1. For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and *briefly* argue why your expression is correct.

   (a) All strings except 001.

   Solution: Enumerate all valid strings of length at most 3 by brute force, and then include all strings of length at least 4, exactly as in the Lab 1½ solutions.

$$\begin{aligned} &\varepsilon + 0 + 1 + 00 + 01 + 10 + 11 \\ &+ 000 \qquad\quad + 010 + 011 \\ &+ 100 + 101 + 110 + 111 \\ &+ (0+1)(0+1)(0+1)(0+1)(0+1)^* \end{aligned}$$

   ■

   Non-solution ("clever"): If the string is non-empty, it must start with one of the prefixes 1, 01, or 000.

$$\varepsilon + (1 + 01 + 000)(0 + 1)^*$$

   This expression incorrectly excludes both proper prefixes of 001 and strings for which 001 is a proper prefix. Everybody gets full credit for this subproblem as extra credit. ♣

   Solution (clever but correct anyway): Either the string is a proper prefix of 001; or 001 is a proper prefix of the string; or the string starts with one of the prefixes 1, 01, or 000.

$$\varepsilon + 0 + 00 \ + \ 001(0+1)(0+1)^* \ + \ (1 + 01 + 000)(0+1)^*$$

   ■

   (b) All strings that end with the suffix 001001.

   Solution: $(0+1)^*001001$ — Any string followed by 001001. ■

   (c) All strings that contain the substring 001.

   Solution: $(0+1)^* \cdot 001 \cdot (0+1)^*$ — Any string followed by 001 followed by any string. ■

   (d) All strings that contain the subsequence 001.

   Solution: $(0+1)^* \cdot 0 \cdot (0+1)^* \cdot 0 \cdot (0+1)^* \cdot 1 \cdot (0+1)^*$ — Alternate between arbitrary strings and symbols in 001. ■

(e) All strings that do not contain the substring 001.

Solution: $(1+01)^*0^*$ — Every 0 is immediately followed by a 1, except possibly for a run of 0s that ends the string. ■

Solution: $1^*(011^*)^*0^*$ — After an optional initial run of 1s, every 0 is followed by a non-empty run of 1s, except for an optional final run of 0s. ■

(f) All strings that do not contain the subsequence 001.

Solution: $1^*0^* + 1^*01^*10^*$ — Either every 0 is after every 1, or exactly one 0 appears before the last 1. ■

Solution: $1^* + 1^*01^*0^*$ — Either the string has no 0s, or every 1 after the first 0 is before all other 0s. ■

Rubric: 10 points:

- Parts (a)–(d): 1 for each regular expression + ½ for each explanation
- Parts (e)–(f): 1 for each regular expression + 1 for each explanation

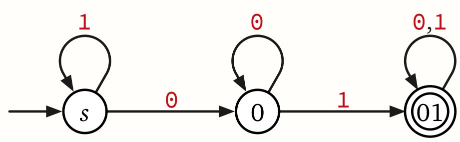
These are not the only correct answers!

2. Let L denote the set of all strings in $\{0,1\}^*$ that contain all four strings 00, 01, 10, and 11 as substrings. For example, the strings 110011 and 01001011101001 are in L, but the strings 00111 and 1010101 are not.

   Formally describe a DFA with input alphabet $\Sigma = \{0,1\}$ that accepts the language L, by explicitly describing the states Q, the start state s, the accept states A, and the transition function δ. Argue that your machine accepts every string in L and nothing else, by explaining what each state in your DFA *means*.

Solution (Four-way product construction): We build a DFA M has a product of four smaller DFAs M_0, M_1, M_2, and M_3, which respectively accept strings containing the substrings 00, 01, 10, and 11.

M_0 and M_3 are described in the lecture notes. M_1 is the following three-state machine:



The states of M_1 are defined has follows:

- s — The start state. We have not read any 0s.
- 0 — We have read at least one 0, but no 1 after a 0.
- 01 — The unique accept state. We have read a 0 followed by a 1.

Finally, M_2 is obtained from M_1 by swapping 0 and 1 in the edge labels.

