V22.0453-001 Honors Theory of Computation

Problem Set 3 Solutions

Problem 1

Solution: The class of languages recognized by these machines is the exactly the class of *regular languages*, thus this TM variant is not equivalent to the standard version. The intuition is that such a TM *cannot* remember what it has written on tape cells to the left of the current head position, as the TM is unable to return to these cells and thus its computation in the future is independent of the contents of these cells. Also for this reason, we can assume without loss of generality that the head of the TM never goes beyond the cell containing the last symbol of the input string. (Why?) Stay-put transitions can be simulated by ε-moves. Details follow.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ be any TM with stay-put instead of left. We constuct a NFA $N = (Q', \Sigma, \delta', q_{\text{start}}, F)$ that simulates M as follows. The special state q_{start} is the start state of N. Every other element of Q' (i.e. state of N) is of the form $\langle q, a\rangle$, where $q \in Q$ and $a \in \Gamma$. Set $F = \{\langle q_{\text{accept}}, a\rangle : a \in \Gamma\}$, where q_{accept} is the accept state of M.

We now describe the transition function δ'.

- For each $a \in \Sigma$, set $\delta'(q_{\text{start}}, a) = \{\langle q_0, a\rangle\}$, where q_0 is the start state of M.
- For each $p, q \in Q$ where $p \notin \{q_{\text{accept}}, q_{\text{reject}}\}$, for each $a \in \Gamma$, if M has a transition of the form $\delta(p, a) = (q, b, R)$, then for each $c \in \Sigma$, set $\delta'(\langle p, a\rangle, c) = \{\langle q, c\rangle\}$.
- For each $p, q \in Q$ where $p \notin \{q_{\text{accept}}, q_{\text{reject}}\}$, for each $a \in \Sigma$, if M has a transition of the form $\delta(p, a) = (q, b, S)$, then set $\delta'(\langle p, a\rangle, \varepsilon) = \{\langle q, b\rangle\}$.
- For each $a \in \Gamma$ and $c \in \Sigma$, set $\delta'(\langle q_{\text{accept}}, a\rangle, c) = \{\langle q_{\text{accept}}, a\rangle\}$.
- For each $a \in \Gamma$ and $c \in \Sigma$, set $\delta'(\langle q_{\text{reject}}, a\rangle, c) = \{\langle q_{\text{reject}}, a\rangle\}$.

It is not hard to see that $L(N) = L(M)$.

On the other hand, it is not hard to see that every NFA can be simulated by TM with stay-put instead of left.

Problem 2

Solution:

(a) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a TM M that recognizes $L_1 \cup L_2$: On input w,

1. Run M_1 and M_2 on w "in parallel". That is, in each step, M runs one step of M_1 and one step of M_2.
2. If either of M_1 and M_2 accepts, output ACCEPT. If both reject, output REJECT.

If M_1 or M_2 accepts w, then M will halt and accept w since M_1 and M_2 are run in parallel and an accepting TM will halt and accept w in a finite number of steps. If both M_1 and M_2 reject w, then M will reject w. If neither M_1 nor M_2 accepts w and one of them loops on w, then M will loop on w. Thus $L(M) = L_1 \cup L_2$, and Turing-recognizable languages are closed under union.

(b) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a NTM N that recognizes $L_1 \circ L_2$: On input w,

1. Nondeterministically split w into two parts $w = xy$.
2. Run M_1 on x. If it rejects, halt output REJECT. If it accepts, go to Step 3.
3. Run M_2 on w. If it accepts, output ACCEPT. If it rejects, output REJECT.

If $w \in L_1 \circ L_2$, then there is a way to split w into two parts $w = xy$ such that $x \in L_1$ and $y \in L_2$, thus, M_1 halts and accepts x, and M_2 halts and accepts y. Hence at least one branch of N will accept w. On the other hand, if $w \notin L_1 \circ L_2$, then for every possible splitting $w = xy$, $x \notin L_1$ or $y \notin L_2$, so M_1 does not accept x or M_2 does not accept y. Thus, all branch of N will reject w. Therefore, $L(N) = L_1 \circ L_2$, and Turing-recognizable languages are closed under concatenation.

(c) Let L be a Turing-recognizable languages, and let M be a TM that recognizes L. We construct a TM M' that recognizes L^*: On input w,

1. Nondeterministically choose an integer k and split w into k parts $w = w_1 w_2 \cdots w_k$.
2. Run M on each w_i. If M accepts all of $w_1, \ldots, w_k$, output ACCEPT. If M rejects one of $w_1, \ldots, w_k$, output REJECT.

If $w \in L^*$, then there is a splitting of $w = w_1 w_2 \ldots w_k$ such that $w_i \in L$ for each i, and thus at least one branch of M' will accept w. On the other hand, if $w \notin L^*$, then for every possible splitting $w = w_1 w_2 \ldots w_k$, $w_i \notin L$ for at least one i, thus all branches of M' will reject w. Therefore, $L(M') = L^*$, and Turing-recognizable languages are closed under the star operation.

