Solutions: Homework 7

Prepared by Chien-Chung Huang

1.

<u>Answer</u>: Let L be an infinite Turing recognizable languages. Let E be an enumerator for L (since L is Turing recognizable, E exists). In the following, we prove the problem statement by specifying an enumerator E' that enumerates a subset of L in canonical order. Informally, E' runs E and, instead of outputting every string that E outputs, E' outputs only some of the strings output by E. Specifically, E' uses the following rule: each time E outputs a string s, E' includes s in its output if and only if s is bigger than the strings that E' has already included in its output. The following is a more precise specification of the enumerator E':

E' = "(1) LastStringOutput = *none*
(2) Run E until it outputs a string s.
(3) **if** (LastStringOutput = *none*) or ($s >$ LastStringOutput) **then**
output s
LastStringOutput = s
(4) Go to step (2) to resume the running of enumerator E."

We make the following observations:

a) $L(E')$ is a subset of $L(E)$ (because every string output by E' is output also by E).

b) E' enumerates strings in canonical order (because of the condition in Step 3).

c) $L(E')$ is infinite (because, since $L(E)$ is infinite, E eventually outputs a string bigger than every string that it has output thus far).

The statement of Problem 2 follows from the above three observations.

2.1

We create a decider to show that this U_{DFA} is decidable. The basic idea to mark the states which can be reached by the transition functions step by step. Eventually, if all states are marked, then we reject, otherwise, we accept.

Decider for U_{DFA}: "On input $< D >$, where D is a description of DFA
(1) If $< D >$ does not describe a DFA, reject.
(2) Mark the start state of D.
(3) Mark any state that has a transition into it from a marked state.
(3.1) Repeat Step 3 until no further state can be marked.
If any state is unmarked, ACCEPT, else, REJECT".

2.2

The idea is to utilize the machine E_{PDA} to prove U_{PDA}'s decidability. Specifically, we can transform the PDA into a set of PDAs and uses the E_{PDA} to decide whether this PDA contains a useless state.

Decider for U_{PDA} "On input $< P >$, where P is a description of PDA.
(1) If $< P >$ does not describe a PDA, reject.
(2) For all $q \in Q$
(2.1) Set $F = q$ and creates a new description of PDA $< P' >$
(2.2) Run E_{PDA} on $< P' >$.
(2.3) If E_{PDA} accepts, ACCEPT.
(3) REJECT."

2.3

The basic idea is to try to reduce U_{TM} to E_{TM}, which we know is a decider, to prove the undecidability of U_{TM}.

If a Turing Machine accepts empty language, then its accept state must be useless. More precisely, given a Turing Machine N, we can first use the decider M of U_{TM} (if such a machine exists) to decide which states in N are useful and which are not. If the accept state is useful, then this N is not describing an empty lanuage, otherwise, it is. In this manner, we create a decider for E_{TM}, which we know can not exist.

Decider for E_{TM} "On input $< N >$, where N is a description of some Turing Machine.
(1) Let Q be the set of states in N.
(2) If M rejects $< N >$, then REJECT.
(3) for each $Q' \subset Q$
(3.1) If M rejects $< N_{Q'} >$
(3.1.1) If $\forall q \notin Q'$, M accepts $< N_{Q' \cup q} >$
(3.1.1.1) If $q_{accept} \in Q'$, then REJECT, else ACCEPT."

3.1

a) $\{< M >|$ M is a TM and M accepts at least two strings $\}$

Answer: A is Turing recognizable, but not decidable.

To prove that A is Turing recognizable, we specify the following recognizer R for A:

$R =$ "On input $< M >$, where M is a TM,
(1) Convert M into an equivalent enumerator E.
(2) Run E.
(3) If E ever outputs two different strings, accept."

For a proof of undecidability of A, we note that the property " languages contains at least two strings" is a nontrivial property (because some Turing recognizable languages contain at least two strings and some Turing recognizable languages don't). Therefore, by Rice's theorem, A is undecidable.

Here is an alternative direct proof of undecidability of A. We present a reduction from A_{TM} to A:

"On input $< M, w >$, where M is a TM and w is a string:

(1) Construct Turing machine N that works as follows:

$N =$ "On input u:

(a) Run M on w.

