ON AUTOMATIC GENERATION OF COUNTEREXAMPLES

by

W. W. Bledsoe, A. Michael Ballantyne

July 1979

ATP-44A
(Revised ATP-44)

This work was supported in part by NSF Grant
MCS77-20701 at The University of Texas at Austin

On Automatic Generation of Counterexamples

W. W. Bledsoe, A. Michael Ballantyne

Abstract

In this paper we discuss the counterexample problem, point out the need for an unskolemization procedure, describe some methods for the automatic generation of counterexamples for real function variables, and give the results of some experiments.

Extended Abstract

In proving theorems it is often valuable to show that a certain proposed subgoal is invalid, as, for example, when one tries to use an inappropriate lemma, and a counterexample can often be used to show this invalidity. Such a process was used by Gelernter in his geometry prover [5] and has been proposed and used in a limited way by others. (See [11].)

In actuality, what we do when we "give a counterexample" is to <u>prove the</u> negation of the fully qualified theorem. For example, if we want to give a counterexample to the formula

(1) $(\text{Continuous } f[a,b] \to \exists x(a \le x \le b \wedge f(x) = 0))$

we first quantify (1)

(2) $\forall f\ \forall a\ \forall b\ (\text{Continuous } f[a,b] \to \exists x\,(a \le x \le b \wedge f(x) = 0)$

and prove its negation,

$$\exists f\ \exists a\ \exists b\ (\text{Continuous } f[a,b] \wedge \forall x \sim (a \le x \le b \wedge f(x) = 0))$$

So finding a counterexample of (1) will, in this case, require finding an instantiation for the <u>variables</u> f, a, and b, in (3). Here, any function f which is continuous but never 0 will suffice, if we give a and b values so that $a \leq b$. (e.g., $a = 0$, $b = 1$, $f = \lambda x 1$).

Notice that (3) is a "higher-order" theorem, whereas (1) is first order (because f can be treated as a constant function in (1). It is often the case that the "counterexample theorem" requires the instantiation of a higher-order variable (such as f in the above example), but it also seems to be the case that such a higher-order variable is often easy to obtain.

In this paper we will discuss the counterexample problem and show (by examples) that an efficient counterexample generator can save much time in automatic theorem proving. We will describe an "unskolemization" process and why it is desirable to obtain a fully quantified form from a skolemized formula. And finally, we will discuss several approaches for generating counterexamples. In this we will emphasize "function" counterexamples for the real line; i.e., the automatic proof of theorems of the type (3) above where the existence of one or more functions of a real variable is required.

Unskolemization

The reason for finding a counterexample to a formula F is to show that it is invalid, i.e., that some interpretation of its symbols is false. It is not enough to try to show that F is false since many such F's are neither true nor false. But, as was indicated in example (1) above, if we first fully quantify F, we can achieve the desired result by proving the negation of F.

Since most automatic provers employ a skolemized (quantifier-free) form, it is often necessary to "unskolemize" F, to regain its full quantification before attempting to prove its negation. Unskolemization seems to be a simple task, and indeed it is when the formula F being unskolemized is the whole original theorem. But when F is obtained from all or part of the original theorem by using a lemma, it sometimes contains a skolem function with different arguments in different parts of it, and in such a case the original quantification cannot be recaptured. For example, the skolemized formula,

$$\begin{aligned}&[a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b)\\ &\quad \wedge (s \leq t_x \vee x < s \vee 0 < f(s))\\ &\quad \rightarrow f(t_b) \leq 0 \wedge \ell < t_b].\end{aligned} \tag{4}$$

which is part of a formula derived from Example 7 or [3], has the skolem function t occurring with both arguments x and b. (a, b, f, and ℓ are constants.) It is unskolemized to

$$\begin{aligned}&\forall g\ \forall f\ \forall a\ \forall b\ \forall \ell\\ &[a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b)\\ &\quad \wedge \forall x\ \forall s\ (s \leq g(x) \vee x < s \vee 0 < f(s))\\ &\quad \rightarrow f(g(b)) \leq 0 \wedge \ell < g(b)].\end{aligned} \tag{5}$$

To find a counterexample to (4), we prove the negation of (5), namely,

$$\begin{aligned}&\exists g\ \exists f\ \exists b\ \exists \ell\\ &[a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b)\\ &\quad \wedge \forall x\ \forall s\ (s \leq g(x) \vee x < s \vee 0 < f(s))\\ &\quad \wedge (0 < f(g(b)) \vee g(b) \leq \ell)].\end{aligned} \tag{6}$$

For example, the following values will satisfy (6):

$$a = 0,\ b = 1,\ \ell = 0$$

$$f = \lambda x x,\ g = \lambda x\,\frac{x}{2}\ .$$

Methods for Generating Counterexamples

Once the formula F has fully quantified and negated, we still have the task of proving the resulting formula. As was mentioned earlier, that is often a theorem in higher-order logic. We will consider here only those theorems where one or more functions of a real variable are required.

