CSE 311: Foundations of Computing I

Homework 7 (due November 22nd at 11:00 PM)

Directions: *Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. However, you may use results from lecture, the theorems handout, and previous homeworks without proof.*

1. Up the Ladder to the Proof (20 points)

Consider the following grammar

$$\mathbf{S} \to \mathbf{SS} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0 \mid \varepsilon$$

It is not hard to check that every string generated by this grammar has an equal number of 0s and 1s. (This is easy to prove using structural induction.) In this problem, you will prove that every string with an equal number of 0s and 1s can be generated by this grammar. Those two facts, together, say that the language defined by this grammar is exactly the set of binary strings with an equal number of 0s and 1s.

Prove, by strong induction, that every binary string of length n with an equal number of 0s and 1s can be generated by the grammar above. (If true for all n, then every binary string with an equal number of 0s and 1s can be generated by the grammar since every string has some length n.)

Notation: The string x is generated by the grammar above iff there is a sequence of substitutions $\mathbf{S} \Rightarrow \cdots \Rightarrow x$ turning the string "**S**" into x. Feel free to use the "$\Rightarrow \cdots \Rightarrow$" notation in your proof if it helps.

Extended Hint

For this proof, some machinery may be helpful. Let $x \in \{0,1\}^*$ be a string with an equal number of 0s and 1s. Write the string in terms of its individual characters as $x = x_1x_2\cdots x_n$, where each $x_i \in \{0,1\}$ and n is the length of x. We then define the function $f_x(k)$ to be the number of 1s minus the number of 0s in $x_1x_2\cdots x_k$ (the length k *prefix* of x). Observe that $f_x(0) = 0$, since $x_1x_2\cdots x_k = \varepsilon$ when $k = 0$, and that $f_x(n) = 0$, since $x_1x_2\cdots x_n = x$ has an equal number of 0s and 1s (by assumption).

Hint 1: Prove the Inductive Step in three cases. The first case is $\forall j \in [k]\ (f_x(j) > 0)$, the second case is $\forall j \in [k]\ (f_x(j) < 0)$, and the third case is when neither of those holds.

Hint 2: You may use without proof the following fact about any function g with integer inputs and outputs and the property that $|g(k) - g(k+1)| \leq 1$ for all k: if there exists an i such that $g(i) < 0$ and a j such that $g(j) > 0$, then there exists an ℓ, between i and j, such that $g(\ell) = 0$. (Note that the function f_x, above, has this property for any string x: it increases by 1 if $x_{k+1} = 1$ and decreases by 1 if $x_{k+1} = 0$. So this fact tells us that, if f_x is negative at some point and positive at another, then it must be zero somewhere in between.)

2. Extra Credit 1: Zero Hour (0 points)

Prove the fact you were allowed to use in Problem 1: for any function $g : \mathbb{N} \to \mathbb{Z}$ with the property that $|g(k) - g(k+1)| \leq 1$ for all $k \in \mathbb{N}$, if there exists an i such that $g(i) < 0$ and a j such that $g(j) > 0$, then there exists an ℓ such that $g(\ell) = 0$.

3. Grammar School (18 points)

For each of the following, construct context-free grammars that generate the given set of strings.

If your grammar has more than one variable, we will ask you to write a sentence describing what sets of strings you expect each variable in your grammar to generate. For example, if your grammar was:

$$\begin{aligned} \mathbf{S} &\to \mathbf{E} \mid \mathbf{O} \\ \mathbf{O} &\to \mathbf{EC} \\ \mathbf{E} &\to \mathbf{EE} \mid \mathbf{CC} \\ \mathbf{C} &\to 0 \mid 1 \end{aligned}$$

You could say "C generates binary strings of length one, E generates (non-empty) even length binary strings, and O generates odd length binary strings." It is also fine to use a regular expression, rather than English, to describe the strings generated by a variable (assuming such a regular expression exists).

(a) [6 Points] Binary strings matching the regular expression "$000(1 \cup 01 \cup 001)^*$".

Hint: You can use the procedure described in lecture to convert the RE to a CFG.

(b) [6 Points] All strings over $\{0, 1, 2\}$ of the form $x2y$, with $x, y \in \{0, 1\}^*$ and y a subsequence of x^R (i.e., it is x^R with some characters possibly removed).

(c) [6 Points] All strings over $\{0, 1, 2\}$ where the number of 0s is the same as the number of 1s and there is exactly one 2.

Hint: Try modifying the grammar from problem 1. (You may need to introduce new variables.)

4. Extra Credit 2: As If (0 points)

Consider the following context-free grammar.

$$\begin{aligned} \langle \textbf{Stmt} \rangle &\to \langle \textbf{Assign} \rangle \mid \langle \textbf{IfThen} \rangle \mid \langle \textbf{IfThenElse} \rangle \mid \langle \textbf{BeginEnd} \rangle \\ \langle \textbf{IfThen} \rangle &\to \text{if condition then } \langle \textbf{Stmt} \rangle \\ \langle \textbf{IfThenElse} \rangle &\to \text{if condition then } \langle \textbf{Stmt} \rangle \text{ else } \langle \textbf{Stmt} \rangle \\ \langle \textbf{BeginEnd} \rangle &\to \text{begin } \langle \textbf{StmtList} \rangle \text{ end} \\ \langle \textbf{StmtList} \rangle &\to \langle \textbf{StmtList} \rangle \langle \textbf{Stmt} \rangle \mid \langle \textbf{Stmt} \rangle \\ \langle \textbf{Assign} \rangle &\to \texttt{a := 1} \end{aligned}$$

This is a natural-looking grammar for part of a programming language, but unfortunately the grammar is "ambiguous" in the sense that it can be parsed in different ways (that have distinct meanings).

