Definition: *A grammar (V_N,Σ, P, S) is called an unrestricted grammar if all its productions are in the form $LS \rightarrow RS$, where $LS \in (V_N \cup \Sigma)+$and $RS \in (VN \cup \Sigma)^*$.*

In an unrestricted grammar, any combination of terminal and non-terminal can appear at both ends of the production rule. The only restriction is that the null string (λ) cannot appear at the left hand side of the production rule. (Notice for LS, it is + not *.) In the Chomsky hierarchy, type 0 or unrestricted grammar is the superset of all grammars, and thus it accepts more languages than type 1 or other grammars.

Already it is mentioned that unrestricted grammar is accepted by the TM, and the language accepted by a TM is called recursively enumerable language. In the following section, we shall discuss the relation between unrestricted grammar and recursively enumerable language.

Theorem 10.1: Any language set generated by an unrestricted grammar is recursively enumerable.

Proof: Let G = (V_N,Σ, P, S) be an unrestricted grammar. A grammar defi nes a set of procedures for enumerating all strings $\in L(G)$.

□ List all w ∈ L(G) such that w is derived from the production rules P in one step. As S is the start symbol, we can write that w is the set of strings derived from S in one step. Symbolically, it can be written as $S \rightarrow w$.

□ List all w ∈ L(G) such that w is derived from the production rules P in two steps. We can write it as $S \rightarrow x \rightarrow w$.

By the process, the derivation progresses.

If S be the set of all strings, then an enumeration procedure for S is a TM that generates all strings of S one by one in a fi nite amount of time. If, for a set, there exists an enumeration procedure, the set is countable. The set of strings generated by an unrestricted language is countable. Thus, the derivation procedures can be enumerated on a TM.

Hence it is proved that the language generated by an unrestricted grammar is recursively enumerable.

10.2.1 Turing Machine to Unrestricted Grammar

A TM accepts type 0 languages. Type 0 language is generated by type 0 grammar. Type 0 grammar can be constructed directly from the TM's transitional functions. The rules are discussed in the following.

Let the string to be traversed by a TM M be S which is enclosed between two end markers Ψ S $. Note that the IDs are enclosed within brackets. The acceptance of the string S by M means the transformation of ID from [q_0Ψ S $] to [$q_f B$].$Here$,$q_0$ is the initial state and qf is the fi nal or halt state. The length of the ID may change if the read–write head of the TM reaches the left hand side or right hand side brackets. Thus, the productions rules of the grammar equivalent to the transition of ID are divided into two steps: (i) no change in length and (ii) change in length. Assume that the transition table is given as follows.

Input state	a_1	⋯	a_j
q_1			
⋮			
q_i			

□**No change in length:**

-Right move: If there is a transitional function

$$\delta(q_i, a_j) \rightarrow (q_k, a_l, R)$$

of the TM, then the equivalent production rule of the grammar is

$$q_i a_j \rightarrow a_l q_k$$

-Left move: If there is a transitional function

$$\delta(q_i, a_j) \rightarrow (q_k, a_l, L)$$

of the TM, then the equivalent production rule of the grammar is in the form

$$a_p q_i a_j q_k a_p a_l \text{ for all} a_p \in \Gamma$$

(If there are n allowable tape symbols, then for a left movement production n transitional functions are added to the production rule.)

□**Change in length:**

-Left bracket '[' at the left end: If there is a production

$$\delta(q_i, a_j) \rightarrow (q_k, a_l, L)$$

TM going to traverse the left boundary '[‘): then the production

$$[q_i a_j \rightarrow q_k B a_l$$

is added to the production rule. (Here B represents the blank.)

If B appears next to the left bracket, it can be deleted by the production

$$[B \rightarrow [$$

-Right bracket ']' at the right end: If B appears just before ']', then it can be deleted by the production

$$a_p B] \rightarrow a_p] \text{ for all} a_p \in \Gamma$$

If TM going to traverse to the right of ']', then the length is increased due to the insertion of B. The productions are

$$q_i] \rightarrow q_i B] \text{ for all} q_i \in Q$$

□ The string is confi ned by two end markers $\Psi and \$$. To introduce the end markers, the following productions are added.

$$[q_i \Psi \rightarrow [q_i \text{ for all} q_i \in Q$$
$$a_p \$ \rightarrow a_p \text{ for all} a_p() \in \Gamma$$

To remove brackets '[' and ']' from [qfB], the following production is added,

$$[q_f B] \rightarrow S$$

Here, S is the start symbol of the grammar.

