4 (100 PTS.) Regular expressions.

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.

4.A. (10 PTS.) All strings that end in 1011.

Solution: $(0+1)^*1011$.

The expression $(0+1)^*$ matches a string of 1's and 0's of arbitrary length. The substring 1011 in the end of the regular expression ensures all matching strings must end with it.

4.B. (10 PTS.) All strings except 11.

Solution: $\epsilon + 1 + 0 + 00 + 01 + 10 + (0+1)^3(0+1)^*$
The first part $\epsilon+1+0+00+01+10$ covers all strings up to length 2, explicitly not listing 11. The expression $(0+1)^3(0+1)^*$ represents all strings of length three or greater.

4.C. (10 PTS.) All strings that contain 101 or 010 as a substring.

Solution: $(0+1)^*(101+010)(0+1)^*$.

The expression $(0+1)^*$ refers to all binary strings of length zero or more. A string with 111 substring, would have 111 with any number of pre/post-ceding 0's and 1's . Matching 000 is accomplished by adding it to the substring expression.

4.D. (10 PTS.) All strings that contain 111 and 000 as a subsequence (the resulting expression is long – describe how you got your expression, instead of writing it out explicitly).

Solution: Let S be the set of all strings of length 6 that have the desired property. That is, we have

$$S = \left\{ \begin{array}{l} 000111, 001011, 001101, 001110, 010011, \\ 010101, 010110, 011001, 011010, 011100, \\ 111000, 110100, 110010, 110001, 101100, \\ 101010, 101001, 100110, 100101, 100011 \end{array} \right\}.$$

There are $\binom{6}{3} = 6 \cdot 5 \cdot 4/6 = 20$, such strings, and they all belong to our language. For a string s, let $f(s)$ be the regular expression of inserting $(0+1)^*$ between any two characters of s, and also in the beginning of s and the end of s. For example, we have $f(01) = (0+1)^*0(0+1)^*1(0+1)^*$. The desired expression is

$$+_{s \in S} f(s).$$

4.E. (10 PTS.) The language containing all strings that do not contain 111 as a substring.

Solution: $((\varepsilon + 1 + 11)0^+)^*(\varepsilon + 1 + 11)0^* = ((\varepsilon + 1 + 11)0)^*(\varepsilon + 1 + 11)0^*$.

A ***run*** is a string s is a maximal substring of s, all made of the same character. Given a string w in the language, break it into blocks, where a block starts just before a run of 1s start. For example,

$$010001101011 = 0|1000|110|10|11.$$

Such a block starts with a run of 1s of length zero, one, or two. It is then followed by a run of at least one 0. The only exception is the last block, which does have to end with zeros.

4.F. (10 PTS.) All strings that do *not* contain 000 as a subsequence.

Solution: This is just a complicated way to say that the string can contain at most two 0s. Before, in between and after the zeros, we can have a run of ones. As such, the solution is: $1^*(\varepsilon + 0)1^*(\varepsilon + 0)1^*$.

4.G. (10 PTS.) Strings in which every occurrence of the substring 00 appears before every occurrence of the substring 11.

Solution: $(10 + 0)^*(1 + 10)^*$.
Indeed, break the string in the language into two parts. The prefix p that contains no consecutive 1s, and the suffix s that contain no consecutive 0s.
The expression $(10 + 0)^*$ includes all strings with no consecutive 1s, and matches p. Similarly, the expression $(1 + 10)^*$ includes all strings with no consecutive 0s.

4.H. (10 PTS.) Strings that do not contain the subsequence 010.

Solution: $1^*0^*1^*$.
The condition implies that there could not be two runs of zeros in the string.

4.I. (10 PTS.) Strings that do not contain the subsequence 0101010.

Solution: $1^*0^*1^*0^*1^*0^*1^*$.
The condition implies that there could be at most three runs of zeros in the string.

4.J. (10 PTS.) Strings that do not contain the subsequence 10.

Solution: 0^*1^*.
Such strings can not contain a run of 1s followed by a run of 0s. As such, it can contain only a run of 0s followed by a run of 1s.

