HW2CSE105W24: Homework assignment 2

CSE105W24

Due: February 1st at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment,

You will practice designing multiple representations of regular languages and working with general constructions of automata to demonstrate the richness of the class of regular languages. You will also distinguish between regular and nonregular languages using both closure arguments and the pumping lemma.

Resources: To review the topics for this assignment, see the class material from Weeks 2-4. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Chapter 1, Section 2.2. Chapter 1 exercises 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12, 1.14, 1.15, 1.16, 1.17, 1.19, 1.20, 1.21, 1.22, 1.29, 1.30. Chapter 1 problems 1.49, 1.50, 1.51.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. For "graded for correctness" questions: collaboration is allowed only with CSE 105 students in your group; if your group has questions about a problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza. For "graded for completeness" questions: collaboration is allowed with any CSE 105 students this quarter; if your group has questions about a problem, you may ask in drop-in help hours or post a public post on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if

you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.
- You may not collaborate on homework questions graded for correctness with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw2CSE105W24".

Assigned questions

1. **Number representations** (12 points): Integers can be represented using base b expansions, for a convenient choice of base b: for b an integer greater than 1 and n a positive integer, the **base b expansion of n** is defined to be

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

Notice: *The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{Z} \mid 0 \leq x < b\}$ whose leftmost character is nonzero.*

An important property of base b expansions of integers is that, for each integer b greater than 1, each positive integer $n = (a_{k-1} \cdots a_1 a_0)_b$, and each nonnegative integer a less than b,

$$bn + a = (a_{k-1} \cdots a_1 a_0 a)_b$$

In other words, shifting the base b expansion to the left results in multiplying the integer value by the base. In this question we'll explore building deterministic finite automata that recognize languages that correspond to useful sets of integers.

(a) (*Graded for correctness*) [1] Design a DFA that recognizes the set of binary (base 2) expansions of positive integers that are powers of 2. A complete solution will include the state diagram of your DFA and a brief justification of your construction by explaining the role each state plays in the machine, as well as a brief justification about how the strings accepted and rejected by the machine connect to the specified language.

Hints: (1) A power of 2 is an integer x that can be written as 2^y for some nonnegative integer y, (2) the DFA should accept the strings 100, 10 and 100000 and should reject the strings 010, 1101, and ε (can you see why?).

(b) (*Graded for completeness*) [2] Consider arbitrary positive integer m. Design a DFA that recognizes the set of binary (base 2) expansions of positive integers that are multiples of m. A complete solution will include the formal definition of your DFA (paramterized by m) and a brief justification of your construction by explaining the role each state plays in the machine, as well as a brief justification about how the strings accepted and rejected by the machine connect to the specified language.

Hints: (1) Consider having a state for each possible remainder upon division by m. (2) To determine transitions, notice that reading a new character will shift what we already read over by one slot.

(c) (*Graded for correctness*) Choose a positive integer m_0 between 4 and 8 (inclusive) and draw the state diagram of a DFA recognizing the language over $\{0, 1, 2\}$

$$\{w \in \{0,1,2\}^* \mid w \text{ is a base 3 expansion of a positive integer that is a multiple of } m_0\}$$

A complete solution will include the state diagram of your DFA and a brief justification of your construction by explaining the role each state plays in the machine, as well as a brief justification about how the strings accepted and rejected by the machine connect to the specified language.

---

[1] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

[2] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we expect you to include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

Bonus extension to think about (ungraded): Which other languages related to sets of integers can be proved to be regular using a similar strategy?

2. **Multiple representations** (10 points): For any language $L \subseteq \Sigma^*$, recall that we define its *complement* as

$$\overline{L} := \Sigma^* - L = \{w \in \Sigma^* \mid w \notin L\}$$

That is, the complement of L contains all and only those strings which are not in L. Our notation for regular expressions does not include the complement symbol. However, it turns out that the complement of a language described by a regular expression is guaranteed to also be describable by a (different) regular expression. For example, over the alphabet $\Sigma = \{0, 1\}$, the complement of the language described by the regular expression Σ^*0 is described by the regular expression $\varepsilon \cup \Sigma^*1$ because any string that does not end in 0 must either be the empty string or end in 1.

For each of the regular expressions R over the alphabet $\Sigma = \{a, b\}$ below, write the regular expression for $\overline{L(R)}$. Your regular expressions may use the symbols $\varnothing$, ε, a, b, and the following operations to combine them: union, concatenation, and Kleene star.

Briefly justify why your solution for each part works by giving plain English descriptions of the language described by the regular expression and of its complement and connecting them to the regular expression via relevant definitions. An English description that is more detailed than simply negating the description in the original language will likely be helpful in the justification.

Alternatively, you can justify your solution by first designing a DFA that recognizes $L(R)$, using the construction from class and the book to modify this DFA to get a new DFA that recognizes $\overline{L(R)}$, and then applying the constructions from class and the book to convert this new DFA to a regular expression.

For each part of the question, clearly state which approach you're taking and include enough intermediate steps to illustrate your work.

(a) (*Graded for correctness*) a^*b^*

(b) (*Graded for correctness*) $(a \cup b)ab^*$

3. **Applying general constructions** (12 points): In this question, you'll practice working with formal general constructions for NFAs and translating between state diagrams and formal definitions. Consider the following general construction: Let $N_1 = (Q, \Sigma, \delta_1, q_1, F_1)$ be a NFA and assume that $q_0 \notin Q$. Define the new NFA $N_2 = (Q \cup \{q_0\}, \Sigma, \delta_2, q_0, \{q_1\})$ where

$$\delta_2 : (Q \cup \{q_0\}) \times \Sigma_\varepsilon \to (Q \cup \{q_0\})$$

is defined by

$$\delta_2(q, a) = \begin{cases} \{q' \in Q \mid q \in \delta_1(q', a)\} & \text{if } q \in Q,\ a \in \Sigma_\varepsilon \\ F_1 & \text{if } q = q_0,\ a = \varepsilon \\ \emptyset & \text{if } q = q_0,\ a \in \Sigma \end{cases}$$

(a) (*Graded for correctness*) Illustrate this construction by defining a specific example NFA N and applying the construction above to create a new NFA. Your example NFA should

- Have exactly three states (all reachable from the start state),
- Have at least one spontaneous move (arrow labelled ε),
- Accept at least one string and reject at least one string, and
- Not have any states labelled q_0.