□ Now, reverse the direction of the arrow in the transitional functions. The generated production rules are the production rules of the grammar. The grammar obtained in this process is called generative grammar.

Example 10.1

Consider a TM with the following transitional functions. (This is the TM for 1's complement.) Convert this to an equivalent type 0 grammar.

Input state	0	1	B
Q_0	$Q_0, 1, R$	$Q_0, 0, R$	Q_0, B, R
Q_f			

Solution:

i) For the transitional function $\delta(Q_0, 0) \rightarrow (Q_0, 1, R)$, the production rule is

$$Q_0 0 \rightarrow 1 Q_0$$

For transitional function $\delta(Q_0, 1) \rightarrow (-Q0, 0, R)$, the production rule is

$$Q_0 1 \rightarrow 0 Q_0$$

For transitional function $\delta(Q_0, B) \rightarrow (Q_f, B, R)$, the production rule is

$$Q_0 B \rightarrow 0 Q_f$$

ii) The production rule corresponding to the left end is

$$[B \rightarrow [$$

The production rules corresponding to the right end are

$$0B] \rightarrow 0]1B] \rightarrow 1]BB] \rightarrow B][\text{for tape symbols}]$$
$$Q_0] \rightarrow Q_0 B]Q1] \rightarrow Q_1 B]$$

iii) The production rules for introducing end markers are

$$[Q_0 \Psi \rightarrow [Q_0[Q_f \Psi \rightarrow [Q_f$$
$$0\$ \rightarrow 01\$ \rightarrow 1 \text{ [B is excluded]}$$
$$[Q_f B] \rightarrow S$$

By reversing the direction of the arrow, the productions become

$$S \rightarrow [Q_f B], 1 \rightarrow 1\$, 0 \rightarrow 0\$$$
$$[Q_0 \rightarrow [Q_0, [Q_f \rightarrow [Q_f, Q_1 B] \rightarrow Q_1]$$
$$Q_0 B] \rightarrow Q_0], B] \rightarrow BB], 1] \rightarrow 1B]$$
$$0] \rightarrow 0B], [\rightarrow [B, 0Qf \rightarrow Q_0 B$$
$$0Q_0 \rightarrow Q_0 11 Q_0 \rightarrow Q_0 0$$

Example 10.2

Consider a TM with the following transitional functions. (This is the TM which accepts an even string of 'a'.) Convert this to an equivalent type 0 grammar. Check it for the string 'aa'.

Input state	A	B
Q_0	Q_1, B, R	
Q_1	Q_1, B, R	

where Q0 is the initial and fi nal state. **Solution:**

i) For transitional function (Q0, a) → (Q1, B, R), the production rule is

$$[B \to [$$
$$Q_0 a \to BQ_1$$

For the second transitional function (Q1, a) → (Q0, B, R), the production rule is

$$Q_1 a \to BQ_0$$

ii) The production rule corresponding to the left end is

$$[B \to [$$

The production rules corresponding to the right end are

$$aB] \to a]BB] \to B][\text{for tape symbols}]$$
$$Q_0] \to Q_0 B]Q_1] \to Q_1 B]$$

iii) The production rules for introducing end markers are

$$[Q_0\Psi[Q0[Q_1\Psi \to [Q_1$$
$$a\$a[B\text{is excluded}]$$
$$[Q_0 B] \to S$$

By reversing the direction of the arrow, the productions become

$$BQ_1 \to Q_0 aBQ_0 \to Q_1 a$$
$$[\to [Ba] \to aB]$$
$$B] \to BB]Q + 0B] \to Q_0]$$
$$Q_1 B] \to Q_1][Q_0 \to [Q_0\Psi$$
$$[Q_1 \to [Q_1\Psi a \to a\$$
$$S \to [Q_0 B]$$

Checking: The string is 'aa'. It means from S we have to produce $[Q_0\Psi aa\$]$.

$$S \to [Q_0 B]$$
$$\to [Q_0](AsQ_0 B] \to Q_0]) \to [BQ_0](As[\to [B)$$