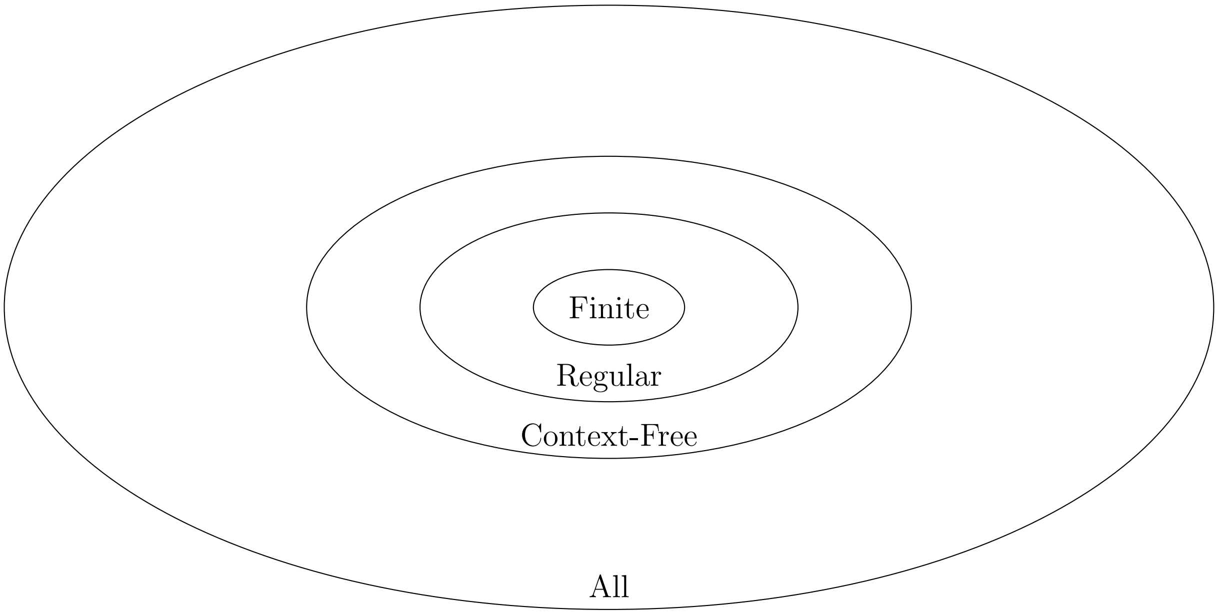
Context-free Languages

So far, we have explored languages using DFA's, NFA's regular expressions, and set notation. We are starting to form a hierarchy of the "patterns" different families of languages can encapsulate. So far we have:



DFA's, NFA's and regular expressions can only describe regular languages, we focus on grammars to describe context-free languages for now.

Context-free languages are encapsulated by **Context-free grammars** which are described by the quadruple :

$$G = \{\Sigma, V, R, S\}$$

where Σ is the terminal alphabet, V is the set of non-terminals (or variables), R is the set of productions and S is the starting variable. Productions have the form:

$$A \to x,$$

where $A \in V$ the set of variables, and $x \in (V \cup \Sigma)^*$, the set of variables union the terminals, kleene starred.

Warm up

Question 1. Let $L = \{a^n b^m a^n b^k \mid n, m, k \geq 0\}$ over $\Sigma = \{a, b\}$. Find a CFG for L.

Answer:

One strategy is to try to simplify the language into a union or concatenation of simpler languages. In this case we notice that

$$L = L_1 \cdot L_2 = \{a^n b^m a^n \mid n, m \geq 0\} \cdot \{b^*\}.$$

So we can create a grammar for each and then glue them together.

A grammar for L_1 can be given by:

$$\begin{aligned} S_1 &\to aS_1a \mid B \mid \lambda \\ B &\to bB \mid \lambda \end{aligned}$$

and one for L_2 can be given by:

$$S_2 \to bS_2 \mid \lambda.$$

So, then using the concatenation in grammars technique, the grammar for all of L is :

$$\begin{aligned} S &\to S_1S_2 \\ S_1 &\to aS_1a \mid B \mid \lambda \\ B &\to bB \mid \lambda \\ S_2 &\to bS_2 \mid \lambda. \end{aligned}$$

This can be simplified to $G(L)$:

$$\begin{aligned} S &\to S_1S_2 \\ S_1 &\to aS_1a \mid S_2 \mid \lambda \\ S_2 &\to bS_2 \mid \lambda. \end{aligned}$$

Question 2. Let $L = \{xy \in \Sigma^* \mid x \text{ and } y \text{ are both palindromes.}\}$ over $\Sigma = \{a, b\}$. Find a grammar for L.

Answer:

Context-free grammars really shine when it comes to palindromes. We can use the same technique as in question 1.

$$L = L_1 \cdot L_2 = \{x \in \Sigma^* \mid x \text{ is a palindrome.}\} \cdot \{y \in \Sigma^* \mid y \text{ is a palindrome.}\}$$

A grammar for L_1 (and L_2) is:

$$S_i \to aS_ia \mid bS_ib \mid a \mid b \mid \lambda$$

Putting L_1 and L_2 together using the concatenation in grammars technique we get $G(L)$:

$$\begin{aligned} S &\to S_1S_2 \\ S_1 &\to aS_1a \mid bS_1b \mid a \mid b \mid \lambda \\ S_2 &\to aS_2a \mid bS_2b \mid a \mid b \mid \lambda. \end{aligned}$$

Which of course can be simplified to:

$$\begin{aligned} S &\to S_1S_1 \\ S_1 &\to aS_1a \mid bS_1b \mid a \mid b \mid \lambda. \end{aligned}$$

Question 3. (2011 Final) Define what one intuitively might mean by properly nested parenthesis structures involving 2 parenthesis say ( ) and [ ]. Think about balance and nesting. Give a grammar that generates all properly nest parentheses. (**Hint:** $\Sigma = \{[,], (,)\}$)

Answer:

Proper nesting means that every left bracket has a right, and it follows the nesting rules of most common programming languages. These ideas are similar to palindromes but have more nesting complexity. A grammar $G(L)$ for such language is:

$$S \to SS \mid [S] \mid (S) \mid \lambda.$$

Another grammar that would work is:

$$S \to S[S]S \mid S(S)S \mid \lambda.$$