Function Inversion

Nachum Dershowitz[*]
University of Illinois, Urbana, IL 61801, USA
Tel-Aviv University, Ramat-Aviv, Israel
and
Subrata Mitra
III Phase, Bangalore, India

Abstract

An algorithm is given for inverting functions defined by left-linear ground convergent rewrite systems, with left sides restricted in depth and right sides not having defined symbols at the top.

1 Introduction

We are interested in the problem of solving equations of the form $t = N$, where t is an arbitrary term containing defined function symbols, constructors (that is, undefined function symbols), and variables, while N is a *ground constructor term*, by which we mean a term containing only constructors and constants. For example, given a definition of squaring, the square root of 9 is found by solving $x^2 = sssssssss0$ (using the unary successor constructor s to represent integers), and the solution is $x = sss0$. We describe a broad class of functions that can be inverted in this manner.

To solve such problems, we restrict ourselves here to equational theories that can be presented as (rather typical) functional programs in the form of ground convergent left-linear rewrite systems, that is, finite sets of equations that compute unique output values when applied (from left-to-right) to input expressions, and whose defining equations do not have multiple occurrences of variables on their left side. For non-left-linear systems, function inversion is known to be as hard as equation solving. Even for arbitrary linear systems, the problem is still unsolvable. By placing (not wholly unrealistic) syntactic restrictions on the sides of rules, we can, however, show termination of our inversion algorithm.

[*]Supported in part by the National Science Foundation under grant CCR-97-00070.

The following is a program for squaring meeting our criteria (with standard abbreviating conventions):

$$\begin{array}{rcl} 0+x & \rightarrow & x \\ sx+y & \rightarrow & s(x+y) \\ 0\times x & \rightarrow & 0 \\ x\times 0 & \rightarrow & 0 \\ sx\times sy & \rightarrow & s(y+x\times sy) \\ 0^2 & \rightarrow & 0 \\ (sx)^2 & \rightarrow & s(x^2+ss0\times x) \end{array}$$

Function inversion has widespread applications in symbolic computation, functional-logic programming languages, and automated deduction.

2 Nomenclature

A *rewrite system* is a finite set of ordered equations between terms, called *rewrite rules*. A rule $l \rightarrow r$, equating terms l and r, applies to a term t if t contains a subterm s such that $l\tau = s$ for some matching substitution τ. We write $t \rightarrow t'$ to indicate that t' is obtained from t by replacing the subterm s of t with $r\tau$. A term N is said to be in *normal form* if there is no term N' such that $N \rightarrow N'$.

A *constructor*, for our purposes, is a function symbol not heading any left side. A term is *ground* if it has no variables. The *depth* of a term t is the maximal number of nodes in a path in its tree representation. For example, the term $f(g(x), 0)$ is of depth 3.

A rewrite system R is *ground convergent* if every ground term has exactly one normal form and all computations terminate. A system is *left-linear* if no variable appears more than once on a left side. Most functional programs have these properties.

A *(sub)goal* of our procedure takes the form $s \rightarrow^? t$, where s and t share no variables, and has a *solution* σ, assigning terms to variables in s, if and only if $s\sigma \rightarrow \cdots \rightarrow t\tau$, for some substitution τ of terms for variables in t. There may of course be more than one solution to a goal. We need not compute all solutions: we can ignore less general solutions than ones we do compute (e.g. $x \mapsto sz$ subsumes $x \mapsto sss0$); we can ignore solutions that are not themselves normal forms, since they must be equal to normal-form solutions (e.g. $x \mapsto 0 \times z$ is covered by $x \mapsto 0$ for the equational theory defined by the squaring example). If a rewrite system R is ground convergent, then an equation $s = N$, for normal form N, has a solution σ in the equational theory of R, if and only if σ is a solution to the goal $s \rightarrow^? N$.

3 The Algorithm

We give an algorithm for function inversion for any theory presented by a left-linear ground convergent rewrite system R with the following additional conditions on its rules:

1. Each left side is of depth at most 3.
2. If a right side is a variable or constant, then the left side of that rule is of depth at most 2.
3. Whenever a right side is not a variable, it is headed by a constructor at its root.

The above definition of squaring obeys the restrictions, and therefore has a solvable inversion problem.

At each stage of the algorithm, we have a set of subgoals. We (don't care) non-deterministically choose one, $s \to^? t$, and consider the following cases.

1. If s and t are identical (in particular, if they are identical constants), the subgoal may be removed.
2. If t is a variable, just remove this subgoal.
3. Suppose s is a variable x:
   (a) If x is already bound to a term u, then bind it instead to the most general unifier of u and t.
   (b) If u and t are not unifiable, fail.
   (c) If x is unbound, add the binding $x \mapsto t$.
4. If neither s nor t is a variable, try *both* of the following:
   (a) Whenever $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$, for the same leading function symbol f, replace the goal with the multiset of goals, $s_1 \to^? t_1, \ldots, s_n \to^? t_n$.
   (b) Suppose $s = f(s_1, \ldots, s_n)$ and there is a rule $f(l_1, \ldots, l_n) \to r$ in R (with all its variables renamed apart from those in the goal), then do one of the following:
      i. If r and t are the same constant, then replace the goal with the subgoals $s_1 \to^? l_1, \cdots, s_n \to^? l_n$.
      ii. If r is a constant, but $t \neq r$, fail.
      iii. If r is a variable x, then replace the goal with the subgoals $s_1 \to^? l_1\sigma, \cdots, s_n \to^? l_n\sigma$, where σ is $x \mapsto t$.
      iv. If r is headed by a constructor that is not at the head of t, then fail.
      v. If none of the above hold, then by assumption 2, $t = c(t_1, \ldots, t_m)$ and $r = c(r_1, \ldots, r_m)$, for some constructor c. First solve the goal $r \to^? t$, by recursively solving the subgoals $r_1 \to^? t_1, \ldots, r_m \to^? t_m$ in succession. For each solution σ of all m subgoals, solve $s_1 \to^? l_1\sigma, \cdots, s_n \to^? l_n\sigma$.

