جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
Department of Cyber Security

Subject: Computation Theory
Class: 3rd

Lecturer: Msc :Muntather AL-mussawee

Lecture: (7)

ChomskyNormalForm(CNF)

Chomsky Normal Form (CNF)

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 Non-terminal N **nullable** if:

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

Replacement Rule

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

$$X \rightarrow \text{old string}$$

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 Non-terminals, 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:

$S \rightarrow Xa$
$X \rightarrow aX \mid bX \mid \wedge$

X is the only nullable Non-terminal.
The new CFG is:

$S \rightarrow Xa \mid a$
$X \rightarrow aX \mid bX \mid a \mid b$

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

$(a + b)^*bb(a + b)^*$

$S \rightarrow XY$
$X \rightarrow Zb$
$Y \rightarrow bW$
$Z \rightarrow AB$
$W \rightarrow Z$
$A \rightarrow aA \mid bA \mid \wedge$
$B \rightarrow Ba \mid Bb \mid \wedge$

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

$S \rightarrow XY$
$X \rightarrow Zb \mid b$
$Y \rightarrow bW \mid b$
$Z \rightarrow AB \mid A \mid B$
$W \rightarrow Z$
$A \rightarrow aA \mid bA \mid a \mid b$
$B \rightarrow Ba \mid Bb \mid a \mid b$

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

$S \rightarrow X \mid YaY \mid aSb \mid b$
$X \rightarrow YY \mid b \mid \wedge$
$Y \rightarrow aY \mid aaX$

Convert CFG to 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:

Non-terminal → string of exactly two Non-terminals

or

Non-terminal → One Terminal

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

Conversion steps:

1- Deleting Λ- productions.
2- Convert right side to Non-terminals.
3- Convert to CNF.

Example: Convert the following CFG into CNF:

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

1- Deleting Λ- productions.

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

2- Convert right side to Non-terminals.

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 Non-terminals.

$S \rightarrow YA \mid XB$

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

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

$X \rightarrow a$

$Y \rightarrow b$

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

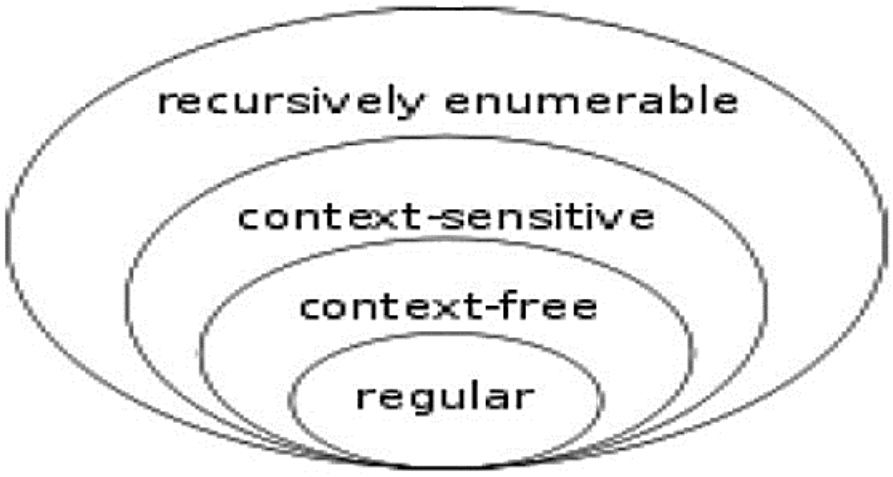
4- E→ E + E
E→ E*E
E → (E)
E→ 7
The terminals here are + * ( ) 7.

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