There are several approaches one could take in proving these higher-order theorems.

One could use a higher-order prover such as those proposed by Huet [7]. Andrews [1], Darlington [4], Pietrzykowski [10], and Haynes and Henschen [6], and this seems to offer great hope in the long run, especially as more powerful methods are developed and special heuristics are used.

Another approach is to use the reduction method of Minor [9], which is an extension of methods of Behmann and Ackermann, to reduce the higher-order theorems to first order, and use existing first-order provers. This seems to be promising for an interesting class of examples.

Still another approach is a "finite specification" method, which uses the fact that a function f (even a continuous function) can always be found whenever its values are specified at only a finite number of points. For example, there is a (continuous) function satisfying the specifications $f(0) = 1$ and $f(2) = 0$. These specifications can be generalized (to some extent) to include inequalities (e.g., $f(0) \leq 0$, $f(2) > 0$), variables (e.g., $f(a) \leq 0$, $f(b) \geq c$), and intervals (e.g., $\forall x(0 \leq x \leq 1 \rightarrow f(x) < 0)$, $f(2) > 0$). In the case of continuous functions, specifications on intervals must satisfy compatible end conditions (for example, $\forall x(0 \leq x < 1 \rightarrow f(x) < 0)$ is not compatible with $f(1) > 0$).

And finally, another approach is to employ a "data base of examples" and a mechanism for selecting out of it a counterexample suitable for the purpose at hand. Michener's ideas [8] should be useful in this approach. We believe that a mixture of these will prove useful in the near future.

These approaches can be divided into two classes: those that produce actual examples for the function in question (e.g., $\lambda x x$ for f, or $\lambda x(-1+2x)$ for f); and those that require an auxiliary (first-order) theorem to be proved. When an actual example can be found, it greatly simplifies the process, but it is not always easy to find an example satisfying the given restraints.

For example, if in the theorem

$$(a \neq b \rightarrow \exists f(f(a) = 0 \wedge f(b) \neq 0)) \tag{7}$$

we put f = (if x = a then 0, else 1), then (7) becomes

$$(a \neq b \rightarrow 0 = 0 \wedge [(b = a \rightarrow 0 \neq 0) \wedge (b \neq a \rightarrow 1 \neq 0)]) \tag{8}$$

or

$$(a \neq b \rightarrow b \neq a).$$

And, if in (6) we put $f = \lambda x x$, and $g = \lambda x \frac{x}{2}$, then (6) becomes

$$\begin{aligned} &\exists a \ \exists b \ \exists \ell \ [a \leq b \wedge a \leq 0 \wedge 0 \leq b \\ &\wedge \forall x \ \forall x \ (s \leq \tfrac{x}{2} \wedge x < s \wedge 0 < s) \\ &\wedge (0 < \tfrac{b}{2} \vee \tfrac{b}{2} \leq \ell)] \end{aligned} \tag{9}$$

which is considerably easier than (6). But how is the program to know to choose these particular values of f and g? Minor's method does not experience this difficulty in finding f and g, but it does have the difficulty that its resulting first-order theorems (see (10) and (11) below) are harder than (8) and (9).

Minor's method converts (7) to

(10) $$(a \neq b \rightarrow \forall x \ \exists u \ \ (u = 0 \vee x = a) \wedge (u \neq \vee \ x \neq b) \ ),$$

and converts (6) to

(11) $$\begin{aligned} &\exists a \, \exists b \ \exists \ell \ \forall s \ \ \forall x \ \exists u \ \ \exists v \ [a \leq b \\ &\wedge (u \leq 0 \vee s \neq a) \wedge (0 \leq u \vee s \neq b) \\ &\wedge (s \leq v \vee x < s \vee 0 < u) \\ &\wedge (0 < u \vee s \neq v \vee x \neq b \vee v \leq \ell) \\ &\wedge D(u,s) \wedge D(v,x)\,], \end{aligned}$$

where D is Minor's "dependency" relation, which is characterized by a set of axioms. (See [9].)

