ON THE Π_2^0-COMPLETENESS OF CONTEXTUAL EQUIVALENCE

AKRAM EL-KORASHY

ABSTRACT. In this note, we present a proof of Π_2^0-completeness of contextual equivalence of Turing machines via a reduction of the language `FIN` to contextual inequivalence. `FIN` is the language of encodings $\langle M_i \rangle$ of Turing machines M_i for which the set W_i of inputs on which M_i halts (or the set of accepting inputs of M_i) is finite. `FIN` is known to be Σ_2^0-complete.

1. Introduction

The complexity of contextual equivalence and other similar notions of equivalence of programs (or infinite objects described by λ-terms) has been thoroughly studied in [1]. In this note, we describe a simple notion of contextual equivalence of deterministic Turing machines, expressed as co-termination in terms of Kleene's T predicate. We show the existence of a reduction to the complement of this notion of contextual equivalence from the set `FIN` of indexes $\langle M_i \rangle$ of deterministic Turing machines M_i for which the set W_i of inputs on which M_i halts is finite. The set `FIN` is known to be Σ_2^0-complete [2]. Showing Σ_2^0-completeness of contextual inequivalence is the same as showing Π_2^0-completeness of contextual equivalence [2]. The significant consequence of proving Π_2^0-completeness of contextual equivalence [1] is knowing that the set of pairs of contextually-equivalent Turing machines is non-recursively enumerable [3].

2. Definitions and Informal Proof

We recall some facts and claims about encodings of Turing machines and Kleene's arithmetical hierarchy from [5, 4, 2], and define the notion of contextual equivalence that we use in the hardness proof.

Membership in Kleene's arithmetical hierarchy can be described as follows:

Recursive relations: $\Sigma_0^0 = \Pi_0^0 =$ the class of all recursive relations over natural numbers.

Σ_{n+1}^0-relations: If the relation $R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is Π_n^0 then the relation $S(n_1, \ldots, n_l) = \exists m_1 \cdots \exists m_k R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is defined to be Σ_{n+1}^0.

Π_{n+1}^0-relations: If the relation $R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is Σ_n^0 then the relation $S(n_1, \ldots, n_l) = \forall m_1 \cdots \forall m_k R(n_1, \ldots, n_l, m_1, \ldots, m_k)$ is defined to be Π_{n+1}^0.

Key words and phrases. Contextual Equivalence, Arithmetical Hierarchy.

[1] Proving so is not a novel contribution of this note, but was nevertheless discovered independently of [1] and [3], and is done by reduction from a different problem.

Definition 2.1 (Kleene's predicate)**.** By Kleene's normal form theorem [5], we know that for every k there is a *recursive* $(k+2)$-ary predicate T_k (called Kleene's predicate) such that for a Gödel number $\langle M\rangle$ of a Turing machine M, inputs $i_1,\dots,i_k$, and a number x, $T_k(\langle M\rangle, x, i_1,\dots,i_k)$ holds iff x encodes a halting configuration of M on inputs $i_1,\dots,i_k$.

Remark 2.2. T_k is in $\Sigma^0_0 = \Pi^0_0$.

Definition 2.3 (Contextual Equivalence (Co-termination))**.**

$$\begin{aligned} ctxeq(\langle M\rangle, \langle M'\rangle) \overset{def}{=} \forall i_1,\dots,i_k, x, \exists x_h, x'_h.& \\ &(\neg T_k(\langle M\rangle, x, i_1,\dots,i_k) \wedge \neg T_k(\langle M'\rangle, x, i_1,\dots,i_k)) \vee \\ &(T_k(\langle M\rangle, x_h, i_1,\dots,i_k) \wedge T_k(\langle M\rangle, x'_h, i_1,\dots,i_k)) \end{aligned}$$

where $\langle M\rangle, \langle M'\rangle$ are Gödel encodings of Turing machines M and M' each computing a k-ary function.

Remark 2.4. *ctxeq* is a Π^0_2-relation (follows from Remark 2.2 and the definition of a Π^0_2-relation).

Remark 2.5. $\overline{ctxeq}$ is a Σ^0_2-relation (follows from Remark 2.4 by observing that negating a formula in prenex normal form results in a formula with every quantifier flipped).

