HW1 : Regular Expressions and Finite Automata

CSE105Sp23

Due: April 11th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment,

You will practice reading and applying the definitions of alphabets, strings, languages, Kleene star, and regular expressions. You will use regular expressions and relate them to languages and finite automata. You will use precise notation to formally define the state diagram of finite automata, and you will use clear English to describe computations of finite automata informally.

Resources: To review the topics for this assignment, see the class material from Week 1. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Section 0, 1.3, 1.1. Chapter 1 exercises 1.1, 1.2, 1.3, 1.18, 1.23.

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. The lowest HW score will not be included in your overall HW average. 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. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) 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 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 "hw1CSE105Sp23".

Assigned questions

1. **Functions over sets of strings** (17 points):
For this question, fix the alphabets $\Sigma = \{0,1\}$ and $\Gamma = \{0,1,2\}$.

Whenever K is a set of strings over Γ and L is a set of strings over Σ, we can use the following rules to define associated sets of strings:

$$\begin{aligned}
\textsc{Substring}(K) &:= \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\} \\
\textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2s in } w \text{ is a string in } L\} \\
&= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\}
\end{aligned}$$

Note: Formally, Substring and Rep are functions whose domains and codomains are specified as

$$\textsc{Substring} : \mathcal{P}(\Gamma^*) \to \mathcal{P}(\Gamma^*)$$

and

$$\textsc{Rep} : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Gamma^*)$$

In other words, SUBSTRING maps sets of strings with characters $\{0, 1, 2\}$ to associated sets of strings with characters $\{0, 1, 2\}$; and REP maps sets of strings with characters in $\{0, 1\}$ to associated sets of strings with characters in $\{0, 1, 2\}$.

(a) (*Graded for correctness*) [1] Consider $w = 0120$ (which is a string in Γ^*). List every element of the set SUBSTRING($\{w\}$). In other words, fill in the blank

$$\text{SUBSTRING}(\{w\}) = \{\underline{\hspace{8cm}}\}$$

Briefly justify your answer by referring back to the relevant definitions.

Not graded, but good to think about: Why do we need the curly braces—"{" and "}"—around w for the input to SUBSTRING*?*

(b) (*Graded for correctness*) Specify an example language A over Γ such that $A \neq \Gamma^*$ and yet SUBSTRING(A) $= \Gamma^*$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language A and a precise and clear description of the result of computing SUBSTRING(A) using relevant definitions to justify this description and to justify the set equality with Γ^*, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

(c) (*Graded for completeness*) [2] Define the language B to be the language over Σ described by the regular expression

$$\Sigma^* 1 \Sigma^*$$

In plain English, we might explain that B is the set of all strings of 0s and 1s that contain a 1. Give a plain English explanation for the set of strings REP(B).

(d) (*Graded for correctness*) Prove/disprove: For every finite language L over Σ, REP(L) is also a finite set of strings. A complete answer will either give a general argument starting with an arbitrary finite language and proving that the result of applying REP is also finite, or will give a counterexample (which is a specific example of a finite language L for which applying REP gives an infinite language, with justification referring back to the relevant definitions).

Note: A finite language is a set of finitely many strings. This includes the possibility that L is the empty set!

(e) (*Graded for completeness*) Write a template for a regular expression that describes REP(L) when L is described by a regular expression R. You may use union, concatenation, Kleene star, and Σ, Γ, and R. (We're using the shorthand for regular expressions describing alphabets from page 64.)

[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 ask that you 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.

2. **Deciphering regular expressions** (22 points):
For this question, let's fix the regular expression over the alphabet $\{0,1\}$

$$R = 0^*(1 \cup 10)^*$$

For each choice of strings of length 3, $a, b, c \in \{0,1\}^3$ we can define the regular expression:

$$X_{a,b,c} = 0(a \cup b \cup c)^*$$

(a) (*Graded for completeness*) Give a plain English explanation for the language described by the regular expression R. This continues a theme from Problem 1—before trying to prove formal statements about a specific regular expression, it's often good to try to translate it into a form that is more easy to reason about. Typically speaking, the shorter and more concise your plain English description is, the more useful it will be in reasoning about the language.

(b) (*Graded for correctness*) Suppose $a = 000$, $b = 001$, $c = 011$ so

$$X_{a,b,c} = 0(000 \cup 001 \cup 011)^*$$

Show that $L(R) \not\subseteq L(X_{a,b,c})$ by giving some string in $L(R)$ which is not in $L(X_{a,b,c})$, and justifying this choice referring back to relevant definitions.

(c) (*Graded for correctness*) More generally, prove that

$$L(R) \not\subseteq L(X_{a,b,c})$$

for *all* possible strings $a, b, c \in \{0,1\}^3$. Hint: What are the possible lengths of strings in $L(R)$ (and why does this help)?

