CMPU 240 · Fall 2022

Exam 1 Practice

Topics

1 Language theory

Encoding problems as languages and inputs as string

- Definitions of *alphabet*, *string*, and *language*
- Describe a language using set-builder notation
- Know the effects of concatenation, Kleene star, and union as operators on languages
- Understand the difference between $\varnothing$ and ε

2 Deterministic finite automata

A simple model of computation with finite memory

- 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 δ, which needs to be defined for every input symbol in every state
- Understand the essence of DFAS: finite memory

3 Nondeterministic finite automata

A more complex model of computation that's easier to work with

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

4 Regular expressions

Describing regular languages as combinations of simpler regular languages

- Recursive definition of regular expressions
- Precedence of the operators in a regular expression (*, 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

5 The Pumping Lemma

A language is non-regular if no automaton with a finite number of states could recognize its strings

- 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 in M *en route* to accept. Therefore, we can eliminate ("pump down") or repeat ("pump up") 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

Preserving regularity, for fun and profit

- Understand the proofs that regular languages are closed under union, concatenation, Kleene star, complement, and intersection
- Use closure properties to prove that a language is regular.

  > E.g., 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.

Try the following problems so we can discuss any questions or difficulties you have, either during the review session or during my office hours before the exam. The problems won't be collected or graded. Collaboration is encouraged; please discuss these problems with classmates to help each other. After the review session, I'll release example solutions.

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 Σ be an alphabet. Give a short English description of the set $\wp(\Sigma^*)$. 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

You can approximate the number of syllables in an English word by counting the number of vowels in the word (including `y`), *except* for

- vowels that have vowels directly before them, and
- the letter `e`, if it appears by itself at the end of a word.

This approach isn't always correct, but it's a good approximation. For example:

- `program` has two vowels and two syllables: pro·gram.
- `peach` has two vowels, but they're consecutive so it's only one syllable.
- `deduce` has two syllables since the final `e` does not count by our rule
- `oboe` has two syllables. It ends with an `e`, but the `e` is preceded by another vowel.
- `why` has one syllable since `y` counts as a vowel.
- `enqueue` has two syllables, en·queue, since the consecutive vowels `ueue` all count as one.

The strings in this language don't need to be English words, e.g., it include the nonsense word `wekruvbsdf`.

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}\}$. Design the state diagram of a DFA for the language

$$\{w \in \Sigma^* \mid w \text{ has at least two syllables according to the rules above}\}.$$

To make your diagram simpler, you can label some transitions as V (for vowels) or C (for consonants) rather than write all the letters.

Problem 3

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

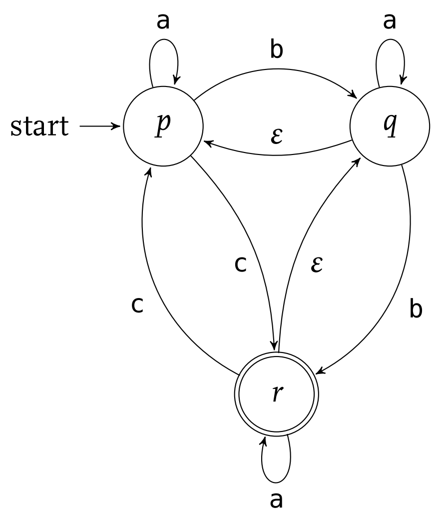
$$L = \{w \in \Sigma^* \mid w \text{ is a substring of } \texttt{fjord}\}.$$

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

Design an NFA for L.

Problem 4

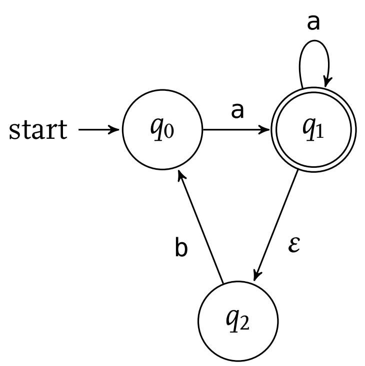
Let M be the NFA below. The input alphabet is $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.



Use the subset construction to create a deterministic finite automaton that is equivalent to M. I recommend computing the ε-closure for each state before you begin.

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

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

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

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

Problem 7

Prove that the regular languages are closed under set difference. That is, prove that if L_1 and L_2 are regular languages over some alphabet Σ, then the language $L_1 - L_2$ is also regular.

Problem 8

Use the Pumping Lemma for regular languages and/or closure properties to prove the following languages are *not* regular.

a. $L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{a}^m\mathtt{b}^n \mid m, n \geq 0\}$.

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