Lecture Five

Context Free Grammar (CFG)

A context-free grammar, called a CFG, is a collection of three things:

1- An alphabet ∑ of letters called terminals from which we are going to make strings that will be the words of a language.
2- A set of symbols called nonterminals, one of which is the symbol S, standing for "start"
3- A finite set of productions of the form:

one nonterminal → finite string of terminals **and/or** nonterminals

Example: Let G(L) = ({S}, {a}, P, S), where P is:

S → aS
S →Λ

If we apply production (S → aS) four times and then apply production (S →Λ) we generate the following string: a^4

S → aS
→ aaS
→ aaaS
→ aaaaS
→ aaaaΛ

The language generated by this CFG is exactly a*.

Example: Let G(L) = ({S}, {a}, P, S), where P is:

S → SS | a | Λ

In this Language we can get the following derivation:

S → SS
→ SSS
→ SaS
→ ΛaS
→ ΛaSS
→ ΛaaS
→ ΛaaΛ = aa

The language generated by this set of productions is also just the language a*.

Example: Let the terminals be **a** and **b**, let the only nonterminal be **S**, and let the productions be:

$$S \longrightarrow aS \mid bS \mid a \mid b$$

We can produce the word baab as follows:

$$\begin{aligned} S &\longrightarrow bS \\ &\longrightarrow baS \\ &\longrightarrow baaS \\ &\longrightarrow baab \end{aligned}$$

The derivation above applied in order productions 2, 1, 1, 4. The language generated by this CFG is the set of all possible strings of the letters a and b except for the null string, which we cannot generate.

For example, to generate babb we apply in order productions 2, 1, 2, 4, as below:

$$\begin{aligned} S &\longrightarrow bS \\ &\longrightarrow baS \\ &\longrightarrow babS \\ &\longrightarrow babb \end{aligned}$$

Example: Let the terminals be a and b. Let the nonterminals be S, X, and Y. Let the productions be:

$$\begin{aligned} S &\longrightarrow X \\ S &\longrightarrow Y \\ X &\longrightarrow \wedge \\ Y &\longrightarrow aY \mid bY \mid a \mid b \end{aligned}$$

We can produce the word aabb as follows:

$$\begin{aligned} S &\longrightarrow Y \\ &\longrightarrow aY \\ &\longrightarrow aaY \\ &\longrightarrow aabY \\ &\longrightarrow aabb \end{aligned}$$

The language generated by this CFG is exactly a*b*.

Example: Let the terminals be a and b, let the nonterminals be S and X, and let the productions be:

$$\begin{aligned} S &\longrightarrow XaaX \\ X &\longrightarrow aX \mid bX \mid \wedge \end{aligned}$$

We already know from the previous example that the last three productions will allow us to generate any word we want from the nonterminal X. If the nonterminal

X appears in any working string, we can apply productions to turn it into any word we want. Therefore, the words generated from S have the form:

anything aa anything

or

(a + b)*aa(a + b)*

Convert CFG to ∧- free CFG

Theorem

If L is a context-free language generated by a CFG that includes ∧-productions, then there is a different context-free grammar that has no ∧-productions that generates either the whole language L (if L does not include the word ∧) or else generates the language of all the words in L that are not ∧.

Definition:

In a given CFG, we call a nonterminal N **nullable** if:

- There is a production: N→ ∧
  Or
- There is a derivation that start at N and leads to Λ: N→…→ ∧

Replacement Rule

1. Delete all ∧-productions.
2. Add the following productions: For every production

X → old string

add enough new productions of the form X → … that the right side will account for any modification of the old string that can be formed by deleting all possible subsets of nullable nonterminals, except that we do not allow X → ∧ to be formed even if all the characters in this old right-side string are nullable.

Example: Consider the CFG:

S → a | Xb | aYa
X → Y | ∧
Y → b | X

X and Y are nullable.
The new CFG is:

S → a | Xb | aYa | b | aa
X → Y
Y → b | X

Example: Consider the CFG:

S → Xa
X → aX | bX | ∧

X is the only nullable nonterminal.
The new CFG is:

S → Xa | a
X → aX | bX | a | b

Example: Consider this inefficient CFG for the language defined by:

(a + b)*bb(a + b)*

S → ZW
Z → Zb | AB
W→ bW | Z
A → aA | bA | ∧
B → Ba | Bb | ∧

A, B, W and Z are nullable.
The new CFG is:

S → ZW
Z → Zb | AB | b | A | B
W → bW | Z | b
A → aA | bA | a | b
B → Ba | Bb | a | b

Homework: Convert the following CFG to ∧- free CFG:

S → X | YaY | aSb | b
X → YY | b | ∧
Y → aY | aaX

Chomsky Normal Form (CNF)

Theorem

For any CFL the non- Λ words of L can be generated by a grammar in which all productions are of one of two forms:

Nonterminal → string of exactly two Nonterminals

or

Nonterminal → One Terminal

It is said to be in **Chomsky Normal Form (CNF)**.

