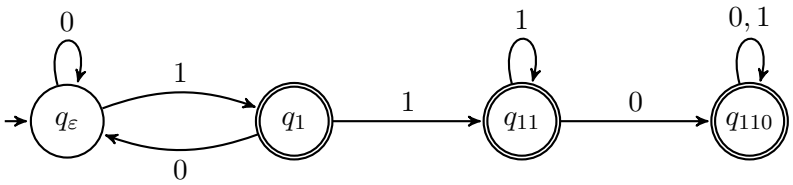
Suggested Solutions to Midterm Problems

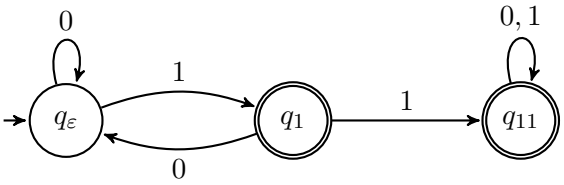
1. Let $A = \{w \in \{0,1\}^* \mid w \text{ contains 110 as a substring or ends with 1}\}$.

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

   Solution.
   First attempt:



   The above is correct, but contains redundancy. Once two consecutive 1s are read (reaching state q_{11}), the input string is destined to be accepted no matter what the remaining symbols are. So, we have the following smaller and optimal DFA for A.



   □

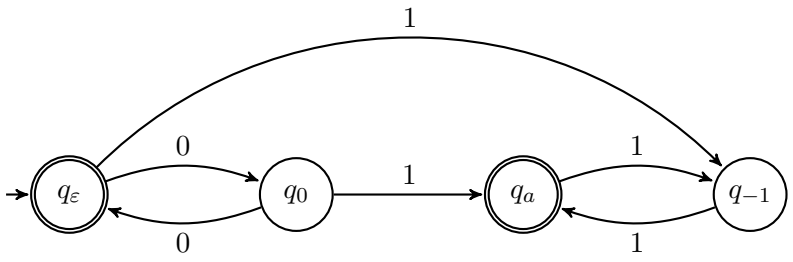
   (b) Give a regular expression, as short as possible, that describes A. The shorter your regular expression is, the more points you will be credited for this problem.

   Solution. Let $\Sigma = \{0,1\}$. Then, A is described by $\Sigma^*(110\Sigma^* \cup 1)$ or $\Sigma^*1(1\Sigma^* \cup \varepsilon)$. □

2. Let $B = \{w \in 0^*1^* \mid \text{the number of 0s and that of 1s in } w \text{ are either both even or both odd}\}$. (Note that, in every string of B, all the 1s should follow the 0s, if any.)

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

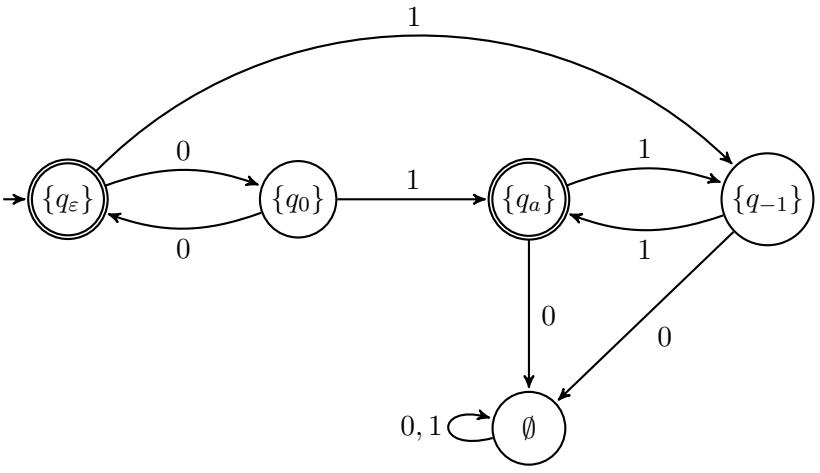
   Solution.



