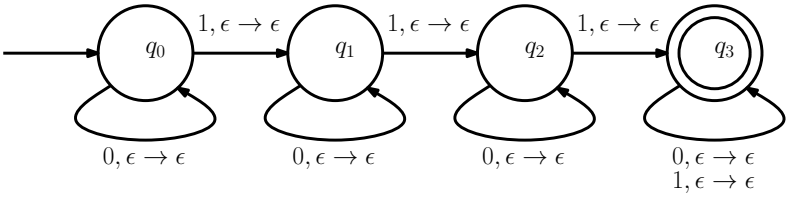
Computational Models, Spring 2016 Exercise #3

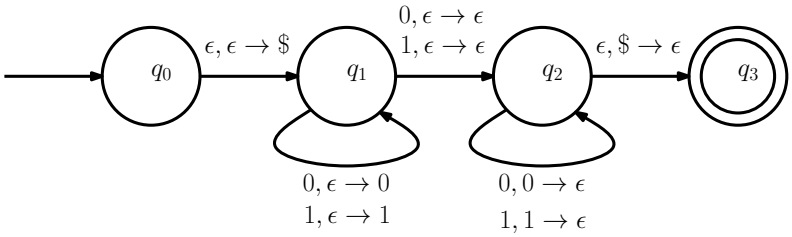
Context free grammars and Pushdown automata (and regular languages)

1. Give pushdown automata that recognize the following languages (only a drawing is required).

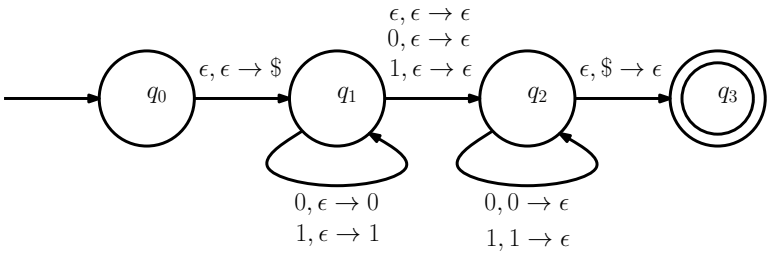
   (a) $L_1 = \{w \in \{0,1\}^* \ s.t.\ w$ contains at least three 1s$\}$



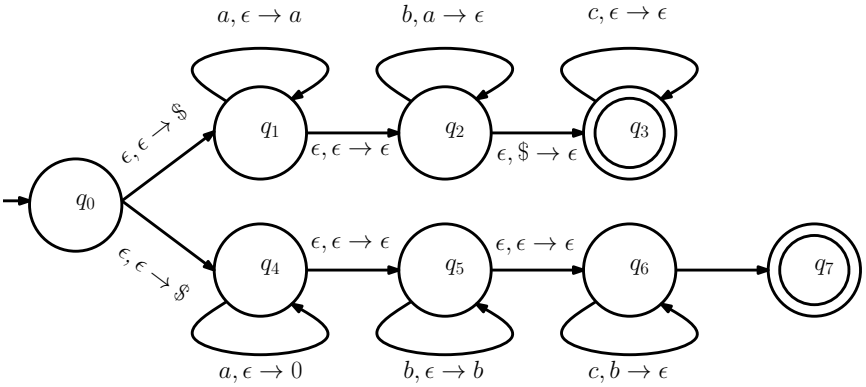
   (b) $L_2 = \{w \in \{0,1\}^* \ s.t.\ w = w^R$ and the length of w is odd$\}$



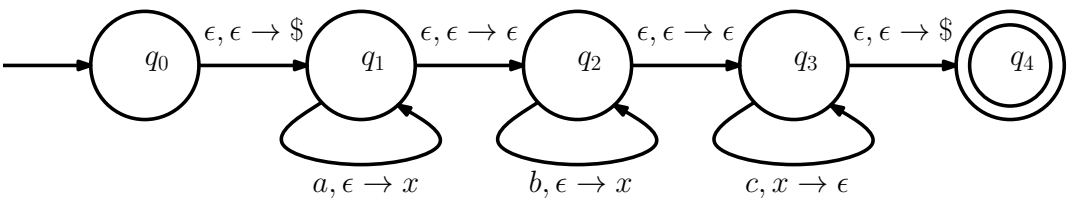
   (c) $L_3 = \{w \in \{0,1\}^* \ s.t.\ w = w^R\}$



