# Context Free Languages

Last Updated: November 17th, 2025

## 1 Introduction

Context free languages are foundational for defining several types of computing languages that occur in practice; including programming languages, markup languages, and languages for communication protocols. CFL's were first studied in relation to natural-language processing in the 1950's.

Their importance stems from the following.

1. CFL's are significantly more expressive than regular languages in that they are capable of defining recursive languages that may have unlimited recursive depth.
2. a CFL can be recognized by a pushdown automaton (PDA). Unlike DFA's a PDA has unlimited memory, albeit in the form of a stack whose access is limited to i) reading the top of the stack, ii) pushing on to the stack, and iii) popping the top of the stack. PDA's are also interesting because their nondeterministic counterparts (NPDA's) are more powerful than PDA's, and the set of languages accepted by an NPDA is equal to the set of CFL languages.
3. Letting `CFL` denote the class of all context-free languages, it can be shown that $\texttt{CFL} \subseteq \texttt{P}$. This is due to the fact that there exists a dynamic-programming algorithm that can decide any `CFL` language in $\mathrm{O}(n^3)$ steps, where $n$ is the length of the input word. See Sipser, page 290.

Although every regular language is also a CFL (see the exercises), the converse is not true. For example, it can be proved that the language

$$L = \{a^n b^n | n \geq 0\}$$

is a CFL but is *not* regular.

**Proposition 1.** The language

$$L = \{a^n b^n | n \geq 0\}$$

is not regular.

**Proof.**

1. Suppose that $L$ is regular and let $M$ be a DFA that accepts $L$ and has initial state $q_0$.
2. Since $M$ has a finite number of states and there are infinitely many $n$ values, there must be at least one state $q$ for which there is an $n_1 > 0$ and $n_2 > 0$ for which
   (a) $n_1 \neq n_2$
   (b) When either $a^{n_1}$ or $a^{n_2}$ is read starting in state $q_0$, the computation ends at $q$.
   (c) When reading $b^{n_1}$ starting in state $q$, the computation ends in an accepting state $q_a$.
3. But then the facts stated in 2) imply that, when starting in state $q_0$ and reading $a^{n_2}b^{n_1}$, the computation moves to state $q$ after reading $a^{n_2}$, followed by moving to state $q_a$ after reading $b^{n_1}$.
4. Then by 3), it follows that $M$ accepts $a^{n_2}b^{n_1}$ which is a contradiction, since $n_1 \neq n_2$. □

# 2 Context-Free Grammars

A **Context-Free Grammar (CFG)** is a 4-tuple $(V, \Sigma, R, S)$, where

1. $V$ is a finite set of variables
2. $\Sigma$ is a finite set that is disjoint from $V$, called the **terminal set**
3. $R$ is a finite set of rules where each **rule** has the form

   $$A \to s,$$

   where $A \in V$ and $s \in (V \cup \Sigma)^*$. Variable $A$ is referred to as the **head** of the rule, while $s$ is referred to its **body**.
4. $S \in V$ is the **start variable**

$S = B \mid C \mid$

**Example 1.** Consider the set of rules

$$R = \{S \to SS, S \to aSb, S \to \varepsilon\}.$$

Then we may use this set of rules to define a CFG $G = (V, \Sigma, R, S)$, where

$$V = \{S\},$$

$$\Sigma = \{a, b\},$$

and variable $S$ is the start variable.

For brevity we may list together rules having the same head as follows.

$$S \to SS \mid aSb \mid \varepsilon.$$

Here, each of the rule bodies is separated by a vertical bar.

**Example 2.** One common use of CFG's is to provide grammatical formalism for natural languages. For example, consider the set of rules $R$:

$$\langle \text{SENTENCE} \rangle \to \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle$$

$$\langle \text{NOUN-PHRASE} \rangle \to \langle \text{COMPLEX-NOUN} \rangle \mid \langle \text{COMPLEX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle$$

$$\langle \text{VERB-PHRASE} \rangle \to \langle \text{COMPLEX-VERB} \rangle \mid \langle \text{COMPLEX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle$$

$$\langle \text{PREP-PHRASE} \rangle \to \langle \text{PREP} \rangle \langle \text{COMPLEX-NOUN} \rangle$$

$$\langle \text{COMPLEX-NOUN} \rangle \to \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$$

$$\langle \text{COMPLEX-VERB} \rangle \to \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle$$

$$\langle \text{ARTICLE} \rangle \to \text{a} \mid \text{the}$$

$$\langle \text{NOUN} \rangle \to \text{trainer} \mid \text{dog} \mid \text{whistle}$$

$$\langle \text{VERB} \rangle \to \text{calls} \mid \text{pets} \mid \text{sees}$$