□

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

Solution.



□

3. If A is any language, let $A_{\frac{1}{2}\_}$ be the set of all first halves of strings in A so that

$$A_{\frac{1}{2}\_} = \{x \mid \text{ for some } y, |x| = |y| \text{ and } xy \in A\}.$$

Show that if A is regular, then so is $A_{\frac{1}{2}\_}$.

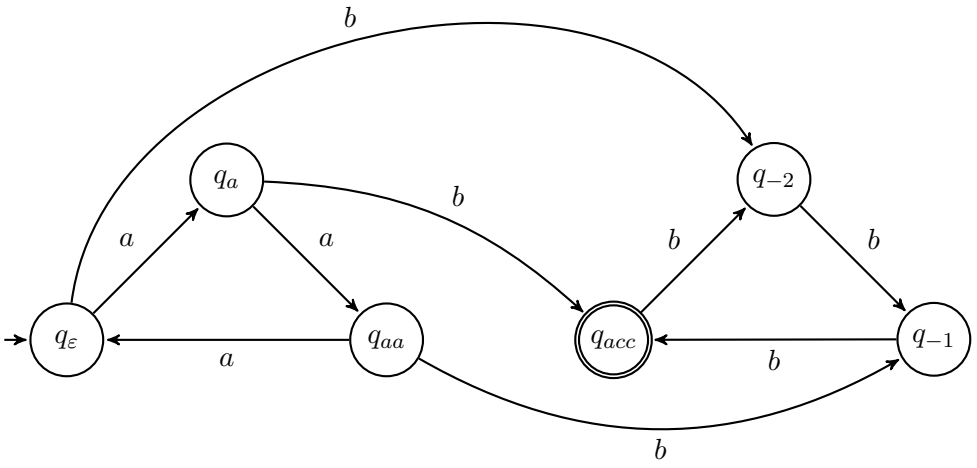
Solution. Suppose M is a DFA that recognizes A. For a finite automaton to recognize $A_{\frac{1}{2}\_}$ (so as to prove $A_{\frac{1}{2}\_}$ is regular), the automaton must be able to "foresee," when some input x is exhausted, whether there exists a string y of the same length as x such that xy will drive M to an accept state; note that string y may be of arbitrary contents but must have the same length as x does. To accomplish this task of "foreseeing" the existence of an adequate y, we will rely on an "extended reverse" M_E^R of M that is to be run side by side with M. M_E^R is obtained from M by reversing the direction of every transition of M and swapping the start and the accept states, resulting in almost an NFA (with possibly multiple start states). For the ease of exposition, we adopt in the following a variant of NFA where there may be more than one start states but without any ε-transition.

To recognize $A_{\frac{1}{2}\_}$, we have an NFA simulate simultaneously M and M_E^R. Note that M and M_E^R have the same set of states. If M and M_E^R meet at the same state when some input x is exhausted, it means that xx^R, with x^R playing the role of y, can drive M to an accept state. Such an input should be accepted. To allow arbitrary contents (not just x^R) for string y, we further extend the symbol on every transition of M_E^R to the whole alphabet; i.e., whenever a transition exists from one state to another for some symbol, we add a transition between the two states for every other symbol.

Formally, let the aformentioned $M = (Q, \Sigma, \delta_A, q_0, F)$ and $M_E^R = (Q, \Sigma, \delta_E^R, F, \{q_0\})$, where $\delta_E^R(q, a) = \{q' \mid \delta_A(q', a) = q \text{ or } \delta_A(q', b) = q \text{ for some } b\}$. Then, the NFA for $A_{\frac{1}{2}\_}$ is $(Q \times Q, \Sigma, \delta, \{q_0\} \times F, \{(q, q) \mid q \in Q\})$, where $\delta((q_A, q_E^R), a) = \{\delta_A(q_A, a)\} \times \delta_E^R(q_E^R, a)$. □

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

   Solution. Let us refer to $\{a^n b^{(n \bmod K)} \mid n > 0\}$ as B_K. With the only exception of $K = 1$, B_K is regular for any given positive integral constant K (i.e., when $K \geq 2$). In particular, mapping symbols a and b respectively to 0 and 1, $B_2 \cup \{\varepsilon\}$ equals language B in Problem 2. The NFA for B may be easily adapted and generalized to recognize B_K for any $K \geq 2$. Below is an NFA for $K = 3$.



