Exercise 1

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

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 \( \emptyset = \varepsilon \)? Briefly explain your answer.

e. Does \( \emptyset = \{\varepsilon\} \)? 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;
- \(\Sigma\) is a non-empty finite set whose elements are called characters (or symbols);
- \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function, described below;
- \(q_0 \in Q\) is the start state;
- \(F \subseteq Q\) is the set of accept (or final) states.

When we’ve drawn DFAs, we’ve represented the transitions by arrows labeled with characters – or as a table with rows and columns corresponding to states and symbols respectively. However, in this formal definition, the transition function \(\delta\) 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 \(\alpha\) 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 \(Q = \ldots, \Sigma = \ldots, \text{etc.}\)”

For this problem, please don’t define the transition function \(\delta\) with a diagram or a table. Instead, define it like a mathematical function.
b. Is it possible for a DFA to have no *accepting* states? If so, define a DFA with no accepting 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

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

where \( \Sigma = \{ a, b, c \} . \)

While it's possible to do this completely deterministically, it's easier if you use the guess-and-check approach from class.

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

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!