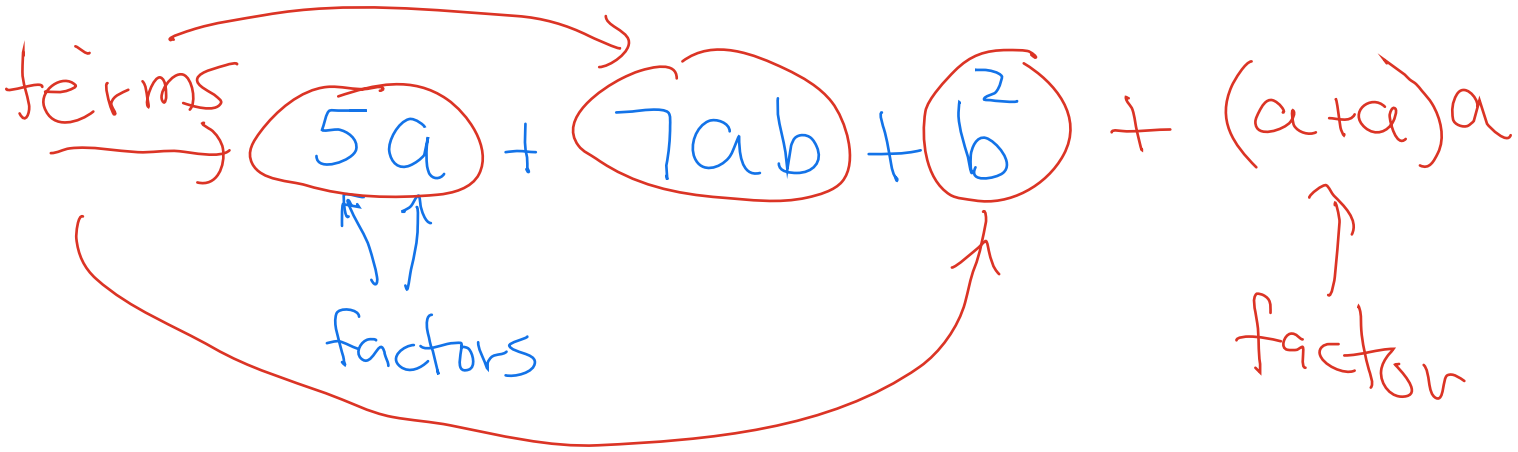
$$\langle \text{PREP} \rangle \to \text{with} \mid \text{in}$$

Here, the variables are the ten parts of speech delimited by $\langle\ \rangle$, $\Sigma$ is the lowercase English alphabet, including the space character, and $\langle \text{SENTENCE} \rangle$ is the start variable.

**Example 3.** A CFG may also be used to define the syntax of a programming language. One fundamental language component to any programming language is that of an *expression*. The following rules imply a CFG for defining expressions formed by a single terminal `a`, parentheses, and the two arithmetic operations + and ×. Here $E$ stands for `expression`, $T$ for `term`, and $F$ for `factor`.

$$E \to E + T \mid T$$
$$T \to T \times F \mid F$$
$$F \to (E) \mid a$$

We have $V = \{E, T, F\}$, $\Sigma = \{+, \times, a, (, )\}$, and $E$ is the start variable.



## 2.1 Grammar derivations

Let $G = (V, \Sigma, R, S)$ be a CFG, then the language $D(G) \in (V \cup \Sigma)^*$ of **derived words** is structurally defined as follows.

**Atom** $S \in D(G)$.

**Compound Rule** Suppose $s \in D(G)$, $s$ is of the form $uAv$ for some $u, v \in (V \cup \Sigma)^*$, $A \in V$, and $A \to \gamma$ is a rule of $G$, then

$$u\gamma v \in D(G).$$

In this case we write

$$s \Rightarrow u\gamma v,$$

and say that $s$ **yields** $u\gamma v$. In words, to get a new derived word, take an existing derived word and replace one of its variables $A$ with the body of a rule whose head is $A$.

The subset $L(G)$ of derived words $w \in D(G)$ for which $w \in \Sigma^*$ is called the **context-free language (CFL)** associated with $G$. Thus, the words of $L(G)$ consist only of terminal symbols.

## 2.2 The Derivation relation

Let $u$ and $v$ be words in $(V \cup \Sigma)^*$. We say that $u$ **derives** $v$, written

$$u \stackrel{*}{\Rightarrow} v$$

if and only if either $u = v$ or there is a sequence of words $w_1, w_2, \ldots, w_n$ such that

$$u = w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n = v.$$

Such a sequence is called a **derivation sequence** from $u$ to $v$.

$$L(G) = \{w \in \Sigma^* | S \stackrel{*}{\Rightarrow} w\}.$$

$\Sigma$ – terminal alphabet

**Example 4.** For the arithmetic-expression CFG from Example 3 and having rule set

$$E \to E + T \mid T$$

$$T \to T \times F \mid F$$

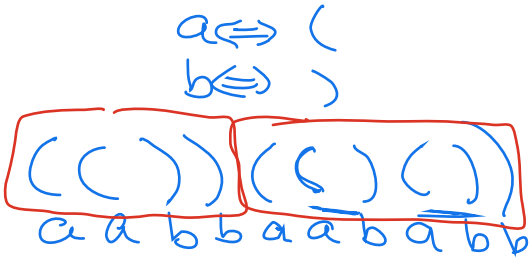
$$F \to (E) \mid a$$

which of the following statements are true?

1. $E \Rightarrow a \times a$ false $E \overset{*}{\Rightarrow} a \times a$ is true
2. $a + F \Rightarrow a + (E)$ true
3. $E \overset{*}{\Rightarrow} a + T$ true
4. $(F) \overset{*}{\Rightarrow} (a \times a)(a + a)$ false

**Example 5.** Use the CFG from Example 1 to derives the word `aabbaababb`.

$$S \to SS \mid aSb \mid \varepsilon.$$




**Solution.**

$S \to SS \to aSbS \to aaSbbS \to aabbS \to aabbaSb \to aabbaSSb \to aabbaaSbSb \to aabbaabSb \to aabbaababb$

## 2.3 Derivation parse trees

Determining if an arbitrary word belongs to $L(G)$ is of fundamental importance. But in addition, it is sometimes important to know the structure of the grammar's derivation of the word. For example, if a CFG generates arithmetic expressions, then knowing the structure of the derivation allows one to readily evaluate the expression (assuming the expression terminals have assigned values and the expression operations are properly defined). A **parse tree** for a word $w \in L(G)$ is a tree whose structure and node labels reflect the derivation $w$, where, from left to right, the leaves of the tree are labeled with the letters of $w$. Indeed, consider the derivation sequence

$$S = w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n = w.$$

Then the parse tree for $w$ can be defined in a step-by-step manner. To begin the parse tree $T_1$ for $S = w_1$ consists of a single node labeled with $S$.