Apply the construction above to create the new NFA. A complete submission will include the state diagram of your example NFA N and the state diagram of the NFA resulting from this construction.

(b) (*Graded for correctness*) Use Theorem 1.39 on page 55 of the book (see also page 7 in Week 3 notes) to construct a DFA equivalent to your example NFA N from part (a). A complete submission will include the state diagram of your example NFA N and the state diagram of the DFA resulting from this construction, with the correct state labels for this DFA. You may prune the DFA so that only the "macro-states" reachable from the start state are included.

(c) (*Graded for completeness*) Explain the relationship between N_1 and N_2 in the general construction. Give an example string that is accepted by your example NFA N and is rejected by the NFA that results from applying the general construction that illustrates this relationship, or explain why there is no such example string.

4. **Pumping** (8 points):

(a) (*Graded for correctness*) Give an example of a language over the alphabet $\{0, 1\}$ that has cardinality 3 and for which 5 is a pumping length and 4 is not a pumping length. A complete solution will give a clear and precise description of the language, a justification for why 5 is a pumping length, and a justification for why 4 is not a pumping length. Is this language regular?

(b) (*Graded for completeness*) Consider the following attempted "proof" that the set

$$X = \{uw \mid u \text{ and } w \text{ are strings over } \{0, 1\} \text{ and have the same length}\}$$

is nonregular.

> "Proof" that X_1 is not regular using the Pumping Lemma: Let p be an arbitrary positive integer. We will show that p is not a pumping length for X_1.
> Choose s to be the string 1^p0^p, which is in X_1 because we can choose $u = 1^p$ and $w = 0^p$ which each have length p. Since s is in X_1 and has length greater than or equal to p, if p were to be a pumping length for X_1, s ought to be pump'able. That is, there should be a way of dividing s into parts x, y, z where $s = xyz$, $|y| > 0$, $|xy| \leq p$, and for each $i \geq 0$, $xy^iz \in X_1$. Suppose x, y, z are such that $s = xyz$, $|y| > 0$ and $|xy| \leq p$. Since the first p letters of s are all 1 and $|xy| \leq p$, we know that x and y are made up of all 1s. If we let $i = 2$, we get a string xy^iz that is not in X_1 because repeating y twice adds 1s to u but not to w, and strings in X_1 are required to have u and w be the same length. Thus, s is not pumpable (even though it should have been if p were to be a pumping length) and so p is not a pumping length for X_1. Since p was arbitrary, we have demonstrated that X_1 has no pumping length. By the Pumping Lemma, this implies that X_1 is nonregular.

Find the (first and/or most significant) logical error in the "proof" and describe why it's wrong. Then, either prove that the set is actually regular (by finding a regular expression that describes it or a DFA/NFA that recognizes it, and justifying why) **or** fix the proof so that it is logically sound.

(c) (*Graded for completeness*) In class and in the reading so far, we've seen the following examples of nonregular sets:

$$\{0^n1^n \mid n \geq 0\} \qquad \{0^n1^m \mid 0 \leq m \leq n\} \qquad \{0^n1^m0^n \mid n, m \geq 0\}$$
$$\{0^n1^n \mid n \geq 2\} \qquad \{0^i1^{2i} \mid 0 \leq i\} \qquad \{w \in \{0,1\}^* \mid w = w^R\}$$
$$\{0^n1^m \mid 0 \leq n \leq m\} \qquad \{0^i1^{i+1} \mid 0 \leq i\} \qquad \{ww^R \mid w \in \{0,1\}^*\}$$

Modify one of these sets in some way and use the Pumping Lemma to prove that the resulting set is still nonregular.

5. **Regular and nonregular languages** (8 points): In Week 2's review quiz, we saw the definition that a set X is said to be **closed under an operation** if, for any elements in X, applying to them gives an element in X. For example, the set of integers is closed under multiplication because if we take any two integers, their product is also an integer .

Prove or disprove each closure claim statement below about the class of regular languages and the class of nonregular languages. Your arguments may refer to theorems proved in the textbook and class, and if they do, should include specific page numbers and references (i.e. write out the claim that was proved in the book and/or class).

Recall the definitions we have:

For languages L_1, L_2 over the alphabet $\Sigma_1 = \{0, 1\}$, we have the associated sets of strings

$$SUBSTRING(L_1) = \{w \in \Sigma_1^* \mid \text{there exist } a, b \in \Sigma_1^* \text{ such that } awb \in L_1\}$$

and

$$L_1 \circ L_2 = \{w \in \Sigma_1^* \mid w = uv \text{ for some strings } u \in L_1 \text{ and } v \in L_2\}$$

(a) (*Graded for completeness*) The set of regular languages over $\{0, 1\}$ is closed under set-wise concatenation.

(b) (*Graded for completeness*) The set of nonregular languages over $\{0, 1\}$ is closed under set-wise concatenation.

(c) (*Graded for completeness*) The set of regular languages over $\{0, 1\}$ is closed under the $SUBSTRING$ operation.

(d) (*Graded for completeness*) The set of nonregular languages over $\{0, 1\}$ is closed under the $SUBSTRING$ operation.