Unit 4
Syntax

CFG, Probabilistic CFG
Word's Constituency (Phrase level, Sentence level),
Parsing (Top-Down and Bottom-Up),
CYK Parser, Probabilistic Parsing

Natural Language Processing (NLP)
MDS 555

Objective

- CFG
- Probabilistic CFG
- Word’s Constituency (Phrase level, Sentence level)
- Parsing (Top-Down and Bottom-Up)
- CYK Parser
- Probabilistic Parsing

Grammar

- Grammar is the structure and system of a language
- It consists of
  - Syntax
  - Morphology

Constituency

- Syntactic constituency is the idea that groups of words/characters can behave as single units, or constituents.
- Part of developing a grammar involves building an inventory of the constituents in the language.
- **Constituents**
  - are groups of words behaving as single units and consist of phrases, words, or morphemes

Constituency

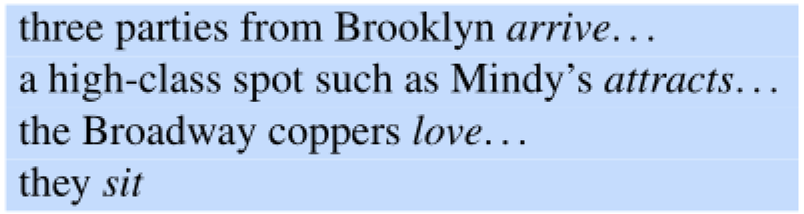
- Consider the noun phrase, a sequence of words surrounding at least one noun.

  noun phrase = determiner/quantifier + adjectives + noun (+ complements/modifiers)

  - Here are some examples of noun phrases

    Those three little kittens
    A basket of a fresh fruits

  - One piece of evidence is that they can all appear in similar syntactic environments, for example, before a verb



three parties from Brooklyn *arrive*. . .
a high-class spot such as Mindy's *attracts*. . .
the Broadway coppers *love*. . .
they *sit*

Morphemes

- A morpheme is the smallest unit of meaning in a language
- A typical word consists of one or more morphemes
- Morphemes lack independence, as words can comprise multiple morphemes
- For example, “apple” is a word and also a morpheme. “Apples” is a word comprised of two morphemes, “apple” and “-s”, which is used to signify the noun is plural.

Morphemes

- Free morpheme: A free morpheme is the smallest unit of meaning in a language that can stand alone as a word.
  - घर, पानी, काम
- Bound morpheme: A bound morpheme is the smallest unit of meaning that cannot stand alone.
  - -हरू, -को, -ले, -ला
    - -s (plural, as in cats)
    - -ed (past tense, as in walked)
    - un- (negation, as in unhappy)

Words

- Words are the smallest units with independent meanings, making them the focus of our analysis
- In NLP in word level we perform: POS Tagging
- Primary POS/Tags
  - Noun(N), Verb(V), Adjective(ADJ), Adverb(ADV)

Phrases

- A phrase can consist of a single word or a combination of words, depending on its position and role in a sentence.
- There are five major categories of phrases:
  - Noun Phrase (NP)
  - Verb Phrase (VP)
  - Adjective Phrase (ADJP)
  - Adverb Phrase (ADVP)
  - Preposition Phrase (PP)

Phrases

- **Noun phrase (NP):** These phrases revolve around a noun as the head word and often serve as subjects or objects of verbs. They can typically be replaced by a pronoun without affecting the sentence's syntactical correctness.
  - – the tall boy
- **Verb phrase (VP):** Verb phrases have a verb as the headword, and they can take various forms. Some include finite verb components, while others focus on the finite verb itself. These play a significant role in both constituency and dependency grammars.
  - A phrase built around a verb. It can include auxiliary verbs, adverbs, or objects.
  - Example: “is playing football”
    - – Head verb: playing
    - – Auxiliary: is
    - – Object: football

Phrases

- **Adjective phrase (ADJP):** These phrases feature an adjective as the head word and serve to describe or qualify nouns and pronouns in a sentence.
    - “very beautiful”
        - Head adjective: beautiful
        - Modifier: very (adverb)
- **Adverb phrase (ADVP):** Adverb phrases use an adverb as the head word and serve as modifiers for nouns, verbs, or other adverbs.
- **Prepositional phrase (PP):** Prepositional phrases involve a preposition as the head word and other lexical components, providing additional details that describe other words or phrases.
    - Example: “in the park”
        - Preposition: in
        - Noun Phrase: the park

Context Free Grammars

- A widely used formal system for modeling constituent structure in natural language is the context-free grammar, or CFG.
- Context-free grammars are also called phrase-structure grammars, and the formalism is equivalent to Backus-Naur form,(BNF).
- The idea of basing a grammar on constituent structure dates back to the psychologist Wilhelm Wundt (1900) but was not formalized until Chomsky (1956) and, independently, Backus (1959).

CFG

- A context-free grammar consists of a set of rules or productions, each of which expresses the ways that symbols of the language can be grouped and ordered together, and a lexicon of words and symbols.

$$
\begin{aligned}
&\text{NP} \rightarrow \text{Det Nominal}\\
&\text{NP} \rightarrow \text{ProperNoun}\\
&\text{Nominal} \rightarrow \text{Noun} \mid \text{Nominal Noun}
\end{aligned}
$$

  – For example, the above productions express that an **NP** (or noun phrase) can be composed of
    - either a Proper Noun or a determiner (Det) followed by a Nominal;
    - a Nominal in turn can consist of one or more Nouns.

