Formal Language and Automata Theory

Chapter Four

Context Free Languages

Context free language

- Not all languages are regular.

Con't

Grammar:

- A formalism to generate strings in a language by a process of replacing
- symbols.

  has 4 elements (tuples) represented as: G= (N, Σ, P, S) where
- N is a finite set of non-terminal symbols. In natural languages, this can be syntactic categories, phrases or sentences.
- Σ is a finite set of terminal symbols (disjoint from N). It consists of elements of target language such as words and letters in natural language.
- P is a finite set of production rules of the form a $\rightarrow$ b with at least one nonterminal in a.
- S is member of N called the start symbol (special non-terminal symbol). In natural languages, the start symbol is a sentence.

Con’t

Hierarchy of Grammars/Languages

- Also known as Chomsky Classification, the hierarchy of grammars/languages represents a hierarchy of expressiveness of grammars.

• Different classes of grammars/languages are defined by putting different constraints on production rules resulting in different structural complexity of sentences of natural languages.

• Chomsky classification consists of the following four levels of grammars/languages:

  - Type 0 (Unrestricted / Recursively Enumerable)
  - Type I (Context-Sensitive)
  - Type II (Context-Free)
  - Type III (Regular)

Con’t

Type 0 (Unrestricted):

- No limitation on production rules
- At least one non-terminal on left hand side

e.g. $S \rightarrow S\ S$

$S \rightarrow A\ B\ C$

$A\ B \rightarrow B\ A$

$B\ A \rightarrow A\ B$

$A\ C \rightarrow C\ A$

$C\ A \rightarrow A\ C$

$B\ C \rightarrow C\ B$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

$S \rightarrow \epsilon$

Valid strings generated include: ϵ, abc, aabbcc, cabcab, etc…

Type I (Context-Sensitive):

- Production rule:

$\alpha 1 B \alpha 2 \rightarrow \alpha 1 \beta \alpha 2$ where

B is non-terminal symbol

$\alpha 1$, $\alpha 2$, β are all (possibly empty) sequences of terminal and non-terminal symbols ($\alpha 1$ is

left context and $\alpha 2$ is right context.

$S \rightarrow \epsilon$ is allowed if S does not appear on right hand side of any rule

These rules are used in natural languages to describe subject-verb agreement with respect to number, i.e. singular or plural as reflected in sentences: *the students come* and *the student comes*.

Con't

For example, the following production rules can be used to describe such contexts.

S → NP VP [S=Sentence, NP= Noun Phrase, VP= Verb Phrase]

NP → Det Nsing [Det= Determiner, Nsing= Noun (singular)]

NP → Det Nplur [Nplur= Noun (plural)]

Nsing VP → Nsing Vsing [Vsing= Verb (singular)]

Nplur VP → Nplur Vplur [Vplur= Verb (plural)]

Det → the

Nsing → student

Nplur → students

Vsing → comes

Vplur → come

Note: Context-Sensitive Languages/Grammars are subsets of Unrestricted Languages/ Grammars.

Con’t

Type II (Context-Free):

- Production rule:
- Exactly one non-terminal on left hand side, but anything on the right hand side.
- These rules are used to describe grammars of natural languages that are context-free. For
- example, past tenses of English are context-free with respect to the subject. Thus, it is grammatically correct to construct the sentences: *the students came* and *the student came*.

Con’t

- The following production rules can be used to represent such context-free grammars.
- S → NP VP [S=Sentence, NP= Noun Phrase, VP= Verb Phrase]
- NP → Det N [Det= Determiner, N= Noun]
- VP→ V [V= Verb]
- Det → the
- N→ student
- N→ students
- V→ came
- Context-Free Grammars are important since they are:
- Restricted enough to build efficient parsers
- Powerful enough to describe the syntax of most programming languages
- Note: Context-Free Languages/Grammars are subsets of Context-Sensitive Languages/Grammars.

