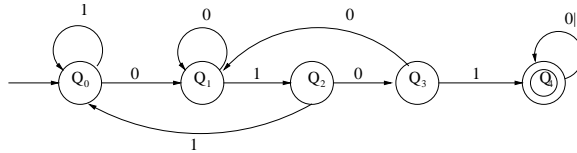


CS530
Partial Solutions for HW1

1. **Designing DFAs and NFAs.** For each of the following, draw a state diagram for a DFA or NFA (as required) that recognizes the specified language. In all cases the alphabet is $\{0, 1\}$.

- (a) Sipser exercise 1.6, part c. $L_1 = \{w \mid w \text{ contains the substring } 0101\}$. Provide a DFA recognizing L_1 . Notice that you are required to write DFA, not NFA here.

Solution:



Q_0 – start state

Q_1 – first 0 in the substring 0101

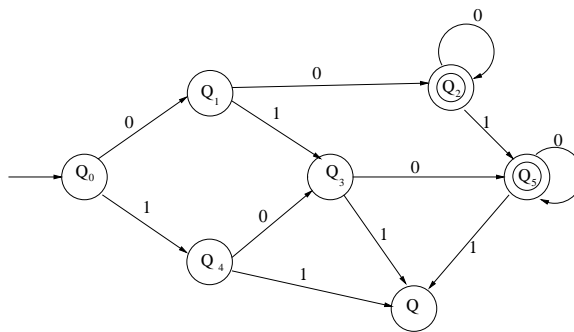
Q_2 – second 1 in the substring 0101

Q_3 – third 0 in the substring 0101

Q_4 – fourth 1 in the substring 0101 (final state)

- (b) Sipser exercise 1.6, part j. $L_2 = \{w \mid w \text{ contains at least two } 0\text{s and at most one } 1\}$. Provide a DFA recognizing L_2 . Notice that you are required to write DFA, not NFA here.

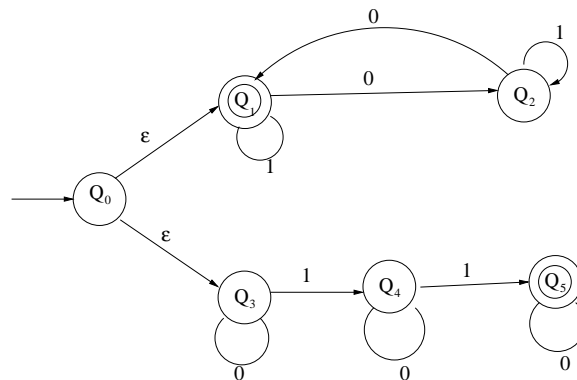
Solution:



Q_0 – start state

- Q_1 – one 0 without 1
- Q_2 – two or more 0s without 1
- Q_3 – one 0 and one 1
- Q_4 – one 1 without 0
- Q_5 – two or more 0s with one 1 (final state)
- Q – stuck state

- (c) Sipser exercise 1.7, part c. $L_3 = \{w \mid w \text{ contains an even number of 0s, or exactly two 1s}\}$. Provide an NFA with at most six states that recognizes L_3 . Notice that you can use NFA here. **Solution:**



- Q_0 – start state
- Q_1 – even 0 (final state)
- Q_2 – odd 0
- Q_3 – zero 1
- Q_4 – one 1
- Q_5 – two 1 (final state)

2. **Proving an FA recognizes a language.** For one of the automata you designed in problem 1, prove that the machine recognizes *exactly* the specified language. To do this, you will need to prove that your automaton (1) accepts all strings in the language and (2) does not accept any string not in the language.

Solution: Take 1 (b) for the proof.

- (1) The DFA accepts all the strings belonging to L_2 .

Because that L_2 contains all the strings which contains at least two 0s and at most one 1, it only contains FOUR kinds of strings, 00^+ , 00^+1

$10^*, 010^+, 100^+$. (Please NOTE that 0^+ means there are at least ONE 0s, and 0^* means there are at least ZERO 0s). The final states, Q_2 and Q_5 correspond to these strings; $Q_2 : 00^+, Q_5 : 00 + 10^*, 010^+, 100^+$. Now the only thing I need to do is to prove that there are only four kinds of strings, mentioned above, belong to L_2 .

Lemma 0.1 *There are only four kinds of strings belong to L_2 .*

Proof: There are ONLY two general cases; L_2 contains only ONE 1 or contains NO 1. 1) If L_2 contains NO 1, there should be no less than TWO 0s in L_2 . 00^+ belongs to this case 2) If L_2 contains ONE 1, there are ONLY three sub-cases: there is no 0 ahead of 1, there is only one 0 ahead of 1, and there are two or more 0s before 1. $100^+, 010^+,$ and $00 + 10^*$ satisfy this case. Due to 1) and 2), there are only four kinds of strings belong to L_2 as mentioned above.

