CSE 311: Foundations of Computing I

Homework 7 (due Wednesday, November 19)

Directions: *Write up carefully argued solutions to the following problems. The first task is to be complete and correct. The more subtle task is to keep it simple and succinct. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. You may use any results proven in lecture without proof. Anything else must be argued rigorously. Unless otherwise specified, all answers are expected to be given in closed form.*

0. Constructing Four Grammars (30 points)

You will use `https://grinch.cs.washington.edu/cfg/` to submit your answers for this question, like the last homework.

Unfortunately, though, while the CFG site will **try** to tell you if the result is wrong, it cannot **guarantee** it (and the English description will be graded manually as well). That said, we believe it will generally automatically find counterexamples to grammars you might accidentally construct.

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 were:

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

We would expect you to say "E generates (non-empty) even length binary strings; O generates odd length binary strings; C generates binary strings of length one."

(a) [5 Points] The set of all binary strings that are of odd length and have 1 as their middle character.

(b) [5 Points] $\{1^m 0^n 1^{m+n} : m, n \geq 0\}$

(c) [10 Points] All binary strings that contain at least two 0's and at most two 1's.

(d) [10 Points] $\{1^m 0^n 1^p : m, n, p \geq 0$ and $(m \geq n$ or $n \leq p)\}$

1. You're My Transitive-Reflexive Closure (10 points)

Disprove the statement "For all relations R, the transitive-reflexive closure of R equals the transitive-reflexive closure of R^3."

2. Set Up To Relate (20 points)

Let A be a set. Let R and S be transitive relations on A.

(a) [10 Points] Is $R \cup S$ necessarily transitive? Prove your answer.

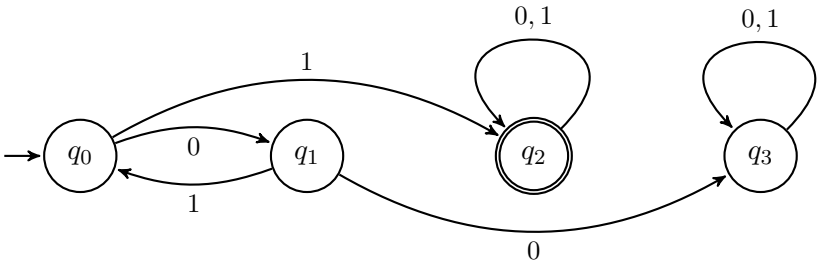
(b) [10 Points] Is $R \cap S$ necessarily transitive? Prove your answer.

3. Family Times (20 points)

We use $\mathbb{Z}^+$ to mean the set of positive integers.
Let $R \subseteq (\mathbb{Z}^+ \times \mathbb{Z}^+) \times (\mathbb{Z}^+ \times \mathbb{Z}^+)$ be the relation given by $((a, b), (c, d)) \in R$ if and only if $ad = bc$.
Prove that R is reflexive, symmetric and transitive.

4. Plain, Old, Regular States (20 points)

Consider the following DFA, M:



For each of the four states, q, in M, write a regular expression that matches exactly the strings that end at the state q when starting from the initial state.

For this question, please go back to `https://grinch.cs.washington.edu/regex/` to submit your answers. Do not submit them on paper.

5. EXTRA CREDIT: Lock and Key (-NoValue- points)

In Adam's office, there is a peculiar safe with some documents that he would really like to recover. The combination is a subset of $[A - Z]^* \setminus \{\varepsilon\}$.
Each combination can either *open*, *jam*, or *still* the lock. If a combination *jam*s the lock, then it will no longer ever work. If a combination *still*s the lock, then it is as if no combination were entered at all.
We use greek letters like α and β to denote arbitrary combinations. We define the operation $\alpha \hookrightarrow \beta$ as follows:

(a) For any combination α, QαQ $\hookrightarrow \alpha$

(b) If $\alpha \hookrightarrow \beta$, then L$\alpha \hookrightarrow$ Qβ

(c) If $\alpha \hookrightarrow \beta$, then V$\alpha \hookrightarrow \beta^{\mathcal{R}}$

(d) If $\alpha \hookrightarrow \beta$, then R$\alpha \hookrightarrow \beta\beta$

(e) If $\alpha \hookrightarrow \beta$, then ($\alpha$ *jam*s the lock $\rightarrow \beta$ *still*s it) and (α *still*s the lock $\rightarrow \beta$ *jam*s it)

Find the shortest possible combination that is guaranteed to open the lock, and prove your answer correct.

6. EXTRA CREDIT: Ambiguous Grammar (-NoValue- points)

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 different meanings).

(a) [-NoValue- Points] Show an example of a string in the language that has two different parse trees.

(b) [-NoValue- Points] Give a new grammar for the same language that is **unambiguous** in the sense that every string has a unique parse tree.