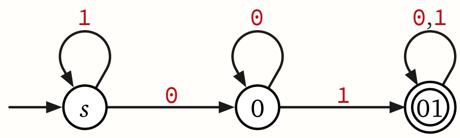
The final DFA M is a four-way product construction with $3^4 = 81$ states.

$$\begin{aligned} Q &= Q_0 \times Q_1 \times Q_2 \times Q_3 \\ &= \big\{(q_0,q_1,q_2,q_3) \bigm| q_0 \in Q_0,\ q_1 \in Q_1,\ q_2 \in Q_2,\ q_3 \in Q_3\big\} \\ s &= (s_0,s_1,s_2,s_3) \\ \delta((q_0,q_1,q_2,q_3),a) &= \big(\delta_0(q_0,a),\ \delta_1(q_1,a),\ \delta_2(q_2,a),\ \delta_3(q_3,a)\big) \\ A &= A_0 \times A_1 \times A_2 \times A_3 \\ &= \big\{(q_0,q_1,q_2,q_3) \bigm| q_0 \in A_0,\ q_1 \in A_1,\ q_2 \in A_2,\ q_3 \in A_3\big\} \end{aligned}$$

■

Solution (Four-way product construction): We build a DFA M has a product of four smaller DFAs M_0, M_1, M_2, and M_3, which respectively accept strings containing the substrings `00`, `01`, `10`, and `11`.

M_0 and M_3 are described in the lecture notes. M_1 is the following three-state machine:



The states of M_1 are defined has follows:

- s — The start state. We have not read any `0`s.
- 0 — We have read at least one `0`, but no `1` after a `0`.
- 01 — The unique accept state. We have read a `0` followed by a `1`.

Finally, M_2 is obtained from M_1 by swapping `0` and `1` in the edge labels. Each of these four machines has a unique accepting state; for each index i, let t_i denote the accepting ("target") state of machine M_i.

The final DFA M is a four-way product construction, which we execute by repeated pairwise product constructions, as described in class and in the notes.

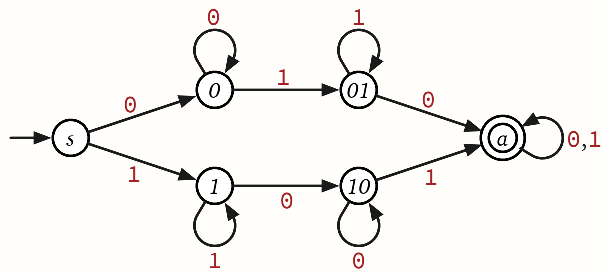
- M_4 is a standard product construction of M_0 and M_1, with unique accepting state $t_4 = (t_0, t_1)$. This machine accepts all strings containing both `00` and `01` as substrings.
- M_5 is a standard product construction of M_2 and M_3, with unique accepting state $t_5 = (t_2, t_3)$. This machine accepts all strings containing both `10` and `11` substrings.
- Finally, M is a standard product construction of M_4 and M_4, with unique accepting state $(t_4, t_5) = ((t_0, t_1), (t_2, t_3))$. This machine accepts the language L.

■

Solution (Two-way product construction): We define a DFA M has a product of three smaller DFAs M_1, and M_2, where

- M_1 accepts all strings containing the substrings 00 and 11, and
- M_2 accepts all strings containing the substrings 01 and 10.

M_1 is already described in the lecture notes. M_2 is the following six-state DFA:



The states of M_3 are defined as follows:

- s — The start state; we haven't read any input
- 0 — We've read only 0s.
- 1 — We've read only 1s.
- 01 — We've read 01 but not 10.
- 10 — We've read 10 but not 01.
- a — The accept state; we've read both 01 and 10.

The final DFA M is a standard product construction with $8 \times 6 = 48$ states.

$$
\begin{aligned}
Q &= Q_1 \times Q_2 = \left\{(q_1, q_2) \;\middle|\; q_1 \in Q_1,\ q_2 \in Q_2\right\} \\
s &= (s_1, s_2) \\
\delta((q_1, q_2), a) &= \left(\delta_0(q_1, a),\ \delta_1(q_2, a)\right) \\
A &= A_1 \times A_2 = \left\{(q_1, q_2) \;\middle|\; q_1 \in A_1,\ q_2 \in A_3\right\}
\end{aligned}
$$

■

Solution (Direct construction): We formally define a DFA $M = (Q, \Sigma, s, A, \delta)$ as follows, where $\Sigma = \{0, 1\}$. Recall that $\Sigma^2 = \{00, 01, 10, 11\}$.

- The states are $Q = \big\{(S, a) \bigm| S \subseteq \Sigma^2 \text{ and } a \in \Sigma\big\} \cup \{s\}$.
  - For any subset $S \subseteq \Sigma^2$ and any symbol $a \in \Sigma$, the state (S, a) indicates that we have read all substrings in S but no other length-2 substrings, and the last symbol read was a.
  - The special state s is the start state.

