COMP 3803 — Solutions Assignment 2

Question 1: Give regular expressions describing the following languages. In all cases, the alphabet is $\{0, 1\}$. Justify your answers.

- $\{w : w$ contains an even number of 0s and each 0 is followed by at least one $1\}$.
- $\{w : w$ contains exactly two 0s and at least two 1s$\}$.
- $\{w :$ every odd position in w is $1\}$.

Solution: First we do

$\{w : w$ contains an even number of 0s and each 0 is followed by at least one $1\}$.

Each string in this language is

- empty or
-
  - starts with any number of 1's
  - followed by any number of the following: one 0 followed by at least one 1, one 0 followed by at least one 1.

This gives the regular expression

$$1^* \left(011^*011^*\right)^*$$

Next we do

$\{w : w$ contains exactly two 0s and at least two 1s$\}$.

Consider an arbitrary string in this language. It contains exactly two 0's; these 0's split the string into three pieces (left, middle, and right), where each piece is empty or contains only 1's. Since there must be at least two 1's, there are six possibilities:

- the left piece contains at least two 1's: $111^*01^*01^*$.
- the middle piece contains at least two 1's: $1^*0111^*01^*$.
- the right piece contains at least two 1's: $1^*01^*0111^*$.
- the left piece contains at least one 1 and the middle piece contains at least one 1: $11^*011^*01^*$.
- the left piece contains at least one 1 and the right piece contains at least one 1: $11^*01^*011^*$.

- the middle piece contains at least one 1 and the right piece contains at least one 1: $1^*011^*011^*$.

This leads to the regular expression:

$$111^*01^*01^* \cup 1^*0111^*01^* \cup 1^*01^*0111^* \cup 11^*011^*01^* \cup 11^*01^*011^* \cup 1^*011^*011^*$$

Finally, we do

$$\{w : \text{ every odd position in } w \text{ is } 1\}.$$

Each string in this language is empty or has the following form: one 1, followed by one bit, followed by one 1, followed by one bit, followed by one 1, etc. Keep in mind that the length of the string can be odd or even. Here is the regular expression:

$$(1(0 \cup 1))^* \cup (1(0 \cup 1))^* 1$$

You can also write this as

$$(1(0 \cup 1))^* (\epsilon \cup 1)$$

Question 2: Use the construction given in class to convert the regular expression

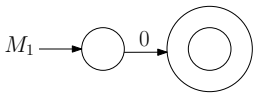
$$((0 \cup 1)(11)^* \cup 0)^*$$

to an NFA. The alphabet is $\{0, 1\}$.

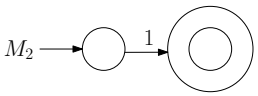
Solution: We first consider how the regular expression is "built":

- Take the regular expressions 0 and 1, and combine them into the regular expression $0 \cup 1$.
- Take the regular expressions 1 and 1, and combine them into the regular expression 11.
- Take the regular expression 11, and turn it into the regular expression $(11)^*$.
- Take the regular expressions $0 \cup 1$ and $(11)^*$, and combine them into the regular expression $(0 \cup 1)(11)^*$.
- Take the regular expressions $(0 \cup 1)(11)^*$ and 0, and combine them into the regular expression $(0 \cup 1)(11)^* \cup 0$.
- Take the regular expression $(0 \cup 1)(11)^* \cup 0$, and turn it into the regular expression $((0 \cup 1)(11)^* \cup 0)^*$.

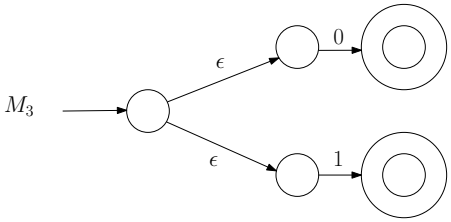
First, we construct an NFA M_1 that accepts the language described by the regular expression 0:



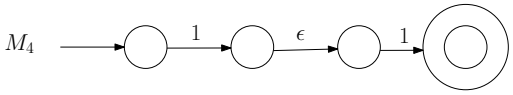
Next, we construct an NFA M_2 that accepts the language described by the regular expression 1:



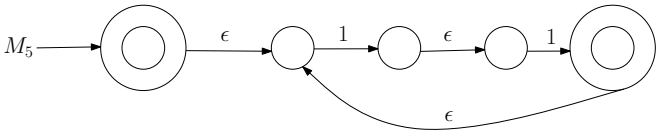
Next, we apply the union construction to M_1 and M_2. This gives an NFA M_3 that accepts the language described by the regular expression $0 \cup 1$:



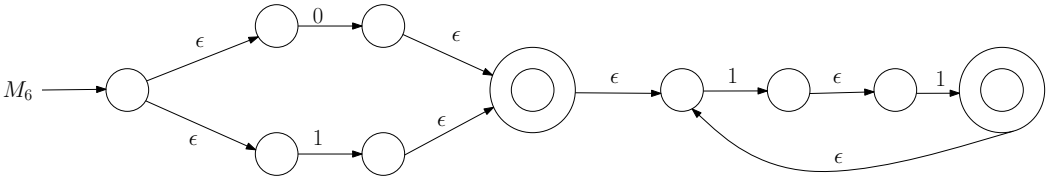
Next, we apply the concatenation construction to M_2 and M_2. This gives an NFA M_4 that accepts the language described by the regular expression 11:



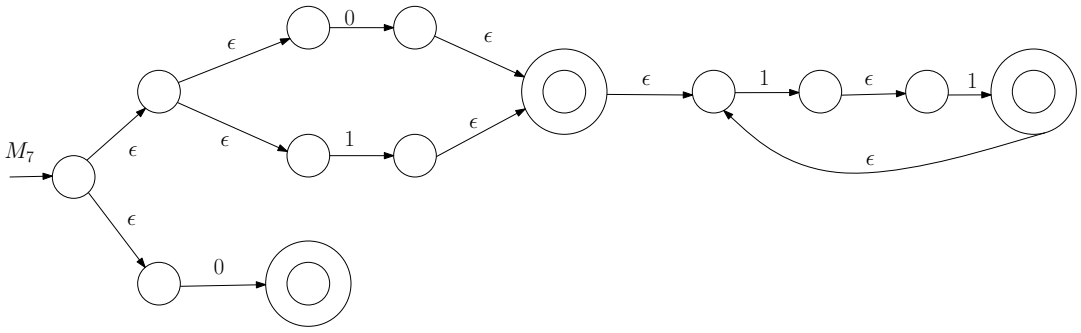
Next, we apply the star construction to M_4. This gives an NFA M_5 that accepts the language described by the regular expression $(11)^*$:



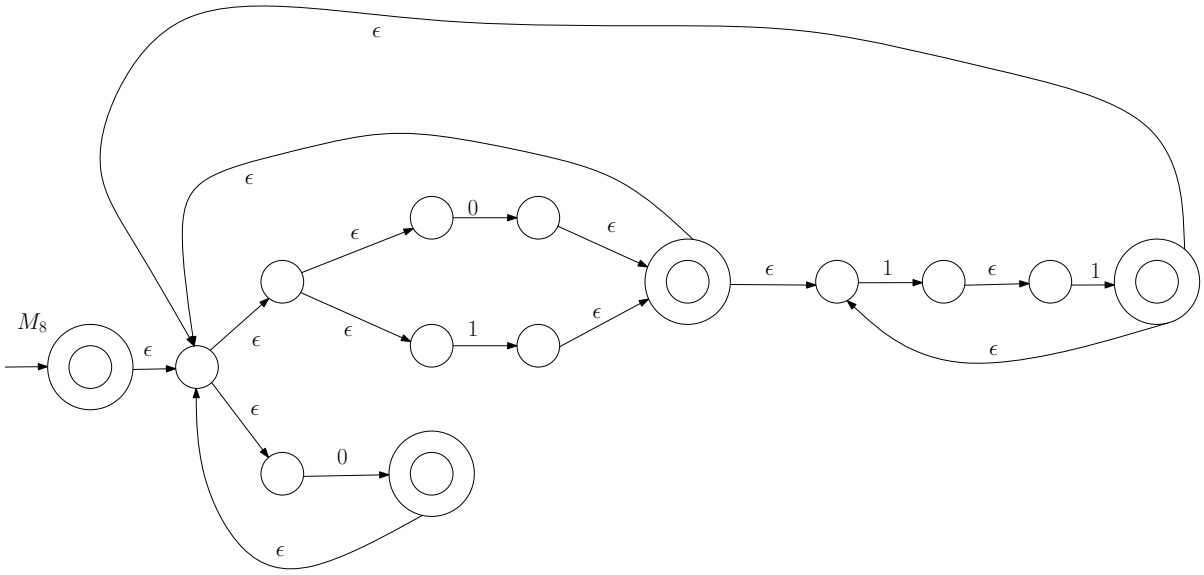
Next, we apply the concatenation construction to M_3 and M_5. This gives an NFA M_6 that accepts the language described by the regular expression $(0 \cup 1)(11)^*$:



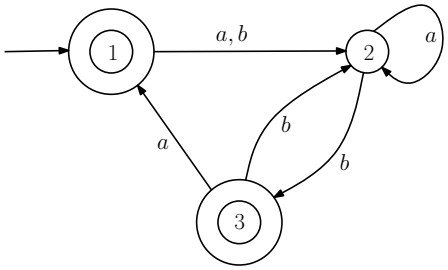
Next, we apply the union construction to M_6 and M_1. This gives an NFA M_7 that accepts the language described by the regular expression $(0 \cup 1)(11)^* \cup 0$:



Finally, we apply the star construction to M_7. This gives an NFA M_8 that accepts the language described by the regular expression $((0 \cup 1)(11)^* \cup 0)^*$:



Question 3: Use the construction given in class to convert the following DFA to a regular expression.



Solution: For each state $i = 1, 2, 3$, we define L_i to be the set of all strings w in $\{a, b\}^*$ such that the path in the state diagram that starts in state i and corresponds to w ends in one

of the two accept states. We obtain the following set of equations:

$$\begin{aligned} L_1 &= \epsilon \cup (a \cup b) L_2 && (1) \\ L_2 &= a L_2 \cup b L_3 && (2) \\ L_3 &= \epsilon \cup a L_1 \cup b L_2 && (3) \end{aligned}$$

Since 1 is the start state, we need a regular expression for L_1.

We use the following tool to solve these equations:

$$\text{If } L = BL \cup C \text{ and } \epsilon \notin B, \text{ then } L = B^* C. \quad (4)$$

We solve the equations (1), (2), and (3), in the following way: By substituting (3) into (2), we obtain

$$L_2 = a L_2 \cup b \left(\epsilon \cup a L_1 \cup b L_2 \right),$$

which we rewrite as

$$L_2 = a L_2 \cup b \cup b a L_1 \cup b b L_2,$$

which we rewrite as

$$L_2 = (a \cup bb) L_2 \cup (b \cup ba L_1). \quad (5)$$

This equation is in the form of (4), with $L = L_2$, $B = a \cup bb$, and $C = b \cup ba L_1$. Since ϵ is not in the language described by B, we can apply (4) to (5), and we obtain

$$L_2 = (a \cup bb)^* \left(b \cup ba L_1 \right),$$

which we rewrite as

$$L_2 = (a \cup bb)^* b \cup (a \cup bb)^* ba L_1. \quad (6)$$

By substituting (6) into (1), we obtain

$$L_1 = \epsilon \cup (a \cup b) \left((a \cup bb)^* b \cup (a \cup bb)^* ba L_1 \right),$$

which we rewrite as

$$L_1 = \epsilon \cup (a \cup b) (a \cup bb)^* b \cup (a \cup b) (a \cup bb)^* ba L_1,$$

which we rewrite as

$$L_1 = (a \cup b) (a \cup bb)^* ba L_1 \cup (\epsilon \cup (a \cup b) (a \cup bb)^* b). \quad (7)$$

This equation is in the form of (4), with $L = L_1$,

$$B = (a \cup b) (a \cup bb)^* ba,$$

and

$$C = \epsilon \cup (a \cup b) (a \cup bb)^* b.$$

Since ϵ is not in the language described by B, we can apply (4) to (7), and we obtain

$$L_1 = ((a \cup b)(a \cup bb)^* ba)^* (\epsilon \cup (a \cup b)(a \cup bb)^* b).$$

Question 4: Let R be a regular expression and let A be the language described by R. Explain how to obtain a regular expression that describes the complement $\overline{A}$ of A. You may use any result that was proven in class and Assignment 1.

