CS310 Natural Language Processing
自然语言处理
Lecture 05 - Context-Free Grammars and Constituency Parsing

Instructor: Yang Xu

主讲人：徐炀

xuyang@sustech.edu.cn

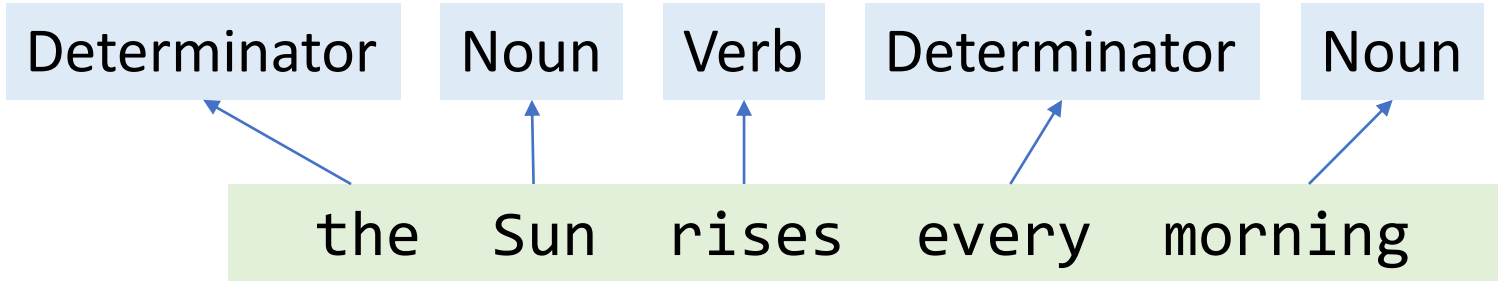
Overview

- Context Free Grammars (CFG)
- Constituency Parsing
- Neural Constituency Parsing

Grammar ≈ Syntax

- Greek: *syntaxis*, meaning "setting out together"
- The way words are arranged together

- Syntax: *n.*句法，句法规则;
- Part-of-speech is already a kind of syntactic notions



Constituency

- Groups of words can behave as **single units**, or **constituents** (*n.* 成分)
- Ex. noun phrases (***NP***, sequence of words surrounding at least one noun)

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- Why they form constituents? (the following NPs all appear before verbs)

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

Invalid sentences:

*from *arrive*...	*as *attracts*...
*the *is*...	*spot *sat*...

Evidence for constituency (more examples)

- Preposed (前置) or postposed (后置)

"On February 2nd" is a prepositional phrase (***PP***)

On February 2nd, I'd like to fly from Shanghai to Osaka.

I'd like to fly on February 2nd from Shanghai to Osaka.

I'd like to fly from Shanghai to Osaka on February 2nd.

介词短语（PP）的完整

* On February, I'd like to fly 2nd from Shanghai to Osaka.

* I'd like to fly on February 2nd from Shanghai to Osaka.

I'd like to fly on February from Shanghai to Osaka 2nd.

Terminology

- Phrase: 词组
- Noun Phrase: 名词词组； Prepositional Phrase: 介词词组
- Proper Noun: 专有名词
- Nominal: 名词性的词

How do we use constituency to describe grammar?

- How do we describe rules such as "*Noun Phrases can occur before verbs*", "*Prepositional Phrases can occur after verbs*" and so on?
- Need a more formal way!
- -- Context-Free Grammar

Context-Free Grammars (CFG)

Wundt, German psychologist
Chomsky, American linguist
Backus & Naur, American computer scientists (FORTRAN, ALGO)

- Also called Phrase-Structure Grammar
- Wilhelm Wundt (1900), Chomsky (1956), Backus (1959), Naur (1963)
- A set of **rules** or **productions** that express the ways symbols can be grouped and ordered together.

Ex.

限定词+名词性

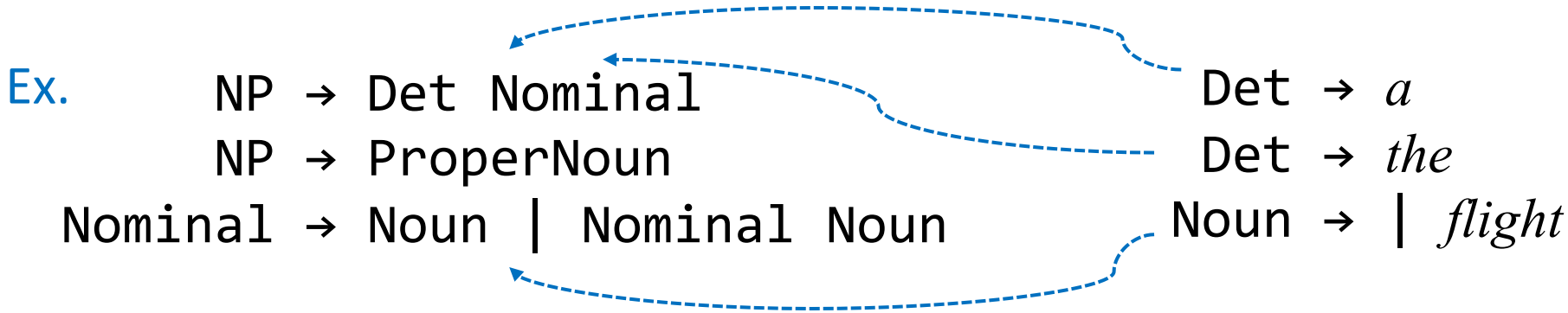
```
NP → Det Nominal
NP → ProperNoun
Nominal → Noun | Nominal Noun
```

Nominal可以是一个名词，或者多个名词组合（递归规则）

A **NP** (**noun phrase**) can be composed of
- either a *ProperNoun* (专有名词)
- or a determiner (*Det* 冠词) followed by a *Nominal* (名词性词)

A Nominal can be one or more *Nouns* (名词)

Rules can be hierarchically embedded



Two classes of symbols: **terminal** and **non-terminal**

终结符（terminal symbols）是语言中的实际单词，如“a”、“the”、“flight”；终结符是推导的最终结果，不能再被规则展开

- **terminal:** symbols that are actual words in language (“a”, “the”, …)
- **non-terminal**: symbols that express abstractions over terminals

抽象的语法类别；可以通过规则
展开成其他符号

CFG as a device for generating sentences

(生成)

Rules:

名词性短语

① NP → Det Nominal

② NP → ProperNoun

③ Nominal → Noun | Nominal Noun

名词性成分

④ Det → *a*

⑤ Det → *the*

⑥ Noun → | *flight*

Start from *NP*

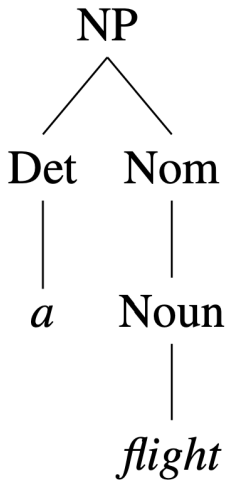
Use rule ① to rewrite *NP* as: *Det Nominal*

Use rule ③ to rewrite *Nominal* as: *Noun*

Use rule ④ to rewrite *Det* as: *a*

Use rule ⑥ to rewrite *Noun* as: *flight*

Result: **parse tree**

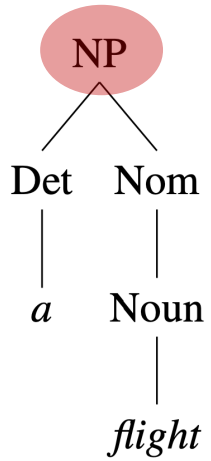


CFG: More terms and rules

A grammar must have one designated **start symbol**, often called ***S***.

Common rules:

NP **dominates** all nodes in the tree



A sentence can consist of a noun phrase followed by a verb phrase

`S → NP VP` *I prefer a morning flight*

A VP can consist of a verb followed by assorted things

`VP → Verb NP` *prefer a morning flight*

Or VP may have a verb followed by a prepositional phrase

`VP → Verb PP` *leaving on Thursday*

A PP generally has a preposition followed by a noun phrase

`PP → Preposition NP` *from Los Angeles*

CFG: Examples from AITS corpus

A sample lexicon

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*

A sample of grammar rules $\mathcal{L}_0$

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

Air Travel Information System (ATIS) pilot corpus
https://aclanthology.org/H90-1021/

Use ATIS grammar to generate "AITS-language"

S → NP VP Start with S, expand it with NP VP

Choose a random expansion of NP: *I*

NP →	*Pronoun*	I
|	*Proper-Noun*	Los Angeles
|	*Det Nominal*	a + flight

Choose a random expansion of VP: *Verb NP*

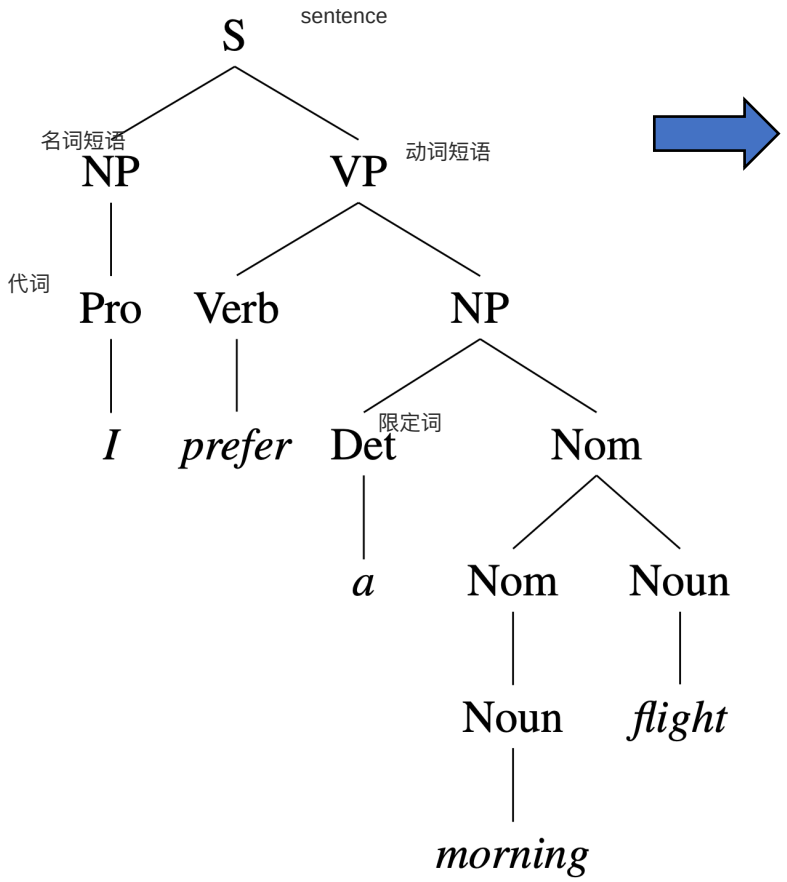
VP →	*Verb*	do
|	*Verb NP*	want + a flight
|	*Verb NP PP*	leave + Boston + in the morning
|	*Verb PP*	leaving + on Thursday

Verb NP

- want + a flight
- want + a morning flight
- *prefer + a morning flight*
- ……

I prefer a morning flight

CFG: Bracketed Notation




```
(S (NP (PRO I)
   (VP (V prefer)
       (NP (Det a) (Nom (N morning) (N flight))))))
```

Sentences that can be derived by a grammar are called **grammatical;** those cannot be derived are referred to as **ungrammatical**.

This distinguishment is vague, and only a very *simplified* model of how natural language really work.

