Homework 3

COMS W3261, Summer A 2022

This homework is due **Monday, 6/12/2023, 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.[1] 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.

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

1 Problem 1 (14 points)

1. (3 points). What is the language of the grammar G_1 below? Express this language as a set, with a sentence, or as a simple regular expression and explain your reasoning.

   (Note that here we use the abbreviated or 'rules-only' way to write a grammar. The variables $\{S, A, B, C\}$ can be read off the lefthand side, and the terminals $\{0, 1, 2, 3\}$ are the remaining symbols.)

$$S \to 0A \mid 1B \mid 2C$$
$$A \to 1B \mid 2C$$
$$B \to 2C$$
$$C \to 3$$

   This grammar generates a finite language that we can enumerate by trying all possible sequences of rules. Alternatively, we can simplify things slightly by observing that C is always replaced with the terminal 3, and thus B is always replaced with the terminal substring 23, to get the following grammar:

$$S \to 0A \mid 123 \mid 23$$
$$A \to 123 \mid 23$$

   At this point, we can try all possible sequences of rules, which gives the language

$$\{23, 123, 023, 0123\}.$$

2. (2 points). How does the language of G_1 change if we add the rule $S \to SS$ to the grammar?

   We have already seen that, if we *don't* use this rule, the variable S derives the set {23, 123, 023, 0123}.

   Thus any possible derivation of a terminal string using this grammar uses the rule $S \to SS$ a certain number of times, and then each S variable is replaced with a string from the set $\{23, 123, 023, 0123\}$.

   The new rule allows us to create any positive number of concatenated S variables, which lets us make any string in the set $\{23, 123, 023, 0123\}^+$.

3. (3 points). What is the language of the grammar G_2 below? Express this language as a set, with a sentence, or as a simple regular expression and explain your reasoning.

$$S \to AB$$

$$A \to 01A10 \mid 0$$
$$B \to 1B \mid \epsilon$$

The first rule guarantees that strings produced by this grammar will consist of two concatenated substrings: one derived from the variable A, and one derived from the variable B. The string derived from the variable A contains an equal number of 01 and 10 substrings, determined by the number of times we use the rule $A \to 01A10$, with a single 0 in the middle. The string derived from B can produce a substring of 1's of any length.

Thus the language of this grammar is thus

$$\{(01)^n 0 (10)^n 1^k \mid n, k \geq 0\},$$

equivalently, "the language of strings consisting of n repeated 01 substrings, for some $n \geq 0$, then a single 0, then n repeated 10 substrings, and finally k repeated 1 substrings for some $k \geq 0$".

4. (3 points). Design a grammar for the language

$$D = \{1^n 0^{2m} 1^m 0^{2n} \mid m, n \geq 2\}$$

and explain why your grammar produces D. (This language includes such strings as 110000110000.) You may use the brief representation of grammars (i.e., rules-only) or write out the full 4-tuple.

There are several ways to generate this language, but one is as follows:

$$S \to 11A0000$$
$$A \to 1A00 \mid 0000B11$$
$$B \to 00B1 \mid \epsilon$$

Note that we've used the rules $S \to 11A0000$ and $A \to 0000B11$ to ensure $m, n \geq 2$.

5. (3 points). Design a grammar for the language represented by the regular expression

$$R_1 = 1^+ \cup 0^+ \cup 1^*01^*$$

and explain why your grammar produces the same language. You may use the brief representation of grammars (i.e., rules-only) or write-out the full 4-tuple.

Here we can explicitly use our rules for building a CFG that matches a given regular expression, or just read R_1 carefully to understand its meaning and build accordingly.

R_1 describes a language in which strings match at least one of three sub-regular expressions: 1^+, 0^+, or 1^*01^*. Accordingly, we'll start by transforming our start variable into one of three variables and work from there.

$$S \to A \mid B \mid C$$

$$A \to 1A \mid 1$$
$$B \to 0B \mid 0$$
$$C \to D0D$$
$$D \to 1D \mid \epsilon$$

If we want, we can get clever and use even fewer rules. (For example, remove A and replace $S \to A$ with $S \to 1D$.)

Rationale: The goal of this question is to practice interpreting and building context-free grammars.

References: Sipser p. 102 and Lightning Review 5 (CFG definition and deriving strings), Sipser p.105 (figuring out the language of a CFG), and Sipser p.106-107 (tips for building CFGs).

2 Problem 2 (12 points)

1. (6 points.) Prove that the language

$$A = \{a^i b^j c^k \mid i + j \geq k\}$$

over the alphabet $\Sigma = \{a, b, c\}$ is nonregular using the pumping lemma.

First, assume for contradiction that A is regular. By the pumping lemma, under our assumption of regularity there exists some number p such that every string $s \in A$ with $s \geq p$ can be divided into 3 substrings x, y, and z such that (1) xy^iz is in the language for all $i \geq 0$, (2) $|y| > 0$, and (3) $|xy| \leq p$.

