Programming Languages
J. J. Horning Editor

An Algorithm for Reasoning About Equality

Robert E. Shostak
Stanford Research Institute

A simple technique for reasoning about equalities that is fast and complete for ground formulas with function symbols and equality is presented. A proof of correctness is given as well.

Key Words and Phrases: theorem proving, deduction, program verification, equality
CR Categories: 3.64, 3.66, 5.21

1. Introduction

To be useful for program verification, a deductive system must be able to reason proficiently about equality. Important as its semantics are, equality is often handled in an ad hoc and incomplete way—most commonly with a rewrite rule that substitutes equals for equals with some heuristic guidance. This article presents a simple algorithm for reasoning about equality that is fast, complete (for ground formulas with function symbols and equality), and useful in a variety of theorem-proving situations. A proof of the theorem on which the algorithm is based is given as well.

2. An Example

Let us first consider an example formula and how one could go about proving it. The formula given below is of the kind one encounters in verifying programs involving array indexing:

$$(I = J \wedge K = L \wedge A[I] = B[K] \wedge J = A[J] \wedge M = B[L]) \supset A[M] = B[K]).$$

Here, A and B are function symbols (corresponding to arrays) while I, J, K, L, and M are universally quantified variables (correponding to program variables).

One might approach such a formula by working backwards from the conclusion, substituting equals for equals until the left-hand side is transformed into the right-hand side. With a little patience, the following proof is obtained:

$A[M]$
$= A[B[L]]$ (using $M = B[L]$)
$= A[B[K]]$ (using $K = L$)
$= A[A[I]]$ (using $A[I] = B[K]$)
$= A[A[J]]$ (using $I = J$)
$= A[J]$ (using $J = A[J]$)
$= A[I]$ (using $I = J$ again)
$= B[K]$ (using $A[I] = B[K]$ again).

Of course, one could just as easily work from $B[K]$ rather than from $A[M]$, or work from both simultaneously; the links needed in the chain are the same in either case.

While this "backward substitution" method and other methods that transform formulas through a sequence of substitutions are logically sound, they are not particularly well suited to machine deduction simply because there is no easy way of knowing what substitution is the "right" one to make at each step. Indeed, a program working on the formula given above could grind on forever (for example, by repeated application of the substitution $J \rightarrow A[J]$), generating terms of ever-increasing depth of nesting.

Intuitively, it would not seem necessary to generate terms beyond a certain depth. It is easy, however, to construct examples showing that the critical depth (the smallest depth necessary to consider) cannot be calculated solely as a function of the depths of the terms appearing in the original formula; in particular, the critical depth is not simply the maximum of these depths. The reader can easily convince himself, for example, that no backward-substitution proof can be carried out for the formula given above without generating a term of at least depth 3. (The maximum depth of terms occurring in the formula is only 2.) Even if one could conveniently calculate the critical depth, one would still, in general, generate many more terms than are necessary.

Fortunately, this difficulty with substitution-transformation methods is not inherent in the problem. The next section presents a more efficient method that considers only the terms appearing in the original formula.

3. The Procedure

The method given here may be described formally as a decision procedure for the subclass of predicate calculus with function symbols and equality whose formulas have only universal quantifiers in prenex form. While the decidability of this subclass is well known, the clas-

This work was supported in part by AFOSR Contract F44620-73-C-0068.

Author's address: Stanford Research Institute, Menlo Park, CA 94025.

sical decision procedure for it [1] produces a combinational explosion that makes that method computationally infeasible for nontrivial problems.

Since the decision procedure presented here operates on the negation of the formula to be proved, it can also be viewed as a refutation procedure for ground formulas with function symbols and equality. The universally quantified variables become Skolem constants in the Skolemization of the negation.

The procedure is carried out as follows. The matrix of the formula F to be decided is first negated and placed in disjunctive normal form (d.n.f.). Next, all atomic formulas other than equalities are replaced by equalities as follows. For each n-ary predicate symbol P_i^n occurring in the formula, a new n-ary function symbol f_i^n is introduced. Each atomic formula $P_i^n(t_1, t_2, \ldots t_n)$ occurring in the formula is then replaced by the equality $f_i^n(t_1, t_2 \ldots, t_n) = c$, where c is a constant. The modified d.n.f. is clearly intersatisfiable with the original one, and is satisfiable if and only if one of its disjuncts is satisfiable. Each disjunct, moreover, consists of a conjunction of equalities and negations of equalities. The problem is thus reduced to testing the satisfiability of each such conjunction.

