Halting and Equivalence of Schemes over Recursive Theories

Dexter Kozen
Department of Computer Science
Cornell University
Ithaca, New York 14853-7501, USA
kozen@cs.cornell.edu

October 28, 2002

Abstract

Let Σ be a fixed first-order signature. In this note we consider the following decision problems.

(i) Given a recursive ground theory T over Σ, a program scheme p over Σ, and input values specified by ground terms $t_1, \ldots, t_n$, does p halt on input $t_1, \ldots, t_n$ in all models of T?

(ii) Given a recursive ground theory T over Σ and two program schemes p and q over Σ, are p and q equivalent in all models of T?

When T is empty, these two problems are the classical halting and equivalence problems for program schemes, respectively. We show that problem (i) is r.e.-complete and problem (ii) is Π_2^0-complete. Both these problems remain hard for their respective complexity classes even if T is empty and Σ is restricted to contain only a single constant, a single unary function symbol, and a single monadic predicate. It follows from (ii) that there can exist no relatively complete deductive system for scheme equivalence.

Let Σ be a fixed first-order signature. A *ground formula* over Σ is a Boolean combination of atomic formulas $P(t_1, \ldots, t_n)$ of Σ, where the t_i are ground terms (no occurrences of variables). A *ground theory* over Σ is a consistent set of ground formulas closed under entailment. A set E of ground formulas is a *complete extension* of a ground theory T if E contains T and each ground formula or its negation appears in E.

Theorem 1 *The following problem is r.e.-complete: Given a recursive ground theory T over Σ, a program scheme p over Σ, and input values specified by ground terms $\overline{t} = t_1, \ldots, t_n$, does p halt on input $\overline{t}$ in all models of T? The problem remains r.e.-hard even if $T = \varnothing$ and Σ is restricted to contain only a single constant, a single unary function symbol, and a single monadic predicate.*

Theorem 2 *The following problem is Π^0_2-complete: Given a recursive ground theory T over Σ and two schemes* p *and* q *over Σ, are* p *and* q *equivalent in all models of T? The problem remains Π^0_2-hard even if $T = \varnothing$ and Σ is restricted to contain only a single constant, a single unary function symbol, and a single monadic predicate.*

When $T = \varnothing$, these are the classical halting and equivalence problems for program schemes. Note that for the upper bounds, the recursive theory T is part of the input. Classical lower bound proofs (see [5]) establish the r.e. hardness of the two problems for the case $T = \varnothing$.

Proof of Theorem 1. Let T be a recursive ground theory. It suffices to restrict our attention to Herbrand models of T. These models are in one-to-one correspondence with the complete extensions of T.

First we show that the problem is r.e. Given p and $\bar{t}$, we simulate the computation of p on input $\bar{t}$ on all Herbrand models of T simultaneously, using the decidability of T to resolve tests. Each branch of the simulation maintains a finite set E of ground atomic formulas consistent with T, initially empty. Whenever a test $P(s_1, \ldots, s_k)$ is encountered, we consult T and E to determine which branch to take. If the truth value of $P(s_1, \ldots, s_k)$ is determined by T and E, that is, if $T \vDash E \rightarrow P(u_1, \ldots, u_k)$ or $T \vDash E \rightarrow \neg P(u_1, \ldots, u_k)$, where the ground term u_i is the current value of s_i, $1 \leq i \leq k$, then we just take the appropriate branch. Otherwise, if both $P(u_1, \ldots, u_k)$ and $\neg P(u_1, \ldots, u_k)$ are consistent with $T \cup E$, then the simulation branches, extending E with $P(u_1, \ldots, u_k)$ on one branch and $\neg P(u_1, \ldots, u_k)$ on the other. In each simulation step, all current branches are simulated for one step in a round-robin fashion. We thus simulate the computation of p on all possible complete extensions of T simultaneously. If p halts on all such extensions, then by König's Lemma there is a uniform bound on the halting time of all branches of the computation. The simulation halts successfully when that bound is discovered.

