Computational Models, Spring 2013 Exrecise #4

Turing Machines $\mathcal{R}, \mathcal{RE}$, co-$\mathcal{RE}$

1. Let $A = \{Q_A, \Sigma, \delta_A, q_{0A}, F_A\}$ be a DFA and let $P = \{Q_P, \Sigma, \Gamma, \delta_P, q_{0P}, F_P\}$ be a deterministic PDA

   (a) Construct **formally** a TM that accepts $L(A)$
   Let $M = (Q, \Sigma, \Gamma, \delta, q_{0A}, q_a, q_r)$ where:
   $Q = Q_A \cup \{q_a, q_r\}$ $\quad \Gamma = \Sigma \cup \{\sqcup\}$ $\quad$ **and**
   $\delta(q, \sigma) = (\delta_A(q, \sigma), \sigma, R)$ **for** $\sigma \in \Sigma$.
   $\delta(q, \sqcup) = (q_a, \sqcup, R)$ **if** $q \in F_A$.
   $\delta(q, \sqcup) = (q_r, \sqcup, R)$ **if** $q \notin F_A$.

   (b) Construct **formally** a TM that accepts $L(P)$
   Use a two tape TM where the second tape will simulate the stack.
   Let $M = (Q, \Sigma, \Gamma, \delta, q_{0P}, q_a, q_r)$ where:
   $Q = Q_A \cup \{q_a, q_r\} \cup Q_{new}$ $\quad \Gamma = \Sigma \cup \{\sqcup\}$ $\quad$ **and**
   The transition function is: $\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, S\}^2$ **such that:**
   The head of the first tape always moves right. We always keep the context of the first tape intact.

   - **Reading a blank on the the second tape is like having an empty stack so we cannot perform a pop operation. We can follow all operation which include no pops. The second head stays in place:**
     $\delta(q, \sigma_1, \sqcup) = (q', \sigma_1, \sigma_2', R, S)$ **such that** $\delta_P(q, \sigma_1, \epsilon) = (q', \sigma_2')$
     Now, we know that the second head points to the top of the simulated stack.
   - **For every pop operation with no push operation we need to move the second head left: Thus, for every transition of the form** $\delta_p(q, \sigma_1, \sigma_2) = (q', \epsilon)$ **add a transition** $\delta(q, \sigma_1, \sigma_2) = (q', \sigma_1, \sqcup, R, L)$
   - **For every pop operation with a push operation we need to replace the head of the stack and not move the second head: Thus, for every transition of the form** $\delta_p(q, \sigma_1, \sigma_2) = (q', \sigma_2')$ **add a transition** $\delta(q, \sigma_1, \sigma_2) = (q', \sigma_1, \sigma_2', R, S)$
   - **For every push operation with no pop operation we need to move the head right and then write on the top of the stack. We do this by adding an intermediate transition that remembers our state and what we want to write on the second tape.**
     I.e. we add the following states: $Q_{new} = \{(q, \sigma) \ s.t. \ q \in Q_p$ **and** $\sigma \in \Sigma\}$
     and the following generic rule (for all $\sigma_1 \in \Sigma$ **and all new states):**
     $\delta((q, \sigma_2), \sigma_1, \sqcup) = (q, \sigma_1, \sigma_2, S, S)$
     Now, for every transition of the form $\delta_p(q, \sigma_1, \epsilon) = (q', \sigma_2')$ **add the following transitions** $\delta(q, \sigma_1, \sigma_2) = ((q', \sigma_2'), \sigma_1, \sigma_2, R, R)$ **when When we finished reading the input we check if we are in an accepting state or not:**
     $\delta(q, \sqcup, \sqcup) = \delta(q, \sqcup, \sigma) = (q_a, \sqcup, \sqcup, S, S)$ **if** $q \in F_P$.
     $\delta(q, \sqcup, \sqcup) = \delta(q, \sqcup, \sigma) = (q_r, \sqcup, \sqcup, S, S)$ **if** $q \notin F_P$.

   If we want to use only one tape, we can notice that the above works exactly the same just that the second tape is simulated after the input. We can add a symbol (say with the unused symbol #) after the input. Now, we need to go back and forth between the currently read input symbol and the currently use of the stack (push and / or pop). In order to know how to go to the stack we can simply move right

until we reach the first blank and then push or go back one and pop. In order to know how to return to the current input symbol, we need to mark every symbol that we have read (say by replacing it with the unused symbol #).
While moving to the stack section we need to:

(i) remember the state we are in.
(ii)remember the operation we want to do (push, pop or both).
While moving from the stack section back to the input section we need to:
(i) remember the state we are in.
Thus we add the states $Q_{to\_stack} = \{(q, \sigma_1, \sigma_2)\ s.t.\ q \in Q_p\ \sigma_1, \sigma_2 \in \Sigma \cup \{\epsilon\}\}$ **and** $Q_{from\_stack} = \{(q, L)\ s.t.\ q \in Q_p\}$**.**
To summarize:

When reading from the input we add the transition: $\delta(q, \sigma) = ((q', \sigma_1, \sigma_2), \#, R)$ **for every** $\delta_p(q, \sigma, \sigma_1) = (q', \sigma_2)$
Similarly add transitions for adding initial #, going right, performing push and pop

operations and going left.