4.K. (Not for credit, do not submit a solution.) Strings that do not contain the subsequence 111000.

Solution: Build a DFA for the language, and then convert it to a regular expression. Not fun.

5 (100 PTS.) DFA I

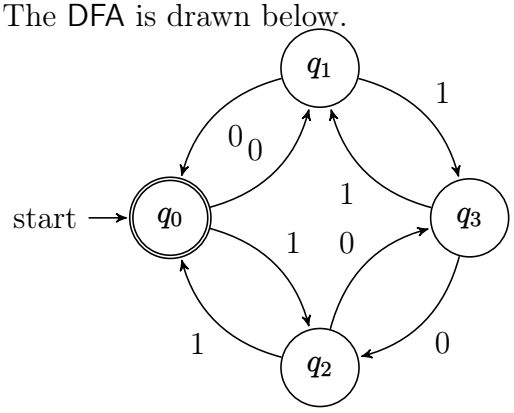
Let $\Sigma = \{0, 1\}$. Let L be the set of all strings in Σ^* that contain an even number of 0s and an even number of 1s.

5.A. (50 PTS.) Describe a DFA over Σ that accepts the language L. Argue that your machine accepts every string in L and nothing else, by explaining what each state in your DFA *means*. (Hint: Zero is even)

You may either draw the DFA or describe it formally, but the states Q, the start state s, the accepting states A, and the transition function δ must be clearly specified, in either case.

Solution:

The DFA is drawn below.



(i) q_0: Strings with even number of 1s and even number of 0's.
(ii) q_1: Strings with even number of 1s and odd number of 0's.
(iii) q_2: Strings with odd number of 1s and even number of 0's.
(iv) q_3: Strings with odd number of 1s and odd number of 0's.

Because zero is an even number, the initial and accepting state are the same. You can have four states which represent if either the number of 0's and the number of 1's is even/odd. In the above DFA, all the transitions are drawn to obey the listed definitions of states.

5.B. (50 PTS.) (Harder.) Give a regular expression for L, and briefly argue why the expression is correct. (Hint: First solve the much easier case where the strings do not contain any consecutive 0s or 1s.)

Solution:

Let L be the desired language. Note, that if a string $w \in L$, then a string w' resulting from removing from it a substring 00 or 11 is still in L (because the number of zeros and ones in w' is even, and $|w'| = |w| - 2$.

As such, given a string $w \in L$, consider the string resulting from repeatedly removing from w any substring that is either 00 or 11.

Clearly, in the end of this tragic removal of all the consecutive pairs of 0s and 1s, what remains must be an alternating string w'' of the form $(0101)^* + (1010)^*$. Let $r = (00 + 11)^*$, and observe that we can regenerate the original string by injecting between any two characters of w'', a string that belongs to r. As such, the regular expression for L is

$$(r0r1r0r1r)^* + (r1r0r1r0r)^*, \qquad \text{where} \qquad r = (00 + 11)^*.$$

Solution:

Here is an alternative solution, using a more advanced technique of converting the DFA to a regular expression.

$(11 + 00 + (10 + 01)(11 + 00)^*(10 + 01))^*$

This example is intended to demonstrate deriving regular expressions for non-intuitive problems using DFA diagrams. Using the DFA above you capture state functions as:

1. $q_0 = \varepsilon + q_1 0 + q_2 1$
2. $q_1 = q_0 0 + q_3 1$
3. $q_2 = q_0 1 + q_3 0$
4. $q_3 = q_1 1 + q_2 0$

Subbing (2)/(3) into (4) yields $q_3 = (q_0 0 + q_3 1)1 + (q_0 1 + q_3 0)0 = q_0(10 + 01) + q_3(11 + 00)$. Using Arden's rule gives us $q_3 = q_0(10 + 01)(11 + 00)^*$.

Next we sub (2)/(3) into (1) yielding $q_0 = \varepsilon + q_0(11 + 00) + q_3(10 + 01)$. Subbing in the expression for q_3 and using Arden's rule again yields the solution above.

6 (100 PTS.) DFA II

Let L_1, L_2, and L_3 be regular languages over Σ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, and $M_3 = (Q_3, \Sigma, \delta_3, s_3, A_3)$, respectively.

