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 $(B \times B) \times B$

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

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

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

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

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 $\{\epsilon, a, ac\} \cdot \{a, c, aa\}$

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

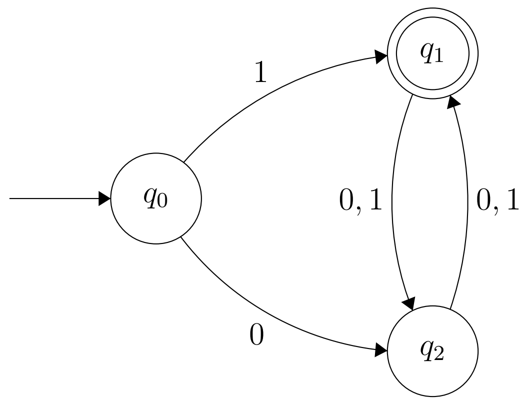
(c) Show the Cartesian Product $\{\epsilon\} \times \Sigma$

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

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

Problem 3

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



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 $10 = 1 + 1$.

(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).

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

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

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 $\Sigma = \{a, b\}$.

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

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

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

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 $|w|$.)

Problem 6

Briefly explain the system used in the Sipser textbook to number the sections, subsections, exercises, problems, figures, examples, theorems, etc..