Problem Set 7

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

We have an online tool you can use to design, test, and submit the CFGs in this problem set. To use it, visit the CS103 website and click the "CFG Editor" link under the "Resources" header. We strongly recommend this tool, as it makes it easy to design, test, and submit your solutions. If you submit through this system, please have only one team member submit your grammars, and include the name of that person in your submission.

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.

Good luck, and have fun!

Due Friday, February 26 at the start of class.

Problem One: The Myhill-Nerode Theorem (4 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 $\Sigma = \{\texttt{a}, \texttt{b}\}$ 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. 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 an important one!

Problem Two: Balanced Parentheses (5 Points)

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

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

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 $L_2 = \{ w \in \Sigma^* \mid w$ is a string of balanced parentheses and w'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. Look back at your proof from part (i) of this problem. Why doesn't that proof show that L_2 is nonregular? Be specific.

Problem Three: Subsets of Regular Languages (5 Points)

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

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L = \{ w \in \Sigma^* \mid w$ has the same number of $\texttt{a}$'s and $\texttt{b}$'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 $L' \subseteq \{ \texttt{a}^n\texttt{b}^n \mid n \in \mathbb{N} \}$ 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 $\Sigma = \{\texttt{a}, \texttt{b}\}$ 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 (3 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 (11 Points)

Below are a list of alphabets and languages over those alphabets. For each language, design a context-free grammar that generates that language. **Please use our online tool to design, test, and submit the CFGs in this problem**. To use it, visit the CS103 website and click the "CFG Editor" link under the "Resources" header. You should only have one member from each team submit your grammars; tell us who this person is when you submit the rest of the problems through GradeScope.

i. Given $\Sigma = \{$a, b, c$\}$, write a CFG for the language $\{ w \in \Sigma^* \mid w$ contains aa as a substring $\}$. For example, the strings aa, baac, and ccaabb are all in the language, but aba is not.

ii. Given $\Sigma = \{$a, b$\}$, write a CFG for the language $L = \{ w \in \Sigma^* \mid w$ is a palindrome and $|w| \geq 10 \}$. (Recall that a palindrome is a string that's the same when read forwards and backwards.) For example, we have abaaaaaaba $\in L$ and abbbabababbba $\in L$, but abba $\notin L$ and bbbbbbbbba $\notin L$.

iii. Given $\Sigma = \{$a, 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 $\notin L$.

iv. Given $\Sigma = \{$1, +, $\overset{?}{=}\}$, write a context-free grammar for the language $\{ 1^m+1^n\overset{?}{=}1^{m+n} \mid m, n \in \mathbb{N} \}$. For example, the strings 111+1$\overset{?}{=}$1111 and +1$\overset{?}{=}$1 are in the language, but 1+11$\overset{?}{=}$11 is not, nor is the string 1+1+1=111.

v. Given $\Sigma = \{$a, 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, $\underline{\texttt{b}}\texttt{aaa} \in L$, $\underline{\texttt{b}}\texttt{bbb} \in L$, $\underline{\texttt{ab}}\texttt{bbbbba} \in L$, $\underline{\texttt{bbb}}\texttt{aaabbbaaa} \in L$, $\underline{\texttt{aba}}\texttt{bbbbbbbbb} \in L$, but $\underline{\texttt{a}}\texttt{bbb} \notin L$, $\varepsilon \notin L$, b $\notin$ L, $\underline{\texttt{aa}}\texttt{bbbbaa} \notin L$, and $\underline{\texttt{aaa}}\texttt{bbbbbbbbb} \notin L$. (For simplicity, I've underlined the first quarter of the characters in each string).

vi. 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 $\Sigma = \{$y, 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 $\}$. For example, the strings yyyddd, ydyd, and yyydddddyyy are all in the language. Write a CFG for L.

Problem Seven: Right-Linear Grammars (4 Points)

A context-free grammar is called a ***right-linear grammar*** if every production in the grammar has one of the following three forms:

- A → ε
- A → B, where B is a nonterminal.
- A → aB, where a is a terminal and B is a nonterminal.

For example, the following is a right-linear grammar:

A → aB | bB | ε

B → aC | bA | C

C → bA | aA | ε

The right-linear grammars are all context-free grammars, so their languages are all context-free. However, it turns out that this class of grammars precisely describe the regular languages. That is, a language L is regular if and only if there is a right-linear grammar G such that $L = \mathscr{L}(G)$.

i. Let G be a right-linear grammar. Describe how to construct an NFA N such that $\mathscr{L}(G) = \mathscr{L}(N)$. You don't need to formally prove that your construction is correct, but you should give a good intuitive explanation as to why the grammar has the same language as the generated NFA. Additionally, to illustrate your construction, show the NFA you'd construct from the following right-linear grammar:

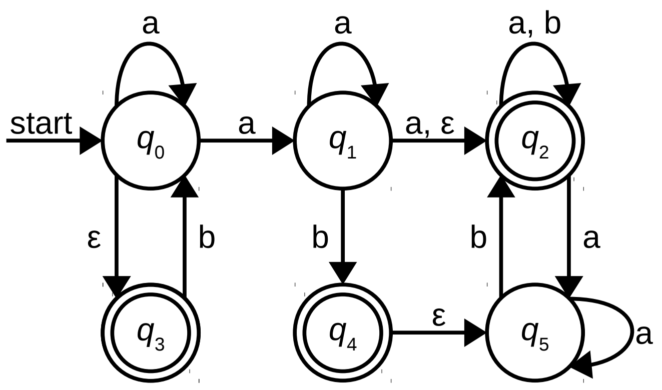
A → aB | bC

B → aB | ε

C → aD | A | bC

D → aD | bD | ε

ii. Let N be an NFA. Describe how to construct a right-linear grammar G such that $\mathscr{L}(G) = \mathscr{L}(N)$. You don't need to formally prove that your construction is correct, but you should give a good intuitive explanation as to why the generated grammar has the same language as the NFA. Additionally, to illustrate your construction, show the grammar that you'd construct from the following NFA:



Extra Credit Problem: NFA State Lower Bounds (1 Point Extra Credit)

As you saw in Problem Four, the Myhill-Nerode theorem can be used to prove lower bounds on the size of any DFA for a particular language. However, the conditions given by the Myhill-Nerode theorem are not powerful enough to lower-bound the size of an NFA for a particular language. (If you're not sure why this is, try finding a counterexample). This question explores a variation on the Myhill-Nerode theorem that can be used to lower-bound NFA sizes.

Let L be a language over Σ. A ***generalized fooling set*** for L is a set $\mathscr{F} \subseteq \Sigma^* \times \Sigma^*$ is a set with the following properties:

- For any $(x, y) \in \mathscr{F}$, we have $xy \in L$.
- For any distinct pairs (x_1, y_1), $(x_2, y_2) \in \mathscr{F}$, we have $x_1y_2 \notin L$ or $x_2y_1 \notin L$ (this is an inclusive OR.)

Prove that if there is a generalized fooling set $\mathscr{F}$ for the language L that contains n pairs of strings, then any NFA for L must have at least n states.