(d) Let L_1 and L_2 be two Turing-recognizable languages, and let M_1 and M_2 be TMs that recognizes L_1 and L_2 respectively. We construct a TM M that recognizes $L_1 \cap L_2$: On input w,

1. Run M_1 on w. If it rejects, halt and output REJECT. If it accepts, go to Step 2.
2. Run M_2 on w. If it accepts, output ACCEPT. If it rejects, output REJECT.

Clearly $L(M) = L_1 \cap L_2$, and thus Turing-recognizable languages are closed under intersection.

Problem 3

Solution: Given a c.f.g. G, we give an algorithm to decide whether $L(G)$ is infinite. Let n be the number of variables in the grammar and b be the maximum length of the right hand side of any rule. We know from the proof of pumping lemma that $p = b^n$ serves as the pumping length.

We show that $L(G)$ is infinite $\iff$ it contains a string of length l where $p < l \leq 2l$.
(Proof of $\Longleftarrow$:) If there is a string $s \in L(G), |s| > p$, we know by pumping lemma that no matter how one partitions $s = uvwxy, |vxw| \leq p, |vx| > 0$, we have $uv^iwx^iy \in L(G)\ \forall\ i \geq 0$. Thus $L(G)$ is infinite.
(Proof of $\Longrightarrow$:) If $L(G)$ is infinite, then let $s_1 \in L(G)$ be a string of length $> p$. Partition $s_1 = uvwxy, |vxw| \leq p, |vx| > 0$. We know by pumping lemma that the string $s_2 = uwy \in L(G)$. Note that $|s|_1 - p \leq |s_2| < |s_1|$. We can use this argument repetitively and produce a sequence of strings $s_1, s_2, s_3, \ldots$ such that all of them are in $L(G)$ and

$$|s_i| > p \implies |s_i| - p \leq |s_{i+1}| < |s_i|$$

Hence we will find a string s_{i_0} whose length l satisfies $p < l \leq 2p$.

Thus in order to decide whether $L(G)$ is infinite, it suffices to decide whether the grammar generates a string of length l where $p < l \leq 2l$. This can be done by trying out all strings of length in this range.

Problem 4

Solution: We observe that $L(R) \subseteq L(S)$ if and only $L(R) \cap \overline{L(S)} = \emptyset$. Therefore the following TM decides

$$L = \{\langle R, S\rangle : R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\}.$$

On input $\langle R, S\rangle$:

1. Construct a DFA D such that $L(D) = L(R) \cap \overline{L(S)}$.
2. Run a TM T that decides $\mathrm{E_{DFA}}$ on input $\langle D\rangle$ (i.e. decides whether $L(D)$ is empty. We did this in class).
3. If T accepts, output ACCEPT. If T rejects, output REJECT.

Problem 5

Solution: We show that $\mathrm{A_{TM}}$ reduces to $A = \left\{\langle M\rangle : w^R \in L(M) \text{ whenever } w \in L(M)\right\}$. Since $\mathrm{A_{TM}}$ is undecidable, it follows that A is undecidable.

The reduction from $\mathrm{A_{TM}}$ to A maps $\langle M, w\rangle$ to $\langle M'\rangle$ where M' is the following TM with M and w built in:
On input x:

1. If $x = 01$, then halt and output ACCEPT.

2. Else if $x \neq 10$, then halt and output REJECT.

3. Else (if $x = 10$), then simulate M on w. If M accepts w, then output ACCEPT; if M rejects w, then output REJECT.

It is easy to see that if $\langle M, w\rangle \in \mathrm{A_{TM}}$, that is, if M accepts w, then $L(M') = \{01, 10\}$, so $\langle M'\rangle \in A$. On the other hand, if $\langle M, w\rangle \notin \mathrm{A_{TM}}$, that is, if M does not accept w, then $L(M') = \{01\}$, so $\langle M'\rangle \notin A$. Therefore the above mapping is a reduction from $\mathrm{A_{TM}}$ to A.

Note: You could also use Rice's Theorem.

Problem 6

Solution: Suppose that exactly k of the machines $\{M_1, M_2, M_3\}$ halt on corresponding inputs. The idea is to figure out the value of k. Once we know k, we can decide which ones halt: simply simulate the three machines on corresponding inputs simultaneously (one step of every machine at a time), and halt when exactly k of them halt. Since we know k, we are guaranteed to halt.

This is how we figure out the value of k. First construct a machine M that simulates the three machines on corresponding inputs simultaneously and halts if at least two of them halt. Now decide whether M halts by asking the oracle.

(Case 1): If M halts, we know that $k \geq 2$. By running the machines simultaneously again, we figure out which two of the machines halt. Then by asking the oracle, we can figure out whether the third machine halts.

(Case 2): If M does not halt, we know that $k \leq 1$. Construct a machine M' that simulates the three machines on corresponding inputs simultaneously and halts if at least one of them halts. Decide whether M' halts by asking the oracle. If M' halts, then we know that $k = 1$. If M' does not halt, we know that $k = 0$.