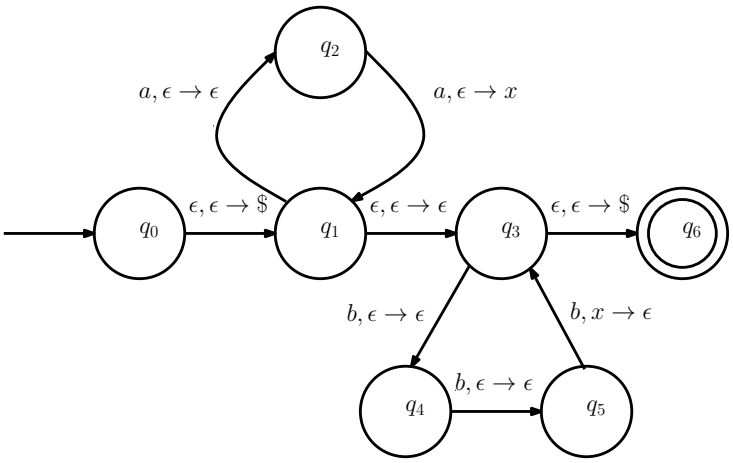
   (d) $L_4 = \{a^i b^j c^k \mid i, j, k \geq 0$ and $i = j$ or $j = k\}$



   (e) $L_5 = \{a^i b^j c^k \mid i, j, k \geq 0$ and $i + j = k\}$



(f) $L_6 = \{a^{2n}b^{3n} \mid n \geq 0\}$



2. For each of the following languages, give a context free grammars that derives the language (no need to prove your answer).

   (a) $L = \{a^i b^j c^i \mid 0 \leq i, j\}$
   $\mathbf{G} = (\{\mathbf{S}, \mathbf{T}\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \mathbf{R}, \mathbf{S})$ with R:
   $\mathbf{S} \to \mathbf{aSc}|\mathbf{T}$
   $\mathbf{T} \to \varepsilon|\mathbf{bT}$

   (b) $L = \{a^i b^j c^k \mid i + k = j\}$
   $\mathbf{G} = (\{\mathbf{S}, \mathbf{A}, \mathbf{C}\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \mathbf{R}, \mathbf{S})$ with R:
   $\mathbf{S} \to \mathbf{AC}$
   $\mathbf{A} \to \varepsilon|\mathbf{aAb}$
   $\mathbf{C} \to \varepsilon|\mathbf{bCc}$

   (c) $L = \{w \mid w \in \{0,1\}^*$ and w has twice as many 1's as 0's$\}$
   $\mathbf{G} = (\{\mathbf{S}\}, \{\mathbf{0}, \mathbf{1}\}, \mathbf{R}, \mathbf{S})$ with R:
   $\mathbf{S} \to \varepsilon|\mathbf{S1S1S0S}|\mathbf{S1S0S1S}|\mathbf{S0S1S1S}$

   (d) $L = \{a^i b^i \mid 0 \leq i\} \cup \{0\}^* \cup \{1\}^*$
   $\mathbf{G} = (\{\mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{\mathbf{a}, \mathbf{b}, \mathbf{0}, \mathbf{1}\}, \mathbf{R}, \mathbf{S})$ with R:
   $\mathbf{S} \to \mathbf{A}|\mathbf{B}|\mathbf{C}$
   $\mathbf{A} \to \varepsilon|\mathbf{aAb}$
   $\mathbf{B} \to \varepsilon|\mathbf{0B}$
   $\mathbf{C} \to \varepsilon|\mathbf{1C}$

   (e) $L = \{a^i b^j \mid i \neq j\}$
   $\mathbf{G} = (\{\mathbf{S}, \mathbf{A}, \mathbf{B}\}, \{\mathbf{a}, \mathbf{b}\}, \mathbf{R}, \mathbf{S})$ with R:
   $\mathbf{S} \to \mathbf{aSb}|\mathbf{aA}|\mathbf{Bb}$
   $\mathbf{A} \to \varepsilon|\mathbf{aA}$
   $\mathbf{B} \to \varepsilon|\mathbf{Bb}$

3. Prove that the following languages are not context free.

   (a) $L = \{a^n b^{n+1} c^{n+2} \mid n \geq 0\}$
   See solution to subsection c.