Definition 2.6 (Accepting inputs of a Turing machine)**.** For a Turing machine M computing a unary function, define the set $W_{\langle M\rangle}$ of accepting inputs as $\{i \mid \exists x.\ T_1(\langle M\rangle, i, x)\}$.

Definition 2.7 (Turing machines with finitely many accepting inputs)**.** The set $\texttt{FIN} = \{\langle M\rangle \mid W_{\langle M\rangle} \text{ is finite}\}$ is given by the following predicate:

$$\texttt{FIN}(\langle M\rangle) \overset{def}{=} \exists n \forall i \forall x.\ |i| < n \vee \neg T_1(\langle M\rangle, i, x)$$

Definition 2.8 (Σ^0_n (Π^0_n)-completeness [2])**.** A set A is Σ^0_n-complete if it is in Σ^0_n and each Σ^0_n-set B (i.e, B is described by a Σ^0_n-relation) is many-one reducible to A, i.e., there is a recursive function f such that $i \in B$ iff $f(i) \in A$. And similarly for Π^0_n-completeness.

Remark 2.9. $\texttt{FIN}$ is Σ^0_2-complete [2].

Remark 2.10. A set is Π^0_n-complete iff its complement is Σ^0_n-complete [2].

Remark 2.11. If a set A is Σ^0_n (resp. Π^0_n)-complete, and A is many-one reducible to B, and B is in Σ^0_n (resp. Π^0_n), then B is Σ^0_n (resp. Π^0_n)-complete (follows from Definition 2.8 and composability of recursive functions [5]).

Lemma 2.12 ($\texttt{FIN}$ is many-one reducible to contextual **in**equivalence)**.** *There exist recursive functions f_1, f_2 such that $\forall i\ i \in \texttt{FIN} \iff (f_1(i), f_2(i)) \in \overline{ctxeq}$.* [2]

Here is an informal proof:

[2]Many-one reducibility as expressed here with the existence of two functions f_1 and f_2 can be described as per Definition 2.8 with just one function f that returns the encoding of a pair of the results of f_1 and f_2. But for convenience, we already alternatively defined *ctxeq* to be a binary relation.

f_1 **exists:** Define $f_1(\_) \stackrel{def}{=} \langle M_\perp \rangle$ to be the constant function that ignores its input and returns the Gödel encoding $\langle M_\perp \rangle$ where $M_\perp$ is the Turing machine that computes the everywhere-undefined function $f(x) = \perp$ (i.e., $M_\perp$ does not halt on any input).

f_2 **exists:** Define $f_2(\langle M \rangle) \stackrel{def}{=} \langle M_{search} \rangle$ where M_{search} is the Turing machine that computes the function: $\mu\ maxlen.\ \forall n' \geq maxlen.\ \forall w.\ (|w| = n' \implies \neg \exists x.\ T_1(\langle M \rangle, w, x))$

$\langle M \rangle \in \mathtt{FIN} \implies (f_1(\langle M \rangle), f_2(\langle M \rangle)) \in \overline{ctxeq}$**:** For an arbitrary number $\langle M \rangle$, under the assumption $\langle M \rangle \in \mathtt{FIN}$, we get from definition 2.7 that there is a maximum length n where any input i whose length $|i|$ is $\geq n$ must satisfy the formula $\forall x\ \neg T_1(\langle M \rangle, i, x)$. By obtaining such maximum length n, we show that it satisfies the search criterion for the $\mu maxlen$ search operator that M_{search} computes (by putting the search criterion in prenex normal form). So we know that M_{search} will halt. But we also know that $M_\perp$ never halts, so we conclude by definition 2.3 that $ctxeq(\langle M_\perp \rangle, \langle M_{search} \rangle)$ will not hold. So, $(f_1(\langle M \rangle), f_2(\langle M \rangle)) \in \overline{ctxeq}$, which is our thesis.

$\langle M \rangle \in \mathtt{FIN} \impliedby (f_1(\langle M \rangle), f_2(\langle M \rangle)) \in \overline{ctxeq}$**:** For an arbitrary number $\langle M \rangle$, under the assumption that $(f_1(\langle M \rangle), f_2(\langle M \rangle)) \in \overline{ctxeq}$, we know that $M_\perp$ which is encoded by $f_1(\langle M \rangle)$ never halts. And from our assumption that $f_2(\langle M \rangle) = \langle M_{search} \rangle$ is not contextually equivalent to $\langle M_\perp \rangle$, then we conclude that there must be at least one value for *maxlen* for which M_{search} halts. Obtaining this value *maxlen*, we observe that substituting it for the existentially quantified n in definition 2.7 makes the predicate `FIN` true for $\langle M \rangle$, which satisfies our thesis.

