Context-free Languages

Context-free Languages

A **context-free grammar** is a 4-tuple $G = (V, \Sigma, R, S)$. Here

1. V is the set of **variables** (or **nonterminals**),
2. Σ is the set of **terminals**,
3. R is the set of **substitution rules** (or **production rules**), each of which is of the form

$$A \to w,$$

for some nonterminal A and some word w over $V \cup \Sigma$; and

4. S is a nonterminal called the **start symbol**.

Substitution

Given a word $x \in (V \cup \Sigma)^*$ of the form yAz and a rule $A \to w$, x can be turned into ywz by substituting the A with w.

Note

- If A does not appear in x, the rule has no effect on x.
- If there are multiple rules for substituting A, then you nondeterministically choose the one you apply.
- If there are multiple occurrences of A, then you nondeterministically choose the one and the rule is applied.

Derivation

We write $u \Rightarrow v$ to mean that v can be produced from u by applying in sequence production rules; that is, if there is a sequence $[u_0, \ldots, u_m]$ of strings over $V \cup \Sigma$ and there is a sequence $[r_1, \ldots, r_m]$ of rules R such that

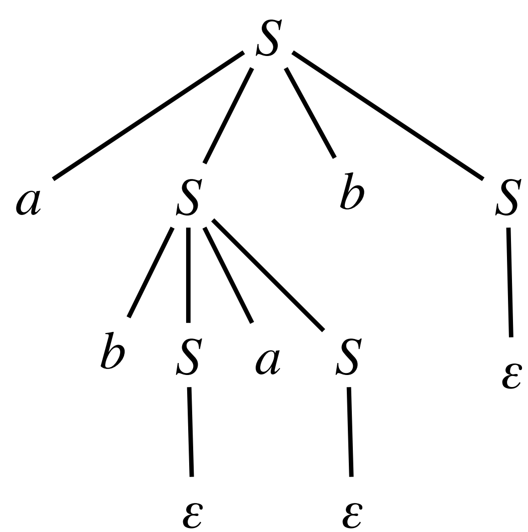
- $u_0 = u$ and $u_m = v$;
- for each i, $1 \leq i \leq m$, u_i can be obtained from u_{i-1} by applying r_i.

We say that G produces $w \in \Sigma^*$ if $S \Rightarrow w$, i.e., w can be obtained from S by derivation.

Parse Tree

A **parse tree** (or **derivation tree**) is a tree that depicts the process of derivation.

Since each derivation step substitutes one nonterminal, the series of substitutions can be visualized using a tree.



Example

The strings over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ consisting of an equal number of $\mathtt{a}$'s and $\mathtt{b}$'s.

Example

The strings over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ consisting of an equal number of a's and b's.

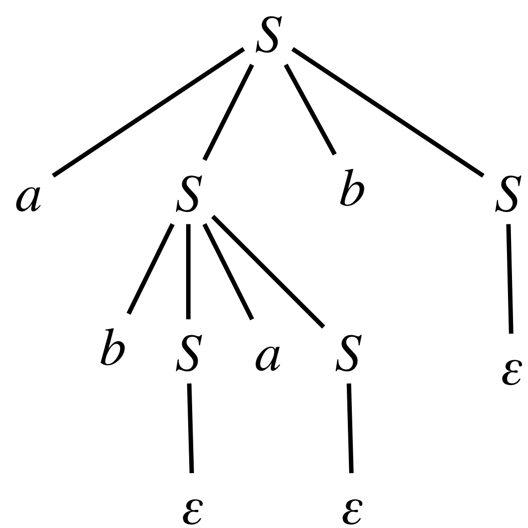
$V = \{S\}$ and the derivation rules are $S \rightarrow \epsilon \mid \mathtt{a}S\mathtt{b}S \mid \mathtt{b}S\mathtt{a}S$.

Example

The strings over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ consisting of an equal number of a's and b's.
$V = \{S\}$ and the derivation rules are $S \to \epsilon \mid \mathtt{a}S\mathtt{b}S \mid \mathtt{b}S\mathtt{a}S$.

abab is derived as follows:

$$S \Rightarrow \mathtt{a}S\mathtt{b}S \Rightarrow \mathtt{ab}S\mathtt{a}S\mathtt{b}S \Rightarrow \mathtt{ab}S\mathtt{ab}S \Rightarrow \mathtt{abab}S \Rightarrow \mathtt{abab}.$$



Why Does the Grammar Work?

Let L be the language. For each word over $\{\mathtt{a}, \mathtt{b}\}$, let $d(w)$ be the number of a's in w minus the number of b's in w. We observe:

- For all w, $w \in L$ if and only if $d(w) = 0$.
- $d(\epsilon) = 0$; $d(\mathtt{a}) = 1$; $d(\mathtt{b}) = -1$.
- For all u and v, $d(uv) = d(u) + d(v)$.

