Suggested Solutions to Midterm Problems

1. (a) Give a symmetric and transitive but not reflexive binary relation on $A = \{a, b, c, d\}$ that includes $\{(a, b), (b, c)\}$; it may be a good idea to represent the relation by a directed graph.

   Solution. (謝佳慈)

   $R = \{(a, b), (b, c), (a, c), (b, a), (c, b), (c, a), (a, a), (b, b), (c, c)\}$. (Note: $\{d, d\} \notin R$.) □

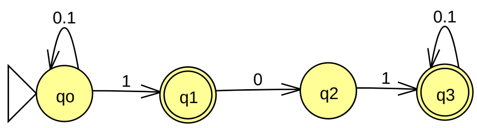
   (b) Let $R = \{(a, c), (b, c), (b, d)\}$ be a binary relation on $A = \{a, b, c, d, e\}$. Find the smallest equivalence relation on A that includes R; again, it may be a good idea to represent the relation by a directed graph.

   Solution. (謝佳慈)

   $R = \{(a, c), (b, c), (b, d), (c, a), (c, b), (d, b), (a, b), (b, a), (a, d), (d, a), (c, d), (d, c),$
   $(a, a), (b, b), (c, c), (d, d), (e, e)\}$. □

2. (a) Draw the state diagram of an NFA, with as few states as possible, that recognizes $\{w \in \{0, 1\}^* \mid w \text{ contains } 101 \text{ as a substring or ends with } 1\}$. The fewer states your NFA has, the more points you will be credited for this problem. (5 points)

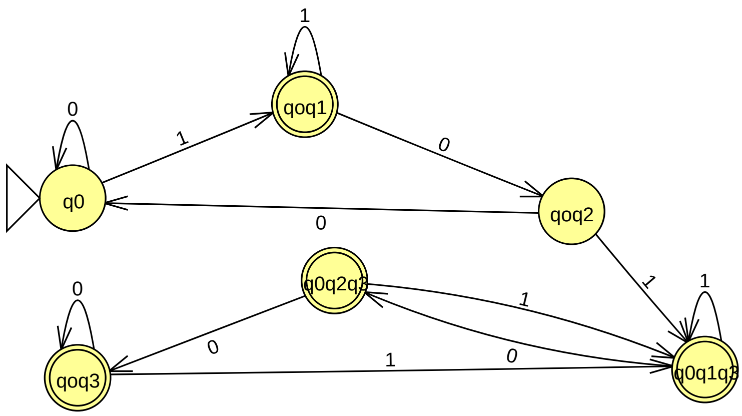
   Solution. (謝佳慈)



□

   (b) Convert the NFA in (a) systematically into an equivalent DFA (using the procedure discussed in class); do not attempt to optimize the number of states. (10 points)

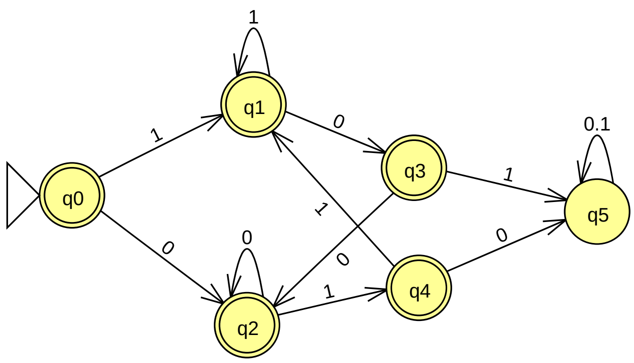
   Solution. (謝佳慈)



□

3. (a) Draw the state diagram of a DFA, with as few states as possible, that recognizes $\{w \in \{0,1\}^* \mid w$ does not contain 101 or 010 as a substring$\}$. The fewer states your DFA has, the more points you will be credited for this problem. (10 points)

Solution. (謝佳慈)



□

(b) Translate the DFA in (a) systematically to an equivalent context-free grammar (using the procedure discussed in class), which is called regular grammar. (5 points)