Formal definition of CFG

- A context-free grammar G is defined by 4 parameters:
- ***N, Σ, R, S***

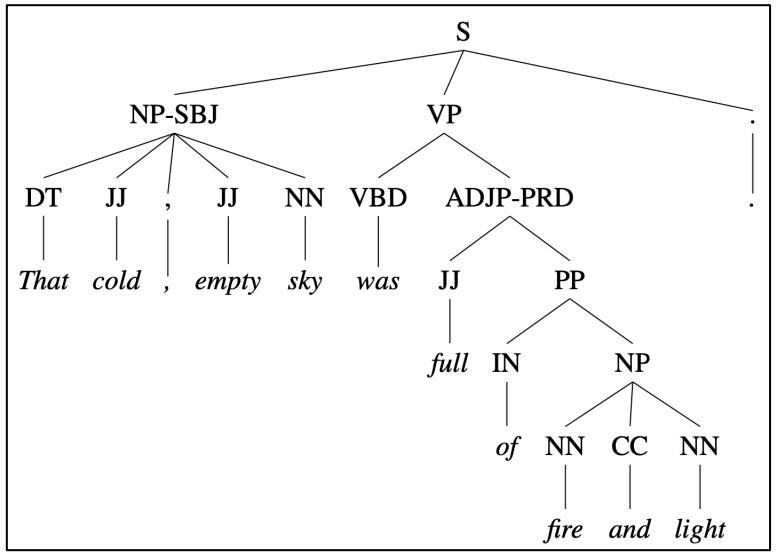
  - ***N***: a set of **non-terminal** symbols
  - ***Σ***: a set of **terminal** symbols
  - ***R***: a set of rules (or productions), each of the form A $\rightarrow \beta$ where A $\in N$, and $\beta \in \Sigma \cup N$
  - ***S***: a start symbol

Syntactic parsing: the problem of mapping from a string of words to its parse tree

Treebank 句法树库

- A corpus in which every sentence is annotated with a parse tree
- Penn Treebank project: English, Arabic, Chinese.

```
((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) ))))
   (. .) ))
```



First parsed by a parser (algorithm), then hand-corrected by human linguists

The grammar is very flat
Ex, approximately 4,500 different rules for expanding VPs

```
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
VP → ADVP VB PP
……
```

Chomsky normal form (CNF)

限制文法的产生式（production rules）形式，使其更易于处理

Noam Chomsky, "the father of modern linguistics"

- It is sometimes useful to have a **normal form** for grammars
- A grammar is in Chomsky normal form (CNF) if each production is

either $A \rightarrow B\ C$ or $A \rightarrow a$

两个非终结符　　终止符

- That is, the right-hand side has ***two*** **non-terminals** or ***one*** **terminal**
- Any context-free grammar can be converted to a (weakly) equivalent CNF

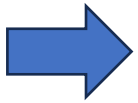
Ex. $A \rightarrow B\ C\ D$

$$A \rightarrow B\ X$$
$$X \rightarrow C\ D$$

Why CNF?

- Result in **binary branching** parse trees
- Produce smaller grammars

```
VP → VBD NP PP
VP → VBD NP PP PP
VP → VBD NP PP PP PP
VP → VBD NP PP PP PP PP
…
```

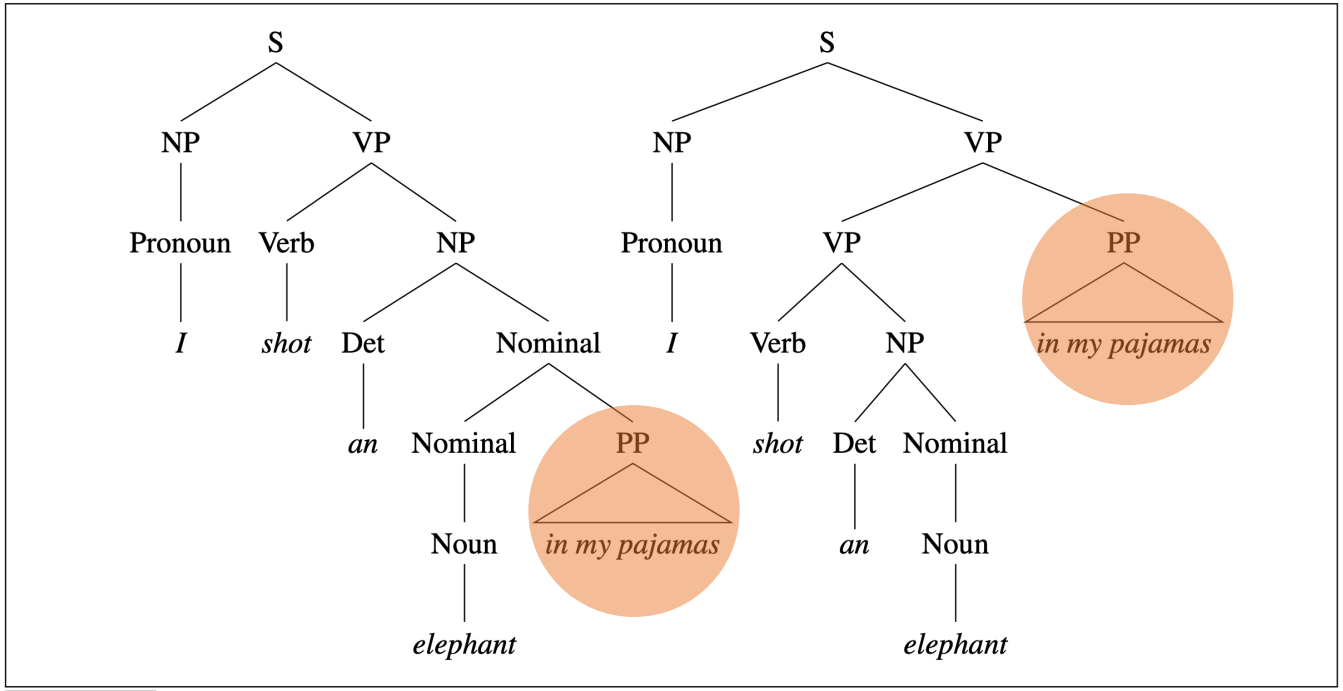


```
VP → VBD NP PP
VP → VP PP
```

Problem in syntactic parsing: Ambiguity

- Structural ambiguity occurs when the grammar can assign more than one parse to a sentence.

I shot an elephant in my pajamas



Left: elephant is in pajamas

Right: the action of shooting is done by me in my pajamas

PP-attachment ambiguity: the preposition phrase can be attached either as part of the NP or as part of the VP.

Need a way to deal with these syntactic disambiguation

Overview

- Context Free Grammars (CFG)
- **Constituency Parsing** 成分句法分析
- Neural Constituency Parsing

Parsing: A Dynamic Programming Approach

- **Dynamic programming** provides a powerful framework for addressing the problems caused by **ambiguity** in grammars
  - A **dynamic programming** approach fills in a table of solutions to subproblems
  - The complete table has the solution to all the subproblems
  - In the case of parsing: the subproblems represent parse trees for all the constituents detected in the input

子问题指的是输入句子中各个片段的可能解析树（parse trees）。这些解析树对应于检测到的语法成分（constituents），例如名词短语 (NP)、动词短语 (VP) 等

- **Basic assumption**: the **context-free nature** of grammar rules -- once a constituent has been discovered in a segment of input, we can store it and make available for use in any subsequent derivation that might require it
- CKY algorithm (Cocke-Kasami-Younger, 1961)

一旦在一个输入片段中发现了一个语法成分（例如 NP 或 VP），就可以将其存储并在后续的推导（derivation）中重用，而不依赖于上下文。

CKY 算法从输入句子的单词开始，逐步组合更长的片段，填充一个三角形表（通常称为解析表），记录每个子字符串可能的语法类别和解析树。最终，表的左上角单元格会包含整个句子的所有可能解析。

CKY Algo. Preprocessing: Conversion to CNF

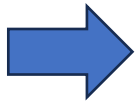
- CKY algorithm requires grammars to first be in Chomsky Normal Form (CNF)
- I.e., the right-hand side of each rule must expand either to **two non-terminals** or to a **single terminal**

either $A \rightarrow B\ C$ or $A \rightarrow a$

- The remedy for rules that mix terminals and non-terminals is to introduce a new **dummy non-terminal**

哑非终结符 是为特定终结符或子结构引入的临时非终结符，它不改变文法的生成能力（即弱等价），但使其形式符合 CNF

INF-VP → to VP



INF-VP → TO VP
TO → to

CKY Algo. Preprocessing: Conversion to CNF

- For rules with right hand side longer than 2, introduce new non-terminals:

$A \rightarrow B\ C\ \gamma$ ⇒ $A \rightarrow X1\ \gamma$, $X1 \rightarrow B\ C$

Ex.

$S \rightarrow Aux\ NP\ VP$ ⇒ $S \rightarrow X1\ VP$, $X1 \rightarrow Aux\ NP$

- For rules with a single non-terminal on the right (unit productions), rewrite the right hand side with non-unit production rules that they ultimately lead to

对于右边为单个非终结符的规则（单位产生式），用最终导向的非单位产生式规则重写右边。也就是说，将单位产生式替换为它所指向的非终结符的最终终结符扩展。

$Nominal \rightarrow Noun$, $Noun \rightarrow book \mid flight$

消除间接推导，直接从nomial到noun的延伸

$Nominal \rightarrow book \mid flight$

将 Nominal → Noun 替换为 Noun 的所有可能扩展，即 book 和 flight

Grammar in CNF: Example

Grammar $\mathcal{L}_1$ (with adds to $\mathcal{L}_0$)

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

Grammar $\mathcal{L}_1$ in CNF

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

CKY Recognition

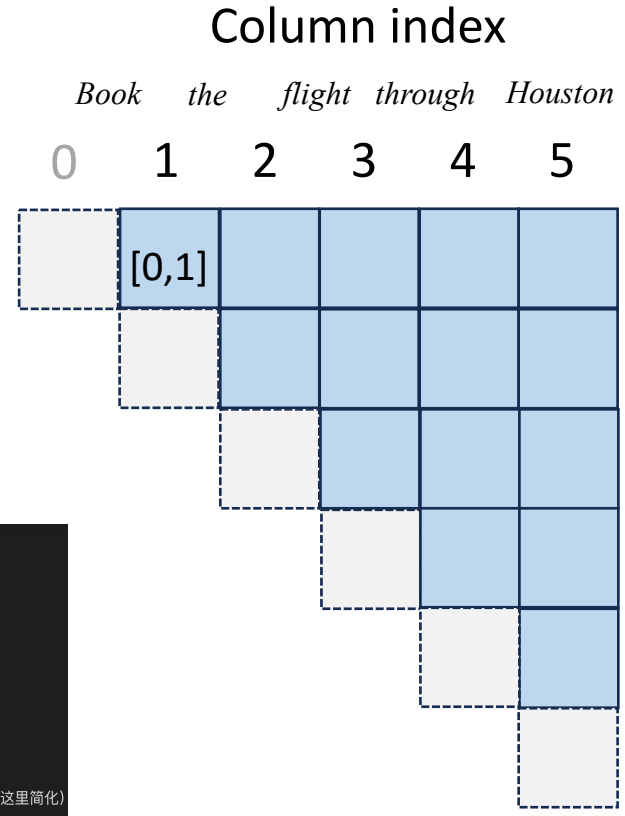
动态规划方法，用于判断一个句子是否符合给定的上下文无关文法（CFG），并识别其可能的语法结

- For a sentence of length n, work with an $(n+1)\times(n+1)$ **matrix** m, using the upper triangular portion ⇒ **parse table**
- Cell $m[i,j]$ contains the set of non-terminals representing all constituents that span position i through j.