   □

5. Let B be the collection of binary strings that contain at least one 1 in their second half, i.e., $B = \{uv \mid u \in \Sigma^*, v \in \Sigma^* 1 \Sigma^*, \text{ and } |u| \geq |v|\}$, where $\Sigma = \{0, 1\}$. Give a CFG that generates B.

   Solution. The following CFG generates B.

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & 0A \mid 1A \mid \varepsilon \\ B & \rightarrow & 0B0 \mid 0B1 \mid 1B0 \mid 1B1 \mid C \\ C & \rightarrow & 0D1 \mid 1D1 \\ D & \rightarrow & 0D0 \mid 0D1 \mid 1D0 \mid 1D1 \mid \varepsilon \end{array}$$

   Variable B generates binary strings of an even length, where the second half contains at least one 1 (enforced by variable C), while variable A adds some 0 or 1, probably none, to the front of B. □

6. Consider the following context-free grammar:

$$S \rightarrow aSaSb \mid aSbSa \mid bSaSa \mid SS \mid \varepsilon$$

   Prove that every string over $\{a, b\}$ with twice as many a's as b's (including the empty string) can be generated from S. (Hint: by induction on the length of a string.)

   Solution. The proof is by induction on the length $|s|$ of a string s where the number of a's is twice the number of b's. It is apparent that $|s|$ equals $3n$ for some $n \geq 0$.

Base case ($|s| = 0$ or $|s| = 3$): When $|s| = 0$, s is the empty string, which can be generated by the rule $S \to \varepsilon$. When $|s| = 3$, there are three possible strings that satisfy the condition, namely aab, aba, and baa. All of them can be generated from S. For instance, aab can be generated from S as follows: $S \Rightarrow aSaSb \Rightarrow aaSb \Rightarrow aab$.

Inductive step ($|s| > 3$): Symbols in the string s may be divided (not necessarily consecutive in positions) into groups of two a's and one b so that every symbol belongs to exactly one group. If we scan s symbol by symbol from left to right and try to divide the symbols into groups of two a's and one b as soon as that becomes possible, either we will reach a point before the end of s where all symbols so far have been successfully grouped or such grouping is never completed until we reach the very last symbol of s.

Case 1: Scanning left to right, we reach a point *before* the end of s where all symbols so far can be successfully grouped. Let $s = xy$ such that scanning the last symbol of x defines the point we have reached, i.e., x is the shortest prefix of s that has twice as many a's as b's. Clearly, the suffix y must also have twice as many a's as b's. From the induction hypothesis, both x and y can be generated from S. It follows that s can be generated from S as follows: $S \Rightarrow SS \Rightarrow^* xS \Rightarrow^* xy$

Case 2: The grouping of two a's and one b has never been completed until we reach the very last symbol of s. To help the analysis, we define $balance(x)$ for any string x over $\{a, b\}$ as follows:

$$balance(x) = \begin{cases} 0 & \text{if } x = \varepsilon \\ balance(y) + 1 & \text{if } x = ya \\ balance(y) - 2 & \text{if } x = yb \end{cases}$$

It is clear that $balance(x) = 0$ iff x has twice as many a's as b's. In the case under consideration, while we scan s from left to right, we have never seen a non-empty proper prefix x of s such that $balance(x)$ is 0.