(d) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0,1,2\}^3$ such that

$$L(X_{a,b,c}) \subseteq L(R)$$

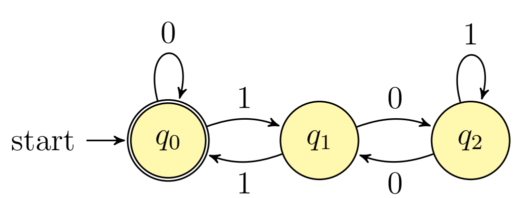
Briefly justify your answer by explaining how an arbitrary element of $L(X_{a,b,c})$ is guaranteed to be an element of $L(R)$.

(e) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0,1,2\}^3$ such that

$$L(X_{a,b,c}) \not\subseteq L(R)$$

Briefly justify your answer by giving a counterexample string that is in $L(X_{a,b,c})$ and is not in $L(R)$ (and explaining why using relevant definitions).

3. **The right transition function can make or break a DFA** (6 points):
Consider the finite automaton $(Q, \Sigma, \delta, q_0, F)$ depicted below



where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, and $F = \{q_0\}$.

(a) (*Graded for completeness*) Find and fix the mistake in the following symbolic description of the transition function $\delta: Q \times \Sigma \to Q$: for each $j \in \{0, 1\}$

$$\delta(q_0, j) = q_j \qquad \delta(q_1, j) = q_{1-j} \qquad \delta(q_2, j) = q_{1+j}$$

(b) (*Graded for correctness*) Keeping the same set of states $Q = \{q_0, q_1, q_2\}$, alphabet $\Sigma = \{0, 1\}$, starting state q_0, and set of accepting states $F = \{q_0\}$, change the transition function δ so that the resulting finite automaton recognizes the language described by the regular expression

$$0^* \cup \Sigma^* 1000^*$$

Briefly justify why the resulting finite automaton works by describing the role of each state with your new transition function and relating it to a plain English description of the language described by the regular expression.

Note: with regular expressions $*$ binds more tightly than concatenation so $1000^* = (100)(0^*)$.

(Challenge question, not graded) There is a beautiful plain English description of the language recognized by the finite automaton with the state diagram depicted at the start of Problem 3. What is it?

4. **Being precise with terminology** (5 points):
For each of the following statements, determine if it is true, false, or if the question doesn't even make sense (because the statement isn't well formed or doesn't use terms in ways consistent with definitions from class).

(a) (*Graded for completeness*) The empty string is in every language.

(b) (*Graded for completeness*) Σ^* is a language.

(c) (*Graded for completeness*) Every language is a regular expression.

(d) (*Graded for completeness*) Alphabets are infinite.

(e) (*Graded for completeness*) There is a (finite) number $k \in \mathcal{N}$ such that every DFA has fewer than k states.

HW2 : Regular Languages and Automata Constructions Due: April 18th at 5pm (no penalty late submission until 8am next morning), via Gradescope

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.

Resources: To review the topics you are working with for this assignment, see the class material from Week 1 and Week 2. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Section 1.1, 1.2, 1.3. 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.

Key Concepts: Regular expressions, language described by a regular expression, deterministic finite automata (DFAs), regular languages, closure of the class of regular languages under certain operations, nondeterministic finite automata (NFA).

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. The lowest HW score will not be included in your overall HW average. 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. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) 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 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 "hw2CSE105Sp23".

Assigned questions

1. **It can be hard to give a good complement** (15 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 $\Sigma = \{0,1\}$ below, write the regular expression for $\overline{L(R)}$. Your regular expressions may use the symbols $\varnothing$, ε, 0, 1, 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.

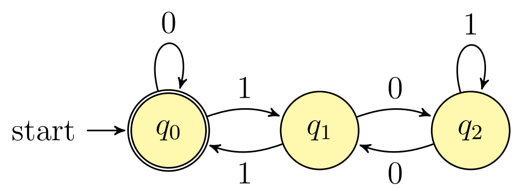
   (a) (*Graded for correctness*) [3] $(\Sigma\Sigma)^*$

[3]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.

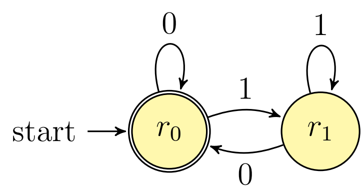
(b) (*Graded for correctness*) $\Sigma^*11\Sigma^*$

(c) (*Graded for correctness*) $0^*10^*10^*$

2. **Closure of the class of regular languages under intersection** (12 points):
For this question, let $\Sigma = \{0, 1\}$. Recall the DFA over Σ from the previous homework:



We'll call the language recognized by the DFA above A. Let's also define a new language $B \subseteq \Sigma^*$ to be the language recognized by the DFA over Σ with state diagram below:



(a) (*Graded for correctness*) Using the construction for the intersection of two regular languages (Sipser page 46), draw the state diagram for a DFA recognizing the intersection of the languages A and B. The labels of each one of your states should be the ordered pair of labels for the states from the two machines above. Your diagram should have 6 states.

