CS 181: Formal Languages and Automata Theory	**Spring 2021**
Homework 1	
Assigned: Tue 3/30	*Due: Mon 4/5, 5:00pm PDT*

Problem 1

Let X be the set $\{w, x, y, z\}$, and let B be the set $\{0, 1\}$.

(a) Show the Cartesian product $(B \times B) \times B$

$\{((0,0),0),((0,0),1),((0,1),0),((0,1),1),\ ((1,0),0),((1,0),1),((1,1),0),((1,1),1)\}$

(b) Show the Cartesian product $B \times (B \times B)$

$\{(0,(0,0)),(0,(0,1)),(0,(1,0)),(0,(1,1)),\ (1,(0,0)),(1,(0,1)),(1,(1,0)),(1,(1,1))\}$

(c) Show the Cartesian product $B \times B \times B$

$\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),\ (1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$

(d) What is the cardinality of the power set $\mathcal{P}(X \times X)$

65536

(e) Show the Cartesian product $\{\epsilon, a, ac\} \times \{a, c, aa\}$.

$\{(\epsilon,a),(\epsilon,c),(\epsilon,aa),(a,a),(a,c),(a,aa),(ac,a),(ac,c),(ac,aa)\}$

Problem 2

Let alphabet $\Sigma = \{a, b, c\}$, and let L_2 be any non-empty set of strings over Σ, i.e., a language over Σ.

- The ϵ denotes the empty string defined in the text on page 14 and to be discussed in class on Thursday.
- The $+$ denotes the Kleene operation to be defined in lecture on Thursday, related to the $*$ operation defined in the text on page 44.
- The $\cdot$ denotes the language concatenation operation to be defined in lecture on Thursday, and defined in the text on page 44.

(a) Show the language concatenation $\{\epsilon, a, ac\} \cdot \{a, c, aa\}$

$\{a, c, aa, ac, aaa, aca, acc, acaa\}$

(b) Show the language concatenation $L_2^+ \cdot \{\}$

$\emptyset$

(c) Show the Cartesian Product $\{\epsilon\} \times \Sigma$

$\{(\epsilon, a), (\epsilon, b), (\epsilon, c)\}$

(d) Show the Cartesian Product $\{\} \times L_2^+$

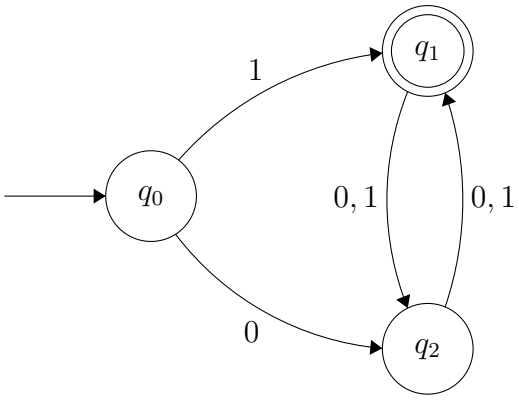
$\emptyset$

(e) What is the language concatenation $\{\epsilon\} \cdot L_2^+$? Does it contain ϵ?

L_2^+. It will contain ϵ if and only if L_2, contains ϵ.

Problem 3

Briefly describe in plain English the language, L_3, over alphabet $\Sigma = \{0, 1\}$ accepted by this DFA:



The DFA accepts strings such that if it starts with 1 then the length of the input is an odd number of symbols and if it starts with a 0 then the length of the input is an even number of symbols.

Problem 4

Classify each of the following languages as a Finite State Language (FSL) or Not Finite State (Non-FSL).

- If you think the language is a FSL, show a DFA for the language and briefly explain how it works to correctly recognize the language. Show your DFA as a *fully specified* state diagram. *Be sure to clearly indicate your initial state and accepting state(s).*
- If you think it is Non-FSL, briefly justify your answer in one or two short English sentences giving your intuition for why you think it is Non-FSL.

(a) Let alphabet $\Sigma = \{0, 1, +, =\}$

$$L_{4a} = \{w \in \Sigma^+ \mid w \text{ is of the form } x + y = z, \text{ where } x, y, z \in \{0, 1\}^+\}$$

In the definition of L_{4a}, the intent is that z is the binary sum of x and y, where x, y, z are interpreted as binary numbers with the least significant bit on the right. Note that leading 0's are allowed and are ignored. E.g., L_{4a} contains $1+10 = 11$, $1+10 = 000011$, and $11 + 01 = 100$. L_{4a} does not contain $01 + 11 = 10$, $1 + 1$, $0 = 0$, and $10 = 1 + 1$.

Solution:

Not finite state. As discussed in class, a DFA has only a finite amount of memory capacity, and it can only read the input tape once left to right. In order to check the binary arithmetic, it would have to store the first binary number. Since that prefix can be indefinitely long, the DFA cannot store it. In order to track and verify sums of arbitrarily long binary strings, we would need an arbitrarily large number of states.

