So we can say that ∈ free type-0 grammar is said to be length increasing grammar.

10.5. LINEAR BOUNDED AUTOMATA

A Linear Bounded Automata (LBA) is a non-deterministic turing machine satisfying the following two conditions.

(*a*) Its input alphabet includes two special symbols $\$_1$ and $\$_2$, the left and right end marker, respectively.

(*b*) The LBA has no more left from $\$_1$ and no right from $\$_2$, nor may it print another symbol over $\$_1$ or $\$_2$.

The linear bound automata is simply a turing machine which instead of having potentially infinite tape on which to compute, is restricted to the portion of the tape containing the input plus the two tape squares holding the end markers.

LBA will be denoted as

$T_m = (Q, \Sigma, \Gamma, \delta, q_0, h, \$_1, \$_2)$ where Q, Σ, δ, S are as for non-deterministic turing machine; $\$_1$ and $\$_2$ are symbols in Γ, the left and right end markers.

The $L(T_m)$ the languages accepted by turing machine, is

$\{W / W$ is in Σ^* and $(q_0, \$_1\ W\ \$_2) \overset{*}{\underset{T_m}{\vdash}} (h, \$_1\ y\ \$_2)\}$

Here y is for yes and h shows halting of LBA after accepting the string W.

11.5. RECURSION

Definition We know that recursion is widely used term in computer science. "A function which calls itself directly or indirectly and terminates after finite number of steps is known as recursive function."

In recursive functions, terminating point is also known as base point. A recursive function will be good if it has terminating point and it call to itself should be nearer to its terminating point. The terminating point is also known as base point.

Let g be a function of $n + 1$ arguments, g' be a function of n arguments, and h be a function of m arguments where $m \leq n + 2$. We define the function g recursively using the g' and h function as follows :

$$\text{If } (x_1, x_2, \ldots, x_n, 0) = g'(y_1, y_2, \ldots, y_n) \text{ then } g(x_1, x_2, \ldots, x_n, x_{n+1} + 1)$$
$$= h(x_1, x_2, \ldots, x_n, x_{n+1}, g(x_1, x_2, \ldots, x_n, x_{n+1})),$$

where g' and h are defined functions.

Example 11.1. Define m ! recursively.

Solution. Let $g(m) = m ! = m * (m - 1) * (m - 2) \ldots * 1$

We know that $0 ! = 1 ! = 1,$

So $g(0) = 1$(it is a terminating point)

$$h(x, y) = x * y$$

$$g(n + 1) = h(n + 1, g(n)) \quad \text{(Indirect call)}$$

$= 7$

11.6. PRIMITIVE RECURSIVE FUNCTION

Definition A function is primitive if it follows the condition :

(*i*) It is an initial function, or

(*ii*) It is obtained from recursion or composition of initial functions.

Most popular total functions are primitive recursive functions; however some total functions are not. For example, the ackerman function is total function but not primitive recursive.

Example 11.3. Show that addition function of two positive integers is primitive recursive.

Solution. In Example 11.2, we have defined the addition function f for m and n as $f(m, 0) = m, f(m, n + 1) = f(m, n) + 1$. We define f using the initial functions as follows :

$$f(m, 0) = m = P_1'(m),$$

and

$$f(m, n + 1) = f(m, n) + 1 = S(f(m, n))$$

For example, let us consider $f(7, 2)$

$$\begin{aligned} f(7, 2) &= S(f(7, 1)) \\ &= f(7, 1) + 1 \\ &= S(f(7, 0)) + 1 \\ &= f(7, 0) + 1 + 1 \\ &= 7 + 1 + 1 \\ &= 9 \end{aligned}$$

Hence, f is a primitive recursive function.

Theorem 11.1. Every primitive recursive function is turing computable.

Proof. We know that primitive functions are defined using initial functions with sometimes applying composition and recursion. So if it is possible to design

11.8. TOTAL RECURSIVE FUNCTION

Definition A total function is a subclass of partial function. A function is called total function if it is defined for all its arguments. Let $f(a_1, a_2, ..., a_n)$ be a function and defined on function $g(b_1, b_2, ..., b_m)$, then f is total function if every element of f is assigned to some unique element of function g.

Example 11.4. Addition of two positive integer is a total function.

Solution. Let $f(m, n) = m + n$ is the addition function from $N \times N$ to N of two positive integers m and n. This function can be defined recursively as

$$f(m, 0) = 1, \; f(m, n + 1) = f(m, n) + 1.$$

Since, f is defined for all values of $m, n \in N$. So f is a total function.

Example 11.5. Multiplication of two positive integers is a total function.

Solution. Let $g(m, n)$ is multiplication function from $N \times N$ to N of two positive integers m and n, defined recursively as : $g(m, 0) = 0$, $g(m, n + 1) = m + g(m, n)$. Since, g is defined for all $m, n \in N$. So g is a total recursive function.