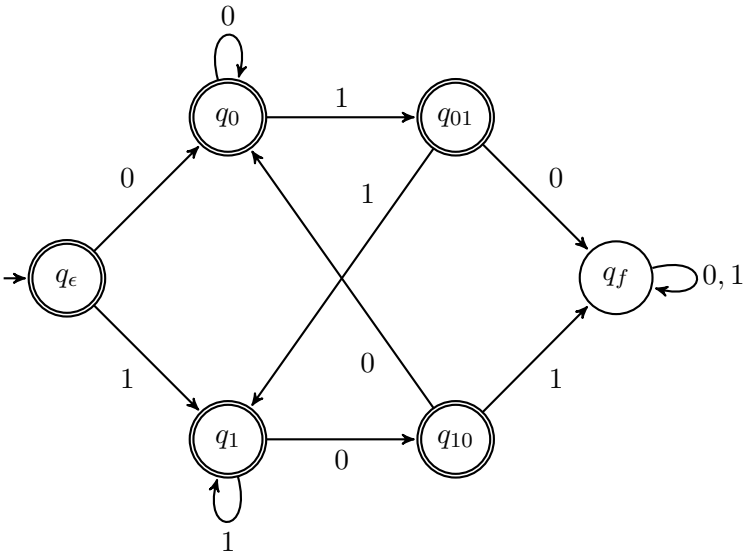
Suggested Solutions to Midterm Problems

1. Draw the state diagram of a DFA, with as few states as possible, that recognizes the language $\{w \in \{0,1\}^* \mid w \text{ doesn't contain 101 or 010 as a substring}\}$.

   Solution.

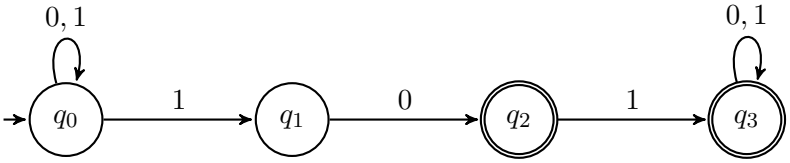


   □

2. Let $L = \{w \in \{0,1\}^* \mid w \text{ contains 101 as a substring or ends with 10}\}$.

   (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.

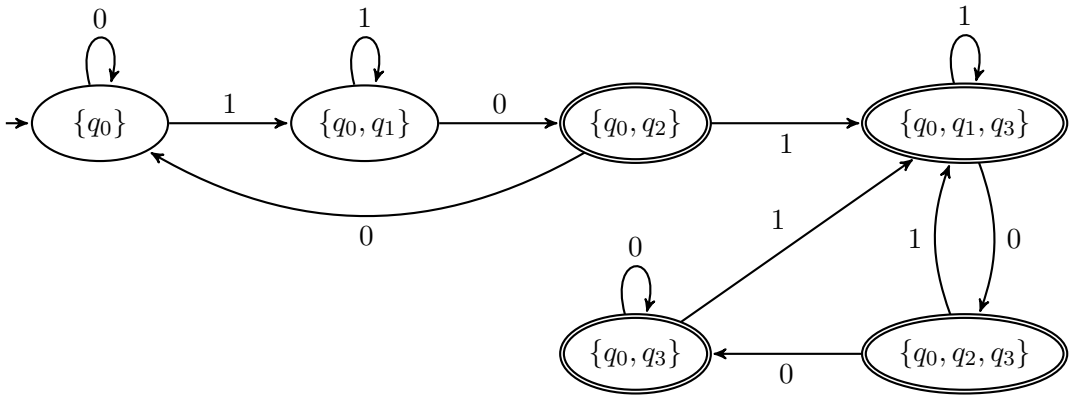
   Solution.



   □

   (b) Convert the preceding NFA systematically into an equivalent DFA (using the procedure discussed in class). Do not attempt to optimize the number of states, though you may omit the unreachable states.

   Solution.



□

3. Is the language $\{a^n b^{(n \mod K)} \mid n > 0$ and K is a positive integral constant$\}$ regular? Please justify your answer.

   Solution. The language, for every given integer $K \geq 1$, is regular. There are several ways to show this. We give a regular expression to describe the language, considering two cases $K = 1$ and $K > 1$ separately.