(a) [0 Points] Show an example of a string in the language that has two different parse trees that are meaningfully different (i.e., they represent programs that would behave differently when executed).

(b) [0 Points] Give **two different grammars** for this language that are both unambiguous but produce different parse trees from each other.

5. All Your Base (21 points)

Recall our definition of the set **Lists** from Homework 6:

Basis Elements: null $\in$ **Lists**.

Recursive Step: for any $x \in \mathbb{Z}$, if $L \in$ **Lists**, then Node$(x, L) \in$ **Lists**.

In this problem, we consider using these lists to store arbitrary-size integers. Specifically, we will store the individual digits of the base-b representation of the number in a list.[1]

The following function, value$_b$, takes a list containing the digits of some number, written in base b, and returns its actual value as an integer:

$$\begin{aligned} \text{value}_b(\text{null}) &= 0 \\ \text{value}_b(\text{Node}(x, L)) &= x + b \cdot \text{value}_b(L) \qquad \text{for any } x \in \mathbb{Z} \text{ and } L \in \textbf{Lists} \end{aligned}$$

(a) [3 Points] Calculate value$_{10}$(Node$(1,$ Node$(9,$ Node$(3,$ null$))))$. Show your work.

Next, we consider functions that perform arithmetic on numbers stored in lists. The function add$_b(L, y)$ takes the number whose base-b digits are stored in the list L and an arbitrary integer y and returns a list containing the base-b digits of their sum. The function mult$_b(L, r)$ similarly returns a list containing the base-b digits of the product of the number stored in L and the integer r.

Rather than defining these (familiar) functions, we instead describe properties that they should have:

$$\begin{aligned} \text{value}_b(\text{add}_b(L, y)) &= \text{value}_b(L) + y \qquad \text{for any } y \in \mathbb{Z} \text{ and } L \in \textbf{Lists} \qquad (\textbf{Fact A}) \\ \text{value}_b(\text{mult}_b(L, r)) &= \text{value}_b(L) \cdot r \qquad \text{for any } r \in \mathbb{Z} \text{ and } L \in \textbf{Lists} \qquad (\textbf{Fact B}) \end{aligned}$$

(b) [1 Point] Briefly explain why any correct implementation of add$_b$ and mult$_b$ should satisfy **Facts A-B**.

Below, we will define a function, convert$_{b\to c}$, that converts a number stored as its base-b digits into the same number stored as its base-c digits.

(c) [1 Point] Briefly explain why a correct implementation of convert$_{b\to c}$ should have the property that value$_c$(convert$_{b\to c}(L)$) = value$_b(L)$ for all $L \in$ **Lists**.

Now, we define convert$_{b\to c}$ as follows:

$$\begin{aligned} \text{convert}_{b\to c}(\text{null}) &= \text{null} \\ \text{convert}_{b\to c}(\text{Node}(x, L)) &= \text{add}_c(\text{mult}_c(\text{convert}_{b\to c}(L), b), x) \qquad \text{for any } x \in \mathbb{Z} \text{ and } L \in \textbf{Lists} \end{aligned}$$

(d) [16 Points] Let $b, c \geq 2$ be integers. Prove that value$_c$(convert$_{b\to c}(L)$) = value$_b(L)$ for all $L \in$ **Lists**. You may assume that add and mult are implemented correctly, so they satisfy **Facts A-B**.

[1] Here and below, all bases are integers greater than or equal to 2.

6. Acting Up (16 points)

Consider the set of all actors and actresses listed in the Internet Movie Database (IMDB). In each part of this problem, we define a relation R on this set. For each one, state whether R is or is not reflexive, symmetric, antisymmetric, and/or transitive. (No proofs.)

(a) [4 Points] $(a, b) \in R$ iff a and b were in the same movie.

(b) [4 Points] $(a, b) \in R$ iff a and b were in *none* of the same movies.

(c) [4 Points] $(a, b) \in R$ iff a and b were in *all* of the same movies.

(d) [4 Points] $(a, b) \in R$ iff the total (sum) of the box office revenues from every movie a was in was greater than the total box office revenues from every movie b was in.

7. Machine Shop [Online] (25 points)

For each of the following, create a *DFA* that recognizes exactly the language given.

(a) [5 Points] Binary strings such that none of the substrings of adjacent 1s (i.e. "$11\cdots1$") have odd length.[2]

(b) [5 Points] Binary strings that have at least one 1 and an even number of 0s.

(c) [5 Points] Binary strings with at least two 1s.

(d) [5 Points] Binary strings with at least two 1s **or** at most two 0s but **not both**.

(e) [5 Points] Binary strings not containing the substring 1011.

Submit and check your answers to this question here:

`https://grinch.cs.washington.edu/cse311/fsm`

Think carefully about your answer to make sure it is correct before submitting. You have only 5 chances to submit a correct answer.

[2] Here, I mean only substrings containing *all* the adjacent 1s. For example, "011110" is included because the substring of adjacent 1s has length 4. Technically, it also has "111" as a substring, but that substring does not contain all (four) adjacent 1s.