We now show that the problem is r.e.-hard in the restricted case $\Sigma = \{a, f, P\}$, where a is a constant, f is a unary function symbol, and P is a unary relation symbol. We will encode the halting problem for deterministic Turing machines. Given a deterministic Turing machine M and a string x over M's input alphabet, we will construct a scheme p with no input or output and a finite atomic theory T such that p halts on all complete extensions of T iff M halts on input x. The encoding technique used here is fairly standard, but we include the argument for completeness and because we need the resulting scheme p in a certain special form for the proof of Theorem 2.

The Herbrand domain over a and f is isomorphic to the natural numbers with 0 and successor. An Herbrand model H over this domain is represented by an infinite binary string whose n^{th} digit is 1 iff $P(f^n(a))$ in H. The correspondence is one-to-one. We will use these strings to encode computation histories of M.

Each string x over M's input alphabet determines a unique finite or infinite computation history $\#\alpha^x_0\#\alpha^x_1\#\alpha^x_2\#\cdots$, where α^x_i is a string over a finite alphabet Δ encoding the instantaneous configuration of M on input x at time i (tape contents, head position, current state). The configurations α^x_i are separated by a symbol $\# \notin \Delta$. The computation history in turn can be encoded in binary. Finally, an infinite binary string can be encoded by the truth values of $P(f^n(a))$ for successive n.

The ground theory T describes the starting configuration $\#\alpha_0^x\#$ of M on input x. Thus T consists of finitely many ground atomic formulas. Any complete extension of T describes either the unique valid computation history of M on input x or a garbage string. The scheme p can read the n^{th} bit of this string in the corresponding Herbrand model by testing the value of $P(f^n(a))$. It starts by scanning the initial part of the string to check that it is of the form $\#\alpha_0^y\#$ for some y. (This step is not strictly necessary for this proof, since we are restricting our attention to models of T, in which this step will always succeed; but it will be useful later in the proof of Theorem 2.) Next, p scans the string from left to right to determine whether each successive α_{i+1}^x follows from α_i^x in one step according to the transition rules of M. It does this by comparing corresponding bits in α_i^x and α_{i+1}^x using two variables to simulate pointers into the string. If the current value of variable x is $f^n(a)$, then testing $P(x)$ reads the n^{th} bit of the string. The pointer is advanced by the assignment $x := f(x)$.

If p discovers an error, so that the string does not represent a computation history of M on some input, it halts immediately. It also halts if it ever encounters a halting state of M anywhere in the string. Thus the only complete extension of T that would cause p not to halt is the one describing the valid computation history of M on x in the case that M does not halt on x. Thus p halts on all complete extensions of T iff M halts on x.

We can further restrict to $T = \varnothing$ by observing that the T in this construction is finite, so it can be hard-wired into the scheme p itself. Thus the initial format check that p performs can be modified to check whether T holds and halt immediately if not. However, for purposes of the proof of Theorem 2 below, it will be important that p not depend on the input x but only on the machine M. $\square$

Proof of Theorem 2. Two schemes are equivalent over all models of T iff they are equivalent over all Herbrand models of T. As above, each Herbrand model of T is uniquely represented by a complete extension of T.

First we show that equivalence of schemes over models of T is Π_2^0. Equivalently, inequivalence of schemes over models of T is Σ_2^0. It suffices to show that inequivalence of schemes over models of T can be determined by an IND program over $\mathbb{N}$ with an $\exists\forall$ alternation structure [4].

The two schemes p and q are not equivalent over models of T iff there exists an Herbrand model H of T and input values $\overline{t} = t_1, \ldots, t_n$ such that when interpreted over H, either

(i) both p and q halt on input $\overline{t}$ and produce different output values;

(ii) p halts on $\overline{t}$ and q does not; or

(iii) q halts on $\overline{t}$ and p does not.

We start by selecting existentially the input $\overline{t}$ and the alternative (i), (ii) or (iii) to check.

