# 1 Context-Free Grammars

Context-free languages are useful for studying computer languages as well as human languages.

- Context-free languages are recognized by push-down automata (PDA) in the same way that regular languages are recognized by finite automata.

- A push-down automaton has an infinite amount of memory but it is only accessed in a last-in first-out (LIFO) manner, that is, like a stack.

- Thus push-down automata are more powerful than finite automata; for example, $\{a^n b^n : n \geq 0\}$ is a context-free language.

- But push-down automata are less powerful than Turing machines, which we'll study later.

- Turing machines have an infinite amount of memory that can be accessed in an arbitrary manner.

## 1.1 Example: Subset of English

We'll start with an example of a context-free grammar and a context-free language and then proceed to the general formalism.

Suppose we consider a restricted subset of English with the following parts of speech:

| | |
|---|---|
| $S$ | sentence |
| $U$ | subject |
| $P$ | predicate |
| $V$ | verb |
| $O$ | object |
| $A$ | article |
| $N$ | noun |

- Then we can write $S \rightarrow UP.$, for example, to indicate that a sentence can be a subject followed by a predicate followed by a period.

- This is called a *rule* and a collection of rules is called a *grammar*.

So here is a simple grammar for a very small subset of English:

| | | |
|---|---|---|
| $S \to UP.$ | $N \to$ boy_ | $N \to$ girl_ |
| $U \to AN$ | $N \to$ ball_ | $U \to$ he_ |
| $P \to VO$ | $N \to$ rock_ | $U \to$ she_ |
| $O \to AN$ | $N \to$ pumpkin_ | $U \to$ it_ |
| $A \to$ a_ | $V \to$ hit_ | $O \to$ him_ |
| $A \to$ the_ | $V \to$ threw_ | $O \to$ her_ |
| | $V \to$ ate_ | $O \to$ it_ |

- The upper case letters are called *nonterminals* and correspond to parts of speech in an English grammar.
- The lower case letters are called *terminals* and are what actually appears in sentences.
- Sentences can be derived by starting from the *start symbol*, here $S$, and continuing to do replacements using these rules until all the nonterminals are eliminated.

Here is an example derivation, with $\Rightarrow$ being used to indicate a replacement using a rule:

$S \Rightarrow UP. \Rightarrow ANP. \Rightarrow$ the_$NP. \Rightarrow$ the_boy_$P. \Rightarrow$ the_boy_$VO. \Rightarrow$ the_boy_hit_$O \Rightarrow$ the_boy_hit_$AN. \Rightarrow$ the_boy_hit_a_$N. \Rightarrow$ the_boy_hit_a_ball_.

Other sentences can also be derived such as

a_boy_threw_the_rock_.
the_ball_hit_it_.
a_pumpkin_ate_the_ball_.
a_ball_threw_a_pumpkin_.

and so on.

- Clearly this is not a complete model of English grammar!
- For that, it is necessary to add some information about semantics, or the meaning of words.
- However, it seems that the human mind naturally forms grammars that are similar to context-free grammars, which helps to show the importance of context-free grammars.

## 1.2 General Formalism

In general a *context-free grammar* $G$ is a 4-tuple $(V, \Sigma, R, S)$ where $V$ is a set of *variables*, $\Sigma$ is an alphabet of *terminal symbols*, $R$ is a set of *rules*, and $S$ is a *start symbol*.

The elements of $V - \Sigma$ are called *nonterminals* and are analogous to parts of speech.

Here is an example grammar:

$G = (V, \Sigma, R, S)$ where $V = \{S, a, b\}$, $\Sigma = \{a, b\}$, and $R$ has the rules $S \to aSb$ and $S \to \epsilon$.

- Nonterminals are usually represented by capital letters and terminals by lower case letters.
- Therefore one can give a context-free grammar just by giving the rules and the start symbol, without giving a 4-tuple.

So the preceding grammar could be represented this way:

$$\begin{gathered} S \to aSb \\ S \to \epsilon \end{gathered}$$

Here is a *derivation* in this grammar:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb.$$

Such a derivation has to eliminate the nonterminals. We can also write this derivation as

$$S \Rightarrow^* aabb.$$

Given a context-free grammar $G$, the *language generated by* $G$, $L(G)$, is the set of strings of terminals that can be derived from the start symbol of $G$.

