Problem Set 6

What are the limits of regular languages? What are the powers of context-free grammars? In this problem set, you'll find out.

As always, please feel free to drop by office hours, ask on Piazza, or send us emails if you have any questions. We'd be happy to help out.

This problem set has 36 possible points. It is weighted at 5% of your total grade.

Good luck, and have fun!

Due Monday, May 18th at the start of lecture.

Problem One: The Myhill-Nerode Theorem (5 Points)

The Myhill-Nerode theorem is one of the trickier and more nuanced theorems we've covered this quarter. This question explores what the theorem means and, importantly, what it *doesn't* mean.

Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let $L = \{ w \in \Sigma^* \mid |w| \text{ is even} \}$.

i. Show that L is a regular language.

ii. Prove that there is a infinite set $S \subseteq \Sigma^*$ where there are infinitely many pairs of distinct strings $x, y \in S$ such that $x \not\equiv_L y$.

iii. Without using the Myhill-Nerode theorem, prove that there is *no* infinite set $S \subseteq \Sigma^*$ where *all* pairs of distinct strings $x, y \in S$ satisfy $x \not\equiv_L y$.

The distinction between parts (ii) and (iii) is important for understanding the Myhill-Nerode theorem. A language is nonregular not if you can find infinitely many pairs of distinguishable strings, but rather if you can find infinitely many strings that are all *pairwise* distinguishable. This is a subtle distinction, but it's a critically important one.

Problem Two: Balanced Parentheses (6 Points)

Consider the following language over $\Sigma = \{\mathtt{(}, \mathtt{)}\}$:

$$L_1 = \{ w \in \Sigma^* \mid w \text{ is a string of balanced parentheses} \}$$

This question explores properties of this language.

i. Prove that L_1 is not a regular language.

Let's say that the *nesting depth* of a string of balanced parentheses is the maximum number of unmatched open parentheses at any point inside the string. For example, the string `(((())))` has nesting depth three, the string `(()())()` has nesting depth two, and the string ε has nesting depth zero.

Consider the language $L_2 = \{ w \in \Sigma^* \mid w \text{ is a string of balanced parentheses and } w\text{'s nesting depth is at most four} \}$. For example, `(((())))` $\in L_2$, `(()())` $\in L_2$, and `((((()))))(())` $\in L_2$, but `((((()))))` $\notin L_2$ because although it's a string of balanced parentheses, the nesting goes five levels deep.

ii. Design a DFA for L_2, showing that L_2 is regular.

iii. Since L_2 is regular, it's also context-free. Design a context-free grammar for L_2.

Problem Three: Subsets of Regular Languages (6 Points)

The first nonregular language we encountered was the language $\{ \mathbf{a}^n\mathbf{b}^n \mid n \in \mathbb{N} \}$. This problem explores a related language and some of its properties.

Let $\Sigma = \{ \mathbf{a}, \mathbf{b} \}$ and let $L = \{ w \in \Sigma^* \mid w$ has the same number of **a**'s and **b**'s $\}$.

i. Prove that L is not a regular language.

ii. Find a language $L' \subseteq L$ such that L' contains infinitely many strings and L' is a regular language. Justify why L' is infinite and show that it's regular.

iii. Prove that there is no language $L' \subseteq \{ \mathbf{a}^n\mathbf{b}^n \mid n \in \mathbb{N} \}$ that contains infinitely many strings and is a regular language.

Your results from parts (ii) and (iii) show that if a language is nonregular, it *might* contain an infinite regular language as a subset, but it might not. It really depends on the choice of language.

Problem Four: State Lower Bounds (5 Points)

The Myhill-Nerode theorem we proved in lecture is actually a special case of a more general theorem about regular languages that can be used to prove lower bounds on the number of states necessary to construct a DFA for a given language.

i. Let L be a language over Σ. Suppose there's a *finite* set S such that any two distinct strings $x, y \in S$ are distinguishable relative to L (that is, $x \not\equiv_L y$). Prove that any DFA for L must have at least $|S|$ states.

ii. Let $\Sigma = \{ \mathbf{a}, \mathbf{b} \}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_4 2 \}$. Design the smallest possible DFA for L. Then, using the theorem you proved in part (i), prove that the DFA you designed is the smallest possible DFA for L.

Problem Five: Closure Properties Revisited (4 Points)

When building up the regular expressions, we explored several closure properties of the regular languages. This problem explores some of their nuances.

