

# Homework 2

COMS W3261, Summer A 2022

This homework is due **Monday, 6/5/2023, at 11:59pm EST**. Submit to GradeScope (course code: K3VK75). If you use late days, the absolute latest we can accept a submission is Friday at 11:59 PM EST.

**Grading policy reminder:**  $\LaTeX$  is preferred, but neatly typed or handwritten solutions are acceptable.<sup>1</sup> Feel free to use the .tex file for the homework as a template to write up your answers, or use the template posted on the course website. Your TAs may dock points for indecipherable writing.

Proofs should be complete; that is, include enough information that a reader can clearly tell that the argument is rigorous.

If a question is ambiguous, please state your assumptions. This way, we can give you credit for correct work. (Even better, post on Ed so that we can resolve the ambiguity.)

## $\LaTeX$ resources.

- [Detexify](#) is a nice tool that lets you draw a symbol and returns the  $\LaTeX$  codes for similar symbols.
- The tool [Table Generator](#) makes building tables in  $\LaTeX$  much easier.
- The tool [Finite State Machine Designer](#) may be useful for drawing automata. See also this example ([PDF](#)) ([.tex](#)) of how to make fancy edges (courtesy of Eumin Hong).
- The website [mathcha.io](#) allows you to draw diagrams and convert them to  $\LaTeX$  code.
- To use the previous drawing tools (and for most drawing in  $\LaTeX$ ), you'll need to use the package Tikz (add the command “`\usepackage{tikz}`” to the preamble of your .tex file to import the package).
- [This tutorial](#) is a helpful guide to positioning figures.

---

<sup>1</sup>The website [Overleaf](#) (essentially Google Docs for LaTeX) may make compiling and organizing your .tex files easier. Here's a quick [tutorial](#).

## Problem 1 (14 points)

Examine each of the following regular expressions and write down the language it describes using set notation or 1-2 sentences. (Example:  $01^+ = \{w \mid w \text{ consists of a single } 0 \text{ followed by at least one } 1\}$  or “This regular expression describes the language of strings that consist of a single 0 followed by at least one 1”.)

1. (1 point.)  $\Sigma\Sigma\Sigma\Sigma\Sigma 1$ , where  $\Sigma = \{0, 1\}$ .
2. (1 point.)  $A^*(a \cup e \cup i \cup o \cup u)A^*$ , where  $A$  denotes the alphabet  $\{a, b, c, \dots, x, y, z\}$ , that is, the lowercase roman letters.
3. (1 point.)  $(\Sigma 1)^* \cup (1\Sigma)^*$ , where  $\Sigma = \{0, 1\}$ .
4. (1 point.)  $(0^*10^*1)^*0^*$ , where  $\Sigma = \{0, 1\}$ .
5. (1 point.)  $1(0 \cup 1 \cup 2 \cup 3 \cup 4)DDD \cup (00501 \cup 00544 \cup 06390)$ , where  $D$  is the decimal alphabet  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ .

Bonus (0 points): to what real-world set does this regular expression correspond?

Write regular expressions that evaluate to the languages given.

6. (1 point.) Odd, positive integers written in base 10. You may reference the set  $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  if you want or define your own alphabet(s).
7. (1 point.) All strings over  $\{0, 1\}$  that have an even number of 0's.
8. (3 points.) All strings over  $\{a, b\}$ , including the empty string, that have no  $b$ 's next to each other. (For example:  $\epsilon, b, aaa, bab, abaab$  are included,  $abb$  and  $bbab$  are not.)  
Note that we lack a way to directly exclude certain (sub)strings: the set difference operator ( $\setminus$  or  $-$ ) isn't part of our definition of a regular expression.
9. (4 points.) Consider the alphabet  $\Sigma = \{\rightarrow, \xrightarrow[0]{}, \xrightarrow[1]{}, \bigcirc, \odot\}$ . Here  $\rightarrow$  indicates a start symbol,  $\xrightarrow[0]{}$  and  $\xrightarrow[1]{}$  indicate labeled transitions,  $\bigcirc$  indicates a reject state, and  $\odot$  indicates an accept state. (So the string  $\rightarrow \bigcirc \xrightarrow[0]{} \bigcirc \xrightarrow[0]{} \odot$  corresponds to an NFA that accepts only the string '00', and  $\rightarrow \odot$  corresponds to an NFA that accepts only the string  $\epsilon$ .)