If alternative (i) was selected, we simulate p and q on input $\overline{t}$, maintaining a finite set E of ground atomic formulas and using T and E as in the proof of Theorem 1 to resolve tests. Whenever a test is encountered that is not determined by T and E, we guess the truth value and extend E accordingly. Thus we are nondeterministically guessing the model H as we go along. This is done by existential branching in the IND program. We continue the simulation until both p and q halt, then compare output values, accepting if they differ.

If alternative (ii) was selected, we simulate p on $\bar{t}$ until it halts, maintaining the guessed truth values of undetermined tests in the set E as above. When p has halted, we have a consistent extension $T \cup E$ of T, where E consists of the finitely many tests that were guessed during the computation of p. So far we have only used existential branching. We must now verify that there exists a complete extension of $T \cup E$ in which q does not halt on input $\bar{t}$. By Theorem 1, this problem is Π_1^0-complete, so we can solve it with a purely universally-branching IND computation.

The argument for alternative (iii) is symmetric.

For the lower bound, we reduce the totality problem for Turing machines, a well-known Π_2^0-complete problem, to the equivalence problem. The totality problem is to determine whether a given Turing machine M halts on all inputs. As above, it will suffice to consider $T = \varnothing$ and $\Sigma = \{a, f, P\}$.

Given a deterministic Turing machine M, we construct two schemes p and q with no input or output that are equivalent iff M halts on all inputs. The scheme p is the one constructed in the proof of Theorem 1. As in that proof, each input string x over M's input alphabet determines a unique computation history, and the scheme p checks that the Herbrand model in which it is running encodes a valid computation history of M on some input.

Now unlike the proof of Theorem 1, there is an extra source of non-halting. Recall that there is an initial format check in which p checks that the string has a prefix of the form $\#\alpha_0^x\#$ for some x. If there is no second occurrence of $\#$ in the string, then p will loop infinitely looking for it. If it does detect a second occurrence of $\#$, then as before, the only source of non-halting is if M does not halt on x. We therefore build q to simply check for a prefix of the form $\#\alpha_0^x\#$ exactly as p does and halt immediately when it encounters the second occurrence of $\#$. Thus p does not halt in the Herbrand model H iff the string represented by H either

(i) does not have a prefix of the form $\#\alpha_0^x\#$, or

(ii) does have a prefix of the form $\#\alpha_0^x\#$ and represents a non-halting computation history of M on x;

and q does not halt in H in case (i) only. Therefore p and q are equivalent iff M halts on all inputs. $\square$

In [1], axioms were proposed for reasoning equationally about input/output relations of first-order program schemes over Σ. These axioms have been shown to be adequate for some fairly intricate equivalence arguments arising in program optimization [1, 2]. However, unlike the propositional case, it follows from Theorem 2 that there can exist no finite relatively complete axiomatization for first-order scheme equivalence. If such an axiomatization did exist, then the scheme equivalence problem over a given first-order theory T would be r.e. in T. But it is decidable whether a given first-order sentence φ is a consequence of a given finite set T of ground formulas over the signature $\Sigma = \{a, f, P\}$, since $T \vDash \varphi$ iff $\forall P\ (T \rightarrow \varphi)$ is a valid sentence of the monadic second-order theory of successor, a well-known decidable theory [3]. By Theorem 2, the scheme equivalence problem over such theories T is still Π^0_2-hard, therefore not r.e. in the decidable first-order theory generated by T.

Acknowledgements

This work was supported in part by NSF grant CCR-0105586 and by ONR Grant N00014-01-1-0968. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the US Government.

References

[1] Allegra Angus and Dexter Kozen. Kleene algebra with tests and program schematology. Technical Report 2001-1844, Computer Science Department, Cornell University, July 2001.

[2] Adam Barth and Dexter Kozen. Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Technical Report 2002-1865, Computer Science Department, Cornell University, June 2002.

[3] J. R. Büchi. On a decision method in restricted second order arithmetics. In *Proc. Int. Congr. Logic, Methodology and Philosophy of Science*, pages 1–12. Stanford University Press, 1962.

[4] David Harel and Dexter Kozen. A programming language for the inductive sets, and applications. *Information and Control*, 63(1–2):118–139, 1984.

[5] Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill, 1974.