So, if $w \in L$ and if $w_1 = \mathtt{a}$, there exists some $k \geq 2$ such that $d(w_1 \cdots w_k) = 0$ and $d(w_1 \cdots w_{k-1}) = d(w_1) = 1$. This implies that $w(w_2 \cdots w_{k-1}) = d(w_{k+1} \cdots w_n) = 0$.

This gives $S \to \mathtt{a}S\mathtt{b}S$. By exchanging the role between a and b, we have $S \to \mathtt{b}S\mathtt{a}S$.

Ambiguity and Leftmost Derivation

It looks like there are many different ways to produce the same word $w \in L(G)$ for a grammar G.

Ambiguity and Leftmost Derivation

It looks like there are many different ways to produce the same word $w \in L(G)$ for a grammar G.

Of course, this is true because when multiple nonterminals appear on an intermediate word, the order in which the nonterminals are chosen for substitution doesn't affect the word produced.

Ambiguity and Leftmost Derivation

It looks like there are many different ways to produce the same word $w \in L(G)$ for a grammar G.

Of course, this is true because when multiple nonterminals appear on an intermediate word, the order in which the nonterminals are chosen for substitution doesn't affect the word produced.

But what if you force the order to be always from left to right, will sill there exist multiple ways to derive the target word?

Leftmost Derivation

A **leftmost derivation** is the derivation in which each production rule is applied to the leftmost nonterminal at the moment.

Leftmost Derivation

A **leftmost derivation** is the derivation in which each production rule is applied to the leftmost nonterminal at the moment.
For abab in the previous example,

$$S \Rightarrow \mathtt{a}\underline{S}\mathtt{b}S \Rightarrow \mathtt{ab}\underline{S}\mathtt{a}S\mathtt{b}S \Rightarrow \mathtt{aba}\underline{S}\mathtt{b}S \Rightarrow \mathtt{abab}\underline{S} \Rightarrow \mathtt{abab}$$

is a leftmost derivation,

Leftmost Derivation

A **leftmost derivation** is the derivation in which each production rule is applied to the leftmost nonterminal at the moment.
For abab in the previous example,

$$S \Rightarrow \mathtt{a}\underline{S}\mathtt{b}S \Rightarrow \mathtt{ab}\underline{S}\mathtt{a}S\mathtt{b}S \Rightarrow \mathtt{aba}\underline{S}\mathtt{b}S \Rightarrow \mathtt{abab}\underline{S} \Rightarrow \mathtt{abab}$$

is a leftmost derivation, while

$$S \Rightarrow \mathtt{a}S\mathtt{b}\underline{S} \Rightarrow \mathtt{a}S\mathtt{ba}\underline{S}\mathtt{b}S \Rightarrow \mathtt{a}S\mathtt{bab}\underline{S} \Rightarrow \mathtt{a}\underline{S}\mathtt{bab} \Rightarrow \mathtt{abab}$$

isn't one.

Ambiguity and Leftmost Derivation

A context-free grammar is **unambiguous** if it has a unique leftmost derivation for every word it generates. Otherwise, the grammar is **ambiguous**.

There is a context-free language that is **inherently ambiguous** — every grammar that produces the language is ambiguous.

Chomsky Normal Form

A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky normal form** if each rule in R is of the following form:

- $S \to \epsilon$ (note that S is the start symbol).
- $A \to BC$ for some $B, C \in V - \{S\}$ and
- $A \to a$ for some $a \in \Sigma$.

Chomsky Normal Form

A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky normal form** if each rule in R is of the following form:

- $S \to \epsilon$ (note that S is the start symbol).
- $A \to BC$ for some $B, C \in V - \{S\}$ and
- $A \to a$ for some $a \in \Sigma$.

Theorem. **Each context-free language is generated by a Chomsky normal form grammar.**

Converting an Arbitrary CFG to a CNF Grammar

Let $G = (V, \Sigma, R, S)$ be an arbitrary CFG and let $L = L(G)$.

We will convert this to a CNF grammar $G' = (V', \Sigma, R', S_0)$.

Step 1: Finding All “Nullable” Variables

We need to eliminate all rules of the form $A \to \epsilon$.

A variable A of G is **nullable** if $A \Rightarrow \epsilon$; that is, the grammar can produce ϵ from A.

Finding All "Nullable" Variables

We find all "nullable" variables as follows:

- Initialize a set U as the collection of all variables A such that $A \to \epsilon$ is a valid production rule.
- While there is a variable B not in U such that the rule set Rhas $B \to A_1 \cdots A_k$ such that $A_1, \ldots, A_k$ are all variables and all members of U, update U with $U \cup \{B\}$.

Does ϵ Belong to $L(G)$?

After computing U, whether $\epsilon \in L(G)$ can be tested by examining whether $S \in U$

$$S \in U \Leftrightarrow \epsilon \in L(G).$$

If that is the case, we will add later a new rule $S_0 \rightarrow \epsilon$.