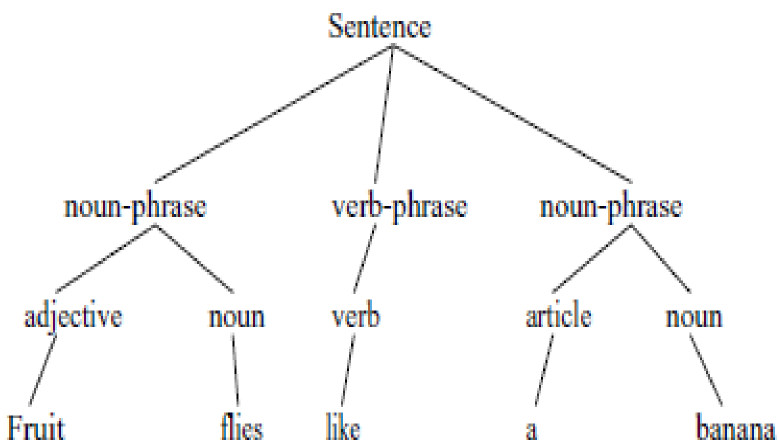
Con’t

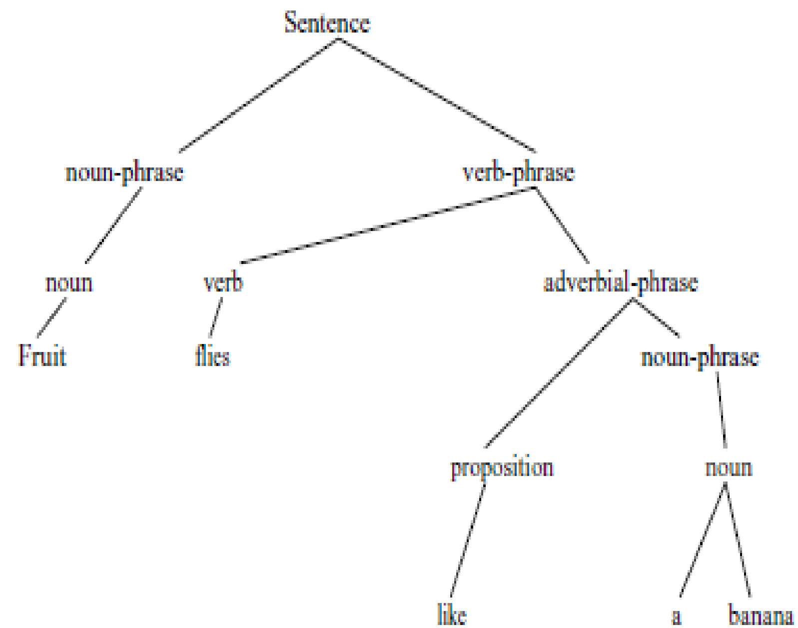
Type III (Regular):

- Production rule:
- Exactly one non-terminal on left hand side, and one terminal and at most one nonterminal on right hand side.
- Examples:
- $A \rightarrow aB$ Right Regular Grammar
- $A \rightarrow Ba$ Left Regular Grammar
- $A \rightarrow a$
- Note: Regular Languages/Grammars are subsets of Context-Free Languages/Grammars.

Con’t

- In natural language processing, one way of showing the analysis of a sentence is through the use of a syntax tree. E.g. “ fruit flies like a banana”

