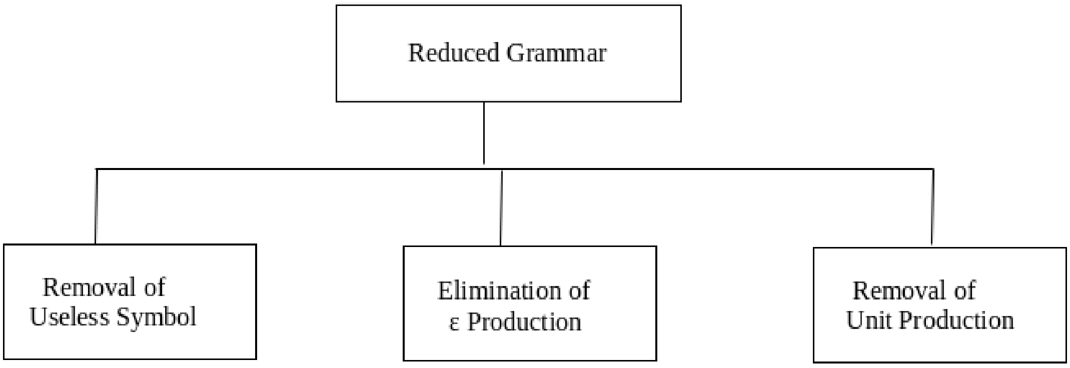
Simplification of CFG

As we have seen, various languages can efficiently be represented by a context-free grammar. All the grammar are not always optimized that means the grammar may consist of some extra symbols(non-terminal). Having extra symbols, unnecessary increase the length of grammar. Simplification of grammar means reduction of grammar by removing useless symbols. The properties of reduced grammar are given below:

1. Each variable (i.e. non-terminal) and each terminal of G appears in the derivation of some word in L.
2. There should not be any production as X → Y where X and Y are non-terminal.
3. If ε is not in the language L then there need not to be the production X → ε.

Let us study the reduction process in detail./p>



Removal of Useless Symbols

A symbol can be useless if it does not appear on the right-hand side of the production rule and does not take part in the derivation of any string. That symbol is known as a useless symbol. Similarly, a variable can be useless if it does not take part in the derivation of any string. That variable is known as a useless variable.

For Example:

> T → aaB | abA | aaT
>
> A → aA
>
> B → ab | b
>
> C → ad

In the above example, the variable 'C' will never occur in the derivation of any string, so the production C → ad is useless. So we will eliminate it, and the other productions are written in such a way that variable C can never reach from the starting variable 'T'.

Production A → aA is also useless because there is no way to terminate it. If it never terminates, then it can never produce a string. Hence this production can never take part in any derivation.

To remove this useless production A → aA, we will first find all the variables which will never lead to a terminal string such as variable 'A'. Then we will remove all the productions in which the variable 'B' occurs.

Elimination of ε Production

The productions of type S → ε are called ε productions. These type of productions can only be removed from those grammars that do not generate ε.

Step 1: First find out all nullable non-terminal variable which derives ε.

Step 2: For each production A → a, construct all production A → x, where x is obtained from a by removing one or more non-terminal from step 1.

Step 3: Now combine the result of step 2 with the original production and remove ε productions.

Example:

Remove the production from the following CFG by preserving the meaning of it.

> S → XYX
> X → 0X | ε
> Y → 1Y | ε

Solution:

Now, while removing ε production, we are deleting the rule X → ε and Y → ε. To preserve the meaning of CFG we are actually placing ε at the right-hand side whenever X and Y have appeared.

Let us take

> S → XYX

If the first X at right-hand side is ε. Then

> S → YX

Similarly if the last X in R.H.S. = ε. Then

> S → XY

If Y = ε then

> S → XX

If Y and X are ε then,

> S → X

If both X are replaced by ε

> S → Y

Now,

> S → XY | YX | XX | X | Y

Now let us consider

> X → 0X

If we place ε at right-hand side for X then,

> X → 0
> X → 0X | 0

Similarly Y → 1Y | 1

Collectively we can rewrite the CFG with removed ε production as

S → XY | YX | XX | X | Y
X → 0X | 0
Y → 1Y | 1

Removing Unit Productions

The unit productions are the productions in which one non-terminal gives another non-terminal. Use the following steps to remove unit production:

Step 1: To remove X → Y, add production X → a to the grammar rule whenever Y → a occurs in the grammar.

Step 2: Now delete X → Y from the grammar.

Step 3: Repeat step 1 and step 2 until all unit productions are removed.

For example:

S → 0A | 1B | C
A → 0S | 00
B → 1 | A
C → 01

Solution:

S → C is a unit production. But while removing S → C we have to consider what C gives. So, we can add a rule to S.

S → 0A | 1B | 01

Similarly, B → A is also a unit production so we can modify it as

B → 1 | 0S | 00

Thus finally we can write CFG without unit production as

S → 0A | 1B | 01

A → 0S | 00

B → 1 | 0S | 00

C → 01

← Prev

Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Splunk

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

React Native

Python Design Patterns

Python Design Patterns

Python Pillow tutorial

Python Pillow

Python Turtle tutorial

Python Turtle

Keras tutorial

Keras

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 Data Structures	DAA tutorial DAA	Operating System tutorial Operating System	Computer Network tutorial 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		