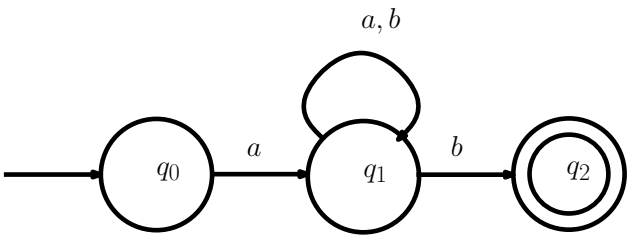
   (b) $L = \{a^n b^m c^n d^m \mid m, n \geq 0\}$
   Assume that language $\mathbf{L} = \{\mathbf{a^n b^m c^n d^m} \mid \mathbf{m, n \geq 0}\}$ is context-free. By the pumping lemma, there is a number p such that every string $\mathbf{w \in L}$ with length p or more can be written as $\mathbf{w = uvxyz}$ where (i) $\mathbf{|vxy| \leq p}$ (ii) $\mathbf{|vy| > 0}$ and (iii) $\mathbf{\forall i \geq 0}$, $\mathbf{uv^i xy^i z \in L}$. The

string $\mathbf{w = a^p b^p c^p d^p}$ must have a decomposition uvxyz that satisfies the preceding conditions. Consider the string $\mathbf{uv^2xy^2z}$ obtained by pumping w with $\mathbf{i = 2}$. Since $\mathbf{|vy| > 0}$ by condition (ii), we have that vxy contains at least one terminal. Without loss of generality, assume that vxy contains a terminal which is either a or c (similar argument for the case that the terminal is either b or d). Condition (i) implies that vxy is a string that cannot contain both a and c types of terminal. Thus, $\mathbf{uv^2xy^2z}$ increases the number of either a's or cs, but not the both, compared with w. Hence $\mathbf{uv^2xy^2z \notin L}$, a contradiction.

(c) $L = \{a^i b^j c^k \mid 0 < i < j < k < 2i\}$

Assume that language $\mathbf{L = \{a^i b^j c^k \mid 0 < i < j < k < 2i\}}$ is context-free. By the pumping lemma, there is a number p such that every string $\mathbf{w \in L}$ with length p or more can be written as $\mathbf{w = uvxyz}$ where (i) $\mathbf{|vxy| \leq p}$ (ii) $\mathbf{|vy| > 0}$ and (iii) $\mathbf{\forall i \geq 0,\ uv^i xy^i z \in L}$. Without loss of generality, assume that $\mathbf{p \geq 3}$. The string $\mathbf{w = a^p b^{p+1} c^{p+2}}$ must have a decomposition uvxyz that satisfies the preceding conditions. Consider the string $\mathbf{uv^p xy^p z}$ obtained by pumping w with $\mathbf{i = p}$. Since $\mathbf{|w| \leq p}$ by condition (i), we have that vxy contains only one type of terminal or the concatenation of either a and b types, or b and c types. If c is not contained in vxy, pumping v and y only increases the number of a's or b$'s$. Thus, the new string cannot keep the number of a's less than the number of b's which is less than the number of c's, i.e. $\mathbf{p + 2}$. If $\mathbf{c \in vxy}$, then $\mathbf{a \notin vxy}$. Thus, $\mathbf{uv^p xy^p z}$ would have at least $\mathbf{(p + 2) + (p - 1) = 2p + 1}$ number of c's while keeping the number of a's the same, i.e. k. Hence $\mathbf{uv^p xy^p z \notin L}$.