CFG

NP → Det Nominal

NP → ProperNoun

Nominal → Noun | Nominal Noun

- Example

  NP → Det Nominal → The book

  NP → ProperNoun → Nepal

  Nominal → Nominal Noun → science book

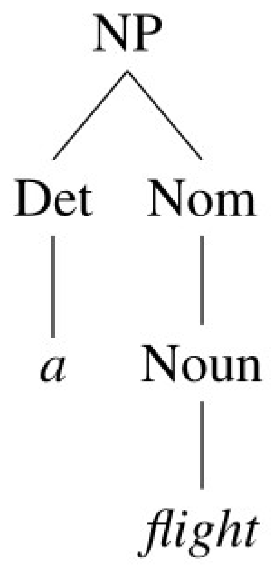
CFG

NP → Det Nominal
NP → ProperNoun
Nominal → Noun | Nominal Noun

- A CFG can be thought of in two ways:
  - as a device for generating sentences and
  - as a device for assigning a structure to a given sentence.
- Viewing a CFG as a generator,
  - we can read the → arrow as “rewrite the symbol on the left with the string of symbols on the right”.
    - So starting from the symbol:
      NP
    - we can use our first rule to rewrite NP as: DET Nominal
    - and then rewrite Nominal as: Noun
    - and finally rewrite these parts-of-speech as: a<DET> flight<NOUN>

CFG

- We say the string **a flight** can be derived from the non-terminal NP
  - A CFG can be used to generate a set of strings. This sequence of rule expansions is called a derivation of the string of words
  - It is common to represent a derivation by a parse tree (commonly shown inverted with the root at the top)



Let's add a few additional rules to our inventory. The following rule expresses the fact that a sentence can consist of a noun phrase followed by a **verb phrase**:

$$S \rightarrow NP\ VP \quad \text{I prefer a morning flight}$$

A verb phrase in English consists of a verb followed by assorted other things; for example, one kind of verb phrase consists of a verb followed by a noun phrase:

$$VP \rightarrow Verb\ NP \quad \text{prefer a morning flight}$$

Or the verb may be followed by a noun phrase and a prepositional phrase:

$$VP \rightarrow Verb\ NP\ PP \quad \text{leave Boston in the morning}$$

Or the verb phrase may have a verb followed by a prepositional phrase alone:

$$VP \rightarrow Verb\ PP \quad \text{leaving on Thursday}$$

A prepositional phrase generally has a preposition followed by a noun phrase. For example, a common type of prepositional phrase in the ATIS corpus is used to indicate location or direction:

$$PP \rightarrow Preposition\ NP \quad \text{from Los Angeles}$$

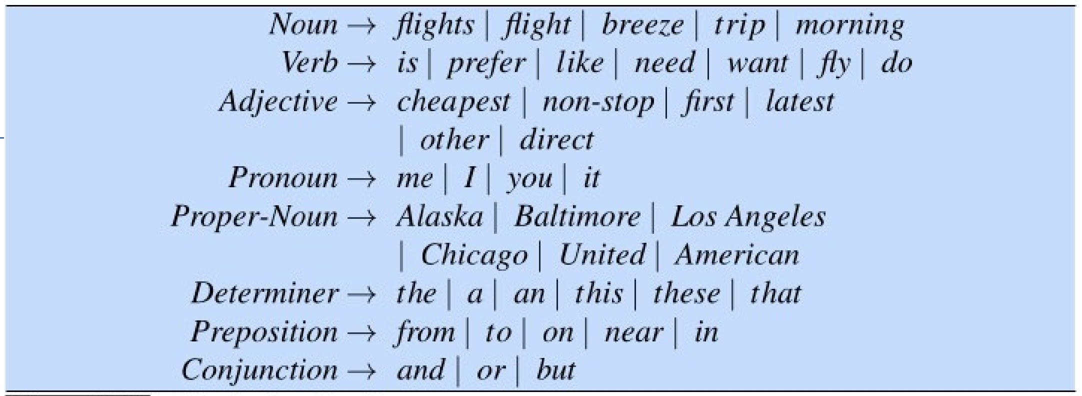
The *NP* inside a *PP* need not be a location; *PPs* are often used with times and dates, and with other nouns as well; they can be arbitrarily complex. Here are ten examples from the ATIS corpus:

to Seattle	on these flights
in Minneapolis	about the ground transportation in Chicago
on Wednesday	of the round trip flight on United Airlines
in the evening	of the AP fifty seven flight
on the ninth of July	with a stopover in Nashville

