Halting and Equivalence of Program Schemes in Models of Arbitrary Theories

Dexter Kozen

Cornell University, Ithaca, New York 14853-7501, USA
`kozen@cs.cornell.edu`
`http://www.cs.cornell.edu/~kozen`

In Honor of Yuri Gurevich
on the Occasion of his Seventieth Birthday

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

(i) Given a first-order theory or ground theory T over Σ of Turing degree α, 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 first-order theory or ground theory T over Σ of Turing degree α 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 Σ_1^α-complete and problem (ii) is Π_2^α-complete. Both problems remain hard for their respective complexity classes even if 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 over models of theories of any Turing degree.

Keywords: dynamic model theory, program scheme, scheme equivalence. Mathematics Subject Classification Codes: 03B70, 3D10, 03D15, 03D28, 03D75, 68Q17.

Let Σ be an arbitrary but 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 literal* is a ground atomic formula $P(t_1, \ldots, t_n)$ or its negation. A *ground theory* over Σ is a consistent set of ground formulas closed under entailment. A (*first-order*) *theory* over Σ is a consistent set of first-order formulas closed under entailment. A ground theory E is a *complete ground extension* of T if E contains T and every ground formula or its negation appears in E. A first-order theory E is a *complete extension* of T if E contains T and every first-order sentence or its negation appears in E.

Theorem 1. *Let α be an arbitrary Turing degree. The following problems are Σ_1^α-complete and Π_2^α-complete, respectively:*

(i) Given a ground theory T over Σ of Turing degree α, a program scheme p over Σ, and input values specified by ground terms $\bar{t} = t_1, \ldots, t_n$, does p halt on input $\bar{t}$ in all models of T?

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

Both problems remain hard for their respective complexity classes even if Σ is restricted to contain only a single constant, a single unary function symbol, and a single monadic predicate. Both problems remain complete for their respective complexity classes if "ground theory" in the statement of the problem is replaced by "first-order theory".

Note that for both problems, the theory T is part of the input in the form of an oracle.

If T is the empty ground theory, these are the classical halting and equivalence problems for program schemes, respectively. Classical lower bound proofs (see [7]) establish the r.e. hardness of the two problems for this case. The Π_2^0-hardness of (ii) for this case can also be shown to follow without much difficulty from a result of [4].

Proof. Let T be a first-order theory or ground theory of Turing degree α. For the upper bounds, we do not actually need T to be closed under entailment, but only that ground entailment relative to T is decidable; that is, if ϕ is a ground formula, then $T \vDash \phi$ is decidable with an oracle for T.

We first consider problem (i). For the upper bound, we wish to show that the problem is in Σ_1^α. Given p and $\bar{t}$, we simulate the computation of p on input $\bar{t}$ on all complete extensions of T simultaneously, using the oracle for T to resolve tests. Each branch of the simulation maintains a finite set E of ground literals consistent with T, initially empty. Whenever a test $P(s_1, \ldots, s_k)$ is encountered, we substitute the current values of the program variables, which are ground terms, for the variables to obtain a ground atomic formula $P(u_1, \ldots, u_k)$, then consult T and E to determine which branch to take. If the truth value of the test $P(s_1, \ldots, s_k)$ is determined by T and E, that is, if $T \vDash E \to P(u_1, \ldots, u_k)$ or $T \vDash E \to \neg P(u_1, \ldots, u_k)$, 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 problem (i) is Σ_1^α-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. For now we will assume that we have two unary relation symbols Q, R; we can later encode these in a single P by taking $P(f^{2n}(a)) = Q(f^n(a))$ and $P(f^{2n+1}(a)) = R(f^n(a))$.

Let $A \subseteq \mathbb{N}$ be any set. We will show how to encode the halting problem for deterministic oracle Turing machines M with oracle A. This problem is Σ_1^A-complete. Given an input x over M's input alphabet, we will construct a ground theory T_0 Turing-equivalent to A and a scheme p with no input or output such that p halts on all models of T_0 iff M halts on input x. Later, we will extend T_0 to a first-order theory T of the same complexity. 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 (ii).

Consider the Herbrand domain consisting of all terms over a and f. This domain is isomorphic to the natural numbers with 0 and successor under the correspondence $n \mapsto f^n(a)$. An Herbrand model H over this domain is represented by a pair of semi-infinite binary strings representing the truth values of $Q(f^n(a))$ and $R(f^n(a))$ for $n \geq 0$. The correspondence is one-to-one. We will use the string corresponding to Q to encode a computation history of M and the string corresponding to R to encode the oracle A.