  There are 33 states altogether.
- There are exactly two accept states: $A = \big\{(\Sigma^2, 0),\ (\Sigma^2, 1)\big\}$.
- Finally, the transition function δ is defined as follows:

$$\begin{aligned} \delta(s, a) &= (\varnothing, a) && \text{for all } a \in \Sigma \\ \delta\big((S, a), b\big) &= (S \cup \{ab\}, b) && \text{for all } S \subseteq \Sigma^2 \text{ and } a, b \in \Sigma \end{aligned}$$

For example, The state $(\{00, 01\}, 1)$ indicates that we have already read the substrings 00 and 01, but not the substrings 10 and 11, and the last input symbol read was a 1. If the next input symbol is a 0, then we transition to

$$\delta\big((\{00, 01\}, \underline{1}),\ \underline{0}\big) \;=\; (\{00, 01, \underline{10}\}, 0)$$

because now we *have* read the substring 10.

This DFA can be simplified by observing that the following 12 states are unreachable from s and therefore can be discarded.

$(\{00\},1)$	$(\{10\},1)$	$(\{00,10\},1)$
$(\{01\},0)$	$(\{11\},0)$	$(\{01,11\},0)$
$(\{00,01\},0)$	$(\{00,11\},0)$	$(\{00,01,11\},0)$
$(\{00,11\},1)$	$(\{10,11\},1)$	$(\{00,10,11\},1)$

Moreover, because the two accepting states only transition to each other, they can be merged into a single accepting state. The resulting DFA has 20 states, which is actually the smallest possible for this language. ■

Rubric: 10 points; standard DFA rubric. These are not the only solutions! In particular, the last solution is not the only way to describe the minimal 20-state DFA.

3. Let L be the set of all strings in $\{0,1\}^*$ that contain *exactly one* occurrence of the substring 010.

(a) Give a regular expression for L, and briefly argue why your expression is correct. *[Hint: You may find the shorthand notation $A^+ = AA^*$ useful.]*

Solution:

$$1^* \cdot (0^+11^+)^* \cdot (0^+10^+) \cdot (11^+0^+)^* \cdot 1^*$$

- Somewhere in the string is a single 1 surrounded by non-empty runs of 0s. Call this the *central substring*.
- Every run of 1s before the central substring either starts the string, or has length at least 2 and is preceded by a run of 0s.
- Every run of 1s after the central substring either ends the string, or has length at least 2 and is followed by a run of 0s.

■

Solution: The following regular expression is extracted from the DFA in part (b) using Han and Wood's algorithm:

$$(1 + 00^*11)^* \cdot (00^*10) \cdot (0 + 111^*0)^* \cdot 1^*$$

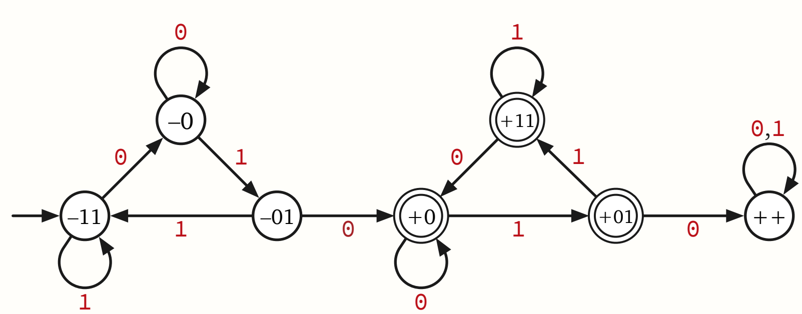
- The first subexpression $(1 + 00^*11)^*$ represents all strings that take the DFA from the start state -11 back to the start state -11.
- The second subexpression (00^*10) represents all strings that take the DFA from the start state -11 to the first accepting state $+0$ without revisiting -11.
- The third subexpression $(0 + 111^*0)^*$ represents all strings that take the DFA from state $+0$ to $+0$
- Finally, the fourth subexpression 1^* represents all strings that take the DFA from $+0$ to another accept state without revisiting $+0$.

■

Rubric: 5 points: Standard regular expression rubric (scaled). These are not the only correct solutions. The second solution requires a correct DFA in part (b).

(b) Describe a DFA over the alphabet $\Sigma = \{0,1\}$ that accepts the language L.

Solution: The following seven-state DFA accepts L.



The states are mnemonically named as follows:

- −11 — The start state. Either we have read only `1`s, or the last two symbols read were `11` and we have not seen `010`.
- −0 — The last symbol read was `0` and we have not seen `010`.
- −01 — the last two symbols read were `01` and we have not seen `010`.
- +0 — The last symbol read was `0` and we *have* seen `010`.
- +01 — The last two symbols read were `01` and we *have* seen `010`.
- +11 — The last two symbols read were `11` and we *have* seen `010`.
- ++ — A dump state. We have seen `010` more than once.

■

Rubric: 5 points: Standard DFA design rubric (scaled)