Step 2: Initialization

From this point on we will assume that $\epsilon \notin L(G)$.

- Initialize V' with the set V.
- Add to R' all the rules in R of the form $B \to y$ such that y is nonempty, is not equal to B, and has no nullable variables.
- Add a new start variable S_0 to V' and add a new rule $S_0 \to S$.

Step 3: Elimination of All Nullable Variables

For each rule of the form $B \to y$ in R such that a nullable variable appears in y do the following:

- Create all rules produced from $B \to y$ by selecting, independently at each position in y where the letter is a nullable variable, whether to keep the variable in place or replace it with ϵ.
- Add all the rules thus created (which includes the original $B \to y$) into R' except for $B \to \epsilon$ and $B \to B$ if such a rule is at all created.

Step 4: Elimination of Unit Rules

We will remove unit rules.

While R' contains a unit rule $A \to B$ such that $B \in V$, pick such a rule r and do the following:

- Remove r.
- For each rule $B \to w$ in R', add $A \to w$ to R' if $w \neq A$.

Current Situation

Each rule in R' is one of the following forms:

- $A \rightarrow b$ for some $b \in \Sigma$.
- $A \rightarrow w$ for some $w \in (V \cup \Sigma)^*$ having length ≥ 2.

Step 5: Normalization Part 1

We will substitute variables on the right-hand side of any rules having length > 1:

For each terminal d

- add a new variable D,
- add a new rule $D \to d$, and
- for each rule $A \to u$ such that $|u| \geq 2$ and d appears in u, replace each occurrence of d with a D.

Step 5: Normalization Part 1

We will substitute variables on the right-hand side of any rules having length > 1:

For each terminal d

- add a new variable D,
- add a new rule $D \to d$, and
- for each rule $A \to u$ such that $|u| \geq 2$ and d appears in u, replace each occurrence of d with a D.

Now each rule is one of the following forms:

- $A \to b$ for some $b \in \Sigma$.
- $A \to w$ for some $w \in V^*$ having length ≥ 2.

Step 6: Normalization Part 2

We will substitute a long rule by a series of rules:

For each rule $A \to w_1 \ldots w_m$ such that $m \geq 3$, do the following:

- Add a new variable X.
- Replace $A \to w$ by two rules: $A \to w_1 X$ and $X \to w_2 \cdots w_m$.

Step 6: Normalization Part 2

We will substitute a long rule by a series of rules:

For each rule $A \to w_1 \ldots w_m$ such that $m \geq 3$, do the following:

- Add a new variable X.
- Replace $A \to w$ by two rules: $A \to w_1X$ and $X \to w_2 \cdots w_m$.

Conversion is complete.

Example

$V = \{S\}, \Sigma = \{\mathtt{a}, \mathtt{b}\}$, and R consists of $S \to \epsilon \mid \mathtt{a}S\mathtt{b}S \mid \mathtt{b}S\mathtt{a}S$

Step 1 Add $S_0 \to S \mid \epsilon$.

Step 2 Eliminate $S \to \epsilon$. The rules are

$$
\begin{aligned}
S_0 &\to S \mid \epsilon, \\
S &\to \mathtt{ab} \mid \mathtt{ab}S \mid \mathtt{a}S\mathtt{b}S \mid \mathtt{a}S\mathtt{b} \mid \\
&\quad \mathtt{ba} \mid \mathtt{ba}S \mid \mathtt{b}S\mathtt{a}S \mid \mathtt{b}S\mathtt{a}.
\end{aligned}
$$

STEP 3 Eliminate $S_0 \to S$ and add

$$
\begin{aligned}
S_0 &\to \mathtt{ab} \mid \mathtt{ab}S \mid \mathtt{a}S\mathtt{b}S \mid \mathtt{a}S\mathtt{b} \mid \\
&\quad \mathtt{ba} \mid \mathtt{ba}S \mid \mathtt{b}S\mathtt{a}S \mid \mathtt{b}S\mathtt{a}
\end{aligned}
$$

Example (cont'd)

STEP 4 The rules are

$$S_0 \to \epsilon, \quad A \to \mathtt{a}, \quad B \to \mathtt{b},$$
$$S_0 \to AB \mid AX_1 \mid AX_2 \mid AX_3 \mid BA \mid BY_1 \mid BY_2 \mid BY_3,$$
$$S \to AB \mid AX_1 \mid AX_2 \mid AX_3 \mid BA \mid BY_1 \mid BY_2 \mid BY_3,$$
$$X_1 \to BS, \quad X_2 \to SX_4,$$
$$X_3 \to SB, \quad X_4 \to BS,$$
$$Y_1 \to AS, \quad Y_2 \to SY_4,$$
$$Y_3 \to SA, \quad Y_4 \to AS.$$

Here we are using the same variables $X_1, \ldots, X_4$ and $Y_1, \ldots, Y_3$ for S_0 and S.