MA/CSSE 474 Day 20 Summary

1) **Review:** CFG G = (V, G, R, S), where
   a) Σ is the ***terminal alphabet***; **N** *is the* ***nonterminal alphabet***; **V** = $\Sigma \cup N$ is the **rule alphabet**; **R** is the set of **productions** of the form $A \rightarrow \beta$, where $A \in N$ and $\beta \in V^*$; and **S** $\in$ N is the **start symbol.**
   b) **One derivation step:** $x \Rightarrow_G y$ iff $\exists \alpha,\beta,\gamma \in V^*, A \in N\ ((x = \alpha A \beta) \wedge (A \rightarrow \gamma \in R) \wedge (y = \alpha \gamma \beta))$
   c) $\Rightarrow_G^*$ is the reflexive, transitive closure of $\Rightarrow_G$
   d) The **language defined by a grammar**: **L(G)** = $\{w \in \Sigma^* : S \Rightarrow_G^* w\}$
   e) A language L is ***context-free*** iff there is some context-free grammar G such that $L = L(G)$. **CFL.**

2) Prove that a grammar is correct: L: $A^nB^n = \{a^n b^n : n \geq 0\}$ G: $S \rightarrow a S b,\ S \rightarrow \varepsilon$
   a) Show that if $w \in L$, then $S \Rightarrow^* w$.
   Induction on what? What to prove by induction? Use this to prove what we want.

3) Show that if $S \Rightarrow^* w$, then $w \in L$.
   Induction on what? What to prove by induction? Use this to prove what we want.

4) Often we intend for the syntax of a CFL (such as a programming language) to imply structure. This is true of programming languages, of course.
5) A ***parse tree***, derived from a grammar $G = (V, \Sigma, R, S)$, is a rooted, ordered tree in which:
   a) Every leaf node is labeled with an element of $\Sigma \cup \{\varepsilon\}$,
   b) The root node is labeled *S*,
   c) Every other node is labeled with an element of *N*, and
   d) If *m* is a non-leaf node labeled *X* and the (ordered) children of *m* are labeled $x_1, x_2, \ldots, x_n$,
      i) then *R* contains the rule $X \rightarrow x_1\, x_2, \ldots\, x_n$.
6) CFG's can generate strings by substituting for nonterminals in any order.
   a) Practical algorithms use a specific order
   b) Leftmost and rightmost are the most common orders
7) A grammar is ***ambiguous*** if some string it generates has two different parse trees
   a) Equivalently, two different leftmost derivations, or two different rightmost derivations
8) A CFL is ***inherently ambiguous*** if *every* CFG that generates it is ambiguous.
   a) $L = \{a^n b^n c^m: n, m \geq 0\} \cup \{a^n b^m c^m: n, m \geq 0\}$
9) **Ambiguity and undecidability.** Both of the following problems are undecidable:
   a) Given a context-free grammar *G*, is *G* ambiguous?
   b) Given a context-free language *L*, is *L* inherently ambiguous?
10) Nonterminal A is *nullable* iff $A \Rightarrow^* \varepsilon$. Algorithm for finding nullable nonterminals is similar to others we've seen.
11) Given G, we can easily find a grammar with no ε-productions that generates L(G) – { ε }
12) We can eliminate symmetric recursive rules
13) A ***normal form*** *F* for a set *C* of data objects is a form, i.e., a set of syntactically valid objects, with the following two properties:
    a) For every element *c* of *C*, except possibly a finite set of special cases, there exists some element *f* of *F* such that *f* is equivalent to *c* with respect to some set of tasks.
    b) *F* is simpler than the original form in which the of *C* are written.
       i) By "simpler" we mean that at least some tasks are easier to perform on elements of *F* than they would be on elements of *C*.
14) **Chomsky Normal Form**, in which all rules are of one of the following two forms:
    a) $X \rightarrow a$, where $a \in \Sigma$, or
    b) $X \rightarrow BC$, where *B* and *C* are elements of $V - \Sigma$.
15) Converting a grammar to CNF is straightforward; read about it in the book and figure it out.
16) **Greibach Normal Form**, in which all rules are of the form $X \rightarrow a\,\beta$, where $a \in \Sigma$ and $\beta \in N^*$.
    a) You do not need to look at the algorithm for converting to GNF.