CL1

Syntax, Context-Free Grammar, and Constituency Parsing

Parameswari Krishnamurthy

Language Technologies Research Centre
IIIT-Hyderabad

param.krishna@iiit.ac.in

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.
- These rules determine how words and phrases are combined to form well-structured sentences.

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.
- These rules determine how words and phrases are combined to form well-structured sentences.
- **Well-formedness vs Ill-formedness:**
  - A sentence is well-formed if it follows the syntactic rules of the language.

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.
- These rules determine how words and phrases are combined to form well-structured sentences.
- **Well-formedness vs Ill-formedness:**
  - A sentence is well-formed if it follows the syntactic rules of the language.
  - **Example:**
    - *I am happy.* (Well-formed)

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.
- These rules determine how words and phrases are combined to form well-structured sentences.
- **Well-formedness vs Ill-formedness:**
  - A sentence is well-formed if it follows the syntactic rules of the language.
  - **Example:**
    - *I am happy.* (Well-formed)
    - **I saw girl the tall.* (Ill-formed – incorrect word order)

Introduction to Syntax

- Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a language.
- These rules determine how words and phrases are combined to form well-structured sentences.
- **Well-formedness vs Ill-formedness:**
  - A sentence is well-formed if it follows the syntactic rules of the language.
  - **Example:**
    - *I am happy.* (Well-formed)
    - **I saw girl the tall.* (Ill-formed – incorrect word order)
  - Syntax helps identify these grammatical errors by enforcing the structure.

Key Concepts in Syntax

- **Syntactic Categories:** These are categories of words that have similar syntactic behavior.
  - **Lexical Categories:** These include content words such as:
    - Nouns: Refers to people, objects, or ideas (e.g., *boy, school, idea*).
    - Verbs: Describes actions or states (e.g., *run, think, is*).
    - Adjectives: Provides descriptive qualities of nouns (e.g., *big, red, fast*).
    - Adverbs: Modifies verbs, adjectives, or other adverbs (e.g., *quickly, very, always*).

Key Concepts in Syntax

- **Syntactic Categories:** These are categories of words that have similar syntactic behavior.
  - **Lexical Categories:** These include content words such as:
    - Nouns: Refers to people, objects, or ideas (e.g., *boy, school, idea*).
    - Verbs: Describes actions or states (e.g., *run, think, is*).
    - Adjectives: Provides descriptive qualities of nouns (e.g., *big, red, fast*).
    - Adverbs: Modifies verbs, adjectives, or other adverbs (e.g., *quickly, very, always*).
  - **Functional Categories:** These serve grammatical purposes and include:
    - Determiners: Specifies a noun (e.g., *the, a, an*).
    - Prepositions: Shows relationships between nouns (e.g., *in, on, under*).
    - Conjunctions: Connects words, phrases, or clauses (e.g., *and, but, because*).
- **Example:**
  - *The cat sat on the mat.*
    - *The* (Determiner), *cat* (Noun), *sat* (Verb), *on* (Preposition), *the mat* (Noun Phrase).

Phrasal Categories

- **Noun Phrase (NP):**
  - A group of words that functions as a noun in a sentence.

Phrasal Categories

- **Noun Phrase (NP):**
  - A group of words that functions as a noun in a sentence.
  - **Examples:**
    - *John* (Simple NP)
    - *The little boy* (NP with Determiner and Adjective)
    - *The cat on the mat* (NP with Prepositional Phrase)
  - **Phrase structure rule:** NP $\rightarrow$ (Det) (Adj)* N (PP)

Phrasal Categories

- **Verb Phrase (VP):**
  - A phrase that includes a verb and its complements or modifiers.

Phrasal Categories

- **Verb Phrase (VP):**
  - A phrase that includes a verb and its complements or modifiers.
  - **Examples:**
    - *She runs* (Simple VP)
    - *John sang a song* (VP with Object)
    - *John sang a song in the shower* (VP with Prepositional Phrase)
  - **Phrase structure rule:** VP $\rightarrow$ V (NP)* (PP) (Adv)*

Phrasal Categories

- **Prepositional Phrase (PP):**
  - A phrase that begins with a preposition and is followed by a noun phrase.

