MA/CSSE 474 Day 27 Summary

If L is a context-free language, then
$\exists k \geq 1$ ($\forall$ strings $w \in L$, where $|w| \geq k$
($\exists u, v, x, y, z$ ($w = uvxyz$, $vy \neq \varepsilon$, $|vxy| \leq k$,
and
$\forall q \geq 0$ (uv^qxy^qz is in L)))).

Main ideas from today:

1) $\{xcy : x, y \in \{0, 1\}^* \text{ and } x \neq y\}$

2) **Variations on PDA**: Acceptance by accepting state only, replace stack with queue, two stacks.

3) CFL closure:
   a) Union. New start symbol: add productions $S \rightarrow S_1$, $S \rightarrow S_2$
   b) Concatenation. New start symbol: add production $S \rightarrow S_1S_2$
   c) Kleene Star. New start symbol: add productions $S \rightarrow \varepsilon$, $S \rightarrow S\,S_1$
   d) Reverse. Transform grammar to Chomsky Normal form. Replace each production $A \rightarrow BC$ by $A \rightarrow CB$
   e) Not closed under complement: Consider $A^nB^nC^n$. (done a few days ago)
   f) Not closed under intersection: $L_1 = \{a^nb^nc^m : n, m \geq 0\}$ $L_2 = \{a^mb^nc^n : n, m \geq 0\}$
   g) Intersection of a CFL and a regular language is CF (same for difference of a regular lang. and a CF lang.)
   h) Don't try to use closure backwards! Sam principle as for regular languages.
4) A PDA may never halt or never finish reading its input.
5) Nondeterminism can lead to exponential running time.
6) Deterministic PDA M:
   a) Δ_M contains no pairs of transitions that compete with each other, and
   b) whenever M is in an accepting configuration it has no available moves.
7) A language L is ***deterministic context-free*** (DCFL) iff $L\$$ can be accepted by some deterministic PDA.
   a) $L = a^* \cup \{a^nb^n : n > 0\}$ demonstrates the need for the \$ "end-of-input" symbol (details on slides).
   b) DCFLs are closed under complement, but not under union or intersection (we will not show these)
8) Every CFL over a single-letter alphabet must be regular.
9) Algorithms and decision problems for CFLs
   a) Membership: Given a CFL L and a string w, is w in L?
      i) How not to do it (examples are on the slides)
         (1) there is a CFG G that generates L. Try derivations in G and see whether any of them generates w.
         (2) there is a PDA M that accepts L. Run M on w.
      ii) But, if grammar is in CNF …. (ε is handled as a special case).
         (1) Works but not very efficient
         (2) There is an $O(N^3)$ dynamic programming algorithm (CKY, a.k.a. CYK)
      iii) Or, can build a PDA with no ε-transitions from a GNF grammar.
   b) Emptiness. Remove unproductive nonterminals form grammar. L empty iff S is not removed.
   c) Finiteness. Let b be the branching factor of CFG. If language is infinite, some string of length between b^n and $b^n + b^{n+1}$ will be accepted. Enumerate and try them all.
   d) Undecidable questions about CFLs:
      i) Is $L = \Sigma^*$?
      ii) Is the complement of L context-free?
      iii) Is L regular?
      iv) Is $L_1 = L_2$?
      v) Is $L_1 \subseteq L_2$?
      vi) Is $L_1 \cap L_2 = \varnothing$?
      vii) Is L inherently ambiguous?
      viii) Is G ambiguous?

10) **Turing machine (TM)** intro (if there is time, which will be amazing if it happens!)
   a) Tape alphabet, blank symbol, two-way-infinite tape, read/write head.
   b) Based on current state and tape symbol, the TM
      i) Changes to next state
      ii) Writes a symbol on current tape square
      iii) Moves left or right (
         (1) In some other authors' equivalent TM models, staying on same square is option. Not here.
   c) **Formal TM definition.** A deterministic TM M is (K, Σ, Γ, δ, s, H):
      i) K is a finite set of states;
      ii) Σ is the input alphabet, which does not contain ☐;
      iii) Γ is the tape alphabet, which must contain ☐ and have Σ as a subset.
      iv) $s \in K$ is the initial state;
      v) $H \subseteq K$ is the set of halting states;
      vi) δ is the transition **function**: (for a nondeterministic TM, we will need a more general relation Δ)
         (1) $(K - H) \times \Gamma$ to $K \times \Gamma \times \{\rightarrow, \leftarrow\}$
         non-halting state × tape char → state × tape char × direction to move (R or L)

   d) A TM is not guaranteed to halt. And there is no algorithm to take a TM M and find an equivalent TM that is guaranteed to halt.
11) Example: M takes as input a string in the language: $\{a^i b^j, 0 \le j \le i\}$, and adds b's as required to make the number of b's equal the number of a's.
12) **Trace its action on** aab: