Designing Context-Free Grammars.

- Some basic techniques:
  - Matching – enforcing the fact that your grammar generates matching pairs of symbols.
    Example: $L = \{0^n 1^{2n} \mid n \geq 0\}$. The context-free grammar for L is
    $$S \to 0S11 \mid \varepsilon$$
    The grammar forces every 0 to match to 11.
    What about $L = \{0^n 1^m \mid 2n \leq m \leq 3n\}$?
    Another example: $L = \{w \mid w \in \{0,1\}^*$ and of even length $\}$.
    $$S \to 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid \varepsilon$$
    The grammar forces every symbol to match to another symbol.
  - Using recursive relationships – when you can represent strings in the language in terms of shorter strings in the language.
    Example: L – the language of all strings of balanced brackets. For example, $(()())() \in L$, but $(() \notin L$.
    Every string w in L can be viewed as either
    - uv, for $u \in L$, $v \in L$, or
    - (u), for $u \in L$.

    This gives the following grammar:
    $$S \to SS \mid (S) \mid \varepsilon$$
  - Thinking of each nonterminal A as representing a language of all strings w derivable from A. Then, combining these languages into one, using the rules.
    Example: $L = \{0^n 1^m \mid n \neq m\}$.
    Let $L_1 = \{0^n 1^m \mid n > m\}$, and $L_2 = \{0^n 1^m \mid n < m\}$. Then, if S_1 generates L_1, and S_2 generates L_2, our grammar will be
    $$S \to S_1 \mid S_2$$
    L_1 is just the language of strings $0^n 1^n$ with one or more extra 0's in front. So,
    $$S_1 \to 0S_1 \mid 0E$$
    $$E \to 0E1 \mid \varepsilon$$
    L_2 is just the language of strings $0^n 1^n$ with one or more extra 1's in the end. So,
    $$S_2 \to S_2 1 \mid E1$$
    $$E \to 0E1 \mid \varepsilon$$

- **Example:** $L = \{w \mid w \in \{a,b\}^*, \#_a(w) = \#_b(w)\}$, where $\#_i(w)$ is the number of occurrences of a symbol i in a string w. In other words, L is the language of all strings that has an equal number of a's and b's.

  Solution: Take $w \in L$. Then either

  - $w = ax$, where $\#_a(x) = \#_b(x) - 1$, or
  - $w = by$, where $\#_a(y) = \#_b(y) + 1$.

  Let A to be a nonterminal that generates $L_A = \{x \mid \#_a(x) = \#_b(x) - 1\}$, and B be a nonterminal that generates $L_B = \{y \mid \#_a(y) = \#_b(x) + 1\}$. Then, the grammar for L is

  $$S \to aA \mid bB \mid \varepsilon$$

  Let's deal with A. Take $x \in L_A$, i.e. $\#_a(x) = \#_b(x) - 1$. Then, either

  - $x = az$, where $\#_a(z) = \#_b(z) - 2$, or
  - $x = bt$, where $\#_a(t) = \#_b(t)$.

  In the second case, $t \in L$, and so we will write $A \to bS$.

  What to do with the first case ? Observation: z must be a concatenation of two strings from L_A. Why ? Intuitively – z has 2 less a's than b's. At the beginning of z, the difference between the numbers of a's and b's is 0, at the end of z the difference is -2. Since this difference may only change by 1 on every additional symbol of z, it had to become -1 before it became -2.

  Formally: let $f(w) = \#_a(w) - \#_b(w)$, and let w_i stand for the prefix of w of length i. Then, for a string w of length n, $w \in L_A$ implies that $f(w_0) = 0$, while $f(w_n) = -2$. It follows that for some i, $f(w_i) = -1$. Thus, $w = w_i u$, where $u \in L_A$, and $v \in L_A$.

  Thus, for L_A, we have the following rule $A \to aAA \mid bS$. Similarly, for L_B we obtain $B \to bBB \mid aS$.

  Exercise: carefully derive the above rule for B.

  Putting everything together, we get

  $$\begin{aligned} S &\to aA \mid bB \mid \varepsilon \\ A &\to aAA \mid bS \\ B &\to bBB \mid aS \end{aligned}$$

  □

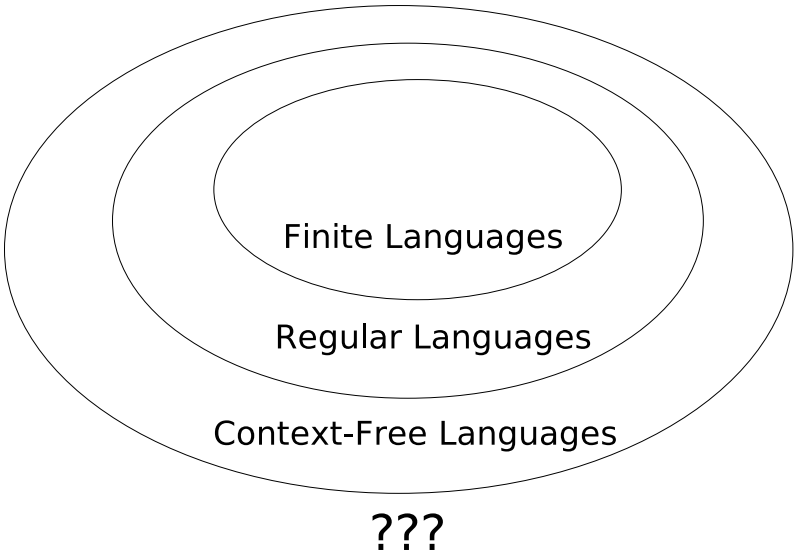
  Practice Problems: textbook 2.1, 2.4, 2.6, 2.9, 2.21, 2.22.

- How to show that the constructed grammar does indeed generate some required language ? If L is some language, and G is a CFG, then to prove that $L = L(G)$ you have to prove two things:
  - Every string w generated by G is in L. Typically, this is a proof by induction on the number of steps in the derivation of w.
  - Every string w in L can be generated by G. Typically, this is a proof by induction on the length, or some other parameter, of w.

Every Regular Language is Context-Free.

- We saw that some context-free languages are not regular (for example, $L = \{0^n 1^n \mid n \geq 0\}$). Now, we will show that every regular language is context-free. This implies that regular languages are a *proper* subset of context-free languages.

  Thus, so far, the picture of the world is as follows:



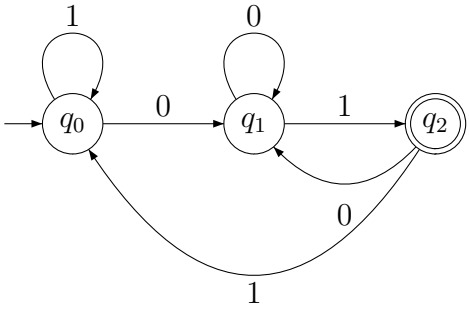
- **Theorem** For each regular language L there exists a context-free grammar G, such that $L = L(G)$.

  Proof: Idea – given a DFA, construct a context-free grammar that simulates its behavior:
  - Each state of the DFA will be represented by a nonterminal. The initial state will correspond to the start nonterminal.
  - For each transition $\delta(q_i, a) = q_j$, add a rule $q_i \rightarrow aq_j$.

– For each accepting state q_f, add a rule $q_f \to \varepsilon$.

– The

For example, for the DFA



The corresponding grammar would be

$$\begin{aligned} q_0 &\to 0q_1 \mid 1q_0 \\ q_1 &\to 0q_1 \mid 1q_2 \\ q_2 &\to 0q_1 \mid 1q_0 \mid \varepsilon \end{aligned}$$

Formally, let $M = (Q, \Sigma, \delta, q_0, F)$ be the DFA that recognizes L. Then, the corresponding context-free grammar $G = (V, \Sigma, R, S)$ is defined as follows:

– $V = Q$ (here we assume that $Q \cap \Sigma = \emptyset$. If not, rename the states).

– $S = q_0$.

– $R = \{q_i \to aq_j \mid \delta(q_i, a) = q_j\} \cup \{q_f \to \varepsilon \mid q_f \in F\}$.

Then, the theorem follows from the following claim:

Claim: Let w be a non-empty string over Σ, $|w| = n$. Then $(r_0, \ldots, r_n)$ is a computation of M on w if and only if $r_0 \stackrel{*}{\Rightarrow}_G wr_n$.

Proof of claim: by induction on the length of w.

Base case: $|w| = 1$. Then, if the computation of M on w is (r_0, r_1), by the definition of computation, $\delta(r_0, w) = r_1$, and, by construction of G, the rule $r_0 \to wr_1$ is in the grammar G, and so, by the definition of derivation, $r_0 \Rightarrow wr_1$, and in particular $r_0 \stackrel{*}{\Rightarrow} wr_1$.

On the other hand, if $r_0 \stackrel{*}{\Rightarrow} wr_1$, then by construction of G $r_0 \Rightarrow wr_1$ (because, in G every non-ε rule adds one terminal symbol, and in wr_1 we have exactly one terminal).

Thus, G must contain a rule $r_0 \to wr_1$, which implies that $\delta(r_0, w) = r_1$, and so the computation of M on w is (r_0, r_1).

Inducitve step: assume that the claim is true for any string w of length n. Prove the claim for any string of length $n + 1$.

Exercise: finish the proof of the claim. Make sure that you prove both the "if" and the "only if" directions.

Exercise: finish the proof of the theorem using the Claim. Make sure that you prove both directions of the language equality.

□