Phrasal Categories

- **Prepositional Phrase (PP):**
  - A phrase that begins with a preposition and is followed by a noun phrase.
  - **Examples:**
    - *in the house*
    - *with a friend*
  - **Phrase structure rule:** PP $\rightarrow$ P NP

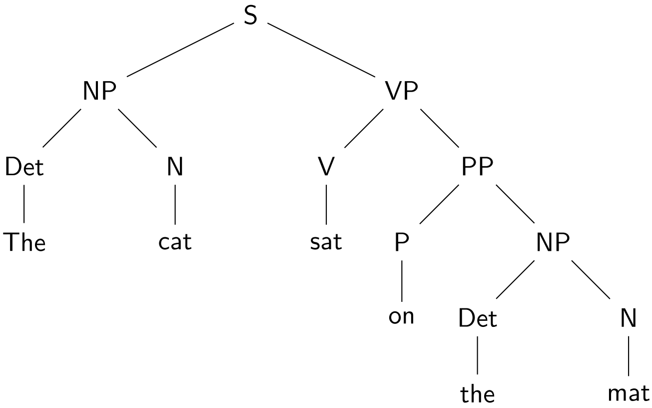
Phrase Structure Grammar (PSG)

- **Phrase Structure Grammar (PSG)** is a type of generative grammar that describes the syntactic structure of sentences using phrase structure rules.

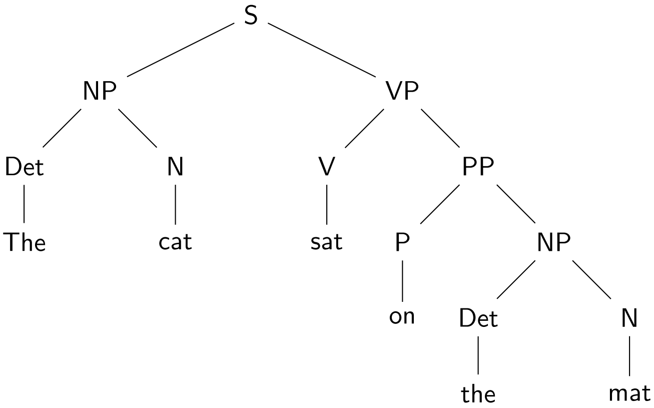
Phrase Structure Grammar (PSG)

- **Phrase Structure Grammar (PSG)** is a type of generative grammar that describes the syntactic structure of sentences using phrase structure rules.
- It breaks down a sentence into its constituent parts, or phrases, which are categorized into syntactic types (e.g., NP, VP, PP).
- **Example Sentence:** *The cat sat on the mat.*

Phrase Structure Grammar (PSG)



Phrase Structure Grammar (PSG)



- **Phrase Structure Rule:**

$$S \rightarrow NP\ VP$$
$$NP \rightarrow Det\ N$$
$$VP \rightarrow V\ PP$$
$$PP \rightarrow P\ NP$$

Sentence Structure in Syntax

- A sentence must contain both a Noun Phrase (NP) and a Verb Phrase (VP).

Sentence Structure in Syntax

- A sentence must contain both a Noun Phrase (NP) and a Verb Phrase (VP).

- The most basic sentence structure can be described using the following phrase structure rule:
  - **S → NP VP**
  - **Example:**
    - *The dog barked.*
    - NP = *The dog*
    - VP = *barked*

Sentence Structure in Syntax

- A sentence must contain both a Noun Phrase (NP) and a Verb Phrase (VP).
- The most basic sentence structure can be described using the following phrase structure rule:
- **S $\rightarrow$ NP VP**
- **Example:**
  - *The dog barked.*
  - NP = *The dog*
  - VP = *barked*
- A sentence is ill-formed if it does not follow this basic structure.
- **Example of Ill-formed Sentences:**
- **The dog.* (Missing VP)
- **Barked.* (Missing NP)

Types of Clauses and Sentences

- **Types of Clauses:**

- Independent Clause: A clause that can stand alone as a sentence (e.g., *I went to the store*).

- Dependent Clause: A clause that cannot stand alone and depends on the main clause (e.g., *If I go out*).

Types of Clauses and Sentences

- **Types of Clauses:**
- Independent Clause: A clause that can stand alone as a sentence (e.g., *I went to the store*).
- Dependent Clause: A clause that cannot stand alone and depends on the main clause (e.g., *If I go out*).
- **Types of Sentences:**
- Simple Sentence: Contains one independent clause (e.g., *I like pizza*).
- Compound Sentence: Contains two or more independent clauses (e.g., *I like pizza and he likes pasta*).
- Complex Sentence: Contains one independent clause and at least one dependent clause (e.g., *I laughed when he fell*).
- Compound-Complex Sentence: Contains at least two independent clauses and one or more dependent clauses (e.g., *I laughed when he fell, but he was fine*).

Complexities in Syntax: Ambiguities, Garden-Path, Recursiveness, Ellipsis

- **Ambiguities:**

- Structural Ambiguity: Occurs when a sentence can be parsed in more than one way (e.g., *I saw a man with a telescope*).

- Coordination Ambiguity: When the scope of conjunctions like "and" is unclear (e.g., *old men and women*).

Complexities in Syntax: Ambiguities, Garden-Path, Recursiveness, Ellipsis

- **Ambiguities:**
- Structural Ambiguity: Occurs when a sentence can be parsed in more than one way (e.g., *I saw a man with a telescope*).
- Coordination Ambiguity: When the scope of conjunctions like "and" is unclear (e.g., *old men and women*).
- **Garden-Path Sentences:**
- Sentences that lead to an incorrect initial interpretation (e.g., *The old man the boat*).

Complexities in Syntax: Ambiguities, Garden-Path, Recursiveness, Ellipsis

- **Ambiguities:**
- Structural Ambiguity: Occurs when a sentence can be parsed in more than one way (e.g., *I saw a man with a telescope*).
- Coordination Ambiguity: When the scope of conjunctions like "and" is unclear (e.g., *old men and women*).
- **Garden-Path Sentences:**
- Sentences that lead to an incorrect initial interpretation (e.g., *The old man the boat*).
- **Recursiveness:**
- A structure that repeats itself within a sentence (e.g., *This is the cat that ate the rat that ate the cheese*).

Complexities in Syntax: Ambiguities, Garden-Path, Recursiveness, Ellipsis

- **Ambiguities:**
- Structural Ambiguity: Occurs when a sentence can be parsed in more than one way (e.g., *I saw a man with a telescope*).
- Coordination Ambiguity: When the scope of conjunctions like "and" is unclear (e.g., *old men and women*).
- **Garden-Path Sentences:**
- Sentences that lead to an incorrect initial interpretation (e.g., *The old man the boat*).
- **Recursiveness:**
- A structure that repeats itself within a sentence (e.g., *This is the cat that ate the rat that ate the cheese*).
- **Ellipsis:**
- Omission of words that are understood in context (e.g., *I did it; he didn't*).

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.
- It consists of:
  - A set of non-terminal symbols (e.g., *NP*, *VP*).

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.
- It consists of:

- A set of non-terminal symbols (e.g., *NP*, *VP*).
- A set of terminal symbols (e.g., words like *flight*, *the*).

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.
- It consists of:

- A set of non-terminal symbols (e.g., *NP*, *VP*).
- A set of terminal symbols (e.g., words like *flight*, *the*).
- A set of production rules (e.g., $S \rightarrow NP\ VP$).

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.
- It consists of:

- A set of non-terminal symbols (e.g., *NP*, *VP*).
- A set of terminal symbols (e.g., words like *flight*, *the*).
- A set of production rules (e.g., $S \rightarrow NP\ VP$).
- A designated start symbol (often S for "sentence").

Introduction to Context-Free Grammar (CFG)

- Context-Free Grammar (CFG) is a formal system for describing the syntax of natural languages.
- CFGs use production rules to define how words and phrases can be grouped and ordered.
- It consists of:
  - A set of non-terminal symbols (e.g., *NP*, *VP*).
  - A set of terminal symbols (e.g., words like *flight*, *the*).
  - A set of production rules (e.g., $S \rightarrow NP\ VP$).
  - A designated start symbol (often S for "sentence").
- **Example:**
  - $S \rightarrow NP\ VP$ (A sentence consists of a noun phrase and a verb phrase)

CFG Rules and Example

- CFG rules express how symbols can be rewritten as other symbols:

CFG Rules and Example

- CFG rules express how symbols can be rewritten as other symbols:
- **NP $\rightarrow$ Det Nominal** (A noun phrase can be a determiner followed by a nominal).
- **VP $\rightarrow$ Verb NP** (A verb phrase can be a verb followed by a noun phrase).
- **Nominal $\rightarrow$ Noun | Nominal Noun** (A nominal can be a noun or a series of nouns).
- The symbols on the left of the arrow are non-terminal symbols, while those on the right can be terminals or non-terminals.

CFG Rules and Example

- CFG rules express how symbols can be rewritten as other symbols:
- **NP $\rightarrow$ Det Nominal** (A noun phrase can be a determiner followed by a nominal).
- **VP $\rightarrow$ Verb NP** (A verb phrase can be a verb followed by a noun phrase).
- **Nominal $\rightarrow$ Noun | Nominal Noun** (A nominal can be a noun or a series of nouns).
- The symbols on the left of the arrow are non-terminal symbols, while those on the right can be terminals or non-terminals.
- **Example:**
- *NP* $\rightarrow$ *Det Nominal* $\rightarrow$ *Det Noun* $\rightarrow$ *the flight*
- This means that "the flight" is a valid noun phrase.

Parsing with CFG and Parse Trees

- Parsing is the process of assigning a syntactic structure (like a tree) to a sentence according to a grammar.

Parsing with CFG and Parse Trees

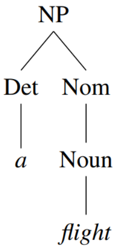
- Parsing is the process of assigning a syntactic structure (like a tree) to a sentence according to a grammar.
- **Parse Trees** are graphical representations of the syntactic structure of a sentence.
- Each node represents a symbol, and its children represent what it can be rewritten into CFG rules.

Parsing with CFG and Parse Trees

- Parsing is the process of assigning a syntactic structure (like a tree) to a sentence according to a grammar.
- **Parse Trees** are graphical representations of the syntactic structure of a sentence.
- Each node represents a symbol, and its children represent what it can be rewritten into CFG rules.

Example of Parsing: ”a flight”

- **Example:** Parsing ”a flight”
- NP $\rightarrow$ *Det Nominal*
- Det $\rightarrow$ *a*
- Nominal $\rightarrow$ *Noun*
- Noun $\rightarrow$ *flight*
- This generates the following parse tree:



$\mathcal{L}_0$ and the lexicon for $\mathcal{L}_0$

- The grammar rules we have seen so far, which we will call $\mathcal{L}_0$

$\mathcal{L}_0$ and the lexicon for $\mathcal{L}_0$

- The grammar rules we have seen so far, which we will call $\mathcal{L}_0$
- We can use the or-symbol '|' to indicate that a non-terminal has alternate possible expansions.

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

Figure 1: The lexicon for $\mathcal{L}_0$

$\mathcal{L}_0$ and the lexicon for $\mathcal{L}_0$

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

Figure 2: The grammar for $\mathcal{L}_0$, with example phrases for each rule.

$\mathcal{L}_0$ and the lexicon for $\mathcal{L}_0$

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

Figure 2: The grammar for $\mathcal{L}_0$, with example phrases for each rule.

Parse Tree

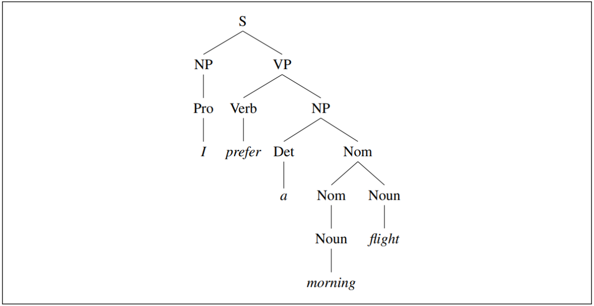


Figure 3: The parse tree for "I prefer a morning flight" according to grammar $\mathcal{L}_0$

Parse Tree

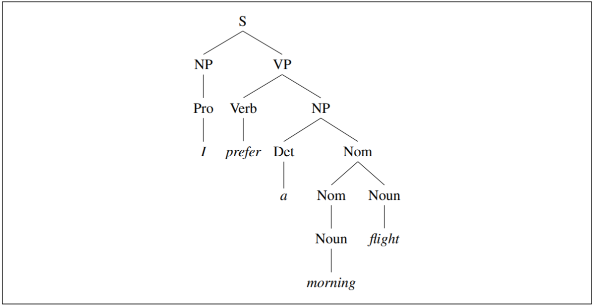


Figure 3: The parse tree for "I prefer a morning flight" according to grammar $\mathcal{L}_0$

- We can also represent a parse tree in a more compact format called bracketed notation. Here is the bracketed representation of the parse tree from the figure above:
- $[_S[_{NP}[_{Pro}I]][_{VP}[_V prefer][_{NP}[_{Det}a][_{Nom}[_N morning][_{Nom}[_N flight]]]]]]$

Treebanks

- **Definition:** A treebank is a corpus in which every sentence is annotated with a parse tree.

Treebanks

- **Definition:** A treebank is a corpus in which every sentence is annotated with a parse tree.
- **Role:** Treebanks are crucial for parsing and linguistic investigations of syntactic phenomena.

Treebanks

- **Definition:** A treebank is a corpus in which every sentence is annotated with a parse tree.
- **Role:** Treebanks are crucial for parsing and linguistic investigations of syntactic phenomena.
- **Creation:** Generally made by running a parser over sentences and then having the parses hand-corrected by human linguists.

Treebanks

- **Definition:** A treebank is a corpus in which every sentence is annotated with a parse tree.
- **Role:** Treebanks are crucial for parsing and linguistic investigations of syntactic phenomena.
- **Creation:** Generally made by running a parser over sentences and then having the parses hand-corrected by human linguists.
- **Examples:**

- Penn Treebank includes English, Arabic, and Chinese treebanks.
- Treebank representations often use LISP-style parenthesized notation or standard node-and-line trees.

Treebanks

- **Definition:** A treebank is a corpus in which every sentence is annotated with a parse tree.
- **Role:** Treebanks are crucial for parsing and linguistic investigations of syntactic phenomena.
- **Creation:** Generally made by running a parser over sentences and then having the parses hand-corrected by human linguists.
- **Examples:**
  - Penn Treebank includes English, Arabic, and Chinese treebanks.
  - Treebank representations often use LISP-style parenthesized notation or standard node-and-line trees.
- **Grammar Extraction:** Sentences in a treebank implicitly constitute a grammar of the language, allowing extraction of CFG rules.

Penn Treebank Sentences

```
((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 4: Parses from the LDC Treebank3

Penn Treebank

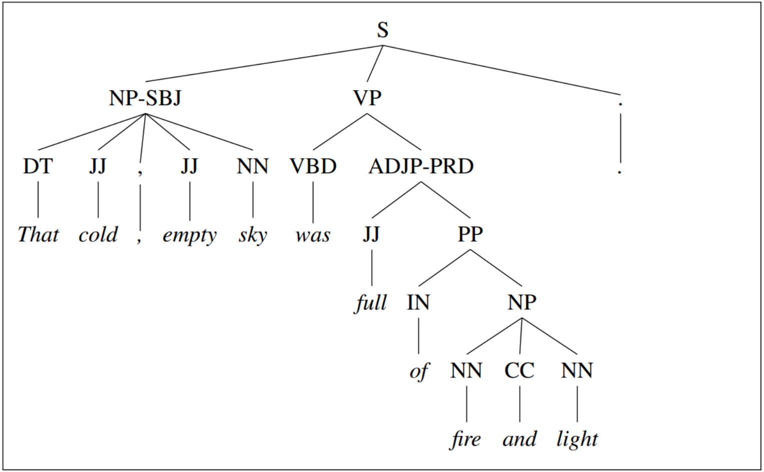


Figure 5: The tree corresponding to the Brown corpus sentence in the previous figure.

Penn Treebank

Grammar	Lexicon
$S \rightarrow NP\ VP\ .$	$DT \rightarrow the \mid that$
$S \rightarrow NP\ VP$	$JJ \rightarrow cold \mid empty \mid full$
$NP \rightarrow DT\ NN$	$NN \rightarrow sky \mid fire \mid light \mid flight \mid tomorrow$
$NP \rightarrow NN\ CC\ NN$	$CC \rightarrow and$
$NP \rightarrow DT\ JJ\ ,\ JJ\ NN$	$IN \rightarrow of \mid at$
$NP \rightarrow NN$	$CD \rightarrow eleven$
$VP \rightarrow MD\ VP$	$RB \rightarrow a.m.$
$VP \rightarrow VBD\ ADJP$	$VB \rightarrow arrive$
$VP \rightarrow MD\ VP$	$VBD \rightarrow was \mid said$
$VP \rightarrow VB\ PP\ NP$	$MD \rightarrow should \mid would$
$ADJP \rightarrow JJ\ PP$	
$PP \rightarrow IN\ NP$	
$PP \rightarrow IN\ NP\ RB$	

Figure 6: CFG grammar rules and lexicon from the treebank sentences in the previous figure.

CFG Rules from Penn Treebank

- The sentences in a treebank implicitly constitute a grammar for the language.
- From parsed sentences, CFG rules can be extracted.

CFG Rules from Penn Treebank

- The sentences in a treebank implicitly constitute a grammar for the language.
- From parsed sentences, CFG rules can be extracted.
- The grammar used in the Penn Treebank is very flat, leading to many rules.
- Example: There are approximately 4,500 different rules for expanding Verb Phrases (VP).
- These rules include variations for different combinations of prepositional phrases (PP) and verb arguments.

CFG Rules from Penn Treebank

- The sentences in a treebank implicitly constitute a grammar for the language.
- From parsed sentences, CFG rules can be extracted.
- The grammar used in the Penn Treebank is very flat, leading to many rules.
- Example: There are approximately 4,500 different rules for expanding Verb Phrases (VP).
- These rules include variations for different combinations of prepositional phrases (PP) and verb arguments.
- **Examples of CFG Rules for VP Expansion:**
- `VP → VBD PP`
- `VP → VBD PP PP`
- `VP → VBD PP PP PP`
- `VP → VBD PP PP PP PP`
- `VP → VB ADVP PP`
- `VP → VB PP ADVP`

Grammar Equivalence and Normal Form

- **Grammar Equivalence:**

- Two grammars are **strongly equivalent** if they generate the same set of strings and assign the same phrase structure to each sentence (allowing renaming of non-terminal symbols).

Grammar Equivalence and Normal Form

- **Grammar Equivalence:**
- Two grammars are **strongly equivalent** if they generate the same set of strings and assign the same phrase structure to each sentence (allowing renaming of non-terminal symbols).
- Two grammars are **weakly equivalent** if they generate the same set of strings but do not necessarily assign the same phrase structure.
- **Normal Form:**
- **Chomsky Normal Form (CNF):** A context-free grammar is in CNF if each rule is of the form $A \to BC$ (two non-terminals) or $A \to a$ (a terminal), where A, B, C are non-terminals, and a is a terminal. It cannot have empty (ϵ) productions.

Grammar Equivalence and Normal Form

- **Grammar Equivalence:**
- Two grammars are **strongly equivalent** if they generate the same set of strings and assign the same phrase structure to each sentence (allowing renaming of non-terminal symbols).
- Two grammars are **weakly equivalent** if they generate the same set of strings but do not necessarily assign the same phrase structure.
- **Normal Form:**
- **Chomsky Normal Form (CNF):** A context-free grammar is in CNF if each rule is of the form $A \rightarrow BC$ (two non-terminals) or $A \rightarrow a$ (a terminal), where A, B, C are non-terminals, and a is a terminal. It cannot have empty (ϵ) productions.
- **Binary Branching:** CNF ensures that every rule leads to binary branching, meaning that each rule breaks down into at most two components.

Grammar Equivalence and Normal Form

- **Grammar Equivalence:**
- Two grammars are **strongly equivalent** if they generate the same set of strings and assign the same phrase structure to each sentence (allowing renaming of non-terminal symbols).
- Two grammars are **weakly equivalent** if they generate the same set of strings but do not necessarily assign the same phrase structure.
- **Normal Form:**
- **Chomsky Normal Form (CNF):** A context-free grammar is in CNF if each rule is of the form $A \rightarrow BC$ (two non-terminals) or $A \rightarrow a$ (a terminal), where A, B, C are non-terminals, and a is a terminal. It cannot have empty (ϵ) productions.
- **Binary Branching:** CNF ensures that every rule leads to binary branching, meaning that each rule breaks down into at most two components.
- **Conversion:** Any context-free grammar can be converted into CNF. For example, a rule like $A \rightarrow BCD$ can be split into two binary rules:
- $A \rightarrow BX$
- $X \rightarrow CD$

Grammar Equivalence and Normal Form

- **Grammar Equivalence:**
- Two grammars are **strongly equivalent** if they generate the same set of strings and assign the same phrase structure to each sentence (allowing renaming of non-terminal symbols).
- Two grammars are **weakly equivalent** if they generate the same set of strings but do not necessarily assign the same phrase structure.
- **Normal Form:**
- **Chomsky Normal Form (CNF):** A context-free grammar is in CNF if each rule is of the form $A \to BC$ (two non-terminals) or $A \to a$ (a terminal), where A, B, C are non-terminals, and a is a terminal. It cannot have empty (ϵ) productions.
- **Binary Branching:** CNF ensures that every rule leads to binary branching, meaning that each rule breaks down into at most two components.
- **Conversion:** Any context-free grammar can be converted into CNF. For example, a rule like $A \to BCD$ can be split into two binary rules:
- $A \to BX$
- $X \to CD$
- **Advantages:** Using binary branching can sometimes produce smaller

Grammar Equivalence and Normal Form

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 7: The $\mathcal{L}_1$ miniature English grammar and lexicon.

Ambiguities in Context-Free Grammar (CFG)

- Ambiguity occurs when a sentence can be parsed in more than one way.

Ambiguities in Context-Free Grammar (CFG)

- Ambiguity occurs when a sentence can be parsed in more than one way.

- **Types of Ambiguities:**

- Structural Ambiguity: Multiple valid parse trees for a sentence (e.g., *I saw the man with the telescope*).

- Lexical Ambiguity: A word can have multiple meanings (e.g., *bank* could refer to a financial institution or the side of a river).

Ambiguities in Context-Free Grammar (CFG)

- Ambiguity occurs when a sentence can be parsed in more than one way.

- **Types of Ambiguities:**

- Structural Ambiguity: Multiple valid parse trees for a sentence (e.g., *I saw the man with the telescope*).
- Lexical Ambiguity: A word can have multiple meanings (e.g., *bank* could refer to a financial institution or the side of a river).

- **Example:** *I saw the man with the telescope*

- Ambiguous because "with the telescope" can modify either "saw" or "the man."
- Two possible parse trees:
- *I [saw the man] [with the telescope]* (I used the telescope to see the man).
- *I saw [the man with the telescope]* (The man I saw had a telescope).

Ambiguity:
I saw the man [with the telescope] vs. I saw [the man with the telescope]

Introduction to Constituency Parsing

- Constituency Parsing is the process of analyzing the syntactic structure of a sentence by breaking it down into its constituent parts (phrases or subphrases).

Introduction to Constituency Parsing

- Constituency Parsing is the process of analyzing the syntactic structure of a sentence by breaking it down into its constituent parts (phrases or subphrases).
- Constituents behave as single units within a sentence, and parsing aims to identify these groupings.
- The most common constituents are noun phrases, verb phrases, and prepositional phrases.

Introduction to Constituency Parsing