Now suppose a parse tree $T_k$ has been associated with $w_k$, the $k$ th word of the derivation. Moreover, assume that, from left to right, the leaves of $T_k$ are labeled in one-to-one correspondence with the symbols of $w_k$. Moreover, assume that $w_k$ has the form $w_k = uAv$, where $A$ is substituted for a word $\gamma$, so that $w_{k+1} = u\gamma v$. Then $T_{k+1}$ is obtained from $T_k$ by assigning the leaf node labeled with $A$ a number of children equal to the length of $\gamma$ and for which, from left to right, the $i$ th child is labeled with the ith symbol of $\gamma$.

**Example 6.** Use the CFG from Example 3 to derive the expression $a \times (a + a)$, and provide the parse tree associated with the derivation.

$$E \to E + T \mid T$$

$$T \to T \times F \mid F$$

$$F \to (E) \mid a$$

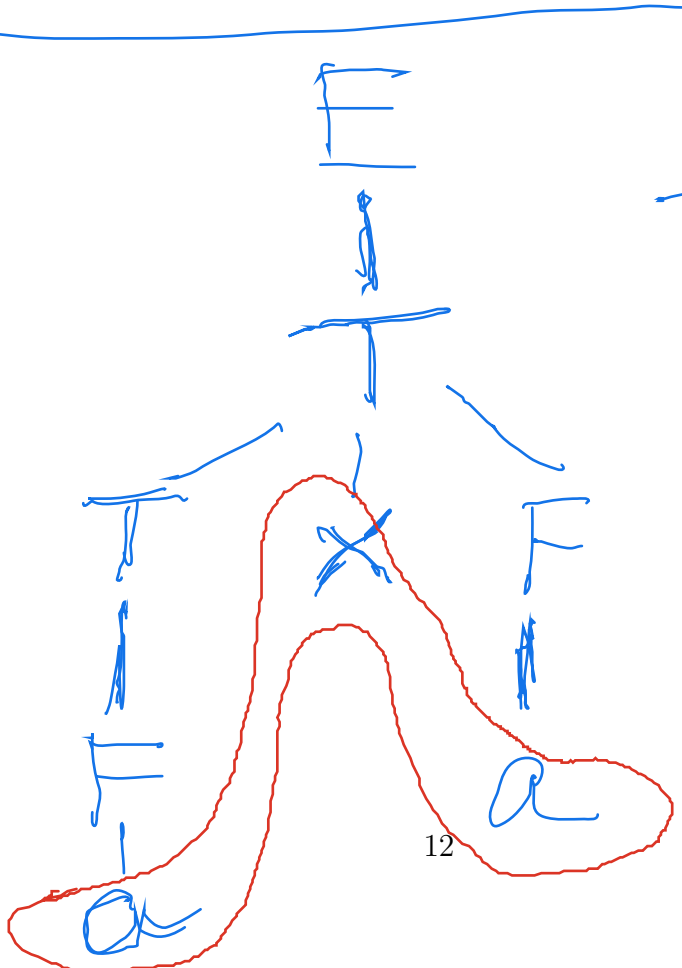
**Solution.** $E \to T \to T \times F \to F \times F \to$

$a \times F \to a \times (E) \to$

$a \times (E + T) \to a \times (T + T) \to$

$a \times (F + T) \to a \times (a + T) \to$

$a \times (a + F) \to a \times (a + a)$




$E \to T$

$T \to T \times F$

$T \to F$

$F \to a$

$a \times a$

## 2.4 Ambiguity

Given a CFG $G$, and a word $w \in L(G)$, there may be several different derivations of $w$ from start symbol $S$. Many of these derivations however will yield identical parse trees. But in the event that two different derivation sequences of $w$ from $S$ yield two different parse trees, then we call $G$ **ambiguous**. It turns out that an easy way to check for ambiguity is to check that no word $w$ has more than one *leftmost derivation*.

Given grammar $G = (V, \Sigma, R, S)$ and word $w \in L(G)$, a derivation sequence $S = w_0 \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_{n-1} \Rightarrow w_n = w$ is called a **leftmost derivation** of $w$ provided that, for every $0 \leq i \leq n-1$, the yielding of $w_i$ from $w_{i-1}$ was obtained by replacing the leftmost variable $A$ of $w_{i-1}$ with the body of one of a rule whose head is $A$. Therefore, if $w$ has more than one leftmost derivation, it must be the case that a different sequence of rules were used to derive $w$. When this happens we call $G$ ambiguous, since some words in the grammar have more than one parsing structure.

