Computational Models - Exercise #4 solution sketch

1. We shall describe $M = (Q, \Sigma = \{0,1\}, \Gamma = \Sigma \cup \{\sqcup, \$\}, \delta, q_0, q_a, q_r)$, where

$$Q = \{q_0, q_1, q_2, q_{r,0}, q_{r,1}, q_{add}, q_{l,0}, q_{l,1}, q_{l,\sqcup}, q_a, q_r\}.$$

   The overall idea: Reach the LSB and as long as the bit you see is 1, change it to 0 and move a step left. Once you see 0, change it to 1, reach the rightmost part of the number and accept. The only caveat is that in case the carry reaches the MSB, we need to move the whole number one cell right to "make room" for the new 1. We will use a model that allows "stay" steps, but obviously every stay step can be translated to left and right ones.

   The description of our states:

   - $q_{r,0}, q_{r,1}$ are for shifting the input right (and either writing 0 or 1 afterwards).
   - q_{add} is for the increment itself.
   - q_1 is for moving right before shifting the input.
   - $q_{l,0}, q_{l,1}, q_{l,\sqcup}$ are for shifting the input left.
   - q_2 is moving right once we finish.

   The δ function is given by:

   - $\delta(q_0, 0) = (q_{r,0}, \$, R)$, $\delta(q_0, 1) = (q_{r,1}, \$, R)$, $\delta(q_0, \sqcup) = (q_a, \sqcup, S)$.
   - $\delta(q_{r,0}, 0) = (q_{r,0}, 0, R)$, $\delta(q_{r,0}, 1) = (q_{r,1}, 0, R)$, $\delta(q_{r,0}, \sqcup) = (q_{add}, 0, S)$.
   - $\delta(q_{r,1}, 0) = (q_{r,0}, 1, R)$, $\delta(q_{r,1}, 1) = (q_{r,1}, 1, R)$, $\delta(q_{r,1}, \sqcup) = (q_{add}, 1, S)$.
   - $\delta(q_{add}, 0) = (q_1, 1, R)$, $\delta(q_{add}, 1) = (q_{add}, 0, L)$, $\delta(q_{add}, \$) = (q_2, 1, R)$.
   - $\delta(q_1, 0) = (q_1, 0, R)$, $\delta(q_1, 1) = (q_1, 1, R)$, $\delta(q_1, \sqcup) = (q_{l,\sqcup}, \sqcup, L)$.
   - $\delta(q_{l,\sqcup}, 0) = (q_{l,0}, \sqcup, L)$, $\delta(q_{l,\sqcup}, 1) = (q_{l,1}, \sqcup, L)$.
   - $\delta(q_{l,0}, 0) = (q_{l,0}, 0, L)$, $\delta(q_{l,0}, 1) = (q_{l,1}, 0, L)$, $\delta(q_{l,0}, \$) = (q_2, 0, R)$.
   - $\delta(q_{l,1}, 0) = (q_{l,0}, 1, L)$, $\delta(q_{l,1}, 1) = (q_{l,1}, 1, L)$, $\delta(q_{l,1}, \$) = (q_2, 1, R)$.
   - $\delta(q_2, 0) = (q_2, 0, R)$, $\delta(q_2, 1) = (q_2, 1, R)$, $\delta(q_2, \sqcup) = (q_a, \sqcup, S)$.

   We can make δ total in an arbitrary way.

2. We shall describe $M = (Q', \Sigma, \Gamma, \delta', q_0, q_a, q_r)$ in the two-tapes model with stay steps, where $Q' = Q \cup \{q_a, q_r\} \cup Q_{new}$ for Q_{new} soon to be defined, $\Gamma = \Sigma \cup \{\sqcup\}$ and the transition function is $\delta' : Q' \times \Gamma^2 \mapsto Q' \times \Gamma^2 \times \{L, R, S\}^2$. The second tape will simulate the stack such that the head of the first tape always moves right and we always keep the content of the first tape intact. The details are as follows:

- 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 operations which include no pops. The second head stays in place: $\delta'(q, \sigma_1, \sqcup) = (q', \sigma_1, \sigma_2', R, S)$ such that $\delta(q, \sigma_1, \varepsilon) = (q', \sigma_2')$. 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 $\delta(q, \sigma_1, \sigma_2) = (q', \varepsilon)$ 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 $\delta(q, \sigma_1, \sigma_2) = (q', \sigma_2')$ 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: $Q_{new} = \{(q, \sigma) \text{ s.t. } q \in Q \text{ and } \sigma \in \Sigma\}$ 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 $\delta(q, \sigma_1, \varepsilon) = (q', \sigma_2')$ add the following transition $\delta'(q, \sigma_1, \sigma_2) = ((q', \sigma_2'), \sigma_1, \sigma_2, R, R)$
- 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 $q \in F$ and $\delta'(q, \sqcup, \sqcup) = \delta'(q, \sqcup, \sigma) = (q_r, \sqcup, \sqcup, S, S)$ if $q \notin F$.

3. The models are equivalent. One direction is trivial. For the other direction, let M be a TM in the standard model. There are two key tricks. First, we can recopy the entire (non-blank) tape between transitions. Well, really we are almost only copying the tape, as we also need to record the configuration after one tape transition. This requires one extra tape symbol as a delimiter, and an expanded alphabet to record the position of the tape head as we copy. In this approach, two writes are required. The first write is used while copying to record the symbol over fresh tape, the second write is used to mark a symbol as having been copied.

