CMPU 240 · Spring 2026

Assignment 2

Submissions due: 10 February, 1:30 p.m.
Corrections due: 12 February, 1:30 p.m.

Exercise 1

It may be surprising how important the empty string and the empty language are in language theory.[1] This exercise makes sure you have a clear understanding of these concepts.

[1] Consider what mathematics would be like without the number 0!

a. Is there any language L where $\varepsilon \in L$? If so, give an example of one. If not, explain why not.

b. Is there any language L where $\varepsilon \notin L$? If so, give an example of one. If not, explain why not.

c. Is there any language L where $\varepsilon \subseteq L$? If so, give an example of one. If not, explain why not.

d. Does $\varnothing = \varepsilon$? Briefly explain your answer.

e. Does $\varnothing = \{\varepsilon\}$? Briefly explain your answer.

Exercise 2

Recall that the formal mathematical definition of a deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set whose elements are called *states*;
- Σ is a non-empty finite set whose elements are called *characters*;
- $\delta : Q \times \Sigma \to Q$ is the *transition function*, described below;
- $q_0 \in Q$ is the start state;
- $F \subseteq Q$ is the set of accept states.

When we've drawn DFAs, we've represented the transitions by arrows labeled with characters. However, in this formal definition, the transition function δ is what specifies these transitions. Specifically, for any state $q \in Q$ and any symbol $\alpha \in \Sigma$, the transition from state q on symbol α is given by $\delta(q, \alpha)$.

This question explores some properties of this definition:

a. Is it possible for a DFA to have no states? If so, define a DFA with no states as a 5-tuple and explain why it meets the above requirements. If not, explain why this is not possible.

To define a machine using a 5-tuple, use this template:

"Let $D = (Q, \Sigma, \delta, q_0, F)$, where Q = ..., Σ = ..., etc."

For this problem, please don't define the transition function δ with a diagram or a table. Instead, define it like a mathematical function.

b. Is it possible for a DFA to have no *accept* states? If so, define a DFA with no accept states as a 5-tuple, and explain why it meets the above requirements. If not, explain why this is not possible.

c. In class, we said that a DFA must obey the rule that for any state and any symbol, there has to be exactly one transition defined on that symbol. What part of the definition guarantees this?

d. Is it possible for a DFA to have an unreachable state (i.e., a state that is never entered, regardless of what string you run the DFA on)? If so, define a DFA with an unreachable state as a 5-tuple, and explain why it meets the above requirements. If not, explain why this is not possible.

Exercise 3

Construct a nondeterministic finite automaton (NFA) to recognize the language

$$\{w \in \Sigma^* \mid w \text{ ends in } \texttt{a}, \texttt{bb}, \text{ or } \texttt{ccc}\},$$

where $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$.

While it's possible to do this completely deterministically, we want you to use the guess-and-check approach from class, which will make it easier!

Exercise 4

Construct a nondeterministic finite automaton (NFA) to recognize the language

$$\{w \in \Sigma^* \mid \text{some character in } \Sigma \text{ appears at most twice in } w\},$$

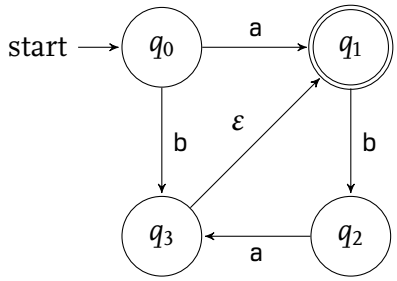
where $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$.

While a DFA for this language would require at least 64 states, an NFA needs far fewer. Consider: What would you do if you knew which character was going to appear at most twice? Embrace the nondeterminism!

Exercise 5

Convert the following NFA (with ε-transitions) to an equivalent DFA. First compute the ε-closure for each of the states in the NFA and then use the subset construction to create a DFA, either in tabular representation or using the elements of the formal 5-tuple.

Don't design a new DFA from scratch. I want you to practice the conversion process that we used to prove that an equivalent DFA exists for every NFA.



Acknowledgments

This assignment includes exercises adapted from Keith Schwarz, Stanford University.