6.045J/18.400J: Automata, Computability and Complexity Prof. Nancy Lynch

Homework 2.5 (FAKE): Solutions

Due: never *Vinod Vaikuntanathan*

This "fake" homework is intended as a study guide covering the material on lectures 4 and 5.

Problem 1: (Taken from Sipser 1.18.) Give regular expressions generating the following languages. In all cases the alphabet is $\{0, 1\}$.

1. $L_1 = \{w|w \text{ contains the substring } 0101\}$.
2. $L_2 = \{w \mid w \text{ does not contain } 100 \text{ as a substring}\}$.
3. $L_3 = \{w|w \text{ starts with 0 and has odd length, or starts with 1 and has even length }\}$.
4. $L_4 = \{w| \text{ the length of } w \text{ is at most } 5\}$.
5. $L_5 = \{w| \text{ contains at least one 0 and at most one 1 }\}$.
6. $L_6 = \{w|w \neq \epsilon\}$.

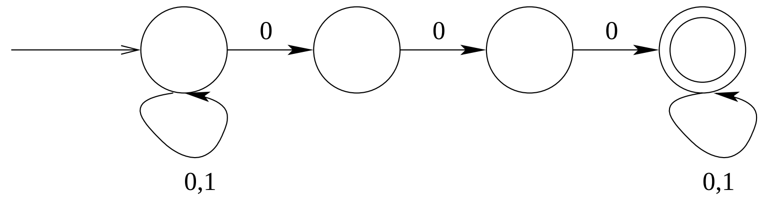
Solution 1:

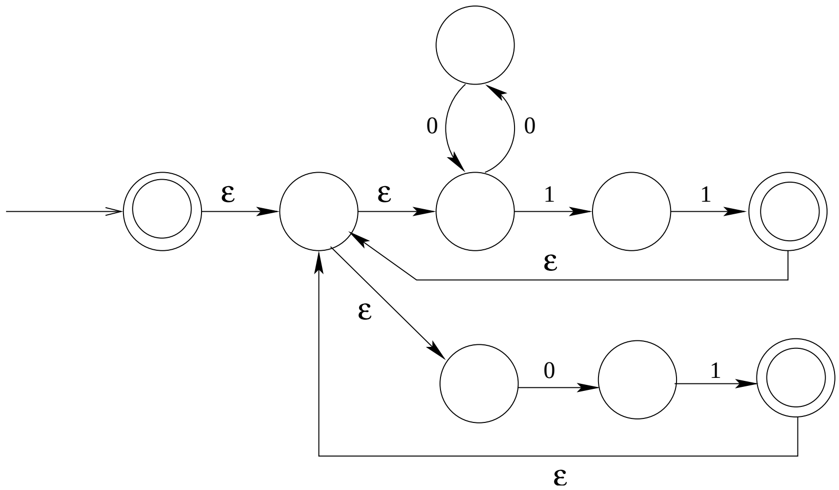
1. L_1 : $(0 \cup 1)^*0101(0 \cup 1)^*$.
2. L_2 : $0^*1^*(01 \cup 1)^*(0 \cup \epsilon)$.
3. L_3 : $(0 \cup 1(0 \cup 1))((0 \cup 1)(0 \cup 1))^*$.
4. L_4 : $(0 \cup 1 \cup \epsilon)(0 \cup 1 \cup \epsilon)(0 \cup 1 \cup \epsilon)(0 \cup 1 \cup \epsilon)(0 \cup 1 \cup \epsilon)$.
5. L_5 : $00^*10^* \cup 0^*100^* \cup 00^*$. Could also be written (slightly more compactly) as $0^*(10 \cup 01 \cup 0)0^*$.
6. L_6 : $(0 \cup 1)(0 \cup 1)^*$

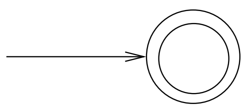
Problem 2: (Sipser 1.19.) Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

1. $(0 \cup 1)^*000(0 \cup 1)^*$
2. $(((00)^*(11)) \cup 01)^*$
3. $\emptyset^*$

Solution 2:

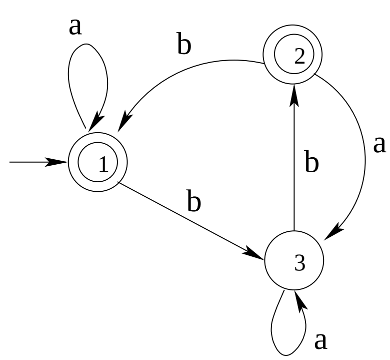
1. 

2. 

3. 

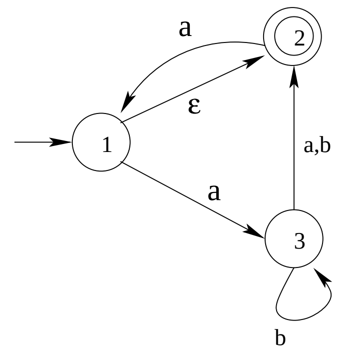
Problem 3: Convert the following finite automata to equivalent regular expressions:

1. The DFA depicted in the following diagram. Use the procedure described in Lemma 1.60.



Solution 3: $\{a \cup (ba^*b)(aa^*b)^*b\}^*\{ba^*b(aa^*b)^* \cup \epsilon\}$.