We'll choose the contradiction string $w = a^p b^p c^{2p}$. Since $|w| > p$ and $w \in A$, the pumping lemma tells us that w can be written as xyz where x, y, and z are substrings satisfying the pumping conditions.

To divide w in a way that satisfies (2) and (3), we must have xy consisting of only a symbols, y consisting of one or more a symbols. Thus $xy^0z = xz = a^{p-|y|}b^pc^{2p}$ must be in the language to satisfy (1). However, as $|y| > 0$ by (2), the number of a's and b's in $a^{p-|y|}b^pc^{2p}$ is less than the number of c's. Thus there is no way to partition this string to satisfy (1), (2), and (3) simultaneously.

Thus our assumption leads to a contradiction, and we can conclude that A does not satisfy the pumping lemma and is nonregular.

2. (6 points.) Prove that the language

$$B = \{a^i b^j c^k \mid i < j \text{ OR } i > k; \text{ also } i, j, k \geq 1\}$$

over the alphabet $\Sigma = \{a, b, c\}$ is nonregular using the pumping lemma.

First, assume for contradiction that B is regular. By the pumping lemma, under our assumption of regularity there exists some number p such that every string $s \in B$ with $s \geq p$ can be divided into 3 substrings x, y, and z such that (1) xy^iz is in the language for all $i \geq 0$, (2) $|y| > 0$, and (3) $|xy| \leq p$.

We'll choose the contradiction string $w = a^p b^{p+1} c^{2p}$. Since $|w| > p$ and $w \in B$, the pumping lemma tells us that w can be written as xyz where x, y, and z are substrings satisfying the pumping conditions.

To divide w in a way that satisfies (2) and (3), we must have xy consisting of only a symbols and y consisting of one or more a symbols. Thus $xy^2z = xz = a^{p+|y|}b^{p+1}c^{2p}$ must be in the language by (1). However, as $p > |y| > 0$ by (2) and (3), in $a^{p+|y|}b^{p+1}c^{2p}$ the number of a's is equal to or greater than the number of b's and less than or equal to the number of c's. Thus there is no way to partition this string to satisfy (1), (2), and (3) simultaneously.

Thus our assumption leads to a contradiction, and we can conclude that B does not satisfy the pumping lemma and is nonregular.

Rationale: The goal of this question is to practice using the pumping lemma to show that languages are nonregular.
References: Sipser p. 78-79 and Lightning Review 4 (the pumping lemma), Sipser p.80-82 and Lightning Review

5 (using the pumping lemma).

3 Problem 3 (6 points)

1. (5 points.) We've proved that the regular languages are closed under regular operations such as complement, union, concatenation and star: if we apply these operations to regular languages, we get a regular language. However, we have not proved that the *non*regular languages are closed under the regular operations.

   Suppose A and B are nonregular languages. Is their union $A \cup B$ guaranteed to be nonregular? If so, provide a proof. If not, provide a counterexample.

   No, this is not guaranteed. For example, consider the languages $A = \{0^n 1^m \mid n \geq m\}$ and $B = \{0^n 1^m \mid n \leq m\}$. Neither of these languages is regular, as we can show with pumping lemma proofs very similar to those we've used in class and previously in this homework. However, the union $A \cup B$ is the language $\{0^n 1^m \mid n, m \geq 0\}$, which is equivalent to the regular expression 0^*1^*.

2. (1 point.) Give an example of a language C such that $C \cap A$ is regular for any language A, whether A is regular or not. (You should be able to do this for an arbitrary alphabet Σ, but feel free to assume $\Sigma = \{0, 1\}$ for this question if you would like.)

   The empty set $\emptyset$ is the easiest example, but this is true for any finite language C: as long as C is finite, $A \cap C$ is finite, and all finite languages are regular. (To see this, observe that any finite language is a finite union of languages containing a single string.)

Rationale: The goal of this question is to practice reasoning about closure properties and (non)regularity.

References: Languages that we've previously proved to be nonregular (refer to previous questions in this HW, or the Lecture 4 notes) which can serve as examples. Recall that a regular language is any language recognized by some DFA; or equivalently, recognized by some NFA; or equivalently, expressed by some regular expression. A nonregular language is any language that *can't* be recognized/expressed in this way.

4 Problem 4 (8 points)

1. (4 points.) Consider the following "proof" of nonregularity, which contains a logical error:

   (a) Consider the language $A = \{x \in \{0,1\}^* \mid |x| \text{ is divisible by 3. }\}$. We'll assume for contradiction that A is regular.

   (b) By the pumping lemma, under our assumption there exists some number p such that every string $s \in A$ with $s \geq p$ can be divided into 3 substrings x, y, and z such that (1) xy^iz is in the language for all $i \geq 0$, (2) $|y| > 0$ and (3) $|xy| \leq p$.