(b) (*Graded for completeness*) In this part of the problem, you will prove that the general construction for the DFA recognizing intersection of two languages that you used in part (a) does not always produce a DFA with the smallest number of states possible. You will do this by giving one counterexample (that combined with your work in part (a), proves the general claim). Your task: design a DFA with exactly 4 states that recognizes the language $A \cap B$. Briefly justify why your design works by describing the role of each state of your DFA and relating it to a plain English description of the language resulting from the intersection.

(c) (*Graded for correctness*) Later in the class we will learn that there are some languages which are not regular, and in fact, we will learn specific techniques to prove that certain languages are not regular. For the moment, however, we can already investigate closure properties of the class of regular languages just by knowing that a non-regular language exists.

We know (from the textbook and our work in class) that if L and K are regular languages, then $L \cap K$ is regular (for arbitrary languages L and K). Prove that the converse of this statement is false; that is, give a counterexample by giving a specific regular language L so that for each non-regular language X, $L \cap X$ is regular (even though X isn't).

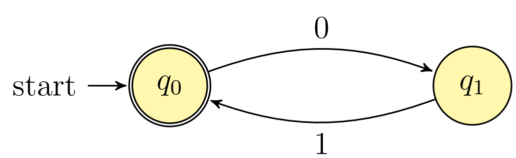
In your solution, justify why L is regular and why $L \cap X$ is regular (for arbitrary X) using relevant definitions.

(Challenge question, not graded) Prove/disprove: For any language L over Σ^*, $L \cap B$ is regular implies L is regular, where B is the specific language from part (a) and (b) of Problem 2.

3. **Closure of the class of regular languages under** SUBSTRING (16 points):
Let $\Gamma = \{0, 1, 2\}$. From the previous homework, recall the function SUBSTRING that has domain and codomain $\mathcal{P}(\Gamma^*)$, where, for each language K over Γ,

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

(a) (*Graded for correctness*) Consider the NFA over Γ with state diagram:



We'll call the language recognized by the NFA above C. Fill in the blanks below:

- An example of a string over Γ that is in C **and** is in SUBSTRING(C) is ____________ because ____________
- An example of a string over Γ that is in C **and** is **not** in SUBSTRING(C) is ____________ because ____________
- An example of a string over Γ that is **not** in C **and** is in SUBSTRING(C) is ____________ because ____________
- An example of a string over Γ that is **not** in C **and** is **not** in SUBSTRING(C) is ____________ because ____________

For each item, you'll either fill in a specific string and a justification that refers back to the relevant definitions, or you'll write "impossible" for the first part of the sentence and justify why it's impossible to find such an example referring back to the relevant definitions.

(b) (*Graded for completeness*) Prove that the class of regular languages is closed under the SUBSTRING operation. Namely, give a general construction that takes an arbitrary NFA and constructs an NFA that recognizes the result of applying SUBSTRING to the language recognized by the original machine. You can describe your construction in words and/or draw a picture to illustrate your construction. You do not have to write down a formal specification.

(c) (*Graded for completeness*) Draw the state diagram of an NFA over Γ that recognizes SUBSTRING(C) (for C the language from part (a) of this Problem), using your construction from part (b) of this Problem, or manually constructing it. Describe the computation(s) of this NFA for each of the sample strings you gave in part (a).

4. **Closure of the class of regular star-free languages under** Rep (7 points):
A language is said to be *star-free* whenever it can be described by a regular expression that has no Kleene star operations, but where complement operation can be incorporated into the expression as many times as you like. For example, the language

$$\{\varepsilon, 0010\}$$

is star-free because it can be described by $\varepsilon \cup 0010$ which does not use the Kleene star operation symbol.

   (a) (*Graded for correctness*) Prove that the set of all strings over $\Gamma = \{0, 1, 2\}$ is star-free. A complete solution will give an expression that describes this language that does not use Kleene star but may incoporate the complement expression as many times as you like, along with a justification that refers back to relevant definitions.

   (b) (*Graded for completeness*) Prove that every finite language is star-free.

   (c) (*Graded for completeness*) Let $\Sigma = \{0, 1\}$. From the previous homework, recall the function Rep that has domain $\mathcal{P}(\Sigma^*)$ and codomain $\mathcal{P}(\Gamma^*)$, where, for each language L over Σ,

$$\textsc{Rep}(L) := \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\}$$

   Show that $\textsc{Rep}(L)$ is a regular and star-free language whenever L is a regular and star-free language. That is, given an expression R describing L, write a regular expression for $\textsc{Rep}(L)$ using only the regular expressions R, $\varnothing$, ε, 0, 1, 2, and the following operations to combine them: union, concatenation, and complement. You may assume that $\overline{R}$ describes $\Sigma^* - L(R)$, that is, the complement for the regular expression R over the alphabet Σ is itself a language over Σ.