6.A. (20 PTS.) Describe formally the product construction of the DFA M that accepts the language $L_1 \cap L_2 \cap L_3$.

Solution:

Let $M = (Q, \Sigma, \delta, s, A)$, where:

(i) Set of states $Q = Q_1 \times Q_2 \times Q_3 = \{(q_1, q_2, q_3) \mid q_1 \in Q_1, q_2 \in Q_2, q_3 \in Q_3\}$.

(ii) Transition function $\delta : Q_1 \times Q_2 \times Q_3 \times \Sigma \to Q_1 \times Q_2 \times Q_3$.

$\forall (q_1, q_2, q_3) \in Q_1 \times Q_2 \times Q_3,\ c \in \Sigma \qquad \delta\big((q_1, q_2, q_3),\ c\big) = \big(\delta_1(q_1, c), \delta_2(q_2, c), \delta_3(q_3, c)\big).$

(iii) Start state: $s = (s_1, s_2, s_3)$.

(iv) Accept states: $A = A_1 \times A_2 \times A_3$.

6.B. (30 PTS.) In the DFA M constructed in (6.A.), a state is a triple (q_1, q_2, q_3). Let δ the transition function of M, and let δ^* be the standard extension of δ to strings. Prove by induction that for any string $w \in \Sigma^*$, we have that

$$\delta^*\big((q_1, q_2, q_3), w\big) = \big(\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w)\big).$$

Solution:

Lemma 2.1. *For any string $w \in \Sigma^*$ and state $q = (q_1, q_2, q_3) \in Q$, we have*

$$\delta^*(q, w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w)). \tag{1}$$

Proof: The proof is by induction on the length of the input string. Let w be any string of length n.

Base case. If $n = 0$, then $w = \epsilon$. Then for any state $q = (q_1, q_2, q_3) \in Q$, we have

$$\delta^*((q_1, q_2, q_3), \epsilon) = (q_1, q_2, q_3) = (\delta_1^*(q_1, \epsilon), \delta_2^*(q_2, \epsilon), \delta_3^*(q_3, \epsilon)).$$

Inductive Hypothesis. Assume that for any string w of length k with $k < n$, and any w, Eq. (1) holds.

Inductive step. We now prove the claim for $n = k$ for $k > 0$. Then $w = ax$ for some $a \in \Sigma$ and $x \in \Sigma^*$ where $|x| < n$. Let $q = (q_1, q_2, q_3) \in Q$. Then we have

$$\begin{aligned}
\delta^*\big((q_1, q_2, q_3), w\big) &= \delta^*\Big(\delta\big((q_1, q_2, q_3), a\big), x\Big) && \text{(by definition of } \delta^*\text{)} \\
&= \delta^*\Big(\big(\delta_1(q_1, a), \delta_2(q_2, a), \delta_3(q_3, a)\big), x\Big) && \text{(by definition of } \delta\text{)}
\end{aligned}$$

Let $q_i' := \delta_i(q_i, a)$, for $i = 1, 2, 3$. Therefore,

$$\begin{aligned}
&\delta^*((q_1, q_2, q_3), w) = \delta^*((q_1', q_2', q_3'), x) \\
&= (\delta_1^*(q_1', x), \delta_2^*(q_2', x), \delta_3^*(q_3', x)) && \text{(by inductive hypothesis since } |x| < n\text{)} \\
&= (\delta_1^*(\delta(q_1, a), x), \delta_2^*(\delta(q_2, a), x), \delta_3^*(\delta(q_3, a), x)) \\
&= (\delta_1^*(q_1, ax), \delta_2^*(q_2, ax), \delta_3^*(q_3, ax)) && \text{(by definition of } \delta_i^* \text{ for each } 1 \le i \le 3\text{)} \\
&= (\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w)) && \text{(since } w = ax\text{).}
\end{aligned}$$

■

6.C. (20 PTS.) Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of M_1, M_2, and M_3 that accepts $L = \left\{ w \;\middle|\; w \text{ is in exactly two of } \{L_1, L_2, L_3\} \right\}$. Formally specify the components Q, δ, s, and A for M in terms of the components of M_1, M_2, and M_3. Argue that your construction is correct.

