

Clauses

1. $f(x_\delta) + g(x_\delta) + \varepsilon_0 < f(x_0) + g(x_0) \vee$
   $f(x_0) + g(x_0) + \varepsilon_0 < f(x_\delta) + g(x_\delta) \vee \quad \delta \leq 0$
2. $0 < \varepsilon_0$
3. $0 < \delta_\varepsilon \vee \varepsilon \leq 0$

   ...

5. $f(x_0) \leq f(y) + \varepsilon \vee \delta_\varepsilon + x_0 < y \vee \delta_\varepsilon + y < x_0 \vee \varepsilon \leq 0$

   ...

10. $x_\delta \leq x_0 + \delta \vee \delta \leq 0$

where δ, ε, ε', and y are variables, x_δ is a skolem function of δ, δ_ε is a skolem function of ε, etc.

11. $g(x_\delta) + \varepsilon_0 < g(x_0) + \varepsilon \vee f(x_0) + g(x_0) + \varepsilon_0 < f(x_\delta) + g(x_\delta)$
    $\delta \leq 0 \vee \delta_\varepsilon + x_0 < x_\delta \vee \delta_\varepsilon + x_\delta < x_0 \vee \varepsilon \leq 0$ 1,5, x_δ/y, removing $f(x_\delta)$

    ...

18. $\varepsilon_0 < \varepsilon + \varepsilon' \vee \delta \leq 0 \vee \delta_\varepsilon < \delta \vee \varepsilon \leq 0 \vee \delta'_{\varepsilon'} < \delta \vee \varepsilon' \leq 0$
19. $\varepsilon_0 < \varepsilon + \varepsilon' \vee \delta_\varepsilon \leq 0 \vee \delta'_\varepsilon \leq 0 \vee \varepsilon \leq 0 \vee \varepsilon' \leq 0$ 18, VE

    ...

24. $\square$

@ 3.12′ (From the intermediate value theorem - See Section 7).

Clauses (We will omit the $\vee$ symbol between literals of clauses)

1. $x < a \quad b < x \quad f(x) \leq 0 \quad t_x < x$
2. $\quad " \quad " \quad " \quad s \leq t_x \quad x < s \quad 0 < f(s)$
3. $\quad " \quad " \quad 0 \leq f(x) \quad x < t_x$
4. $\quad " \quad " \quad " \quad t_x \leq s \quad s < x \quad f(s) < 0$
5. $b < x \quad 0 < f(x) \quad x \leq 1$
6. $1 \leq y \quad z_y \leq b$
7. $\quad " \quad f(z_y) \leq 0$
8. $\quad " \quad y < z_y$
9. $a \leq b$
10. $f(a) \leq 0$
11. $0 \leq f(b)$
12. $0 < f(x) \quad f(x) < 0$

where x, s, and y are variables, and t_x, z_y are functions of x and y respectively.

The prover described in [5] was unable to prove this theorem. There are too many ways to match even with our strategy, But a later version by Hines (not yet published) using multi-step planning, has been able to prove this and others.

6. Proofs using Non-standard Analysis

It is well known that a number of proofs of theorems in real analysis ar made easier by the use of non-standard analysis [17]. This is also true for automatic proofs; the theorems 3.1-3.11, 2.1-2.6, were proved by the prover described in [3] by first converting them to "non-standard form" and then finishing the proofs using various properties of the non-standard concepts.

Some Concepts, Properties, and Definitions in Non-standard Analysis

$x \approx y$ means that x and y belong to the same monad [17] (i.e., "x and y are infinitely close together")

$\approx$ is an equivalence relation (in particular, it is transitive)

st(x) means the standard part of x

$st(x) \approx x$

Standard x means that x is an ordinary real number

Standard st(x)

$x \approx y \Rightarrow st(x) \approx st(y)$

S is compact iff $\forall x\ (x \in S \Rightarrow st(x) \in S)$

f is continuous at r iff $\forall y$ (Standard r $\wedge$ r $\approx$ y $\Rightarrow$ f(r) $\approx$ f(y))

f is uniformly continuous on S iff

$$\forall x \in S\ \forall y \in S\ (x \approx y \Rightarrow f(x) \approx f(y))$$