$$\begin{aligned}
\mathit{Noun} &\rightarrow \mathit{flights} \mid \mathit{flight} \mid \mathit{breeze} \mid \mathit{trip} \mid \mathit{morning} \\
\mathit{Verb} &\rightarrow \mathit{is} \mid \mathit{prefer} \mid \mathit{like} \mid \mathit{need} \mid \mathit{want} \mid \mathit{fly} \mid \mathit{do} \\
\mathit{Adjective} &\rightarrow \mathit{cheapest} \mid \mathit{non\text{-}stop} \mid \mathit{first} \mid \mathit{latest} \\
&\quad \mid \mathit{other} \mid \mathit{direct} \\
\mathit{Pronoun} &\rightarrow \mathit{me} \mid \mathit{I} \mid \mathit{you} \mid \mathit{it} \\
\mathit{Proper\text{-}Noun} &\rightarrow \mathit{Alaska} \mid \mathit{Baltimore} \mid \mathit{Los\ Angeles} \\
&\quad \mid \mathit{Chicago} \mid \mathit{United} \mid \mathit{American} \\
\mathit{Determiner} &\rightarrow \mathit{the} \mid \mathit{a} \mid \mathit{an} \mid \mathit{this} \mid \mathit{these} \mid \mathit{that} \\
\mathit{Preposition} &\rightarrow \mathit{from} \mid \mathit{to} \mid \mathit{on} \mid \mathit{near} \mid \mathit{in} \\
\mathit{Conjunction} &\rightarrow \mathit{and} \mid \mathit{or} \mid \mathit{but}
\end{aligned}$$

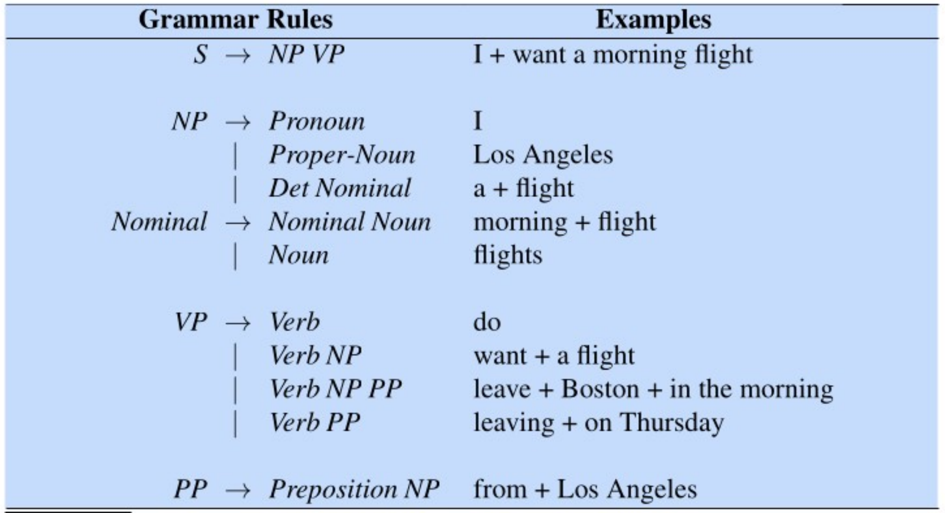
Figure 17.2 The lexicon for $\mathcal{L}_0$.

CFG



Noun →	*flights* | *flight* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly* | *do*
Adjective →	*cheapest* | *non-stop* | *first* | *latest*
	| *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles*
	| *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near* | *in*
Conjunction →	*and* | *or* | *but*

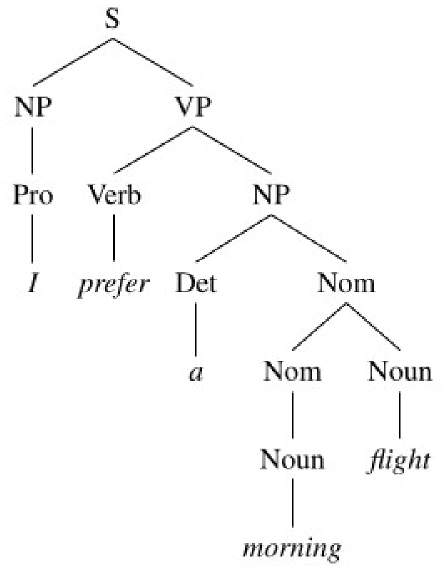
Figure 17.2 The lexicon for $\mathcal{L}_0$.



Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

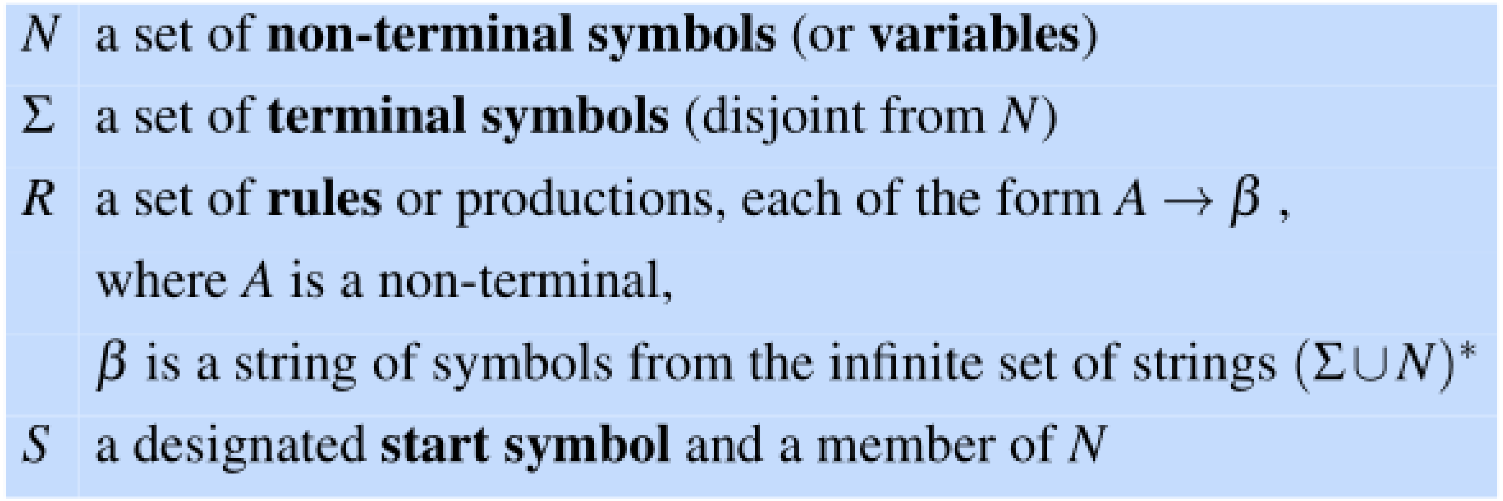
Figure 17.3 The grammar for $\mathcal{L}_0$, with example phrases for each rule.

