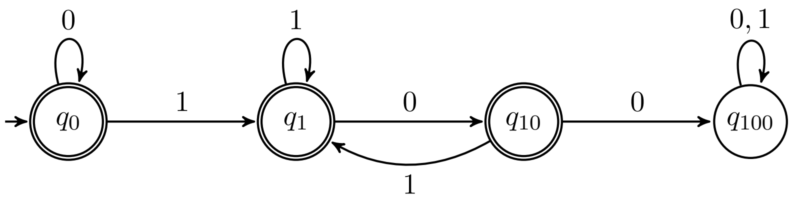
Suggested Solutions to Midterm Problems

1. Draw the state diagram of a DFA, with as few states as possible, that recognizes the language $\{w \in \{0,1\}^* \mid w \text{ doesn't contain the substring } 100\}$.

   Solution.

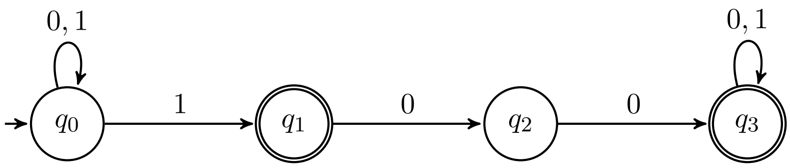


□

2. Let $L = \{w \in \{0,1\}^* \mid w \text{ contains } 100 \text{ as a substring or ends with a } 1\}$.

   (a) Draw the state diagram of an NFA, with as few states as possible, that recognizes L. The fewer states your NFA has, the more points you will be credited for this problem.

   Solution.



□

   (b) Give a regular expression that describes L. The shorter your regular expression is, the more points you will be credited for this problem.

   Solution. $(0 \cup 1)^*1(\epsilon \cup 00(0 \cup 1)^*)$ or $\Sigma^*1(\epsilon \cup 00\Sigma^*)$, where Σ is a shorthand for $(0 \cup 1)$. □

3. For languages A and B, let the *shuffle* of A and B be the language $\{w \mid w = a_1b_1 \cdots a_kb_k$, where $a_1 \cdots a_k \in A$ and $b_1 \cdots b_k \in B$, each $a_i, b_i \in \Sigma^*\}$. Show that the class of regular languages is closed under shuffle.

   Solution. Let $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two DFAs that recognize A and B, respectively. An NFA $M = (Q, \Sigma, \delta, q_0, F)$ that, in each step, simulates either a step of M_A or M_B will recognize the shuffle of A and B. Formally, it is defined as follows:

   - $Q = Q_A \times Q_B$,
   - $\delta((x,y),a) = \{(\delta_A(x,a),y),(x,\delta_B(y,a))\}$ for every $x \in Q_A, y \in Q_B, a \in \Sigma$,
   - $q_0 = (q_A, q_B)$,
   - $F = F_A \times F_B$.

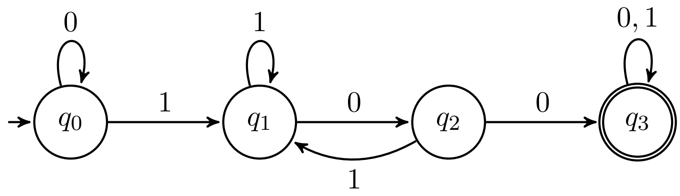
□

4. Given a language $L \subseteq \Sigma^*$, an equivalence relation R_L over Σ^* is defined follows:

$$xR_Ly \text{ iff } \forall z \in \Sigma^*(xz \in L \leftrightarrow yz \in L).$$

Suppose $L = \{w \in \{0,1\}^* \mid w \text{ contains the substring } \texttt{100}\}$. What are the equivalence classes determined by R_L? Please give an intuitive verbal description for each of the equivalence classes.

Solution. Applying Myhill-Nerode Theorem, we may discover the equivalence classes by examining a minimal DFA that recognizes L as below.