   (c) We'll choose the contradiction string 0^{3p}, which has length $3p > p$ and is in the language. We'll show that it fails the conditions of the pumping lemma.

   (d) To satisfy condition (2), it must be true that y is a string containing at least one 0.

   (e) Consider $y = 00$. In this case, the string xy^2z has length $3p + 2$.

   (f) Since $|xyz| = 3p$ is divisible by 3, $|xy^2z| = 3p + 2$ is not divisible by 3 and thus xy^2z is not in the language.

   (g) Thus the string 0^{3p} cannot be pumped, which is a contradiction. Therefore A is nonregular.

   This can't be right: A is a regular language, as we've already seen. In what step does the error occur? Why is this proof invalid?

   The error here occurs in step (e), when we consider only one possibility for y: $y = 00$. To show that 0^{3p} fails the conditions of the pumping lemma, we need to prove that there is *no* way to divide 0^{3p} into x, y, z that can satisfy (1), (2), and (3).

   In this case, 0^{3p} can indeed be pumped: for example, choose $x = \epsilon, y = 000, z = 0^{3p-3}$.

2. (4 points.) The following proof also contains a logical error. In what step does it occur, and why is this proof invalid?

   (a) Consider the language $B = \{a^ib^jc^k \mid i \leq j \text{ OR } j < k \text{ OR } k < i\}$. We'll assume for contradiction that B is regular.

   (b) By the pumping lemma, under our assumption there exists some number p such that every string $s \in A$ with $s \geq p$ can be divided into 3 substrings x, y, and z such that (1) xy^iz is in the language for all $i \geq 0$, (2) $|y| > 0$ and (3) $|xy| \leq p$.

   (c) We'll choose the contradiction string $a^pb^pc^p$. which has length $3p > p$. Moreover, $a^pb^pc^p$ is in the language because the number of a's is less than or equal to the number of b's. We'll show that it fails the conditions of the pumping lemma.

   (d) To satisfy conditions (2) and (3), it must be true that y is a string containing at least one a, and only a's.

   (e) Note that $a^pb^pc^p$ is in the language because it satisfies the first of the three OR'ed conditions in the language definition: if we set $i, j, k = p$, it is true that $i \leq j$ but not true that $j < k$ or $k < i$.

(f) Now consider the string $xy^2z = a^{p+|y|}b^pc^p$. Since it is no longer true that $i \leq j$, this string is not in the language.

(g) Thus the string $a^pb^pc^p$ cannot be pumped, which is a contradiction. Therefore B is nonregular.

The error here comes in step (f), where the analysis of $xy^2z = a^{p+|y|}b^pc^p$ is incomplete. If $i = p + |y|$, and $j, k = p$, it is now true that $k < i$.

As it turns out, this language is regular and the proof can't be salvaged. To see this, consider the set of strings of the form $a^ib^jc^k$ that *aren't* in the language: for such a string, we must have $i > j > k \geq i$, which is impossible. In fact, $B = a^*b^*c^*$.

Rationale: The goal of this question is to practice the pumping lemma from a different angle: common issues that arise when working through PL proofs.

References: Sipser p. 78-79 and Lightning Review 4 (the pumping lemma), Sipser p.80-82 and Lightning Review 5 (using the pumping lemma).

5 Problem 5 (1 bonus point)

The Myhill-Nerode theorem says that a language L is a regular language if and only if L has a finite number of equivalence classes (i.e., L would not be a regular language if it had an infinite number of equivalence classes). Because it gives us an 'if and only if' condition, it's more powerful than the pumping lemma.

Consider the language L, defined over the alphabet $\Sigma = \{0, 1\}$:

$$E = \{w \mid w \text{ starts or ends with the substring } 0\}.$$

Is E regular or nonregular? Prove your claim using the Myhill-Nerode theorem. (Hint: You should define all the equivalence classes for E in terms of distinguishing extensions, or prove that there are an infinite number of equivalence classes under this relation.)

Yes, E is regular. Consider the classes

1. w begins in 0,
2. w begins with 1 and ends in 0,
3. w begins with 1 and ends in 1,
4. $w = \epsilon$.

These three classes partition $\{0, 1\}^*$. Moreover, we claim that pairs of strings in each class are indistinguishable from each other, so the three classes are in fact equivalence classes.

For any two strings x and y in the first equivalence class, any extension z creates a string in the language. For two strings in the second equivalence class, an extension z creates a string in the language if z ends in 0 OR if z is the empty string. For two strings in the third equivalence class, an extension z creates a string in the language if z ends in 0 (and not if z is the empty string). Finally, the fourth equivalence class contains only the empty string, so no pair of strings can be distinguished.

Because this is a finite number of equivalence classes, by the Myhill-Nerode theorem, L must be regular.

Rationale: Optional, just for fun. The bonus point will add 1 to your total score on this HW, which is out of 40.
Resources: Wiki on the Myhill-Nerode theorem and equivalence classes.