For example, suppose the formula to be proved is:

$$[(P_z \vee x = z) \wedge P_x] \supset [P_{g(y)} \vee z \neq g(y)].$$

Putting the negation into disjunctive normal form, we have:

$$(P_z \wedge P_x \wedge \neg P_{g(y)} \wedge z = g(y)) \vee (x = z \wedge P_x \wedge \neg P_{g(y)} \wedge z = g(y)).$$

Introducing the new function symbol f to replace P, we obtain:

$$(f(z) = c \wedge f(x) = c \wedge f(g(y)) \neq c \wedge z = g(y)) \vee (x = z \wedge f(x) = c \wedge f(g(y)) \neq c \wedge z = g(y)).$$

It remains to show how to test each conjunction for satisfiability. Let S be the set of equalities and negations of equalities occurring in the conjunction to be tested. Let T be the set of terms and subterms of terms occurring in S, and define the binary relation $\doteq$ as the smallest relation over $T \times T$ (where $u_1, u_2 \ldots u_n, t_1, t_2, v_1, v_2 \ldots v_n$ denote terms and f denotes a function symbol) that:

(1) Contains all pairs $\langle t_1, t_2 \rangle$ for which '$t_1 = t_2$' $\in S$
(2) Is reflexive, symmetric, and transitive.
(3) Contains the pair $\langle f(u_1, u_2 \ldots u_r), f(v_1, v_2 \ldots v_r) \rangle$ whenever it contains the pairs $\langle u_i, v_i \rangle$, $1 \leq i \leq r$, and $f(u_1, u_2 \ldots u_r), f(v_1, v_2 \ldots v_r)$ are both in T.

The test for satisfiability of S depends on the following theorem (to be proven later):

THEOREM. S is unsatisfiable $\Leftrightarrow$ there exist terms $t_1, t_2 \in T$ such that

'$t_1 \neq t_2$' $\in S$ and $t_1 \doteq t_2$.

The theorem tells us that to determine the satisfiability of S it suffices to consider the negated equalities of S one at a time. If one is found (say $t_1 \neq t_2$) for which $t_1 \doteq t_2$, S is unsatisfiable; otherwise S is satisfiable. Note that the definition of $\doteq$ involves only terms in T.

In order to use the theorem, it is necessary to be able to calculate whether a given pair of terms is in the relation $\doteq$. This can be done in a straightforward way by building the relation from the definition. Condition (1) is used as a basis, and (2) and (3) are repeatedly applied until no new terms are generated. Since $\doteq$ is an equivalence relation, one can conveniently represent it during the construction as a collection of sets of elements of T, each set containing elements known to be in the relation with the other elements of that set.

As an illustration, consider the set $S = \{I = J, K = L, A[I] = B[K], J = A[J], M = B[L], A[M] \neq B[K]\}$ that arises from the example given earlier. The corresponding set T is $\{I, J, K, L, A[I], B[K], A[J], M, B[L], A[M]\}$. The relation $\doteq$ is constructed from its definition as follows.

From the basis (1), one obtains:
$\{\{I, J\}, \{K, L\}, \{A[I], B[K]\}, \{J, A[J]\}, \{M, B(L)\}\}$
Using (2):
$\{\{I, J, A[J]\}\{K, L\}\{A[I], B[K]\}, \{M, B[L]\}\}$
Using (3):
$\{\{I, J, A[J]\}\{K, L\}\{A[I], B[K]\}, \{M, B[L]\}, \{A[I], A[J]\}, \{B[K], B[L]\}\}$
Using (2):
$\{\{I, J, A[J], A[I], B[K], B[L], M\}, \{K, L\}\}$
Using (3):
$\{\{I, J, A[J], A[I], B[K], B[L], M\}, \{K, L,\}, \{A[M], A[I]\}, \{A[M], A[J]\}\}$
Using (2):
$\{\{I, J, A[J], A[I], B[K], B[L], A[M]\}, \{K, L\}\}$

Since (3) yields no new pairs, the construction is complete. Since $A[M] \doteq B[K]$, S must be unsatisfiable.