Write

- a regular expression that generates each NFA that accepts a single string of length 2. ( $\rightarrow \bigcirc \xrightarrow[0]{} \bigcirc \xrightarrow[0]{} \odot$  is such an NFA, so it should be generated by your regular expression.)
- a regular expression that generates each NFA that accepts a single string of only zeroes. Don't count  $\epsilon$  as a string of only zeroes. ( $\rightarrow \bigcirc \xrightarrow[0]{} \bigcirc \xrightarrow[0]{} \odot$  is such an NFA, so it should be generated by your regular expression.)

---

Rationale: This question's goal is to make sure you're comfortable interpreting and building regular expressions. References: Sipser pp. 63-66 (regular expressions, Lightning Review 3 (Regular Expressions)); for 1.10 Sipser pp. 47-52; Lightning Review 2 (NFAs).

## Problem 2 (8 points)

1. (3 points). Using the binary alphabet  $\Sigma = \{0, 1\}$ , draw a state diagram for an NFA **with at most 4 states** that recognizes the regular expression

$$(00 \cup 11)(00 \cup 11)^*.$$

Explain in words why your NFA recognizes the language specified.

[Hint: Feel free to use the techniques we used in our closure proofs, for example, concatenating NFAs. These techniques don't necessarily produce an NFA with the minimum number of states, so you may need to simplify afterwards. Alternatively, start by understanding and/or simplifying the regular expression, then directly build an NFA that recognizes the same language. Don't forget to check strings!]

2. (5 points). Using the alphabet  $\{a, b, c\}$ , draw a state diagram for an NFA **with at most 3 states** that recognizes the regular expression

$$\epsilon \cup a^+c^+ \cup a^*b^*c^*.$$

Explain in words why your NFA recognizes the language specified.

---

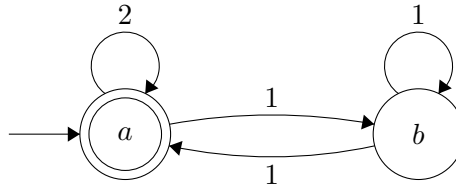
Rationale: The goal of this question is to practice building NFAs that recognize languages defined with regular operations, and simplifying the operations of NFAs.

References: Sipser pp. 63-66 (NFAs), Lightning Review 2 (NFAs); Sipser pp. 59-63 (combining NFAs to recognize languages built with regular operations).

### Problem 3 (9 points)

- (6 points.) In class, we described a systematic procedure for converting an NFA to a DFA by creating states corresponding to the *power set* of the original NFA state set.

The NFA below has two states,  $a$  and  $b$ , and uses the alphabet  $1, 2$ . Draw the state diagram of an equivalent DFA that uses the states  $\emptyset$ ,  $\{a\}$ ,  $\{b\}$ , and  $\{a, b\}$ .



- (3 points.) Describe the language recognized by the pictured NFA (and the DFA you've just drawn) and justify your reasoning using either automaton.

---

Rationale: The goal of this question is to practice thinking about DFAs that simulate other automata, as well as the specific process of converting an NFA into an equivalent DFA.

References: See Sipser pp. 54-58 (Converting DFAs to NFAs), Example 2 (Converting NFAs to DFAs).

## Problem 4\* (1 bonus point)

On page 87, Exercise 1.24, Sipser defines the *Finite State Transducer*, a type of DFA whose output is a string.

- (0.5 points). Read Exercise 1.24 and compute the output of the FST  $T_2$  on the string 'baaabb'.
- (0.5 points). Read Exercise 1.25 and write down a formal definition of an FST.

---

Rationale: Optional, just for fun. Bonus points will be added to your score on this HW, which is out of 31; they don't count for more or less than regular points.