- Constituency Parsing is the process of analyzing the syntactic structure of a sentence by breaking it down into its constituent parts (phrases or subphrases).
- Constituents behave as single units within a sentence, and parsing aims to identify these groupings.
- The most common constituents are noun phrases, verb phrases, and prepositional phrases.
- **Example:** *The cat sat on the mat.*
- Noun Phrase (NP) = *The cat*, Verb Phrase (VP) = *sat on the mat*

Constituents in Natural Language

- Constituents are groups of words that function as a single unit in a sentence.

Constituents in Natural Language

- Constituents are groups of words that function as a single unit in a sentence.
- Evidence for constituents can be found through tests such as substitution, movement, and coordination.

Constituents in Natural Language

- Constituents are groups of words that function as a single unit in a sentence.
- Evidence for constituents can be found through tests such as substitution, movement, and coordination.
- **Example:** *Harry the Horse arrived.*
- The noun phrase *Harry the Horse* can be substituted with *he*, showing that it forms a constituent.

Introduction to CKY Parsing

- Cocke-Kasami-Younger (CKY) algorithm, the most widely used dynamic-programming based approach to parsing. is a dynamic programming algorithm used for parsing sentences using context-free grammars in Chomsky Normal Form (CNF).

- The algorithm efficiently handles parsing by systematically exploring all possible substructures of a sentence.

- CKY parsing is particularly useful for identifying all possible parse trees for a given sentence, which is crucial for handling ambiguity.

Requirements for CKY Parsing

- CKY Parsing requires the grammar to be in Chomsky Normal Form (CNF).
- **Chomsky Normal Form:** A rule in CNF is of the form $A \rightarrow BC$ or $A \rightarrow a$, where:
- A, B, and C are non-terminals.
- a is a terminal (word).
- Conversion to CNF involves eliminating epsilon rules, unit productions, and ensuring all productions are binary branching.

Illustration

$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Figure 8: $\mathcal{L}_1$ Grammar and its conversion to CNF.

CKY Parsing Algorithm

- The CKY algorithm fills a parse table by examining all possible substrings of the input sentence.
- For each substring, the algorithm checks all possible ways to split it into two parts and applies grammar rules to form constituents.
- The table is filled in a bottom-up manner, starting with the smallest substrings and working up to the full sentence.
- The presence of a start symbol (S) spanning the entire input in the parse table indicates that the sentence is grammatically correct.

Example of CKY Parsing

- **Sentence:** *Book the flight through Houston.*

Example of CKY Parsing

- **Sentence:** *Book the flight through Houston.*

- Parse table initialization: Start by filling cells corresponding to single words using lexical rules.

- Iteratively fill the table by combining entries based on binary rules from CNF:
  - Example: If NP and VP are found, check rules that can combine them into larger constituents.

- The goal is to fill the top-right cell with the start symbol S.

CKY Algorithm Workflow

Step-by-Step Process:

1. Initialization: Fill the diagonal of the CKY table with parts of speech for each word in the sentence using lexical rules.
2. Filling the Table:
   - For each cell, consider all possible ways to split the substring into two parts.
   - Check grammar rules for combining the two parts and add any valid constituents to the cell.
3. Final Step: If the top-right cell contains the start symbol 'S', the sentence is valid.

CKY Algorithm Workflow

Step-by-Step Process:

1. Initialization: Fill the diagonal of the CKY table with parts of speech for each word in the sentence using lexical rules.
2. Filling the Table:

- For each cell, consider all possible ways to split the substring into two parts.
- Check grammar rules for combining the two parts and add any valid constituents to the cell.

3. Final Step: If the top-right cell contains the start symbol ‘S‘, the sentence is valid.

