Context-Free Grammars

Grammars and Regular Grammars

For natural languages, each has its own grammar. A Chinese sentence follows Chinese language grammar. An English sentence follows its own grammar.

A sentence may follow a correct grammar without proper meaning. For instance, the following sentence is grammatically correct but nonsense:

A desk eats a lion.

For programming languages, Pascal programs must follow the Pascal programming grammar. A C program has to obey the C programming grammar.

Even numbers must also follow their rules to be written down. The rules can be as follows.

An integer can be either nonnegative number or negative number.

A number consists of one digit or many digits.

For a one digit number, it could be 0, 1, …, 9.

For a many digits number, it starts with a digit of 1, …, 9, and follows with many digits of 0, 1, …, 9.

Using the following graph structure could be easier to understand the above rules.

The following structure consists of rewriting rule, sequence, selection and repetition properties.

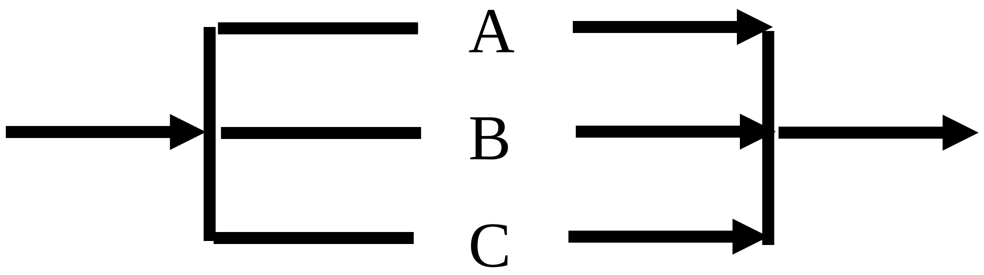
(1) Rewriting rule : Left part A is replaced by the right part B.

$$A \ ::= \ B$$

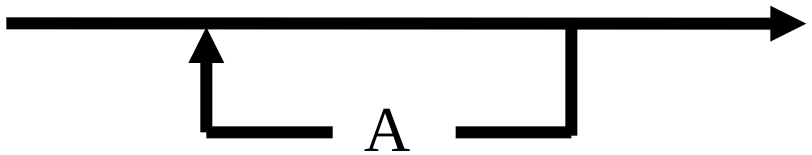
(2) Sequence : That B follows A is shown as

$$A \longrightarrow B$$

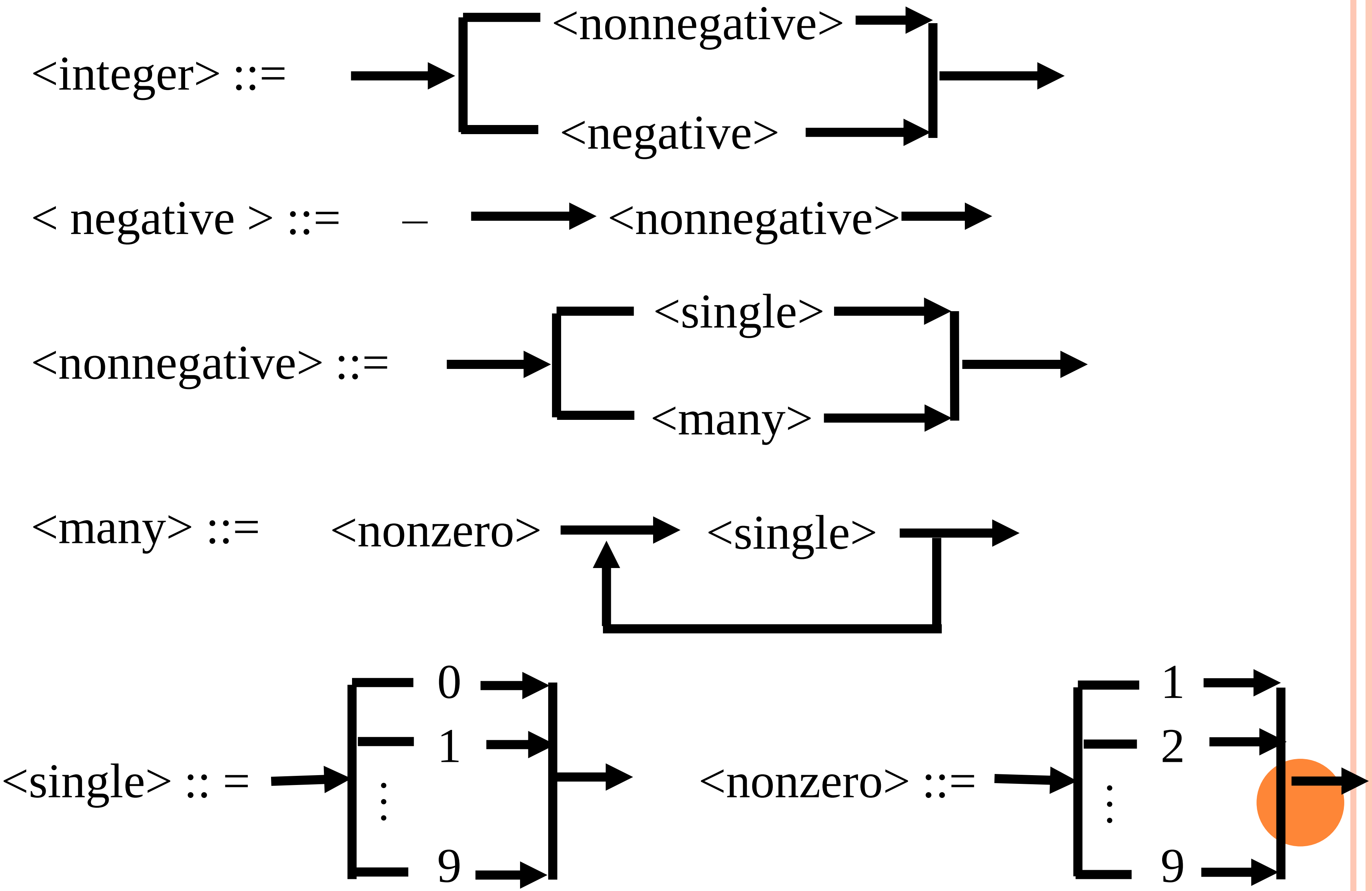
(3) Selection : That select one item from A, B and C is shown as