- We can use this grammar to generate sentences of this “language”.
  - We start with S, expand it to NP VP, then choose a random expansion of NP (let’s say, to I)
  - and a random expansion of VP (let’s say, to Verb NP),
  - and so on until we generate the string
    I prefer a morning flight



Context-Free Grammar (CFG)

- Formal Definition
  - A context-free grammar (CFG) G is a quadruple (N, Σ, R, S) where



N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
S	a designated **start symbol** and a member of N

CFG - Example

- N = {q, f,}
- Σ = {0, 1}
- R = {q → 11q, q → 00f,
  f → 11f, f → ε }
- S = q
- (R= {q → 11q | 00f, f → 11f | ε })

CFG - Rules

- If A → B, then xAy → xBy and we say that
- xAy derivates xBy.

- If s → ⋯ → t, then we write s * t.
- A string x in Σ* is generated by G=(V,Σ,R,S)
  if S * x.
- L(G) = { x in Σ* | S * x}.

CFG - Example

- G = ({S}, {0,1}. {S $\rightarrow$ 0S1 | ε }, S)
  - ε in L(G) because

    S $\rightarrow$ ε .
  - 01 in L(G) because

    S $\rightarrow$ 0S1 $\rightarrow$ 01.
  - 0011 in L(G) because

    S $\rightarrow$ 0S1 $\rightarrow$ 00S11 $\rightarrow$ 0011.
  - L(G) = $\{0^n 1^n \mid n \geq 0\}$

Context-free Language (CFL)

- A language L is context-free if there exists a CFG G such that L = L(G).
- A grammar **G** generates a language **L**

Example

- Some notes:
  - **Note 1:** In P, pipe symbol (|) is used to combine productions into single representation for productions that have same LHS.
    - For example, Det → ‘a’ | ‘the’ derived from two rules Det → ‘a’ and Det → ‘the’. Yet it denotes two rules not one.
  - **Note 2:** The production highlighted in red are referred as grammar, and green are referred as lexicon.
  - **Note 3:**
    - NP – Noun Phrase, VP – Verb Phrase, PP – Prepositional Phrase, Det – Determiner, Aux – Auxiliary verb

Sample derivation

- S → NP VP

  → Det Noun VP

  → the Noun VP

  → the child VP

  → the child Verb NP

  → the child ate NP

  → the child ate Det Noun

  → the child ate a Noun

  → the child ate a cake

P = { S → NP VP

NP → Det Noun | NP PP

PP → Pre NP

VP → Verb NP

Det → ‘a’ | ‘the’

Noun → ‘cake’ | ‘child’ | ‘fork’

Pre → ‘with’

Verb → ‘ate’}

Probabilistic Context Free Grammar (PCFG)

- PCFG is an extension of CFG with a probability for each production rule
- Ambiguity is the reason why we are using probabilistic version of CFG
  - For instance, some sentences may have more than one underlying derivation.
  - The sentence can be parsed in more than one ways.
  - In this case, the parse of the sentence become ambiguous.
- To eliminate this ambiguity, we can use PCFG to find the probability of each parse of the given sentence

PCFG - Definition

- A probabilistic context free grammar G is a quintuple G = (N, T, S, R, P) where
  - (N, T, S, R) is a context free grammar

    where N is set of non-terminal (variable) symbols, T is set of terminal symbols, S is the start symbol and R is the set of production rules where each rule of the form A → S
  - A probability P(A → s) for each rule in R. The properties governing the probability are as follows;
    - P(A → s) is a conditional probability of choosing a rule A → s in a left-most derivation, given that A is the non-terminal that is expanded.
    - The value for each probability lies between 0 and 1.
    - The sum of all probabilities of rules with A as the left hand side non-terminal should be equal to 1.

$$\sum_{A \rightarrow s \in R:\ A = LHS} P(A \rightarrow s) = 1$$

PCFG - Example

- Probabilistic Context Free Grammar G = (N, T, S, R, P)