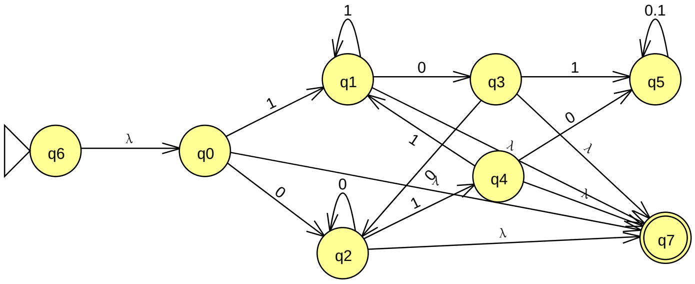
Solution. (謝佳慈)

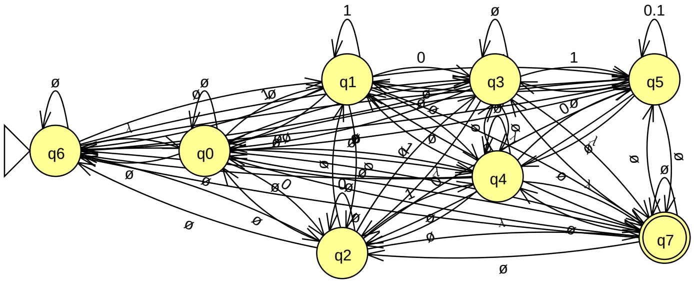
$$
\begin{aligned}
A_0 &\rightarrow 1A_1 \mid 0A_2 \mid \varepsilon \\
A_1 &\rightarrow 0A_3 \mid 1A_1 \mid \varepsilon \\
A_2 &\rightarrow 0A_2 \mid 1A_4 \mid \varepsilon \\
A_3 &\rightarrow 0A_2 \mid \varepsilon \\
A_4 &\rightarrow 1A_1 \mid \varepsilon
\end{aligned}
$$

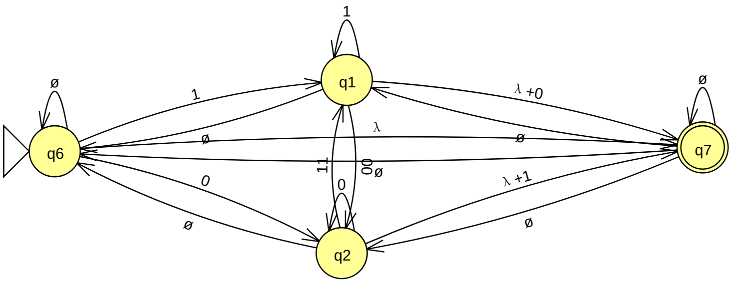
□

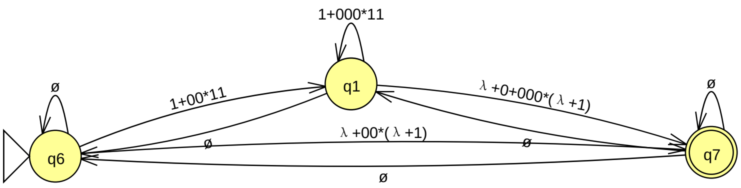
4. Write a regular expression for the language in Problem 3.

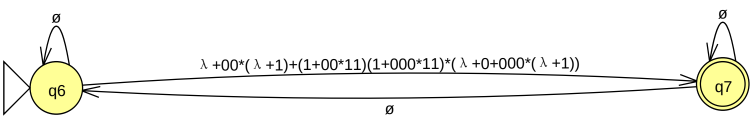
Solution. There are quite a few equivalent regular expressions for the language, depending on how you carry out the conversion steps. Below are a few snapshots of the conversion with the assistance of JFLAP.







