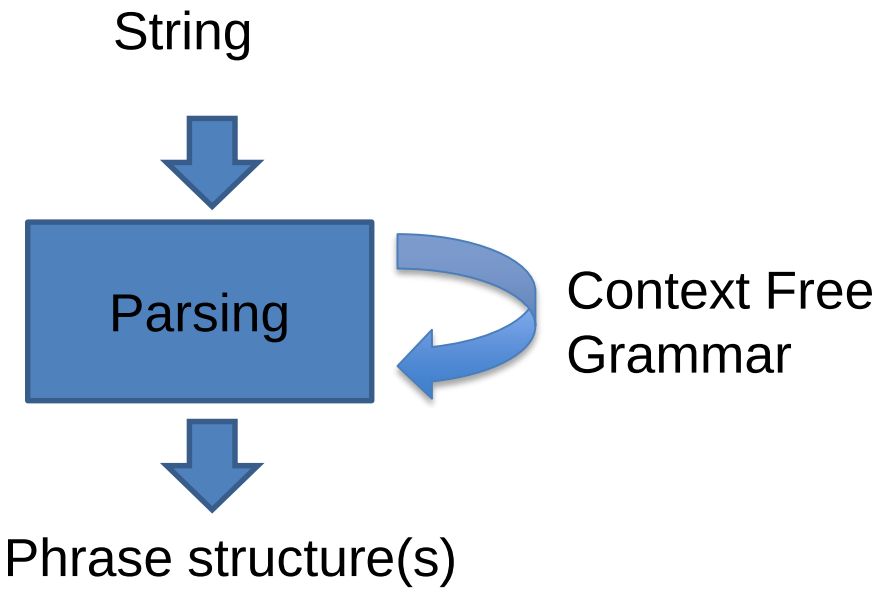
CE/CZ4045 Natural Language Processing

Syntactic Parsing (Chapter 13)

Review

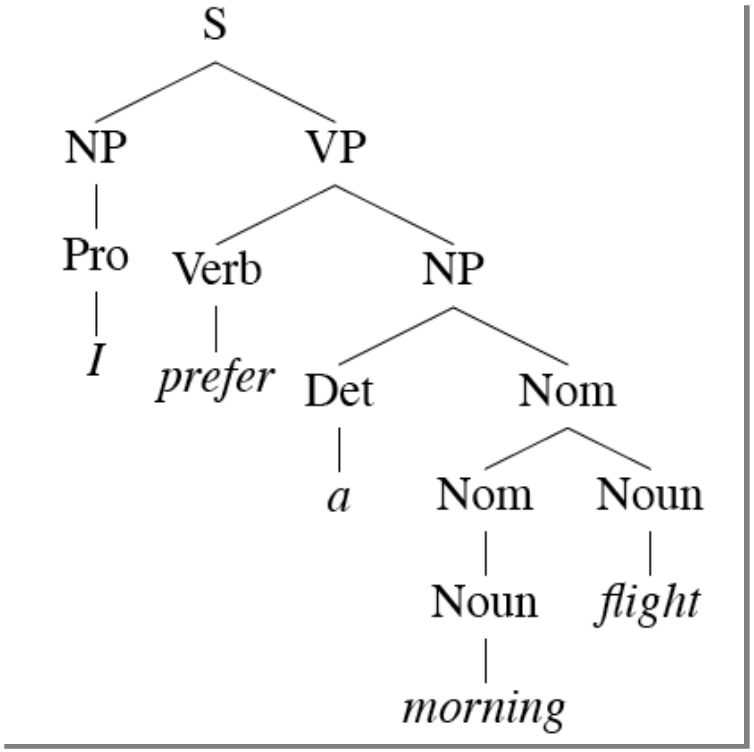
Syntactic parsing in the course



Phrase structure

Review

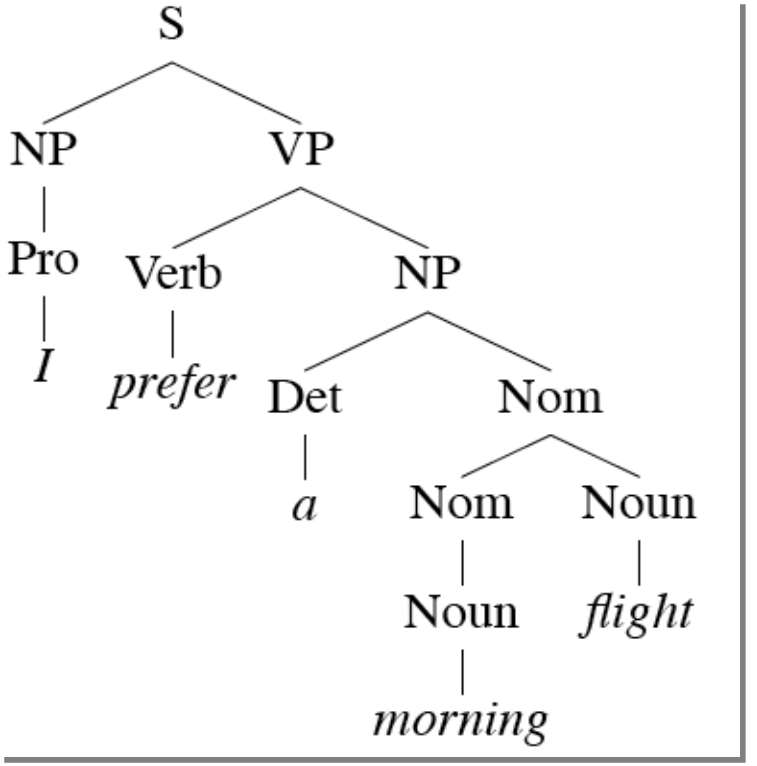
- Organizes words into nested constituents



