CMPU 240 · Spring 2026

Assignment 3

Submissions due: 17 February, 1:30 p.m.
Corrections due: 19 February, 1:30 p.m.

Exercise 1

In class, we saw that if you take the DFA for a language L and flip which states accept and which don't, you have a DFA for its complement, $\overline{L}$.

a. Prove by contradiction that the same is *not* true for NFAs. That is, draw a *simple* NFA for a language L where flipping the accept and non-accept states does not produce an NFA for $\overline{L}$. Briefly (in 1–2 sentences) justify your answer.

b. Briefly explain why the proof from part (a) doesn't contradict the fact that the regular languages are closed under complementation.

Exercise 2

In class, we've seen that the regular languages are closed under complement, union, intersection, concatenation, and Kleene star.

The proofs of these closure properties either directly convert automata for some languages into an automaton for the desired language or they work by combining other closure properties, as we did for intersection.

Using one of these approaches, show that the regular languages are closed under reversal.

DEFINITION For any string $w = w_1 w_2 \cdots w_n$, the *reverse* of w, written w^R, is the string w in reverse order:

$$w^R = w_n \cdots w_2 w_1$$

DEFINITION For any language L, the *reverse* of L, written L^R, is

$$L^R = \{ w^R \mid w \in L \}$$

For example, if

$$L_2 = \{\texttt{cat}, \texttt{dog}\}$$

then

$$L_2^R = \{\texttt{cat}^R, \texttt{dog}^R\}$$
$$= \{\texttt{tac}, \texttt{god}\}$$

Exercise 3

On Unix-style operating systems (like Linux or macOS), files are organized into directories, and you can reference a file by giving the *path* to it – a series of directory names separated by slashes. E.g., the path `/home/mvassar/` might represent Matthew Vassar's home directory, and `/home/mvassar/cmpu240/asmt03.tex` might represent his solutions to this assignment.

Paths that start with a slash character are called *absolute paths* and say exactly where a file is in the file system. Paths that don't start with a slash are called *relative paths* and say where the file is located based on the current directory.[1] For example, if `mvassar` is logged in and is in his home directory, he can use the relative path `cmpu240/asmt03.tex` to access his solution file.

The general pattern is that a file path consists of a series of directory or file names separated by slashes. That path might *start* with a slash, but it isn't required to, and it might *end* with a slash, but it isn't required to. However, you can't have two consecutive slashes.[2]

Let $\Sigma = \{\texttt{a}, \texttt{/}\}$. Write a regular expression for $L = \{w \in \Sigma^* \mid w$ represents the name of a file system path on a Unix-style system$\}$. For example,

$$\texttt{/aaa/a/aa} \in L \qquad \texttt{//a//} \notin L$$
$$\texttt{/} \in L \qquad \texttt{a//a} \notin L$$
$$\texttt{a} \in L \qquad \varepsilon \notin L$$
$$\texttt{/a/a/a/} \in L$$
$$\texttt{aaa/} \in L$$

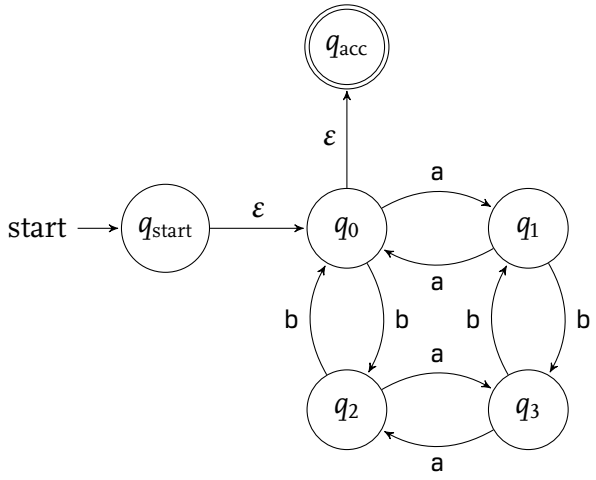
[1] You can also have paths that start with ~, which is an abbreviation for "my home directory", but we'll ignore these for this exercise.

[2] Another simplification; in some cases, you technically *can* have multiple consecutive slashes, but it's never a good idea.

Exercise 4

The state-elimination algorithm gives a way to transform a finite automaton into a regular expression. It's an elegant algorithm once you get the hang of it, so for this problem you'll try applying it to an example.

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$, and let $L_3 = \{w \in \Sigma^* \mid w \text{ has an even number of } \texttt{a}\text{s and } \texttt{b}\text{s}\}$. Below is a finite automaton for L_3 that we've prepared for the state-elimination algorithm by adding a new start state q_{start} and a new accept state q_{acc}:



Go slowly. Remember that to eliminate a state q_{rip}, you should identify all pairs of states q_{in} and q_{out} where there's a transition from q_{in} to q_{rip} and from q_{rip} to q_{out}, then add shortcut edges from q_{in} to q_{out} to bypass state q_{rip}. Remember that q_{in} and q_{out} can be the same state.

a. Follow the state-elimination algorithm to remove q_1 and then q_2. Show your result at this point.

If you've done everything right, for part a, none of the transitions should have Kleene stars in them.

b. Finish the state-elimination algorithm by removing q_3 and then q_0, showing your work.

You should start seeing Kleene stars appear as you remove the remaining states in part b.