2. The NFA depicted in the following diagram.



Solution 3: $\{ab^*(a\cup b)\cup\epsilon\}\Big\{a\{ab^*(a\cup b)\cup\epsilon\}\Big\}^*$.

Problem 4: Use the pumping lemma to show that the following languages are not regular.

(a) $A_1 = \{0^a1^b2^c \mid 0 \le a \le b \le c\}$.

(b) (From Sipser 1.29.) $A_2 = \{a^{2^n} \mid n \ge 0\}$. (Here, a^{2^n} means a string of 2^n a's.)

(c) $A_3 = \{0^{n^2} \mid n \ge 0\}$.

(d) Do you see something in common between the arguments used to answer parts (b) and (c) ? Generalize the arguments of parts (b) and (c) to show that for any function $f : \mathbb{N} \to \mathbb{N}$ which obeys the inequality $f(n+1) - f(n) > n$, the language $A_4 = \{0^{f(n)} \mid n \ge 0\}$ is not regular.

Solution 4:

(a) Let p be the pumping length guaranteed by the pumping lemma. Choose the string $0^p1^p2^p$. Let this be arbitrarily divided into xyz. By Condition 3 of the pumping lemma (refer Sipser Lemma 1.70), $|xy| \le p$ and $|y| > 0$. This means that y is a non-empty string that consists of zeros only. (Why ? Because the length of xy is at most p and the length of the initial strip of 0's in $0^p1^p2^p$ is p.) Now, pump it up: xy^2z must have at least $p+1$ 0's, but exactly p 1's. Therefore it is not in A_1.

(b) Let p be the pumping length, again. Choose the string a^{2^p}. Let this be arbitrarily divided into xyz. By Condition 3 of the pumping lemma (refer Sipser Lemma 1.70), $|xy| \le p$ and $|y| > 0$. Therefore $2^p < |xy^2z| \le 2^p + p < 2^{p+1}$ (Use Fact 1). Therefore, the string xy^2z consists of a number of a's that is not a power of 2, and is not in A_2.

Fact 1 *For all* $p \ge 0$, $2^p > p$.

(c) Let $p > 0$ be the pumping length. Choose the string 0^{p^2}. Let it be divided into xyz. As before, $0 < |y| \le p$. Therefore, the length of xy^2z is $p^2 + |y|$. But we know that $p^2 < p^2 + |y| \le p^2 + p < (p+1)^2$. Therefore, the string xy^2z is not in A_3. And we are done.

(d) The generalization is rather straightforward. Let p be the pumping length. Choose the string $0^{f(p)}$. Let it be divided into xyz. As before, $0 < |y| \le p$. Therefore, the length of xy^2z is $f(p) + |y|$. But we know that $p^2 < p^2 + |y| \le f(p) + p < f(p+1)$ (The last inequality is due to the constraint on f). Therefore, the string xy^2z is not in A_4.

Problem 5: (Sipser 1.46.) Prove that the following languages are not regular. You may use the pumping lemma and the closure of the class of regular languages under union, intersection and complement.

(a) $\{0^n 1^m 0^n \mid m, n \geq 0\}$

(b) $\{w \mid w \in \{0,1\}^* \text{ is not a palindrome}\}$.[1]

Solution 5:

(a) One way to show this is to directly use pumping lemma. Let p be the pumping length. Consider the string $0^p 1 0^p$. Suppose it is written as xyz where $|xy| \leq p$ and $|y| > 0$. In particular, consider $xy^2z = 0^q 1 0^p$, where $q > p$ – this string is not in the language. Therefore, the given language is not regular.

(b) Using an argument essentially identical to that used to show part (a) of this problem (see above), we can show that the language $\{0^n 1 0^n \mid n \geq 0\}$ is not regular.

Now, suppose $\{w \mid w \in \{0,1\}^* \text{ is not a palindrome}\}$ is regular. Then, the complement of the language, i.e, $\{w \mid w \in \{0,1\}^* \text{ is a palindrome}\}$ is also regular (by the closure of regular languages under complement). Consider the intersection of this language with the regular language 0^*10^*. The intersection is exactly the language $\{0^n 1 0^n \mid n \geq 0\}$, which is regular, if $\{w \mid w \in \{0,1\}^* \text{ is a palindrome}\}$ is regular (by the closure of regular languages under intersection). But we proved (see the paragraph above) that this language is not regular. Therefore $\{w \mid w \text{ is not a palindrome}\}$ is not a regular language.

Problem 6: (Sipser 1.53.) Let $\Sigma = \{0, 1, +, =\}$ and

$$ADD = \{x = y + z \mid x, y, z \text{ are binary integers, and x is the sum of y and z }\}$$

Show that ADD is not regular.

Solution 6: Let p be the pumping length. Consider the string "$0^p = 1^p + 0^p$". If we write this string as xyz, where y contains an '=' or a '+', then the string xy^2z contains 2 '+' symbols or 2 '=' symbols, which means xy^2z is not in ADD. Therefore, the string y should be made up of symbols from either the string to the left of '=' or the one right after '=' or the one to the right of '+'. In each of these cases, pumping up the string once yields a string that is not in ADD.

[1] A palindrome is a string that reads the same forward and backward. i.e, $w = w^{\mathcal{R}}$.