Problem Set 6

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

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

This problem set has 125 possible points. It is weighted at 7% of your total grade. The earlier questions serve as a warm-up for the later problems, so do be aware that the difficulty of the problems does increase over the course of this problem set.

Good luck, and have fun!

Due Monday, November 11th at 2:15

Problem One: The Complexity of Addition (16 Points)

This problem explores the question

How hard is it to add two numbers?

Suppose that we want to check whether $x + y = z$, where x, y, and z are all natural numbers. If we want to phrase this as a problem as a question of strings and languages, we will need to find some way to standardize our notation. In this problem, we will be using the ***unary number system***, a number system in which the number n is represented by writing out n `1`'s. For example, the number 5 would be written as `11111`, the number 7 as `1111111`, and the number 12 as `111111111111`. Given the alphabet $\Sigma = \{\texttt{1}, \texttt{+}, \stackrel{?}{=}\}$, we can encode $x + y = z$ by writing out x, y, and z in unary. For example:

4 + 3 = 7 would be encoded as `111+1111`$\stackrel{?}{=}$`1111111`

7 + 1 = 8 would be encoded as `1111111+1`$\stackrel{?}{=}$`11111111`

0 + 1 = 1 would be encoded as `+1`$\stackrel{?}{=}$`1`

Consider the language $ADD = \{\texttt{1}^m\texttt{+1}^n \stackrel{?}{=} \texttt{1}^{m+n} \mid m, n \in \mathbb{N}\}$. That is, *ADD* consists of strings encoding two unary numbers and their sum.

i. Prove that *ADD* is not a regular language.

ii. Write a context-free grammar for *ADD*. Show a derivation of `1+1`$\stackrel{?}{=}$`11` and `+111`$\stackrel{?}{=}$`111` using your grammar. This proves that *ADD* is context-free. *(If you are hand-writing your grammar, please make sure your ones are easily distinguished from the vertical bars used to separate productions. Thanks!)*

Problem Two: Minimum-State DFAs (28 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.

i. Suppose that S is a finite set of n strings distinguishable relative to L. Prove that any DFA for L must have at least n states. *(Hint: Start with the proof of Myhill-Nerode from lecture and make appropriate modifications.)*

ii. In Problem Set 5, you build an NFA for the language $L = \{ w \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}^* \mid$ the last character of w appears nowhere else in the string, and $|w| \geq 1 \}$. Prove that any DFA for this language must have at least 32 states.

iii. In Problem Set 5, you built an NFA for the language $L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w$ contains at least two `1`'s with exactly five characters between them. $\}$ Prove that any DFA for this language must have at least 64 states.

Problem Three: Primes are Nonregular (20 Points)

A natural number $n > 1$ is called *composite* iff it can be written as $n = rs$ for natural numbers r and s, where $r \geq 2$ and $s \geq 2$. A natural number $n > 1$ is called *prime* iff it is not composite.

Let $\Sigma = \{ \mathtt{a} \}$ and consider the language $L = \{ \mathtt{a}^n \mid n \text{ is prime} \}$. For example:

- $\varepsilon \notin L$
- $\mathtt{a} \notin L$
- $\mathtt{a}^2 \in L$
- $\mathtt{a}^3 \in L$
- $\mathtt{a}^4 \notin L$
- $\mathtt{a}^5 \in L$
- $\mathtt{a}^6 \notin L$
- $\mathtt{a}^7 \in L$
- $\mathtt{a}^8 \notin L$

Your goal of this problem is to show that L isn't regular. You can do so with the Myhill-Nerode theorem, though it's a bit tricky. Therefore, we'll walk you through a few smaller steps before building up to the result.

i. Let p be a prime number and let k be some positive natural number. Prove that there is a natural number n for which $p + kn$ is composite.

ii. Let p_1 and p_2 be prime numbers where $p_1 < p_2$. Prove that there is some value of n for which the number $p_1 + n(p_2 - p_1)$ is prime but $p_2 + n(p_2 - p_1)$ is composite.