**Example 7.** Show that the grammar defined by the following rules is ambiguous.

$$\langle \text{SENTENCE} \rangle \to \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle$$

$$\langle \text{NOUN-PHRASE} \rangle \to \langle \text{COMPLEX-NOUN} \rangle \mid \langle \text{COMPLEX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle$$

$$\langle \text{VERB-PHRASE} \rangle \to \langle \text{COMPLEX-VERB} \rangle \mid \langle \text{COMPLEX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle$$

$$\langle \text{PREP-PHRASE} \rangle \to \langle \text{PREP} \rangle \langle \text{COMPLEX-NOUN} \rangle$$

$$\langle \text{COMPLEX-NOUN} \rangle \to \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$$

$$\langle \text{COMPLEX-VERB} \rangle \to \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle$$

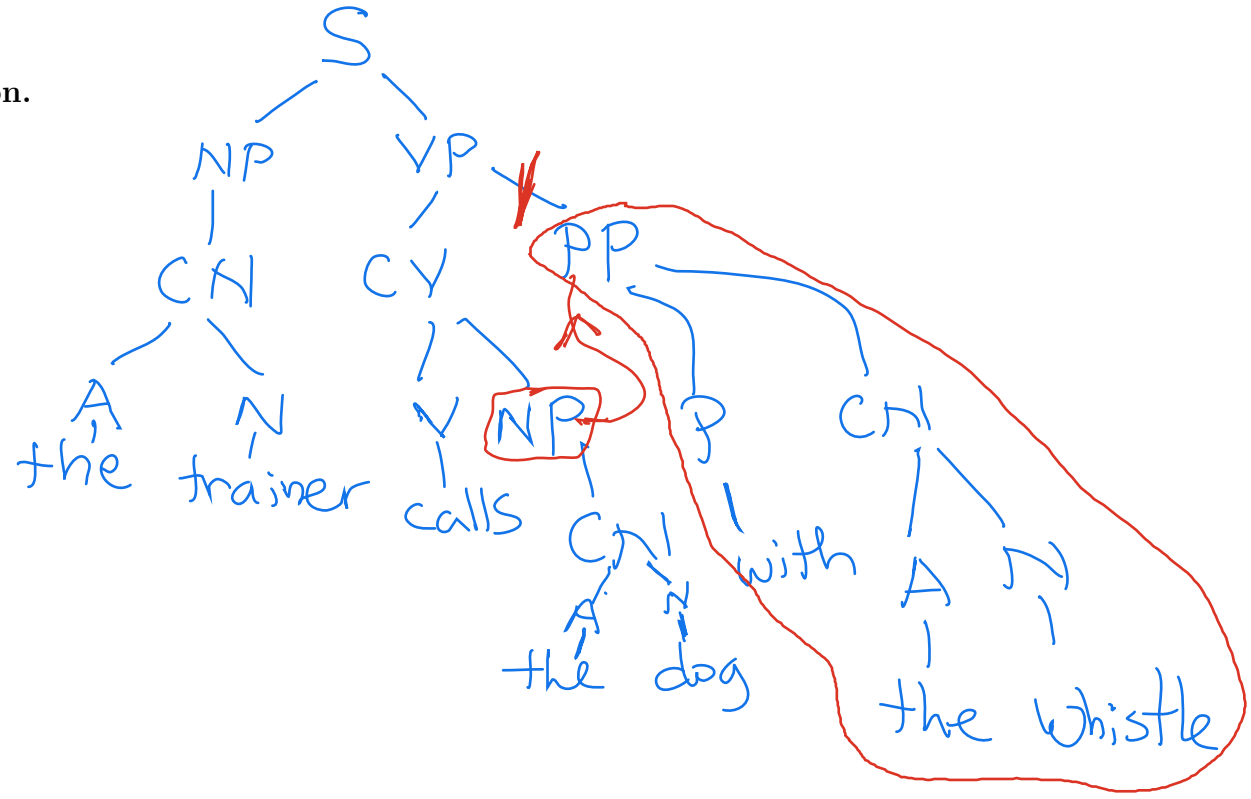
$$\langle \text{ARTICLE} \rangle \to \text{a} \mid \text{the}$$

$$\langle \text{NOUN} \rangle \to \text{trainer} \mid \text{dog} \mid \text{whistle}$$

$$\langle \text{VERB} \rangle \to \text{calls} \mid \text{pets} \mid \text{sees}$$