(4) Repetition : Repeat to select item A for 0, 1, or many times.



The structure of integers can be written as follows.



The structure of integers can also be written in Backus-Naur form shown as follows. The notation → stands for rewriting and the symbol | stands for selection. Repetition is replaced by recursion.

<integer> → <nonnegative> | <negative>

< negative > → – <nonnegative>

<nonnegative> → <single> | <nonzero> <number>

<number> → <single> | <single> <number>

<nonzero> → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<single> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The symbols used in Backus-Naur form are variables and terminals.

The set of variables is {<integer>, <nonnegative>, <negative>, <nonzero>, <single>, <number>}

The set of terminals is {–, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

To generate a string of terminals 123, we start from the variable <integer> and follow the above rules as follows.

<integer> $\Rightarrow$ <nonnegative>

$\Rightarrow$ <nonzero> <number>

$\Rightarrow$ 1 <number>

$\Rightarrow$ 1 <single> <number>

$\Rightarrow$ 1 2 <number>

$\Rightarrow$ 1 2 <single>

$\Rightarrow$ 1 2 3

The notation $\Rightarrow$ stands for derivation.

The notation $\overset{*}{\Rightarrow}$ stands for many derivations.

We have that

<integer> $\overset{*}{\Rightarrow}$ 1 2 3

Regular Grammars

A regular language can be accepted by a finite state automaton and denoted by a regular expression.

In this section, we shall show that a regular language can be generated by a regular grammar.

Definition 1: A **regular grammar** $G = (V, T, P, S)$ is defined as follows.

(1) V is a finite set of variable. S is the start symbol in V.

(2) T is a finite set of terminals, and $T \cap V = \varnothing$.

(3) P is a finite set of productions or rewriting rules. Each production is of the form:

$A \rightarrow aB$, where $A, B \in V$ and $a \in T$, or

$A \rightarrow a$, where $A \in V$ and $a \in T$.

Definition 2: The set generated by a regular grammar G = (V, T, P, S) is $\{\omega \in T^* \mid S \Rightarrow^* \omega\}$ denoted by L(G).

Example 2: Find L(G) for G = (V, T, P, S), where V = {S, A, B}, T = {0, 1} and P contains the following productions:

$$S \to 0A \mid 1B \mid 1$$

$$A \to 0S \mid 1B \mid 1$$

$$B \to 0B \mid 1A \mid 0$$

Solution:

The set generated by a regular grammar G is

$$\{\omega \in \{0, 1\}^* \mid \omega \text{ has odd number of 1's}\}.$$

See also example 3 and example 2 of section 2.5 for the result.

Theorem 1: Let L be a regular language. Then there is a regular grammar G such that L(G) = L.

Proof:

L is regular, there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L.

