Computational Models, Spring 2016 Exercise #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 Construct *formally* a TM that accepts $L(A)$

   Let $M = (Q, \Sigma, \Gamma, \delta, q_{0A}, q_a, q_r)$ **where:**
   $Q = Q_A \cup \{q_a, q_r\}$ $\Gamma = \Sigma \cup \{\sqcup\}$ **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** $q \notin F_A$**.**

2. In this question we are concerned with $< M >$, the encoding of M according to the convention we seen in class.

   (a) Let $M = \{Q, \Sigma, \Gamma, \delta, q_1, q_2, q_3\}$ be a TM such that: $Q = \{q_1, q_2, q_3\}$ $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$ and δ is defined as follows: $\delta(q_1, 0) = (q_1, 0, R)$; $\delta(q_1, 1) = (q_3, 1, L)$; $\delta(q_1, \sqcup) = (q_2, \sqcup, L)$

   i. What is the language accepted by the TM? (no need to prove your answer).
   The language is $\mathbf{0}^*$

   ii. What us $< M >$?

   To encode M, we only encode the transition function δ**. For each rule** $\delta(q_i, \gamma_j) = (q_k, \gamma_\ell, D_b)$**, we add the string** $0^i 1 0^j 1 0^k 1 0^\ell 1 0^b$**. Different rules are separated by 11. Notice that** $0 = \gamma_1,\ 1 = \gamma_2,$ **and** $\sqcup = \gamma_3$ **(see slide 26 in Lecture 8).**
   For the rule $\delta(q_1, 0) = (q_1, 0, R)$ **we have** $i = 1, j = 1, k = 1, l = 1, b = 2$**. Thus, it is encoded as** $0^1 1 0^1 1 0^1 1 0^1 1 0^2$**.**
   For the rule $\delta(q_1, 1) = (q_3, 1, L)$ **we have** $i = 1, j = 2, k = 3, l = 2, b = 1$**. Thus, it is encoded as** $0^1 1 0^2 1 0^3 1 0^2 1 0^1$**.**
   For the rule $\delta(q_1, \sqcup) = (q_2, \sqcup, L)$ **we have** $i = 1, j = 3, k = 2, l = 3, b = 1$**. Thus, it is encoded as** $0^1 1 0^3 1 0^2 1 0^3 1 0^1$**.**

   To conclude: $< M >= 0^1 1 0^1 1 0^1 1 0^1 1 0^2 1 1 0^1 1 0^2 1 0^3 1 0^2 1 0^1 1 1 0^1 1 0^3 1 0^2 1 0^3 1 0^1$

   (b) Write an algorithm (in pseudocode) that on input w, checks that w encodes a valid TM $< M >$. E.g., you need to validate that the structure is correct, that $\delta(q_2, a)$ is undefined for every symbol a, etc. (recall that q_2 is by convention the accepting state).

3. Give a verbal description of a Turing Machine that accepts the following languages. There is no need to give a formal definition but you should give a detailed explanation of each step.

   (a) $L = \{w \in \Sigma^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$ above $\Sigma = \{a, b, c\}$

   Main idea: Use multiple tapes: one for the input, one for counting the number of 'a's, one for counting the number of 'b's, and one for counting the number of 'c's. Move through input and for each symbol and a symbol on its tape. When input is read count that all three tapes are of same size.
   Pseudo-code:

   - **Part (1) - Initialize**
     - **Write \$ on tapes 2,3,4 and move Right on tapes 2,3,4**
     - **GoTo Part 2**
   - **Part (2) - Read**
     - **If input is** $(\sqcup, \sqcup, \sqcup, \sqcup)$ **GoTo Part 3.**
     - **If input is** $(a, \sqcup, \sqcup, \sqcup)$ **write** $(a, a, \sqcup, \sqcup)$ **and move right on tapes 1 and 2.**

  - **If input is $(b, \sqcup, \sqcup, \sqcup)$ write $(b, \sqcup, b, \sqcup)$ and move right on tapes 1 and 3.**
  - **If input is $(c, \sqcup, \sqcup, \sqcup)$ write $(c, \sqcup, \sqcup, c)$ and move right on tapes 1 and 4.**
- **Part (3) - Check**
  - **If input is $(\sqcup, \sqcup, \sqcup, \sqcup)$ write $(\sqcup, \sqcup, \sqcup, \sqcup)$ and move left on tapes 2-4.**
  - **If input is $(\sqcup, a, b, c)$ write $(\sqcup, \sqcup, \sqcup, \sqcup)$ and move left on tapes 2-4.**
  - **If input is $(\sqcup, \$, \$, \$)$ accept.**
  - **If input contains one or two \$ signs, reject.**

(b) $L = \{\{0\}^{n^2} \mid n \in N \text{ and } n > 0\}$ above $\Sigma = \{0, 1\}$

Main idea: Use multiple tapes: one for the input, one to remember which n we are testing now, one to serve as a counter, and one to compute the current value of n^2 and to compare to original tape. Move through input and for each symbol and a symbol on its tape.
Pseudo-code:

- **Write 1 on the second and third tape**
- **(†) Write on the fourth tape the number of zeros on the second tape.**
- **Decrement counter on third tape.**
- **If we have not reached zero, repeat step (†).**
- **If we have reached zero, compare with first tape: return accept if equal, return reject if fourth tape is longer than first, else continue by**
  - **increment second tape and copy to third tape.**
  - **erase fourth tape.**
  - **GoTo step (†).**

(c) $L = \{a^i b^j c^k \mid i \cdot j = k \text{ and } i, j, k \geq 0\}$ above $\Sigma = \{a, b, c\}$
Main idea: The machine will read the tape and for each 'a' that it will read, will erase a 'c' for each 'b' there is.
Pseudo-code:

- **Check if the input is in the form $a^*b^*c^*$. If not, reject.**
- **If the current letter is a blank - accept.**
- **If the current letter is a 'c' - reject.**
- **(†) If the current letter is a 'b' - check that all c's were erases. If so, accept. If not, reject**
- **Erase first 'a' and move head right**
  - **As long as current letter is 'a' - move right**
  - **If the current letter is a 'c' - reject.**
  - **($\star$) If the current letter is a 'b' - replace with 'x' and move right.**
    - **As long as current letter is 'b' or 'y'- move right**
    - **If the current letter is a blank - reject.**
    - **If the current letter is a 'c' - replace by 'y' and move left until the first letter to the right of 'x'.**
    - **If the current letter is a 'y' - reconstruct all b's be replacing each 'x' with 'b'. If not GoTo($\star$).**
  - **Go left to the first non-blank and GoTo (†)**

4. Let us define a generalization of Turing Machines to include a *finite memory* of size n. We denote such a Turing Machine formally as:

$$M_{\text{mem}} = (Q, \Sigma, \gamma, \delta_{\text{mem}}, n, q_0, qa, q_r),$$

where all the definitions are identical to the Turing Machine defined in class except that there is the *finite* memory size n and the transition function δ_{mem}. At each step, the transition depends on the current state, the input on the tape and all the memory. The transition to the next step can update the entire memory. Formally: $\delta_{\text{mem}} : Q \times \Gamma \times \Gamma^n \to Q \times \Gamma \times \Gamma^n \times \{L, R\}$.

Given a finite memory Turing Machine M_{mem}, define *formally* a (standard) Turing Machine M such that $L(M) = L(M_{\text{mem}})$. Namely, both Turing Machines accept the same language. Explain your answer.

Bonus: Show more than one formal construction.
Bonus: Give a formal proof.

Given a *finite memory* TM

$$M_{\mathbf{mem}} = (Q, \Sigma, \gamma, \delta_{\mathbf{mem}}, n, q_0, q_a, q_r).$$

We define a (standard) Turing Machine M as follows:
(a) Extend γ to account for the memory:

$$M = (Q, \Sigma, \gamma', \delta', , q_0, q_a, q_r)$$

Where: $\gamma' = \Gamma^{n+1}$ and
$\delta'(q, (\sigma_0 \dots \sigma_n)) = (r, (\tau_0 \dots \tau_n), D)$ **for every:**
$\delta(q, \sigma_0, \dots, \sigma_n)) = (r, \tau_0, \dots, \tau_n, D)$
(b) Similarly, one can extend the states to account for the memory.

5. 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.**

6. Let $L_1, L_2 \in RE \setminus R$ can the following occur? If your answer is "yes" give a concrete example. If your answer is "no" give a sketch of a proof.
   (a) $L_1 \cap L_2 \in R$
   Yes, take

   $$L_1 = \{< M, x >, \textbf{M is a TM that halts on x and has an even number of states}\}$$

   $$L_2 = \{< M, x >, \textbf{M is a TM that halts on x and has an odd number of states}\}$$

   It is easy to see that $L_1, L_2 \in RE \setminus R$ but their intersection is empty hence in R.
   (b) $L_1 \cup L_2 \in R$
   Yes, take

   $$L_1 = \{< M, x >, \textbf{M is a TM that halts on x or has an even number of states}\}$$

   $$L_2 = \{< M, x >, \textbf{M is a TM that halts on x or has an odd number of states}\}$$

   It is easy to see that $L_1, L_2 \in RE \setminus R$ but their union is all pairs $< M, x >$ hence in R.

(c) $L_1 \cap L_2 \in R$ and $L_1 \cup L_2 \in R$
No, then we could construct a TM that decides L_1: Let M_1, M_2 be the TM that accepts L_1 and L_2, respectively. Let M_u, M_i be the TM that decides $L_1 \cap L_2$ and $L_1 \cup L_2$, respectively.
The TM that decides L_1: If $M_i(x) = T$ return T.
If $M_u(x) = F$ return F.
If not, the run M_1, M_2 simultaneously on x until one accepts.
If $M_1(x)$ accepts return T.
If $M_2(x)$ accepts return F.

7. Prove or contradict: R is closed under infinite (countable) union. Namely, given $L_1, L_2 \ldots$ such that $L_i \in R$ then $\bigcup_{i=1}^{\infty} L_i \in R$.
The claim is not true. Let L be a language not in R (hence it has an infinite number of words), and enumerate the words in some order. For every i, let L_i be the language that contains only the i'th word of L. Clearly, every L_i is decidable as it is finite but $L = \bigcup_{i=1}^{\infty} L_i \notin R$.