2. Context Free Languages

Context-Free Grammar: A context-free grammer (CFG) is a 4-tuple (V, Σ, R, S), where

- V is a finite set called the **variables**;
- Σ is a finite set, disjoint from V, called the **terminals**;
- R is a finite set of **rules**, with each rule being a variable and a string of variables and terminals, that is,

$$V \to (V \cup \Sigma)^*$$

- $S \in V$ is the **start variable**.

Derivations: Let u, v, w be strings of variables and terminals, and

$$A \to w \quad \in R$$

Then uAv yields uwv: $uAv \Rightarrow uwv$. u derives v, written $u \overset{*}{\Rightarrow} v$, if

- $u = v$, or
- there is a sequence $u_1, u_2, \cdots, u_k$ for $k \geq 0$ and

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

Language of the grammar: the language of the grammer is $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$, which is a **context-free language**.

Examples

- Language $\{0^n 1^n \mid n \geq 0\}$, grammer

$$S_1 \to 0S_1 1 \mid \epsilon$$

- Language $\{1^n 0^n \mid n \geq 0\}$, grammer

$$S_2 \to 1S_2 0 \mid \epsilon$$

- Language $\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$

$$S \to S_1 \mid S_2$$

Leftmost Derivations: A derivation of a string w in a grammar G is a **leftmost derivation** if at every step the leftmost remaining variable is the one replaced.

Ambiguity: A string w is derived **ambiguously** is a context free grammer G if it has two or more different leftmost derivations. Grammer G is **ambiguous** if it generates some strings ambiguously.

- For example, $EXP \to EXP + EXP \mid EXP \cdot EXP \mid (EXP) \mid a$. Then the string $a + a \cdot a$ has two different derivations, thus it generate the string **ambiguously**.
- Not all ambiguity can be resolved, called **inherently ambiguous**. For example, $\{a^i b^j c^k \mid i = j \vee j = k\}$.
- **There is no algorithm for resolving ambiguity.**
- **There is not even an algorithm for finding out whether a given CFG is ambiguous.**
- However, there are **standard techniques for writing an unambiguous grammer** that help in most cases.

Chomsky Normal Form: A context-free grammer is in Chomsky normal form if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is any terminal and A, B, C are any variables, except that B and C may be not the start variable. In addition, we permit the rule $S \to \epsilon$, where S is the start variable.

Theorem. Any context-free language is generated by a context-free grammer in Chomsky normal form.

Proof.

1. Add a new start variable S_0 with the rule $S_0 \to S$, where S is the original start variable.
2. Remove every $A \to \epsilon$, where $A \neq S$.
   For each occurrence of A on the right-hand side of a rule, we add a new rule with that occurrence deleted.
   a) $R \to uAv$ we add $R \to uv$;
   b) Do the above operation for *each* occurrence of A: e.g. $R \to uAvAw$, we add $R \to uvAw \mid uAvw \mid uvw$.
   c) For $R \to A$, we add $R \to \epsilon$ unless we had previously removed $R \to \epsilon$.
3. Remove every $A \to B$.
   Whenever a rule $B \to u$ appears, where u is a string of variables and terminals, we add the rule $A \to u$ unless this was previously removed.

4. Replace each rule $A \to u_1 u_2 \cdots u_k$ with $k \geq 3$ and each u_i is a variable or terminal with the rules

$$A \to u_1 A_1, A_1 \to u_2 A_2, A_2 \to u_2 A_3, \cdots, \text{ and } A_{k-2} \to u_{k-1} u_k.$$

The A_i's are new variables. We replace any terminal u_i with the new variable U_i and add $U_i \to u_i$.

Theorem. If G is a context-free grammer in Chomsky normal form then any $w \in L(G)$ such that $w \neq \epsilon$ can be derived from the start state in exactly $2|w| - 1$ steps.

Pushdown automata: A pushdown automata (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- Q is a finite of *states*;
- Σ is a finite set of *input alphabet*;
- Γ is a finite set of *stack alphabet*,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the *transition function*;
- $q_0 \in Q$ is the *start state*;
- $F \subseteq Q$ is the set of *acceptance states*.

Note. The "stack" can contain at most 1 element.

Computation by PDA: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. M accepts input w if w can be written as $w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequences of states $r_0, r_1, \cdots, r_m \in Q$ and strings $s_0, s_1, \cdots, s_k \in \Gamma^*$ exist that satisfy the following three conditions.

- $r_0 = q_0$ and $s_0 = \epsilon$.
- For $i = 0, 1, \cdots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$.
- $r_m \in F$.

Theorem. A language is context free iff some pushdown automaton recognizes it.

Theorem. The context-free languages are closed under union, concatenation and kleene star.

Proof. Let two operand context-free languages be $N_1 = (V_1, \Sigma_1, R_1, S_1)$ and $N_2 = (V_2, \Sigma_2, R_2, S_2)$

- Union: add a new transition $S \to S_1 \mid S_2$;
- Concatenation: add a new transition $S \to S_1 S_2$;
- Kleene star: add a new transition $S \to S S_1 \mid \epsilon$.

Theorem. The intersection of a context-free language with a regular language is a context-free language.

Proof. Let PFA $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, s_1, F_1)$ and DFA $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$, build $M = (Q, \Sigma, \Gamma_1, \Delta, s, F)$.