Key Advantages of CKY Parsing:

- Handles ambiguous sentences by finding all possible parses.
- Efficient dynamic programming approach.
- Suitable for sentences parsed using CNF grammars.

CKY Parsing Algorithm (Illustration)

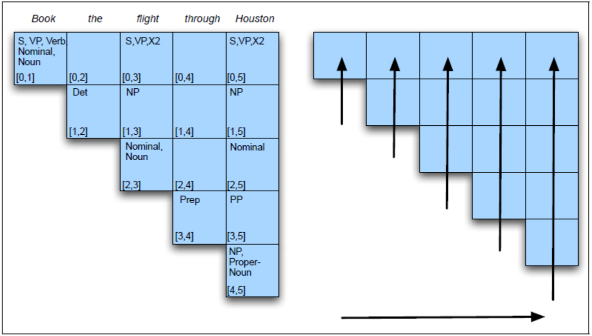
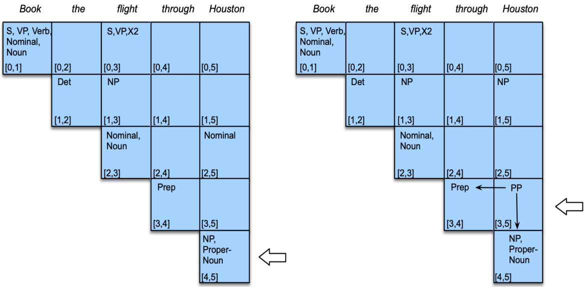
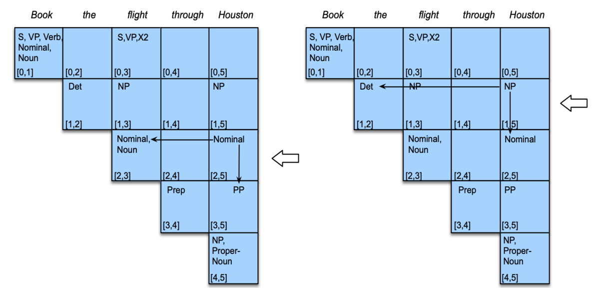


Figure 9: Completed parse table for Book the flight through Houston.

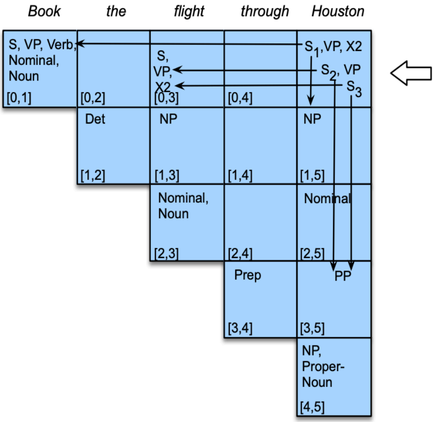
CKY in Practice



CKY in Practice



CKY in Practice



Practical Considerations of CKY Parsing

- Efficiency: CKY is efficient for parsing because it avoids redundant computations by storing results of subproblems.

Practical Considerations of CKY Parsing

- Efficiency: CKY is efficient for parsing because it avoids redundant computations by storing results of subproblems.

- Ambiguity Handling: CKY can handle multiple parse trees by maintaining all possible constituents in the parse table.

Practical Considerations of CKY Parsing

- Efficiency: CKY is efficient for parsing because it avoids redundant computations by storing results of subproblems.

- Ambiguity Handling: CKY can handle multiple parse trees by maintaining all possible constituents in the parse table.

- Limitations: Requires conversion to CNF, which can lead to an increase in grammar size and complexity.

Practical Considerations of CKY Parsing

- Efficiency: CKY is efficient for parsing because it avoids redundant computations by storing results of subproblems.
- Ambiguity Handling: CKY can handle multiple parse trees by maintaining all possible constituents in the parse table.
- Limitations: Requires conversion to CNF, which can lead to an increase in grammar size and complexity.
- Applications: Used in natural language processing tasks like syntax checking, language understanding, and as a foundation for more advanced parsing algorithms.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.
- **Phrasal Categories** (such as NP, VP, PP) represent groups of words functioning as a single syntactic unit.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.
- **Phrasal Categories** (such as NP, VP, PP) represent groups of words functioning as a single syntactic unit.
- **Constituency Parsing** breaks down sentences into their constituents, identifying how words group together.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.
- **Phrasal Categories** (such as NP, VP, PP) represent groups of words functioning as a single syntactic unit.
- **Constituency Parsing** breaks down sentences into their constituents, identifying how words group together.
- **Context-Free Grammar** (CFG) provides a formal system to describe sentence structure using production rules.

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.
- **Phrasal Categories** (such as NP, VP, PP) represent groups of words functioning as a single syntactic unit.
- **Constituency Parsing** breaks down sentences into their constituents, identifying how words group together.
- **Context-Free Grammar** (CFG) provides a formal system to describe sentence structure using production rules.
- **CKY Parsing** is an efficient algorithm for parsing sentences using a grammar in Chomsky Normal Form (CNF).

Conclusion

- **Syntax** defines the structure of sentences, specifying how words combine to form grammatically correct sentences.
- **Syntactic Categories** (like Noun, Verb, Adjective) help categorize words based on their grammatical roles.
- **Phrasal Categories** (such as NP, VP, PP) represent groups of words functioning as a single syntactic unit.
- **Constituency Parsing** breaks down sentences into their constituents, identifying how words group together.
- **Context-Free Grammar** (CFG) provides a formal system to describe sentence structure using production rules.
- **CKY Parsing** is an efficient algorithm for parsing sentences using a grammar in Chomsky Normal Form (CNF).
- **Treebanks** provide syntactic annotations for sentences, allowing extraction of rules and aiding in parsing tasks.