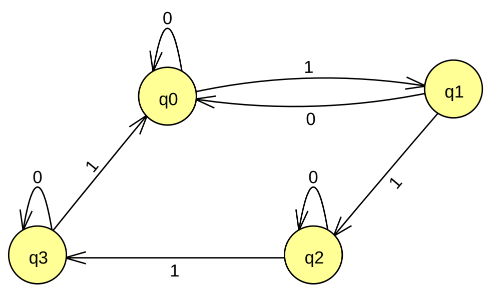




□

5. A *synchronizing sequence* for a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and some "home" state $h \in Q$ is a string $s \in \Sigma^*$ such that, for every $q \in Q$, $\delta(q, s) = h$. A DFA is said to be *synchronizable* if it has a synchronizing sequence for some state. Try to find a 4-state synchronizable DFA with a synchronizing sequence as long as possible. The longer the synchronizing sequence is, the more points you will be credited for this problem.

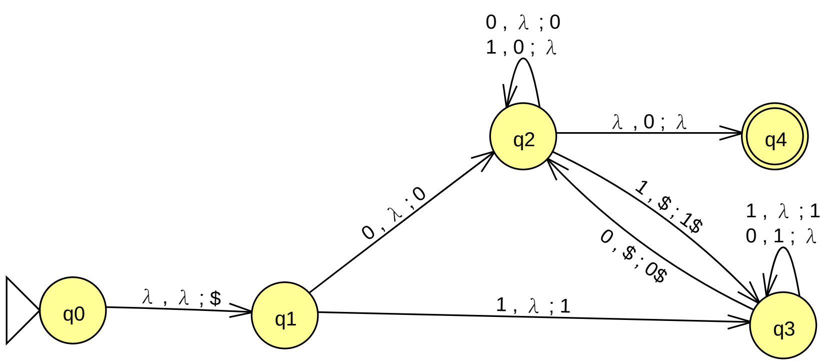
   Solution. Though not clearly stated in the problem, the synchronizing sequence we seek should be minimal in the sense that none of its proper substrings is also a synchronizing sequence. (Otherwise, the problem is not very interesting.) Below is a 4-state synchronizable DFA with a minimal synchronizing sequence of length 5, namely 01010. (Other 4-state synchronizable DFAs with a longer minimal synchronizing sequence might exist.)



□ Note: the final state is not marked, as it is irrelevant.

6. Draw the state diagram of a pushdown automaton (PDA) that recognizes the following language: $\{w \mid w \in \{0,1\}^* \text{ and } w \text{ has more 0's than 1's}\}$. Please make the PDA as simple as possible and explain the intuition behind the PDA.

   Solution. (詹文欽)



□

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. (詹文欽)

Prove by construction:

We run a finite automaton "in parallel" with a PDA, and the result is another PDA. Formally, let

$$P = (Q_P, \sum, \Gamma, \delta_P, q_P, Z_0, F_P)$$

be a PDA that accepts L (by final state), and let

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

be a DFA for R. Construct PDA

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

where $\delta((q,p), a, X)$ is defined to be the set of all pairs $((r,s),\gamma)$ such that:

1. $s = \hat{\delta}_A(p, a)$ and

2. $(r, \gamma) \in \delta_P(q, a, X)$.

□

8. For two given languages A and B, define $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$. Prove that, if A and B are regular, then $A \diamond B$ is context-free. (Hint: construct a PDA where the stack is used to ensure that x and y are of equal length.)

   Solution. Given finite-state automata N_A and N_B respectively for A and B, the basic idea is to construct a PDA for recognizing $A \diamond B$ that first simulates N_A and then non-deterministically switches to simulate N_B. The PDA counts the number of symbols while simulating N_A by pushing a marker onto the stack whenever it reads an input symbol and it later cancels out the markers with the input symbols while simulating N_B.