The theorem,

> If f is continuous on the compact set S, then
> f is uniformly continuous on S,

is proved by first converted it to non-standard form,

$(x \in S \Rightarrow st(x) \in S)$ Compact

$(r \in S \wedge y \in S \wedge$ Standard r $\wedge$ r $\approx$ y $\Rightarrow$ f(r) $\approx$ f(y)) Continuity

$$x_o \in S \wedge y_o \in S \wedge x_o \approx y_o \longrightarrow f(x_o) \approx f(y_o)$$

and then establishing the following facts:

$x_o \in S$, $y_o \in S$, $x_o \approx y_o$,	Given (Hypothesis 3)
$st(x_o) \in S$, $st(y_o) \in S$,	Hypothesis 1
$x_o \approx st(x_o)$, $y_o \approx st(y_o)$	Property of $\approx$
Standard $st(x_o)$, Standard $st(y_o)$	Property of $\approx$
$st(x_o) \approx st(y_o)$	Property of $\approx$, since $x_o \approx y_o$
$f(x_o) \approx f(st(x_o))$	Hyp 2, with $st(x_o)/r$, x_o/y
$\approx f(st(y_o))$	Hyp 2, with $st(x_o)/r$, $st(y_o)/y$
$\approx f(y_o)$	Hyp 2, with $st(y_o)/r$, y_o/y
$f(x_o) \approx f(y_o)$	Transitivity of $\approx$.

The prover described in [3] uses a typing mechanism (whereby entities are typed as "real", "infinitesimal", etc), and a data base of relevant facts, reductions, simplification, etc, to facilitate its proof, and others like it.

As was mentioned earlier, the prover has great success on those theorems for which it applies but is severely limited in its applicability.

7. Set Variable Instantiation

Theorems in higher order logic are in general harder to prove than first order theorems; higher order logic is incomplete and (proper) higher order instantiations are more difficult to find. When the higher order variables involved are all universally quantified, then the theorem is essentially first order, but when one or more higher order variables are existentially quantified, the proof is often much harder. Examples are: the set variable A in

$$\exists A \subseteq R \; (\; A \text{ is dense in } R \; \wedge \; (R \sim A) \text{ is dense in } R \;);$$

the function variable f in

$$\exists f \; \forall x \in R \; (\; \text{Continuous } f \; x \; \wedge \; \sim \text{differentiable } f \; x \;);$$

the family variable G in

> If F is a family of open sets covering a regular topological space X, then there exists a family G of open sets which covers X and for which the family of closures of members of G is a refinement of F;

and the set variable A in

> 3.12 (Intermediate Value Theorem)
>
> $$\forall A \; (\; A \neq \phi \; \wedge \; \text{bounded } A \; \rightarrow \; \exists l \; (\; l = \sup A \;))$$
>
> $$\wedge \; a \leq b \; \wedge \; f \text{ is continuous on } [a,b] \; \wedge \; f(a) \leq 0 \leq f(b)$$
>
> $$\longrightarrow \quad \exists x \; (\; a \leq x \leq b \; \wedge \; f(x) = 0 \;)$$

Andrews' prover, described elsewhere in this volume , is ideally suited for such theorems. See also [21-25]. These provers, though powerful in concept, have yet to be developed to the point where they can prove even moderately hard theorems in higher order logic. (Though there are a number of interesting exceptions). Therefore we have looked to special ad hoc methods to handle a subset of higher order logic, those theorems which require the instantiation of a set variable.

So given a theorem of the form

$$\exists A \; P(A)$$

we desire to give a "value" to A of the form

$$A = \{x: \; Q(x) \; \},$$

where Q(x) is described in terms of the symbols of P(A). The central concept in this work is to determine this value for A (i.e., the description of Q) in a series of steps, by keying on subformulas of the form

(1) $\qquad (\; x \in A \; \rightarrow \; P(x) \;)$

and

(2) $t \in A$

and others, within the theorem. Each of these triggers the building of a partial description of Q (e.g., {x: P(x) } for (1)), and these partial descriptions are combined to make up the complete description.