Index: 0 *Book* 1 *the* 2 *flight* 3 *through* 4 *Houston* 5

Ex. Cell $m[0,3]$ contains the set of non-terminals for all constituents spanning 0 to 3

Task: Fill in the parse table correctly in a bottom-up way



- 步骤 1：初始化
  - $m[0,1]=\{Noun\}$（"Book"）
  - $m[1,2]=\{Det\}$（"the"）
  - $m[2,3]=\{Noun\}$（"flight"）
  - $m[3,4]=\{Prep\}$（"through"）
  - $m[4,5]=\{Noun\}$（"Houston"）
- 步骤 2：组合
  - $m[0,2]=\{NP\}$（因为 Det Noun → NP，结合 "the" 和 "flight"）
  - $m[2,5]=\{NP\}$（因为 Prep NP → PP，然后 PP 可进一步扩展，但这里简化）
  - $m[0,3]$ 需要检查所有可能组合，例如 NP + Noun 或其他规则。
- 步骤 3：扩展
  - $m[0,5]$ 最终可能包含句子的起始符号 S，如果整个句子符合文法。

CKY Recognition: Task breakdown

非终止符一定能拆成两个孩子，那么就一定能找到这个k来拆分

- Because grammar is in **CNF** ⇒ each non-terminal entry has exactly **two children**.
- For each constituent represented by entry $[i, j]$, there must be a position in the input, k, where it can be split into two parts $(i < k < j)$
- The first component $[i, k]$ is to the left of $[i, j]$, at the same row i
- The second component $[k, j]$ is beneath $[i, j]$, at the same column j
- Then, $[i, j]$ is the **combination** of $[i, k]$ and $[k, j]$ for all possible k

The smallest problems are $[j-1, j]$, i.e., spans of single word

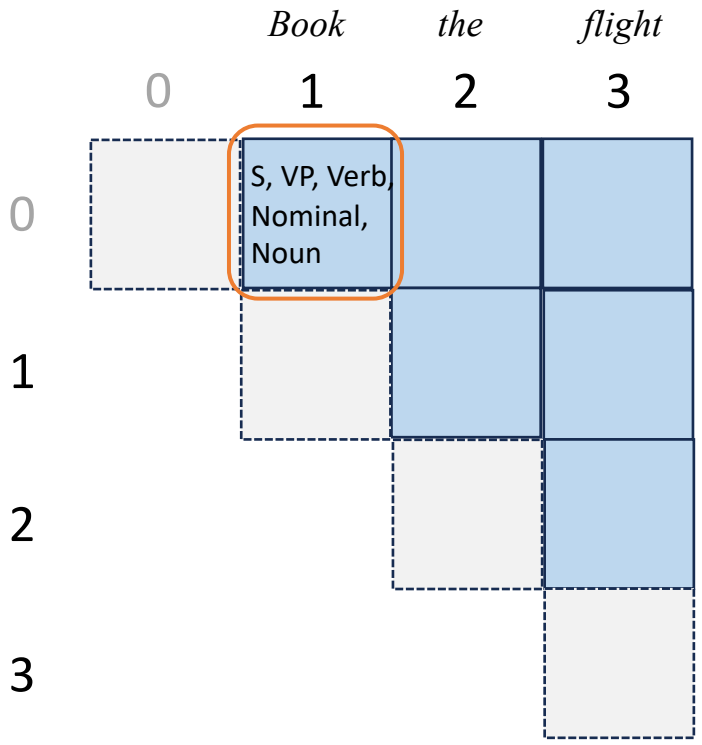
In order to solve the **problem**, we need to solve **smaller problems** first

- Very typical dynamic programming approach - *optimal substructure*

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3

Solve span [0, 1]



5 production rules match with ***"Book"***

Store them in **cell [0,1]**

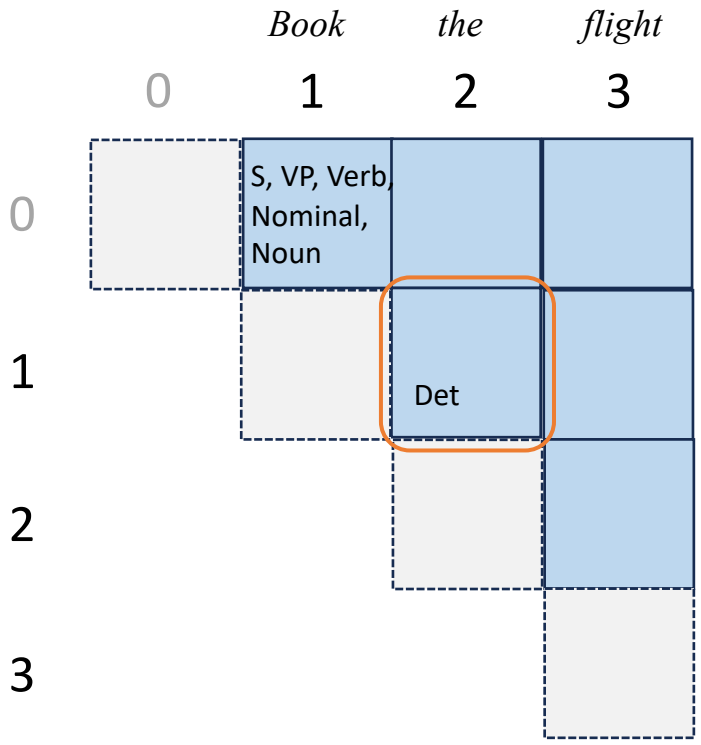
$m[0,1]$
$= \{S, Nominal, VP, Noun, Verb\}$

$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3

Solve span [1, 2]



1 production rule matches with ***"the"***

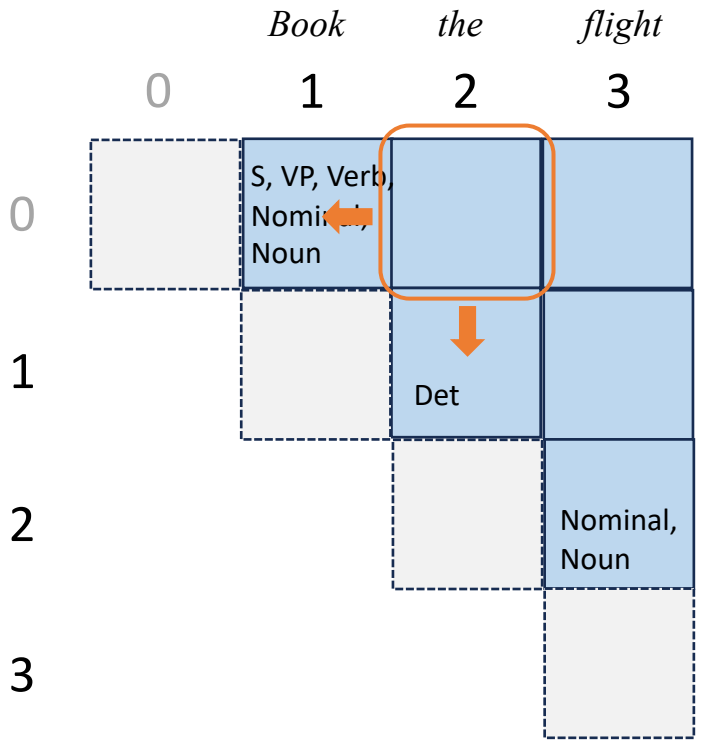
Store it in **cell [1,2]**

$m[1,2] = \{Det\}$

$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3



Solve span [0, 2]

Look up the grammar table to find a rule $A \rightarrow B\ C$ such that $B \in m[0,1]$ and $C \in m[1,2]$

No rule is found, so let $m[0,2] = \emptyset$

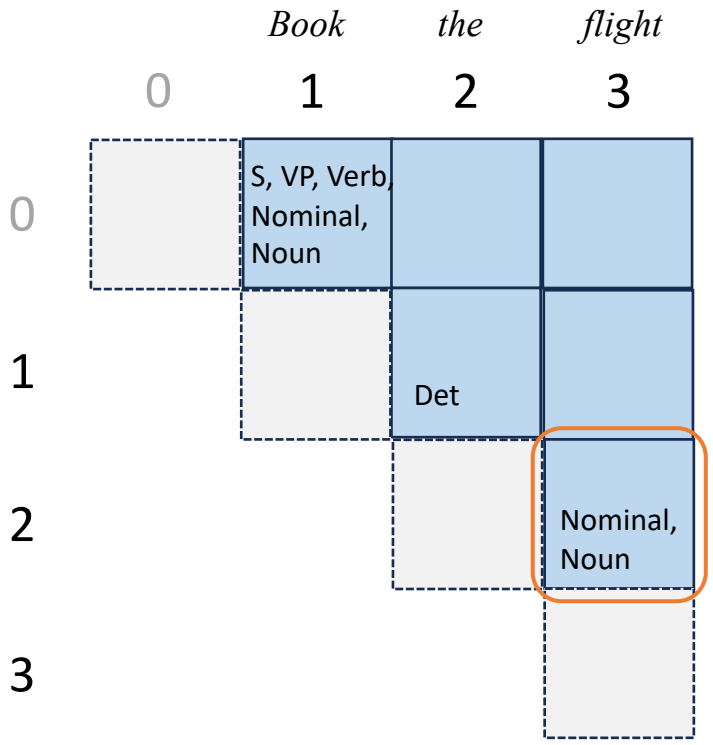
It means the phrase "*Book the*" is incomplete in grammar

$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3

Solve span [2, 3]



1 production rule matches with ***"flight"***

Store it in **cell [2,3]**

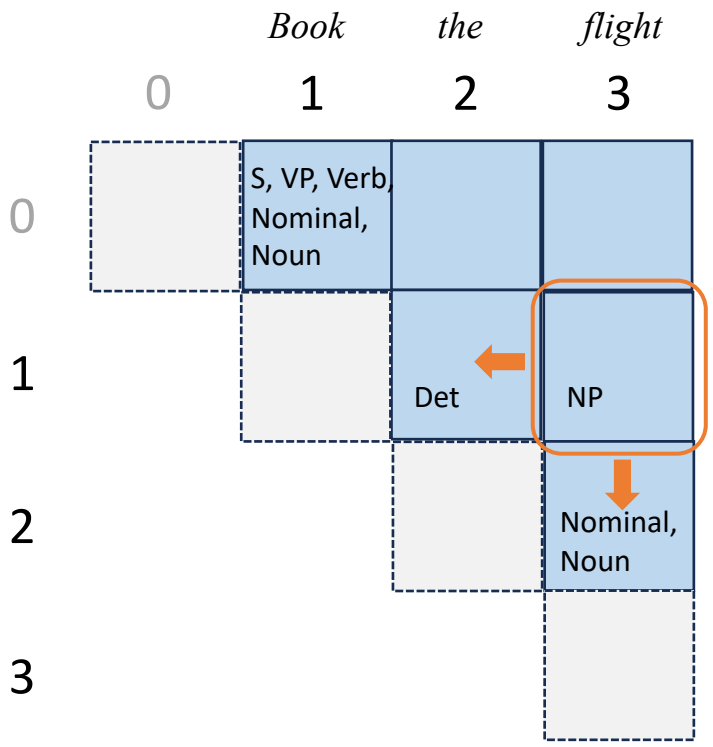
$m[2,3]$
$= \{Nominal, Noun\}$

$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3

Solve span [1, 3]



Look up the grammar table to find a rule $A \rightarrow B\ C$ such that
$B \in m[1,2]$ and $C \in m[2,3]$

Found one rule!

$$NP \rightarrow Det\ Nominal$$

Store it:

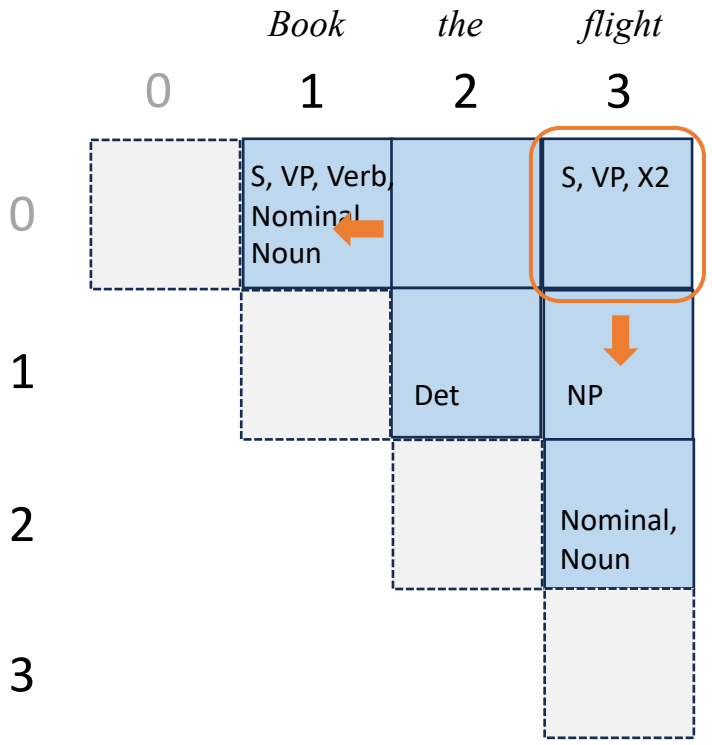
$$m[2,3] = \{NP\}$$

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

$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$

CKY Recognition: Example

Index: 0 *Book* 1 *the* 2 *flight* 3



Solve span [0, 3]

There are two ways to split:

$[0,3] = [0,1] + [1,3]$,

$[0,3] = [0,2] + [2,3]$

But since $m[0,2] = \emptyset$, only consider the first split

Look up the grammar table to find a rule $A \rightarrow B\ C$ such that

$B \in m[0,1]$ and $C \in m[1,3]$

Found three rules

$$m[0,3] = \{S, VP, X2\}$$

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

$Det \rightarrow that \mid this \mid the \mid a$

$Noun \rightarrow book \mid flight \mid meal \mid money$

$Verb \rightarrow book \mid include \mid prefer$

CKY Algorithm

Outer loop: solves the **span** $[0, j]$

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
      table[j − 1, j] ← table[j − 1, j] ∪ A
    for i ← from j − 2 down to 0 do
      for k ← i + 1 to j − 1 do
        for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
          table[i,j] ← table[i,j] ∪ A
```

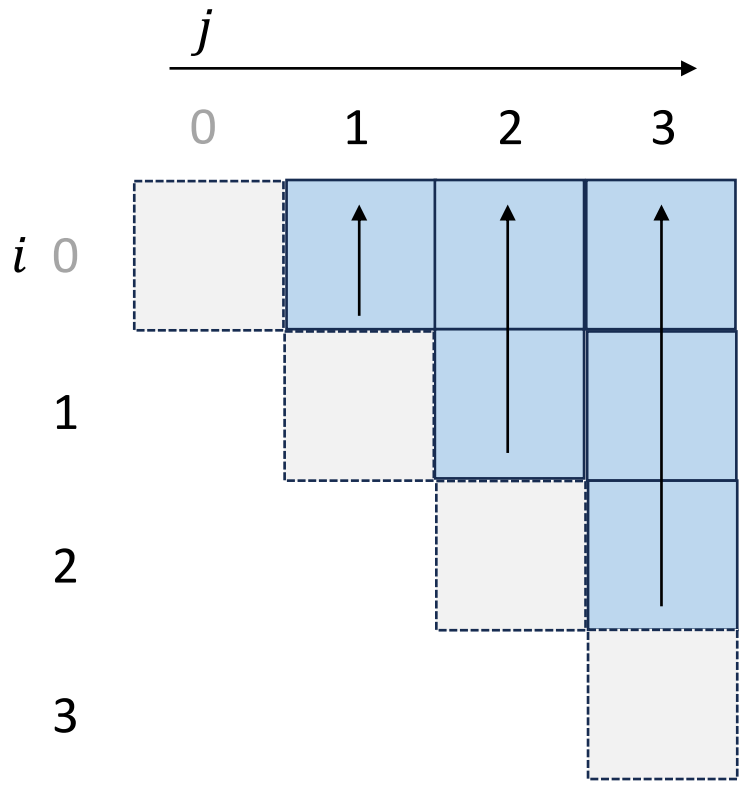
逐步解决从开头（位置 0）到 j的跨度；逐步扩展解析的范围

First inner loop: solves the base case **span** $[j-1, j]$

对于每个位置，直接找到和文法匹配的非终止符A

倒序遍历起点

Fig. CKY Algorithm. Ch 17, pp 14, SLP3



Second inner loop: solves the subproblems ranging from **span** $[j-2, j]$ to **span** $[0, j]$

inner-inner loop: solves each subproblem by breaking it down **span** $[i, j] \Rightarrow$ **span** $[i, k]$ + $[k, j]$

CKY Parsing

之前介绍的 CKY 算法主要作为一个 识别器（recognizer），其目标是判断一个句子是否符合给定的上下文无关文法（CFG），并填充解析表以标识可能的非终结符
仅仅识别还不足以满足解析需求。解析器需要返回句子的所有可能解析树（parse trees）

- The CKY algorithm given in previous slide is a **recognizer**, not a **parser**
- As a parser, it should return all possible parses (trees) for a given input
- Need to make two changes:
- 1. Augment table entries with ***pointers*** (to entries from which the current non-terminal is derived)

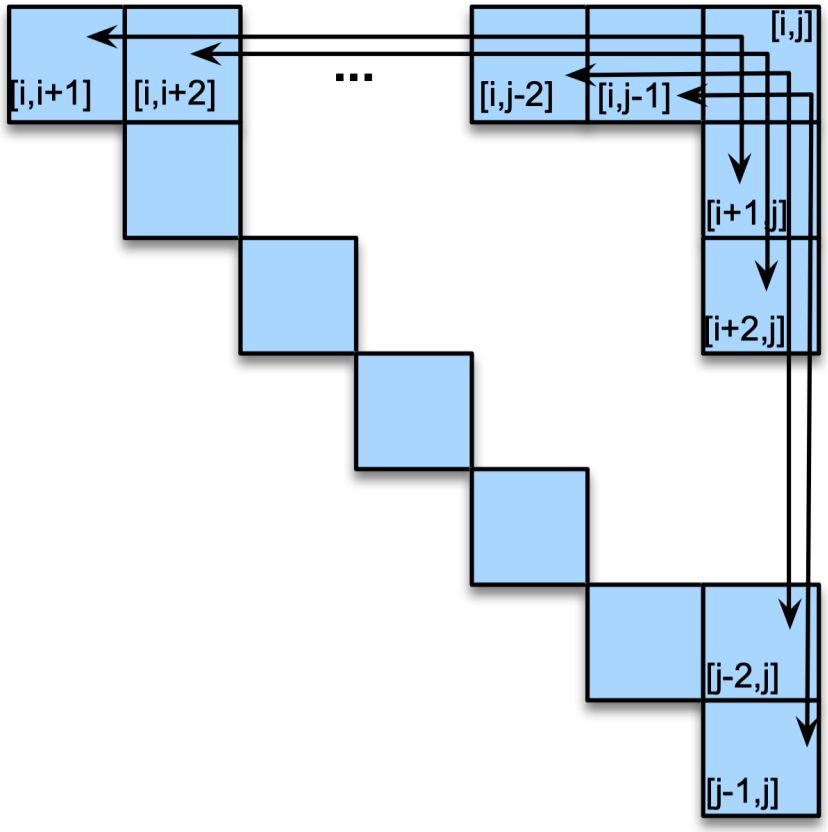
指向它是从哪些子条目（constituents）推导而来的　　记录（i，k），（k，j）

- 2. Allow multiple versions of the same non-terminal to be entered

允许表中同一个非终结符出现多次，反映不同的推导路径

- With these changes, the completed table contains all possible parses for the input
- choose an S from cell $[0, n]$ ⇒ recursively retrieve its constituents from the table

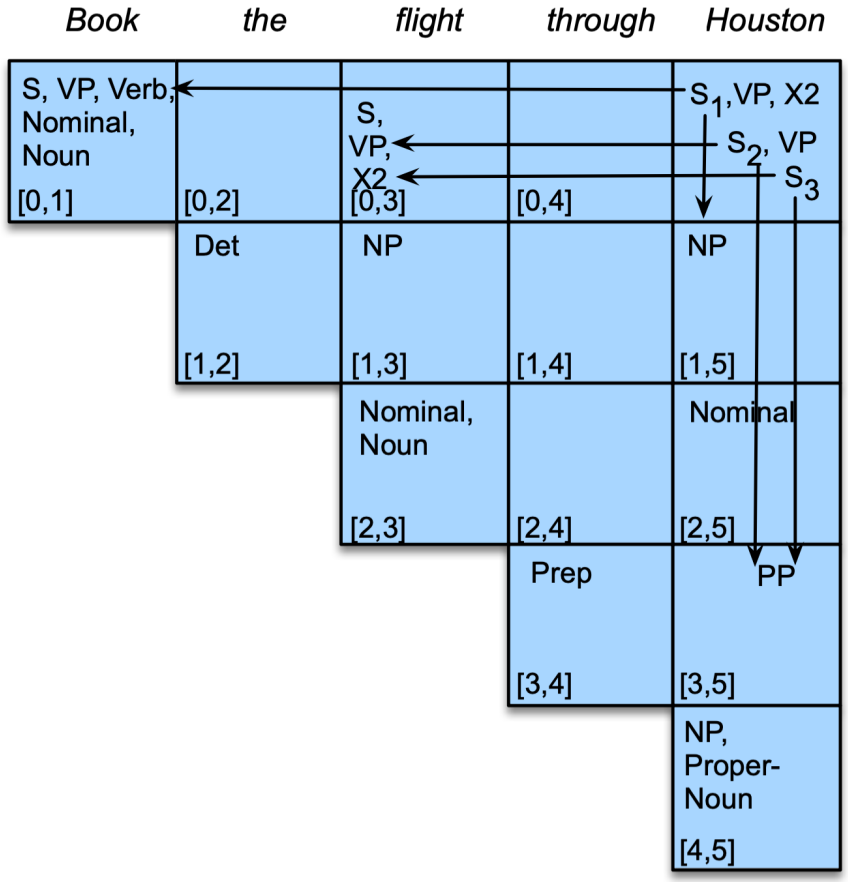
CKY Parsing - Add pointers



Entry $[i, j]$ contains a pair of pointers for each non-terminal in it, which point to the children non-terminals in entry $[i, k]$ and $[k, j]$
$(\mathrm{i} + 1 \leq k \leq j - 1)$

Fig. 17.13., Ch 17, pp 15, SLP3

CKY Parsing - Multiple versions for same non-terminal



Start with cell $[0,5]$, which contains three versions of S non-terminal:

$S_1 \rightarrow Verb\ \ NP$

$S_2 \rightarrow VP\ \ PP$

$S_3 \rightarrow X2\ \ PP$

Follow the pointers recursively to build the parse tree

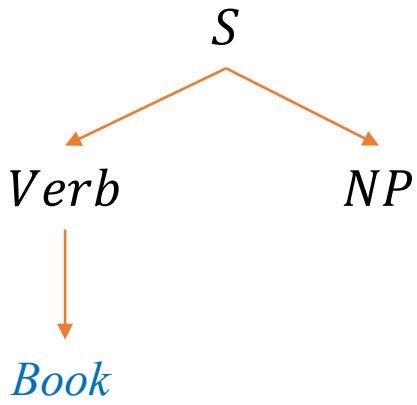


Fig. 17.14., Ch 17, pp 16, SLP3

CKY Parsing - Build the parse tree recursively

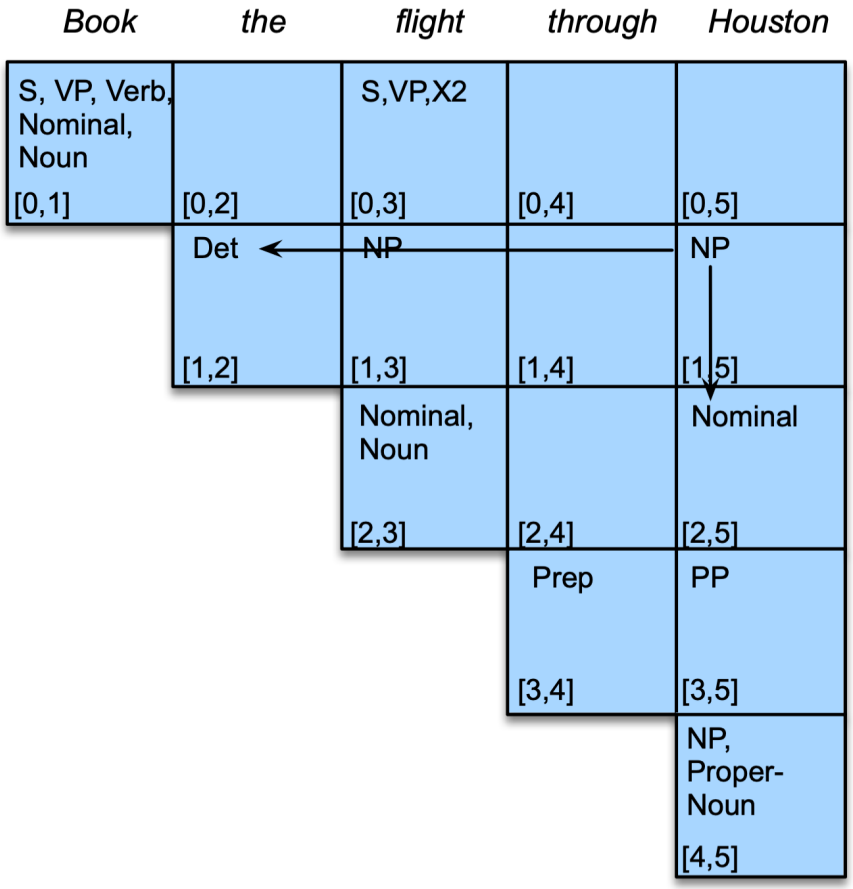
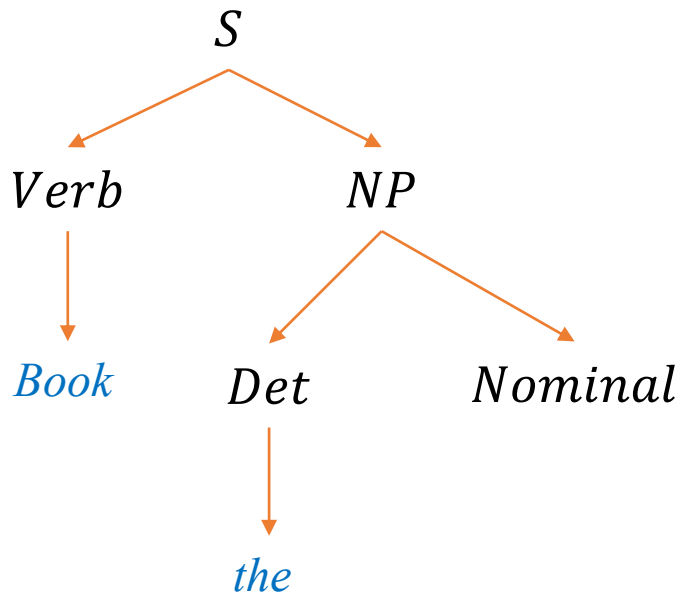


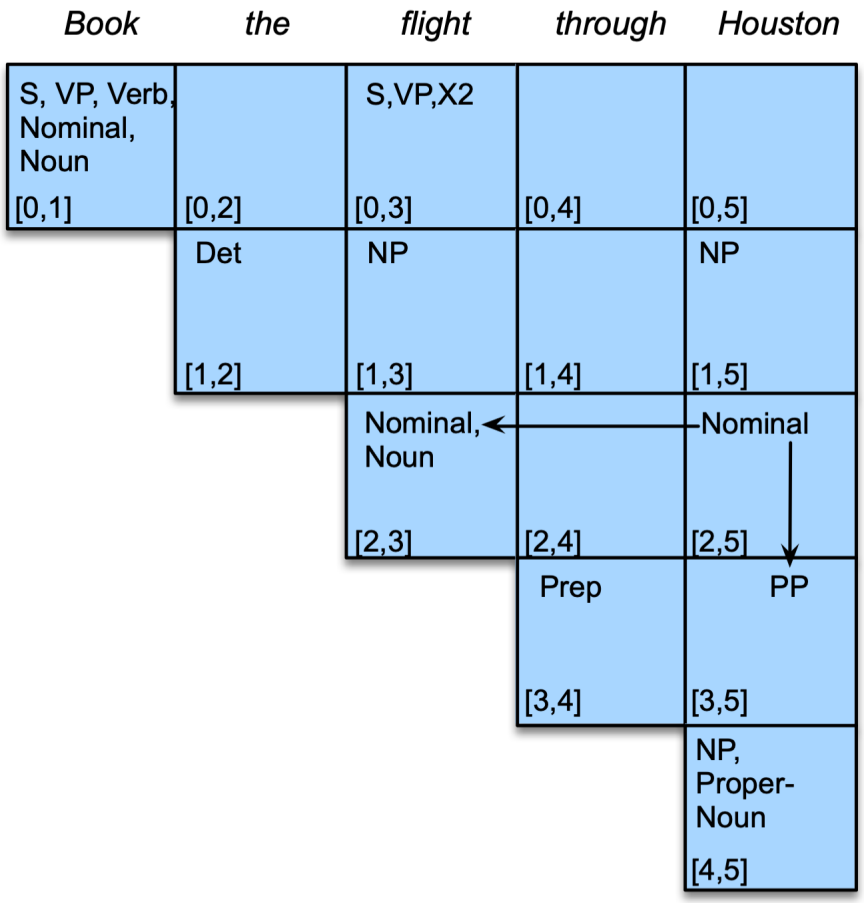
Fig. 17.14., Ch 17, pp 16, SLP3

Go to cell [1,5]

$NP \rightarrow Det \ \ Nominal$



CKY Parsing - Build the parse tree recursively



Go to cell [2,5]

$Nominal \rightarrow Nominal\ \ PP$

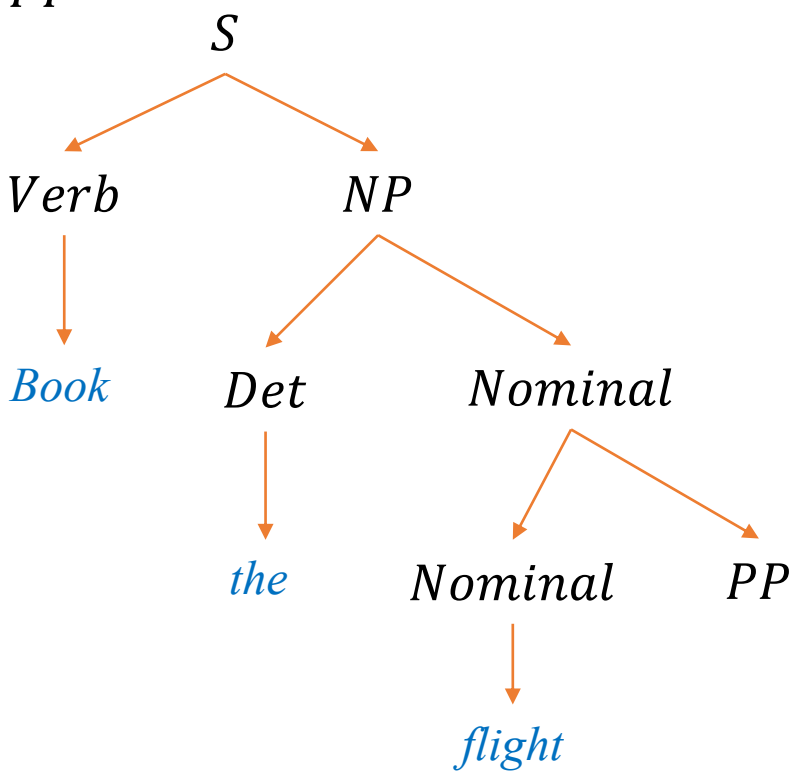
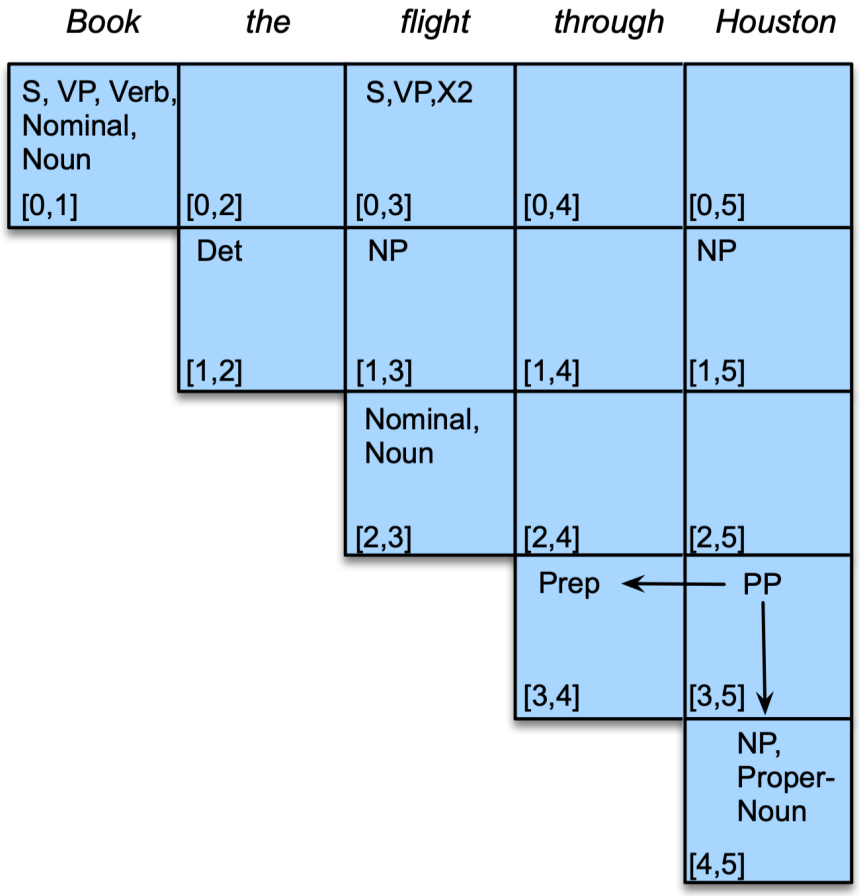


Fig. 17.14., Ch 17, pp 16, SLP3

CKY Parsing - Build the parse tree recursively



Go to cell [3,5]

$PP \rightarrow Prep\ \ NP$

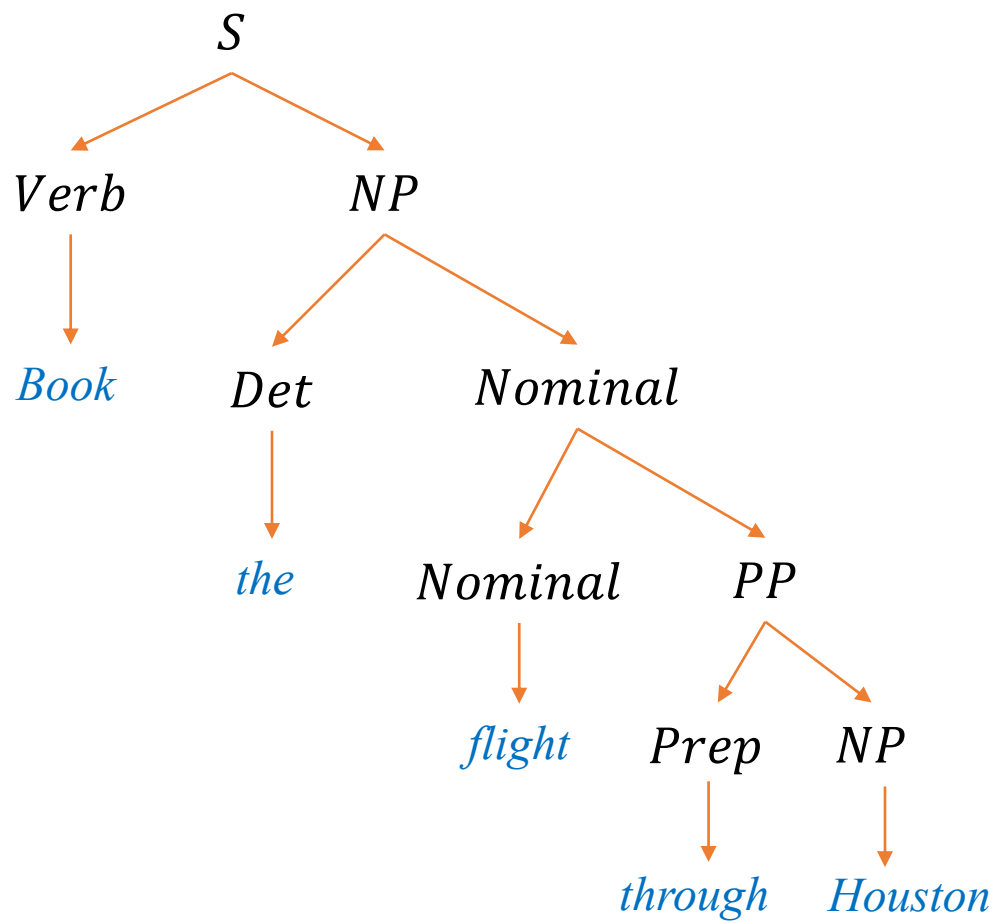


Fig. 17.14., Ch 17, pp 16, SLP3

Basic CKY Parsing's Problem

- It doesn't tell which parse tree is the correct one!

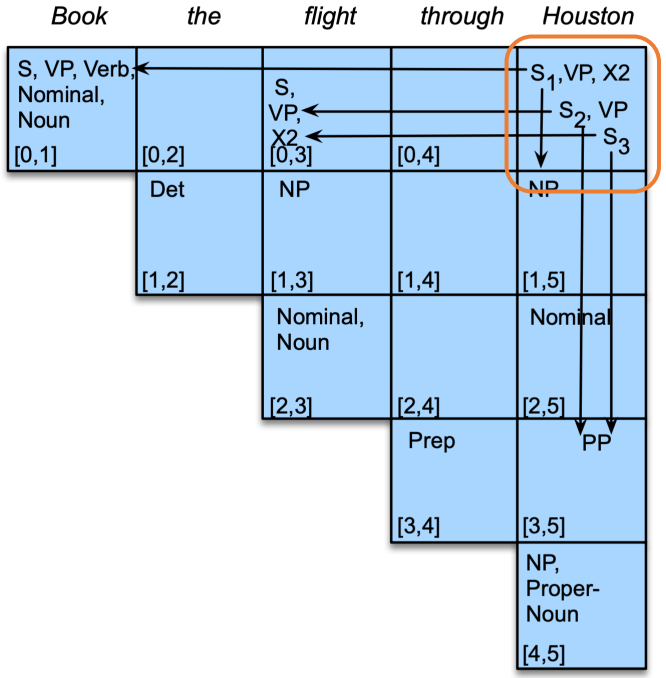
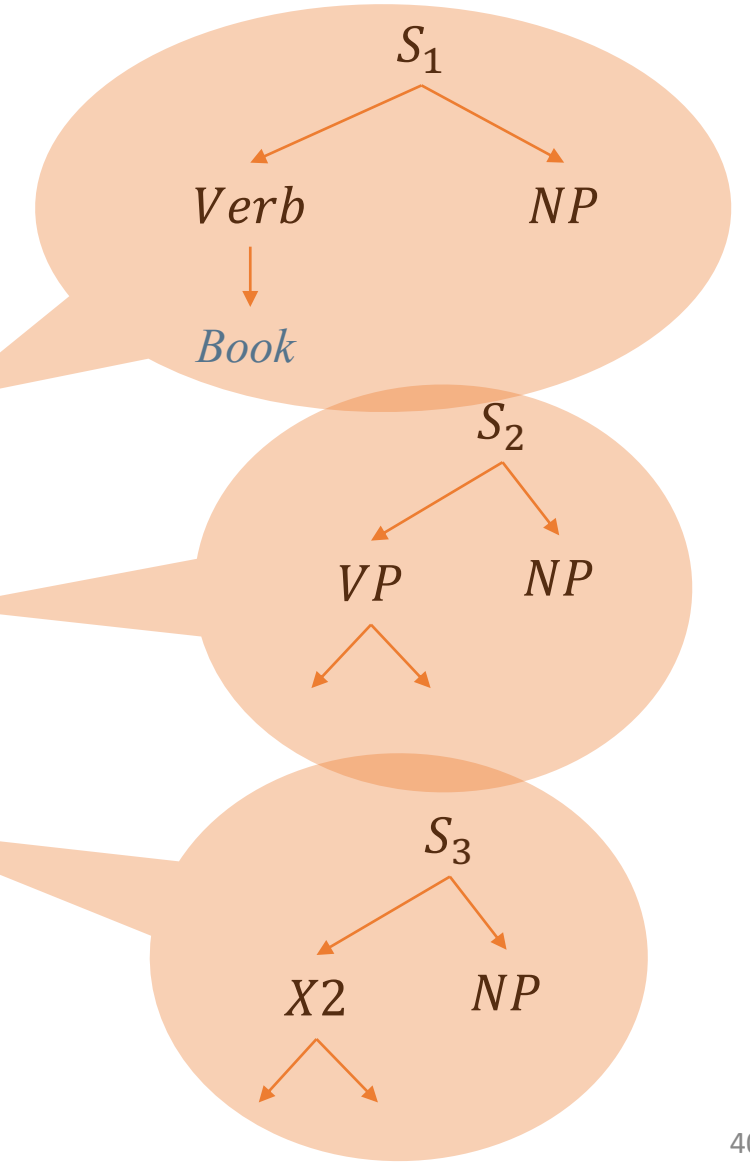


Fig. 17.14., Ch 17, pp 16, SLP3

$S_1 \rightarrow Verb\ \ NP$

$S_2 \rightarrow VP\ \ PP$

$S_3 \rightarrow X2\ \ PP$

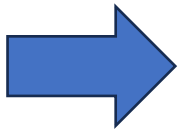


Pre-Neural Solutions

- Augment CFG to **Probabilistic Context-Free Grammar (PCFG)**

CFG

- ***N***: a set of **non-terminal** symbols
- ***Σ***: a set of **terminal** symbols
- ***R***: a set of rules (or productions), each of the form $A \rightarrow \beta$ where $A \in N$, and $\beta \in \Sigma \cup N$
- ***S***: a start symbol



PCFG

- ***N***: a set of **non-terminal** symbols
- ***Σ***: a set of **terminal** symbols
- ***R***: a set of rules (or productions), each of the form $\mathbf{A \rightarrow \beta\ [p]}$ where $A \in N$, and $\beta \in \Sigma \cup N$ and $p \in [0,1]$ is the conditional probability $\boldsymbol{p(\beta|A)}$
- ***S***: a start symbol

Consider all possible expansions of a non-terminal, then

$$\sum_{\beta} p(A \rightarrow \beta) = 1$$

PCFG Example

A PCFG version to the grammar $\mathcal{L}_1$

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $trip$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.05] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| NWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [.40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]	| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

Probabilities are estimated based on a corpus

- These numbers are for pedagogical purposes
- Any real corpus would have more rules and the real probabilities will be much smaller

Use PCFG for disambiguating parsing results

- The probability of a particular parse tree T is defined as the product of probabilities of all the n rules to expand each of the n non-terminals

$$p(T) = \prod_{i=1}^{n} p(A_i \rightarrow \beta_i)$$

- Consider all possible parse trees of a sentence S, we need to pick the tree with the highest probability

$$\hat{T}(S) = \arg\max_{T} p(T)$$

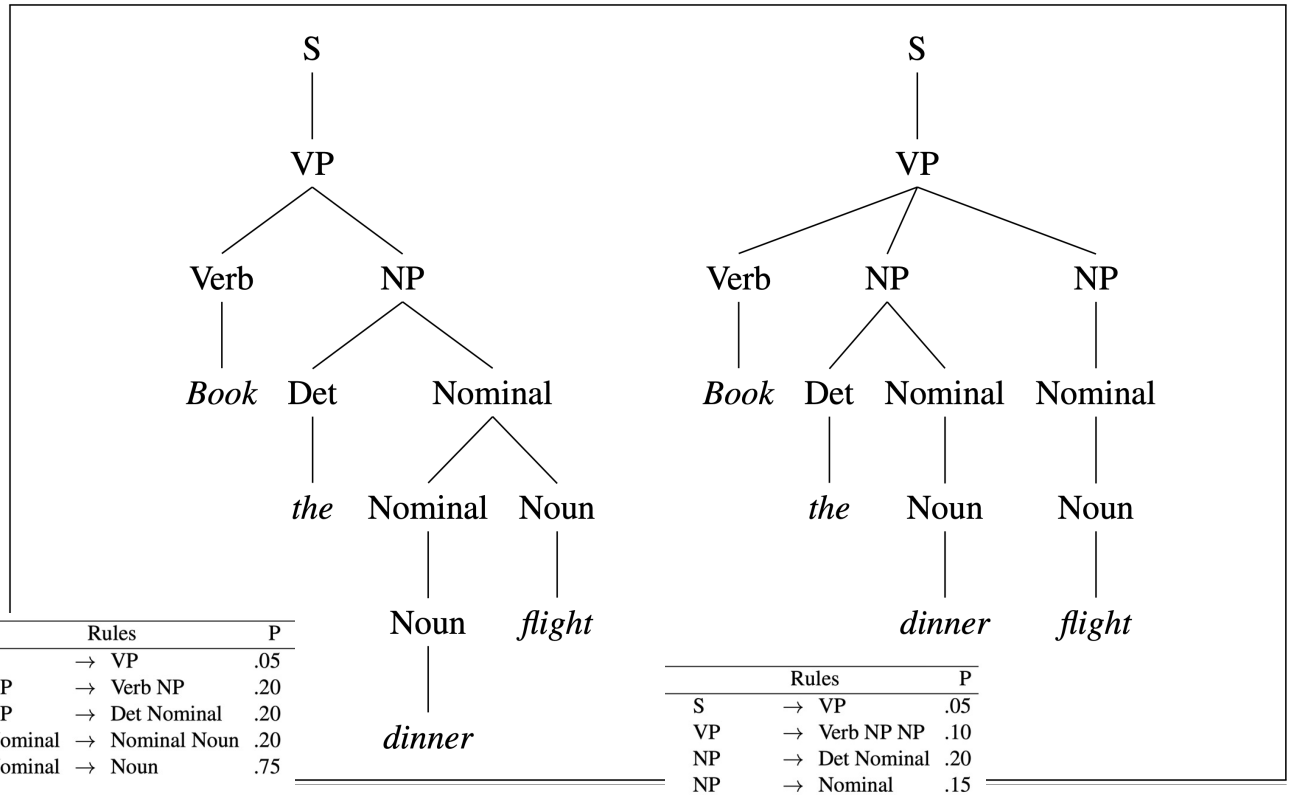
PCFG Usage Example

Sentence: *Book the dinner flight*

two senses:

"Book a flight that serves dinner"

"Book a flight on behalf of 'the dinner'"



	Rules	P
S	→ VP	.05
VP	→ Verb NP	.20
NP	→ Det Nominal	.20
Nominal	→ Nominal Noun	.20
Nominal	→ Noun	.75
Verb	→ book	.30
Det	→ the	.60
Noun	→ dinner	.10
Noun	→ flight	.40

	Rules	P
S	→ VP	.05
VP	→ Verb NP NP	.10
