pp. 101-108. **Context Free Grammars** (Sec. 2.1)

- Remember: languages are sets of strings
- Also not all languages are regular: B = $\{0^n1^n | n\geq0\}$
- **Context Free Languages (CFL)**: a superset of Regular Languages – but still NOT ALL POSSIBLE languages
  - E.g. B = $\{0^n1^n | n\geq0\}$ is context free
  - But $\{a^nb^nc^nd^n | n > 0\}$ is not context free
- (p. 102) Defined by **Context Free Grammars (CFG)**
  - ∑ as before + set of **substitution rules** + **start variable**
    - **Terminals** are symbols from alphabet
    - **Nonterminals**: name of set of strings
      - Sometimes in "<>"
      - **Start variable** = non-terminal for entire language
    - Substitution rules: how to replace a nonterminal with some string
      - Rule format: **LHS -> RHS**
        - **LHS**: nonterminal
        - **RHS**: a string or expression over strings:
          - Concatenation of strings
          - Using both nonterminals & **terminals**
      - "|" = shorthand for "or"

- (p. 102) Example of CFG with $\sum = \{0,1\}$, Start = A

  A -> 0A1 | B

  B -> #
- (p. 103) **Parse Tree**: Generate a tree of strings
  - starting with root as some variable
  - Successively replace some variable on leaves by RHS of rule with that variable as LHS
  - See p. 103 for parse tree and another grammar
- **Formal Definition**: G = (V, ∑, R, S)
  - V is set of names for variables (the "non-terminals")
  - ∑ is alphabet (must be disjoint from V)
  - R is a set of rules: <var> -> string
  - S a start variable from V
- **Derivation** of one string v from another:
  - Assume u, v strings from (∑ U V)*
    - u **yields** v, written u => v, if
      - either u = v
      - or u = xyz where
        - y is a variable
        - There is a rule of form y -> w (w a string)
        - Where xwz = v
      - Or a series of such substitutions $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow .. u_k \Rightarrow v$

- **L(G) = Language** of grammar G = {w|w in ∑*, & S => w}
- (pp. 104-105) have more examples
- (p. 106) Constructing CFG from complex CFLs
  - 1. Many CFLs are unions of simpler ones
    - Construct CFGs for each piece, with start states S_i
      - and then S-> S_1 |… S_k (akin to an ε edge)
    - e.g. p. 106
  - 2. (p. 107) Constructing a CFG for a regular language
    - Start with a DFA accepting the regular language
    - For each state q_i in Q, define a variable R_i
    - **If q_0 is start state then make R_0 the start variable**
    - **If $\delta(q_i, a) = q_j$ add rule R_i -> aR_j**
    - **For each accept state q_i, add rule R_i -> ε**
    - Example: DFA on Fig. 1.44 (p. 58)
      - V = {R_0, R_2, R_3, R_{13}, R_{23}, R_{123}}, S = R_{13}
      - Rules: R_{13} -> aR_{13}; R_{13} -> bR_2; R_2 -> aR_{23}; R_2 -> bR_3; R_{23} -> aR_{123};
      - R_{23} -> bR_3; R_3 -> aR_{13}; R_3 -> bR_0; R_{123} -> aR_{123}; R_{123} -> bR_{23};
      - R_0 -> aR_0; R_0 -> bR_0; R_{13} -> ε
      - E.g. R_{13} => bR_2 =>bbR_3 => $bbaR_{13}$ => bba
  - 3. (p. 107) Language has concatenation of 2 or more strings that seem coupled (e.g. {0^n1^n| n≥0})
    - Use rules like R -> uRv to build left & right in sync

- 4. (p. 107) many strings contain (recursive) substrings that are used in other structures (e.g. p.105 – arithmetic exprs)
    - Use separate variable for each such structure
- (p. 108) **Ambiguity**
  - Some grammars can generate same string in >1 parse trees
  - If this is possible, grammar is called **ambiguous**
  - Some CFLs inherently ambiguous (see Problem 2.29)
  - E.g. (p. 108) 2 different parse trees for a+a*a from:

    <expr> -> <expr>+<expr> | <expr>*<expr> | a
      - <expr> => <expr>*<expr> => <expr>*a => <expr>+<expr>*a => <expr>+a*a => a+a*a
      - <expr> => <expr>+<expr> => a+<expr> => a+<expr>*<expr> => a+a*<expr> = a+a*a
  - Multiple derivations possible from same parse tree
    - eg. <expr> => <expr>+<expr> => a+<expr> => a+<expr>*<expr> => a+<expr>*a = a+a*a
  - Define **leftmost derivation** if we *always* replace **leftmost** nonterminal *first* at each step
  - String w is **derived ambiguously** in G if it has ≥2 leftmost derivations