The regular languages are closed under complementation: If L is regular, so is $\overline{L}$.

i. Prove or disprove: the *nonregular* languages are closed under complementation.

The regular languages are closed under union: If L_1 and L_2 are regular, so is $L_1 \cup L_2$.

ii. Prove or disprove: the *nonregular* languages are closed under union.

We know that the union of any two regular languages is regular. Using induction, we can show that the union of any finite number of regular languages is also regular. As a result, we say that the regular languages are closed under *finite union*.

An *infinite union* is the union of infinitely many sets. For example, the rational numbers can be expressed as the infinite union $\{ {}^x/_1 \mid x \in \mathbb{Z} \} \cup \{ {}^x/_2 \mid x \in \mathbb{Z} \} \cup \{ {}^x/_3 \mid x \in \mathbb{Z} \} \cup \ldots$ out to infinity.

iii. Prove or disprove: the regular languages are closed under infinite union.

Problem Six: Designing CFGs (10 Points)

Below are a list of alphabets and languages over those alphabets. For each language, design a context-free grammar that generates that language, then show derivations for the indicated strings.

i. Given $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$, write a CFG for the language $\{ w \in \Sigma^* \mid w$ contains **aa** as a substring $\}$. Show a derivation of the strings **aa**, **baac**, and **ccaabb** using your grammar.

ii. Given $\Sigma = \{\texttt{a}, \texttt{b}\}$, write a CFG for the language $L = \{ w \in \Sigma^* \mid w$ is ***not*** a palindrome $\}$. That is, w is not the same when read forwards and backwards, so **aab** $\in L$ and **baabab** $\in L$, but **aba** $\notin L$ and **bb** $\notin L$. Show derivations of **aab** and **abbaba**.

iii. Given $\Sigma = \{\texttt{1}, \texttt{+}, \stackrel{?}{=}\}$, write a context-free grammar for the language $\{ \texttt{1}^m\texttt{+}\texttt{1}^n\stackrel{?}{=}\texttt{1}^{m+n} \mid m, n \in \mathbb{N} \}$. Show derivations for **111+1**$\stackrel{?}{=}$**1111** and for **+1**$\stackrel{?}{=}$**1**.

iv. Given $\Sigma = \{\texttt{a}, \texttt{b}\}$, write a CFG for the language $L = \{ w \in \Sigma^* \mid |w| \equiv_4 0$, and the first quarter of the characters in w contains at least one $b \}$. For example, **baaa** $\in L$, **bbbb** $\in L$, **abbbbbba** $\in L$, **bbbaaabbbaaa** $\in L$, **abbbbbbbbbb** $\in L$, but **abbb** $\notin L$, $\varepsilon \notin L$, **b** $\notin$ L, **aabbbbaa** $\notin L$, and **aaabbbbbbbbb** $\notin L$. (For simplicity, I've underlined the first quarter of the characters in each string). Show a derivation of **baaa**, **abaaaaaa**, and **baaaaaaa**.

v. Let's imagine that you're going for a walk with your dog, but this time don't have a leash. As in Problem Set Five, let $\Sigma = \{\texttt{y}, \texttt{d}\}$, where **y** means that you take a step forward and **d** means that your dog takes a step forward. A string in Σ^* can be thought of as a series of events in which either you or your dog moves forward one unit. For example, the string "**yydd**" means that you take two steps forward, then your dog takes two steps forward. Let $L = \{ w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$. Write a CFG that generates L. Show derivations of the strings **yyyddd**, **ydyd**, and **yyddddyy** using your grammar.

Extra Credit Problem: Voting Machines (1 Point Extra Credit)

You are part of a team working on designing a voting machine for a general election. You are in charge of designing the software systems necessary to ensure that no one can vote twice. (For the purposes of this problem, let's imagine that someone else is tasked with recording vote totals.) Specifically, after everyone finishes casting votes, you would like to be able to inspect the machine and determine, with 100% certainty, whether anyone voted twice. You don't need to report *who* voted twice, just *whether* anyone voted twice.

Because people will be coming to vote at the machine all throughout election day, the voting machine needs to have some kind of persistent storage space to record information about who has voted so far. It also needs some kind of "working memory" for other tasks like letting the user choose who to vote for, displaying information, etc. For the purposes of this problem, we're only interested in the persistent storage space.

Prove that if there are $n \geq 1$ people in the electorate and the voting machine behaves correctly on all inputs, it must have at least $n+1$ bits of memory in its persistent storage.