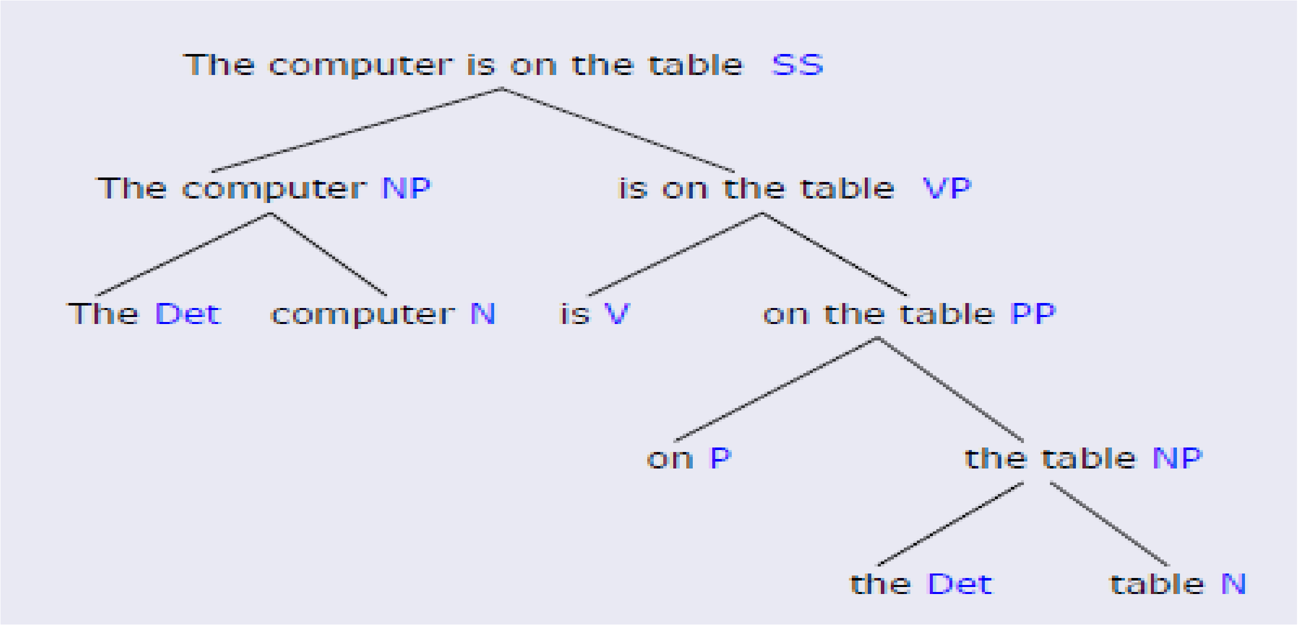




Tree representation

Simple sentence

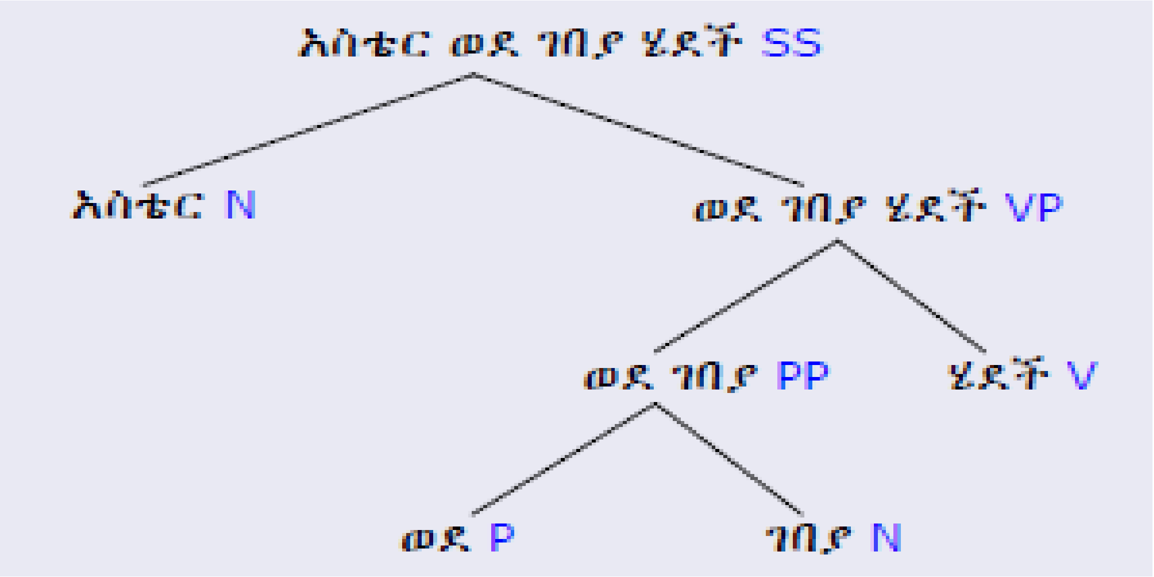
Example(for English)



