

Lecture 14.1: CFG Definitions, Terminologies, Examples-1

Presenter: Azwad Anjum Islam (AAI)

Scribe: Mujtahid Al-Islam Akon (AKO)

In this lecture, you will learn the formal definition of Context-free Grammar and some examples to show you how to create CFGs for some basic problems.

Formal definition of a CFG

A CFG is formally described by a four-tuple, $G = \langle V, T, P, S \rangle$ where –

- V is the set of non-terminal symbols.
- T is the set of terminal symbols.
- P is the set of production rules
- S is the start symbol. Note, $S \in V$

Every rule is of the form $A \rightarrow (V \cup T)^*$ where $A \in V$.

Derivation: We usually say, S derives a string w if we can reach to w starting from S by a number of replacements or substitutions of production rules.

Shortcut notation for rule set: The rule set is usually described in a shortcut notation as follows.

General	Short Notation
$S \rightarrow 0S1$ $S \rightarrow \epsilon$	$S \rightarrow 0S1 \epsilon$

In the shortcut notation, all production rules for a single non-terminal symbol are grouped and separated by a vertical bar symbol ($|$).

CFL as a Superset of RL

As Context-free Language is a superset of Regular Language, each Regular Expression must have an equivalent Context-free Grammar. It is sufficient to show that each of three basic regular operations can be achieved using CFG.

Basic Regular Operation	Regular Expression	Context-free Grammar	Optimized version
OR	$a b$	$S \rightarrow A$ $S \rightarrow B$ $A \rightarrow a$ $B \rightarrow b$	$S \rightarrow A B$ $A \rightarrow a$ $B \rightarrow b$
		$S \rightarrow a$ $S \rightarrow b$	$S \rightarrow a b$

Concatenation	ab	$S \rightarrow AB$ $A \rightarrow a$ $B \rightarrow b$	-
		$S \rightarrow ab$	-
Kleene Star	a^*	$S \rightarrow AS$ $A \rightarrow a$ $S \rightarrow \epsilon$	$S \rightarrow AS \epsilon$ $A \rightarrow a$
		$S \rightarrow aS$ $S \rightarrow \epsilon$	$S \rightarrow aS \epsilon$
		$S \rightarrow SA$ $A \rightarrow a$ $S \rightarrow \epsilon$	$S \rightarrow SA \epsilon$ $A \rightarrow a$
		$S \rightarrow SA$ $A \rightarrow a$ $S \rightarrow \epsilon$	$S \rightarrow Sa \epsilon$

How $S \rightarrow aS|\epsilon$ works as a^* :

1. As a^* contains ϵ , we get $S \rightarrow \epsilon$
2. Let $S = aaaaaa \dots aaaa$. The bold portion looks exactly the same as S . So, we can replace the the **bold** portion with an S (See the figure). So, we get $S \rightarrow aS$.
3. Combine these two we get $S \rightarrow aS|\epsilon$.

$$S = aaaaaa \dots aaaa$$

$$\therefore S = a \mathbf{S}$$

Verify yourself to see how the above logic works after deriving some strings.

Some Examples of CFG

$$\{w | w = 0^n 1^n, n \geq 0\}$$

Language/Problem	CFG	Explanations
$\{w w = 0^n 1^n, n \geq 0\}$	$S \rightarrow 0S1$ $S \rightarrow \epsilon$	$S = 000 \dots 111$ $\therefore S = 0 \mathbf{S} 1$
$\{w w = 0^n 1^{2n}, n \geq 0\}$	$S \rightarrow 0S11$ $S \rightarrow \epsilon$	Find out yourself

Grammar for valid mathematical Expressions

Operators to consider: $+$ $-$ \times \div $()$

Assume, all numbers are single digit integers.

Let E be the start symbol for the expression.

- A single digit number can be any of the 10 decimal digits yielding the grammar:

$$Num \rightarrow 0|1|2|3|4|5|6|7|8|9$$

- Note, any number is by default an expression. This yields $E \rightarrow Num$
- Any two expressions can be connected putting any binary operator between them. This yields the following-

$$E \rightarrow E + E$$

$$E \rightarrow E - E$$

$$E \rightarrow E \times E$$

$$E \rightarrow E \div E$$

- We can put parenthesis around an expression to make it a bundle to maintain precedence. This yields-

$$E \rightarrow (E)$$

- Combining,

$$E \rightarrow (E)$$

$$E \rightarrow E + E$$

$$E \rightarrow E - E$$

$$E \rightarrow E \times E$$

$$E \rightarrow E \div E$$

$$E \rightarrow Num$$

$$Num \rightarrow 0|1|2|3|4|5|6|7|8|9$$

Let's see how this grammar derives few expressions:

$4 \times (3 - 2)$	$4 \times 3 - 2$
$E \rightarrow E \times E$	$E \rightarrow E \times E$
$\rightarrow E \times (E)$	$\rightarrow E \times E - E$
$\rightarrow E \times (E - E)$	$\rightarrow Num \times Num - Num$
$\rightarrow Num \times (Num - Num)$	$\rightarrow 4 \times 3 - 2$
$\rightarrow 4 \times (3 - 2)$	

Though the above grammar works for identifying correct mathematical expressions, interestingly it cannot evaluate the expressions correctly always. For example: the expression on the right ($4 \times 3 - 2$) should evaluate to 10. But the way I derived in the above table will evaluate to 4! Can you find out the correct derivation to evaluate to 10?