

A PROVER FOR GENERAL INEQUALITIES

W. W. Bledsoe
Peter Bruell
Department of Mathematics
The University of Texas at Austin
Austin, Texas 78712

Robert Shostak
SRI International
333 Ravenswood Avenue
Menlo Park, California 92025

A variation of an earlier prover (described in Machine Intelligence 8) is used to prove theorems about general inequalities, i.e., first-order logic with equality where the only predicate symbols are $\leq$, $<$, and $=$, and where function symbols are admitted. Transcripts of some proofs are given.

1. INTRODUCTION

This paper, which is an abridgement of [1], describes a program that has been used to prove a number of first-order theorems involving inequalities (and equalities). It is a variation of an earlier natural deduction prover [2]. The class of formulas dealt with permits arbitrary quantification and uninterpreted function symbols and can therefore encode predicate calculus. Our intention, however, was not to provide a general-purpose prover for first-order logic but rather to be able to prove naturally arising formulas involving inequalities. We have made use, where possible, of previously developed techniques. Among these are decision procedures for ground Presburger formulas [3-8] and the restriction variable methods of [9]. We emphasize that this prover is completely automatic and that all examples were proved without recourse to human interaction. So far as we are aware, this prover is the first one able to prove formulas of the kind typified by the examples in an automatic way.

2. THE PRINCIPAL PARTS

Because of space limitations we are forced to refer the interested reader to [1] for the descriptions of the algorithms that the program employs. However, we must mention that the heart of the program is the routine IMPLY, which coordinates the search for the proof of a theorem. The guiding principle of this search is the retention of the implication symbol as the main connective of the theorem being proved.

Without going into detail we also note that

when the conclusion of a theorem is an inequality, IMPLY causes a function named PROVE-LE to be called. This function in turn calls several subfunctions:

1. LESS=

A quick routine designed to solve such trivial subgoals as $4 \leq 5$, $6 < \infty$, etc.

2. PROVE-LE-GROUND-CASE

Called if the conclusion is a ground inequality in an attempt to establish whether this inequality follows from ground inequalities in the hypothesis (stored in a special data structure called the TYPELIST).

3. RESTRICTION-LE

Called if the conclusion is a non-ground inequality which can be "solved" by simply placing restrictions on variables which occur in it.

4. MATCH-IN-TYPELIST

Called when the conclusion is a non-ground inequality (which cannot be "solved" by RESTRICTION-LE) in an attempt to "solve" it by finding a match with a ground inequality stored on the TYPELIST.

5. MATCH-LE

Called if the conclusion is an inequality and none of the above techniques succeeds (or applies) in an attempt to use the hypotheses of the theorem other than those on the TYPELIST.

In the section of examples we use the following abbreviations for these functions:

GLE for PROVE-LE-GROUND-CASE

RLE for RESTRICTION-LE
 MTY for MATCH-IN-TYPELIST
 MLE for MATCH-LE

Data Base

A data base, DB, is employed which is "dynamic" or "contextual," in that it may change as the proof progresses. It has three parts: A-UNIT, RESTRICTION-LIST (or RL), and TYPELIST (or TY). The A-UNIT is not used here (see [9]).

RESTRICTION-LIST (or RL) is the place where restrictions on variables are recorded. If, for example, an entry

(1) ('int' x ($\leq$ a) ($\leq$ b))

is present in RL, it means that the variable x (which is to be instantiated, or bound to a value) has not yet been given a definite value but has been restricted to reside in the interval $a < x \leq b$. If later x is bound to a particular value c, then c must satisfy this same constraint, $a < c \leq b$. Other restrictions like $a \leq x \leq b$, $a \leq x < b$, etc. are possible.

TYPELIST (or TY) is the place where ground inequality information is stored. If an entry

(2) ('int' x ($\leq$ a) ($\leq$ b))

is present in TY it means that one of our hypotheses is $a \leq x \leq b$. Notice the fundamental difference between this and (1) of RL. In RL, x is a variable to be instantiated, whereas in (2) it is a constant. RL represents knowledge about the "solution" to their variables (i.e., the bindings for the variables), whereas TY represents given knowledge or hypothesis knowledge that can be used to obtain the solution. Getting a contradiction in TY is desirable in that it completes the proof of the current subgoal, whereas a contradiction in RL is undesirable in that it indicates a failure to find an acceptable solution (binding) for x. An entry like (1) represents a whole interval of acceptable solutions for x (provided that it can be proved that $a < b$).

Other concepts such as REDUCTIONS, controlled backtracking, and algebraic simplification are used to advantage.

3. EXAMPLES

In the following examples we will depict a call to IMPLY as follows:

(TL) ([TY] $\wedge$ H $\Rightarrow$ C) (RL)

Here the theorem label, TL, a map for keeping track of subgoals, is in the left margin. The

TYPELIST, TY, being part of the hypothesis, is shown with H to the left of the arrow implying C. The restriction list, RL, is shown to the right. In the descriptions of proofs of the examples we will often leave out some of the steps when clarity is not impaired. Also, the components TL, TY, H, C, and RL will not all be given at each step.