The basic notion for this is due to Darlington [30] and Bledsoe [4], and is also closely related to some earlier work of Behmann [31] on a decision procedure for monatic logic. See also [36,37]

Once a description, {x: Q(x) } has been obtained, the symbol "A" in the theorem is replaced by {x: Q(x) }, and the resulting first order theorem is proved. Examples 3.12, 4.1-4.2, and others were proved using these methods by the prover described in [4].

When this prover was applied to the intermediate value theorem 3.12 (see above), the value

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

was obtained, which, when substituted for A in the theorem, resulted in the new (first order) theorem

3.12′ $(LUB \wedge L1 \wedge L2 \wedge a \leq b \wedge f(a) \leq 0 \leq f(b) \longrightarrow \exists x (a \leq x \leq b \wedge f(x) \leq 0 \leq f(x)))$

where LUB, L1, and L2 are as follows

LUB: $([\exists r (r \leq b \wedge f(r) \leq 0) \wedge \exists u \forall t (t \leq b \wedge f(t) \leq 0 \rightarrow t \leq u)] \longrightarrow \exists l [\forall x (x \leq b \wedge f(x) \leq 0 \rightarrow x \leq l) \wedge \forall y (\forall z (z \leq b \wedge f(z) \leq 0 \rightarrow z \leq y) \rightarrow l \leq y)]$;

L1: $\forall x (a \leq x \leq b \wedge 0 < f(x) \rightarrow \exists t (t < x \wedge \forall s (t < s \leq x \rightarrow 0 < f(s))))$;

L2: $\forall x (\quad " \quad f(x) < 0 \rightarrow \exists t (x < t \wedge \forall s (x \leq s < t \rightarrow f(s) < 0))$;

(In this proof we have used the continuity lemmas L1 and L2 instead of the full definition of continuity of f).

While 3.12′ is of first order, it too is a difficult theorem for automat: provers (and fairly hard for humans).

During the first pass when the value {x: $x \leq b \wedge f(x) \leq 0$ } was obtain for A, the prover also obtained the binding l/x. Of course the proof of 3.1: becomes much easier when this value, l, is substituted for x. In fact the general-inequality prover [5] described in Section 5 can easily prove 3.12′ when x is replaced by l and cannot when x is left as a variable. (A more recent version by Hines (unpublished) has proved 3.12′).

It is interesting to note that the original theorem, 3.12, though of higher order, is nevertheless, easier to prove than its first order derivate, 3.12′, provided that both the instantiation for A and and that for x are used in the second pass. This is consistent with human behavior; most mathematicians find 3.12 easier to prove directly than 3.12′.

8. Remarks

Much remains to be done if we are to have automatic provers which even begin to compete with their human counterparts. We feel that such power will not become available until we begin to incorporate yet other concepts, such as

Analogy

The use of examples (as counterexamples, and as aids in discovering the proof) [32]

Use of special cases

Conjecturing (see Lenat's paper in this volume)

and much better overall planning (agenda mechanisms, etc.). See [9],

References.

1. Bledsoe, W.W. (1971): Splitting and Reduction Heuristics in Automatic Theorem Proving. Artificial Intelligence 2, No. 1, pp.55-77.

2. Bledsoe, W. W., Boyer, R. S., Henneman, W. H., Computer proofs of limit theorems. Artificial Intelligence 3, 1971, 27-60.

3. Ballantyne, A. M., Bledsoe, W. W., Automatic proofs of theorems in analysis using non-standard techniques, JACM 24 (1977) 353-374.

4. Bledsoe, W. W., A maximal method for set variables in automatic theorem proving, Machine Intelligence 9, (eds. J.E. Hayes, D. Michie, L.I. Mikulich) Ellis Harwood Lim., Chichester, 1982, 53-100.

5. Bledsoe, W. W., Hines, L. M., Variable Elimination and chaining in a resolution-based prover for inequalities", Proc. 5th Conference on Automated Deduction, Les Arcs, France, July 8-11, 1980, (Eds. W. Bibel, R. Kowalski), Springer-Verlag, pp. 70-87.

