1. Grammar formalisms

- CFG
- Grammar rules
- Treebanks
- Normal forms for Grammar
  - CNF
  - GNF

2. Dependency - Based syntax.

- Dependency Grammar
- Feature structures
- Unification of feature structure

3. Parsing Techniques

- syntactic parsing
- Ambiguity
- Dynamic programming parsing
- Shallow parsing.

4. Probabilistic Parsing.

1. PCFG
2. P CYK Alg.
3. Probabilistic Lexicalized CFGs

Module-3 (NLP)

Context free Grammar.
CFG

NLP is the capability of computer software to understand the natural language.

↓

* There are variety of languages in the world.
* Each language has its own structure, like SVO, SOV

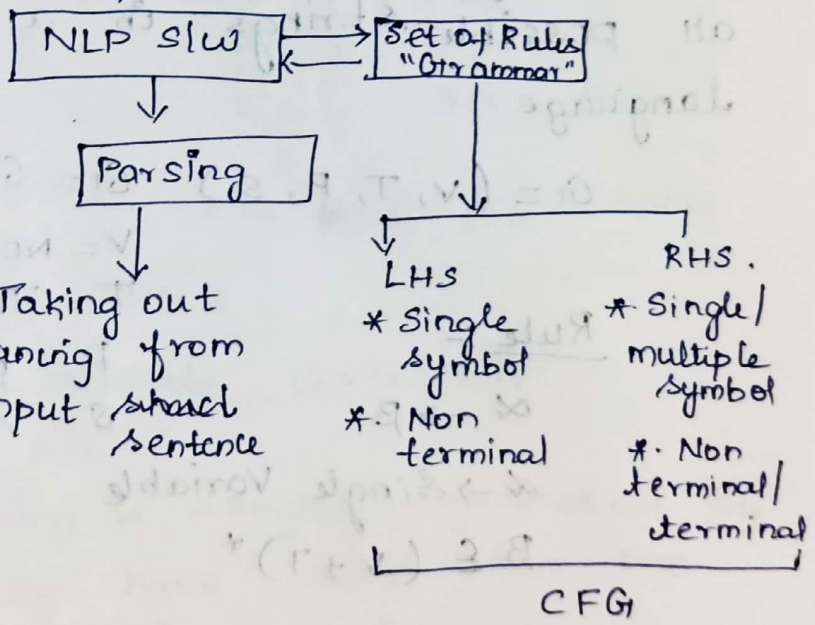
English

S V O SOVX.
I eat mango.

↓ called
Grammar
↓ Has
certain set of rules
→ What is allowable
→ What is not allowed

Other Lang :-
S O V
O S V

Natural Language as input (sentences)



Ex :-

John hit the ball

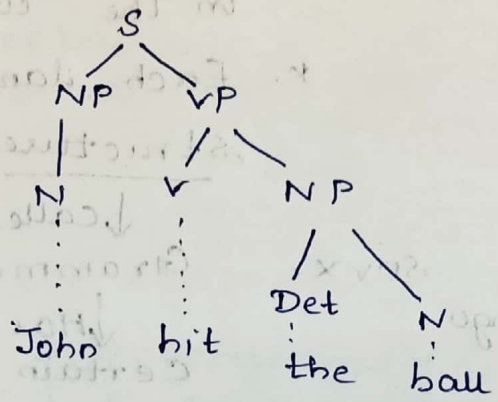
$S \rightarrow NP \quad VP$

$VP \rightarrow V \quad NP$

$NP \rightarrow N$

$NP \rightarrow Det$

L.H.S. R.H.S



CFG :-

Context free grammer is a formal grammer which is used to generate all possible strings in a given formal language.

$G = (V, T, P, S)$ G = Grammer

V = Non-terminal set

T = Terminal set

P = production rule

S = Start Symbol.

Rule :-

$\alpha \rightarrow \beta$

$\alpha \rightarrow$ Single Variable

$\beta \in (V+T)^*$

T = { this, that, the, book, flight, we, read, John, ball ... } All the words in vocabulary & terminal

V = { S, NP, N, VP, V, Det, Nour, Aux ... }

P = {

S → NP VP

S → Aux NP VP

S → VP

NP → Det Noun

VP → Verb

VP → Verb NP

}

Det → the | a | this | that

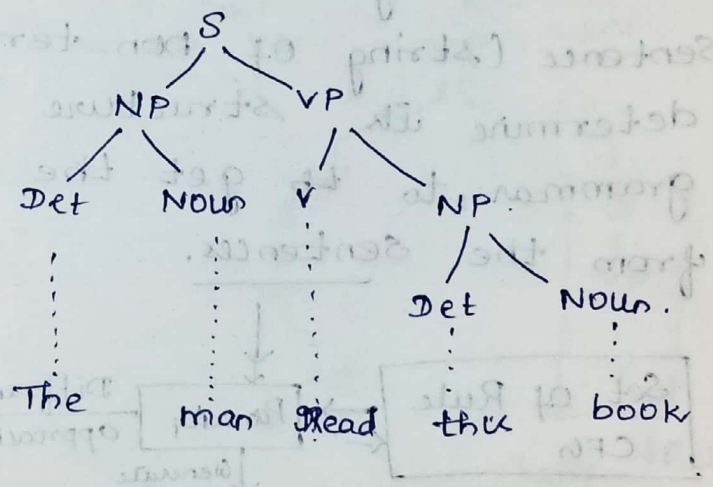
Noun → book | flight | John | ball

Verb → book | include | Read

Aux → does | is

Ex :-

The man read this book



Properties of CFG :-

⇒ Set of possible derivation

⇒ String SET*

⇒ Each string is language generated by CFG may have more than one derivation (Ambiguity)

book < study purpose
book < Bus ticket

Bank < financial
Bank < River.

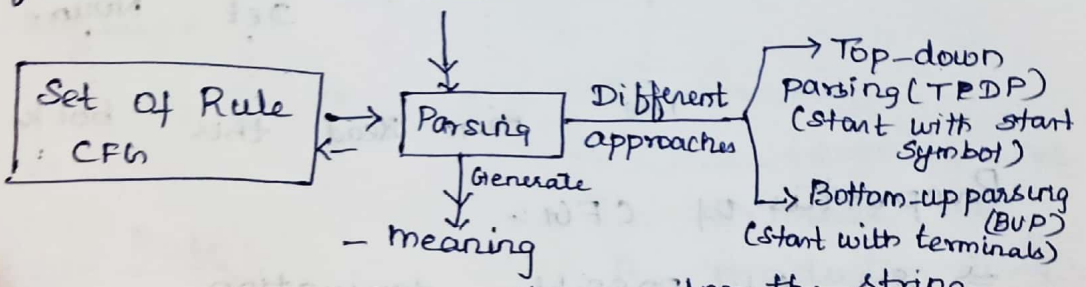
Summary :-

⇒ CFG is a list of rules that define the set of all well formed sentences in a language.

⇒ Each rule has a left hand side, which identified a syntatic category, & a right hand side define its alternative components parts.

CFG Parsing :-

Parsing is a method of analyzing a sentence (string of non-terminals) to determine its structure according to the grammar, to get the meaning out from the sentences.



police are looking for a thief with a bicycle

police with bicycle

OR thief " "

Ex :-

String OR Sentence given :-

book that flight.

Rules Given.