So, there are four equivalence classes corresponding to the four states:

(a) The subset of $\{0,1\}^*$ containing ε, 0, and all strings ending with 00 but without 100 as a substring.

(b) The subset containing all strings ending with 1 but without 100 as a substring.

(c) The subset containing all strings ending with 10 but without 100 as a substring.

(d) The subset containing all strings with 100 as a substring.

□

5. An *all*-NFA M is a 5-tuple $(Q, \Sigma, \delta, q, F)$ that accepts $x \in \Sigma^*$ if every possible state that M could be after reading input x is a state from F. Note, in contrast, that an ordinary NFA accepts a string if *some* state among these possible states is an accept state. Please give a formal definition of this computation model, as we did in class for an NFA, including a formal definition of the computation of an all-NFA on some input word.

Solution. We offer two different formal definitions for an all-NFA, one with ε-transitions (like for an NFA given in class) and the other without but with multiple start/initial states.

An *all*-NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(a) Q is a finite set of states,

(b) Σ is a finite alphabet,

(c) $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,

(d) $q_0 \in Q$ is the start state, and

(e) $F \subseteq Q$ is the set of accept states.

A *run* of an all-NFA on a word w, seen as $y_1y_2 \ldots y_m$ with $y_i \in \Sigma_\varepsilon$, is a sequence of states $r_0, r_1, \ldots, r_m$ such that $r_0 = q_0$ and $\delta(r_i, y_{i+1}) = r_{i+1}$ for $i = 0, 1, \ldots, m-1$. The run is *accepting* if $r_m \in F$. An all-NFA M *accepts* a word w if M has at least one run on w and every run is accepting.

Alternatively, an *all*-NFA is a 5-tuple $(Q, \Sigma, \delta, Q_0, F)$, where

(a) Q is a finite set of states,

(b) Σ is a finite alphabet,

(c) $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$ is the transition function,

(d) $Q_0 \subseteq Q$ is the set of start states, and

(e) $F \subseteq Q$ is the set of accept states.

To facilitate the formal definition of computation of such an all-NFA, we first extend the transition δ to sets of states such that $\delta(Q', a) = \bigcup_{q \in Q'} \delta(q, a)$, for $Q' \subseteq Q$ and $a \in \Sigma$. A *run* of an all-NFA on a word $w = w_1 w_2 \dots w_n$ with $w_i \in \Sigma$, is a sequence of sets of states $R_0, R_1, \dots, R_n$ such that $R_0 = Q_0$, $\delta(R_i, w_{i+1}) = R_{i+1}$, and, for every $q \in R_i$, there is some $q' \in R_{i+1}$ s.t. $q' \in \delta(q, w_{i+1})$, for $i = 0, 1, \dots, n-1$. The run is *accepting* if $R_n \subseteq F$. An all-NFA M *accepts* a word w if M has an accepting run on w. □

6. Consider the following CFG discussed in class, where for convenience the variables have been renamed with single letters.

$$\begin{array}{rcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T \times F \mid F \\ F & \rightarrow & (E) \mid a \end{array}$$

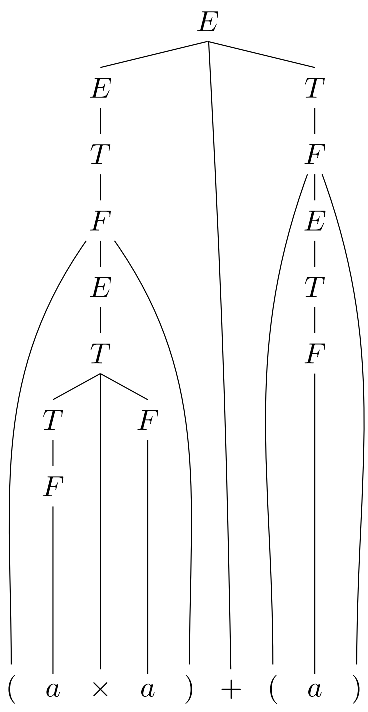
(a) (10 points) Give the (leftmost) derivation and parse tree for the string $(a \times a) + (a)$.