  N = {S, NP, VP, PP, Det, Noun, Verb, Pre}

  T = {‘a’, ‘ate’, ‘cake’, ‘child’, ‘fork’, ‘the’, ‘with’}

  S = S

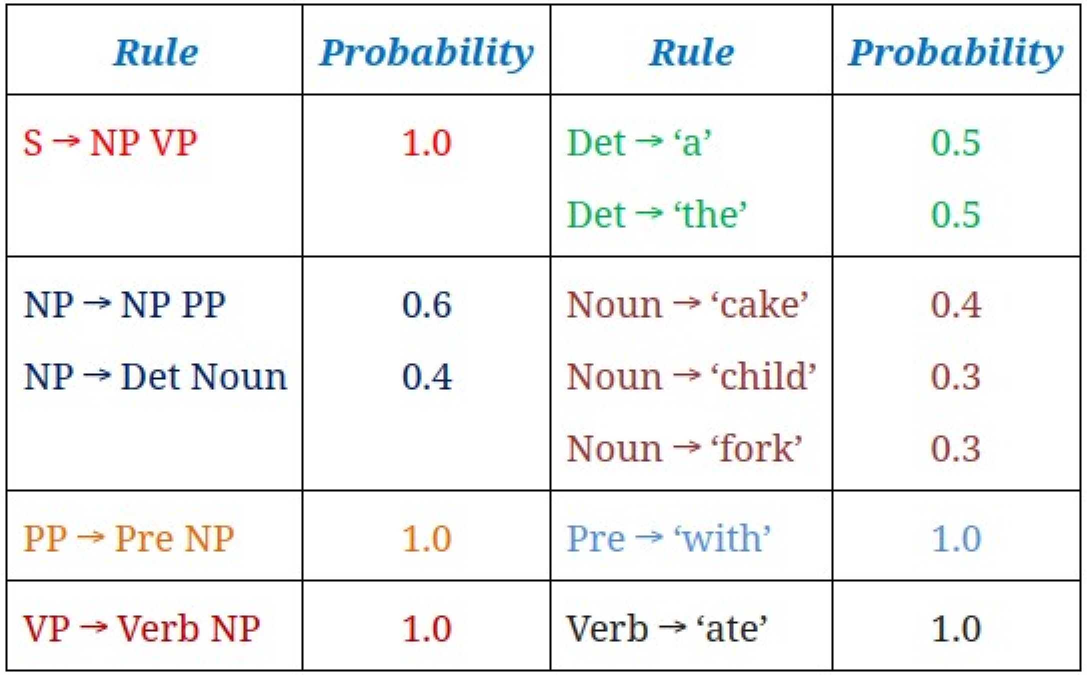
  R = { S → NP VP
  NP → Det Noun | NP PP
  PP → Pre NP
  VP → Verb NP
  Det → ‘a’ | ‘the’
  Noun → ‘cake’ | ‘child’ | ‘fork’
  Pre → ‘with’
  Verb → ‘ate’ }

PCFG - Example

R = {

S → NP VP
NP → Det Noun | NP PP
PP → Pre NP
VP → Verb NP
Det → ‘a’ | ‘the’
Noun → ‘cake’ | ‘child’ | ‘fork’
Pre → ‘with’
Verb → ‘ate’ }

- P = R with associated probability as in the table below



Rule	*Probability*	*Rule*	*Probability*
S → NP VP	1.0	Det → ‘a’	0.5
		Det → ‘the’	0.5
NP → NP PP	0.6	Noun → ‘cake’	0.4
NP → Det Noun	0.4	Noun → ‘child’	0.3
		Noun → ‘fork’	0.3
PP → Pre NP	1.0	Pre → ‘with’	1.0
VP → Verb NP	1.0	Verb → ‘ate’	1.0

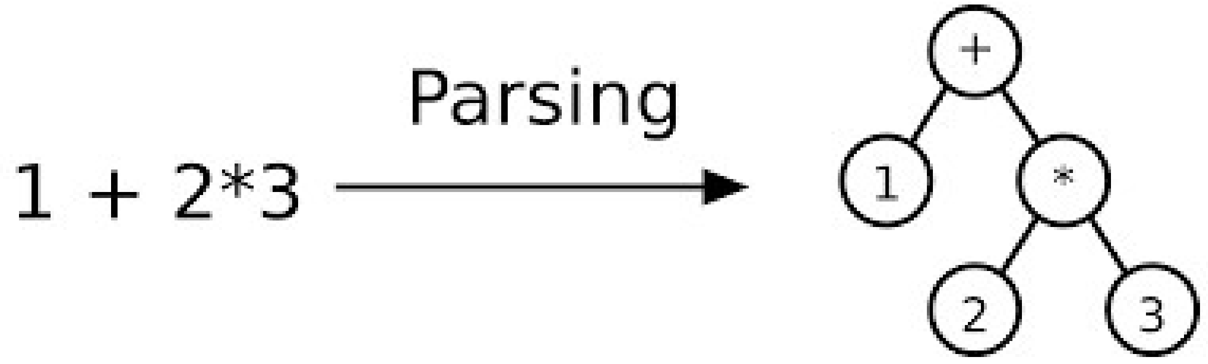
$$\sum_{A \rightarrow s \in R:\, A = NP} P(A \rightarrow s) = P(NP \rightarrow Det\,Noun) + P(NP \rightarrow NP\,PP)$$

$$= 0.4 + 0.6 = 1$$

Please observe from the table, the sum of probability values for all rules that have same left hand side is 1

Parse

- Resolve (a sentence) into its component parts and describe their syntactic roles.
- On NLP – Parsing can be visualized in the tree form