$$\langle \text{PREP} \rangle \to \text{with} \mid \text{in}$$

**Solution.**



**Solution Continued.**

**Example 8.** Provide a CFG $G$ for which

$$L(G) = \{a^n b^n | n \geq 0\}.$$

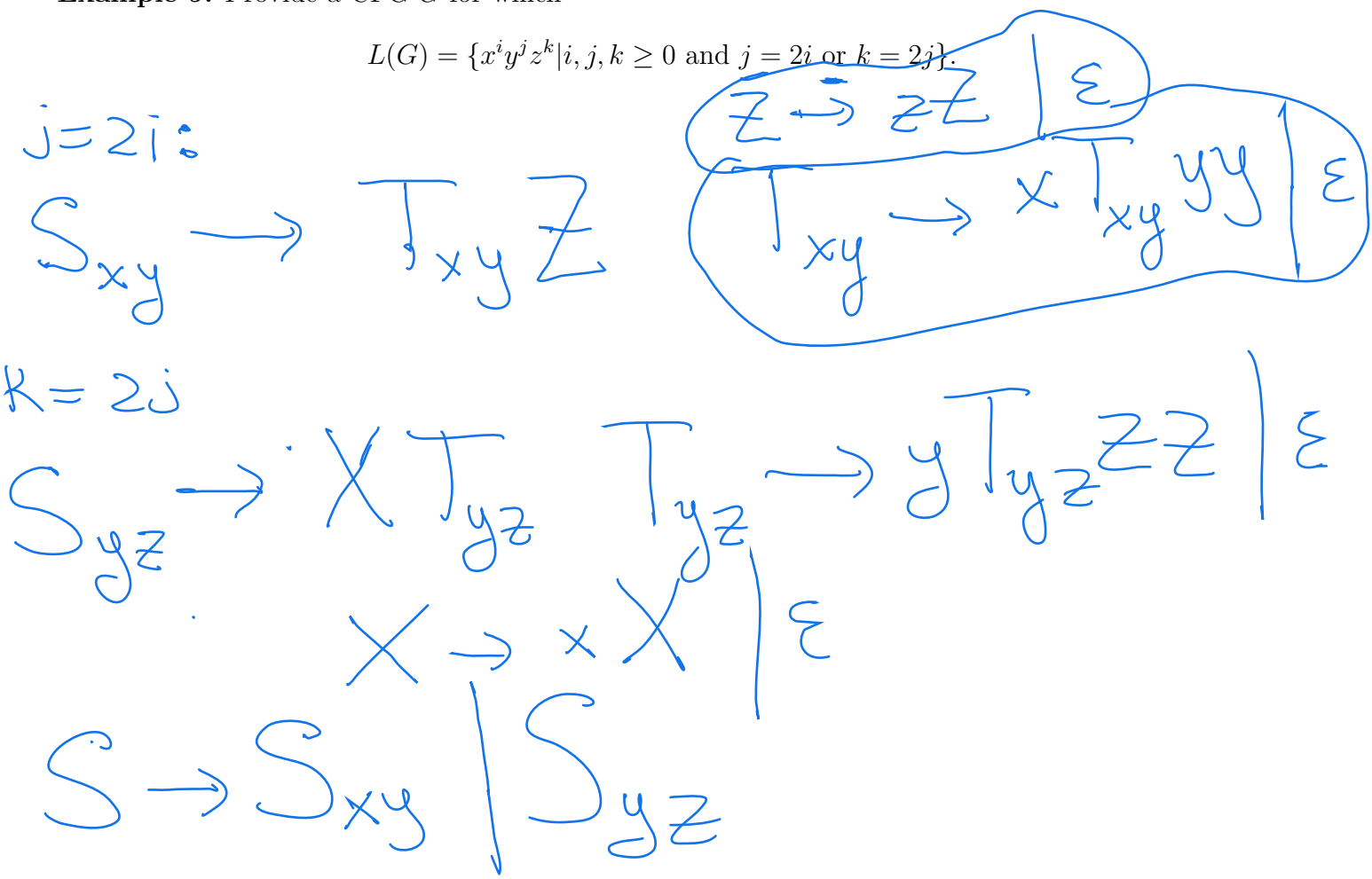
Provide a derivation of $a^3b^3$ and draw its parse tree.

$S \to aSb \mid \varepsilon$

$S \to aSb \to aaSbb \to aaaSbbb \to aaabbb = a^3b^3$

**Example 9.** Provide a CFG $G$ for which

$$L(G) = \{x^i y^j z^k | i, j, k \geq 0 \text{ and } j = 2i \text{ or } k = 2j\}.$$



**Example 10.** Use the CFG from the previous example to provide a derivation of $x^2y^3z^4$ and draw its parse tree.

# 3 Chomsky Normal Form

**Definition 1.** Two CFG's $G_1$ and $G_2$ are said to be equivalent iff $L(G_1) = L(G_2)$.

Sometimes when considering a CFG, it is helpful to assume that its derivations yield binary parse trees. It turns out that any CFG can be converted to an equivalent one that has this property.

**Definition 2.** A CFG $G = (V, \Sigma, R, S)$ is in **Chomsky Normal Form** iff the following conditions hold.

1. The start variable $S$ may not appear on the right-hand side of any rule in $R$.
2. Rule $A \to \varepsilon$ is only allowed when $A = S$.
3. All other rules must be of the form $A \to BC$, where $B, C \in V - \{S\}$, or $A \to a$, where $a \in \Sigma$.

**Algorithm for converting a CFG to one in CNF.**

- **Input:** CFG $G = (V, \Sigma, R, S)$
- **Output:** A new CFG $\hat{G}$ in Chomsky Normal Form that is equivalent to $G$.
- The terminal alphabet of $\hat{G}$ is equal to $\Sigma$
- Initialize the variables and rules of $\hat{G}$ as being equal to those of $G$
- Add a new variable $S_0$ to $\hat{G}$ and designate it as the start variable for $\hat{G}$
- Add the rule $S_0 \to S$ to $\hat{G}$
- **Comment: remove the $\varepsilon$ production rules**
- While there is rule of the form $A \to \varepsilon$ in $\hat{G}$ and $A \neq S_0$
  - For each rule of the form $B \to u_1 A u_2 A \cdots u_n A u_{n+1}$ (where $A$ is not a symbol of any $u_i$ word)
    - Create $2^n$ new rules which represent the different possible ways to make a new rule from $B \to u_1 A u_2 A \cdots u_n A u_{n+1}$ by either keeping or removing each of the $A$ variables.
    - Remove any of these new $2^n$ rules that have the form $D \to \varepsilon$, and have already been processed in the outer while-loop (otherwise the algorithm will loop forever)
  - Add all the newly created rules from the previous loop to $\hat{G}$
  - Remove $A \to \varepsilon$ from $\hat{G}$
- **Comment: eliminate unit rules of the form $A \to B$**
- While there is rule of the form $A \to B$ in $\hat{G}$ and $B \in V$
  - For each rule of the form $B \to u$
    - Create a new rule $A \to u$ and add it to $\hat{G}$ unless it is a unit rule already processed in the outer while loop
  - Remove $A \to B$ from $\hat{G}$
- **Comment: eliminate all remaining rules not in proper form**
- For each rule of the form $A \to u_1 u_2 \dots u_k$, where $k \geq 3$
  - Replace this rule with the set of rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$, …, $A_{k-2} \to u_{k-1} u_k$, where the $A_i$ variables are all new
  - For each $i$, if $u_i$ is a terminal symbol, then replace it with unused variable $U_i$, and add the rule $U_i \to u_i$

