CSCI 432/532, Spring 2024

Homework 2

Due Monday, January 29, 2024 at 9pm Mountain Time

Submission Requirements

- Type or clearly hand-write your solutions into a PDF format so that they are legible and professional. Submit your PDF to the appropriate Moodle dropbox.
- Do not submit your first draft. Type or clearly re-write your solutions for your final submission.
- You may work with a group of up to three students and submit ***one single document*** for the group. Just be sure to list all group members at the top of the document.
- Each homework will include at least one fully solved problem, similar to that week's assigned problems, together with the rubric we would use to grade this problem if it appeared in an actual homework. These model solutions show our recommendations for structure, presentation, and level of detail in your homework solutions. (Obviously, the actual *content* of your solutions won't match the model solutions, because your problems are different!) Note: this copy currently does not have any solved problems, but will updated with solved problems soon.

Academic Integrity

Remember, you may access ***any*** resource in preparing your solution to the homework. However, you must

- write your solutions in your own words, and
- credit every resource you use (for example: "Bob Smith helped me on problem 2. He took this course at UM in Fall 2020"; "I found a solution to a problem similar to this one in the lecture notes for a different course, found at this link: www.profzeno.com/agreatclass/lecture10"; "I asked ChatGPT how to solve problem 1 part (c); "I put my solution for problem 1 part (c) into ChatGPT to check that it was correct and it caught a missing case and suggested some grammer fixes.") If you use the provided LaTeX template, you can use the `sources` environment for this. Ask if you need help!

Grading Rubrics

For the regular expression problems:

Regular expression rubric. 10 points =

+ 2 for a syntactically correct regular expression.
+ 4 for a brief English explanation of your regular expression. This is how you argue that your regular expression is correct.
  - For longer expressions, you should explain each of the major components of your expres-

sion, and separately explain how those components fit together.

- We do not want a transcription; don't just translate the regular-expression notation into English.
- Yes, you need to write it down. Yes, even if it's "obvious". Remember that the goal of the homework is to communicate with people who aren't as clever as you.

+ 4 for correctness.

- -4 for incorrectly answering $\emptyset$ or Σ^*.
- -1 for a single mistake: one typo, excluding exactly one string in the target language, or including exactly one string not in the target language. (The incorrectly handled string is almost always the empty string ε.)
- -2 for incorrectly including/excluding more than one but a finite number of strings.
- -4 for incorrectly including/excluding an infinite number of strings.

• Regular expressions that are more complex than necessary may be penalized. Regular expressions that are significantly too complex may get no credit at all. On the other hand, minimal regular expressions are not required for full credit.

For the DFAs:

NFA/DFA rubric. 10 points =

+ 2 for an unambiguous description of a DFA or NFA, including the states set Q, the start state s, the accepting states A, and the transition function δ.

- Drawings:
  - Use an arrow from nowhere to indicate the start state s.
  - Use doubled circles to indicate accepting states A.
  - If $A = \emptyset$, say so explicitly.
  - If your drawing omits a junk/trash/reject/hell state, say so explicitly.
  - Draw neatly! If we can't read your solution, we can't give you credit for it.
- Text descriptions: You can describe the transition function either using a 2d array, using mathematical notation, or using an algorithm.
  - You must explicitly specify $\delta(q, a)$ for every state q and every symbol a.
  - If you are describing an NFA with ε-transitions, you must explicitly specify $\delta(q, \varepsilon)$ for every state q.
  - If you are describing a DFA, then every value $\delta(q, a)$ must be a single state.
  - If you are describing an NFA, then every value $\delta(q, a)$ must be a set of states.
  - In addition, if you are describing an NFA with ε-transitions, then every value $\delta(q, \varepsilon)$ must be a set of states.
- Product constructions: You must give a complete description of each of the DFAs you are combining (as either drawings, text, or recursive products), together with the accepting states of the product DFA. In particular, we will not assume that product constructions compute intersections by default.

+ 4 for briefly explaining the purpose of each state in English. This is how you argue that your DFA or NFA is correct.

- In particular, each state must have a mnemonic name.

- For product constructions, explaining the states in the factor DFAs is both necessary and sufficient.
- Yes, we mean it. A perfectly correct drawing of a perfectly correct DFA with no state explanation is worth at most 6 points.

+ 4 for correctness.

- -1 for a single mistake: a single misdirected transition, a single missing or extra accepting state, rejecting exactly one string that should be accepted, or accepting exactly one string that should be accepted. (The incorrectly accepted/rejected string is almost always the empty string ε.)
- -4 for incorrectly accepting every string, or incorrectly rejecting every string.
- -2 for incorrectly accepting/rejecting more than one but a finite number of strings.
- -4 for incorrectly accepting/rejecting an infinite number of strings.

1. For each of the following languages over the alphabet $\{0, 1\}^*$, give an equivalent regular expression, and briefly explain why your regular expression is correct. Note that there are infinitely many correct answers for each language.

   (a) All strings that begin with the prefix `001`, end with the suffix `100`, and contain an odd number of `1`'s.

   (b) All strings that contain both `0011` and `1100` as substrings.

   (c) All strings that contain the substring `01` an odd number of times.

2. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, describe a DFA that accepts the language, and briefly describe the purpose of each state. You can describe your DFA using a drawing or using formal mathematical notation.

(a) All strings in 1^*01^*.

(b) All strings containing the substring 01010010.

(c) All strings that represent a multiple of 5 in base 3. For example, this language contains the string 10100, because $101003 = 90_{10}$ is a multiple of 5. (In general, numbers in base 3 could contain the symbol 2, but those are excluded from this language.)