> A language $L$ is *context free* if it is $L(G)$ for some context-free grammar $G$.

For this grammar, $L(G) = \{a^n b^n : n \geq 0\}$. Thus $\{a^n b^n : n \geq 0\}$ is a context-free language. This shows that not all context-free languages are regular languages.

Formally, a *context-free grammar* $G$ is a quadruple $(V, \Sigma, R, S)$ where

| | |
|---|---|
| V | is an alphabet |
| $\Sigma$ | (the set of *terminals*) is a subset of $V$ |
| R | (the set of *rules*) is a finite subset of $(V - \Sigma) \times V^*$ |
| S | (the *start symbol*) is an element of $V - \Sigma$ |

Members of $V - \Sigma$ are called *nonterminals*. Rules $(A, u)$ are written as $A \rightarrow_G u$ for $A \in V - \Sigma$ and $u \in V^*$.

- We write $u \Rightarrow_G v$ if there are strings $x, y \in V^*$ and $A \in V - \Sigma$ such that $u = xAy$, $v = xwy$, and $A \rightarrow_G w$. That is, $u \Rightarrow v$ means $v$ can be obtained from $u$ by using a rule $A \rightarrow_G w$ and replacing an occurrence of $A$ in $u$ by $w$ to obtain $v$.

- $\Rightarrow_G^*$ is the reflexive transitive closure of $\Rightarrow_G$. So $u \Rightarrow_G^* v$ means that $v$ can be obtained from $u$ by some number of replacements using rules of $G$, possibly no replacements, possibly one or more replacements.

- $L(G)$, the *language generated by* $G$, is $\{w \in \Sigma^* : S \Rightarrow_G^* w\}$. If $L = L(G)$ for some context-free grammar $G$ then $L$ is said to be a *context-free language*.

A sequence

$$w_o \Rightarrow_G w_1 \Rightarrow_G w_2 \Rightarrow_G \ldots \Rightarrow_G w_n$$

is called a *derivation* in $G$ of $w_n$ from $w_0$. Also, $n$ is the *length* of the derivation.

## 1.3 Example: Arithmetic Expressions

Here is another example of a context-free grammar. For this one, we just give the rules:

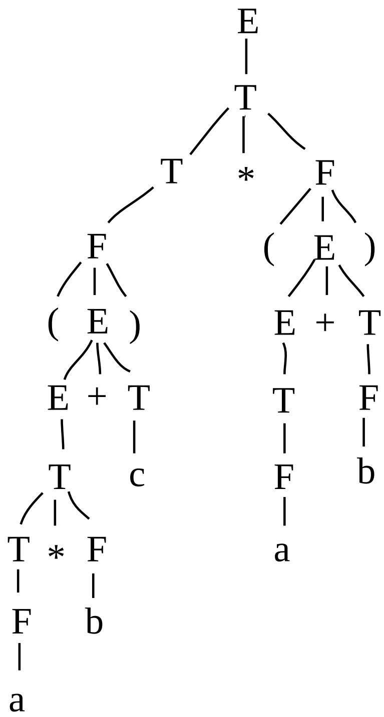
$$\begin{array}{ll} E \to E + T & F \to (E) \\ E \to T & F \to a \\ T \to T * F & F \to b \\ T \to F & F \to c \end{array}$$

Also, $E$ is the start symbol. $E$ represents "expression," $T$ represents "term," and $F$ represents "factor." In this grammar, we can derive strings such as $(a * b + c) * (a + b)$:

$$\begin{gathered} E \Rightarrow \\ T \Rightarrow \\ T * F \Rightarrow \\ T * (E) \Rightarrow \\ T * (E + T) \Rightarrow \\ F * (E + T) \Rightarrow \\ (E) * (E + T) \Rightarrow \\ (E + T) * (E + T) \Rightarrow \\ (T + T) * (E + T) \Rightarrow \\ (T * F + T) * (E + T) \Rightarrow \\ (F * F + T) * (E + T) \Rightarrow \\ (a * F + T) * (E + T) \Rightarrow \\ (a * b + T) * (E + T) \Rightarrow \\ (a * b + F) * (E + T) \Rightarrow \\ (a * b + c) * (E + T) \Rightarrow \\ (a * b + c) * (T + T) \Rightarrow \\ (a * b + c) * (F + T) \Rightarrow \\ (a * b + c) * (a + T) \Rightarrow \\ (a * b + c) * (a + F) \Rightarrow \\ (a * b + c) * (a + b) \end{gathered}$$

- This is a grammar for a limited subset of arithmetic expressions.
- Such grammars are used in programming languages.
- It is designed so that multiplication will have precedence over addition so that for example in the expression $a * b + c$, the multiplication is done before the addition.

