Solution sketch 5 - Computational Models - Spring 2014

1. A deciding TM will first check that the input is a legal description of a PDA (reject otherwise) and then convert it (as described in class) to a CFG that generates (only) the words accepted by the PDA. Now, the CFG is converted to Chomsky Normal Form - this ensures that for a word w of length n, exactly $2n-1$ steps are needed for its derivation (if possible). Finally, the TM will try all derivations of length $2|w|-1$ and decide accordingly regarding the derivability of w. Merely simulating a PDA using an NTM is wrong since the PDA (and therefore the NTM) might have computation branches that don't halt.

2. (a) $co-\mathcal{RE} \setminus \mathcal{R}$. A TM that accepts the complementing language goes over all strings alphabetically, and for each string check if it's in $L(A_1)$ but not in $L(A_2)$ or vice versa (justify why this can be decided!). If such a string exists, answer yes. $\notin \mathcal{R}$: We know that E_{LBA} is undecidable. We give a mapping reduction from E_{LBA}: $f(A) = (A, loopy)$ and we have that $L(A) = \emptyset$ iff $L(A) = L(loopy)$.

   (b) $\mathcal{RE} \setminus \mathcal{R}$. $\mathcal{RE}$: Let $x_1, x_2, ...$ be a lexical ordering of all strings. Use a universal TM U. for $i = 1...\infty$ run M on the inputs $x_1, ..., x_i$ for i steps. accept if M halts on any input (prove correctness). $\notin \mathcal{R}$: $H_{TM} \leq_m L$. Given $\langle M, w\rangle$, we can compute a TM $\langle M'\rangle$ in the following manner. M' will ignore its input x and simply run M on w and return *true*. This reduction is computable. If M halts on w then M' halts on all inputs as required. If M does not halt on w then M' does not halt on any input, as required.

   (c) $\mathcal{R}$. Given a configuration of M we can check if M will move the head to the left by simulating the run of M until M moves the head to the left or until M reads a blank symbol for $|Q|+1$ steps. One of these must happen within a finite time. A TM M_L that decides L will perform this test for the initial configuration with ε as input. If M didn't move the head to the left M_L accepts, otherwise M_L will preform the same test

for the ending configuration of the previous test. After 2015 moves of the head to the left M_L rejects.

(d) $co-\mathcal{RE} \setminus \mathcal{R}$. A TM that accepts the complementing language simulate the run of M on ε until the head moves three times in a row left. $\notin \mathcal{R}$: $A_{TM} \leq_m L$. Given $\langle M, w\rangle$, we can compute a TM $\langle M'\rangle$ in the following manner. M' will ignore the input x and simulate the run of M on w, except that every time M moves twice in a row to the left, M' will move once to the right and once to the left, and when M enters q_{acc} M' first moves 3 times in a row to the left and then accept. This reduction is computable. If M accepts w then M' moves 3 times in a row to the left on ant input, as required. If M does not accepts w then M' does not moves 3 times in a row to the left, as required.

(e) $co-\mathcal{RE} \setminus \mathcal{R}$. A non-deterministic TM for $\overline{L}$ would guess some word x not in $L(1(1\cup 0)^*)$, run M on x, and accept iff M accepted x. $\notin \mathcal{R}$: by Rice's theorem.

(f) Not in $\mathcal{RE} \cup co-\mathcal{RE}$. By the following mapping reduction from ALL_{TM} ($ALL_{TM} = \{\langle M\rangle \mid L(M) = \{0,1\}^*\}$): Given $\langle M\rangle$, return a description of a TM that does the following: if the input has the form $1x$, run M on x, and return the same answer, otherwise, reject.

3. (a) Let $L \notin \mathcal{R}$ we can show that:
   - $L \leq_m A(L)$ by the mapping reduction $f_0(x) = 0x$
   - $\overline{L} \leq_m A(L)$ by the mapping reduction $f_1(x) = 1x$

   Assume by contradiction that $A(L) \in \mathcal{RE}$, thus by the above reductions $L \in \mathcal{RE}$ and $\overline{L} \in \mathcal{RE}$ and therefore $L \in \mathcal{R}$, a contradiction.

   (b) Let $L \notin \mathcal{R}$. We know that $A(L) \notin \mathcal{RE}$, and thus $A(L) \notin \mathcal{R}$. We can show that the following is a mapping reduction from $A(L)$ to $\overline{A(L)}$:

$$f(x) = \begin{cases} 0y & x = 1y \text{ for some } y \in \Sigma^* \\ 1y & x = 0y \text{ for some } y \in \Sigma^* \\ 0 & x = \varepsilon, \varepsilon \in L \\ 1 & x = \varepsilon, \varepsilon \notin L \end{cases}$$

   Note that given L the decision about the value of $f(\varepsilon)$ is pre-determined and not part the computation, thus f is computable.