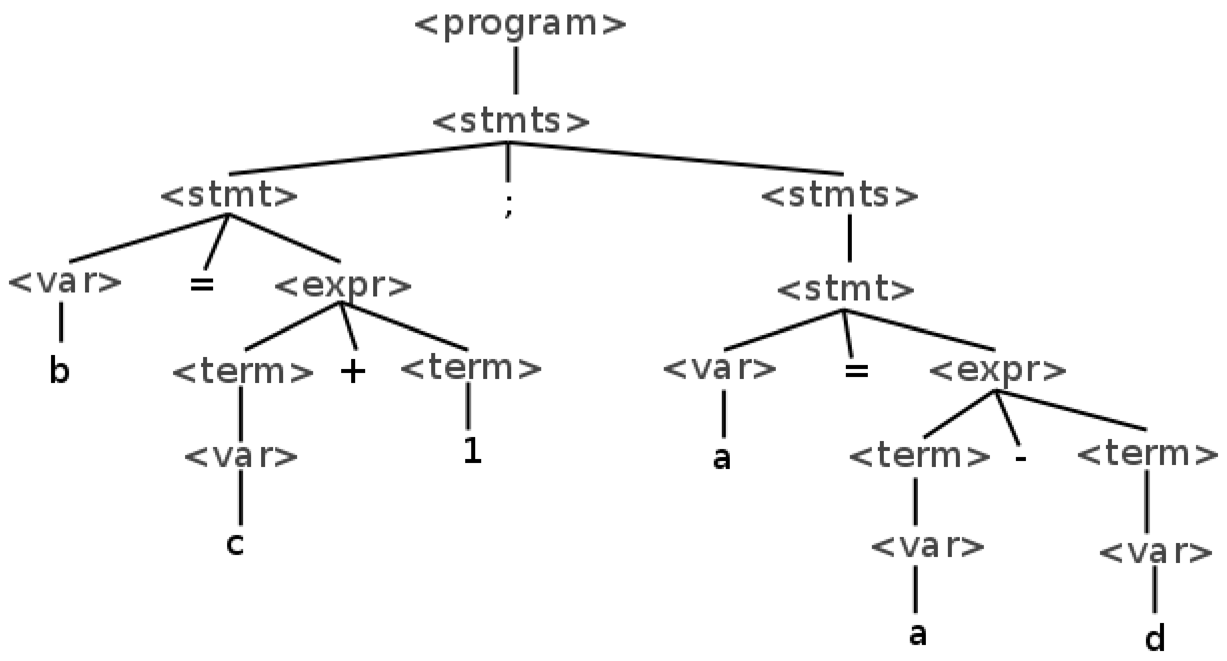


Syntax Parsing

Mostly used in programming

b = c + 1;

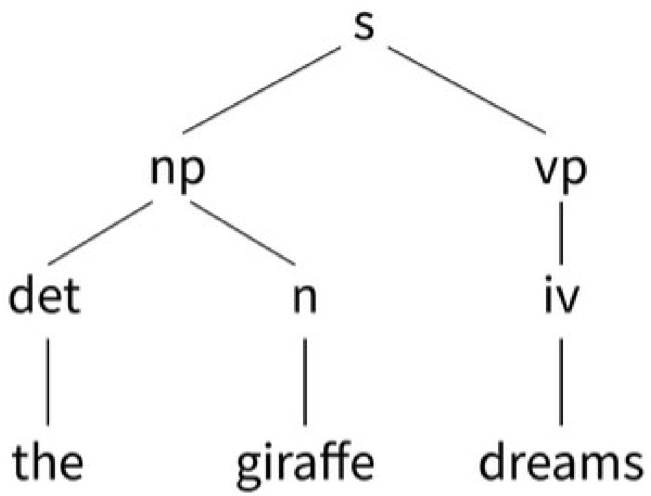
a= a - d



Parse Tree

- A parse of the sentence "the giraffe dreams" is:
  - s => np vp => det n vp => the n vp => the giraffe vp => the giraffe iv => the giraffe dreams

s	$\longrightarrow$	np vp
np	$\longrightarrow$	det n
vp	$\longrightarrow$	tv np
	$\longrightarrow$	iv
det	$\longrightarrow$	the
	$\longrightarrow$	a
	$\longrightarrow$	an
n	$\longrightarrow$	giraffe
	$\longrightarrow$	apple
iv	$\longrightarrow$	dreams
tv	$\longrightarrow$	eats
	$\longrightarrow$	dreams



Parsing

- In natural language processing, parsing is the process of analyzing a sentence to determine its grammatical structure
- There are two main approaches to parsing:
  - top-down parsing
  - bottom-up parsing

Top-down Parsing

- Top-down parsing is a parsing technique that starts with the highest level of a grammar’s production rules, and then works its way down to the lowest level.
  - It begins with the start symbol of the grammar and applies the production rules recursively to expand it into a parse tree.
  - One example of a top-down parsing algorithm is the **Recursive Descent Parsing**.

Top-down Parsing

- For example, consider the following CFG:

  S -> NP VP
  NP -> Det N
  VP -> V NP
  Det -> the | a
  N -> dog | cat | boy | girl
  V -> chased | hugged

- A top-down parser would begin with the start symbol “S” and then apply the production rule “S -> NP VP” to expand it into “NP VP”.
- The parser would then apply the production rule “NP -> Det N” to expand “NP” into “Det N”.