(b) If M halts and accepts w, then accept.
If M halts and rejects w, then reject. "

(2) Output N. "

Observe that $L(N) = \Sigma^*$ if M accepts w
$= \phi$ if M does not accept w.
Importantly, whether or not $L(N)$ contains at least two strings depends on whether M accepts w. Hence, $< N >$ is in A if and only if $< M, w >$ is in A_{TM}. Therefore, the above algorithm is a reduction from A_{TM} to A, and hence A is undecidable.

3.2) $\{< M >| \ M$ is a TM and M accepts exactly two strings $\}$

Answer: B is not Turing recognizable. For a proof, we reduce the complement of A_{TM} to B:

"On input $< M, w >$, where M is a TM and w is a string:

(1) Construct Turing machine N that works as follows:

$N =$ "On input u:

(a) Accept if u is either 0 or 00.

(b) Run M on w.

(c) If M halts and accepts w, then accept.
If M halts and rejects w, then reject. "

(2) Output N. "

Observe that $L(N) = \{0, 00\}$ if M does not accept w
$= \Sigma^*$ if M accepts w.
Importantly, $L(N)$ contains exactly two strings if and only if M does not accept w. Hence, $< N >$ is in B if and only if $< M, w >$ is in the complement of A_{TM}. Therefore, the above algorithm is a reduction from the complement of A_{TM} to B, and hence B is not Turing recognizable.

3.3) $\{< M >| \ M$ is a TM and M halts when it runs on a tape that is initially empty $\}$
(This is the same languages as $\{< M >| \ M$ is a TM and M halts on input $\varepsilon\}$.)

Answer: C is Turing recognizable, but not decidable.

To prove that C is Turing recognizable, we specify the following recognizer R for C:

$R =$ "On input $< M >$, where M is a TM,

(1) Run M on ε.

(2) If M ever enters accept state, accept."

For a proof of undecidability of C, we reduce A_{TM} to C:

"On input $< M, w >$, where M is a TM and w is a string:
(1) Construct Turing machine N that works as follows:
$N =$ "On input u:
(a) Run M on w.
(b) If M halts and rejects w, then
run forever in an infinite loop.
If M halts and accepts w, then
halt and accept. "

(2) Output N. "

Observe that N halts on any input if and only if M accepts w. In particular, N halts on ε if and only if M accept w. Hence, $< N >$ is in C if and only if $< M, w >$ is in A_{TM}. Therefore, the above algorithm is a reduction from A_{TM} to C, and hence C is not decidable.

3.4) $\{< M_1, M_2 >|$ M_1 and M_2 are TMs over the input alphabet $\{0, 1\}$ and $L(M_1)$ is the complement of $L(M_2)$ $\}$

<u>Answer</u>: D is not Turing recognizable. For a proof, we reduce the complement of A_{TM} to D:

"On input $< M, w >$, where M is a TM and w is a string:
(1) Construct Turing machine N such that $L(N) = \Sigma^*$.
(2) Construct Turing machine N' that works as follows:
$N' =$ "On input u:
(a) Run M on w.
(b) If M halts and accepts w, then accept.
If M halts and rejects w, then reject. "

(2) Output $< N, N' >$. "

Observe that $L(N') = \phi$ if M does not accept w
$= \Sigma^*$ if M accepts w
Importantly, $L(N')$ is the complement of $L(N)$ if and only if M does not accept w. Hence, $< N, N' >$ is in D if and only if $< M, w >$ is in the complement of A_{TM}. Therefore, the above algorithm is a reduction from the complement of A_{TM} to D, and hence D is not Turing recognizable.

4. (12 pts) Prove that $EQ_{cfg} = \{< G_1, G_2 >| \, G_1$ and G_2 are CFGs and $L(G_1) = L(G_2)\}$ is co-Turing-recognizable (i.e., prove that the complement of EQ_{cfg} is Turing recognizable).

<u>Answer</u>: The following algorithm R is a recognizer for the complement of EQ_{cfg}, given by $\{< G_1, G_2 >| \, G_1$ and G_2 are CFGs and $L(G_1) \neq L(G_2)\}$
(below, $s_1, s_2, s_3, ...$ denote the strings from Σ^* in canonical order):

$R =$ "On input $< G_1, G_2 >$, where G_1 and G_2 are CFGs:
for $i = 1$ to ∞
Determine whether or not G_1 generates s_i
Determine whether or not G_2 generates s_i
if one of G_1 and G_2 generates s_i and the other does not then accept and halt."

Clearly, R is recognizer for the complement of EQ_{cfg}. Therefore, EQ_{cfg} is co-Turing-recognizable.