This ends the proof. □

(2) Any other strings can not accepted by this DFA.

We note this DFA by M.

1) M can not accept the strings containing more than ONE 1s.

For each path to each final state, there is at most ONE transition step with '1'. So M can not accept the strings containing more than ONE 1s.

2) M can not accept the string containing less than TWO 0s.

For each path to the final state, there is at least TWO transition steps with 0. So for a string with only one 0 or zero 0, it can never reach the final state. Which means M can not accept the strings containing less than TWO 0s.

Considering (1) and (2), we finish the proof.

NOTE: This problem can also be proved by comparing if the regular expressions deduced from the DFA and the language are the same.

3. Given two languages A, B that are regular, show that the following languages are regular

$$(a) A^{\dagger}B = \{a_1b_1a_2b_2a_3b_3 \cdots a_nb_n \mid n \geq 1, a_1a_2a_3 \cdots a_n \in A, \text{ and } b_1b_2b_3 \cdots b_n \in B\}$$

Proof: We construct A' as following, $a_1 \Sigma a_2 \Sigma a_3 \Sigma \cdots a_n \Sigma$, (Σ is any character of the alphabet). If M_1 is the NFA who can accept A' , then M_1 is derived from the original NFA which accepts A in the following form: For every transition step, we add a new state, for example

$$\delta(A_i, a_i) = A_j \implies \delta(A_i, a_i) = A'_i \text{ and } \delta(A'_i, \Sigma) = A_j.$$

And for every final state, we let it be non-final state and add a transition step from each original final state to a new final state.

We construct B' as following, $\Sigma b_1 \Sigma b_2 \Sigma b_3 \cdots \Sigma b_n$. If M_2 is the NFA who can accept B' , then M_2 can be constructed as follow using the original NFA which is corresponding to B : For every transition step, we add a new state, for example

$$\delta(B_i, b_i) = B_j \implies \delta(B_i, \Sigma) = B'_i \text{ and } \delta(B'_i, b_i) = B_j.$$

We add a new start state to transit to the original start state. It thus can prove that A' and B' are regular. $M = M_1 \cap M_2$ can be represented by a NFA since M_1 and M_2 can be represented by NFAs, which is $A \dagger B$. That is, $A \dagger B$ is regular.

This finishes the proof. \square

- (b) $A \# B = \{a_1 b_1 a_2 b_2 a_3 b_3 \cdots a_n b_n \mid n \geq 1, a_1 a_2 a_3 \cdots a_n \in A, \text{ or } b_1 b_2 b_3 \cdots b_n \in B\}$

Proof: First we construct A' and B' as above. NFA M_1 represents A' and NFA M_2 represents B' . We know that $M = M_1 \cup M_2$ is also a NFA, which is equivalent to $A \# B$. So $A \# B$ is also regular.

This finishes the proof \square

- (c) $A \ddagger B = \{a_1 a_2 b_1 a_3 a_4 b_2 a_5 a_6 b_3 \cdots a_{2n-1} a_{2n} b_n \mid n \geq 1, a_1 a_2 a_3 \cdots a_{2n-1} a_{2n} \in A, \text{ and } b_1 b_2 b_3 \cdots b_n \in B\}$

Proof: This question is very similar to subquestion 1 and 2. The only difference we should do is to prove the language $A' = \{a_1 a_2 \Sigma a_3 a_4 \Sigma a_5 a_6 \Sigma \cdots a_{2n-1} a_{2n} \Sigma \mid n \geq 1, a_1 a_2 a_3 \cdots a_{2n-1} a_{2n} \in A\}$ is regular, or we can find a NFA M_1 which represents the language A' . The steps to construct M_1 are as following:

- i. Do Cartesian Production above the set of the states in A . Then we get a set $C = (A_i, A_j) \mid A_i \in A \text{ and } A_j \in A$, which is the state set of M'_1 ;

- ii. For every adjacent two transitions in A , $\delta(A_i, a_i) = A_j$ and $\delta(A_j, a_j) = A_k$, we add a transition in M'_1 as following: $\delta(A_i A_j, a_i a_j) = (a_k, *)$. (Note that $(a_k, *)$ denotes that every state in M'_1 whose first input character is a_k);
- iii. If a_i belongs to the final state of the NFA corresponding to A , then make $(a_i, *)$ and $(*, a_i)$ to be the final states in M_1 ;
- iv. We deduct M'_1 to M_1'' , and denote all the states in M_1'' as C_1, C_2, \dots, C_n ;
- v. We add a new state, for example $\delta(C_i, c_i) = C_j \implies \delta(C_i, c_i) = C'_i$ and $\delta(C'_i, \Sigma) = C_j$.
- vi. For every final state, we let it be non-final state and add an transition step from each original final state to a new final state.