4. Consider the following NFA:



(a) Describe (in words) the language accepted by the NFA and give a regular expression for the language.

This NFA accepts all strings over $\{\mathbf{a, b}\}^*$ that begin with a and end with b. This can be described by the regular expression $\mathbf{a(a + b)^* b}$

(b) Prove your answer formally.
Hint: In your proof, use induction on the length of the input. Be sure to state your induction hypothesis explicitly. Show (inductively) that the following three conditions hold for any w:

i. $q_0 \in \hat{\delta}(q_0, w) \leftrightarrow w = \epsilon$

ii. $q_1 \in \hat{\delta}(q_0, w) \leftrightarrow w$ begins with a

iii. $q_2 \in \hat{\delta}(q_0, w) \leftrightarrow w$ begins with a and ends with b

We will complete this proof using mutual induction, where a set of conditions is created to describe the action of the machine. This set of conditions (i)-(iii) will demonstrate that the machine behaves as described above. Our task is to verify that these conditions hold for all input and that the appropriate input leads to acceptance.

BASE: For $\mathbf{|w| = 0}$, we know the only string with magnitude 0 is ϵ, so $\mathbf{w = \epsilon}$.
(i) Because $\mathbf{q_0}$ is the initial state of the machine, and there are no ϵ-transitions from $\mathbf{q_0}$, then $\mathbf{q_0 \in \hat{\delta}(q_0, w)}$ so the left hand side holds. Since $\mathbf{w = \epsilon}$, the right hand side holds as well and the biconditional is true.
(ii) Since there are no ϵ-transitions from the initial state to $\mathbf{q_1}$, it follows that

$q_1 \notin \hat{\delta}(q_0, w)$ and the left hand side is false. It is clear that w does not start with a and so the right hand side is false as well. Therefore the biconditional is true. (iii) By the same reasoning as (ii), it follows that condition (iii) is also true.

INDUCTIVE HYPOTHESIS: Assume that for $|w| = n$, all three conditions hold.

INDUCTIVE STEP: We now prove that the three conditions hold for $|w| = n + 1$. Let $w = xz$ where $|x| = n$ and $z \in \Sigma$. Thus $|w| = |x| + 1 = n + 1$.

(i) There are no transitions into q_0, so for any w such that $|w| \geq 1$, $q_o \notin \hat{\delta}(q_0, , w)$ and the left hand side is false. Since clearly $w \neq \epsilon$, the right hand side is also false and the biconditional is true.

(ii) Consider the prefix string x. By the inductive hypothesis, all the conditions hold for x, which means that condition (ii) holds for x. This means there are two cases to consider. The first, x starts with a, means that $q_1 \in \hat{\delta}(q_0, x)$. Since there is a transition from q_1 to q_1 on both a or b, it does not matter what symbol z is, it follows that $q_1 \in \hat{\delta}(q_0, w)$. In this case, no matter what z is, x will still start with a and thus w will start with a. Thus the biconditional will always be true. The second case, x starts with b, means that $q_1 \notin \hat{\delta}(q_0, x)$. Further, because there is no transition out of q_0 on a b, the machine is "stuck". Thus is does not matter what symbol z is, it will always be the case that $q_1 \notin \hat{\delta}(q_0, w$. Since x starts with b it follows that no matter what z is, w will also start with b. Thus both sides of the biconditional will always be false and the biconditional itself will always be true.