(b) Let alphabet $\Sigma = \{a, b\}$. In lecture Thursday we will define a "run of symbol $x \in \Sigma$ in word $w \in \Sigma^+$" to be a substring of w containing one or more symbols, x, and no other symbols, and also no additional symbols, x, adjacent to it. (This may not be defined in the text).

$$L_{4b} = \{w \in \Sigma^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length,} \\ \text{and all runs of } b\text{'s in } w \text{ are of odd length}\}$$

Why do you think we said "$w \in \Sigma^+$" instead of allowing ϵ in L_{4b}?

Solution:

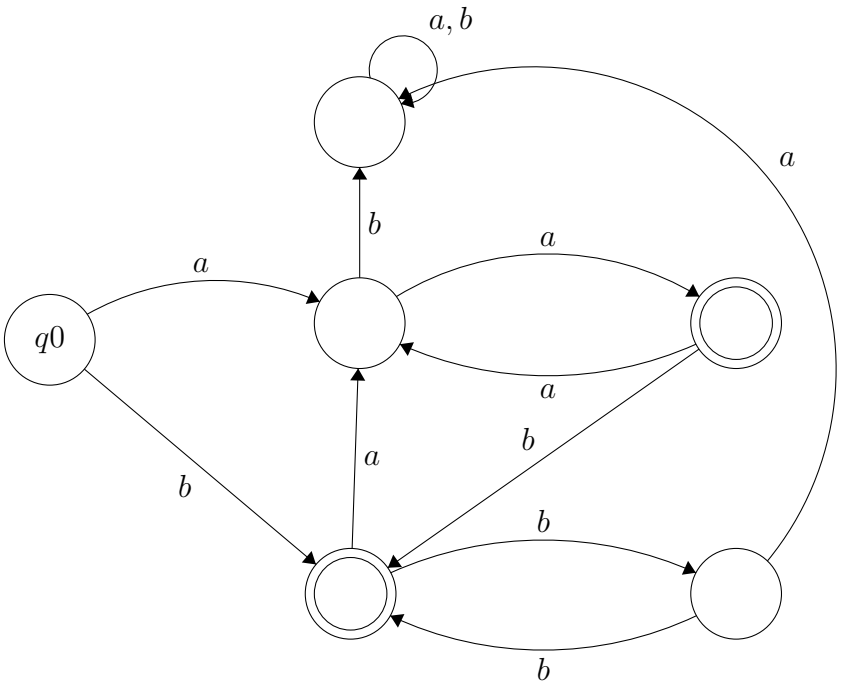


Figure 1: Solution for 4b

Brief explanation: The DFA avoids accepting ϵ since q_0 is not an accepting state. The next two states connected to the initial state handle runs of as *vs.* runs of bs. Each

state is connected to another state to form a loop to keep track of whether the length of the current run is even or odd. The two accepting states detect whether the input so far is valid. When the symbol changes, the DFA can determine at that point whether the previous input violated the language. If the input so far is still valid, the DFA will continue processing the run of the new symbol. If not, it jumps to the dead state at the top and remains there while it consumes the rest of the input.

The reason for excluding the empty string, ϵ, is because it might be ambiguous whether the empty string satisfies the condition "all runs of b's in w are of odd length". Most people would probably say that it is vacuously true for ϵ, but we wanted to avoid that ambiguity to make the problem more clear.

For parts (c) and (d):

- In lecture Thursday, we will define the notation "$|w|$" to be the *length of string* w. I.e., the number of symbols in word w.
- In lecture Thursday, we will define the notation "$\#(x, w)$" to be the number of occurrences of symbol x in string w. This notation is *not* defined in the text.
- The inner "$|\ |$" around the difference "$\#(a, w) - \#(b, w)$" is the usual numerical absolute value function, not string length.

(c) Let alphabet $\Sigma = \{a, b\}$.

$$L_{4c} = \Big\{ w \in \Sigma^* \ \Big|\ |\#(a, w) - \#(b, w)| < 4 \Big\}$$

Not finite state. The DFA would need to track the difference in counts between a and b the whole way through, and this could be an indefinitely large number. In order to do this, we would need an infinite number of states to represent the infinitely many possible differences in occurrences.

(d) Let alphabet $\Sigma = \{a, b\}$.

$$L_{4d} = \Big\{ w \in \Sigma^* \ \Big|\ |\#(a, y) - \#(b, y)| < 4 \text{ for all prefixes, } y \text{ of } w \Big\}$$

Brief explanation: The two parallel paths of the DFA below each keep track of the difference between the number of symbols seen so far. The upper path counts when there is an excess of a's, and the lower path counts when there is an excess of b's. If the difference ever exceeds 3, it goes to the rejecting state.

