# Regular Grammar

## Subjects to be Learned

- Production and Grammar
- Regular Grammar
- Context-Free, Context-Sensitive and Phrase Structure Grammars

## Contents

We have learned three ways of characterising regular languages: regular expressions, finite automata and construction from simple languages using simple operations. There is yet another way of characterizing them, that is by something called grammar. A grammar is a set of rewrite rules which are used to generarte strings by successively rewriting symbols. For example consider the language represented by $a^+$, which is { a, aa, aaa, . . . } . One can generate the strings of this language by the following procedure: Let S be a symbol to start the process with. Rewrite S using one of the following two rules: S -> a , and S -> aS . These rules mean that S is rewritten as a or as aS. To generate the string aa for example, start with S and apply the second rule to replace S with the right hand side of the rule, i.e. aS, to obtain aS. Then apply the first rule to aS to rewrite S as a. That gives us aa. We write S => aS to express that aS is obtained from S by applying a single production. Thus the process of obtaining aa from S is written as S => aS => aa . If we are not interested in the intermediate steps, the fact that aa is obtained from S is written as $S \Rightarrow^* aa$ , In general if a string $\beta$ is obtained from a string $\alpha$ by applying productions of a grammar G, we write $\alpha \Rightarrow^*_G \beta$ and say that $\beta$ **is derived from** $\alpha$ . If there is no ambiguity about the grammar G that is referred to, then we simply write $\alpha \Rightarrow^* \beta$

Formally a **grammar** consists of a set of nonterminals (or variables) V, a set of terminals $\Sigma$ (the alphabet of the language), a start symbol S, which ia a nonterminal, and a set of rewrite rules (productions) P. A production has in general the form $\gamma \rightarrow \alpha$ , where $\gamma$ is a string of terminals and nonterminals with at least one nonterminal in it and $\alpha$ is a string of terminals and nonterminals. A **grammar is regular** if and only if $\gamma$ is a single nonterminal and $\alpha$ is a single terminal or a single terminal followed by a single nonterminal, that is a production is of the form X -> a or X -> aY, where X and Y are nonterminals and a is a terminal.

For example, $\Sigma$ = {a, b}, V = { S } and P = { S -> aS, S -> bS, S -> $\Lambda$ } is a regular grammar and it generates all the strings consisting of a's and b's including the empty string.

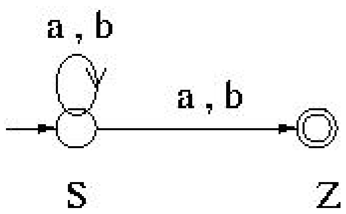
The following theorem holds for regular grammars.

**Theorem 3:** A language L is accepted by an FA i.e. regular, if L - { $\Lambda$ } can be generated by a regular grammar.

This can be proven by constructing an FA for the given grammar as follows: For each nonterminal create a state. S corresponds to the initial state. Add another state as the accepting state Z. Then for every production X -> aY, add the transition $\delta$( X, a ) = Y and for every production X -> a add the transition $\delta$( X, a ) = Z.

For example $\Sigma$ = {a, b}, V = { S } and P = { S -> aS, S -> bS, S -> a, S -> b } form a regular grammar which generates the language $( a + b )^+$. An NFA that recognizes this language can be obtained by creating two states S and Z, and adding transitions $\delta$( S, a ) = { S, Z } and $\delta$( S, b ) = { S, Z } , where S is the initial state and Z is the accepting state of the NFA.

The NFA thus obtained is shown below.



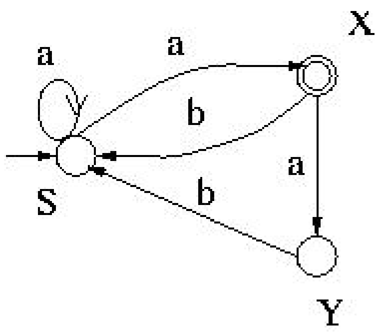

Thus L - { $\Lambda$ } is regular. If L contains $\Lambda$ as its member, then since { $\Lambda$ } is regular , L = ( L -{ $\Lambda$ } ) $\cup$ { $\Lambda$ } is also regular.

Conversely from any NFA < Q, $\Sigma$, $\delta$, $q_0$, A > a regular grammar < Q, $\Sigma$, P, $q_0$ > is obtained as follows: for any a in $\Sigma$ , and nonterminals X and Y, X -> aY is in P if and only if $\delta$(X, a) = Y , and for any a in $\Sigma$ and any nonterminal X, X -> a is in P if and only if $\delta$(X, a) = Y for some accepting state Y.

Thus the following converse of Theorem 3 is obtained.

**Theorem 4 :** If L is regular i.e. accepted by an NFA, then L - { $\Lambda$ } is generated by a regular grammar.

For example, a regular grammar corresponding to the NFA given below is < Q, { a, b }, P, S > , where Q = { S, X, Y } , P = { S -> aS, S -> aX, X -> bS, X -> aY, Y -> bS, S -> a } .




NFA

In addition to regular languages there are three other types of languages in **[Chomsky hierarchy]**: context-free languages, context-sensitive languages and phrase structure languages. They are characterized by context-free grammars, context-sensitive grammars and phrase structure grammars, respectively.
These grammars are distinguished by the kind of productions they have but they also form a hierarchy, that is the set of regular languages is a subset of the set of context-free languages which is in turn a subset of the set of context-sensitive languages and the set of context-sensitive languages is a subset of the set of phrase structure languages.
A grammar is a **context-free grammar** if and only if its production is of the form X -> $\alpha$ , where $\alpha$ is a string of terminals and nonterminals, possibly the empty string.
For example P = { S -> aSb, S -> ab } with $\Sigma$ = { a, b } and V = { S } is a contex-free grammar and it generates the language { $a^n b^n$ | n is a positive integer } . As we shall see later this is an example of context-

free language which is not regular.

A grammar is a **context-sensitive grammar** if and only if its production is of the form $\alpha_1 X \alpha_2$ -> $\alpha_1 \beta \alpha_2$, where X is a nonterminal and $\alpha_1$ , $\alpha_2$ and $\beta$ are strings of terminals and nonterminals, possibly empty except $\beta$ .

Thus the nonterminal X can be rewritten as $\beta$ only in the context of $\alpha_1 X \alpha_2$ .

For example P = { S -> XYZ$S_1$, S -> XYZ, $S_1$ -> XYZ$S_1$, $S_1$ -> XYZ, YX -> XY, ZX -> XZ, ZY -> YZ, X -> a, aX -> aa, aY -> ab, BY -> bb, bZ -> bc, cZ -> cc } with $\Sigma$ = { a, b, c } and V = { X, Y, Z, S, $S_1$ } is a context-sensitive grammar and it generates the language { $a^n b^n c^n$ | n is a positive integer } . It is an example of context-sensitive language which is not context-free.

Context-sensitive grammars are also characterized by productions whose left hand side is not longer than the right hand side, that is, for every production $\alpha$ -> $\beta$ , $|\alpha| \leq |\beta|$ .

For a **phrase structure grammar**, there is no restriction on the form of production, that is a production of a phrase structure grammar can take the form $\alpha$ -> $\beta$, where $\alpha$ and $\beta$ can be any string, but $\alpha$ must contain at least one non-terminal.

## Test Your Understanding of Regular Grammar

Indicate which of the following statements are correct and which are not.
Click True or Fals , then Submit.
There are two sets of questions.

**Next -- Minimization of DFA**

**Back to Study Schedule**

**Back to Table of Contents**