CSE 311: Foundations of Computing I

Homework 7 (due December 3rd 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. A Lot to Feedback, There (0 points)

Approximately how much time (in minutes) did you spend on each problem of this homework? Were any problems especially difficult or especially interesting?

2. Glitz and Grammar [Online] (15 points)

For each of the following, a construct context-free grammar that generates 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) [5 Points] Binary strings matching the regular expression "$000(1 \cup 01 \cup 001)^*$".
Hint: You can use the technique described in lecture to convert this RE to a CFG.

(b) [5 Points] All strings over $\{0, 1, 2\}$ of the form $x\,2\,y$, 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) [5 Points] All binary strings in the set $\{0^m 1^n 0^{n+2m} : m, n \geq 0\}$.

Submit and check your grammars here:

https://grin.cs.washington.edu/

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

Note: You must also include each grammar and sentences describing each new non-terminal, as described above, with the rest of your assignment.

3. Good, Good, Good, Good Relations (12 points)

Consider the set of all pages on Wikipedia. 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 there is a word that appears on both pages a and b.

(b) [4 Points] $(a, b) \in R$ iff every word on page a also appears on page b

(c) [4 Points] $(a, b) \in R$ iff the number of words on page a is twice that on page b

4. Get a Prove On (10 points)

(a) [5 Points] Prove or disprove: If R and S are antisymmetric relations on A, then $R \cup S$ is antisymmetric.

(b) [5 Points] Prove or disprove: If R and S are transitive relations on A, then $R \cup S$ is transitive.

5. Cat Goes "Meow". Dog Goes "Proof" (15 points)

(a) [5 Points] Disprove: If R and S are symmetric relations on A, then $R \circ S$ is symmetric.

(b) [10 Points] Let A be a fixed set. Prove: If R and S are symmetric relations on A and $R \circ S = S \circ R$, then $R \circ S$ is symmetric.

6. Few and Far Machine [Online] (30 points)

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

(a) [5 Points] Binary strings that start with 1 and have odd length.

(b) [5 Points] Binary strings where every occurrence of a 0 is immediately followed by a 11.

(c) [5 Points] Binary strings with an odd number of 1s.

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

(e) [5 Points] Binary strings with at least two 1s **or** at most one 0.

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

Submit and check your answers to this question here:

https://grin.cs.washington.edu/

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

7. Up the Ladder To the Proof (28 points)

Recall the following definition of linked lists of numbers from lecture:

Bases Step: $\texttt{null} \in$ **List**

Recursive Step: For any $x \in \mathbb{R}$, if $L \in$ **List**, then $\texttt{Node}(x, L) \in$ **List**.

The following function (concat) concatenates two lists:

$$\begin{array}{rcll} \text{concat}(\texttt{null}, R) & = & R & \forall R \in \textbf{List} \\ \text{concat}(\texttt{Node}(x, L), R) & = & \texttt{Node}(x, \text{concat}(L, R)) & \forall x \in \mathbb{R}, \forall L, R \in \textbf{List} \end{array}$$

and this next function (rev) reverses a list:

$$\begin{array}{rcll} \text{rev}(\texttt{null}) & = & \texttt{null} & \\ \text{rev}(\texttt{Node}(x, L)) & = & \text{concat}(\text{rev}(L), \texttt{Node}(x, \texttt{null})) & \forall x \in \mathbb{R}, \forall L \in \textbf{List} \end{array}$$

While simple, rev is inefficient. It takes $\Theta(n^2)$ steps to reverse a list of length n. In this problem, we will find a more efficient implementation. Toward that end, we define the following function (rev2):

$$\begin{array}{rcll} \text{rev2}(\texttt{null}, R) & = & R & \\ \text{rev2}(\texttt{Node}(x, L), R) & = & \text{rev2}(L, \texttt{Node}(x, R)) & \forall x \in \mathbb{R}, \forall L \in \textbf{List} \end{array}$$

(a) [2 Points] Calculate $\text{rev2}(\texttt{Node}(3, \texttt{Node}(4, \texttt{null})), \texttt{Node}(2, \texttt{Node}(1, \texttt{null})))$ via the definitions above.

(b) [20 Points] Prove that $\text{rev2}(A, B) = \text{concat}(\text{rev}(A), B)$ for all lists A and B by structural induction on A.

You may use without proof the fact that concat is associative, namely, the fact that we have $\text{concat}(\text{concat}(A, B), C) = \text{concat}(A, \text{concat}(B, C))$ for all $A, B, C \in$ **List**. (We proved the same fact for strings — i.e., $(x \bullet y) \bullet z = x \bullet (y \bullet z)$ — in the last assignment. The proof for lists is almost identical.)

Hint: When proving equality in the inductive step, it may help to work *backward* as well as forward.

(c) [6 Points] Prove that $\text{rev}(A) = \text{rev2}(A, \texttt{null})$ for all lists A. This shows that we can reverse a list by calling rev2, passing in `null` for the second argument, instead of calling rev.

Hints: Prove this by cases over what A looks like: it must be either `null` or $\texttt{Node}(x, L)$ for some $x \in \mathbb{R}$ and $L \in$ **L**. You should cite part (b) in your argument.

8. Extra Credit: Stop! Grammar Time (0 points)

Consider the following context-free grammar.

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

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.