CSE396, Spring 2019 Problem Set 5 Due Thu. 3/28, 11:59pm

Reading. There are "two-and-a-half" more topics in which class coverage will veer from the text and have separate handouts. The first, to come on Tuesday of week 8 into the Thursday, is a version of "Structural Induction" (SI) that applies to context-free grammarg G. It abbreviates a technique for proving soundness statements of the form $L(G) \subseteq B$ that is called "induction on the first step of derivations" in other sources. The text seems to presume that students can do this kind of proof in a few of its problems, but please read the special handout https://cse.buffalo.edu/ regan/cse396/CSE396SI.pdf on the course webpage.

The "half" is that lectures will present the algorithm to convert a CFG into Chomsky normal form differently—in a way that emphasizes the step of identifying "nullable" variables *prior to* the step of bypassing them, whereas the text combines both steps into one. Doing so will bring a benefit later in Chapter 4. Hence for Thursday please also read the handout https://cse.buffalo.edu/ regan/cse396/CSE396ChNF.pdf which is also on the course webpage.

The other "whole" deviation is actually a "−1": Please do *not* read the text's formulation of a pushdown automaton (PDA) in section 2.2. Or skim it, but treat the text's compound use of ϵ's in various places to mean different things the way Perseus treated Medusa's snakes. Instead, when lectures reach chapter 3 they will treat PDAs as a special case of two-tape Turing machines, thus not requiring you to learn a wholly separate notation for PDAs. Furthermore, Chomsky NF leads naturally into section 2.3 anyway, because it enables a simpler visualization of the proof of the CFL Pumping Lemma in which the tree is binary. Section 2.4 will be mentioned only for a few brief passages toward the beginning that have to do with deterministic PDAs (DPDAs), which again will be meshed into Chapter 3. Illustrations of PDAs whch are also on the course webpage alongside Turing machines will be shown later in April.

It may seem strange that we will still be in Chapter 2 through Week 9, with chapters 3,4,5,7 (and one proof in chapter 9 replacing that of the Cook-Levin Theorem in chapter 7) still to come, but compare the numbers of pages in these chapters with what we've read. The later material is higher-level (especially *mapping reductions* in section 5.3 which will superimpose on the coverage of 5.1 while 5.2 is skipped) but it is less varied, so the lectures too will have a straighter path through them.

The date of Prelim II will highly likely be **Thursday, April 25** but this has not yet been fixed.

Homework—part online and all *individual work*—due **Thu.** 3/28, 11:59pm:

(1) Using *TopHat*, the "Worksheet" titled **Spr'19 HW5.1**. There are 10 questions, each worth 2 points, for 20 total. **Important: The questions now give only one attempt.**

(2) For the following languages L_1, L_2 over $\Sigma = \{a, b\}$, design context-free grammars G_1, G_2 such that $L(G_1) = L_1$ and $L(G_2) = L_2$. You need not prove your grammars correct, but as

usual you should include a few comments explaining how and why the grammars work correctly. Also give a CFG G_3 such that $L(G_3) = L_1 \cup L_2$. (12 + 12 + 6 = 30 pts.)

1. $L_1 = \{a^m b^{m+n} a^n : m \geq 1, n \geq 0\}$,
2. $L_2 = \{xby : \#a(x) = \#b(y)\}$.

(3) Consider the following CFG G with terminal alphabet $\Sigma' = \{a, b, e, +, \cdot, *, (,)\}$. Here we've written Σ' because G is supposed to represent syntactically the legal regular expressions over the target alphabet $\Sigma = \{a, b\}$, except that we are not providing a symbol for the empty set (which we've seen is useful for *calculating* regular expressions but do you ever need it for *writing* them in the first place?) and 'e' stands for the empty string in the syntax without literally being ϵ in rules.

$$S \to a \mid b \mid e \mid S + S \mid S \cdot S \mid S^* \mid (S).$$

The periods here and below are just punctuation.

(a) Give both a parse tree and a leftmost derivation for the string $r = ((a \cdot b)^* + a \cdot a)$.

(b) Show that r is ambiguous in G by giving a different parse tree and corresponding leftmost derivation for it.

(c) Briefly explain why one parse should be preferred over the other.

(d) Suppose we added the rule $S \to SS$ and wrote $r' = ((a \cdot b)^* + aa)$ instead. Would doing this fix the ambiguity problem?

(e) Give a CFG G' such that $L(G') = L(G)$ and G' is unambiguous. For proof it suffices to compare your G' with an example from lectures and/or the text, briefly. Then give a parse tree for r (not r') in your G' and explain why the *other* parse tree can no longer be imitated to produce r literally. (6 + 6 + 3 + 3 + 12 = 30 pts., for 80 on the set)