Autumn 1997–98, Prof. Boneh
Solution Set
CS 154, Problem Set 4

4.1 (2.5 points)

Problem: **Consider a restricted class of TM's in which the transition function is defined as $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R\}$. That is, the tape head can only move right. What class of languages does this restricted class of TM's recognize? Give a rigorous proof for your answer.**

The restricted TM's recognize the class of **regular languages**. The point is that TMs and DFAs differ only in how they use the tape, and if the TM can't "back up" to look at what was written in the past, its tape is basically read only.

Step 1: Construct a DFA simulating a restricted TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$. The DFA is

$$M' = (Q, \Gamma, \delta', q_0, F')$$

where $\delta'(q, \sigma) = q'$ iff $\delta(q, \sigma) = (q', \sigma', R)$. Also, we must define $\delta'(q_r, \sigma) = q_r$ and $\delta'(q_a, \sigma) = q_a$, since the TM does not define transitions on q_r or q_a. $F' = \{q_a\}$ at first, but we expand it below.

Step 1a: Prove $w \in L(M) \to w \in L(M')$. M starts in state q_0 looking at w_1 (by definition) and, if $w \in L(M)$, ends up in q_a. We would like to say M' ends up in q_a also, but this is not true because the definitions are not symmetric: M' must end up in q_a *at the end of* w, but M can land there any time. If M stops before reading all of w, we're OK because M' just stays in q_a, eating up the rest of w. If M stops after reading all of w (it read some blanks too) we have a problem, since M could be in $q \neq q_a$ after reading w. We can get around this technicality by expanding the definition of F': for every $q \in Q$, figure out what states we go through while reading $|Q| + 1$ blank symbols. There are three possibilities: we end up in q_a (in which case we add q to F'), we end up in q_r, or we loop, seeing one state twice. If the last two happen, q is not an accept state. This workaround can be done at construction time. It depends on the fact that once we've read w, only the states matter since the tape looks the same from here on out.

With this workaround, we know that M' accepts w if δ and δ' behave the same on every state/letter pair. Then if M ends up in q_a, M' will either end up in q_a, or end up in some other state in F that leads to q_a for the TM. But if $\delta(q, w_i) = q'$, $\delta'(q, w_i) = q'$ too, by construction. Also, since the TM moves right, the TM looks at w_{i+1} next, as does the DFA. Thus, δ and δ' *do* behave the same. (The exceptions, $q = q_r$ or $q = q_a$, also behave appropriately.) Thus, $L(M) \subseteq L(M')$.

Step 1b: Prove $w \in L(M') \to w \in L(M)$. This is the same argument, in reverse. Both M and M' start with the same state/letter pair. If M' ends up in an accepting state on w, then since δ and δ' behave the same, M will also end up in one of those states. (This is true even for the "special" states: q_a and q_r are pits for both M and M'.) If M ends up in q_a we're done, if it's not then M' will read blanks from here on out, and by the definition of F' will eventually end up in q_a.

Step 2: Construct a restricted TM simulating a DFA $M = (Q, \Sigma, \delta, q_0, F)$. The TM is

$$M = (Q \cup \{q_r, q_a\}, \Sigma, \Sigma \cup \{\sqcup\}, \delta', q_0, q_a, q_r)$$

where $\delta'(q, \sigma) = (q', \sigma, R)$ if $\delta(q, \sigma) = q'$. We also define $\delta'(q, \sqcup) = (q_a, \sqcup, R)$ if $q \in F$, and $\delta'(q, \sqcup) = (q_r, \sqcup, R)$ if $q \notin F$. By definition, we don't have to define δ on the new states q_r and q_a.

Step 2a: Prove $w \in L(M) \leftrightarrow w \in L(M')$. (We do both directions at once.) Note that, as in the previous construction, both machines start in the same configuration. Again we prove that δ and δ' behave the same on a state/letter pair (for letters in Σ). By definition, both δ and δ'

transition to the same state on w_i, and since the TM moves right it reads w_{i+1} next, as does the DFA. The exception to the equivalent behavior is when M' reads a blank.

When M processes w it ends up in state q. Since w has no blanks, by the equivalence of δ and δ' M' is also in state q upon reading w. The next tape square after w is a blank, and by the definition of δ' the TM transitions into q_a if $q \in F$ and q_r if $q \notin F$. Thus M' accepts w iff M does.

