Explain the Chomsky Normal form?

# Chomsky Normal Form

A grammar where every production is either of the form A → BC or A → c (where A, B, C are arbitrary variables and c an arbitrary symbol).

Example:

S → AS | a A → SA | b

(If language contains ε, then we allow S → ε where S is start symbol, and forbid S on RHS.)

The key advantage is that in Chomsky Normal Form, every derivation of a string of n letters has exactly 2n − 1 steps. Thus: one can determine if a string is in the language by exhaustive search of all derivations.

The conversion to Chomsky Normal Form has four main steps:

1. Get rid of all ε productions.

2. Get rid of all productions where RHS is one variable.

3. Replace every production that is too long by shorter productions.

4. Move all terminals to productions where RHS is one terminal.

Take any two examples and convert Context Free Grammar to Chomsky Normal Form?

1. S → AbA

A → Aa | ε

Solution:

After the first step, one has:

S → AbA | bA | Ab | b

A → Aa | a

The second step does not apply. After the third step, one has:

S → T A | bA | Ab | b

A → Aa | a

T → Ab

And finally, one has:

S → T A | BA | AB | b

A → AC | a

T → AB

B → b

C → a

2. S → ASB

A → aAS|a|ε

B → SbS|A|bb

Solution:

S0->S

S → ASB

A → aAS|a|ε

B → SbS|A|bb

S0->S

S → ASB|SB

A → aAS|aS|a

B → SbS| A|ε|bb

Now, it creates null production B→ ε, its removal from the grammar yields:

S0->S

S → AS|ASB| SB| S

A → aAS|aS|a

B → SbS| A|bb

Now, it creates unit production B->A, its removal from the grammar yields:

S0->S

S → AS|ASB| SB| S

A → aAS|aS|a

B → SbS|bb|aAS|aS|a

Also, removal of unit production S0->S from grammar yields:

S0-> AS|ASB| SB| S

S → AS|ASB| SB| S

A → aAS|aS|a

B → SbS|bb|aAS|aS|a

Also, removal of unit production S->S and S0->S from grammar yields:

S0-> AS|ASB| SB

S → AS|ASB| SB

A → aAS|aS|a

B → SbS|bb|aAS|aS|a

S0-> AS|ASB| SB

S → AS|ASB| SB

A → XAS|XS|a

B → SYS|bb|XAS|XS|a

X →a

Y→b

Also, B->bb can’t be part of CNF, removing it from grammar yields:

S0-> AS|ASB| SB

S → AS|ASB| SB

A → XAS|XS|a

B → SYS|VV|XAS|XS|a

X → a

Y → b

V → b

S0-> AS|PB| SB

S → AS|ASB| SB

A → XAS|XS|a

B → SYS|VV|XAS|XS|a

X → a

Y → b

V → b

P → AS

Similarly, S->ASB has more than two symbols, removing it from grammar yields:

S0-> AS|PB| SB

S → AS|QB| SB

A → XAS|XS|a

B → SYS|VV|XAS|XS|a

X → a

Y → b

V → b

P → AS

Q → AS

Similarly, A->XAS has more than two symbols, removing it from grammar yields:

S0-> AS|PB| SB

S → AS|QB| SB

A → RS|XS|a

B → SYS|VV|XAS|XS|a

X → a

Y → b

V → b

P → AS

Q → AS

R → XA

Similarly, B->SYS has more than two symbols, removing it from grammar yields:

S0 -> AS|PB| SB

S → AS|QB| SB

A → RS|XS|a

B → TS|VV|XAS|XS|a

X → a

Y → b

V → b

P → AS

Q → AS

R → XA

T → SY

Similarly, B->XAX has more than two symbols, removing it from grammar yields:

S0-> AS|PB| SB

S → AS|QB| SB

A → RS|XS|a

B → TS|VV|US|XS|a

X → a

Y → b

V → b

P → AS

Q → AS

R → XA

T → SY

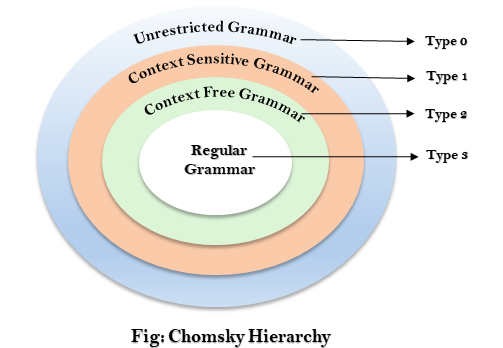
U → XA

Explain Chomsky Hierarchy ?

# Chomsky Hierarchy

# Chomsky Hierarchy represents the class of languages that are accepted by the different machine. The category of language in Chomsky's Hierarchy is as given below:

1. Type 0 known as Unrestricted Grammar.
2. Type 1 known as Context Sensitive Grammar.
3. Type 2 known as Context Free Grammar.
4. Type 3 Regular Grammar.



This is a hierarchy. Therefore every language of type 3 is also of type 2, 1 and 0. Similarly, every language of type 2 is also of type 1 and type 0, etc.

# Type 0 Grammar:

Type 0 grammar is known as Unrestricted grammar. There is no restriction on the grammar rules of these types of languages. These languages can be efficiently modeled by Turing machines.

**For example:**

1. bAa → aa
2. S → s

# Type 1 Grammar:

Type 1 grammar is known as Context Sensitive Grammar. The context sensitive grammar is used to represent context sensitive language. The context sensitive grammar follows the following rules:

* The context sensitive grammar may have more than one symbol on the left hand side of their production rules.
* The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
* The rule of the form A → ε is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.
* The Type 1 grammar should be Type 0. In type 1, Production is in the form of V → T

Where the count of symbol in V is less than or equal to T.

**For example:**

1. S → AT
2. T → xy
3. A → a

# Type 2 Grammar:

Type 2 Grammar is known as Context Free Grammar. Context free languages are the languages which can be represented by the context free grammar (CFG). Type 2 should be type 1. The production rule is of the form

1. A → α

Where A is any single non-terminal and is any combination of terminals and non-terminals.

**For example:**

1. A → aBb
2. A → b
3. B → a

# Type 3 Grammar:

Type 3 Grammar is known as Regular Grammar. Regular languages are those languages which can be described using regular expressions. These languages can be modeled by NFA or DFA.

Type 3 is most restricted form of grammar. The Type 3 grammar should be Type 2 and Type 1. Type 3 should be in the form of

1. V → T\*V / T\*

**For example:**

1. A → xy