Con’t

Simple sentence

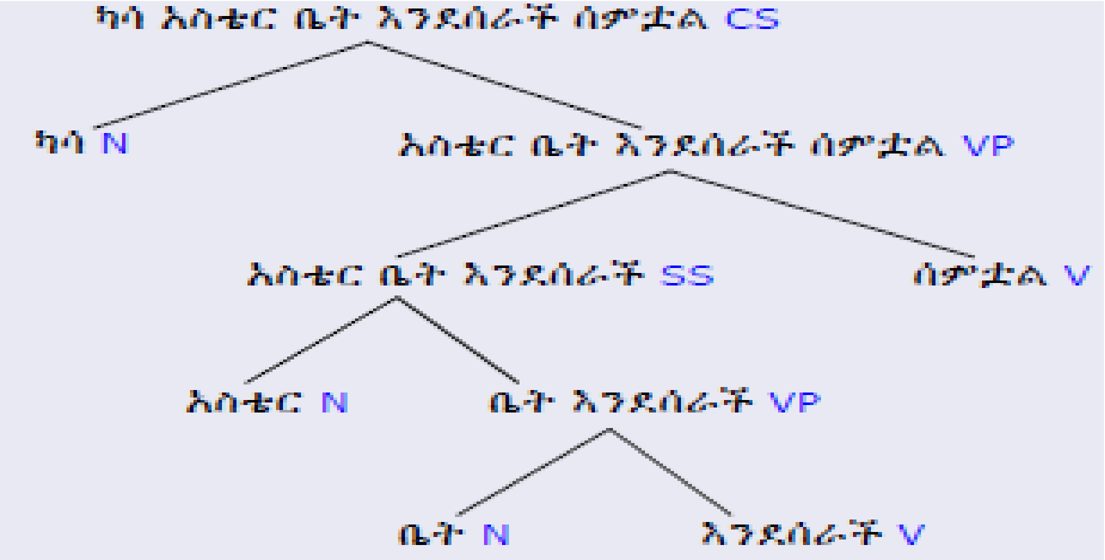
Example(for Amharic)



Con’t

Complex sentence

Example(for Amharic)



Parsing

- Parsing is a derivation process which identifies the structure of sentences using a given grammar.
  - considered as a special case of a search problem.
- two basic methods of searching are used
  - top-down strategy
  - bottom-up strategy

Con’t

Top-down Parsing

- Top-down parsing starts with the symbol S and then searches through different ways to rewrite the symbols until the input sentence is generated.

Con’t

Example

Given the following English grammar.

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$NP \rightarrow NAME$

$NP \rightarrow DET\ N$

$NAME \rightarrow Abebe$

$V \rightarrow killed$

$DET \rightarrow the$

$N \rightarrow lion$

Con’t

Then, the sentence **Abebe killed the lion** can be parsed using top-down strategy as follows.

- S ⇒ NP VP [rewriting S]
- ⇒ NAME VP [rewriting NP]
- ⇒ Abebe VP [rewriting NAME]
- ⇒ Abebe V NP [rewriting VP]
- ⇒ Abebe killed NP [rewriting V]
- ⇒ Abebe killed DET N [rewriting NP]
- ⇒ Abebe killed the N [rewriting DET]
- ⇒ Abebe killed the lion [rewriting N]

Con’t

Bottom-up Parsing

- Bottom-up parsing starts with words in a sentence and uses production rules backward to reduce the sequence of symbols until it consists solely of S.

Con’t

Given the following English grammar.

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$NP \rightarrow NAME$

$NP \rightarrow DET\ N$

$NAME \rightarrow$ Abebe

$V \rightarrow$ killed

$DET \rightarrow$ the

$N \rightarrow$ lion

Con’t

- Then, the sentence **Abebe killed the lion** can be parsed using bottom-up strategy as follows.

Abebe killed the lion	
NAME killed the lion	[rewriting Abebe]
NAME V the lion	[rewriting killed]
NAME V DET lion	[rewriting the]
NAME V DET N	[rewriting lion]
NP V DET N	[rewriting NAME]
NP V NP	[rewriting DET N]
NP VP	[rewriting V NP]
S	[rewriting NP VP]