We claim that s cannot be of the form byb. First we observe that $balance(b) = -2$ and $balance(by)$ must be 2 (for $balance(byb)$ to be 0). An occurrence of a helps increase the value of *balance* by 1 as we scan the string from left to right. To climb up from -2 to 2, we must pass through 0. So, there would have to be a non-empty proper prefix x of $s = byb$ such that $balance(x) = 0$, which is a contradiction. Now, we are left with three forms of s, namely aya, ayb, and bya, to consider. We tackle the case of ayb; others may be treated in a similar way.

If $s = ayb$ and no non-empty proper prefix x of s exists such that $balance(x)$ is 0, we claim that y must be of the form aw; otherwise, $balance(ab) = -1$ and the value of *balance* would to pass through 0 at least once before reaching 2 at $s = ay$, a contradiction. Therefore, s can be divided as $aawb$ where $balance(w)$ must be 0. From the induction hypothesis, w can be generated from S. It follows that $s = aawb$ can be generated from S as follows: $S \Rightarrow aSaSb \Rightarrow aaSb \Rightarrow^* aawb$.

□

7. For any language A, let $SUFFIX(A) = \{v \mid uv \in A \text{ for some string } u\}$. Show that the class of context-free languages is closed under the $SUFFIX$ operation.

   Solution. We need to show that, for every context-free language A, $SUFFIX(A)$ is context free. Given a PDA M_A recognizing A, the basic idea is to construct for $SUFFIX(A)$ a PDA that proceeds in two stages: in the first stage, it simulates an adapted copy of M_A and, in the second stage, an identical copy of M_A. The adapted copy behaves like M_A but does not actually consume the input symbols (or, put in another way, it consumes some

"imaginery" symbols in front of the real input) and, at any state, it nondeterministically makes an ε-transition to the corresponding state in the second copy. Given v as the input, if $uv \in A$ for some string u, then v will drive the constructed PDA to an accept state.

Formally, let $M_A = (Q_A, \Sigma, \Gamma, \delta_A, q_0, F)$ be the PDA that recognizes A. Let Q'_A denote the "primed" version of Q_A, where every state $q' \in Q'_A$ corresponds to a state q in Q_A, i.e., Q'_A is a duplicate of Q_A but with the name q of each state changed to q'. The PDA for $SUFFIX(A)$ is $M_S = (Q'_A \cup Q_A, \Sigma, \Gamma, \delta_S, q'_0, F)$ with δ_S defined as follows.

(a) Simulation of the first adapted copy of M_A. For $q' \in Q'_A$, $a \in \Sigma_\varepsilon$, and $c \in \Gamma_\varepsilon$,

$$\delta_S(q', a, c) = \begin{cases} \emptyset & \text{if } a \neq \varepsilon \\ \{(r', d) \mid (r, d) \in \delta_A(q, b, c) \\ \quad \text{for some } b \in \Sigma_\varepsilon\} & \text{if } a = \varepsilon \text{ and } c \neq \varepsilon \\ \{(r', d) \mid (r, d) \in \delta_A(q, b, c) \\ \quad \text{for some } b \in \Sigma_\varepsilon\} \cup \{(q, \varepsilon)\} & \text{if } a = \varepsilon \text{ and } c = \varepsilon \end{cases}$$

(b) Simulation of the second copy of M_A. For $q \in Q_A$, $a \in \Sigma_\varepsilon$, and $c \in \Gamma_\varepsilon$,

$$\delta_S(q, a, c) = \delta_A(q, a, c)$$

□

8. Let $A/B = \{w \mid wx \in A \text{ for some } x \in B\}$. Show that, if A is context free and B is regular, then A/B is context free.

Solution. Suppose M_A is a PDA recognizing A and M_B a DFA recognizing B. To show that A/B is context free, we construct a PDA $M_{A/B}$ that recognizes A/B. $M_{A/B}$ proceeds in two stages. In the first stage, $M_{A/B}$ simulates M_A on the input w, and additionally in each step nondeterministically guessing that the end of w has been reached, branches to the second stage. When in the second stage, $M_{A/B}$ simulates both M_A and M_B on an imaginary input x, which is "self-supplied", rather than coming from the actual input. $M_{A/B}$ is defined such that every possible x is attempted.