Solution.

The leftmost derivation

$$\begin{array}{rcl} E & \Rightarrow & E + T \\ & \Rightarrow & T + T \\ & \Rightarrow & F + T \\ & \Rightarrow & (E) + T \\ & \Rightarrow & (T) + T \\ & \Rightarrow & (T \times F) + T \\ & \Rightarrow & (F \times F) + T \\ & \Rightarrow & (a \times F) + T \\ & \Rightarrow & (a \times a) + T \\ & \Rightarrow & (a \times a) + F \\ & \Rightarrow & (a \times a) + (E) \\ & \Rightarrow & (a \times a) + (T) \\ & \Rightarrow & (a \times a) + (F) \\ & \Rightarrow & (a \times a) + (a) \end{array}$$

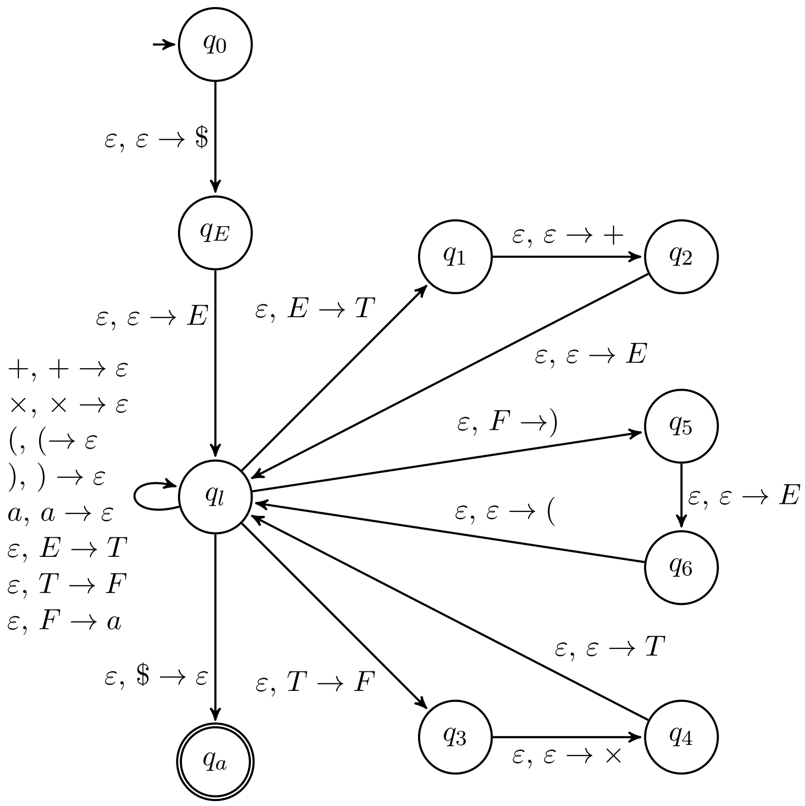
The parse tree



□

(b) (10 points) Convert the grammar into an equivalent PDA (that recognize the same language).

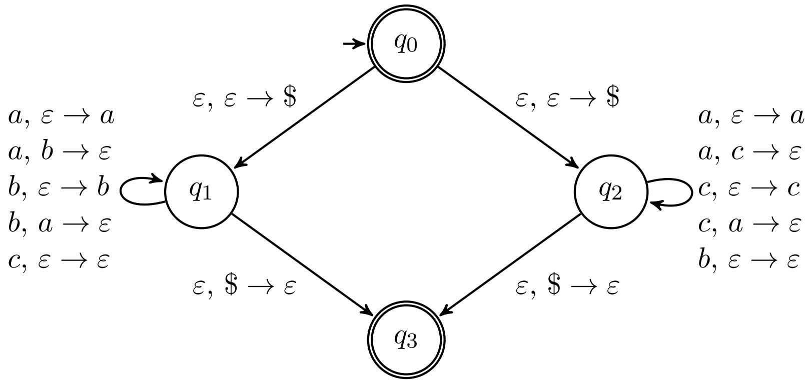
Solution.