The rules for building up $\doteq$ can be implemented efficiently. The authors of [3] have recently coded a very fast implementation that represents terms as graphs and uses the set-union algorithm [5] in the closure step. In [3] it is also shown that their implementation requires only order n^2 deterministic time and linear space, where n is the length of the input S.

It should be noted that while the satisfiability of each set S can thus be determined quite quickly, the procedure as a whole (and the expansion into disjunctive normal form in particular) is of exponential time complexity. This is not surprising, of course, since the decision problem for the class is NP-complete.

The author has coded the procedure in Interlisp for the DEC-10 using a matrix representation of $\doteq$. The program has been tested on a few dozen examples of the kind that arise in program verification applications. It was found that most examples four or five lines long could be handled in just a few seconds. The example presented at the beginning of the paper required less than a second.

4. Proof of the Theorem

The main import of the theorem on which the algorithm is based is that it suffices to "consider" only the terms occurring in formula to be decided. The proof of the theorem is largely concerned with extending the model provided by the relation $\doteq$ from the finite set T to the entire Herbrand Universe. We now restate the theorem and give its proof.

THEOREM *S is satisfiable* $\Leftrightarrow \exists\, t_1, t_2 \in T$ *such that* $t_1 \doteq t_2$ *and* '$t_1 \neq t_2$' $\in S$.

PROOF $\Rightarrow$

Suppose S is satisfiable, $t_1, t_2 \in T$ and $t_1 \doteq t_2$. Let M be a model for S. Because M satisfies the reflexivity, symmetry, transitivity and substitutivity axioms of equality, $t_1 \doteq t_2$ implies that t_1 and t_2 must have the same values in M. Hence, '$t_1 \neq t_2$' cannot be a member of S.

$\Leftarrow$

Suppose there are no terms t_1, t_2 in T such that $t_1 \doteq t_2$ and '$t_1 \neq t_2$' $\in S$. We will show that S is satisfiable by constructing a model M for S. The model must assign a value $v_M(t)$ to each term t in the Herbrand Universe of S in such a way that:

(1) '$t_1 = t_2$' $\in S$ implies $v_M(t_1) = v_M(t_2)$

(2) '$t_1 \neq t_2$' $\in S$ implies $v_M(t_1) \neq v_M(t_2)$

(3) $v_M(x_i) = v_M(y_i)$, $1 \leq i \leq r$, implies $v_M(f(x_1, \ldots x_r)) = v_M(f(y_1, \ldots, y_r))$ (where f ranges over all function symbols and x_i, y_i over all terms)

The first two conditions require that M satisfies each atomic formula of S. The third condition requires M to satisfy the substitutivity axiom of equality.

Before defining v_M we first construct the *term universe* $T_\infty = \cup_{i=0}^{\infty} T_i$ of S inductively as follows:

$T_0 = T$

$T_{i+1} = \{f(t_1, \ldots, t_r) | t_i \in T_i\} \cup T_i$

(where f ranges over all function symbols occurring in S.) Note that the term universe T_∞ is identical as a set to the Herbrand Universe, but is constructed differently.

Next, pick a representative term from each of the equivalence classes induced by $\doteq$ on T, and define the function $a : T \to T$ that assigns to each term in T the representative of its class.

The inductive construction of model M follows:

I. If $t \in T_0$, let $v_M(t) = a(t)$

II. If $t \in T_{j+1} - T_j$, $j \geq 0$, and $t = f(t_1, t_2, \ldots t_r)$, then let

$$v_M(t) = \begin{cases} v_M(f(x_1, \ldots, x_r)) \text{ if } \exists\, f(x_1, \ldots, x_r) \in T_j \\ \qquad \text{and } v_{M(x_i)} = v_M(t_i),\ 1 \leq i < r \\ f(v_M(t_1), \ldots, v_M(t_r)) \quad \text{otherwise.} \end{cases}$$

Note that M is a Herbrand model, i.e., it always assigns values from the Herbrand Universe. The notation "$f(v_M(t_1), \ldots, v_M(t_r))$" is intended to represent the function symbol denoted by f followed by the terms obtained by evaluating $v_M(t_i)$ for each i.

Note also that v_M would not seem to be uniquely defined, owing to the existential choice implicit in the definition. In a moment, however, it will be clear that only one choice is possible.