$S \rightarrow NP \quad VP$

$S \rightarrow VP$

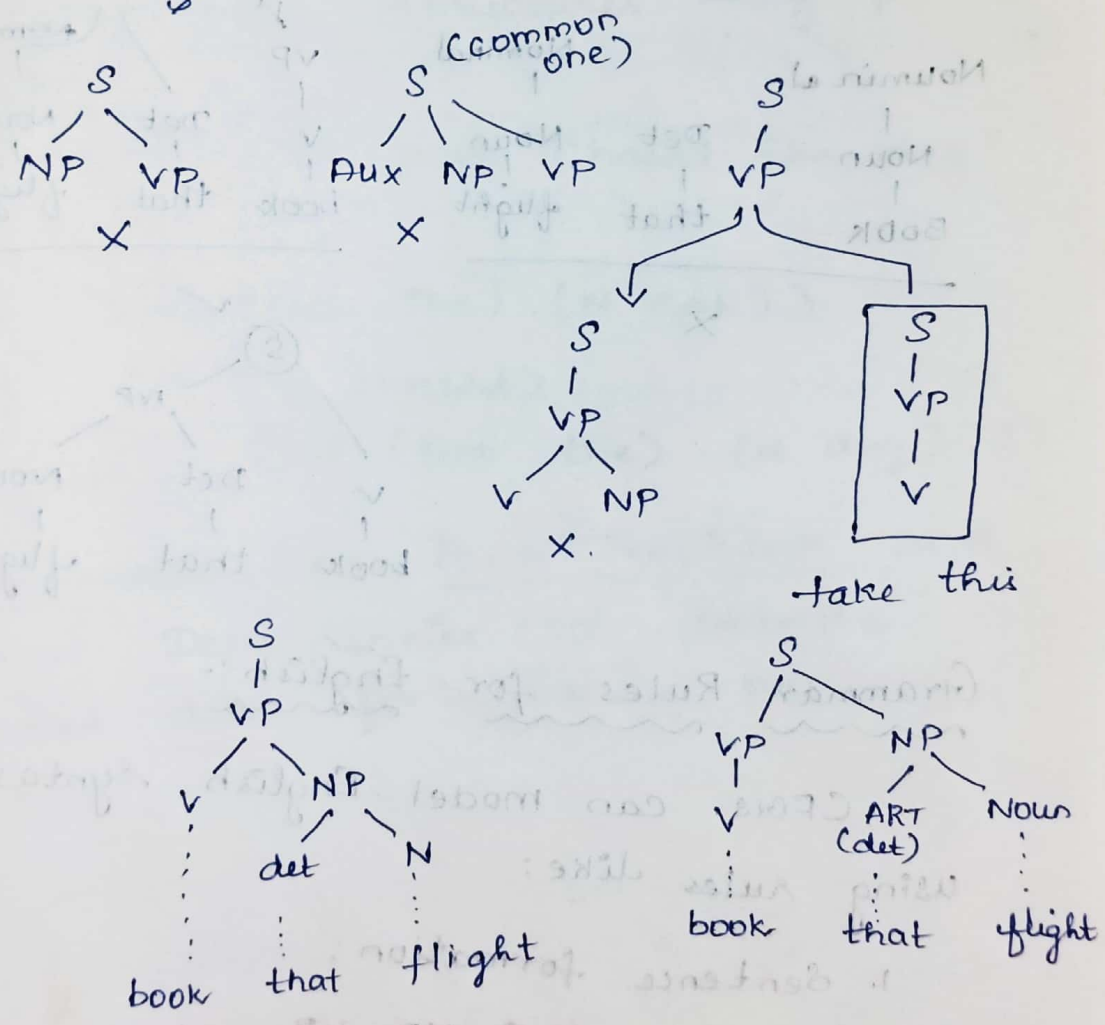
$NP \rightarrow ART^{(det)} \quad N$

$NP \rightarrow ART^{(det)} \quad ADJ \quad N$

$VP \rightarrow V$

$VP \rightarrow V \quad NP.$

Construct parse tree :- TDP



Bottom-up parsing :- (BUP)