NP	→ Det Nominal	.20
NP	→ Nominal	.15
Nominal	→ Noun	.75
Nominal	→ Noun	.75
Verb	→ book	.30
Det	→ the	.60
Noun	→ dinner	.10
Noun	→ flight	.40

Compute using the probabilities of PCFG $\mathcal{L}_0$:

$$P(T_{\text{left}}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2\times10^{-6}}$$

$$P(T_{\text{right}}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1\times10^{-7}}$$

Overview

- Context Free Grammars (CFG)
- Constituency Parsing
- **Neural Constituency Parsing**

Neural Constituency Parsing

消除歧义

- A neural extension of CKY algorithm to **disambiguate** among possible parses
- **Span-based** constituency parsing, or **neural CKY**
- **Idea:**
- Like classic CKY: learns to map a span of words to a constituent, and hierarchically combines larger and larger spans to build the parse-tree bottom-up
- ***Unlike*** classic CKY: doesn't use hand-written grammar to constrain *what* constituents can be combined;

不依赖预定义的上下文无关文法（CFG）来约束成分组合，而是依靠神经网络从数据中学习的跨度表示来预测可能的组

- Instead, just relying on learned neural representations of spans to predict *likely* combinations ⇒ **probabilities**
- Ex. Kitaev et al. (2019)

Neural CKY Architecture

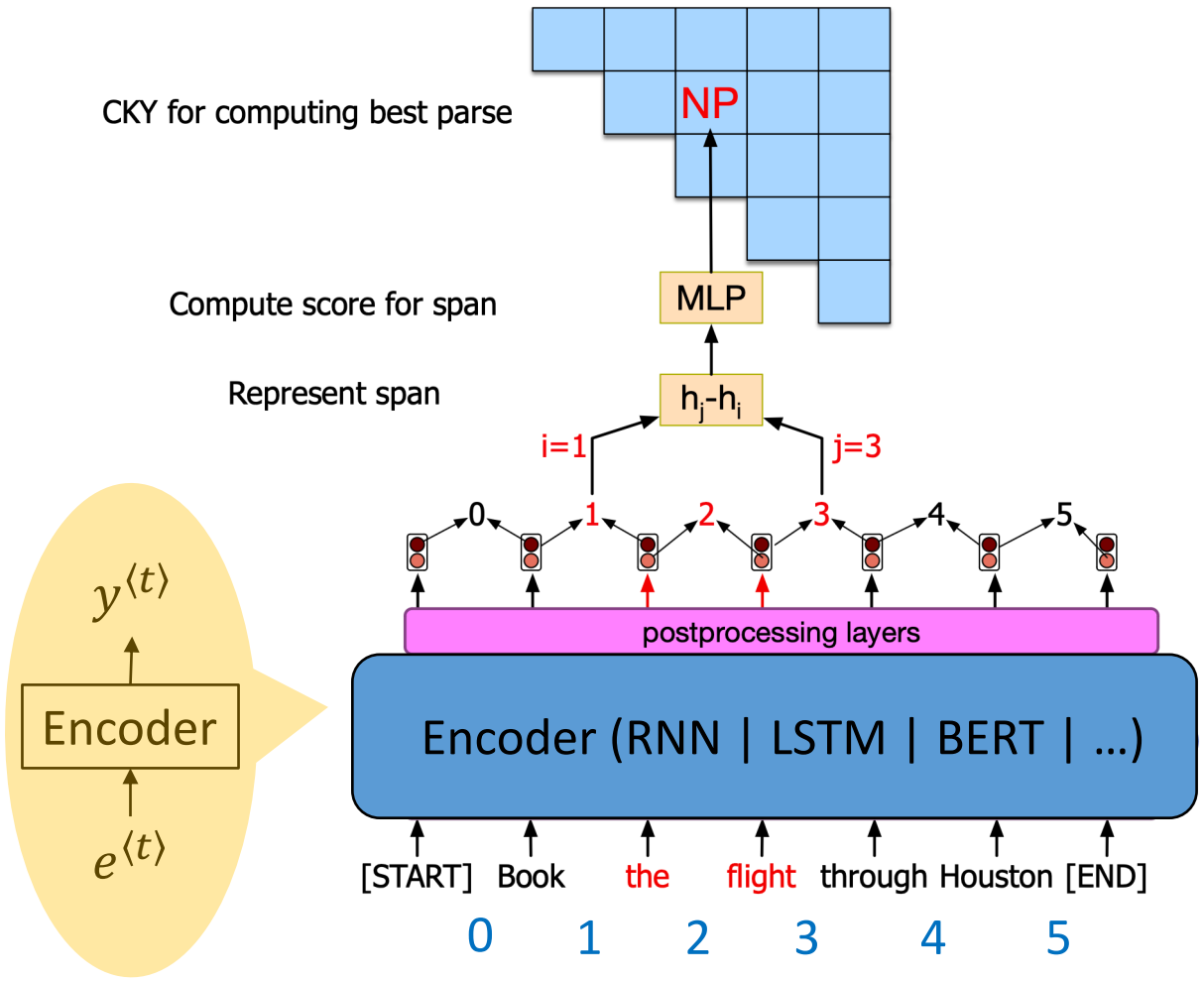


Fig. 17.15., Ch 17, pp 17, SLP3 Based on the architecture from Kitaev et al. (2019)

Consider the span that lies between positions i and j with non-terminal label l

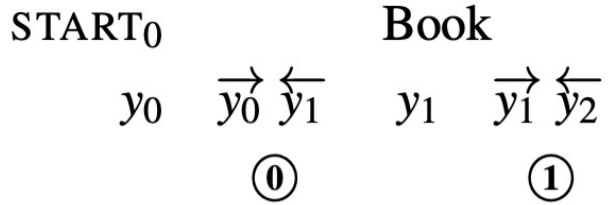
Assign a score $\boldsymbol{s(i,j,l)}$ to this span 表示该跨度被标签标注的置信度

How to compute this score? Need to represent the spans with embedding vectors

First, obtain **word vectors** $\boldsymbol{y^{\langle t \rangle}}$ from the encoder (i.e., hidden state from LSTM, or last layer of Transformer)

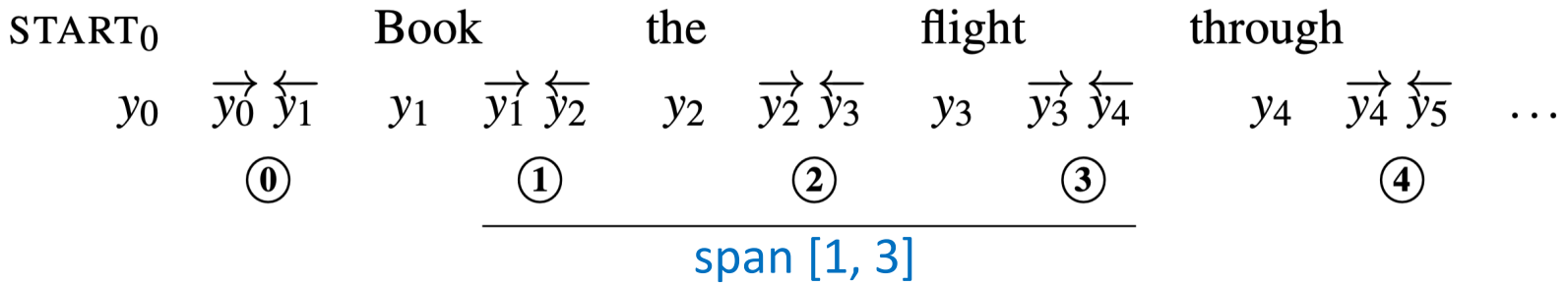
Then, split $\boldsymbol{y^{\langle t \rangle}}$ into two halves:

- $\overrightarrow{\boldsymbol{y}}^{\langle t \rangle}$ for spans **starting** from this position
- $\overleftarrow{\boldsymbol{y}}^{\langle t \rangle}$ for spans **ending** from this position



Represent spans with embedding vectors

- A span can be represented by taking the difference between its **start** and **end** embeddings (Wang and Chang, 2016)



- span $[i, j]$ represented by: $\boldsymbol{v}(\boldsymbol{i}, \boldsymbol{j}) = \left[\overrightarrow{\boldsymbol{y}}^{\langle \boldsymbol{j} \rangle} - \overrightarrow{\boldsymbol{y}}^{\langle \boldsymbol{i} \rangle}; \overleftarrow{\boldsymbol{y}}^{\langle \boldsymbol{j+1} \rangle} - \overleftarrow{\boldsymbol{y}}^{\langle \boldsymbol{i+1} \rangle}\right]$

前向上下文 后向上下文 通过其起点和终点的嵌入向量之差来表示

- Then the span vector is passed through a classifier (e.g., MLP), whose output size is the vocabulary size of non-terminal labels

$$\boldsymbol{s}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{l}) = \boldsymbol{W_2}\mathrm{ReLU}\left(\mathrm{LayerNorm}(\boldsymbol{W_1}\boldsymbol{v}(\boldsymbol{i}, \boldsymbol{j}))\right)$$

ref: Kitaev et al. (2019)

Use Span Scores for Parsing

- Now we have a score for each labeled span $\boldsymbol{s(i, j, l)}$. How about a score for an entire parse tree?
- Formally, a parse tree is a set of labeled spans, where
- the tth span starting at position $i^{\langle t \rangle}$ and ending at $j^{\langle t \rangle}$, with label $l^{\langle t \rangle}$:

$$T = \{(i^{\langle t \rangle}, j^{\langle t \rangle}, l^{\langle t \rangle})\}, t = 1, \ldots, |T|$$

- Compute a score for the whole tree by summing over the scores of its constituent spans:

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l)$$

Use Span Scores for Parsing

- We can choose the final parse tree as the one with the **maximum score**:

$$\hat{T} = \arg\max_{T} s(T)$$

- Simplest method ⇒ **greedy decoding**: choose the highest scoring label for each span
- Drawback: not guaranteed to produce a tree, since the best scored label might not fit into a complete tree
- In practice, 95% of predicted trees from greedy decoding are valid (Gaddy et al., 2018)

Use Span Scores for Parsing

- It is more common to use a variant CKY algorithm to find the parse tree, defined as follow:
- Let $\boldsymbol{s_{\text{best}}(i,j)}$ be the score of the best subtree spanning $[i,j]$
- If $j = i + 1$, i.e., spans of length one, simply choose the best label

跨度只包含1个单

$$s_{\text{best}}(i,j) = \max_l s(i,j,l)$$

- Other wise, use the recursion:

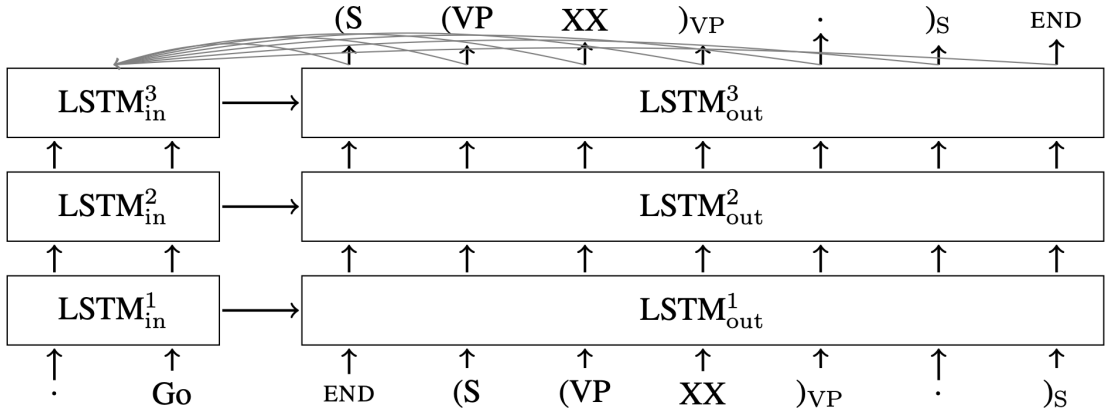
$$s_{\text{best}}(i,j) = \max_l s(i,j,l) + \max_k \left(s_{\text{best}}(i,k) + s_{\text{best}}(k,j) \right)$$

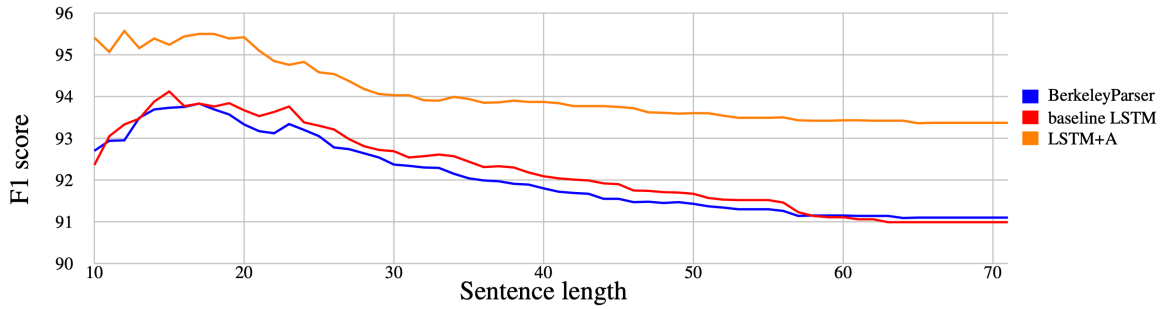
该跨度本身标签得分

所有子树中，两标签得分之和的最大值

Actually, a much earlier neural parsing solution

- Maybe the very first one: Grammar as a Foreign Language. 2015. NIPS
- Author list: Oriol Vinyals, Lukasz Kaiser, …, **Ilya Sutskever**, **Geoffrey Hinton**





LSTM + Attention
Input: the *reversed* syntactic tree (where the title is from)
Output: the correct tree

Evaluating Parsers

- **PARSEVAL** metrics (Black et al., 1991): measures how much the **constituents** in the hypothesis (predicted) parse tree look like the those in a hand-labeled, **reference** parse
- Need a "**golden standard**" (human labeled) tree for each sentence in test set (usually drawn from Penn Treebank)
- A constituent in a hypothesis parse C_h is labeled ***correct***
- if there is a constituent in the reference parse C_r with the same **starting** point i, **ending** point j, and **non-terminal** label.
- Then can define precision and recall:

$$\textbf{precision} = \frac{\#\text{ of correct constituents in } C_h}{\#\text{ of total consitituents in } C_h} \qquad \textbf{recall} = \frac{\#\text{ of correct constituents in } C_h}{\#\text{ of total consitituents in } C_r}$$

LLMs can do Constituency Parsing

Results from DeepSeek v3 (2025.3)

分析以下这句话的句法结构，用成分句法，即constituency parsing 的形式：
夫大块载我以形

完整标注

Copy