Buttom-up Parsing

- Bottom-up parsing is a parsing technique that starts with the sentence’s words and works its way up to the highest level of the grammar’s production rules.
- It begins with the input sentence and applies the production rules in reverse, reducing the input sentence to the start symbol of the grammar.
- One example of a bottom-up parsing algorithm is the **Shift-Reduce Parsing**.

Buttom-up Parsing

- For example, consider the following CFG:

  S -> NP VP
  NP -> Det N
  VP -> V NP
  Det -> the | a
  N -> dog | cat | boy | girl
  V -> chased | hugged

- A bottom-up parser would begin with the input sentence “the dog chased the cat” and would apply the production rules in reverse to reduce it to the start symbol “S”.
- The parser would start by matching
  - “the dog” to the “Det N” production rule,
  - “chased” to the “V” production rule, and
  - “the cat” to another “Det N” production rule.
- These reduce steps will be repeated until the input sentence is reduced to “S”, the start symbol of the grammar.

Probability of a parse tree

- Use of PCFG
  - A sentence can be parsed into more than one way
  - We can have more than one parse trees for the sentence as per the CFG due to ambiguity.

Probability of a parse tree

- Given a parse tree t,
  - with the production rules α1 → β1, α2 → β2, … , αn → βn
  - from R (ie., αi → βi ∈ R), we can find the probability of tree t using PCFG as follows;

$$P(t) = \prod_{i=1}^{n} P(\alpha_i \to \beta_i)$$

- As per the equation, the probability P(t) of parse tree is the product of probabilities of production rules in the tree t.

Probability of a parse tree

- Which is the most probable tree?
  - The probability of the parse tree t1 is greater than the probability of parse tree t2. Hence, t1 is the more probable of the two parses.

$$P(t_1) = \prod_{i=1}^{n} P(\alpha_i \rightarrow \beta_i)$$

$$= P(\mathrm{S} \rightarrow \mathrm{NP\ VP}) * P(\mathrm{NP} \rightarrow \mathrm{astronomers}) * P(\mathrm{VP} \rightarrow \mathrm{V\ NP})$$

$$* P(\mathrm{V} \rightarrow \mathrm{saw}) * P(\mathrm{NP} \rightarrow \mathrm{NP\ PP}) * P(\mathrm{NP} \rightarrow \mathrm{stars})$$

$$* P(\mathrm{PP} \rightarrow \mathrm{P\ NP}) * P(\mathrm{P} \rightarrow \mathrm{with}) * P(\mathrm{NP} \rightarrow \mathrm{ears})$$

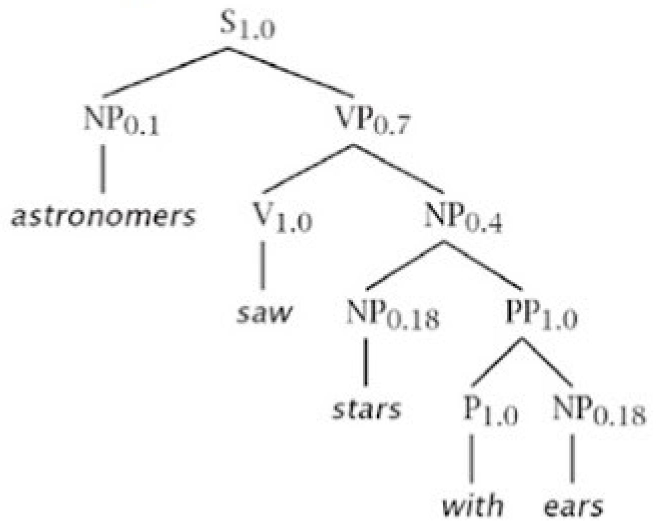
$= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18$

$= 0.0009072$

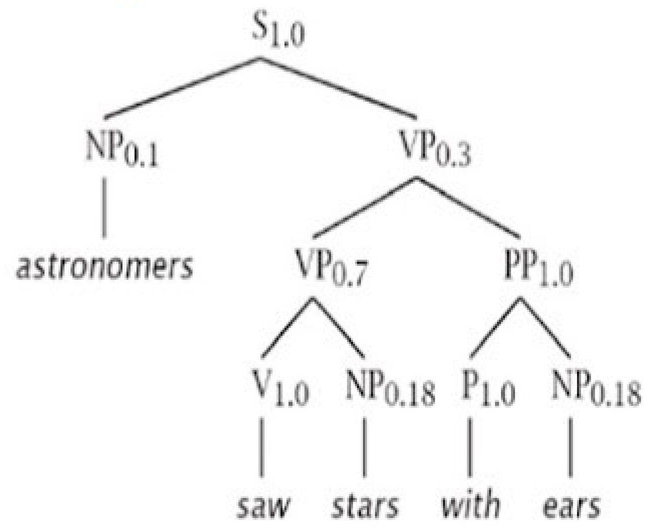
$P(t_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18$