1. Start with the i/p text
2. Derive the text from Rule

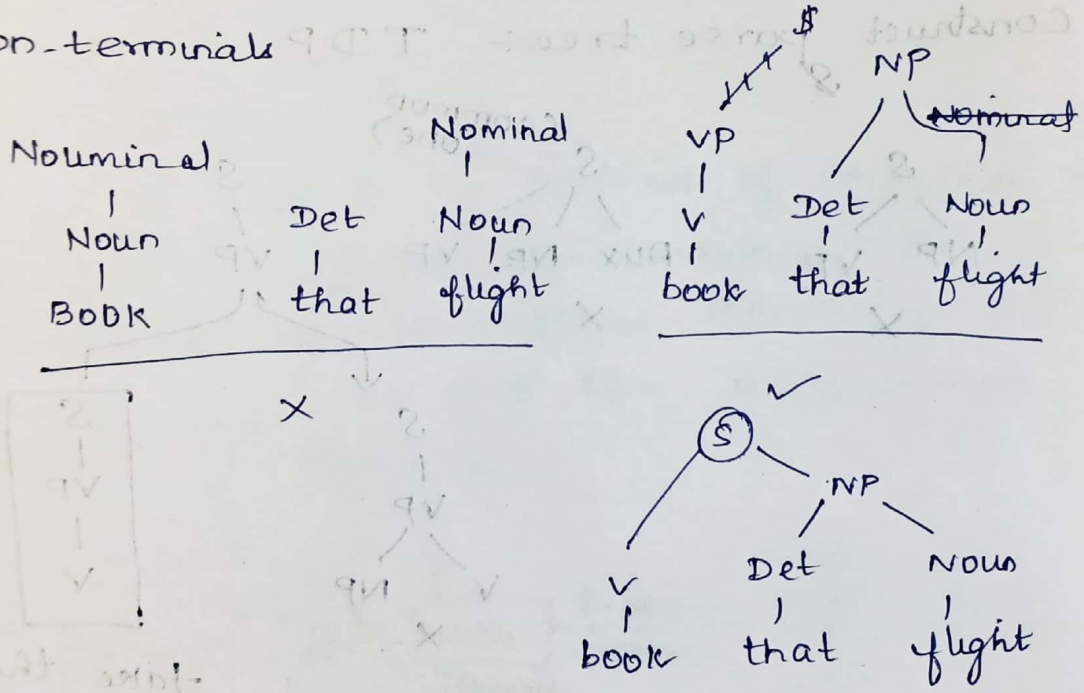
Noun Det Noun

Book that flight

Verb Det Noun

Book that flight

3. Each of these can be derived from non-terminals



Grammar Rules for English :-

CFGs can model English syntax using rules like;

1. Sentence formation :

$S \rightarrow NP \quad VP$

2. Noun phrase:

NP → Det N | N.

3. Verb phrase:

VP → V NP | V

For complex structures, additional rules handles adjective phrases (AP), prepositional phrases (PP) & subordination.

Treebanks :-

A treebank is a corpus where sentences are manually annotated with syntactic structures using parse trees.

Ex :- "The cat chased the dog"

```
(S
  (NP (Det The) (N cat))
  (VP (V chased)
      (NP (Det the) (N dog))))
```

Treebanks like Penn Treebank and Universal Dependencies (UD) provide annotated data for training parsers.

Normal Forms for Grammar.

Grammars can be transformed into normal forms for efficient parsing.

⊛ Chomsky Normal Form (CNF):

Each rule is of the form

$A \rightarrow BC$ or $A \rightarrow a$,

where A, B, C ⇒ non-terminals

a ⇒ terminal.

⊛ Greibach Normal Form (GNF):

Rules have the form $A \rightarrow a\alpha$, where a is terminal & α is a sequence of non terminals.

Ex CNF conversion

From

$S \rightarrow NP \quad VP$

$VP \rightarrow V \quad NP$

$NP \rightarrow Det \quad N$

To CNF:

$S \rightarrow X \quad VP$

$X \rightarrow NP$

$VP \rightarrow V \quad Y$

$Y \rightarrow NP$

$NP \rightarrow Det \quad N.$

Chomsky Normal Form
CNF.

In chomsky Normal Form (CNF) we have a restriction on the length of RHS; which is, elements in RHS should either be two variables or a Terminal.

A CFG is in CNF if the production are in the following forms:

$$A \to a$$
$$A \to BC$$

where A, B & C are non-terminal and a is a terminal.

Steps to convert a given CFG to CNF.

Step 1 :- If the start symbol S occurs on some right side, create a new start symbol S' and a new production $S' \to S$

Step 2: Remove Null productions. (Using the Null production) Rem

Step 3: Remove Unit productions

Step 4: Replace each production $A \to B_1 \ldots B_n$ where $n > 2$, with $A \to B_1 C$ where $C \to B_2 \ldots B_n$

: Repeat this step for all productions having 2 or mor symbols on the right side.