Conversion steps:

1- Deleting ∧- productions.
2- Convert right side to nonterminals.
3- Convert to CNF.

Example: Convert the following CFG into CNF:

S→ aSa | bSb | Xa | a | b | aa | bb
X → ∧ | b

1- Deleting ∧- productions.

S→ aSa | bSb | Xa | a | b | aa | bb
X → b

2- Convert right side to nonterminals.

S → ASA | BSB | XA | AA | BB | a | b
X → b
A → a
B → b

3- Convert to CNF.

S → AR_1 | BR_2 | XA | AA | BB | a | b
R_1 → SA
R_2 → SB
X → b
A → a
B → b

Example: Convert the following CFG into CNF:

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

1- Convert right side to nonterminals.

$S \rightarrow YA \mid XB$

$A \rightarrow YAA \mid XS \mid a$

$B \rightarrow XBB \mid YS \mid b$

$X \rightarrow a$

$Y \rightarrow b$

2- Convert to CNF.

$S \rightarrow YA \mid XB$

$A \rightarrow YR_1 \mid XS \mid a$

$B \rightarrow XR_2 \mid YS \mid b$

$X \rightarrow a$

$Y \rightarrow b$

$R_1 \rightarrow AA$

$R_2 \rightarrow BB$

Example: Convert the following CFG into CNF:

$S \rightarrow AAAAS$

$S \rightarrow AAAA$

$A \rightarrow a$

Convert to CNF:

$S \rightarrow AR_1$ (where $R_1 = AAAS$)

$R_1 \rightarrow AR_2$ (where $R_2 = AAS$)

$R_2 \rightarrow AR_3$ (where $R_3 = AS$)

$R_3 \rightarrow AS$

$S \rightarrow AR_4$ (where $R_4 = AAA$)

$R_4 \rightarrow AR_5$ (where $R_5 = AA$)

$R_5 \rightarrow AA$

$A \rightarrow a$

Homework: Convert the following CFG's to CNF.

1- S →SS | a

2- S→ aSa | SSa | a

3- S → aXX
X → aS | bS | a

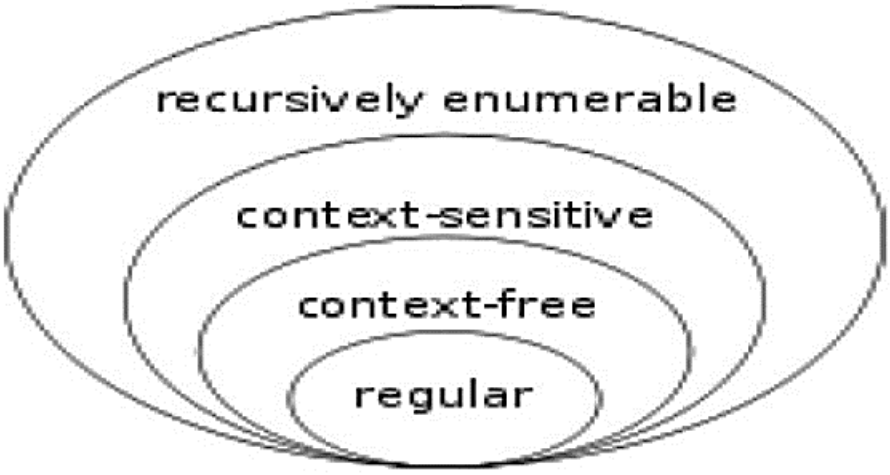
4- E→ E + E
E→ E*E
E → (E)
E→ 7
The terminals here are + * ( ) 7.

Chomsky Hierarchy

Noam Chomsky introduced the Chomsky hierarchy which classifies grammars and languages. This hierarchy can be amended by different types of machines (or automata) which can recognize the appropriate class of languages.

The Chomsky hierarchy consists of the following levels:

Type-0: grammars (unrestricted grammars) include all formal grammars.
Type-1: grammars (context-sensitive grammars) generate the context sensitive languages.
Type-2: grammars (context-free grammars) generate the context-free languages.
Type-3: grammars (regular grammars) generate the regular languages.



Every regular language is context-free, every context-free language, not containing the empty string, is context-sensitive and every context-sensitive language is recursive and every recursive language is recursively enumerable.

The following table summarizes each of Chomsky's four types of grammars, the class of language it generates, the type of automaton that recognizes it, and the form its rules must have.

Type	**Language**	**Grammar**	**Machine**
Type 3	Regular language	Regular grammar RG $N \rightarrow t \mid tN$	Finite Automata FA
Type 2	Context free language	Context free grammar CFG $u \rightarrow v, u \in N^+$ $v \in (N \cup T)^*$	Pushdown automaton PDA
Type 1	Context sensitive language	Context sensitive grammar CSG $u \rightarrow v, (u,v) \in (N \cup T)^+$	Linear bounded automaton LBA
Type 0	Recursively enumerable language	Unrestricted grammar UG $u \rightarrow v, (u,v) \in (N \cup T)^*$	Turing machine TM