The theorem label, TL, starts as an empty list () and grows as the depth of the proof increases. It consists of a sequence of symbols representing the actions that have been taken. Some of the symbols used in TL are

1	first branch of an and-split
2	second branch of an and-split
P $\rightarrow$	promote
CHECK	check that a proposed binding is consistent with RL

In the following examples all symbols a, b, c, f, g, l, etc. that are not explicitly quantified will be treated as skolem constants.

Ex. 1. $(f(l) < 0 \wedge 0 \leq f(b) \wedge b \leq l \rightarrow b < l)$

Proof.

() $(f(l) < 0 \wedge 0 \leq f(b) \wedge b \leq l \rightarrow b < l)$

(P $\rightarrow$) $([f(l) < 0 \wedge 0 \leq f(b) \wedge b \leq l] \Rightarrow b < l)$

PROVE-LE is called, which calls LESS= and RLE, which fail.

PROVE-LE then calls the ground prover GLE which succeeds as follows:

The $\sim(b < l) \equiv (l \leq b)$ is inserted into TY which becomes

$[f(l) < 0 \wedge 0 \leq f(b) \wedge b \leq l \wedge l \leq b]$

CONTRADICTION is called but it finds no contradiction in TY; however, it does find and return the equality unit ($b = l$), which is then applied to TY to obtain

$[f(l) < 0 \wedge 0 \leq f(l)]$

which has a contradiction. Q.E.D.

Ex. 2. $f(l) < 0 \wedge 0 \leq f(b) \wedge l < c \wedge b \leq l \rightarrow$

$\exists y [\forall z (z \leq b \wedge f(z) \leq 0 \rightarrow z \leq y) \wedge y < l]$

(P $\rightarrow$) $([f(l) < 0 \wedge 0 \leq f(b) \wedge l < c \wedge b \leq l]$

$\Rightarrow (Zy \leq b \wedge f(Zy) \leq 0 \rightarrow Zy \leq y) \wedge y < l)$

Note: y is a variable, Zy is a skolem function

of y , and the other letters represent constants.

$$(P \rightarrow 1) \quad ([f(\ell) < 0 \wedge 0 \leq f(b) \wedge \ell < c \wedge b \leq \ell] \\ \Rightarrow (Zy \leq b \wedge f(Zy) \leq 0 \rightarrow Zy \leq y))$$

$$(P \rightarrow 1 P \rightarrow) \quad ([f(\ell) < 0 \wedge 0 \leq f(b) \wedge \ell < c \wedge b \leq \ell] \\ \wedge Zy \leq b \wedge f(Zy) \leq 0 \Rightarrow Zy \leq y)$$

LESS=, RLE, GLE, and MTY fail.

MLE: $Zy \leq b$: b/y

Note: The substitution b/y has succeeded on subgoal $(P \rightarrow 1)$; prover will not proceed to subgoal $(P \rightarrow 2)$ with y replaced by b .

$$(P \rightarrow 2) \quad ([f(\ell) < 0 \wedge 0 \leq f(b) \wedge \ell < c \wedge b \leq \ell] \\ \Rightarrow b < \ell)$$

GLE inserts $\sim(b < \ell)$ into TY which becomes

$$[f(\ell) < 0 \wedge 0 \leq f(b) \wedge \ell < c \wedge b \leq \ell \wedge \ell \leq b]$$

CONTRADICTION discovers the equality $(b = \ell)$ in TY, getting

$$[f(\ell) < 0 \wedge 0 \leq f(\ell) \wedge \ell < c]$$

which holds a contradiction. Q.E.D.

Ex. 3. $(a \leq 2 \leq b) \rightarrow \exists x (0 \leq x \leq 5 \wedge a \leq x)$

$$() \quad (a \leq 2 \wedge 2 \leq b \rightarrow (0 \leq x \wedge x \leq 5) \wedge a \leq x)$$

$$(P \rightarrow) \quad ([a \leq 2 \wedge 2 \leq b] \Rightarrow (0 \leq x \wedge x \leq 5) \wedge a \leq x)$$

$$(P \rightarrow 1) \quad ([\quad] \Rightarrow (0 \leq x \wedge x \leq 5))$$

$$(P \rightarrow 1 1) \quad ([\quad] \Rightarrow 0 \leq x)$$

PROVE-LE is called which calls LESS= which fails.

PROVE-LE then calls RLE which succeeds with

$$\{x: 0 \leq x < \infty\}$$

$$(P \rightarrow 1 2) \quad ([\quad] \Rightarrow x \leq 5)$$

$$\text{RLE} \quad \{x: 0 \leq x \leq 5\}$$

$$(P \rightarrow 2) \quad ([a \leq 2 \wedge 2 \leq b] \Rightarrow a \leq x)$$

$$\text{RLE} \quad \{x: (\max 0 a) \leq x \leq 5\}$$

Here RLE has intersected the interval $\{x: a \leq x < \infty\}$ with $\{x: 0 \leq x < 5\}$ to obtain the desired result. However, before it can return this answer it must verify that it is not

empty; i.e., that $a \leq 5$.

$$(P \rightarrow 2 \text{ CHECK}) \quad ([a \leq 2 \wedge 2 \leq b] \Rightarrow a \leq 5)$$

PROVE-LE calls GLE, which inserts $\sim(a \leq 5) \equiv$

$(5 < a)$ into TY to get

$$[5 < a \leq 2 \wedge 2 \leq b]$$

which has a contradiction (by routine CONTRADICTION). Q.E.D.