   - $K = 1$: In this case, $(n \mod K)$ is always 0. So, the language is the set of strings of any positive number of a's, which may be described by the regular expression a^+ (i.e., aa^*).
   - $K > 1$: The language may be viewed as the union of $\{a^{iK} b^0 \mid i > 0\}$ (i.e., $\{(a^K)^i \mid i > 0\}$) and $\{a^{iK+j} b^j \mid i \geq 0$ and $1 \leq j \leq K-1\}$ (i.e., $\{(a^K)^i a^j b^j \mid i \geq 0$ and $1 \leq j \leq K-1\}$). Hence, it may be described by the regular expression $(a^K)^+ \mid (a^K)^*(ab \mid aabb \mid \cdots \mid a^{(K-1)} b^{(K-1)})$.

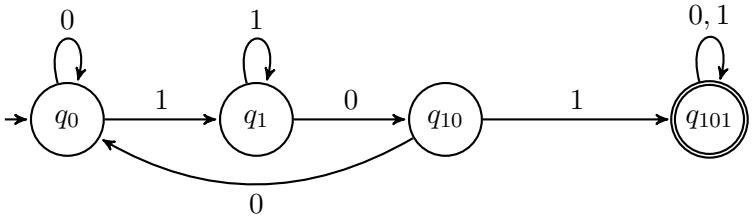
□

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

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

   Suppose $L = \{w \in \{0,1\}^* \mid w$ contains the substring `101`$\}$. 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:

(1) The subset of $\{0,1\}^*$ containing ε and all strings ending with 0 but without 101 as a substring.

(2) The subset containing all strings ending with 1 but without 101 as a substring.

(3) The subset containing all strings ending with 10 but without 101 as a substring.

(4) The subset containing all strings with 101 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 how an all-NFA accepts an 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 $y_1 y_2 \ldots y_m$ 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 $r_{i+1} \in \delta(r_i, y_{i+1})$ 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 $w = w_1 w_2 \ldots w_n$ with $w_i \in \Sigma$, is a sequence of sets of states $R_0, R_1, \ldots, R_n$ 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 $i = 0, 1, \ldots, n-1$. 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. Give a context-free grammar that generates the following language: $\{w \in \{a, b, c\}^* \mid$ the number of a's in w equals that of b's or c's$\}$ (no restriction is imposed on the order in which the input symbols may appear). Please make the CFG as simple as possible and explain the intuition behind it.

Solution. There are several plausible ways of defining the CFG. Below is a more intuitive version:

$$\begin{array}{lcl} S & \rightarrow & T \mid U \\ T & \rightarrow & abT \mid aTb \mid Tab \mid baT \mid bTa \mid Tba \mid TT \mid c \mid \varepsilon \\ U & \rightarrow & acU \mid aUc \mid Uac \mid caU \mid cUa \mid Uca \mid UU \mid b \mid \varepsilon \end{array}$$

It is equivalent to the following much simpler version:

$$\begin{array}{lcl} S & \rightarrow & T \mid U \\ T & \rightarrow & aTb \mid bTa \mid TT \mid c \mid \varepsilon \\ U & \rightarrow & aUc \mid cUa \mid UU \mid b \mid \varepsilon \end{array}$$

The production rules for T (analogous for U) can be seen as derived from those for generating strings of balanced parentheses, the main difference being that T allows pairs of parentheses to appear in reversed order. T may be forced to appear in a particular position of a string (of terminals and non-terminals) during a derivation, which means c can be placed in any desired position of a generated string. □

7. Prove that, if C is a context-free language and R a regular language, then $C \cap R$ is context free. (Hint: combine the finite control part of a PDA and that of an NFA.)

   Solution. The proof is by construction. Given a PDA P for language C and a DFA A for language R, we construct a PDA P' that recognizes $C \cap R$ by simulating P and A in parallel.

   Formally, let

   $$P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, F_P)$$

   be the PDA that recognizes C, and let

   $$A = (Q_A, \Sigma, \delta_A, q_A, F_A)$$

   be the DFA that recognizes R. Construct PDA

   $$P' = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q_P, q_A), F_P \times F_A)$$

   where, for $p \in Q_P$, $q \in Q_A$, $a \in \Sigma_\epsilon$, and $b \in \Gamma_\epsilon$, $\delta((p,q), a, b)$ is defined to be the set of all pairs $((r,s), c)$ such that:

   (1) $(r, c) \in \delta_P(p, a, b)$,
   (2) $s = \delta_A(q, a)$ when $a \in \Sigma$, and
   (3) $s = q$ when $a = \epsilon$.

