Context-Free Grammar (CFG)

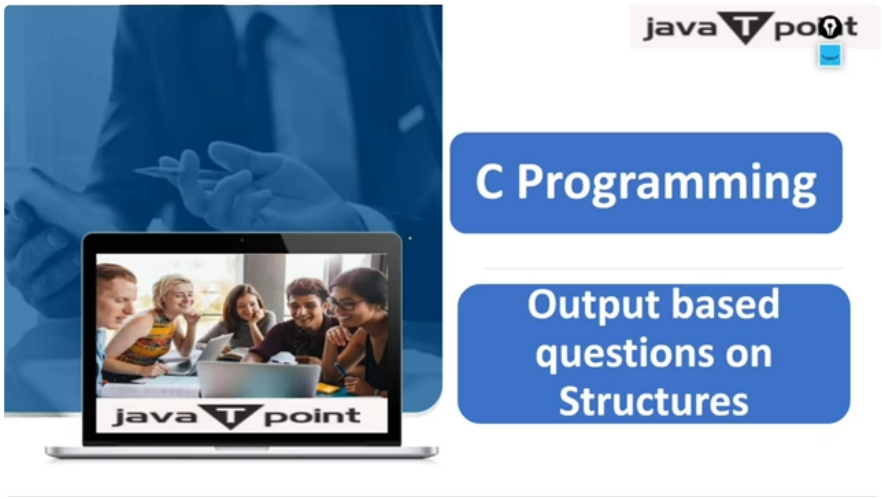
CFG stands for context-free grammar. It is is a formal grammar which is used to generate all possible patterns of strings in a given formal language. Context-free grammar G can be defined by four tuples as:

> G = (V, T, P, S)

Where,

G is the grammar, which consists of a set of the production rule. It is used to generate the string of a language.

T is the final set of a terminal symbol. It is denoted by lower case letters.



V is the final set of a non-terminal symbol. It is denoted by capital letters.

P is a set of production rules, which is used for replacing non-terminals symbols(on the left side of the production) in a string with other terminal or non-terminal symbols(on the right side of the production).

S is the start symbol which is used to derive the string. We can derive the string by repeatedly replacing a non-terminal by the right-hand side of the production until all non-terminal have been replaced by terminal symbols.

Example 1:

Construct the CFG for the language having any number of a's over the set ∑= {a}.

Solution:

As we know the regular expression for the above language is

> r.e. = a*

Production rule for the Regular expression is as follows:

> S → aS rule 1
> S → ε rule 2

Now if we want to derive a string "aaaaaa", we can start with start symbols.

```
 S
aS
aaS          rule 1
aaaS         rule 1
aaaaS        rule 1
aaaaaS       rule 1
aaaaaaS      rule 1
aaaaaaε      rule 2
aaaaaa
```

The r.e. = a* can generate a set of string {ε, a, aa, aaa,.....}. We can have a null string because S is a start symbol and rule 2 gives S → ε.

Example 2:

Construct a CFG for the regular expression (0+1)*

Solution:

The CFG can be given by,

Production rule (P):
S → 0S | 1S
S → ε

The rules are in the combination of 0's and 1's with the start symbol. Since (0+1)* indicates {ε, 0, 1, 01, 10, 00, 11, ...}. In this set, ε is a string, so in the rule, we can set the rule S → ε.

Example 3:

Construct a CFG for a language L = {wcwR | where w € (a, b)*}.

Solution:

The string that can be generated for a given language is {aacaa, bcb, abcba, bacab, abbcbba, ...}

The grammar could be:

> S → aSa rule 1
> S → bSb rule 2
> S → c rule 3

Now if we want to derive a string "abbcbba", we can start with start symbols.

> S → aSa
> S → abSba from rule 2
> S → abbSbba from rule 2
> S → abbcbba from rule 3

Thus any of this kind of string can be derived from the given production rules.

Example 4:

Construct a CFG for the language L = $a^n b^{2n}$ where n>=1.

Solution:

The string that can be generated for a given language is {abb, aabbbb, aaabbbbbb....}.

The grammar could be:

> S → aSbb | abb

Now if we want to derive a string "aabbbb", we can start with start symbols.

> S → aSbb
> S → aabbbb

← Prev Next →

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial Swagger	T-SQL tutorial Transact-SQL	Tumblr tutorial Tumblr
React tutorial ReactJS	Regex tutorial Regex	Reinforcement learning tutorial Reinforcement Learning	R Programming tutorial R Programming	RxJS tutorial RxJS
React Native tutorial	Python Design Patterns	Python Pillow tutorial	Python Turtle tutorial	Keras tutorial Keras

React Native

Python Design Patterns

Python Pillow

Python Turtle

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence Tutorial

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing tutorial

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

DAA tutorial

DAA

Operating System tutorial

Computer Network tutorial

Data Structures

Operating System

Computer Network


Compiler Design tutorial

Compiler Design


Computer Organization and Architecture

Computer Organization


Discrete Mathematics Tutorial

Discrete Mathematics


Ethical Hacking Tutorial

Ethical Hacking


Computer Graphics Tutorial

Computer Graphics


Software Engineering Tutorial

Software Engineering


html tutorial

Web Technology


Cyber Security tutorial

Cyber Security


Automata Tutorial

Automata


C Language tutorial

C Programming


C++ tutorial

C++


Java tutorial

Java


.Net Framework tutorial

.Net


Python tutorial

Python


List of Programs

Programs


Control Systems tutorial

Control System


Data Mining Tutorial

Data Mining


Data Warehouse Tutorial

Data Warehouse