4 Its Correctness

To prove the correctness of the algorithm, we need to show the following:

Soundness Given a goal $s \rightarrow^? t$, if the algorithm produces a solution σ, then $s\sigma \rightarrow \cdots \rightarrow t$.

Completeness Given a goal $s \rightarrow^? t$, if $s\sigma \rightarrow \cdots \rightarrow t$ for some substitution σ, then the algorithm will produce σ.

Termination All paths in the algorithm terminate (either with success or with failure).

To those ends we need the following invariants:

I Variables that appear on the right side of a goal appear only once among all the subgoals. Such variables do not contribute to any solution.

II The size of u in each of the substitutions $x \mapsto u$ of a solution to a goal $s \rightarrow^? t$ is bounded by the size of t.

To see that invariant I holds, just note that right sides are composed of pieces of right sides (which were initially ground), or from subterms of left sides, which are linear.

For invariant II we show by computational induction that the right sides of subgoals are bounded in depth by that of the initial goal: By assumption 2, the l_i in steps 4(b)i and 4(b)iii are of depth 1, so even after substituting t in 4(b)iii, the right side is bounded. Step 4a and the first set of subgoals in step 4(b)v have smaller right sides. We will see that the latter recursive calls terminate. By assumption 1, the l_i in step 4(b)v are at most of depth 2, while t is at least that deep. If σ is bounded by the depth of t_i, then $l_i\sigma$ cannot be deeper than t, since l_i can contribute at most 1 to the depth.

Since the most general unifier of two terms with disjoint variables is bounded in depth by the maximum of their depths, all substitutions created in step 3a or 3c will satisfy the invariant.

Indeed, it is because most general solutions are bounded in size that the inversion problem is decidable for convergent systems.

The proof of termination uses the multiset extension of the lexicographic measure $\langle depth(t), depth(s)\rangle$ of a subgoal $s \rightarrow^? t$. The subgoals in step 4a are smaller, since their right sides are each smaller than t. For step 4(b)i, assumption 2 guarantees that each l_i is of depth 1, so the subgoals are smaller. For step 4(b)iii, assumption 2 guarantees that each l_i is of depth 1, so after the substitution $l_i\sigma$ is either of depth 1 or it is t. In the latter case, the subgoal is smaller by virtue of its smaller left side. For step 4(b)v, we begin with goals with smaller right sides, then continue with goals that have $l_i\sigma$, which are no deeper than t, on the right, and smaller s_i on the left.

Soundness is straightforward, except for step 2, the soundness of which follows from invariant I, since an instance of a solution is also a solution.

For the completeness argument, under the assumption of convergence, we need only consider innermost computations, in which rules are only applied to terms whose subterms are in normal form. Consider any normalizing derivation $s\sigma \to \cdots \to t$. If it has zero steps, then one of steps 1– 3 will find the solution. If it has no steps at a topmost symbol, then step 4a applies. Otherwise, consider the first step at the top, and one of the cases of step 4b will work, by assumption 3.

If a variable x has a normal form looking like u, then if it also has normal form t, the two must unify, as in step 3a.

5 Related Work

"Semantic matching" is the process of enumerating a basis set of substitutions σ (of terms for variables in s) that make t equal to a variable-free (but not necessarily constructor) term in a specified theory, even when the function symbols in t are not fully defined. Unlike our previous work on semantic matching [Dershowitz *et al.*, 1992], here we use a purely syntactic property of depth. Those results were extended in [Aguzzi and Modigliani, 1994] to incorporate positional information.

In [Christian, 1992] it is shown that narrowing (hence, inversion) terminates for terminating systems with all left-side variables at depth 2. Perhaps the algorithm given here can be extended in that direction.

Acknowledgement

We thank Claude Kirchner for his indispensable interest and gracious hospitality.

References

[Aguzzi and Modigliani, 1994] G. Aguzzi and U. Modigliani. A Criterion to Decide the Semantic Matching Problem. In *Proceedings of the International Conference on Logic and Algebra (in memory of Prof. Magari)*, Italy, 1995.

[Christian, 1992] J. Christian. Some Termination Criteria for Narrowing and E-Narrowing. In *Proceedings of the Eleventh International Conference on Automated Deduction*, Saratoga Springs, New York, June 1992. Volume 607, pages 582–588, of *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

[Dershowitz *et al.*, 1992] N. Dershowitz, S. Mitra, and G. Sivakumar. Decidable matching for convergent systems. In *Proceedings of* *the Eleventh Conference on Automated Deduction*, pages 589–602, Saratoga Springs, NY, June 1992. Vol. 607 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin. (On-line at `sal.cs.uiuc.edu/~nachum/papers/match-new.ps.gz`.)