$= 0.0006804$

tree t_1



tree t_2



Probability of a sentence

- Probability of a sentence is the sum of probabilities of all parse trees that can be derived from the sentence under PCFG

$$\sum_{i=1}^{n} P(t_i)$$

- Probability of the sentence “astronomers saw the stars with ears”

$$\sum_{i=1}^{n} P(t_i) = P(t_1) + P(t_2) = 0.0009072 + 0.0006804 = 0.001588$$

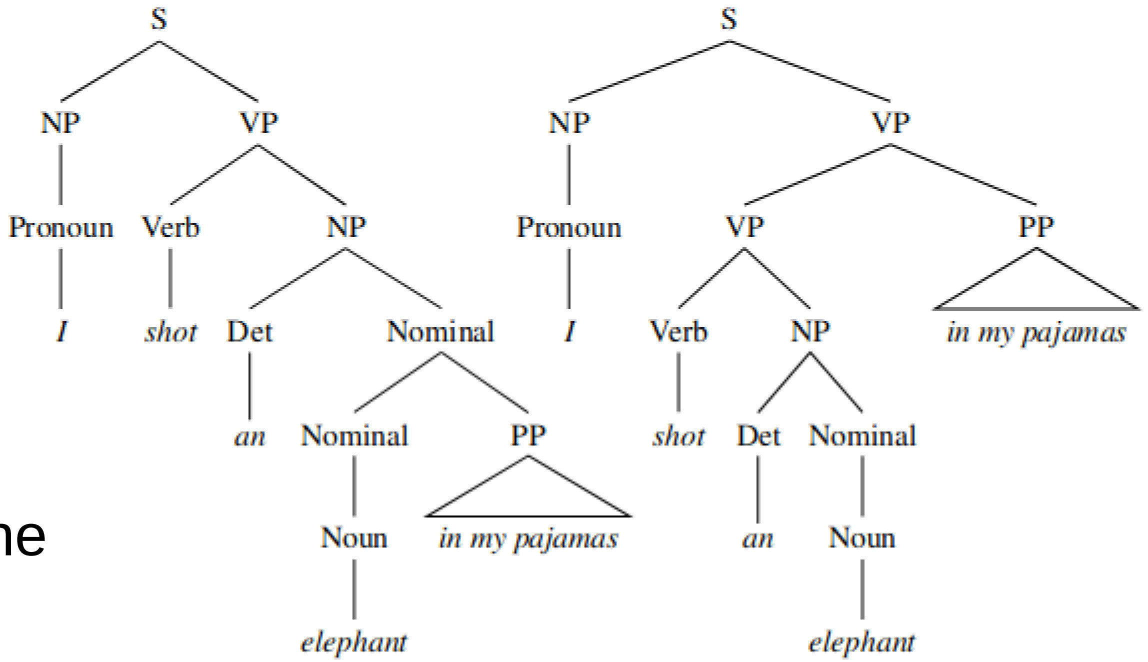
Ambiguity

- Ambiguity is the most serious problem faced by syntactic parsers
- The most common ambiguity is
  - **Structural ambiguity**

Ambiguity

- The phrase in my pajamas can be part of the NP headed by elephant or a part of the VP headed by shot

  - Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas,
  - the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas

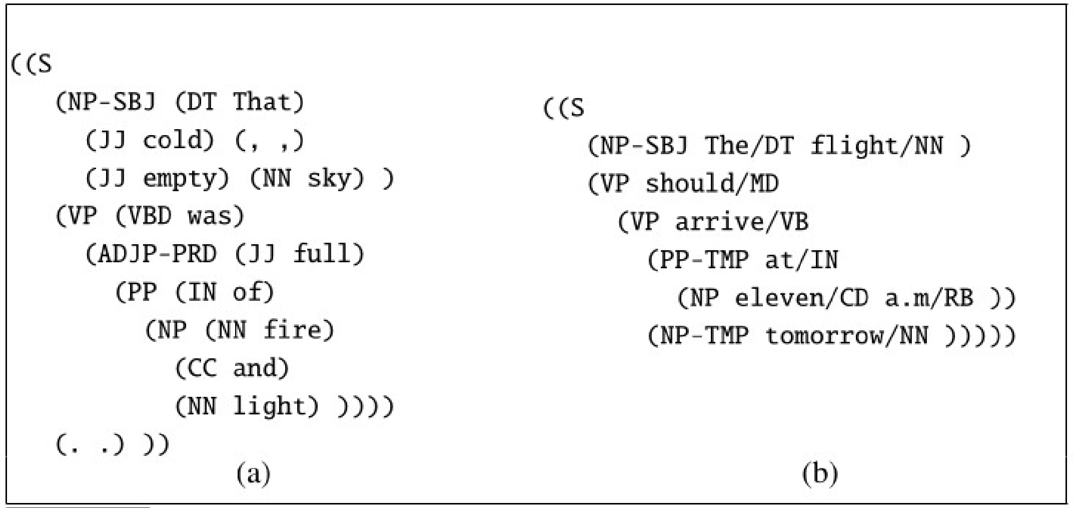


Self Study

- Chomsky Normal Form (CNF)
- Cocke–Younger–Kasami (CYK) algorithm

Treebank

- A corpus in which every sentence is annotated with a parse tree is called a treebank
- Treebanks play an important role in parsing as well as in linguistic investigations of syntactic phenomena



```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Figure 17.5 Parses from the LDC Treebank3 for (a) Brown and (b) ATIS sentences.

Reference

- **Chapter 17** - Speech and Language Processing (3rd Edition)
- Automatic Generation of Python Programs Using Context-Free Grammars: https://arxiv.org/pdf/2403.06503v1

Thank you