Each string x over M's input alphabet determines a unique finite or infinite computation history $\#\alpha_0^x\#\alpha_1^x\#\alpha_2^x\#\cdots$, where α_i^x is a string over a finite alphabet Δ encoding the instantaneous configuration of M on input x at time i consisting of the tape contents, head position, and current state. We also assume that configurations encode all oracle queries and the answers returned by the oracle A (we will be more explicit about the precise format of the encoding below). The configurations α_i^x are separated by a symbol $\# \notin \Delta$. The computation history in turn can be encoded in binary, and this infinite binary string can be encoded by the truth values of $Q(f^n(a))$, $n \geq 0$.

The ground theory T_0 describes the oracle A using R and the starting configuration $\#\alpha_0^x\#$ of M on input x using Q. The description of the starting configuration consists of a finite set S_x of ground literals of the form $Q(f^n(a))$ or $\neg Q(f^n(a))$. The oracle is described by the set

$$\{R(f^n(a)) \mid n \in A\} \cup \{\neg R(f^n(a)) \mid n \notin A\}. \tag{1}$$

In any complete extension of T_0, the infinite string corresponding to Q 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 $Q(f^n(a))$. 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_0, in which this step will always succeed; but it will be needed later in the proof of (ii).) 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 the variable

x is $f^n(a)$, then testing $Q(x)$ reads the n-th bit of the string. The pointer is advanced by the assignment $x := f(x)$.

The scheme p must also verify that oracle responses are correct. Without loss of generality, we can assume that M uses the following mechanism to query the oracle. We assume that M has an integer counter initially set to 0. In each step, M may add one to the counter or not, depending on its current state and the tape symbol it is scanning, according to its transition function. It queries the oracle by entering a distinguished oracle query state. If the current value of the counter is n, then M transits to a distinguished "yes" state if $n \in A$ and to a distinguished "no" state if $n \notin A$. The counter is reset to 0.

For p to verify the correctness of the oracle responses, we assume that the format of the encoding of configurations is $\beta\$0^n$, where β is the description of the current state, tape contents, and head position of M and n is the current value of the counter. If p discovers that M is in the oracle query state while scanning β, then after encountering the \$, it sets a variable $z := a$ and executes $z := f(z)$ for each occurrence of 0 after the \$, so that z will have the value $f^n(a)$ when the next $\#$ is seen. Then it tests $R(z)$ to determine whether $n \in A$. It then checks that in the subsequent configuration, M is in the "yes" or "no" state according as $R(z)$ is true or false, respectively, and that the counter has been reset.

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 Herbrand model of T_0 that would cause p not to halt is the one describing the infinite valid computation history of M on x in the case that M does not halt on x. Thus p halts on all Herbrand models of T_0, thus on all models of T_0, iff M halts on x.

We can further restrict the set S_x describing the start configuration of M to be empty by observing that S_x 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 S_x holds and halt immediately if not. This gives a ground theory T_0 consisting of (1) only, independent of the input x, at the expense of coding information about x in the scheme p. However, for purposes of the proof of (ii) below, it will be important that p not depend on the input x but only on the machine M.

Finally, we must produce a first-order theory T extending T_0 such that T is of no higher Turing degree than T_0 (that is, T is still Turing-equivalent to A) and every Herbrand model of T_0 extends to a model of T. Since the halting of p depends only on the Herbrand substructure, p will halt on all models of T iff it halts on all Herbrand models of T_0. The main issue here is that we must be careful to construct a T whose Turing complexity is no greater than that of the ground theory T_0, otherwise the lower bound will not hold. Note that the first-order theory generated by T_0 may not be suitable, because the best we can guarantee is that it is Σ_1 in A.

To construct T, we augment T_0 with all existential formulas of the form

$$\exists x \bigwedge_{n\in C} P(f^n(x)) \wedge \bigwedge_{m\in D} \neg P(f^m(x)), \tag{2}$$

where C and D are disjoint finite subsets of $\mathbb{N}$. We take T to be the set of logical consequences of T_0 and the formulas (2). Every Herbrand model of T_0 extends to a model of T, because new elements outside the Herbrand domain can be freely added as needed to satisfy the existential formulas (2).

To show that T is Turing-equivalent to A, we observe that since the theory is monadic, every first-order sentence reduces effectively via the laws of first-order logic to a Boolean combination of ground formulas $P(f^n(a))$ and existential formulas (2). The latter are all true in T, so every sentence is equivalent modulo T to a Boolean combination of ground formulas $P(f^n(a))$. Any such formula is consistent with T iff it is consistent with T_0, since as previously observed, every Herbrand model of T_0 extends to a model of T. Thus T Turing-reduces to T_0.

This argument shows that the halting problem for program schemes over models of T_0 or T is hard for Σ_1^A. Since A was arbitrary and both T_0 and T are Turing-equivalent to A, we are finished with the proof of (i).