Let $M_A = (Q_A, \Sigma, \Gamma_A, \delta_A, q^0_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q^0_B, F_B)$. The PDA $M_{A/B} = (Q_{A/B}, \Sigma, \Gamma_{A/B}, \delta_{A/B}, q^0_{A/B}, F_{A/B})$ is defined as follows:

- $Q_{A/B} = Q_A \cup Q_A \times Q_B$.
- $\Gamma_{A/B} = \Gamma_A$.
- $\delta_{A/B}$ is defined according to the two stages:
  (a) Simulation of M_A on actual input and nondeterministic branching to the second stage. For $q_A \in Q_A$, $a \in \Sigma_\varepsilon$, and $c \in \Gamma_\varepsilon$,

$$\delta_{A/B}(q_A, a, c) = \begin{cases} \delta_A(q_A, a, c) \cup \{((q_A, q^0_B), c)\} & \text{if } a = \varepsilon \\ \delta_A(q_A, a, c) \cup \{((q'_A, q^0_B), d) \mid (q'_A, d) \in \delta_A(q_A, a, c)\} & \text{if } a \neq \varepsilon \end{cases}$$

  (b) Simulation of M_A and M_B on imaginary input. For $(q_A, q_B) \in Q_A \times Q_B$, $a \in \Sigma_\varepsilon$, and $c \in \Gamma_\varepsilon$,

$$\delta_{A/B}((q_A, q_B), a, c) = \begin{cases} \{((q'_A, q'_B), d) \mid (q'_A, d) \in \delta_A(q_A, a', c) & \text{if } a = \varepsilon \\ \quad \text{for some } a' \in \Sigma \text{ s.t. } q'_B = \delta_B(q_B, a')\} \cup \\ \{((q'_A, q'_B), d) \mid (q'_A, d) \in \delta_A(q_A, \varepsilon, c) \\ \quad \text{and } q'_B = q_B\} \\ \emptyset & \text{if } a \neq \varepsilon \end{cases}$$

- $q^0_{A/B} = q^0_A$.
  So, $\delta_{A/B}$ includes an ε-transition going from $q^0_{A/B}$ to (q^0_A, q^0_B) with no update on the stack.
- $F_{A/B} = \{(q_A, q_B) \mid q_A \in F_A \text{ and } q_B \in F_B\}$.

□

9. Let $C = \{wtw^R \mid w, t \in \{0,1\}^* \text{ and } |w| = |t|\}$, where w^R is the reverse of w. Prove that C is not context free.

   Solution. We take s to be $1^p0^p(01)^p0^p1^p$, where p is the pumping length, and show that s cannot be pumped. Note that s indeed is of the form wtw^R with $w = 1^p0^p$, $t = (01)^p$, and $|w| = 2p = |t|$. Note also that the $(2p+1)$-th symbol of s is a 0, while the last $(2p+1)$-th symbol is a 1; similarly, the $(2p+2)$-th symbol is a 1, while the last $(2p+2)$-th symbol is a 0. Each of the two pairs of symmetrical positions contain different symbols and are sufficiently far apart, so if we pump up s in between the symmetrical positions particularly, the resulting string becomes longer and will not be of the form wtw^R with $|w| = |t|$. There are basically five ways to divide s into $uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$ and we exam each of them below.

   Case 1: vxy falls (entirely) within the substring 1^p0^p. If either v or y saddles on the middle point and contains both 1 and 0, then when we pump down, the first p symbols will contain some trailing 0s and cannot be the reverse of 1^p at the end of the resulting string (which is of length at least $3p$). Otherwise, either v contains some 1s but no 0s or both v and y contain only 0s. In the first case, when we pump up, the first $2p$ symbols will have more 1s than 0s and hence cannot be the reverse of 0^p1^p at the end of the resulting string (which is of length greater than $6p$). In the second case, when we pump up, the $(2p+1)$-th symbol will remain a 0 and the last $(2p+1)$-th symbol will also remain a 1 and hence the resulting string (of length greater than $6p$) cannot be of the form wtw^R (with w of length at least $2p+1$).