Solution:

We use the construction of **(6.A.)**, except that we need to modify the accept states.

$$A \quad = \quad A_1 \times A_2 \times (Q_3 \setminus A_3) \quad \cup \quad A_1 \times (Q_2 \setminus A_2) \times A_3 \quad \cup \quad A_1 \times A_2 \times (Q_3 \setminus A_3)$$

As for correctness, it follows readily from the product construction above. We need only to argue that M accepts the right language. Consider a string $w \in L_1 \cap L_2 \cap \overline{L_2}$, this happens if and only if

$$\delta_1(s_1, w) \in A_1, \delta_2(s_2, w) \in A_2, \text{ and } \delta_3(s_3, w) \notin A_3.$$

In such a case, we have that $\delta^*((s_1, s_2, s_3), w) \in A_1 \times A_2 \times (Q_3 \setminus A_3) \subseteq A$, as desired. The other cases follow by a similar argument.

One also need to prove the other direction. So, consider a word $w \in L(M)$. This implies that $\delta^*(s, w) \in A$. Which in turn implies that

$$\begin{aligned} \delta^*(s, w) \in A_1 \times A_2 \times (Q_3 \setminus A_3) \quad & \bigvee \quad \delta^*(s, w) \in A_1 \times (Q_2 \setminus A_2) \times A_3 \\ & \bigvee \quad \delta^*(s, w) \in A_1 \times A_2 \times (Q_3 \setminus A_3). \end{aligned}$$

So consider the case $\delta^*(s, w) \in A_1 \times A_2 \times (Q_3 \setminus A_3)$ (the other cases are handled similarly). This implies that

$$\delta_1^*(s_1, w) \in A_1, \qquad \delta_2^*(s_2, w) \in A_2, \qquad \text{and} \qquad \delta_3^*(s_3, w) \in Q_3 \setminus A_3.$$

This implies that $w \in L_1$, $w \in L_2$, and $w \notin L_3$, which implies that $w \in L_1 \cap L_2 \cap \overline{L_3}$. Which in turn implies that $w \in L$, as desired.

6.D. (30 PTS.) You are given a DFA $M = (Q, \Sigma, \delta, s, A)$, for $\Sigma = \{0, 1\}$. Describe in detail how to build a DFA that accepts the language

$$L = \left\{ w \in \Sigma^* \;\middle|\; w \notin L(M), \overline{w} \in L(M) \text{ and } 1^{|w|} \in L(M) \right\}.$$

How many states does your DFA has as a function of $n = |Q|$? Argue that the DFA you constructed indeed accepts the specified language.

Here, for $w = w_1 w_2 \ldots w_m \in \Sigma^*$, the *complement string* $\overline{w}$ is $\overline{w_1}\,\overline{w_2}\,\overline{w_3} \ldots \overline{w_m}$, where $\overline{0} = 1$, and $\overline{1} = 0$.

Solution:

Let M_1 be the DFA that accepts the language $L_1 = \overline{L(M)}$. $M_1 = (Q, \Sigma, \delta, s, Q \setminus A)$.

Let M_2 be the DFA that accepts the language $L_2 = \{\overline{w} \in \Sigma^* \mid w \in L(M)\}$. Let $\delta_2(q, c) = \delta(q, \overline{c})$, and let $M_2 = (Q, \Sigma, \delta_2, s, A)$.

Let M_3 be the DFA that accepts the language $L_3 = \left\{ w \in \Sigma^* \;\middle|\; 1^{|w|} \in L(M) \right\}$. Let $\delta_3(q, c) = \delta(q, 1)$, and let $M_3 = (Q, \Sigma, \delta_3, s, A)$.

The desired language is $L_1 \cap L_2 \cap L_3$. We can build a DFA for this language by doing the product construction on M_1, M_2, M_3, as done in the first part of this question, and this results in a DFA with $|Q|^3 = n^3$ states. Further the correctness of the construction be proven through induction as shown in the first part of this question.