Topics

1 Language theory

- Definitions of alphabet, string, and language
- Describe a languages using set-builder notation
- Understand the effect of concatenation, Kleene star, and union on strings and as operators on languages
- Know the difference between $\emptyset$ and $\varepsilon$

2 Deterministic finite automata

- Design a DFA for a given language
- Describe the language recognized by a given DFA
- Represent a DFA as a state transition diagram, a table, or a 5-tuple
- Understand the transition function $\delta$, which needs to be defined for every input symbol in every state
- Understand the essence of DFAs: finite memory

3 Nondeterministic finite automata

- Understand nondeterministic computation: Accept if there is any accepting computation
- Computation with $\varepsilon$-transitions
- Design simple NFAs by embracing nondeterminism: guess and check!
- Use the subset construction with $\varepsilon$-closure to convert an NFA to an equivalent DFA

4 Regular expressions

- Recursive definition of regular expressions
- Precedence of the operators in a regular expression ($\ast$, concatenation, $\cup$)
- Describe the language of a given a regular expression
- Use the state elimination method to convert a DFA to an equivalent regular expression
- Use Thompson’s algorithm to convert a regular expression to an equivalent NFA
5 The Pumping Lemma

- Understand the intuition of the Pumping Lemma:
  If a string \( w \) is accepted by some \( M \) and \( w \) is as long or longer than the number of states, then \( w \) must “loop” back to some state \( M \) en route to accept. Therefore, we can eliminate (pump 0 times) or repeat (pump 1 or more times) the segment of \( w \) that labels the loop to obtain new accepted strings.

- Use the Pumping Lemma to prove that a given language is not regular

- Know that the Pumping Lemma is a necessary but not sufficient condition for regular languages; you cannot use the Pumping Lemma to prove a language is regular.

6 Closure properties of regular languages

- Understand the proofs that regular languages are closed under the union, concatenation, Kleene star, complement, and intersection

- Use closure properties to prove that a language is regular
  \( L \) is regular if you can find known regular languages \( L_1 \) and \( L_2 \) such that \( L = L_1 \) – \( L_2 \) since regular languages are closed under difference.

- Use closure properties to prove that a language is not regular.
  By contradiction. Show that if \( L \) were regular, by applying closure properties, we could obtain a known non-regular language \( L' \) from \( L \) (and perhaps other regular languages), and therefore we can \( L \) must be non-regular.
Problem 1

Give short answers to each of the following. Be sure to adequately explain your answers for full credit.

a. Give an example of a regular language $R$ and a non-regular language $N$, such that $R \cap N$ is a regular language.

b. True or false: If an NFA with $n$ states accepts no string with length less than $n$, then it must accept no strings at all.

c. Let $\Sigma$ be an alphabet. Give a short English description of the set $\mathcal{P}(\Sigma^*)$. Briefly justify your answer.

I think there's a single “best answer” – you should be able to describe the set in at most ten words.
Problem 2

Let $\Sigma = \{a,b\}$ and let $L$ be the language over $\Sigma$ given by the regular expression $(ab \cup ba)^*$. Design a DFA for $L$.

You can design the DFA directly; you don't need to use Thompson's algorithm to construct an NFA and then convert it to a DFA.
Problem 3

Let $\Sigma = \{d, f, j, o, r\}$ and consider the following language $L$:

$L = \{w \in \Sigma^* \mid w$ is a substring of fjord $\}.$

Recall that a substring is a contiguous range of characters from a string. For example, $fjo \in L$, $jord \in L$, $\varepsilon \in L$, $f \in L$, and $fjord \in L$, but $dor.f \notin L$ since the letters are contiguous, and $fff \notin L$ because there aren't three consecutive $f$s in fjord.

Design an NFA for $L$. 
Problem 4

Let $M$ be the NFA below. The input alphabet is \{a, b, c\}.

Use the subset construction to create a deterministic finite automaton that is equivalent to $M$. Show your steps.
Problem 5

Convert the following NFA into an equivalent regular expression using the state elimination method.

Step 1 (add new start and final states)

Step 2 (after $q_0$ removed)

Step 3 (after $q_1$ removed)

Step 4 (after $q_2$ removed) = Resulting regular expression
Problem 6

For each language, prove whether it is a regular language. If you say it is regular, give a DFA, NFA, or regular expression for $L$. If you say it is not regular, prove this using the Pumping Lemma for regular languages and/or closure properties.

a. \[ L = \{a^m b^m a^n b^n \mid m, n \geq 0\} \]

b. \[ L = \{a^n \mid n \text{ is a multiple of 8}\} \]

c. \[ L = \{x=y+z \mid x, y, z \text{ are binary integers and } x \text{ is the sum of } y \text{ and } z\} \] The alphabet is $\Sigma = \{0, 1, =, +\}$; an example string is $1011=101+110$. 

Problem 7

Write regular expressions for the following languages over $\Sigma = \{a, b\}$.

a. $\{w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same}\}$

b. $\{w \in \Sigma^* \mid w \neq \varepsilon \text{ and } w \text{'s characters alternate between as and bs}\}$