```
        S
      /   \
    NP     VP
    |    / | \
  大块  V  NP  PP
       |  |  /  \
      载 我 以   NP
                 |
                形
```

LLMs can do Constituency Parsing

Results from DeepSeek v3 (2025.3)

把这句话翻译成英文，然后做同样的句法分析

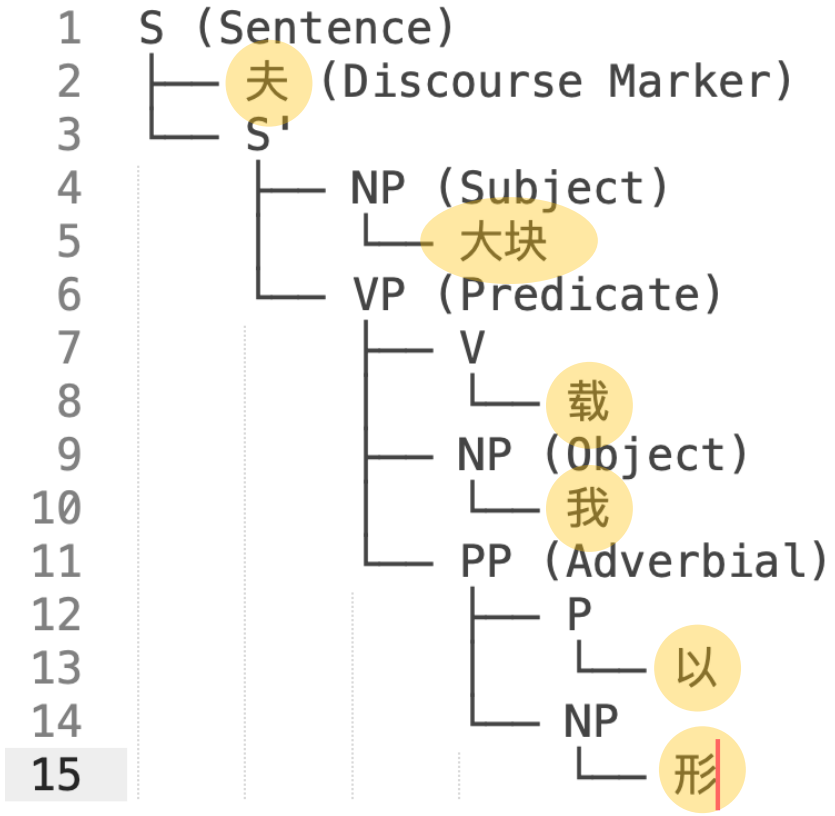
树状结构（Constituency Tree）：

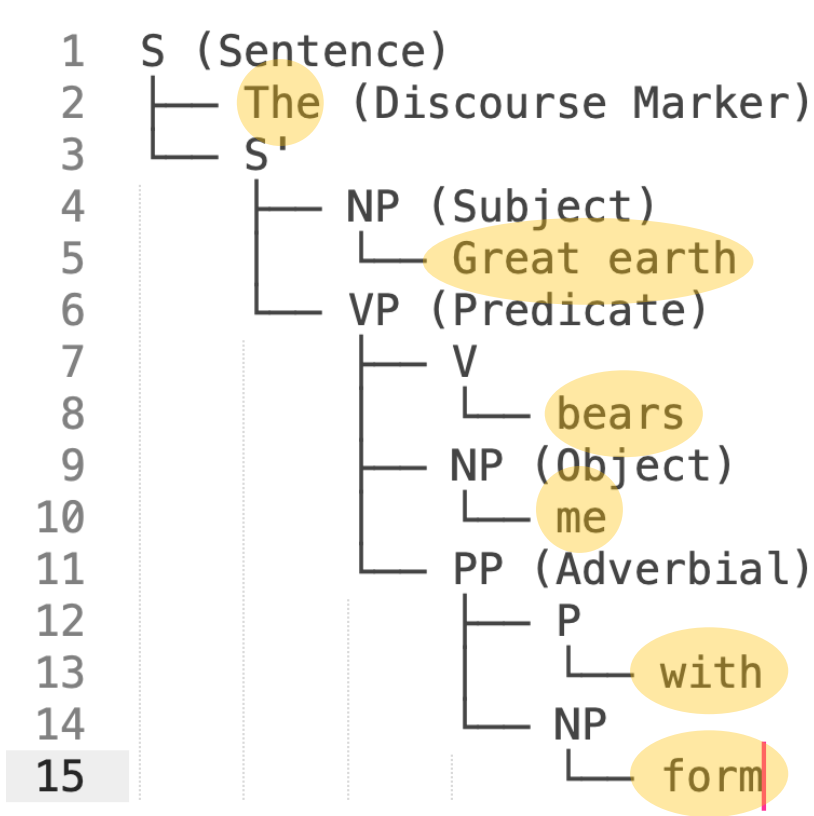
Copy