(iii) The only transition into q_2 is on a b from q_1. So any case which does not ensure $q_1 \in \hat{\delta}(q_0, x$ implies that $q_2 \notin \hat{\delta}(q_0, w$ and from condition (ii) we know that if x does not start with a then $q_1 \notin \hat{\delta}(q_0, w$ and it follows that $q_2 \notin \hat{\delta}(q_0, w)$. Thus in this situation, the biconditional will always be true. Now consider the situation where $q_1 \in \hat{\delta}(q_0, x)$. By the inductive hypothesis and (ii), we know this implies that x starts with a. Here we have two cases: $z = a$ and $z = b$. In the first case, $z = a$, it follows that w ends with a, and not b. Since there is no transition from a_1 to q_2 on an a, then we can conclude that $q_2 \notin \hat{\delta}(q_0, w)$. Likewise, if $q_2 \notin \hat{\delta}(q_0, w)$ then we can surmise that the transition from q_1 to q_2 on a b was not taken and therefore, w does not end in b. Thus the biconditional is true in both directions. In the second case, $z = b$, it follows that w ends in b. Since we know $q_1 \notin \hat{\delta}(q_0, x)$ and there is a transition from q_1 to q_2 on b, it follows that $q_2 \in \hat{\delta}(q_0, w)$. Likewise, if $q_2 \in \hat{\delta}(q_0, w)$, since the only transition into q_2 is from q_1 on a b, then it follows that w ends with b. Again the biconditional is true in both directions.

5. Consider a *minimal* DFA A that works on an alphabet Σ and define $L = L(A)$. Assume that you are told that it has m states, but you know nothing more about A it is a "black box" and your only way of getting information about A is to feed in words and observe whether they are accepted.

   We will construct an algorithm for determining the transition diagram of A from (any finite number of) such observations.

   One extremely expensive method would be to first construct all possible minimal DFAs of size m, (a HUGE number of DFAs this would give you!) then start testing all words from Σ^* in alphabetical enumeration, weeding out all DFAs that on some word behave different from your black box. Then at some point only one of your DFAs is left, i.e., problem solved. Don't do it this way, but reconstruct the equivalence relation $\sim_L$ from the Myhill-Nerode theorem—that gives a much faster reconstruction.

   (a) First, explain (in words, no need to formally prove) that in order to check whether for any words u, v it holds that $u \sim_L v$ we need to test words of length at most m (how many such words are there?).

   We know that $\sim_L$ has m classes. First observe that in order to check whether for any words u, v it holds that $u \sim_L v$, one only has to check whether $uw \in L(A) \leftrightarrow vw \in L(A)$ for all words w of length $\leq m$ (two words u, v are not equivalent iff they can be distinguished by a word w of length $\leq m$, that is, $uw \in L(A)$ but $vw \notin L(A)$ or vice versa). So it is possible to determine whether $u \sim_L v$ by at most Σ^m many checks.

(b) Now, explain how to choose m representative words, one for each equivalence class in $\sim_L$.

We procure representatives $\mathbf{r_1}, \ldots, \mathbf{r_m}$ of the $\sim_\mathbf{L}$-equivalence classes, such that we may write them as $[\mathbf{r_1}], \ldots, [\mathbf{r_m}]$. We may assume that these representatives have length at most $\mathbf{m}$. Choose $\mathbf{r_1}, \ldots, \mathbf{r_m}$ among all words $\mathbf{u}, \mathbf{v}$ of length $\leq \mathbf{m}$ such that the chosen $\mathbf{r_i}$ are pairwise not equivalent. Without loss of generality choose $\mathbf{r_1} = \epsilon$.

(c) Finally, to reconstruct A identify each word chosen in the previous subsection with a different state of A and explain how to reconstruct the transition function.

Identify the states of the DFA you are about to reconstruct with $[\mathbf{r_1}], \ldots, [\mathbf{r_m}]$. Assign $[\mathbf{r_1}] = [\epsilon]$ to be the starting state, according to the construction in the proof of the Myhill-Nerode proposition, and choose as accepting states all $[\mathbf{r_i}]$ where $\mathbf{r_i} \in \mathbf{L(A)}$. Finish your construction by putting for $\mathbf{i} = \mathbf{1}, \ldots, \mathbf{m}, \mathbf{a} \in \mathbf{\Sigma} : \delta([\mathbf{r_i}], \mathbf{a}) = [\mathbf{r_i a}] = [\mathbf{r_j}]$, where you use the method to decide $\mathbf{u} \sim_\mathbf{L} \mathbf{v}$ to find out to which representative $\mathbf{r_j}$ the word $\mathbf{r_i a}$ is equivalent.