The reader is referred to [1] for a description of the proofs, given by the prover, of Examples 4-6 below. LUB, L1, and L2 are lemmas used in these proofs.

$$\text{LUB:} \quad ([\exists u \forall t(t \leq b \wedge f(t) \leq 0 \rightarrow t \leq u) \\ \wedge \exists r(r \leq b \wedge f(r) \leq 0)] \rightarrow \\ [\forall x(x \leq b \wedge f(x) \leq 0 \rightarrow x \leq \ell) \\ \wedge \forall y(\forall z(z \leq b \wedge f(z) \leq 0 \rightarrow z \leq y) \\ \rightarrow \ell \leq y)])]$$

$$\text{L1:} \quad \forall x[a \leq x \leq b \wedge 0 < f(x) \rightarrow \\ \exists t(t < x \wedge \forall s(t < s \leq x \rightarrow 0 < f(s)))] \\ \text{L2:} \quad \forall x[a \leq x \leq b \wedge f(x) < 0 \rightarrow \\ \exists t(x < t \wedge \forall s(x \leq s < t \rightarrow f(s) < 0))]$$

$$\text{Ex. 4.} \quad (\text{LUB} \wedge f(\ell) < 0 \wedge 0 \leq f(b) \wedge \\ \forall s(0 \leq f(s) \wedge \ell \leq s \rightarrow t \leq s) \rightarrow t \leq \ell).$$

$$\text{Ex. 5.} \quad (\text{LUB} \wedge \text{L1} \wedge \text{L2} \wedge a \leq b \wedge f(a) \leq 0 \\ \wedge 0 \leq f(b) \rightarrow 0 \leq f(\ell)).$$

$$\text{Ex. 6.} \quad (\text{LUB} \wedge \text{L1} \wedge \text{L2} \wedge a \leq b \wedge f(a) \leq 0 \\ \wedge 0 \leq f(b) \rightarrow \exists x(f(x) \leq 0 \wedge 0 \leq f(x))).$$

4. REMARKS

The authors would be interested in hearing how well other provers are able to handle these examples, especially Ex. 4-6.

One of the reasons for our success here was our use of devices such as an algebraic simplifier to avoid the explicit use of the axioms of the real number system. Our special handling of inequalities also attempted such a saving by avoiding such axioms as the transitivity of $\leq$,

$(x \leq y \wedge y \leq z \rightarrow x \leq z)$, but much more needs to be done if we are to smoothly and economically handle the "low level" calculations associated with proofs in analysis. For example, special techniques are needed for absolute values, especially when they occur in connection with inequalities.

We don't think of this prover as ultimate in any sense. At best it will become a part of a more powerful prover we are now trying to build, which will include many of the features mentioned in [10]. For instance, recent work indicates that the use of automatically generated counterexamples can greatly speed up the proofs of these examples and others like them.

ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation Grants MSC 77-20701 at The University of Texas at Austin and MCF 76-81425 at SRI International.

REFERENCES

- [1] Bledsoe, W. W., P. Bruell, and R. Shostak. "A Prover for General Inequalities." Math. Dept. Memo ATP-40A, Univ. of Texas at Austin, February, 1979.
- [2] Bledsoe, W. W., and Mabry Tyson. "Typing and Proof by Cases in Program Verification." Machine Intelligence 8 (1977).
- [3] Shostak, Robert. "A Practical Decision Procedure for Arithmetic with Function Symbols." JACM, April, 1979.
- [4] Nelson, Greg, and Derek Oppen. "A Simplifier Based on Efficient Decision Algorithms." In Proc. 5th ACM Symp. on Principles of Programming Languages, 1978.
- [5] Suzuki, Norihisa. "Verifying Programs by Algebraic and Logical Reduction." In Proc. of Int'l Conf. on Reliable Software, IEEE, 1975, 473-481.
- [6] King, J. C. "A Program Verifier." Ph.D. thesis, Carnegie-Mellon University, 1969.
- [7] Hodges, Louis. "Solving Problems by Formula Manipulation in Logic and Linear Inequality." In Proc. IJCAI-71, London, 1971, 553-559.
- [8] Cooper, D. C. "Programs for Mechanical Program Verification." Machine Intelligence 6 (1971), 43-59.

[9] Bledsoe, W. W. "A Maximal Method for Set Variables in Automatic Theorem Proving." Machine Intelligence 9 (1979).

[10] Bledsoe, W. W. "Non-resolution Theorem Proving." AI Jour. 9 (1977), 1-35.