   Case 2: vxy falls within $0^p(01)^{\frac{p}{2}}$. In this case, no matter what v and y contain, when we pump up, the $(2p+1)$-th symbol will remain a 0, while the last $(2p+1)$-th symbol will remain a 1, and hence the resulting string (of length greater than $6p$) cannot be of the form wtw^R (with w of length at least $2p+1$).

   Case 3: vxy falls within $(01)^p$. This is analogous to Case 2.

   Case 4: vxy falls within $(01)^{\frac{p}{2}}0^p$. This case is a bit more subtle and is further divided into five subcases:

   (a) both v and y are within $(01)^{\frac{p}{2}}$. When we pump up, the $(2p+1)$-th symbol will remain a 0 and the last $(2p+1)$-th symbol will remain a 1 and hence the resulting string (of length greater than $6p$) cannot be of the form wtw^R (with w of length at least $2p+1$).

   (b) v is within $(01)^{\frac{p}{2}}$ and y saddles on the middle point. The same argument for Subcase (a) applies, when we pump up.

   (c) v is within $(01)^{\frac{p}{2}}$ and y is within 0^p. If y is nonempty, then when we pump up ($i = 2$), the last $(2p+2)$-th symbol will become a 0, while the $(2p+2)$-th symbol will remain a 1; otherwise (y is empty), when we pump up, the same argument for Subcase (a) applies.

   (d) v saddles on the middle point and y is within 0^p. This is analogous to Subcase (c).

(e) both v and y are within 0^p. This is also analogous to Subcase (c).

Case 5: vxy falls within 0^p1^p. This is analogous to Case 4(c).

□

10. Let D be the language of all binary strings that contain two 1s in their middle third, i.e., $D = \{xyz \mid x, z \in \Sigma^* \text{ and } y \in \Sigma^*1\Sigma^*1\Sigma^*, \text{ where } |x| = |z| \geq |y|\}$, where $\Sigma = \{0, 1\}$. Prove that D is not context free.

Solution. We take s to be $0^p10^{p-2}10^p$, where p is the pumping length, and show that s cannot be pumped. Note that s contains exactly two 1s and indeed is of the form $\alpha\beta\gamma$ with $\alpha = 0^p \in \Sigma^*$, $\gamma = 0^p \in \Sigma^*$, $\beta = 10^{p-2}1 \in \Sigma^*1\Sigma^*1\Sigma^*$, and $|\alpha| = |\beta| \geq |\gamma|$. In dividing s into $uvxyz$ such that $|vy| > 0$, we note a clear restriction that neither v nor y may contain a 1; otherwise, when we pump down, the resulting string is left with at most one 1 and hence cannot belong to D. With this restriction enforced, there are basically five ways to divide s into $uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$ and we exam each of them below.

Case 1: vxy falls within the first third 0^p. In this case, when we pump up, we force the second 1 to move into the last third and the resulting string does not belong to D.

Case 2: vxy straddles on the two sides of the first 1 and falls within $0^{p-m-1}10^m$, for some m, $1 \leq m \leq p-2$. With the stated restriction, both v and y are nonempty and contain only 0s. When we pump up, we force the second 1 to move into the last third and the resulting string does not belong to D.

Case 3: vxy falls within 0^{p-2}. When we pump up, we force the second 1 to move into the last third and the resulting string does not belong to D.

Case 4: vxy straddles on the two sides of the second 1 and falls within $0^{p-m-1}10^m$, for some m, $1 \leq m \leq p-2$. Like in Case 2, both v and y are nonempty and contain only 0s. When we pump up, we force the first 1 to move into the first third and the resulting string does not belong to D.

Case 5: vxy falls within the last third 0^p. When we pump down, we force the second 1 to move into the last third and the resulting string does not belong to D.

□

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