Homework 4

> *Last updated:* Mon, 29 Sep 2025 11:13:28 -0400

Out: Mon Sept 29, 12:00pm EDT (noon) **Due**: Mon Oct 06, 12:00pm EDT (noon)

> Note: Assignments are not officially "released" until—and are subject to change without notice up to—the indicated "Out" date and time. If an assignment is posted early, students may look ahead but are responsible for ensuring that they are always working with the most recent version of the homework.

This assignment looks at regular languages and NFAs.

Homework Problems

1. DFA-NFA Equivalence (10 points)
2. "Closed" Operation Practice (10 points)
3. Recursive String Function (10 points)
4. Another Closed Operation (10 points)
5. README (2 point)

Total: 42 points

Submitting

Submit your solution to this assignment in Gradescope `hw4`. Please assign each page to the correct problem and make sure your solutions are legible.

A submission must also include a `README` containing the required information.

1 DFA-NFA Equivalence

Two machines are considered **equivalent** if they recognize the same language (i.e., they accept the same strings).

Come up with a function $\text{CONVERT}_{\text{DFA-NFA}} : \text{DFA} \to \text{NFA}$ where, given some input DFA M that satisfies the formal definition of DFAs from class, $\text{CONVERT}_{\text{DFA-NFA}}(M) =$ an NFA that satisfies the formal definition of NFAs and is **equivalent** to the input DFA.

Use your answers from Homework 1 DFA Formal Description and Homework 3 DFAs vs NFAs to help guide your thinking, and to double-check that your answer is correct.

To prove "equivalence" of two machines, use a variation of our language-machine "Equivalence Table", except now the columns correspond to the two machines.

In addition, since we are not dealing with specific languages or machines here, each line of the table needs an additional "justification" (can be mix of formalism and English prose).

For example:

Some Example String w	**Result of some DFA M on input w?**	**Result of NFA $N = \text{CONVERT}_{\text{DFA-NFA}}(M)$?**	**Justification for why N accepts/rejects?**

A correct answer must demonstrate knowledge of how an NFA computes, what are the possible results of an NFA computation, and how it compares and differs to an analogous DFA's computation.

Though your answer must work for any machine, the answers for the specific machine from Homework 1 DFA Formal Description and Homework 3 DFAs vs NFAs should help guide your thinking double-check that your answer is correct.

Also, make sure your answer is valid according to all the previous discussions, e.g., does not have unbound variables, follows the formal definitions when appropriate, including components of the correct type.

2 "Closed" Operation Practice

Define:

$$\text{BOTH}(A, B) = \{w \mid w \text{ is in language } A \text{ and is in language } B\}$$

Prove that set of regular languages is closed under the BOTH operation by:

- Giving the equivalent IF-THEN statement that must be proved.
- Giving a proof of this IF-THEN statement.

  The proof must be in the form of a Statements and Justifications Table.

  If the proof needs a machine-language "Equivalence Table", the table must include two example strings, one in the language and one not in the language. Also, each line of the table needs a justification.

  Finally, the proof must use the NFA representation of a regular language. This means you will likely need to define some function $\text{BOTH}_{\text{NFA}} : \text{NFA} \times \text{NFA} \to \text{NFA}$

3 Recursive String Function

We learned in class that a **String** is a sequence of symbols from some **alphabet** set.

We also learned that a **String** can be defined recursively.

For this problem, use the following recursive definition of **String** (which differs slightly from, but is equivalent to, the class definition)

A **String** is either:

- the empty string ε
- aw where a is a symbol in some alphabet and w is a **String**

Now there is a function $f : \Sigma_1 \to \Sigma_2$, where Σ_1 and Σ_2 are alphabets.

Write a recursive function $\text{enc} : \Sigma_1^* \to \Sigma_2^*$ that "encodes" its input **String** by calling f on each character in the String.

Your definition of enc should resemble the structure of the recursive definition of its input. That is, the definition of enc should resemble the recursive definition of strings above. More specifically, your answer should have two parts:

- A **base case** that defines enc when the input is the empty string,
- and a **recursive case** that defines enc when the input is a non-empty **String**. Remember that the definition above says that a non-empty **String** may be written as aw where a is from some alphabet Σ and $w \in \Sigma^*$. The recursive case must also include a **recursive call** to enc where the argument is smaller than the input argument.

Note: The "function" in this problem is a mathematical function. In other words, it maps every element of some set (called the domain), to an element in another (possibly same) set (called the range). This problem does not involve any code so submitted solutions should not include any code.

4 Another Closed Operation

1. Assume there exists some enc function from the Recursive String Function Problem above.

   Write a function ENC on languages, where the output of ENC is a set of strings such that enc has been applied to every string in the input language. (you may assume that the input/output of ENC uses the same alphabet as enc)

2. Prove that the regular languages are closed under ENC