□

7. Draw the state diagram of a pushdown automaton (PDA) that recognizes the following language: $\{w \in \{a,b,c\}^* \mid \text{ the number of } a\text{'s in } w \text{ equals that of } b\text{'s or } c\text{'s}\}$ (no restriction is imposed on the order in which the symbols may appear). Please make the PDA as simple as possible and explain the intuition behind the PDA.

   Solution. A PDA that recognizes the language is shown below. From the intial state, the PDA nondeterministically chooses to check whether the number of a's equals to that of b's (by transiting to q_1) or c's (to q_2). It accepts the input if one of the two checks passes. Take state q_1 for example. State q_1 reacts only to characters a and b, ignoring every c seen. As the input symbols come in no specific order, the number of a's may exceed that of b's at any point and vice versa. In the first case, it pushes an a onto the stack if the next symbol is an a and pops an a out of the stack if the next symbol is a b; analogously in the second case.



□

8. Prove, using the pumping lemma, that $\{a^m b^n c^{m \times n} \mid m, n \geq 1\}$ is not context free.

   Solution. Assume toward a contradiction that p is the pumping length for $\{a^m b^n c^{m \times n} \mid m, n \geq 1\}$, referred to as language A below. Consider a string $s = a^p b^p c^{p^2}$ in A. The string s may be divided as $uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$ in several different ways. We argue below, for each division case, $uv^i xy^i z \notin A$ for some $i \geq 0$ and conclude that s cannot be pumped, leading to a contradiction.

   - Case 1: v and y contain only a's, only b's, or only c's. Let us consider the first case; the other two are similar. In the first case, when i either goes up or down, $uv^i xy^i z$ will have a mismatch between the number of c's (which remains p^2) and the product of the number of a's (which is less or more than p) and that of b's (which remains p).
   - Case 2: v contains only a's and y contains only b's. This is similar to Case 1.
   - Case 3: v contains only b's and y contains only c's. Suppose s is divided as $a^p b^j \cdot b^k \cdot b^{(p-j-k)} c^l \cdot c^m \cdot c^{(p^2-l-m)}$ with $0 \leq k$, $0 \leq m$, and $0 < k+m \leq p$. We need to show that $a^p b^j \cdot (b^k)^i \cdot b^{(p-j-k)} c^l \cdot (c^m)^i \cdot c^{(p^2-l-m)} \notin A$, for some i, i.e., $p \times (j + k \times i + p - j - k) \neq l + m \times i + p^2 - l - m$ or $p \times k \times (i-1) \neq m \times (i-1)$, for some i. The inequality holds when $i = 0$ or 2.
   - Other cases: v contains some a's and some b's or some b's and some c's, or y contains some a's and some b's or some b's and some c's. In these cases, when i goes up, $uv^i xy^i z$ will not even be in the form of $a^* b^* c^*$.

   □

9. For languages A and B over Σ, let the *perfect shuffle* of A and B be the language $\{w \mid w = a_1 b_1 \cdots a_k b_k$, where $a_1 \cdots a_k \in A$ and $b_1 \cdots b_k \in B$, each $a_i, b_i \in \Sigma\}$. Show that the class of context-free languages is *not* closed under perfect shuffle.

   Solution. Let A be the language $\{0^{2i} 1^i \mid i \geq 1\}$ and B be $\{0^i 1^{2i} \mid i \geq 1\}$. Both are clearly context free. Their perfect shuffle equals $\{(00)^i (01)^i (11)^i \mid i \geq 1\}$, which is not context free. (Note: a string in the perfect shuffle must be the result of shuffling two strings of the *same* length.) □

Appendix

- (Pumping Lemma for Context-Free Languages)

  If A is a context-free language, then there is a number p such that, if s is a string in A and $|s| \geq p$, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions:

  1. for each $i \geq 0$, $uv^i xy^i z \in A$,
  2. $|vy| > 0$, and
  3. $|vxy| \leq p$.