6. Robinson, J.A., A machine oriented logic based on the resolution principle, J. ACM 12, 1965, 23-41.

7. Chang, C.L., Lee, R.C.T, Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973.

8. Loveland, D.W., Automated Theorem Proving: A Logical Basis, North-Holland, 1978.

9. Bledsoe, W.W., Non-Resolution Theorem Proving, Artificial Intelligence 9, 1977, 1-35.

10. Boyer, R.S., Moore, J S., A Computational Logic, Academic Press, 1979.

11. Automatic Theorem Proving, in What Can Be Automated, Ed. Arden, B.W., MIT Press, 1980, 448-462.

12. Slagle, J.R., Norton, L., Experiments with an automatic theorem prover having partial ordering rules, CACM 16, 1973, 682-688.

13. Bledsoe, W.W., Bruell, P., Shostak, R., A prover for general inequalities Univ of Texas Math Dep Memo ATP-40, 1978, also IJCAI-79.

14. Knuth, D., Bendix, P., Simple word problems in universal Algebras, in Computational problems in Abstract Algebra Ed. Leech, J., Pergamon Press, 1970, 263-297.

15. Lankford, D., Ballantyne, A.M., Univ of Texas Math Dep Memo's 35,37-39, 1977.

16. Huet, G., Oppen, D., Equations and Rewrite Rules: A Survey, Tech Report CSL-111, SRI-international, Jan 1980.

17. Robinson, Abraham, Nonstandard Analysis, North Holland, 1966.

18. Darlington, J.L., Automatic theorem proving with equality substitutions and mathematical induction, Machine Intelligence 3, 113-127.

19. Burstall, R.M., Properties of Programs by Structural Induction, Comput. J. 12, 41-48, 1969.

20. See Algebraic Computation, in What Can Be Automated, Ed. Arden, B.W., MIT Press, 1980, 513-526.

21. Andrews, P.B., Resolution in Type Theory, J. Sym. Logic 36, 1971, 414-432.

22. Huet, G.P., A unification Algorithm for typed Lambda-Calculus. Theoretical Computer Science 1, 1975, 27-57.

23. Huet, G.P., A Mechanization of Type Theory, IJCAI-73, Stanford, 1973, 139-146.

24. Darlington, J.L., A Partial Mechanization of second-order logic, Machine Intelligence 6, 1971, 91-100.

25. Pietrzykowski, T., A complete mechanization of second order logic, J. Assoc. Comp. Mach. 20, 1973, 333-364.

26. Good, D.I., London, R.L., Bledsoe, W.W., An interactive verification system, Proc. 1975 Intl Conf on Reliable Solfware, Los Angeles, Ca, 1975, 482-492.

27. Bledsoe, W.W., Kunen, K., Shostak, R., Completeness Proofs for Inequality Provers, Univ of Texas Math Dept Memo Atp-65, 1982.

28. Nelson, G., Oppen, D., A Simplifier Based on Efficient Decision Algorithms, Proc 5th ACM Sym. on Principles of Programming Languages, 1978.

29. Shostak, R., A Practical Decision Procedure for Arithmetic with Function Symbols, JACM, 1979.

30. Darlington, J.L., Deductive Plan Formation in Higher Order Logic, Machine Intelligence 7, 1972, 129-137.

31. Behmann, H., Beitrage Zur Algebra Der Logik: Insbesondere Zum Enischeidungsproblem, Mathematische Annalen, 86, 163-229.

32. Ballantyne, A.M., Bledsoe, W.W., On generating and using examples in proof discovery, Machine Intelligence 10, 1982, 3-39.

33. Pastre, D., Demonstration Automatique de Theoremes en Theorie des Ensenbles, Phd Thesis, U. Paris 6, 1976.

34. Suzuki, N., Verifying Programs by Algebraic and Logical Reduction. Proc. Int'l Conf. on Reliable Software, IEEE, 1975, 473-481.

35. Tyson, M., APRVR: A priority-ordered Agenda prover, PhD Dissertation Univ. of Texas, CS Dept., 1981.

36. Minor, J.T., Proving a subset of Second-order logic with first-order proof procedures. PhD Dissertation, Univ. of Texas, CS Dept., 1979.

37. Ferro, A., Omodeo, E.G., Schwartz, J.T., Decision Procedures for Elementary Sublanguages of Set Theory. (NYU).