4.2 (2.5 points)

PROBLEM: **Formally describe a Turing Machine that recognizes the language $L = \{\#x\#y : y = x + 1$ (represented in binary)$\}$.**

Intuitively, we copy x to tape 2, then go from right to left on both x and y. We have two main states, before we account for y's extra bit ("carry"), and after ("nocarry"). We treat ##1 as a legal string (ie $\epsilon = 0$) and allow leading 0s.

I use a 2-tape TM in which each head can move left, right, or stay still on a transition. We know how to convert this machine to a standard 1-tape TM. Here's M:

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_a, q_r\}, \{0, 1, \#\}, \{0, 1, \#, \sqcup\}, \delta, q_0, q_a, q_r).$$

In the transition table for δ I use $\cdot$ to refer to "any character in Γ not already referred to." On the right-hand side, it means "leave the tape square the same." Here's δ:

$(\cdot, \cdot, \#)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Should never see a # on tape 2
$(q_0, \#, \sqcup)$	$\rightarrow$	$(q_1, \#, \sqcup, R, R)$	Mark left edge of tape 2 — well of leading 0s
$(q_0, \cdot, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Ensure w starts with #
$(q_1, 0, \sqcup)$	$\rightarrow$	$(q_1, 0, 0, R, R)$	Copy x to tape 2
$(q_1, 1, \sqcup)$	$\rightarrow$	$(q_1, 1, 1, R, R)$	Ditto
$(q_1, \#, \sqcup)$	$\rightarrow$	$(q_2, \#, \sqcup, R, L)$	Done copying x; now start reading y
$(q_1, \cdot, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Tape 2 not blank, or no 2nd #
$(q_2, \sqcup, \cdot)$	$\rightarrow$	$(q_3, \sqcup, \cdot, L, N)$	We've reached the end of y
$(q_2, \#, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	There are more than 2 #s in the input
$(q_2, \cdot, \cdot)$	$\rightarrow$	$(q_2, \cdot, \cdot, R, N)$	Go to end of y
$(q_3, 1, 0)$	$\rightarrow$	$(q_4, 1, 0, L, L)$	Found our carry bit; no carry needed from now on
$(q_3, 1, \sqcup)$	$\rightarrow$	$(q_4, 1, \sqcup, L, N)$	$\sqcup$ is an infinite well of leading 0s
$(q_3, 0, 1)$	$\rightarrow$	$(q_3, 0, 0, L, L)$	Still looking for the carry bit
$(q_3, 1, 1)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Illegal if we need a carry
$(q_3, 0, 0)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Ditto
$(q_3, 0, \sqcup)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	$\sqcup$ behaves like 0
$(q_3, \sqcup, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Impossible
$(q_3, \#, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Finished y while seeking the carry bit
$(q_4, 1, 1)$	$\rightarrow$	$(q_4, 1, 1, L, L)$	OK after we've found the carry bit
$(q_4, 0, 0)$	$\rightarrow$	$(q_4, 0, 0, L, L)$	Ditto
$(q_4, 0, \sqcup)$	$\rightarrow$	$(q_4, 0, \sqcup, L, N)$	$\sqcup$ behaves like 0
$(q_4, 0, 1)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Bad after we're found the carry bit
$(q_4, 1, 0)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Ditto
$(q_4, 1, \sqcup)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Ditto again
$(q_4, \sqcup, \cdot)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Impossible
$(q_4, \#, 0)$	$\rightarrow$	$(q_4, \#, 0, N, L)$	Make sure only leading 0s are left on tape 2
$(q_4, \#, 1)$	$\rightarrow$	$(q_r, 0, 0, R, R)$	Finished y before x
$(q_4, \#, \sqcup)$	$\rightarrow$	$(q_a, 0, 0, R, R)$	Finished x and y!

4.3 (2.5 points)

PROBLEM: **Prove that if both L and $\overline{L} = \Sigma^* \setminus L$ are R.E. then L is recursive.**

By definition, there is a TM M accepting L and a TM $\overline{M}$ accepting $\overline{L}$. The idea is that we construct a TM that runs w on M and $\overline{M}$ in parallel. If $w \in L$, eventually M will halt and accept (by definition of a TM accepting a language); then we accept. If $w \in \overline{L}$, eventually $\overline{M}$ will halt and accept; then we reject. In both cases we eventually halt. This new machine clearly accepts L, and since it halts on all strings it decides L as well.