**Example 11.** Apply the CNF conversion algorithm to the following CFG.

$S \to ASC|Bb$

$A \to C|S$

$B \to a|\varepsilon$

$C \to b|B$

**Solution.**

1. Add rule $S_0 \to S$

   $\mathbf{S_0 \to S}$
   $S \to ASC|Bb$
   $A \to C|S$
   $B \to a|\varepsilon$
   $C \to b|B$

2. Remove $\varepsilon$-rule $B \to \varepsilon$

   $S_0 \to S$
   $S \to ASC|Bb|\mathbf{b}$
   $A \to C|S$
   $B \to a$
   $C \to b|\varepsilon$

3. Remove $\varepsilon$-rule $C \to \varepsilon$

   $S_0 \to S$
   $S \to ASC|\mathbf{AS}|Bb|b$
   $A \to \varepsilon|S$
   $B \to a$
   $C \to b$

4. Remove $\varepsilon$-rule $A \to \varepsilon$

   $S_0 \to S$
   $S \to ASC|\mathbf{SC}|AS|\mathbf{S}|Bb|b$
   $A \to S$
   $B \to a$

$C \to b$

5. Remove unit rule $S_0 \to S$

$S_0 \to \mathbf{ASC}|\mathbf{SC}|\mathbf{AS}|\mathbf{Bb}|\mathbf{b}$
$S \to ASC|SC|AS|S|Bb|b$
$A \to S$
$B \to a$
$C \to b$

6. Remove unit rule $S \to S$

$S_0 \to \mathbf{ASC}|\mathbf{SC}|\mathbf{AS}|\mathbf{Bb}|\mathbf{b}$
$S \to ASC|SC|AS|Bb|b$
$A \to S$
$B \to a$
$C \to b$

7. Remove unit rule $A \to S$

$S_0 \to ASC|SC|AS|Bb|b$
$S \to ASC|SC|AS|Bb|b$
$A \to \mathbf{ASC}|\mathbf{SC}|\mathbf{AS}|\mathbf{Bb}|\mathbf{b}$
$B \to a$
$C \to b$

8. Add rule $A_1 \to AS$

$S_0 \to \mathbf{A_1}C|SC|AS|Bb|b$
$S \to \mathbf{A_1}C|SC|AS|Bb|b$
$A \to \mathbf{A_1}C|SC|AS|Bb|b$
$B \to a$
$C \to b$
$\mathbf{A_1} \to \mathbf{AS}$

9. Add rule $B_1 \to b$

$S_0 \to A_1C|SC|AS|B\mathbf{B_1}|b$
$S \to A_1C|SC|AS|B\mathbf{B_1}|b$
$A \to A_1C|SC|AS|B\mathbf{B_1}|b$
$B \to a$
$C \to b$
$A_1 \to AS$
$\mathbf{B_1} \to \mathbf{b}$

**Example 12.** Convert the CFG from Example 1 to an equivalent one in Chomsky Normal Form.

**Example 13.** Convert the CFG from Example 3 to an equivalent one in Chomsky Normal Form.

# CFL Core Exercises

1. For the CFG defined in Example 1, provide a derivation for the following words.

   a. ababab

   b. aaababbbab

   c. aababaabbbaabb

2. For the CFG defined in Example 3, provide a derivation and parse tree for the following expressions.

   a. $a$

   b. $a + a$

   c. $a \times (a \times a)$

   d. $((a))$

3. Consider the following rules for some context free grammar $G$.

$$R \to XRX \mid S$$

$$S \to aTb \mid bTa$$

$$T \to XTX \mid X \mid \varepsilon$$

$$X \to a \mid b$$

   Do the following.

   a. Provide the set $V$ of variables, the set $\Sigma$ of terminals, and the start variable. Hint: we may assume that the start variable never appears in the body of a rule, unless the head of the rule is itself the start variable.

   b. Provide three words that are in $L(G)$ and three words that are not in $L(G)$.

   c. Which of the following statements are a) true as stated ? b) true only when $\Rightarrow$ is replaced by $\overset{*}{\Rightarrow}$? c) false even when $\Rightarrow$ is replaced by $\overset{*}{\Rightarrow}$?

      i. $T \Rightarrow aba$

      ii. $T \Rightarrow T$

      iii. $XXX \Rightarrow aba$

      iv. $X \Rightarrow aba$

      v. $T \Rightarrow XX$

      vi. $T \Rightarrow XXX$

      vii. $S \Rightarrow \varepsilon$

   d. Provide a succinct natural-language description of $L(G)$.