Construct a regular grammar $G = (V, T, P, S)$ by the following.

Assume that $Q \cap \Sigma = \emptyset$. Let $V = Q$, $S = q_0$, $T = \Sigma$.

If $\delta(q, a) = p$ and $p \notin F$, then P contains a production as

$$q \to a\, p, \text{ where } a \in T,\ p, q \in V.$$

If $\delta(q, a) = p$ and $p \in F$, then P contains a production as

$$q \to a\, p \mid a$$

It is easy to show that $\delta^*(q, \omega) = p \in F$, iff $S \Rightarrow^* \omega$.

Theorem 2: Let G be a regular grammar G. Then L(G) is regular.

Proof:

Let $G = (V, T, P, S)$ be a regular grammar.

Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows.

Assume that $q_f \notin V$. Let $Q = V \cup \{q_f\}$, $q_0 = S$, $\Sigma = T$ and $F=\{q_f\}$.

If $q \rightarrow a\, p$ is a production in P, then

$$\delta(q, a) = p, \text{ where } a \in \Sigma,\ p, q \in Q.$$

If $q \rightarrow a$ is a production in P, then

$$\delta(q, a) = q_f$$

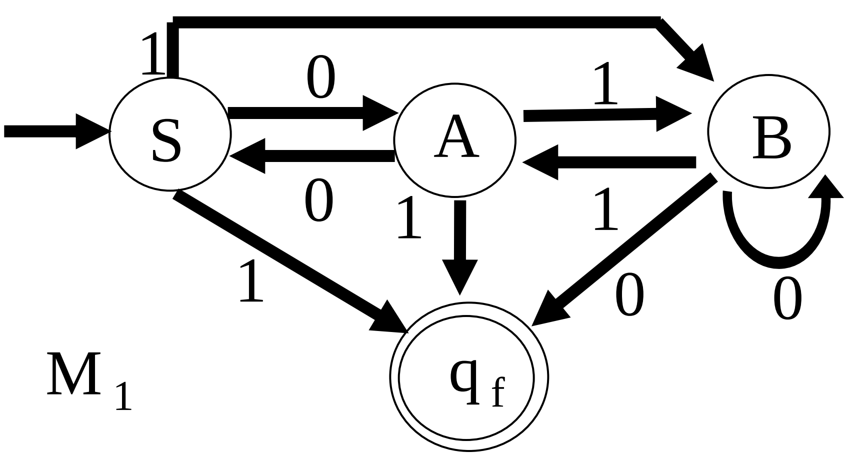
It is easy to show that $\delta^*(q, \omega) = q_f \in F$, iff $S \Rightarrow^* \omega$.

Example 3: Find a DFA M such that L(M) = L(G) for a regular grammar G = (V, T, P, S), where V = {S, A, B}, T = {0, 1} and P contains the following productions:

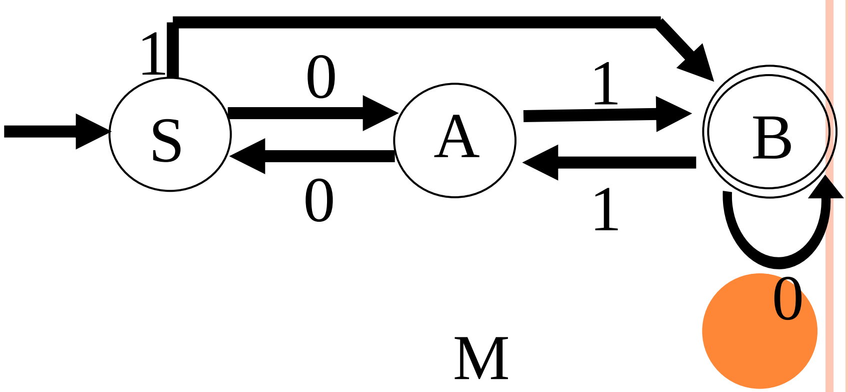
$S \to 0A \mid 1B \mid 1$ $\qquad$ $A \to 0S \mid 1B \mid 1$ $\qquad$ $B \to 0B \mid 1A \mid 0$

Solution:

By theorem 2, construct an NFA M_1 to accept L(G) and modify to a DFA M as follows.



M_1



M

By the previous theorems and theorems in chapter 2, we have the following theorem.

Theorem 3: The class of regular languages, the class of DFA's, the class of regular expressions and the class of regular grammars are equivalent.

Note :

(1) A DFA can recognize a regular set.

(2) A regular expression can represent a regular set.

(3) A regular grammar can generate a regular set.