For numbers $i \in N$ **with** $\neg \exists M.\ i = \langle M \rangle$**:** we can just assume since $i \notin \mathtt{FIN}$ that we can construct $f_1 = f_2$ that returns the encoding $\langle M_1 \rangle$ of any arbitrary Turing machine M_1 (say one computing a constant function) which then satisfies $(f_1(i), f_1(i)) \notin \overline{ctxeq}$ because $\overline{ctxeq}$ is irreflexive. And so, it sufficed for the proof sketch to have otherwise mentioned $\langle M \rangle$ in place of i. Note that checking the predicate "$\neg \exists M.\ i = \langle M \rangle$", i.e., whether a number i is a valid encoding of a Turing machine is decidable. A similar decidability result for the encoding of a model called Unlimited Register Machines (URMs) exists in [5].

This concludes the proof, which is admittedly informal. But it can be made more rigorous by enhancing it with claims similar to the ones in [5] for URMs about why f_1 and f_2 –which construct encodings of Turing machines– are computable.

Theorem 2.13 (Contextual Equivalence is Π_2^0-complete)**.** *ctxeq is Π_2^0-complete.*

Proof. By applying remark 2.9, lemma 2.12, and remark 2.5 to remark 2.11, we conclude that $\overline{ctxeq}$ is Σ_2^0-complete.

So, by remark 2.10, we conclude that *ctxeq* is Π_2^0-complete. □

3. Discussion and Conclusion

Contextual equivalence of Turing machines. In definition 2.3, we present a rather weak notion of contextual equivalence which assumes that every "experiment" or "context" that attempts to distinguish M and M' consists of exactly one execution of each of M and M'. In the setting of Turing machines, one might expect that the notion of contextual equivalence that is familiar from functional programming can be modeled by allowing an arbitrary universal Turing machine (i.e., a context) to take as input the encodings $\langle M\rangle$ and $\langle M'\rangle$, and then contextual equivalence would hold whenever there is no universal Turing machine that halts on one but not the other. The problem with such a notion is that it would be too restrictive that it is actually *wrong* because it would be simply equivalent to string equality, i.e., equality of the encodings $\langle M\rangle$ and $\langle M'\rangle$. We are not aware of more standard languages (than *ctxeq* which is defined in 2.3) in the Turing machine setting that would capture the notion of contextual equivalence more precisely.

Conclusion. In this note, we presented for the notion *ctxeq* of contextual equivalence for Turing machines a proof of Π^0_2-completeness via a reduction from `FIN` to its negation $\overline{ctxeq}$. After working out the proof, we discovered that the result is not novel; similar results exist in [3] from 2006 and [1] from 2012. The significance of the Π^0_2-completeness statement is that the set of pairs of contextually equivalent Turing machines is non-recursively enumerable [3]. Another interpretation of Π^0_2-completeness is undecidability even in the existence of a halting problem oracle [6].

References

1. J. Endrullis, D. Hendriks, R. Bakhshi, *On the Complexity of Equivalence of Specifications of Infinite Objects*, ICFP '12 Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming Pages 153-164
2. P. Hajek, *Arithmetical Hierarchy and Complexity of Computation*, Theoretical Computer Science 8 (1979) 227-237.
3. G. Rosu, *Equality of Streams is a Π^0_2-Complete Problem*, ICFP '06 Proceedings of the eleventh ACM SIGPLAN International Conference on Functional Programming Pages 184-191
4. T. L. Wong *MODEL THEORY OF ARITHMETIC Lecture 1: The arithmetic hierarchy*, `http://www.logic.univie.ac.at/~wongt9/teach/modelarith/1_arithhier.pdf`
5. N.J. Cutland *Computability – An introduction to recursive function theory*, 1982
6. R. A. Shore, Theodore A. Slaman. *Defining the Turing jump.* Mathematical Research Letters 6.5/6 (1999): 711-722.

Akram El-Korashy, MPI-SWS
Current address: MPI-SWS
E-mail address: `first.last@mpi-sws.org`