CS3231
Tutorial 4

1. First method:

   Suppose L is regular language accepted by the DFA $D = (Q, \Sigma, \delta, q_0, F)$.

   Define, ϵ-NFA $N = (Q', \Sigma, \delta', S, F')$ as follows.

   $Q' = \{(q, q', q'') : q, q', q'' \in Q\} \cup \{S\}$, where S is a new starting state.

   $F' = \{(q, q'', q) : q'' \in F\}$

   $\delta'(S, \epsilon) = \{(q_0, q, q) : q \in Q\}$.

   For $a \in \Sigma$, $q', q'', q \in Q$, $\delta'((q', q'', q), a) = \{(\delta(q', a), \delta(q'', b), q) : b \in \Sigma\}$.

   Intuitively, in the above construction, the machine N being in state (q', q'', q) represents that we have guessed the middle state to be q and starting from q_0, after reading the input seen so far, the DFA D would have reached state q' and some string of the same length as the input seen so far would have taken the DFA D from "middle" state q to q''. The start state S is used only to transfer control to (q_0, q, q), where q is the guessed middle state.

   To show that above works, show by induction on length of w that, for all $w \in \Sigma^*$,

   $$\hat{\delta}'(S, w) = \{(\hat{\delta}(q_0, w), q'', q) : q \in Q \text{ and } (\exists u \in \Sigma^*)[|u| = |w|, \hat{\delta}(q, u) = q'']\}$$

   Thus, $\hat{\delta}'(S, w) \cap F' \neq \emptyset$, if and only if there exists a q such that $\hat{\delta}(q_0, w) = q$ and $(\exists u \in \Sigma^*)[|u| = |w| \text{ and } \hat{\delta}(q, u) \in F]$, that is $w \in HALF(L)$.

   Second method:

   Suppose L is regular language accepted by the DFA $D = (Q, \Sigma, \delta, q_0, F)$.

   Define, ϵ-NFA $N = (Q', \Sigma, \delta', S, F')$ as follows.

   $Q' = \{(q, q') : q, q' \in Q\} \cup \{S\}$, where S is a new starting state.

   $F' = \{(q, q) : q \in Q\}$

   $\delta'(S, \epsilon) = \{(q_0, q') : q' \in F\}$.

   For $a \in \Sigma$, $q, q' \in Q$, $\delta'((q, q'), a) = \{(\delta(q, a), q'') : \delta(q'', b) = q', b \in \Sigma\}$.

   Intuitively, in the above construction, the machine N being in state (q, q') represents that starting from q_0, after reading the input seen so far the DFA D would have reached state

q and some string of the same length as the input seen so far, would have taken the DFA D from state q' to a final state.

To show that above works, show by induction on length of w that, for all $w \in \Sigma^*$,

$$\hat{\delta}'(S, w) = \{(\hat{\delta}(q_0, w), q'') : (\exists u \in \Sigma^*)[|u| = |w|, \hat{\delta}(q'', u) \in F]\}$$

Thus, $\hat{\delta}'(S, w) \cap F' \neq \emptyset$, if and only if there exists a $q \in Q$ such that $\hat{\delta}(q_0, w) = q$ and $(\exists u \in \Sigma^*)[|u| = |w|$ and $\hat{\delta}(q, u) \in F]$, that is $w \in HALF(L)$.

2. $\{Q, \{a, b\}, \delta, S, \{S\})$, where $Q = \{A, B, S, T_1, T_2, T_3\}$, and

   $\delta(S, a) = \{T_1, T_2\}$,

   $\delta(T_1, b) = \{A\}$,

   $\delta(T_2, a) = \{B\}$,

   $\delta(A, b) = \{T_3, B\}$,

   $\delta(T_3, a) = \{A\}$

   $\delta(B, a) = \{S\}$

3. (b.1) For each production $A \to \alpha$ in G, have a production $A \to \alpha^R$ in G^R. Rest of the parameters (the set of terminals, non-terminals and the starting symbol) remain the same.

   Then we have that G^R is a left-linear grammar for L^R. This can be proved by induction on length of derivation of strings generated by G and G^R.

   (b.2) For each production $A \to \alpha$ in G, have a production $A \to \alpha^R$ in G^R. Rest of the parameters (the set of terminals, non-terminals and the starting symbol) remain the same.

   Then we have that G^R is a right-linear grammar for L^R. This can be proved by induction on length of strings generated by G and G^R.

   (c) Suppose L is regular. Then L^R is also regular. Thus there is a right-linear grammar G for L^R. Thus, there is a left-linear grammar G^R for L (by part (b.1)).

   Suppose G is a left-linear grammar for L. Then G^R is a right-linear grammar for L^R (by part (b.2)). Thus, by result done in class L^R is regular, and thus L is regular.

4. In the answers, starting symbols is S. Upper case letters denote non-terminals, and lower case letter denote terminals.

   (a) $S \to cAc$

   $A \to aAa|bAb|c$

   (b) $S \to aSB|\epsilon$

   $B \to b|bb|\epsilon$

   (c) $S \to aSbS|bSaS|\epsilon$.

5. Left to student as it depends on the grammar used.

   For the above grammar some possible derivations include:

   $S \Rightarrow aSbS \Rightarrow abS \Rightarrow abbSaS \Rightarrow abbaS \Rightarrow abbaaSbS \Rightarrow abbaabS \Rightarrow abbaab$

   $S \Rightarrow aSbS \Rightarrow abSaSbS \Rightarrow abbSaSaSbS \Rightarrow abbaSaSbS \Rightarrow abbaaSbS \Rightarrow abbaabS \Rightarrow abbaab$

6. The language consists of all strings of even length. The right-linear grammar for it is:

   $S \rightarrow aaS|abS|baS|bbS|\epsilon$.

7. (a) Consider the string $aababb$

   The following are two different left most derivations:

   $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabaBB \Rightarrow aababB \Rightarrow aababb$.

   $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabaBB \Rightarrow aababB \Rightarrow aababb$.

   The grammar generates strings with equal number of a and b (non-zero).

   (b) $S \Rightarrow TS|T$

   $T \Rightarrow aB|bA$

   $A \Rightarrow bAA|a$

   $B \Rightarrow aBB|b$

   Intuitively, T generates strings w with equal number of a's and b's, where no proper non-empty prefix of w has equal number of a's and b's.

   A generates strings w with number of a's being one more than b, where no proper non-empty prefix of w has the same property.

   B generates strings w with number of a's being one less than b, where no proper non-empty prefix of w has the same property.