CSE396 Problem Set 10 Answer Key Spring 2017

The **Final Exam** is set for **Tuesday, May 16** in the lecture room, **Cooke 121**, **11:45am–2:45pm**. We have decided to repeat the terms used for last year's exam in the identical room: *one notes binder* is allowed. The textbook is **not** allowed, and *all electronic devices* are forbidden. If you must bring a backpack or other bulky bag into the room at all, it must be stowed along the side of the room.

(1) Consider the following decision problem:

Exception Throw
Instance: A Java program P and an input stream $x \in ASCII^*$.
Question: Does running P on x throw an ArrayIndexOutOfBoundsException?

Show that this decision problem is undecidable—by arguing concretely that if it were decidable, then the Acceptance Problem for Turing machines would be decidable, which it isn't. (It may help you to know that the *Turing Kit* program, though it has other bugs, never throws this exception. 18 pts.)

Answer: Given a TM M and an argument w to M, define a Java program P as follows: P is the "Turing Kit" program (or JFLAP or some other Turing machine simulator) with two modifications:

- It has M and w embedded in its code, so that when P starts up on any input x (which is ignored) it begins a simulation of M on input w.
- Right after the line that displays the "String accepted" dialog, it has a line that throws the exception—which importantly is the only way that exception can be thrown when running P.

If we had a decider R for the Exception Throw problem, then we could chain R to the above conversion of $\langle M, w\rangle$ into P, and so would get a decider Q for the A_{TM} problem. To wit, $\langle M, w\rangle \in A_{TM} \equiv M$ accepts $w \implies P$ on any x eventually displays "String accepted" and throws the exception, which $\implies \langle P, \lambda\rangle$ is in the language of the Exception Throw problem. But $\langle M, w\rangle \notin A_{TM} \equiv M$ does not accept $w \implies P$ never throws the exception (because the above is the only way the exception can be thrown), so $\langle P, \lambda\rangle \notin$ Exception Throw. So Q, coded to translate $\langle M, w\rangle$ into (say) $\langle P, \lambda\rangle$ and accept iff R accepts $\langle P, \lambda\rangle$, would be a decider for the A_{TM} problem. But since A_{TM} is undecidable, Q is "quixotic" and cannot exist. Hence the decider R cannot exist so Exception Throw is undecidable.

Alternately, you could use just the second modification and make P treat whatever input x comes on `System.in` as a machine+input pair, loading and running them if x successfully decodes as such. If you then translate $f'(\langle M, w\rangle) = \langle P, \langle M, w\rangle\rangle$ (rather than $f(\langle M, w\rangle) = \langle P, \lambda\rangle$ which is implictly going on above), the decoding will be successful and we again get that $P(x)$ throws the exception if and only if M accepts w. Either way, focusing on the logic of $\implies$ and $\iff$ and the translation function f or f' is all you really need—the "window-dressing" of "R" and "S" is automatic. So answers phrased as reductions can be shorter than this one.

(2) Suppose we have a computable function $f(\langle M, w\rangle) = \langle M'\rangle$ with the following properties for any given TM M and string w:

- If M accepts w, then $L(M') = \Sigma^*$.
- If M does not accept w, then $L(M') = \emptyset$.

Explain why this not only mapping-reduces A_{TM} to NE_{TM}, it also mapping-reduces A_{TM} to ALL_{TM} and to K_{TM}. (Note that the latter supplements the simple way K_{TM} was reduced to A_{TM} via the simple function $f'(x) = \langle x, x\rangle$, so that K_{TM} and A_{TM} are *mapping equivalent*—more usually called *many-one equivalent* and written $\equiv_m$ either way. 6 + 12 = 18 pts.)

Answer: All we need to do is continue the $\implies$ logic and see what $\iff$ relations it leads to:

- $\langle M, w\rangle \in A_{TM} \equiv M$ accepts $w \implies$ for all x, M' accepts $x \implies L(M') = \Sigma^* \implies \langle M'\rangle \in ALL_{TM}$.
- $\langle M, w\rangle \notin A_{TM} \equiv M$ does not accept $w \implies$ for all x, M' does not accept $x \implies L(M') = \emptyset \implies L(M') \neq \Sigma^* \implies \langle M'\rangle \notin ALL_{TM}$.
- So $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in ALL_{TM}$, so $A_{TM} \leq_m ALL_{TM}$ via this f.
- $\langle M, w\rangle \in A_{TM} \equiv M$ accepts $w \implies$ for all x, M' accepts $x \implies M'$ accepts the particular x that happens to be its own code $\implies \langle M'\rangle \in K_{TM}$.
- $\langle M, w\rangle \notin A_{TM} \equiv M$ does not accept $w \implies$ for all x, M' does not accept $x \implies M'$ does not accept the particular x that happens to be its own code $\implies \langle M'\rangle \notin K_{TM}$.
- So $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in K_{TM}$, so $A_{TM} \leq_m K_{TM}$ via this selfsame f.

(3) Show that the language $\text{REGULAR}_{\mathsf{CFG}} = \{\text{ CFGs } G : L(G) \text{ is regular }\}$ is undecidable. Note that this is not the same as deciding whether the rules of G obey the format of a "regular grammar" as described on HW7. Rather, the question is whether the language of G is regular. Use the fact that the language of accepting computation histories of a single-tape TM is not regular *unless it is empty*[1], and do a reduction from E_{TM}. (18 pts., for 54 on the set)

Answer: We can convert any given TM M into the grammar $f(M) = G$ that generates the complement of ACH_M (that is ACH for M). Then

- $\langle M\rangle \in E_{TM} \implies L(G) = \Sigma^* \implies \langle G\rangle \in \text{REGULAR}_{\mathsf{CFG}}$; since Σ^* is regular, but
- $\langle M\rangle \notin E_{TM} \implies \mathsf{ACH}_M \neq \emptyset \implies \mathsf{ACH}_M$ is not regular $\implies L(G)$ is not regular (since the class REG is closed under complements) $\implies \langle G\rangle \notin \text{REGULAR}_{\mathsf{CFG}}$

Since the process of building G is computable given any TM M, we have a computable function f such that for all M, $\langle M\rangle \in E_{TM} \iff f(\langle M\rangle) \in \text{REGULAR}_{\mathsf{CFG}}$, so $E_{TM} \leq_m \text{REGULAR}_{\mathsf{CFG}}$, and since E_{TM} is undecidable, this shows that $\text{REGULAR}_{\mathsf{CFG}}$ is undecidable.

[1] Technically, say if there is only one accepting computation C, then the language ACH of accepting computation histories would be finite and hence regular. To fix this, we re-define ACH to allow configurations to have any number m of extra trailing @ chars, provided all configurations in the history have the same m. This and the fact that any halting C has at least two configurations in it makes ACH "embed" the language $\{@^m\#@^m : m \geq 0\}$ so that ACH is non-regular, unless it is empty.