As a hint: you know that there is at least one n for which $p_2 + n(p_2 - p_1)$ is composite, but there might many difference choices of n with this property. Try letting n be the *smallest* possible choice of n for which $p_2 + n(p_2 - p_1)$ is composite.

iii. Prove that L is not a regular language. *(Hint: There are infinitely many prime numbers.)*

Problem Four: Designing CFGs (24 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. For the alphabet $\Sigma = \{ \mathtt{a}, \mathtt{b}, \mathtt{c} \}$, write a CFG for the language $L = \{ w \in \Sigma^* \mid w$ contains **aa** as a substring $\}$. Show a derivation of the strings **aa**, **abaabc**, and **ccaabb** using your grammar.

ii. For the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, 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. For the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, 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** $\in L$, **ababbbbbbbbb** $\in L$, 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**.

iv. Suppose that you live in a one-dimensional world with your house at position 0. One day, you decide to go for a walk by taking steps of size ±1 forward and backward. You begin at your house and, after completing your walk, end up back at your house. Consider the alphabet $\Sigma = \{\mathtt{l}, \mathtt{r}\}$. A string in Σ^* describes a possible walk where at each step you either move a step left (**l**) or a step right (**r**). For example, the string "**lrrrll**" means that you take a step left, then three steps right, then two steps left. Let $L = \{ w \in \Sigma^* \mid w$ describes a series of steps in which you arrive at the same place at which you started $\}$. Write a CFG that generates L. Show derivations of **llrr**, **rlrl**, and **llrrrrll** using your grammar.

Problem Five: Testing Emptiness (12 Points)

Suppose that you have a magic machine (an oracle) that takes as input two CFGs over the same alphabet Σ and returns whether every string generated by the first CFG is also generated by the second CFG. In other words, given two grammars G_1 and G_2 over the same alphabet Σ, the magic machine determines whether $\mathscr{L}(G_1) \subseteq \mathscr{L}(G_2)$. What else could you do with this machine?

i. Suppose you want to test whether G never generates any strings (that is, whether $\mathscr{L}(G) = \varnothing$). Describe how you could use the oracle to decide whether G generates no strings.

ii. Formally prove that your procedure from (i) is correct by proving the following: your procedure reports that $\mathscr{L}(G) = \varnothing$ iff it's actually true that $\mathscr{L}(G) = \varnothing$.

Problem Six: Ambiguous Grammars (20 Points)

In Friday's lecture, you saw the following unambiguous grammar for arithmetic expressions involving addition, subtraction, multiplication, and division:

$$E \rightarrow T \mid T \texttt{ + } E \mid T \texttt{ - } E$$

$$T \rightarrow \texttt{int} \mid \texttt{int * } T \mid \texttt{int / } T$$

i. Suppose that you want to add exponentiation to the grammar so that you can write expressions like `int * int ^ int + int * int + int`. The exponentiation operator `^` should bind more tightly than any other operation and should be *right-associative*, so `int ^ int ^ int` should be interpreted as `int ^ (int ^ int)`. Modify the above grammar to add exponentiation that is right-associative. Make sure that your grammar is still unambiguous, and show a parse tree for `int ^ int ^ int * int + int`.

ii. Now, suppose you want to allow for explicit parenthesization of arithmetic expressions to indicate operator precedence. For example, `(int + int) * int` should be parsed to mean "add the first two `int`s, then multiply the result by an integer." Modify the grammar you developed in part (i) to support parenthesized expressions. Make sure that your grammar is still unambiguous, and give a parse tree for `int ^ (int + int * int) ^ int`.

Problem Seven: Course Feedback (5 Points)

We want this course to be as good as it can be, and we'd really appreciate your feedback on how we're doing. For a free five points, please answer the feedback questions for this problem set, available online at https://docs.google.com/forms/d/1iqZyIOb3KFleK9-9txLGwostvM1l5IEWj-S1A3_Fwdk/viewform. We'll give you full credit no matter what you write (as long as you answer each question), but we'd appreciate it if you were honest in your answers.

Extra Credit Problem: The Power of Regular Languages (5 Points)

For any set $S \subseteq \mathbb{N}$, define $L_S = \{ \texttt{a}^n \mid n \in \mathbb{N}$ and $m \in S$ for all $m \geq n$. $\}$ Prove L_S is regular for any $S \subseteq \mathbb{N}$.