Thus we get M_1 , which is also a NFA. $M = M_1 \cap M_2$ represents $A \dagger B$, which is regular.

This finishes the proof □

4. Assume that the alphabet is $\Sigma = \{a, b, c\}$. Let $N(w, a)$ denote the number of times that character a appeared in string w . Write regular expressions for each of the following languages

- (a) $L_1 = \{w \mid N(w, a) \text{ is odd}\}$.

Solution: $((b|c)^* a (b|c)^* a (b|c)^*)^* a (b|c)^*$

- (b) $L_2 = \{w \mid N(w, b) \text{ is even}\}$,

Solution: $((a|c)^* b (a|c)^* b (a|c)^*)^*$

- (c) $L_3 = \{w \mid N(w, a) \text{ is odd, and } N(w, b) \text{ is even}\}$,

Solution: The even number of a s and b s can be expressed as following: $L = c^* | c^* b c^* b c^* | c^* a c^* a c^* |$

$c^* b c^* b c^* a c^* a c^* | c^* b c^* a c^* b c^* a c^* | c^* a c^* b c^* b c^* a c^* |$

$c^* b c^* a c^* a c^* b c^* | c^* a c^* b c^* a c^* b c^* | c^* a c^* a c^* b c^* b c^*$

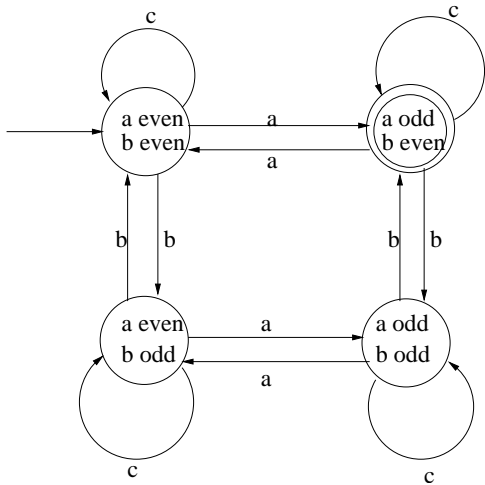
Then the regular expression is $L^* a L^* | L^* b L^* a L^* b L^*$.

NOTE: You can get the regular expression from DFAs as following.

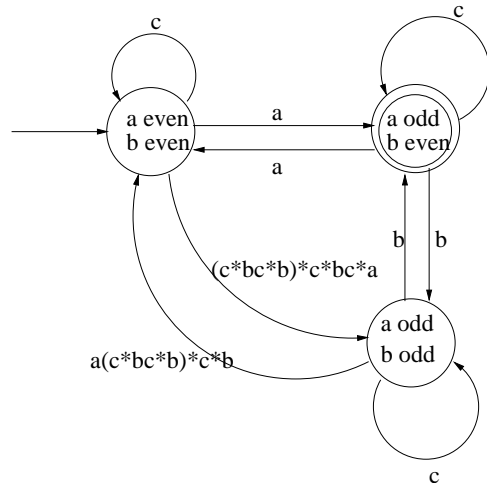
5. Describe the following language by nature language

- (a) 10^*1

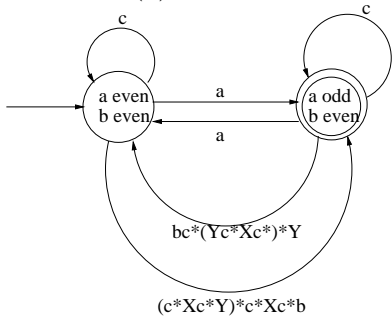
Solution: The language is a set of strings which begin and end with 1 and contains only 0s, if there is any, in other positions.



(a) DFA for the language



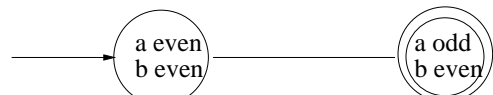
(b) remove one state



(a) remove two states

$$X = (c*bc*b)*c*bc*;$$

$$Y = a(c*bc*b)*c*b$$



$$(c*ac*a|c*ac*V|c*Uc*a|c*Uc*V)*c*(a|U)$$

$$U = (c*Xc*Y)*c*Xc*b$$

$$V = bc*(Yc*Xc)*Y$$

(b) remove three states

Figure 1: Problem 4(c): Regular expression converted from DFA.

(b) $(0|1)^*011(0|1)^*$

Solution: The language is a set of strings containing substring 011.