4. For each of the following context free languages, provide a set of rules that describes the language. Assume that the terminal set for each is $\{0, 1\}$.

a. All binary words with at least three 1's

b. All binary words that start and end with the same symbol

c. All binary words having odd length

d. All binary words of positive even length and for which the middle two bits are 00

e. All binary words that are palindromes (i.e. read the same forwards as backwards)

f. All binary words for which there are twice as many 0's as 1's

g. The empty set

h. The set $\{\varepsilon\}$

i. All binary words for which there are more 1's than 0's.

j. The complement of $\{a^n b^n | n \geq 0\}$

# Exercise Solutions

1. We have the following derivations.

   a. ababab

   $$\text{S} \Rightarrow \text{SS} \Rightarrow \text{aSbS} \Rightarrow \text{abS} \Rightarrow \text{abSS} \Rightarrow \text{abaSbS} \Rightarrow \text{ababS} \Rightarrow \text{ababaSb} \Rightarrow \text{ababab}.$$

   b. aaababbbab

   $$\text{S} \Rightarrow \text{SS} \Rightarrow \text{aSbS} \Rightarrow \text{aaSbbS} \Rightarrow \text{aaSSbbS} \Rightarrow \text{aaaSbSbbS} \Rightarrow \text{aaabSbbS} \Rightarrow \text{aaabaSbbbS}.$$

   $$\Rightarrow \text{aaababbbS} \Rightarrow \text{aaababbbaSb} \Rightarrow \text{aaababbbab}.$$

   c. aababaabbbaabb

   $$\text{S} \Rightarrow \text{SS} \Rightarrow \text{aSbS} \Rightarrow \text{aSSbS} \Rightarrow \text{aaSbSbS} \Rightarrow \text{aabSbS} \Rightarrow \text{aabSSbS}$$
   $$\Rightarrow \text{aabaSbSbS} \Rightarrow \text{aababSbS} \Rightarrow \text{aababaSbbS} \Rightarrow \text{aababaaSbbbS} \Rightarrow \text{aababaabbbS}$$
   $$\Rightarrow \text{aababaabbbaSb} \Rightarrow \text{aababaabbbaaSbb} \Rightarrow \text{aababaabbbaabb}.$$

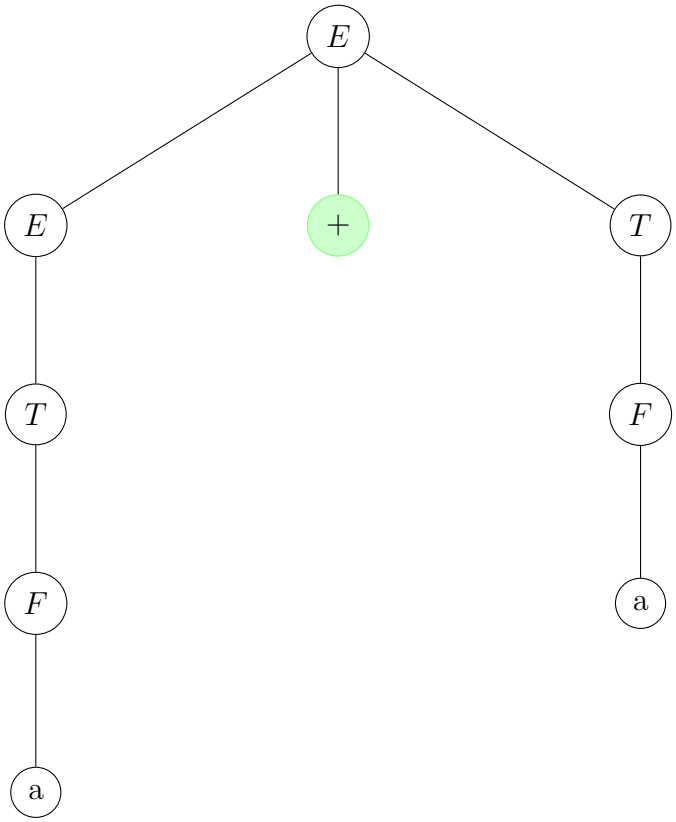
2. For the CFG defined in Example 3, provide a derivation and parse tree for the following expressions.