   Suppose $N_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $N_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$, assuming A and B have the same alphabet. We construct the PDA $M = (Q, \Sigma, \Gamma, \delta, q_{\text{start}}, \{q_{\text{accept}}\})$ for $A \diamond B$ as follows:

   - $Q = \{q_{\text{start}}, q_{\text{accept}}\} \cup Q_A \cup Q_B$, where $q_{\text{start}}, q_{\text{accept}} \notin Q_A \cup Q_B$.
   - $\Gamma = \{x, \$\}$.
   - δ is defined as follows.

$$\begin{cases} \delta(q_{\text{start}}, \varepsilon, \varepsilon) = \{(q_A, \$)\} & \\ \delta(q, a, \varepsilon) = \{(q', x) \mid q' \in \delta_A(q, a)\} & q \in Q_A \text{ and } a \neq \varepsilon \\ \delta(q, \varepsilon, \varepsilon) = \{(q', \varepsilon) \mid q' \in \delta_A(q, \varepsilon)\} & q \in Q_A \\ \delta(q, \varepsilon, \varepsilon) = \{(q_B, \varepsilon)\} & q \in F_A \\ \delta(q, a, x) = \{(q', \varepsilon) \mid q' \in \delta_B(q, a)\} & q \in Q_B \text{ and } a \neq \varepsilon \\ \delta(q, \varepsilon, \varepsilon) = \{(q', \varepsilon) \mid q' \in \delta_B(q, \varepsilon)\} & q \in Q_B \\ \delta(q, \varepsilon, \$) = \{(q_{\text{accept}}, \varepsilon)\} & q \in F_B \\ \delta(q, a, t) = \emptyset & \text{otherwise} \end{cases}$$

It should be clear that $L(M) = A \diamond B$; we omit the detailed proof. $\square$

9. Prove, using the pumping lemma, that $\{x\#wxy \mid w, x, y \in \{a, b\}^*\}$ is not context-free. (Hint: consider $s = a^p b^p \# a^p b^p$, where p is the pumping length.)

   Solution. Following the hint, we take s to be $a^p b^p \# a^p b^p$, where p is the pumping length, and show that s cannot be pumped. There are basically three ways to divide s into $uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$:

   Case 1: vxy falls within the first occurrence of $a^p b^p$ (before #). No matter how we divide s, when we pump *up*, the substring before # will become longer than the one after # and the whole string cannot belong to the language.

   Case 2: vxy falls within the substring $b^p \# a^p$. Neither v nor y may contain #, otherwise we will get more than one #'s when we pump up the string. So, s must be divided as $uvxyz = (a^p b^{p-j-k})(b^j)(b^k \# a^l)(a^m)(a^{p-l-m} b^p)$, where $j, k, l, m \geq 0$ and j and m can not both be 0. If $j > 0$, we pump *up* to get $uv^2xy^2z = (a^p b^{p-j-k})(b^{2j})(b^k \# a^l)(a^{2m})(a^{p-l-m} b^p)$. The substring before # will have more b's than the one after # and hence the whole string cannot belong to the language. If $m > 0$, we pump *down* to get $uv^0xy^0z = (a^p b^{p-j-k})(\varepsilon)(b^k \# a^l)(\varepsilon)(a^{p-l-m} b^p)$. The substring before # will have more a's than the one after # and hence the whole string cannot belong to the language.

   Case 3: vxy falls within the second occurrence of $a^p b^p$ (after #). No matter how we divide s, when we pump *down*, the substring after # will become shorter than the one before # and the whole string cannot belong to the language.

   $\square$

10. Consider the following context-free grammar:

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

   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.) (bonus 10 points)

   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 $S \rightarrow \varepsilon$. 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: $S \Rightarrow aSaSb \Rightarrow aaSb \Rightarrow^* aawb$.

$\square$

Appendix

- Properties of a binary relation R on A:
  - R is *reflexive* if for every $x \in A$, xRx.
  - R is *symmetric* if for every $x, y \in A$, xRy if and only if yRx.
  - R is *transitive* if for every $x, y, z \in A$, xRy and yRz implies xRz.

- 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) $|vxy| \leq p$.