Now, we need to show that M satisfies (1), (2), and (3) above. (1) and (2) hold since '$t_1 = t_2$' $\in S \Rightarrow t_1 \doteq t_2 \Rightarrow a(t_1) = a(t_2) \Rightarrow v_M(t_1) = v_M(t_2)$ and '$t_1 \neq t_2$' $\in S \Rightarrow t_1 \not\doteq t_2 \Rightarrow a(t_1) \neq a(t_2) \Rightarrow v_M(t_1) \neq v_M(t_2)$.

It remains to show that (3) holds, i.e., that $v_M(x_i) = v_M(y_i)$, $1 \leq i \leq r$, implies that $v_M(f(x_1, \ldots, x_r)) = v_M(f(y_1, \ldots y_r))$. This is proved by induction on the maximum m of the term universe heights of $f(x_1, \ldots, x_r)$, $f(y_1, \ldots, y_r)$:

BASIS: $m = 0$

Then $x_i, y_i, f(x_1, \ldots, x_r), f(y_1, \ldots, y_r)$ are all in T, and so

$$v_M(x_i) = v_M(y_i) \Rightarrow a(x_i) = a(y_i) \Rightarrow x_i \doteq y_i$$
$$\Rightarrow f(x_1, \ldots, x_r) \doteq f(y_1, \ldots, y_r) \Rightarrow a(f(x_1, \ldots, x_r)) = a(f(y_1, \ldots, y_r))$$
$$\Rightarrow v_M(f(x_1, \ldots, x_r)) = v_M(f(y_1, \ldots, y_r))$$

as required.

INDUCTION STEP: $m > 0$.

First consider the case in which the height of $f(x_1, \ldots, x_r)$ is strictly greater than that of $f(y_1, \ldots, y_r)$. In this case, $v_M(f(x_1, \ldots, x_r)) = v_M(f(z_1, \ldots, z_r))$, where $f(z_1, \ldots, z_r)$ is of lesser height than $f(x_1, \ldots, x_r)$, and $v_M(x_i) = v_M(z_i)$. (Note that $f(z_1, \ldots, z_r)$ is possibly the same as $f(y_1, \ldots, y_r)$.) Now since $v_M(y_i) = v_M(x_i) = v_M(z_i)$, we have by induction hypothesis that

$$v_M(f(x_1, \ldots, x_r)) = v_M(f(z_1, \ldots, z_r)) = v_M(f(y_1, \ldots, y_r))$$

as required.

In the remaining case, $f(x_1, \ldots, x_r)$ and $f(y_1, \ldots, y_r)$ are of the same height. Now if there exists a term $f(z_1, \ldots, z_r)$ of lower height such that $v_M(z_i) = v_M(x_i)$, the argument above can be used. Otherwise, $v_M(f(x_1, \ldots, x_r)) = f(v_M(x_1), \ldots, v_M(x_r)) = f(v_M(y_1), \ldots, v_M(y_r)) = v_M(f(y_1, \ldots, y_r))$ as required. □

It might be noted that (3) implies the uniqueness of M as it has been defined above. Of course, the uniqueness was not essential to the proof.

Acknowledgment. The author is grateful to Drs. R. S. Boyer, J Strother Moore, E. Horowitz, W. Bledsoe and the reviewers for their observations and general helpfulness.

Received January 1977; revised June 1977

References

1. Ackermann, W. *Solvable Cases of the Decision Problem*, North-Holland Publishing Co., Amsterdam, 1954, pp. 102–103.
2. Mendelson, E. *Introduction to Mathematical Logic*, D. Van Nostrand Co., Inc., Princeton, New Jersey, 1964.
3. Oppen, D., and Nelson, G. Fast Decision Algorithms Based on Union and Find. Proceedings of 8th Symposium on Foundations of Computer Science, Princeton, N.J., Nov. 1977.
4. Robinson, G., and Wos, L. Paramodulation and Theorem-Proving in First-order Theories with Equality. *Machine Intelligence 4*, D. Michie and B. Meltzer, Eds., American Elsivier, N.Y., 1969 pp. 135–150.
5. Tarjan, R. Efficiency of a good but not linear set-union algorithm, *J. ACM*, *22*, 2 (April 1975), 215–225.