The finite restrictions method converts (7) to

$$(a \neq b \rightarrow (\mathrm{TRUE} \wedge a \neq b))$$

or

$$(a \neq b \rightarrow a \neq b).$$

It converts (6) to (12) by eliminating f,

(12) $$\begin{aligned} &\exists g \ \ \exists a \ \ \exists b \ \ \exists \ell \\ &[a \leq b \wedge \mathrm{TRUE} \wedge \mathrm{TRUE} \\ &\wedge \forall s \ \forall x \ \ (s \ \leq g(x) \vee x < s \vee a \neq s) \\ &\wedge (a \neq g(b) \vee g(b) \leq \ell)\,]. \end{aligned}$$

If (12) is simplified to the equivalent form,

(13) $$\begin{aligned} &\exists g \ \ \exists a \ \ \exists b \, \exists \ell \\ &[a \leq b \wedge \forall x \ \ (a \leq g(x) \vee x < a) \\ &\wedge (a < g(b) \vee g(b) < a \vee g(b) \leq \ell]. \end{aligned}$$

then the finite restrictions method can further convert it to (14) by eliminating g,

$$\exists a \;\exists b \;\exists \ell$$

$$(a \leq b \wedge [(\mathrm{TRUE} \wedge \forall x \;(x < a))$$

$$(\mathrm{TRUE} \wedge \forall x \;(b \neq x \vee a < a \vee x < a))$$

$$(\mathrm{TRUE} \wedge \forall x \;(b = x \vee a \leq \ell \vee x < a))],$$

which can be further simplified to

$$\exists a \;\exists b \;\exists \ell \;(a \leq b \wedge [\mathrm{FALSE} \vee b < a \vee a \leq \ell \vee b < a])$$

or

$$\exists a \;\exists b \;\exists \ell \;(a \leq b \wedge (b < a \vee a \leq \ell)).$$

<u>Implementation</u>

Programs for unskolemizing and for Minor's first-order conversion have been implemented in LISP on the DEC-10. A version of the finite specification method has been implemented and used for the examples given in [12].

References

1. Peter Andrews and Eve Cohen. Theorem Proving in Type Theory. Proc. IJCAI-77, Cambridge, Mass., August 1977, p. 566.

2. A. Michael Ballantyne. Some Notes on Computer Generation of Counterexamples in Topology. University of Texas, Mathematics Department Memo ATP-24, 1975.

3. W. W. Bledsoe, Peter Bruell, and Robert Shostak. A Prover for General Inequalities. University of Texas, Mathematics Department Memo ATP-40, 1978.

4. Jared Darlington. Improving the Efficiency of High Order Unification. Proc. IJCAI-77, Cambridge, Mass., August 1977, pp. 520-525.

5. H. Gelernter. Realization of a Geometry Theorem-Proving Machine. Proc. Int. Conf. Information Processing, Paris UNESCO House, 1959, pp. 273-282.

6. L. J. Henschen and G. Haynes. Splitting in Unification in Higher Order Theorem Provers, Department of Computer Sciences, Northwestern University Evanston, Illinois, 1978.

7. G. P. Huet. Experiments with an Interactive Prover for Logic with Equality. Report 1106, Jennings Computing Center, Case Western Reserve University.

8. Edwina R. Michener. The Structure of Mathematical Knowledge. MIT-AI Tech Report 472, August 1978.

9. John T. Minor, III. Proving a Subset of Second-order Logic with First-order Proof Procedures. Ph.D. dissertation, Department of Computer Sciences, University of Texas, Austin, July 1979.

10. T. Pietrzykowski. A Complete Mechanization of Second Order Type Theory. J. ACM, 1973, pp. 333-364.

11. R. Reiter. A Semantically Guided Deductive System for Automatic Theorem Proving. Proc. Third International Joint Conference Artificial Intelligence, 1973, pp. 41-46; IEEE Trans. on Elec. Computing C-25, 1976, pp. 328-334.

12. W. W. Bledsoe. Some Results with Using Counterexamples to Shorten Proofs. University of Texas, Mathematics Department Memo ATP 51, July 1979.