Homework 7: Structural Induction, Regular Expressions, CFGs

Due date: Wednesday May 19 at 11:00 PM (Seattle time, i.e. GMT-7)
If you work with others (and you should!), remember to follow the collaboration policy.
In general, you are graded on both the clarity and accuracy of your work. Your solution should be clear enough that someone in the class who had not seen the problem before would understand it.
We sometimes describe approximately how long our explanations are. These are intended to help you understand approximately how much detail we are expecting. Whenever an exercise asks to "*prove*" or "*show*" a statement without further specification, you are free to choose any proof method that you have learned in 311.

Be sure to read the grading guidelines for more information on what we're looking for.

While working on the assignment, please (roughly) keep track of the time you spent on each problem. This is so that you have an idea of how to answer the feedback questions!

1. Collaborators

List any collaborators (i.e., other students) you worked with and which problems you worked on together, or state that you worked alone.

2. Manhattan Walk [20 points]

Let S be a subset of $\mathbb{Z} \times \mathbb{Z}$ defined recursively as:

Basis Step: $(0, 0) \in S$
Recursive Step: if $(a, b) \in S$ then $(a, b + 1) \in S$ and $(a + 2, b + 1) \in S$

Prove that $\forall (a, b) \in S, a \leq 2b$.

Hint Remember that with structural induction you must show $P(s)$ for every element s that is added by the recursive rule – you will need to show $P()$ holds for two different elements in your inductive step.

3. The Apple Doesn't Fall Far From The... `Tree` [20 points]

In CSE 143, you saw a recursive definition of trees. That definition looks a little different from what we saw in class.

The following definition is analogous to what you saw in 143. We'll call them `JavaTrees`.

Basis Step: `null` is a `JavaTree`.
Recursive Step: If L, R are `JavaTrees` then $(\texttt{data}, L, R)$ is also a `JavaTree`.

Show that for all `JavaTrees`: if they have k copies of `data` then they have $k + 1$ copies of `null`.

(In this problem `data` is not a variable, just a constant: a real tree data structure would put some data there instead.)

Remark: You're effectively showing here that a binary tree with k nodes has $k + 1$ `null` child pointers.

4. What doesn't kill you makes you stronger [22 points]

Consider the following intertwined definitions of recursively defined sets, S, T.

Definition of S
Basis: $0 \in S$
Recursive: If $x \in T$ then $x + 3 \in S$

Definition of T
Basis: $1 \in T$, $3 \in T$
Recursive: If $x \in S$ then $x + 1 \in T$.

Let $P(n)$ be "if n is odd, then $n \in T$" and let $Q(n)$ be "if n is even, then $n \in S$"

(a) Explain why $P(2)$ is true (don't write a formal proof; our answer is one sentence). [2 points]

(b) Prove that $P(n)$ is true for all integers $n \geq 4$. We recommend to define $R(n) := P(n) \wedge Q(n)$ and then prove that $R(n)$ is true for all $n \geq 4$ by strong induction. [20 points]
Hint. Be careful with this proof: you should not use structural induction (you are showing a claim about all integers, not a claim about every member of S or T). Also think carefully about what your inductive step should look like – P and Q are **implications**, that means your inductive hypothesis will be an implication and your inductive step will begin by supposing a hypothesis.
Comment. This proof technique is sometimes called "strengthened induction" (not to be confused with strong vs. weak induction). Instead of proving a claim $\forall n\ P(n)$ by induction it can be more convenient to rather prove a stronger claim $\forall n\ (P(n) \wedge Q(n))$ by induction. In the inductive step this means a stronger conclusion $R(k+1)$ has to be proven — but also the assumption in form of the inductive hypothesis $R(k)$ is stronger.

5. Recursion – See: Recursion [18 points]

For each of the following languages, give a recursive description of the language. Your basis step must explicitly enumerate a finite number of initial elements. Try to use the shortest description possible (in terms of the number of recursive rules and the number of basis rules). Briefly (1-2 sentences) justify that your description defines the same language but do not give us a full proof;

(a) Binary strings that start with 0 and have odd length (i.e. an odd number of characters).

(b) Binary strings x such that $\texttt{len}(x) \equiv 1 \pmod 3$ where $\texttt{len}(x)$ is the number of characters in x.

(c) Binary strings with an odd number of 0s.

6. Constructing Regular Expressions (Online) [20 points]

For each of the following languages, construct a regular expression that matches exactly the given set of strings. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Test your regular expression carefully before submitting; you only have 5 submissions. Because these are auto-graded, we will not award partial credit.

You **must** also take a screenshot of your final submission and include that in your gradescope submission (in case of technical errors with Grin).

(a) Binary strings where every occurence of a 1 is immediately followed by a 0.

(b) Binary strings where no occurence of 00 is immediately followed by a 1.

(c) The set of all binary strings that contain at least one 1 and at most two 0's.

(d) The set of all binary strings that begin with a 1 and have length congruent to $2 \pmod 4$.

7. Context Is Everything. Except for Context-Free Grammars (Online) [10 points]

For each of the following languages, construct a context-free that generates exactly the given set of strings. The start symbol is S. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Test your regular expression carefully before submitting; you only have 5 submissions. Because these are auto-graded, we will not award partial credit.

You **must** also take a screenshot of your final submission and include that in your gradescope submission (in case of technical errors with Grin).

(a) The set of all binary strings that contain at least one 1 and and most two 0's.

(b) Binary strings with an odd number of 0's

8. Extra Credit: Ambiguity

Consider the following context-free grammar.

⟨**Stmt**⟩ → ⟨**Assign**⟩ | ⟨**IfThen**⟩ | ⟨**IfThenElse**⟩ | ⟨**BeginEnd**⟩
⟨**IfThen**⟩ → if condition then ⟨**Stmt**⟩
⟨**IfThenElse**⟩ → if condition then ⟨**Stmt**⟩ else ⟨**Stmt**⟩
⟨**BeginEnd**⟩ → begin ⟨**StmtList**⟩ end
⟨**StmtList**⟩ → ⟨**StmtList**⟩⟨**Stmt**⟩ | ⟨**Stmt**⟩
⟨**Assign**⟩ → `a := 1`

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) 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) Give **two different grammars** for this language that are both unambiguous but produce different parse trees from each other.

9. Feedback

Please keep track of how much time you spend on this homework and answer the following questions. This can help us calibrate future assignments and future iterations of the course, and can help you identify which areas are most challenging for you.

- How many hours did you spend working on this assignment?
- Which problem did you spend the most time on?
- Which problem did you find to be the most confusing?
- Any other feedback for us?