Step 5: If the right side of any production is in the form $A \to aB$ where 'a' is a terminal and A & B are non-terminals, then the production is replaced by $A \to XB$ & $X \to a$.

Repeat this step for every production which is of the form $A \to aB$.

Ex :-

Convert the following CFG to CNF.

P: $S \to ASA \mid aB$, $A \to B \mid S \mid \varepsilon$, $B \to b \mid \varepsilon$

Soln :-

1) Since S appears in RHS, we add a new state S' and $S' \to S$ is added to the production.

P: $S' \to S$, $S \to ASA \mid aB$,
$A \to B \mid S \mid \varepsilon$,
$B \to b \mid \varepsilon$

2) Remove the null productions:
$B \to \varepsilon$ and $A \to \varepsilon$

After Removing $B \to \varepsilon$:

P: $S' \to S$
$S \to ASA \mid aB \mid a$, $A \to B \mid S \mid \varepsilon$, $B \to b$

After Removing $A \to \varepsilon$:

P: $S' \to S$, $S \to ASA \mid aB \mid a \mid AS \mid SA \mid S$,
$A \to B \mid S$, $B \to b$

3) Remove the Unit Production:

$S \to S$, $S' \to S$, $A \to B$ & $A \to S$:

After Removing $S \to S$:

P: $S' \to S$, $S \to ASA|aB|a|AS|SA$,
$A \to B|S$, $B \to b$.

After Removing $S' \to S$:

P: $S' \to ASA|aB|a|AS|SA$,
$S \to ASA|aB|a|AS|SA$,
$A \to B|S$
$B \to b$

After Removing $A \to B$:

P: $S' \to ASA|aB|a|AS|SA$,
$AS \to ASA|aB|a|AS|SA$,
$A \to b|S$
$B \to b$

After Removing $A \to S$:

P: $S' \to ASA|aB|a|AS|SA$,
$S \to ASA|aB|a|AS|SA$
$A \to b|ASA|aB|a|AS|SA$
$B \to b$.

~~After Rem~~

4) Now find out the productions that has more than Two variables in RHS

$S' \to ASA$, $S \to ASA$ & $A \to ASA$.

After removing these, we get:

P: $S' \to AX|aB|a|AS|SA$
$S \to AX|aB|a|AS|SA$
$A \to b|AX|aB|a|AS|SA$
$B \to b$, $X \to SA$

5) Now change the productions

$S' \to aB$, $S \to aB$ and $A \to aB$}

Finally we get:

P: $S \to AX \mid YB \mid a \mid AS \mid SA$

$S \to AX \mid YB \mid a \mid AS \mid SA$

$B \to b \mid AX \mid YB \mid a \mid AS \mid SA$

$B \to b$

$X \to SA$

$Y \to a$

Ex :- 2. Given CFG : convert to CNF.

$S \to NP \ VP$

$VP \to V \ NP \mid V$

$NP \to Det \ N \mid N$

$Det \to$ "the" | "a"

$N \to$ "cat" | "dog"

$V \to$ "chased" | "barked"

convert to CNF.

Soln :-

1. Break long rules into binary rules

$S \to NP \ VP$ (Already in CNF)

$VP \to V \ NP$ (" " ")

$NP \to Det \ N$ (" " ")

$VP \to V$ (Needs modification) $\to$

Introduce new non-terminal X:

$VP \to V \ X$

$X \to \varepsilon$

2. Ensure all right-hand sides contain at most two symbols.

Final CNF Grammar:

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP \mid V\ X$

$X \rightarrow \varepsilon$

$NP \rightarrow Det\ N$

$Det \rightarrow$ "the" | "a"

$N \rightarrow$ "cat" | "dog"

$V \rightarrow$ "chased" | "barked".

CNF - Use :-

→ CNF is a restricted form of CFG that simplifies parsing alg & efficiently processes sentences in NLP.

* → Mandatory for CYK parsing
* → Improves parsing efficiency.
* → Reduces ambiguity in grammars.
* → Works well with probabilistic model (PCFGs)
* → Standardizes grammar for NLP applications.