Solution:

- Use the construction given in class to convert the regular expression R to an NFA M whose language is A.
- Use the construction given in class to convert the NFA M to a DFA whose language is A.
- Use the construction given in Assignment 1 to convert the DFA M to a DFA $\overline{M}$ whose language is the complement $\overline{A}$ of A.
- Use the construction given in class to convert the DFA $\overline{M}$ to a regular expression $\overline{R}$ that describes the language $\overline{A}$.

Question 5: Prove that the following languages are not regular.

1. $\{a^n b^n c^{2n} : n \geq 0\}$.
2. $\{a^{3^n} : n \geq 0\}$. (Remark: a^{3^n} is the string consisting of 3^n many a's.)
3. $\{uvu : u \in \{a,b\}^*, u \neq \epsilon, v \in \{a,b\}^*\}$.
4. $\{a^m b^n : m \geq 0, n \geq 0, m \neq n\}$. (Using the Pumping Lemma for this one is a bit tricky. You can avoid using the Pumping Lemma by combining results about the closure under regular operations.)

Solution: First, we do

$$A = \{a^n b^n c^{2n} : n \geq 0\}.$$

Assume the language A is regular. Let $p \geq 1$ be the pumping length, as given by the Pumping Lemma. Let $s = a^p b^p c^{2p}$. Then $s \in A$ and $|s| = 4p \geq p$. Hence, by the Pumping Lemma, we can write $s = xyz$, where

1. $y \neq \epsilon$,
2. $|xy| \leq p$, and
3. $xy^i z \in A$, for all $i \geq 0$.

Since $|xy| \leq p$, the string y contains only as. Since $y \neq \epsilon$, the string y contains at least one a. Hence, the string $xy^2z = xyyz$ contains more than p many as and exactly p many bs. But this means that xy^2z is not in the language A. This is a contradiction, because, by the Pumping Lemma, this string is an element of A. So we have a contradiction, and we can conclude that A is not regular.

Next we do

$$B = \{a^{3^n} : n \geq 0\}.$$

Assume the language B is regular. Let $p \geq 1$ be the pumping length, as given by the Pumping Lemma. Let $s = a^{3^p}$. Then $s \in B$ and $|s| = 3^p \geq p$. Hence, by the pumping lemma, we can write $s = xyz$, where

1. $y \neq \epsilon$,
2. $|xy| \leq p$, and
3. $xy^iz \in B$, for all $i \geq 0$.

Since $y \neq \epsilon$ and $|xy| \leq p$, the string y has the form $y = a^k$, for some integer k with $1 \leq k \leq p$. Consider the string

$$xy^2z = xyyz = a^{3^p+k}.$$

The length of this string is equal to $3^p + k$.

- Since $k \geq 1$, we have $3^p + k > 3^p$.
- Since $k \leq p$, we have $3^p + k \leq 3^p + p < 3^p + 3^p = 2 \cdot 3^p < 3^{p+1}$.

Hence,

$$3^p < |xy^2z| < 3^{p+1},$$

that is, the length of the string xy^2z is strictly between two consecutive powers of 3. But this means that xy^2z is not in the language B. This is a contradiction, because, by the Pumping Lemma, this string is an element of B. So we have a contradiction, and we can conclude that B is not regular.

Next we do

$$C = \{uvu : u \in \{a, b\}^*, u \neq \epsilon, v \in \{a, b\}^*\}.$$

Assume the language C is regular. Let $p \geq 1$ be the pumping length, as given by the Pumping Lemma. Let $s = a^pba^pb$. Then $s \in C$: If we take $u = a^pb$ and $v = \epsilon$, then $s = uvu$. Also, we have $|s| = 2p + 2 \geq p$. Hence, by the Pumping Lemma, we can write $s = xyz$, where

1. $y \neq \epsilon$,
2. $|xy| \leq p$, and
3. $xy^iz \in C$, for all $i \geq 0$.