Formally, we let our new machine M' be a two-tape machine. First M' copies the input from tape 1 to tape 2 and positions both heads at the far left of their tapes. (I omit the code to do this.) The rest of M' looks like

$$M' = (Q \times \overline{Q} \cup \{q_a, q_r\}, \Sigma, \Gamma, \delta', (q_0, \overline{q_0}), q_a, q_r)$$

For each $\delta(q, \sigma) = (q', \sigma', d)$ and $\overline{\delta}(\overline{q}, \overline{\sigma}) = (\overline{q'}, \overline{\sigma'}, \overline{d})$ $(d = l, r, n)$, we add the following state to M':

$$\delta'((q, \overline{q}), \sigma, \overline{\sigma}) \rightarrow ((q', \overline{q'}), \sigma', \overline{\sigma'}, d, d')$$

We also need to halt. We accept if M accepts and reject if $\overline{M}$ accepts (transitions are for all σ, $\overline{\sigma}$, q, and $\overline{q}$):

$$\delta'((q_a, \overline{q}), \sigma, \overline{\sigma}) \rightarrow (q_a, \sigma, \overline{\sigma}, R, R)$$
$$\delta'((q, \overline{q_a}), \sigma, \overline{\sigma}) \rightarrow (q_r, \sigma, \overline{\sigma}, R, R)$$

It is clear from this definition that after each δ' transition, tape 1 holds what M's tape holds, tape 2 holds what $\overline{M}$'s tape holds, and the state is the pair of states M and $\overline{M}$ are in. Also, we start in the start states of M and $\overline{M}$; thus, we simulate.

The interesting thing is the halting conditions. We halt at (q_a, q) — by the above discussion, we know this is all states where M has accepted. Likewise, we halt at (q, q_a), which is all states in which $\overline{M}$ accepts. Since one machine accepts strings in L, and the other accepts strings not in L, one or the other machine accepts w, and in either case M' halts. Thus it's a decider. Since it accepts only if M accepts, and rejects only if $\overline{M}$ accepts, the language it decides is exactly L.

There's a problem with this above description that only two people on the problem set caught. Most people said just "Run M and $\overline{M}$ in parallel," but what happens if M runs off the left edge of the tape before $\overline{M}$ has a chance to accept? Then our new machine M' will hang. Bad News. The solution is to replace the input w by $\#w$ on each tape and add new rule $\delta'(q, \sigma, \#) \rightarrow (q_r, \#, \#, N, N)$, and likewise for $(q, \#, \sigma)$. Thanks to Munjal Shah and Bob Zhang for pointing this out — I missed it too.

4.4 (2.5 points)

PROBLEM: **Define a singleton-TM to be a two-tape TM where the finite control only has three states $Q = \{q_0, q_{acc}, q_{rej}\}$. Prove that every one-tape TM has an equivalent singleton-TM.**

The singleton-TM adds a letter for each state, and just stores the state on the second tape instead of in the finite control. To wit, if $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$,

$$M_s = (\{q_s, q_a, q_r\}, \Sigma, \Gamma \cup Q, \delta', q_s, q_a, q_r$$

δ' is defined as follows: for every rule $\delta(q, \sigma) \to (q', \sigma', d)$ we define $\delta'(q_s, \sigma, q) \to (q_s, \sigma', q', d, N)$ (where in the first case q and q' are states and in the second case they are letters in $\Gamma \cup Q$.

We must define the rest of δ'. If the first tape has a letter in Q, or the second a letter in Γ, we go to the reject state (this is actually impossible). We must do halting: $\delta'(q_s, \sigma, q_a) \to (q_a, \sigma, q_a, R, R)$ and $\delta'(q_s, \sigma, q_r) \to (q_r, \sigma, q_r, R, R)$. Finally, if there's a blank on the second tape we must initialize: $\delta'(q_s, \sigma, \sqcup) \to (q_s, \sigma, q_0, N, N)$.

It's usually impossible to "prove" that TMs act the way we say they do. (Otherwise, debugging computer programs would be much easier!) But we can give a plausibility argument. First, note the second head never moves, so the first thing that will happen is it will be set to q_0 and forever after will hold the state that M would be in. Clearly M_s does the same thing when tape 2 holds q as M does in state q. Furthermore, when M would get to q_a or q_r, M_s also goes to q_a or q_r. Thus, M_s emulates M.