The above derivation is very lengthy. In order to avoid so much repeated writing, such derivations are often represented as *parse trees*, as follows:




Context-free grammars describe programming languages better than natural human languages, but even programming languages are not fully described by context-free grammars. Still, many parsers for programming languages are based on the theory of context-free languages.

## 1.4 Example: Balanced Parenthesis Expressions

Here is another context-free grammar:

$$S \to \epsilon$$
$$S \to SS$$
$$S \to (S)$$

Also, $S$ is the start symbol. In this grammar one can derive the *balanced paretheses expressions*, which are strings like ()(()) and ()() in which parentheses are nested. This language is not regular; can you show it?

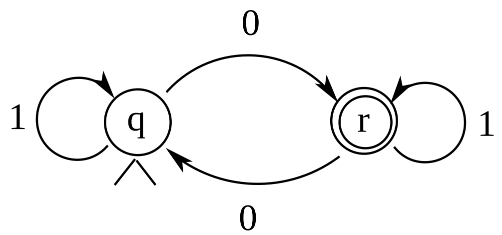
We now show that *all regular languages are context-free.*

## 1.5 Regular languages are context-free

- Suppose $M = (K, \Sigma, \delta, s, F)$ is a deterministic finite state automaton.
- Let $G(M)$ be the context-free grammar $(V, \Sigma, R, S)$ where $V = K \cup \Sigma$, $S = s$, and $R$ has the rules $R = \{q \to ap : \delta(q, a) = p\} \cup \{q \to \epsilon : q \in F\}$.
- Clearly $L(G(M))$ is a context-free language.

But it can be shown that $L(G(M)) = L(M)$, which shows that $L(M)$ is context-free. Because any regular language can be expressed as $L(M)$ for some dfa $M$, this shows that all regular languages are context-free.

Example: Let $M$ be the following automaton:




For this automaton $M$, $G(M) = (\{q, r, 0, 1\}, \{0, 1\}, R, q)$ where $R$ has the rules

$$q \to 0r, q \to 1q, r \to 0q, r \to 1r, r \to \epsilon.$$

The string 0100 is accepted by $M$, with the following computation:

$$q \xrightarrow{0} r \xrightarrow{1} r \xrightarrow{0} q \xrightarrow{0} r.$$

This corresponds to the following derivation in $G$:

$$q \to 0r \to 01r \to 010q \to 0100r \to 0100.$$

- In the same way, arbitrary derivations of a string $w$ in $G(M)$ correspond to accepting computations of the string $w$ in $M$.
- Thus $w \in L(G(M))$ iff $w \in L(M)$, so $L(G(M)) = L(M)$, showing that $L(M)$ is context-free.
- This same construction can be done for an arbitrary finite state automaton $M$, showing that all regular languages are context-free.
- We know that there is at least one context-free language that is not a regular language, so the regular languages are a *proper subset* of the context-free languages.

## 1.6 Problems

Do problem 3.1.3, page 120. Also give a context-free grammar for $\{a^n b^m : n \neq m\}$.

Do problem 3.19(b), page 122.

Generate a context-free grammar for $\{a^i b^j c b^j a^i : i, j \geq 0\}$.

Give a context-free grammar for $\{a^i b^j c b^k a^l : k \geq j \geq 0, l \geq i \geq 0\}$.

It is useful to know how to generate a context-free grammar for a language because this is often done for programming languages.

## 1.7 Computer Languages

Look at the links on the course web page about the relationship of Algol 60 and other computer languages to context-free languages.

## 1.8 Conjecture

This quotation was in an email received March 30, 2015 advertising a new book, "Context-Free Languages and Primitive Words."

> A word is said to be primitive if it cannot be represented as any power of another word. It is a well-known conjecture that the set of all primitive words Q over a non-trivial alphabet is not context-free: this conjecture is still open.