6. Prove that the following languages are non-regular:

(a) $L_1 = \{0^i1^j0^k : j = \max\{i, k\}\}$

Proof: Consider if L_1 is the regular language and then p is the pumping length where s is the string, $s = 0^p1^p0^p$ and such that $s \in L_1$. Then s may be divided into three parts, $s = xyz$, satisfying the following conditions of pumping lemma; 1) $xy^iz \in L_1$ for each $i \geq 0$,

2) $|y| > 0$,

3) $|xy| \leq p$.

Using the case (3), x and y will consist of 0s because they are the first symbols and therefore z will be consisting of the remaining 0s and 1^p0^p .

The case (2) says $|y| > 0$, therefore it should have at least one 0. So we can say, $x = 0^a$ for $a \geq 0$, $y = 0^b$ for $b \geq 0$ and $z = 0^c1^p0^p$ for $c \geq 0$. Applying the values of x, y and z , in $s = xyz$, then

$$s = 0^a0^b0^c1^p0^p \quad (1)$$

Using case(1) that is $xy^iz \in L_1$ for $i = 2$,

$$\begin{aligned} xy^2z &= 0^a \cdot 0^b \cdot 0^b \cdot 0^c \cdot 1^p \cdot 0^p \\ &= 0^{p+b} \cdot 1^p \cdot 0^p \text{ (where } p = a + b + c \text{)} \end{aligned}$$

Thus $xy^2z \notin L_1$, which is not satisfied case(1). Therefore L_1 is proved a non regular language by contradiction.

End of the proof. \square

(b) $L_2 = \{w | w = x\bar{x}, x \in \{0, 1\}^*, \bar{x} \text{ is the complement of } x\}$

Proof: Consider if L_2 is the regular language and then p is the pumping length where s is the string, $s = 0^p1^p$ and such that $s \in L_2$. Then s may be divided into three parts, $s = xyz$, satisfying the following conditions of pumping lemma; 1) $xy^iz \in L_2$ for each $i \geq 0$,

2) $|y| > 0$,

3) $|xy| \leq p$.

Using the case (3), x and y will consist of 0s because they are the first symbols and therefore z will be consisting of the remaining 0s and 1^p .

The case (2) says $|y| > 0$, therefore it should have at least one 0. So we can say, $x = 0^a$ for $a \geq 0$, $y = 0^b$ for $b \geq 0$ and $z = 0^c 1^p$ for $c \geq 0$. Applying the values of x, y and z , in $s = xyz$, then

$$s = 0^a 0^b 0^c 1^p \quad (2)$$

Using case(1) that is $xy^i z \in L_2$ for $i = 2$,

$$\begin{aligned} xy^2 z &= 0^a \cdot 0^b \cdot 0^b \cdot 0^c \cdot 1^p \\ &= 0^{p+b} \cdot 1^p \text{ (where } p = a + b + c) \end{aligned}$$

Thus $xy^2 z \notin L_2$, which is not satisfied case(1). Therefore L_2 is proved a non regular language by contradiction.

End of the proof. \square

- (c) Palindrome $L_p = \{w \mid w \text{ is a palindrome}\}$. Assume that the alphabet is $\Sigma = \{0, 1\}$. A palindrome is a binary string which is equivalent to its reversal. For example, 1, 0110 and 010111010 are palindromes, but 011 is not.

Proof: Consider if L_p is the regular language and then p is the pumping length where s is the string, $s = 0^p 10^p$ and such that $s \in L_p$. Then s may be divided into three parts, $s = xyz$, satisfying the following conditions of pumping lemma; 1) $xy^i z \in L_p$ for each $i \geq 0$,

2) $|y| > 0$,

3) $|xy| \leq p$.

Using the case (3), x and y will consist of 0s because they are the first symbols and therefore z will be consisting of the remaining 0s and the 1 and all right side 0s.

The case (2) says $|y| > 0$, therefore it should have at least one 0. So we can say, $x = 0^a$ for $a \geq 0$, $y = 0^b$ for $b \geq 0$ and $z = 0^c 10^p$ for $c \geq 0$. Applying the values of x, y and z , in $s = xyz$, then

$$s = 0^a 0^b 0^c 10^p \quad (3)$$

Using case(1) that is $xy^i z \in L_p$ for $i = 2$,

$$\begin{aligned} xy^2 z &= 0^a \cdot 0^b \cdot 0^b \cdot 0^c \cdot 10^p \\ &= 0^{p+b} \cdot 10^p \text{ (where } p = a + b + c) \end{aligned}$$

Thus $xy^2z \notin L_p$, which is not satisfied case(1). Therefore L_p is proved a non regular language by contradiction.

End of the proof.

□