   □

8. Prove *by induction* that, if G is a CFG in Chomsky normal form, then for any string $w \in L(G)$ of length $n \geq 1$, exactly $2n - 1$ steps are required for any derivation of w.

   Solution. The proposition still holds even if we include all other strings not in $L(G)$ that can be derived from non-start symbols. We will prove this stronger variant by induction on n, the length of an arbitrary nonempty string w. The strengthening in fact will make

the inductive proof easier, as we will have a stronger induction hypothesis for the inductive step.

Base case ($|w| = 1$): The only way to produce a string of length 1 is by applying at the beginning a rule of the form $A \to a$, which constitutes a one-step derivation.

Inductive step ($|w| = n > 1$): To produce a string of length larger than one, one must first apply a rule of the form $A \to BC$, where B and C are non-start symbols. Suppose the B part eventually produces a string x of length l and the C part a string y of length m such that $xy = w$ and $l + m = n$. From the induction hypothesis, these two parts of derivation take $2l - 1$ and $2m - 1$ steps, respectively. So, the derivation of a string of length n requires $1 + (2l - 1) + (2m - 1) = 2(l + m) - 1 = 2n - 1$ steps. □

9. Prove each of the following statements:

   (a) The class of context-free languages is closed under *union*.

   Solution. Let A and B be two context-free languages. Suppose they may be generated by CFGs (V_A, Σ, R_A, S_A) and (V_B, Σ, R_B, S_B) respectively, where V_A and V_B are disjoint. Then, $(V_A \cup V_B, \Sigma, \{S \to S_A \mid S_B\} \cup R_A \cup R_B, S)$ will be a CFG that generates $L(A) \cup L(B)$. □

   (b) The class of context-free languages is not closed under *intersection*.

   Solution. Let $A = \{a^n b^n c^m \mid n, m \geq 0\}$ and $B = \{a^m b^n c^n \mid n, m \geq 0\}$, which are context free. $A \cap B = \{a^n b^n c^n \mid n \geq 0\}$ is not context free. □

   (c) The class of context-free languages is not closed under *complement*.

   Solution. Intersection may be expressed in terms of complement and union: $A \cap B = \overline{\overline{A} \cup \overline{B}}$. From (a) and (b), the class of context-free languages is closed under the union operation, but it is not closed under the intersection operation. If the class of context-free languages were closed under the complement operation, then it would be closed under intersection, contradicting the result in (b). □

10. Prove, using the pumping lemma, that $\{1^{n^2} \mid n \geq 0\}$ is not context free.

    Solution. Assume toward a contradiction that p is the pumping length for $\{1^{n^2} \mid n \geq 0\}$. Consider a string $s = 1^{p^2}$ in the language. Suppose that s can be pumped by dividing s as $uvxyz = 1^i 1^j 1^k 1^l 1^{p^2-i-j-k-l}$, where $j+l > 0$ ($|vy| \geq 0$) and $j+k+l \leq p$ ($|vxy| \leq p$). If we pump s up to $1^i(1^j)^2 1^k (1^l)^2 1^{p^2-i-j-k-l} = 1^{i+2j+k+2l+p^2-i-j-k-l} = 1^{p^2+j+l}$. As $0 < j+l \leq p$, $p^2 < p^2 + j + l \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$ and hence $1^i(1^j)^2 1^k (1^l)^2 1^{p^2-i-j-k-l}$ is not in $\{1^{n^2} \mid n \geq 0\}$. So, s cannot be pumped, a contradiction. □

Appendix

- A context-free grammar is in **Chomsky normal form** if every rule is of the form

$$\begin{aligned} A &\to BC \text{ or} \\ A &\to a \end{aligned}$$

where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable. In addition,

$$S \to \varepsilon$$

is permitted if S is the start variable.

- (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 $i \geq 0$, $uv^ixy^iz \in A$,
  2. $|vy| > 0$, and
  3. $|vxy| \leq p$.