- $Q = Q_1 \times Q_2$;
- $s = (s_1, s_2)$;
- $F = F_1 \times F_2$;

- Δ is defined as follows:
    - For each PDA rule $(q_1, a, \beta) \to (p_1, r)$ and each $q_2 \in Q_2$, add the following rule to Δ;

$$((q_1, q_2), a, \beta) \to ((p_1, \delta(q_2, a)), r)$$

    - For each PDA rule $(q_1, \epsilon, \beta) \to (p_1, r)$ and each $q_2 \in Q_2$, add the following rule to Δ.

$$((q_1, q_2), a, \beta) \to ((p_1, q_2), r)$$

Lemma. *(the pumping lemma for context-free languages)* If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided as $s = uvxyz$ satisfying the conditions:

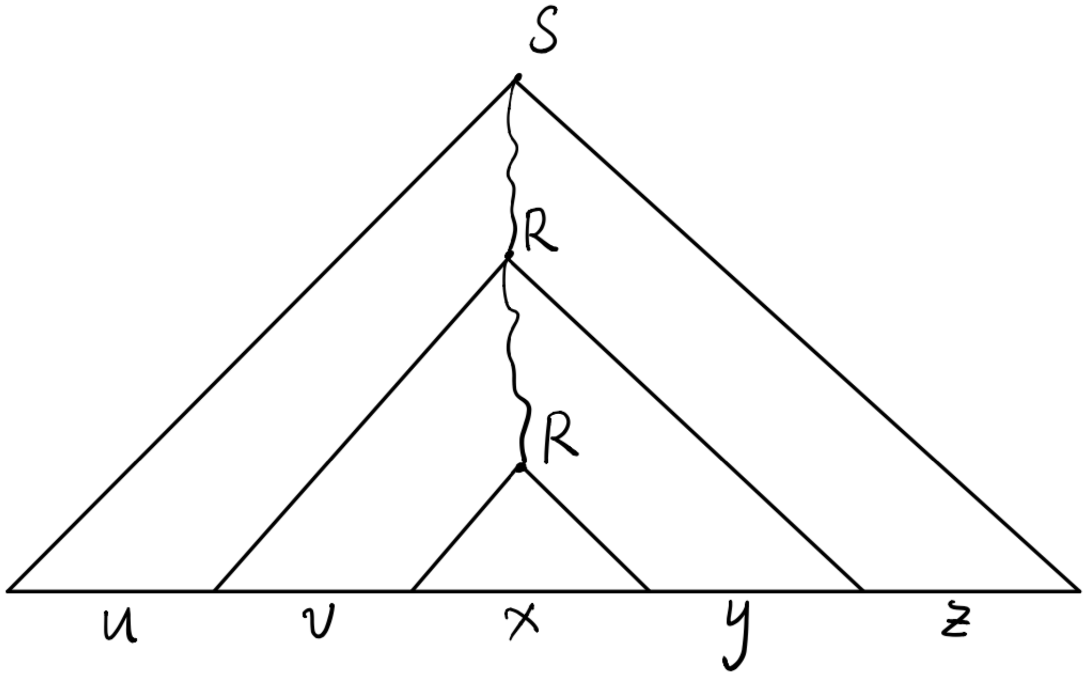
- For each $i \geq 0$, $uv^i xy^i z \in A$;
- $|vy| > 0$;
- $|vxy| < p$.

Proof. Let G be a CFG for CFL A, and let b be the maximum number of symbols in the rhs of a rule. In any parse tree using this grammar, every node can have no more than b children. So, if the height of the parse tree is at most h, the length of the string generated is at most b^h. Conversely, if a generated string is at least $(b^h + 1)$ long, then each of its pase trees must be at least $(h + 1)$ high. Therefore, we choose the pumping length

$$p = b^{|V|+1}$$

Then for any string $s \in A$ with $|s| \geq p$, any of its parse trees must be at least $(|V| + 1)$ high.

Let τ be one parse tree of s with *smallest number of nodes*, whose height is at least $(|V| + 1)$. So τ has a path from the root to a leaf of length $(|V| + 1)$ with $(|V| + 2)$ nodes. One variable R must appear at least twice in the last $(|V| + 1)$ variable nodes on this path. Then we can divide s into $uvxyz$ as follows.



- **Condition 1**: Replace the subtree of the second R by the subtree of the first R will validate the condition;
- **Condition 2**: If $|vy| = 0$, that is, $v = y = \epsilon$, then the path between two R can be eliminated, thus τ cannot have the smallest number of nodes;
- **Condition 3**: To see $|vxy| \leq p = b^{|V|+1}$, note that vxy is generated by the first R. We can always choose R so that the last two occurrences fall within the bottom $(|V| + 1)$ high. A tree of this height can generate a string of length at most $b^{|V|+1} = p$.

Theorem. The context-free language are not closed under intersection or complementation.

Proof. Clearly $\{a^n b^m c^m \mid m, n \geq 0\}$ and $\{a^m b^m c^n \mid m, n \geq 0\}$ are CFL, but there intersection $\{a^n b^n c^n \mid n \geq 0\}$ are not CFL according to the pumping lemma. To the second part of the statement,

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

so CFL is not closed under complementation (otherwise, it should be closed under intersection).

Problems for CFG/PDA

- **Accpetance Problem**: Given a PDA A and a string w, does A accept w?
- **Emptiness Problem**: Given a PDAA, is the language $L(A)$ empty?
- **Equality**: Given two PDA A and B, is $L(A)$ equal to $L(B)$?