CSE396 **Problem Set 5** **Due 3/29, 11:59pm**

Reading:

Tuesday's lecture will finish section 2.1 by covering Chomsky normal form (ChNF). Skip over section 2.2 for now—we will visit it after introducing Turing machines and defining pushdown automata (PDAs) as a special case of two-tape Turing machines. (This saves you having to learn a completely separate machine formalism, whose multiple uses of "ϵ" are miasmic, and the yucky proof that CFGs are equivalent to nondeterministic PDAs is one I skip anyway.) Instead go on to section 2.3 with the awareness that when the grammar is in Chomsky normal form we can do the whole proof with the simplification $b = 2$. Otherwise I am not a fan of Chomsky normal form; to my mind it makes good grammars unreadable. The part of the process that I regard as important is the algorithm for finding all the *nullable* variables—that is, any and all variables that can derive ϵ—and the related (optional for ChNF) algorithms for finding which variables can derive terminal strings at all and which variables can be reached in derivations from S at all. These algorithms are all forms of breadth-first search and will come up more importantly in section 4.1 as ways to decide the problems "is $\epsilon \in L(G)$?" and "is $L(G) = \emptyset$?" given any context-free grammar G. (The problem "is $L(G) = \Sigma^*$?" is however *undecidable*, which already cues you in that there is no general way to "complement a grammar." Section 2.4 will be largely skipped except for where its first few pages talk about deterministic PDAs and the fact that *those* **are** closed under complementation, which already cues you in that not all nondetermninistic PDAs can be converted into equivalent deterministic ones.)

Meanwhile, I have replaced a handwritten handout on the course webpage about "Structural Induction" with a longer typeset one:

`https://www.cse.buffalo.edu/~regan/cse396/CSE396SI.pdf`

Please read this before Thursday's lecture. I have not decided whether to slot this in before or after the "CFL Pumping Lemma" from section 2.3; if the latter, it will come the following Tuesday. Note that there is also a supplementary handout on Chomsky normal form on the course webpage. Read it for the "*NULLABLE*" step but skim the rest. The following porblem set is due Thursday 3/29 at 11:59pm as usual but is a little shorter.

(1) This problem is "HW5 Online Part" on *TopHat*, worth 20 pts. as before.

(2) Consider the following grammar G—it is also on the last page of the new notes on structural induction:

$$\begin{aligned} S &\rightarrow SS \mid 0B \mid 1A \mid \epsilon \\ A &\rightarrow 0S \mid 1AA \\ B &\rightarrow 1S \mid 0BB \end{aligned}$$

(a) Give both a parse tree and a leftmost derivation for the string $x = 10110100$ (3 + 3 = 6 pts.)

(b) Show that x is ambiguous in G. Either a second parse tree or a second leftmost derivation will do. (3 pts.)

(c) Indeed, G is "molto ambiguoso": explain the trivial reason why every string in $L(G)$, even the empty string, has infinitely many different parse trees. (6 pts.—part (d) is overleaf)

(d) Can we *remove* one rule to leave a grammar G' that is equivalent—i.e. $L(G') = L(G)$—and is "less ambiguous"? In general this is a hard question, and the text does not even give a formal technique for proving unambiguity (neither will my notes or lectures), let alone quantify what "less ambiguous" means. But this is about the easiest case of this kind of question, because the resulting G' will be in *Greibach normal form* (GNF), named for Sheila Greibach of UCLA. GNF is not in the text either and just means that the right-hand side of every rule is either ϵ or begins with a terminal symbol. GNF grammars can still be ambiguous, but they help the task of *parsing* so much that cases of unambiguity—or at least having a straightforward parsing algorithm—are easier to see. So this is reasonable to assign for 9 pts., making 24 on the problem. State your G' and argue in words as best you can that G' is still comprehensive and how you would parse strings.

(3) Design a CFG G'' such that $L(G'') = \{x \in \{0,1\}^* : \#0(x) > \#1(x)\}$. *Hint:* Start with the above grammar for the "equal" case and add stuff to it. You need not prove your G'' correct but comments about your strategy should express why it is correct. Note that you can add variables and/or change the start symbol and need not care about ambiguity—some (other) context-free languages only have ambiguous CFGs anyway. (12 pts., for 46 total on the set. Please do not try to find this particular problem on the Internet—even if you do, I've written this to make your answer conform to the thread of ideas in problem (2) anyway.)