2. Prove or disprove:
   (a) $\mathcal{R}$ is closed under complementation.
   True. If $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ **decides** L **then** $M = (Q, \Sigma, \Gamma, \delta, q_0, q_r, q_a)$ **decides** $\bar{L}$**.**
   (b) $\mathcal{RE}$ is closed under complementation.
   False. We know that $A_{TM} \in \mathcal{RE}$ **but** $\bar{A_{TM}} \notin \mathcal{RE}$
   (c) $\mathcal{RE}$ is closed under intersection.
   True. Given M_1 **which decides** L_1 **and** M_2 **which decides** L_2**, we can simply run** M_1 **on our input, then run** M_2 **on our input, and accept iff both accepted.**
   (d) co-$\mathcal{RE}$ is closed under intersection.
   True. Use definitions, De-Morgan rule and the fact that $\mathcal{RE}$ **is closed under union.**
   (e) $\mathcal{RE}$ is closed under Kleene star.
   True. Construct a non-deterministic TM that works as follows: it guesses how to split the input to an arbitrary number of parts, and then runs a TM that accepts M on all parts. It accepts iff M accepted all parts.

3. For each of the following languages, show that they are in $\mathcal{R}$.
   (a) $L = \{w\#w : w \in \{0, 1\}^*\}$
   Construct a TM that works as follows:
      i. **Check that the input is** $\{0, 1\}^*\#\{0, 1\}^*$
      ii. **If the current symbol is # verify that there are no symbols from** Σ **to its left. Accept if so and reject if not.**
      iii. **Else, let** σ **be the current symbol.**
         - **The TM remembers** σ **and replaces it with the symbol** x**.**
         - **The TM moves right to the first symbol after # that is different from** x **and compares to** σ**.**
           If they differ, reject. If the don't, replace it with x**.**
         - **The TM returns the head to the leftmost symbol that is not an** x
         - **GoTO (ii)**

   (b) $L = \{ww : w \in \{0, 1\}^*\}$
   Construct a TM that works as follows:
      i. **Check that the input length is even and record it's length** $2k$**.**

   ii. **Insert the symbol # at the k'th position**
   iii. **Run the previous algorithm**

4. Let L be a infinite language, prove or disprove:

   (a) For every such L there exists a language $L' \subset L$ such that L' **is not** decidable (i.e. $L' \notin \mathcal{R}$)
   Let L be an infinite language, the number of languages $L' \subset L$ is not enumerable. The number of languages in $\mathcal{R}$ is enumerable. Thus, there exists a language $L' \subset L$ such that L' is not decidable.

   (b) For every such L there exists a language $L' \subset L$ such that L' **is** decidable (i.e. $L' \in \mathcal{R}$)
   The statement is not true.
   Let $\Sigma = \{0\}$ and let $M_0, M_1 \ldots$ be an enumeration of TM that decide infinite languages. We will build a set of words $w_0, w_1 \ldots$ inductively:

   - **Let $0^i, 0^j$ be the two shortest words that are accepted by M_0. Let $w_0 = 0^j$**
   - **Assume that we built $w_0, \ldots w_n$: Let $0^i, 0^j$ be the two shortest words longer than $|w_n|$ that are accepted by M_{n+1}. Let $w_{n+1} = 0^j$**
   - **Define $L = \{w_0, w_1, \ldots\}$**

   Clearly, L is infinite. Let $L' \subset L$ be an infinite language. Assume falsely that it is decidable.
   Thus, there exists a TM M_i (from our enumeration) the decides it. During our construction we saw two words that were accepted by M_i and one of them is not in L'. Contradiction.

5. Let $L_1, L_2 \in \mathcal{RE} \setminus \mathcal{R}$. Prove whether the following is possible:

   (a) $L_1 \cup L_2 \in \mathcal{R}$. **Yes. Let $L_1 = \{< M >, x \; s.t. M$ is a TM that halts on x or has an even number of sta**
   Let $L_1 = \{< M >, x \; s.t. M$ is a TM that halts on x or has an odd number of states$\}$.
   It's easy to show that $L_1, L_2 \in \mathcal{RE} \setminus \mathcal{R}$ but $L_1 \cup L_2 \in \mathcal{R}$.

   (b) $L_1 \cup L_2 \in \mathcal{R}$ and $L_1 \cap L_2 \in \mathcal{R}$.
   No. Since then we could construct a TM that decides L_1. Let M_1 and M_2 be the TMs that accept L_1 and L_2 respectively. Let M_u and M_i be the TMs that decide $L_1 \cup L_2$ and $L_1 \cap L_2$ respectively. We shall construct the TM M that decides L_1. M works in the following manner on an input x: If $M_i(x) =$ *true*, return *true*. If $M_u(x) =$ *false*, return *false*. Otherwise, we run M_1 and M_2 in parallel on x. If M_1 accepts, return *true*. If M_2 accepts, return *false*.
   Correctness sketch: M_i and M_u are guaranteed to halt. If x is in the intersection we can return true. If x is not in the union, we can return false. Otherwise, it's either in $L1 \setminus L2$ or in $L2 \setminus L1$. Therefore, it's guaranteed that M_1 or M_2 will return true and we will halt with the correct answer.