4. • **Intersection**. Let $L_1, L_2 \in \mathcal{P}$. Because $L_1 \in \mathcal{P}$ then there exists a TM M_1 with time complexity $O(n^{k_1})$ for some constant k_1. Because $L_2 \in \mathcal{P}$ then there exists a TM M_2 with time complexity $O(n^{k_2})$ for some constant k_2. We construct a decider M with polynomial time complexity deciding $L1 \cap L_2$: On input x: Run M_1 on input x. If M_1

accepted, then run M_2 on input x, else reject. If M_2 also accepted, then accept, else reject.
In the worst case M will run both M_1 and M_2, in which case it uses $O(n^{k_1}) + O(n^{k_2})$ steps. Let $k = max(k_1, k_2)$. We then see that M has time complexity $O(n^k)$ and hence $L(M) = L_1 \cap L_2 \in \mathcal{P}$.

- **Complement**. We simply swap the accept and reject state. The running time of the modified machine does not change.
- **Concatenation**. Assume so that $L_1 \in \mathcal{P}$ and that $L_2 \in \mathcal{P}$. By definition, this means that there exist deciders M_1 and M_2 such that M_1 is a decider for L_1 with time complexity $O(n^{k_1})$ and M_2 is a decider for L_2 with time complexity $O(n^{k_2})$ for some constants k_1 and k_2. The decider for $L_1||L_2$ given an input x, try to find a partition of x into x_1x_2 such that $x_1 \in L_1$ and $x_2 \in L_2$. Here is the decider:
  On input $x = a_1 \dots a_n$
  (a) For $i = 0$ to n do
    i. Let $x_1 = a_1 \dots a_i$ and $x_2 = a_{i+1} \dots a_n$. (By agreement $a1 \dots a_0 = \varepsilon$ and $a_{n+1} \dots a_n = \varepsilon$).
    ii. Run M_1 on the input x_1.
    iii. Run M_2 on the input x_2.
    iv. If both M_1 and M_2 accepted, then accept
  (b) If no choice of x_1 and x_2 led to acceptance, then reject

  The decider has polynomial time complexity. The main loop of the decider is traversed at most $(n+1)$-times. If we run M_1 on a substring of x, this will take at most $O(n^{k_1})$ steps. Similarly, running M_2 on a substring of x will take at most $O(n^{k_2})$ steps. Consequently, a single traversal of the loop body uses no more than $O(n^{k_1}) + O(n^{k_2}) = O(n^k)$ steps, where $k = max(k_1, k_2)$. The whole decider thus uses $(n+1) \cdot O(n^k) = O(n^{k+1})$ steps. Hence $L(M) = L_1||L_2 \in P$.

5. Assume that $L_1, L_2 \in \mathcal{NP}$. This means that there are nondeterministic deciders M_1 and M_2 such that M_1 decides L_1 in nondeterministic time $O(n^k)$ and M_2 deciders L_2 in nondeterministic time $O(n^l)$. The four machines M for the different operations:

   - **Intersection**: Similar to the previous question.
   - **Union**: On input w, M will run M_1 on w. If M_1 accepted then accept. Else run M_2 on w. If M_2 accepted then accept, otherwise, reject.
     the longest branch in any computation tree on input w of length n is $O(n^{\max\{k,l\}})$. So M is a poly-time nondeterministic decider for $L_1 \cup L_2$.

- **Concatenation**: On input w, M will Nondeterministically split w into w_1, w_2 such that $w = w_1w_2$. Run M_1 on w_1. If M_1 rejected then reject. Else run M_2 on w_2. If M_2 rejected then reject. otherwise accept.
  The longest branch in any computation tree on input w of length n is $O(n^{\max\{k,l\}})$
- **Kleene star**: On input w:
  (a) If $w = \varepsilon$ then accept.
  (b) Nondeterministically select a number m such that $1 \leq m \leq |w|$.
  (c) Nondeterministically split w into m pieces such that $w = w_1w_2 \dots w_m$.
  (d) For all i, $1 \leq i \leq m$: run M_1 on w_i. If M_1 rejected then reject.
  (e) Else (M_1 accepted all w_i, $1 \leq i \leq m$), accept.

  the total running time is $O(n^{k+1})$. This means that M is a poly-time nondeterministic decider for ${L_1}^*$.

6. - $\mathcal{P} \subseteq \mathcal{NP}$: $L \in \mathcal{P} \Rightarrow$ There exists a deterministic TM M that decides L in polynomial time $\Rightarrow$ M is a verifier (simply ignore the witness).
   - $\mathcal{NP} \subseteq \mathcal{EXP}$: $L \in \mathcal{NP} \Rightarrow$ There exists a polynomial $p(n)$ such that for an input x with $|x| = n$ there is a polynomial verifier that runs in time $p(n)$. $\Rightarrow$ We will build a TM s.t given an input x, looks at all the possible witnesses (there are $2^{p(n)}$ such witnesses). For each such witness, it runs the verifier for $p(n)$ steps $\Rightarrow$ The total running time is exponential