```
          S
        /   \
      NP      VP
    / | \    /  |  \
 Det Adj N  V  NP   PP
 |   |  |  |  |   /  \
The great earth bears me with form
```

Universals in human languages

What a co-incidence!





Constituency Parsing with LLMs

- Induce constituency parse trees relying only on the internal knowledge of pre-trained language models.
- Competitive with common unsupervised parsers; more effective than typical supervised parsers in few-shot settings. (Kim 2022)

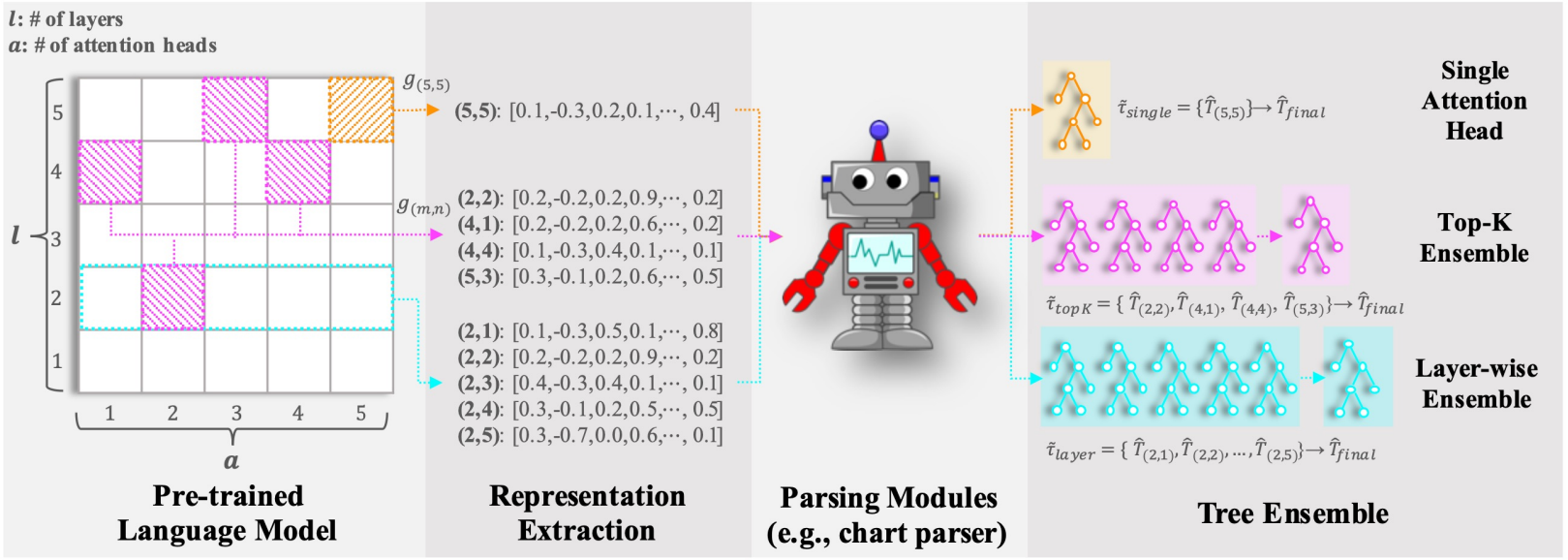


Figure from Taeuk Kim (2022)
Revisiting the Practical Effectiveness of Constituency Parse Extraction from Pre-trained Language Models

References

- Chomsky, N. 1956. Three models for the description of language. IRE Transactions on Information Theory, 2(3):113–124.
- Backus, J. W. 1959. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. *Information Processing: Pro- ceedings of the International Conference on Information Processing, Paris*. UNESCO.
- Naur, P., J. W. Backus, et al. 1960. Report on the algorithmic language ALGOL 60. CACM, 3(5):299–314. Revised in CACM 6:1, 1-17, 1963.
- Younger, D. H. 1967. Recognition and parsing of context-free languages in time n^3. Information and Control, 10:189–208.
- Kasami, T. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.
- Kay, M. 1967. Experiments with a powerful parser. COLING.
- Vinyals, O., Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015. Grammar as a foreign language. *NeurIPS*.
- Kitaev, N., S. Cao, and D. Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. ACL.
- Wang, W. and B. Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. ACL.
- Black, E., et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. Speech and Natural Language Workshop.
- Kim, T. 2022. Revisiting the Practical Effectiveness of Constituency Parse Extraction from Pre-trained Language Models. ACL.