Lecture seven

Topics that must be covered in this lecture:

- ***Grammar***
- ***Chomsky hierarchy of languages.***

Grammar: A grammar is a set of rules which are used to construct a language (combine words to generate sentences).

Each Type of grammar G is defined formally as a mathematical system consisting of a quadruple **(V,∑ , P, S) ,** where :

V: is an alphabet, whose elements are called **nonterminal** symbols.

∑: is an alphabet disjoint from V, whose elements are called terminal symbols.

P: is a relation of finite cardinality on (V∪∑)*, whose elements are called production rules. Moreover, each production rule (α,β) in P, denoted **α→ β** , must have at least one nonterminal symbol in α . In each such production rule, **α** is said to be the left-hand side of the production rule, and **β** is said to be the right-hand side of the production rule.

S is a symbol in V called the start symbol.

Example:

1- G1= (V, ∑, P, S) is a grammar if V = {S},∑={a, b}, and P ={S→ aSb, S→λ }. By definition, the grammar has a single nonterminal symbol S , two terminal symbols a and b, and two production rules S →aSb and S→λ . Both production rules have a left-hand side that consists only of the nonterminal symbol S. The right-hand side of the first production rule is aSb, and the right-hand side of the second production rule is λ .

2- If V2 = {S}, ∑2 = {a, b}, and P2 = {S →aSb, S →λ, ab →S} then (V2, ∑2, P2, S) is not a grammar, because ab →S does not satisfy the requirement that each production rule must contain at least one nonterminal symbol on the left-hand side.

3- let G=({S,A,B},{a,b},{S→AB,A→a, B→b}, S), write elements of this grammar.

V={S,A,B}, ∑={a,b}, P={S→AB,A→a, B→b} .

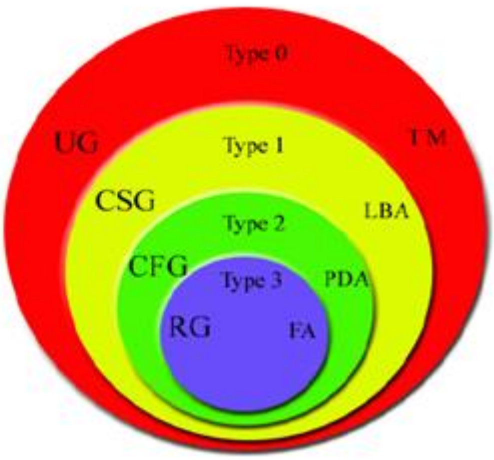
The Chomsky Hierarchy

Noam Chomsky introduced the Chomsky hierarchy which classify grammar and language, and effective for writing computer languages.

The Chomsky hierarchy gave four types of languages and their associated grammars and machines. The four classes are summarized in the table:

Type	Language	Grammar	Machine	Example
Type 3	Regular language	Regular grammar **RG**	Finite Automata **FA**	a*b*
Type 2	Context free language	Context-free grammar **CFG**	Pushdown automaton **PDA**	$a^n b^n$
Type 1	Context sensitive language	Context sensitive grammar **CSG**	Linear bounded automaton **LBA**	$a^n b^n c^n$
Type 0	Recursively enumerable language	Unrestricted grammar **UG**	Turing machine **TM**	any computable function

The four classes are summarized in the figure below:



Type 3:

A regular grammar is any right-linear or left-linear grammar, and regular grammars generate the regular languages. These languages are exactly all languages that can be decided by a finite state automaton. Additionally, this family of formal languages can be obtained by regular expressions.

The rule S → λ is also here allowed if S does not appear on the right side of any rule.

Linear grammar: in any grammar if there exit exactly one variable on the LHS and one variable on the RHS, is known as linear grammars following are example of the linear grammar:

Right Linear Grammar

A grammar is said to be Right Linear if all productions are of The form:

A→xB

A→x

B→y

Where A,B ∈ V

Example:

S → abS/b

Left Linear Grammar

A grammar is said to be Left Linear if all productions are of The form:

A→Bx

A→x

B→y

Where A,B ∈ V

Example:

S → Sbb/b

Type 2:

Context-free grammars generate the context-free languages. These are defined by rules of the form A → **α** with A a nonterminal and γ a string of terminals and non-terminals. These languages are exactly all languages that can be recognized by a non-deterministic pushdown automaton. Context free languages are the theoretical basis for the syntax of most programming languages.

A → **α** **α ∈ (V U t)***

Eg.

S→ SS/aA/bA

S→ λ

S→abB

Type 1:

Context-sensitive grammars generate the context-sensitive languages. These grammars have rules of the form **α→ β, where |α|≤ | | and α , β ∈ (V U t)$^+$**

The rule S → ε is allowed if S does not appear on the right side of any rule or S is a start symbol. The languages described by these grammars are exactly all languages that can be recognized by a non-deterministic Turing machine whose tape is bounded by a constant times the length of the input.

Eg.

S→SS

aA→bAa

BB→aB

S→ λ

Left side <= right side

Type 0:

unrestricted grammars include all formal grammars. They generate exactly all languages that can be recognized by a Turing machine. The language that is recognized by a Turing machine is defined as all the strings on which it halts. These languages are also known as the recursively enumerable languages.

$$\boldsymbol{\alpha \to \beta} \qquad \boldsymbol{\alpha} \in (V \cup t), \boldsymbol{\beta} \in \mathbf{(V \cup t)^*}$$

Eg.

S→SS
S→aAb
aA→Aa
BB→ a
aA→bAa
Ba→bAb

Example: Identify types of grammars:

1- S→aS|a (right linear) (3,2,1,0)
2- S→ AB
A→ a
B→ b
Grammar is Type (2,1,0)
3- S→aSb|bSb|a|b
Grammar is type 2
4- S→ $aS|bS|\epsilon$
Right linear : regular grammar (3,2,1,0)