   a. $a$

   $$E \Rightarrow T \Rightarrow F \Rightarrow \text{a}.$$


b. $a + a$

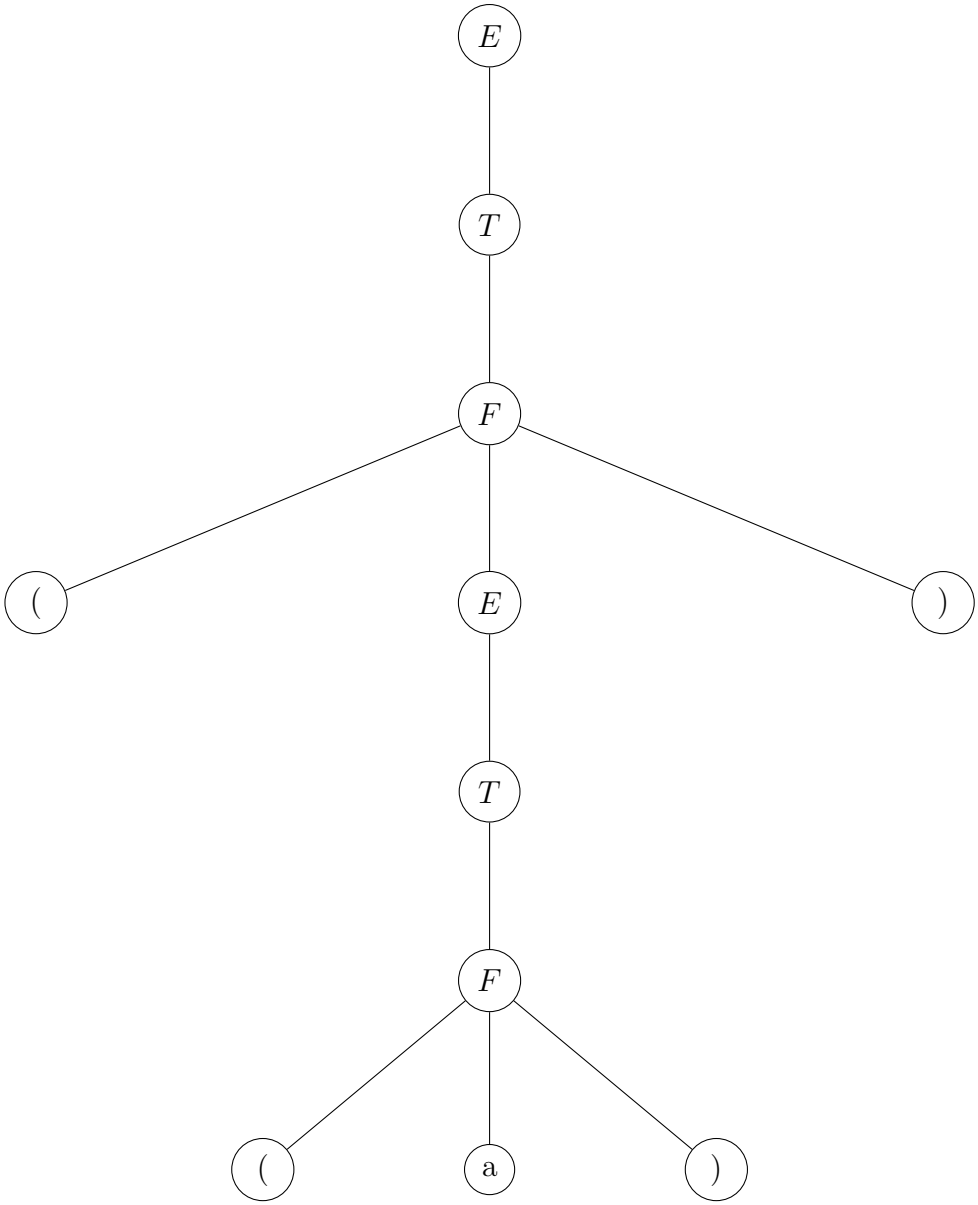
$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow \mathrm{a} + T \Rightarrow \mathrm{a} + F \Rightarrow \mathrm{a} + \mathrm{a}.$$




c. $a \times (a \times a)$

d. $((a))$

$$E \Rightarrow T \Rightarrow F \Rightarrow (E) \Rightarrow (T) \Rightarrow (F) \Rightarrow ((E)) \Rightarrow T \Rightarrow F \Rightarrow ((\mathrm{a})).$$




3. We have the following.

   a. $V = \{R, S, T, X\}$, $\Sigma = \{a, b\}$, and $R$ is the start variable.

   b. $aa, bb, aab \in L(G)$ and $\varepsilon, a, b \notin L(G)$.

   c. Statements iv and vii are false even when "yields" is replaced by "derives". The remaining statements are true when "yields" is replaced by "derives". For iii) it's important to note that, by definition, only one variable replacement is allowed per step, and so it would take three steps to derive $aba$.

   d. All words consisting of a's and b's and having length at least two.

4. We have the following rule sets.

   a. All binary words with at least three 1's

$$S \to 1S1S1S \mid 0 \mid 1 \mid \varepsilon$$

b. All binary words that start and end with the same symbol

$$S \to 0T0 \mid 1T1 \mid 0 \mid 1 \mid \varepsilon$$
$$T \to 0T \mid 1T \mid \varepsilon$$

c. All binary words having odd length

$$S \to 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 0 \mid 1$$

d. All binary words of positive even length and for which the middle two bits are 00

$$S \to 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 00$$

e. All binary words that are palindromes (i.e. read the same forwards as backwards)

$$S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid 00 \mid 11$$

f. All binary words for which there are twice as many 0's as 1's

$$S \to S0S0S1S \mid S1S0S0S \mid S0S1S0S \mid \varepsilon$$

Notice how the rules provide maximum flexibility in terms of which 1 and two 0's are going to be written. This is needed because there could be any ordering of the 1's and 0's. For example, we could have 110000000011. Provide a derivation of this word using the above rules.

g. Any CFG $G = (V, \Sigma, R, S)$ for which $R = \emptyset$.

h. The set $\{\varepsilon\}$

$$S \to \varepsilon$$

i. All binary words for which there are more 1's than 0's.

$$S \to S1S0S \mid S0S1S \mid 1S \mid \varepsilon$$

j. The complement of $\{a^n b^n | n \geq 0\}$. Hint. Notice that the set is equal to the subset of $\{a, b\}^*$ having the form $A \cup B \cup C$, where $A$ consists of odd-length words, $B$ consists of nonempty even-length words for which there is at least one b in the first half of the word, and $C$ consists of nonempty even-length words for which there is at least one a in the second half of the word. Then we have

$$S \to X \mid Y \mid Z$$

where $X$, $Y$, and $Z$ are the respective start symbols for CFG's that describe languages $A$, $B$, and $C$, respectively. Now provide GFG's for each of these languages and take the union the grammars (making sure that the variable set for each grammar is pairwise disjoint with the other two variable sets).