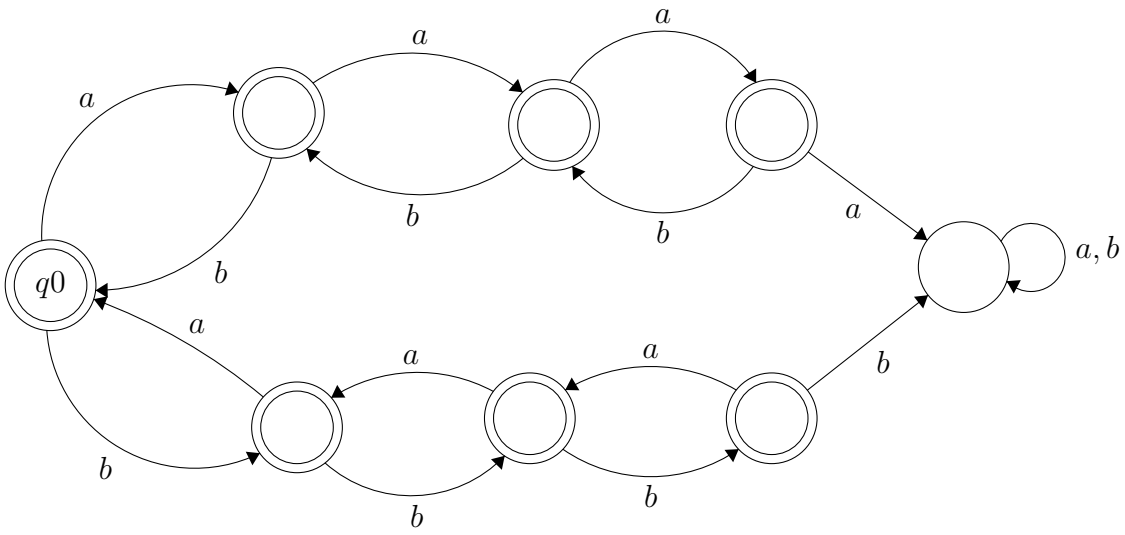


Figure 2: Solution for 4d

Problem 5

Let Σ be an alphabet with at least two symbols. Prove the following statement by induction on $|w|$:

For all strings $w \in \Sigma^*$, if $w = xy$ for some substrings $x, y \in \Sigma^*$, then $w^R = y^R x^R$

Hint: This is actually very easy to prove. The main purpose of this problem is simply to give you an opportunity to refresh your memory about how to set up an inductive proof correctly. It also gives you a chance to practice doing an inductive proof over a string instead of something probably more familiar to you, such as proof by induction over an integer. (Actually, in this case, we could say it's induction over both: w and $|w|$.)

Solution:
We will employ induction on $|w|$:

Basis cases:

$|w| = n = 0$, and $w = xy$:
Then $w = \epsilon = x = y$
$w^R = (xy)^R = (\epsilon\epsilon)^R = \epsilon^R = \epsilon = \epsilon\epsilon = \epsilon^R\epsilon^R = y^R x^R$, since $\epsilon^R = \epsilon$

$|w| = n \geq 1$, and $w = xy$, where x or y is ϵ, but not both:
Say $x = \epsilon$. Otherwise, the argument is symmetrical for x and y. Then:
$w^R = (xy)^R = (\epsilon y)^R = y^R = y^R\epsilon = y^R\epsilon^R = y^R x^R$

Inductive case:

$|w| = n \geq 2$, and $w = xy$, where neither x nor y are ϵ:
Clearly, these three cases cover all possible combinations of x and y.

Assume the statement holds for all strings of length less than n.
Since $y \neq \epsilon$, we can write $y = za$, where $a \in \Sigma$ and $z \in \Sigma^*$, so $w = xy = xza$, and then:
$w^R = (xy)^R = (xza)^R$
The first symbol of the reversal of xza must be the last symbol of xza; so we expect
$w^R = (xza)^R = a(xz)^R$.
Since $|xz| < n$, we can apply the inductive hypothesis to it:
$w^R = a(xz)^R = a(z^R x^R) = (az^R)x^R = (a^R z^R)x^R$
Since $|(a^R z^R)| < n$, we can apply the inductive hypothesis again:
$w^R = (a^R z^R)x^R = (za)^R x^R = y^R x^R$

Problem 6

Briefly explain the system used in the Sipser textbook to number the sections, subsections, exercises, problems, figures, examples, theorems, etc..
Solution:
There is really no one and only correct answer for this. Whatever you write is fine, as long as it represents your understanding of the numbering system. The purpose of the problem was simply to get you to study the system Sipser uses, so you it will be easier for you to find things later in the course.