Since $|xy| \leq p$, the string y is contained in the leftmost a-block of the string s. We also know that y is non-empty. Consider the string $xyyz$. This string starts with an a-block consisting of more than p many as, followed by one b, followed by an a-block consisting of exactly p many as, followed by one b. This string cannot be written as uvu with $u \neq \epsilon$. (Observe: If u is allowed to be ϵ, then $xyyz$ can be written as uvu.) Hence, $xyyz$ is not an element of C. This is a contradiction, because, by the Pumping Lemma, this string is an element of C. So we have a contradiction, and we can conclude that C is not regular.

Finally we do

$$D = \{a^m b^n : m \geq 0, n \geq 0, m \neq n\}.$$

First a solution that does not use the Pumping Lemma. Assume that D is regular. Then from Assignment 1, the complement $\overline{D}$ of D is also regular. Observe that

$$\overline{D} \cap \{a^m b^n : m \geq 0, n \geq 0\} = \{a^n b^n : n \geq 0\}.$$

The language $\{a^m b^n : m \geq 0, n \geq 0\}$ is regular, because it is described by the regular expression a^*b^*. Thus, we know that $\{a^n b^n : n \geq 0\}$ is the intersection of two regular languages.

In class, we have seen that the union of two regular languages is regular. In Assignment 1, we have seen that the complement of a regular language is regular. This, together with De Morgan, implies that the intersection of two regular languages is regular.

It follows that $\{a^n b^n : n \geq 0\}$ is a regular language. But we have seen in class that this language is not regular. We have a contradiction and conclude that D is not regular.

Here is a solution that uses the Pumping Lemma. Assume the language D is regular. Let $p \geq 1$ be the pumping length, as given by the Pumping Lemma. Let $s = a^p b^{p+p!}$. Then $s \in B$ and $|s| = 2p + p! \geq p$. Hence, by the pumping lemma, we can write $s = xyz$, where

1. $y \neq \epsilon$,
2. $|xy| \leq p$, and
3. $xy^i z \in D$, for all $i \geq 0$.

Since $|xy| \leq p$, the string y is contained in the a-block of the string s. We also know that y is non-empty. Thus, we can write y as $y = a^k$, for some integer k with $1 \leq k \leq p$. (Observe that k can be any integer in this range!)

Let $i = 1 + p!/k$. Then i is a positive integer. Consider the string $xy^i z$. This string is given by

$$xy^i z = a^{p+(i-1)k} b^{p+p!}.$$

Since $(i-1)k = p!$, we have

$$xy^i z = a^{p+p!} b^{p+p!}.$$

Hence, $xy^i z$ is not an element of D. This is a contradiction, because, by the Pumping Lemma, this string is an element of D. So we have a contradiction, and we can conclude that D is not regular.

Question 6: Let A be a language consisting of finitely many strings.

1. Prove that A is a regular language.

2. Let n be the maximum length of any string in A. Prove that *every* deterministic finite automaton (DFA) that accepts A has at least $n+1$ states. (*Hint:* How is the pumping length chosen in the proof of the Pumping Lemma?)

Solution: We prove that A is regular. We first observe that a language containing one single string is regular: It is easy to come up with an NFA that accepts only this single string. Alternatively, one single string is a regular expression; hence, by Theorem 2.8.1 on page 56, it is regular.

Since A is finite, it is the finite union of languages, each one containing one single string. Since any finite union of regular languages is regular, it follows that A is regular.

Now the second part of the question: Let n be the maximum length of any string in A. We have to prove that every deterministic finite automaton (DFA) that accepts A has at least $n + 1$ states.

Here is the proof, which is by contradiction. We assume that there is a DFA M that accepts A and that has p states, where $p \leq n$.

Now look at the proof of the Pumping Lemma. For the pumping length, we can take the number of states of M. Hence, we can take the pumping length to be equal to p.

Let s be the longest string in A. Then the length of s is n, which is at least p. Hence, the length of s is at least the pumping length. The Pumping Lemma tells us that we can write $s = xyz$, where (i) $y \neq \epsilon$ and (ii) $xy^iz \in A$ for all $i \geq 0$. Since the strings xy^iz, for all $i \geq 0$, are distinct (because their lengths are distinct), it follows that A contains infinitely many strings. This is a contradiction, because we assumed that A is finite. So our assumption that our DFA has at most n states cannot be true. Therefore, it must have at least $n + 1$ states.