Now we turn to problem (ii). For the upper bound, first we show that equivalence of schemes over models of T is Π_2^T. Equivalently, inequivalence of schemes over models of T is Σ_2^T. It suffices to show that inequivalence of schemes over models of T can be determined by an IND program over $\mathbb{N}$ with oracle T with an $\exists\forall$ alternation structure [5] (see also [6]). As above, we need only that ground entailment relative to T is decidable.

Let p and q be two schemes with input variables $\bar{x} = x_1, \ldots, x_n$. The schemes p and q are not equivalent over models of T iff there exists a complete extension of T with extra constants $\bar{c} = c_1, \ldots, c_n$ in which either

1. both p and q halt on input $\bar{c}$ and produce different output values;
2. p halts on $\bar{c}$ and q does not; or
3. q halts on $\bar{c}$ and p does not.

We start by selecting existentially the alternative 1, 2 or 3 to check.

If alternative 1 was selected, we simulate p and q on input $\bar{c}$, maintaining a finite set E of ground literals and using T and E as in the proof of (i) 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 nondeterministically guess a complete extension of T using 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 2 was selected, we simulate p on $\bar{c}$ 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 $\bar{c}$. By (i), this problem is $\Pi_1^{T \cup E}$, so we can solve it with a purely universally-branching IND computation.

The argument for alternative 3 is symmetric.

For the lower bound, we reduce the totality problem for oracle Turing machines with oracle A, a well-known Π_2^A-complete problem, to the equivalence problem (ii). The totality problem is to determine whether a given machine halts on all inputs. As above, it will suffice to consider $\Sigma = \{a, f, Q, R\}$.

Given a deterministic oracle Turing machine M with oracle A, let T_0 be the ground theory consisting of the formulas (1). Then T_0 is Turing-equivalent to A. We construct two schemes p and q with no input or output that are equivalent in all complete ground extensions of T_0 iff M halts on all inputs. The scheme p is the one constructed in the proof of (i). 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. As in the proof of (i), the oracle A is encoded by the formulas (1). This allows p to verify responses to the oracle queries in the computation history.

Now unlike the proof of (i), 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 $\#\alpha_0^y\#$ for some y. This check was not really necessary in the proof of (i), since a description of the start configuration on input x could be coded in T_0, but it is necessary here. But 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 $\#\alpha_0^x\#$ for some x exactly as p does and halt immediately when it encounters the second occurrence of #. Now p does not halt in the Herbrand model H iff the string represented by the truth values of $Q(f^n(a))$ either

(a) does not have a prefix of the form $\#\alpha_0^x\#$, or
(b) 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 (a) only. Therefore p and q are equivalent iff case (b) never occurs for p; that is, iff M halts on all inputs.

We construct the first-order theory T from T_0 as in the proof of (i) above. Thus the equivalence problem for schemes over models of T_0 or T is Π_2^A-hard. Since A was arbitrary and both T_0 and T are Turing-equivalent to A, we are done. □

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 1(ii) that there can exist no finite relatively complete axiomatization for first-order scheme equivalence over models of a theory T of any Turing degree. If such an axiomatization did exist, then the scheme equivalence problem over models of T would be r.e. in T.

In the case $\alpha = 0$, it is decidable whether a given first-order sentence ϕ is a consequence of a given finite set E of ground formulas over the signature $\Sigma = \{a, f, P\}$, since $E \vDash \phi$ iff $E \to \phi$ is a valid sentence of the first-order theory of a one-to-one unary function with monadic predicate, a well-known decidable theory [3] (note that every Σ-structure is elementarily equivalent to one in which the interpretation of f is one-to-one). By Theorem 1(ii), the scheme equivalence problem is Π^0_2-hard relative to E, therefore also relative to the decidable first-order theory generated by E.

Acknowledgments

Thanks to Andreas Blass for insightful comments, which inspired a strengthening of the results. This work was supported in part by NSF grant CCF-0635028.

References

1. Angus, A., Kozen, D.: Kleene algebra with tests and program schematology. Tech. Rep. TR2001-1844, Computer Science Department, Cornell University (July 2001)
2. Barth, A., Kozen, D.: Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Tech. Rep. TR2002-1865, Computer Science Department, Cornell University (June 2002)
3. Ferrante, J., Rackoff, C.: The computational complexity of logical theories. Lecture Notes in Mathematics, vol. 718. Springer, Heidelberg (1979)
4. Harel, D., Meyer, A.R., Pratt, V.R.: Computability and completeness in logics of programs. In: Proc. 9th Symp. Theory of Comput., pp. 261–268. ACM, New York (1977)
5. Harel, D., Kozen, D.: A programming language for the inductive sets, and applications. Information and Control 63(1-2), 118–139 (1984)
6. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
7. Manna, Z.: Mathematical Theory of Computation. McGraw-Hill, New York (1974)