   The second key observation is that we do not need to use the tape so efficiently. Instead of recording each symbol in one tape square, use two tape cells. The first of each pair is used to record a tape symbol, and the second is used to mark the symbol as having been copied. Each cell is only altered once. Fill in the details.

4. Let M_1 and M_2 be such that $L(M_1) = L_1$ and $L(M_2) = L_2$. Define the nondeterministic TM M that on input x does the following:

   - Denote $m = |x|$.
   - Guess an odd number $p \in \{1, \ldots, m-1\}$ (how?).
   - Guess p integers $1 \leq i_1 < \ldots < i_p \leq m-1$.
   - Denote $u_1, v_1, \ldots, u_{(p+1)/2}, v_{(p+1)/2}$ the partition of x into $p+1$ words according to the indices $i_1, \ldots, i_p$.
   - Set $s \leftarrow 0$.
   - Using controlled execution, run M_{L_1} on $u_1, \ldots, u_{(p+1)/2}$ and M_{L_2} on $v_1, \ldots, v_{(p+1)/2}$ (make sure you know how to accurately describe this step in greater details and why normal sequential simulation fails).

- If at any stage, either M_{L_1} or M_{L_2} accepted, set $s \leftarrow s+1$. If $s = p+1$, accept.

Complete the correctness details yourselves.

5. (a) The claim if false. Let

$$L_1 = \{\langle M, x\rangle : M \text{ is a TM that halts on } x \text{ and has an even number of states}\}$$

and

$$L_2 = \{\langle M, x\rangle : M \text{ is a TM that halts on } x \text{ and has an odd number of states}\}$$

It is easy to see that L_1,$L_2 \in \mathcal{RE} \setminus \mathcal{R}$ but their intersection is empty hence in $\mathcal{R}$.

(b) The claim is true. Let $L_1, L_2 \in \text{co}\mathcal{RE}$, so $\overline{L_1}, \overline{L_2} \in \mathcal{RE}$. It holds that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ and since $\mathcal{RE}$ is closed under union, $\overline{L_1} \cup \overline{L_2} \in \mathcal{RE}$ so $L_1 \cap L_2 \in \text{co}\mathcal{RE}$.

(c) The claim is false – simply take the halting problem and its complement.

(d) The claim is true. Let M be a TM that accepts some language $L \in \mathcal{RE}$ and let h be some homomorphism. A TM that accepts $h(L)$: On input x, start going through all strings w simultaneously, and if $h(w) = x$, start executing M on w (using controlled execution of course). Accept if any of the executions accepts. If $x \in h(L)$ then there exists a w such that $h(w) = x$ and $w \in L$ and we will eventually get to it and execute M to within the desired number of steps. Obviously, if $w \notin L$ we will never accept (in fact, we will never halt).

6. Define the *crossing sequence* at location i on input x, denoted $C_i(x)$, to be the sequence of states M is in when its head crosses the boundary between the i-th cell and the $(i+1)$-th cell, when the TM is run on input x.

Let $(q_1, \ldots, q_k)$ and $(p_1, \ldots, p_h)$ be two finite crossing sequences. It is possible to verify whether or note they are *compatible* with respect to an input symbol a. That is, $(q_1, \ldots, q_k)$ and $(p_1, \ldots, p_h)$ could be at the left boundary and at the right boundary of a tape square which initially contains the symbol a. We will not use this fact, but only the existence of compatibility, and denote $A_a(\vec{q})$ the set of sequences compatible with $\vec{q}$ with respect to an input symbol a.

In our case, when on every input x, $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ visits every cell at most k times, every crossing sequence is of length at most k. Denote $Q' = \bigcup_{0 \le \ell \le k} Q^\ell$ to be the set of ℓ-tuples of states for $\ell \le k$. Obviously, Q' is finite.

Now, define the NFA $N = (Q', \Sigma, \delta', q'_0, F)$ such that $q'_0 = (q_0)$, $F = \{(q_a)\}$ and $\delta'(\vec{q}, a) = A_a(\vec{q})$ (when does the set contain more than one element? when does it contain zero elements?).

The intuitive idea is that N puts together pieces of computations of M as it scans the input string. This is done by guessing successive crossing sequences. If N has guessed that $\vec{q}$ is the crossing sequence at a boundary, and a is the next input symbol then N can guess any valid crossing sequence that $\vec{q}$ is compatible with on input a.

To see that $L(M) = L(N)$, first consider $w \in L(M)$ and look at the crossing sequence generated by an accepting computation of M on w. Each crossing sequence is compatible

with the one at the next boundary, so N can guess the proper crossing sequence (among its other guesses) and accept so $w \in L(N)$.

Next, let $w \in L(N)$ and consider the crossing sequence $\vec{q} = (q_0, \ldots, q_n)$ of M corresponding to the states of N as N scans $w = (a_1, \ldots, a_n)$. For every $0 \leq i < n$, q_i is compatible with q_{i+1} on a_i, so we can construct an accepting computation of M on w by determining when the head switches directions. Fill in the inductive proof yourself.