Review

Context-free grammar

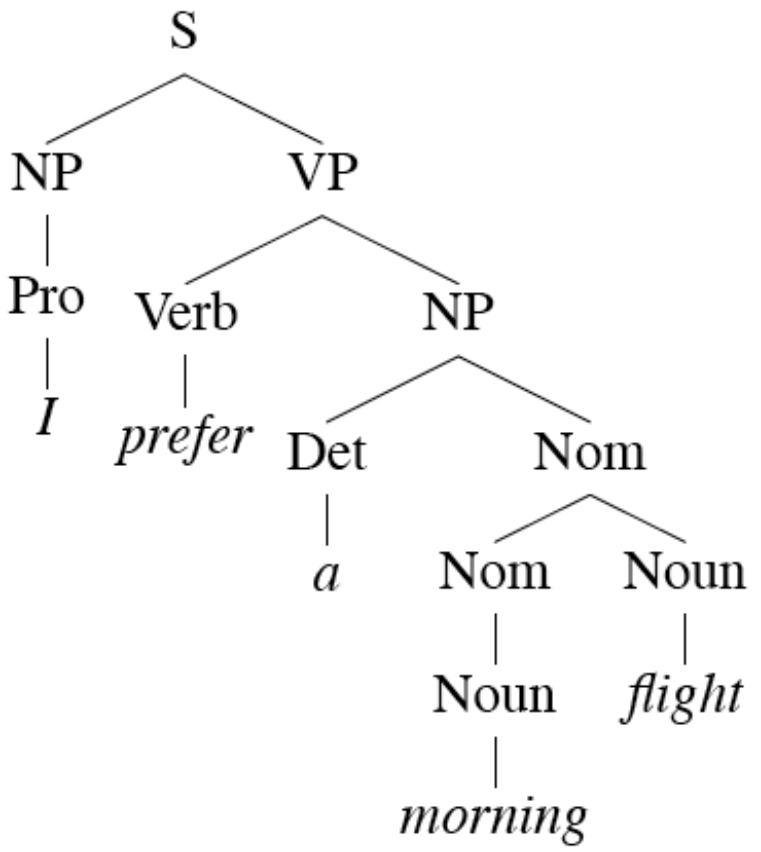
- G = (T, N, S, R)
  - T: a set of terminals (e.g. ‘flight’)
  - N: a set of non-terminals (e.g. Noun)
  - S: the start symbol, a non-terminal
  - R: rules of the form X →γ
    - X: a non-terminal
    - γ: a sequence of terminals and non-terminals



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

Parsing strategies

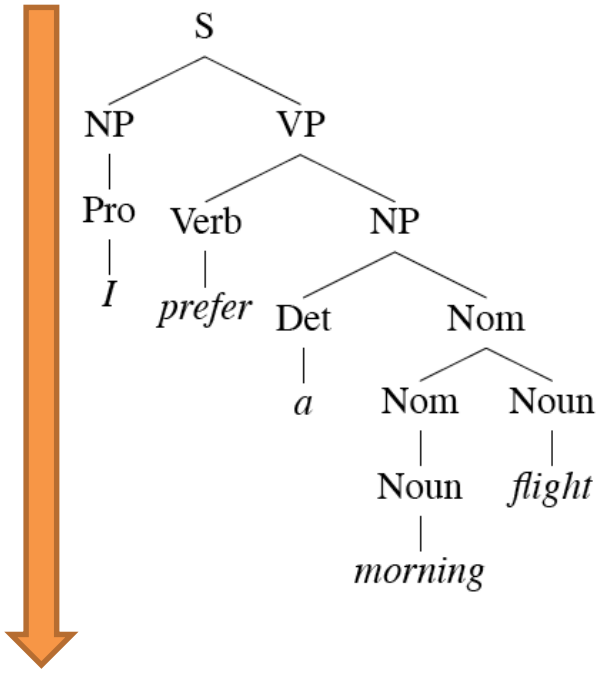
Top-down parsing



Bottom-up parsing

Top-down parsing

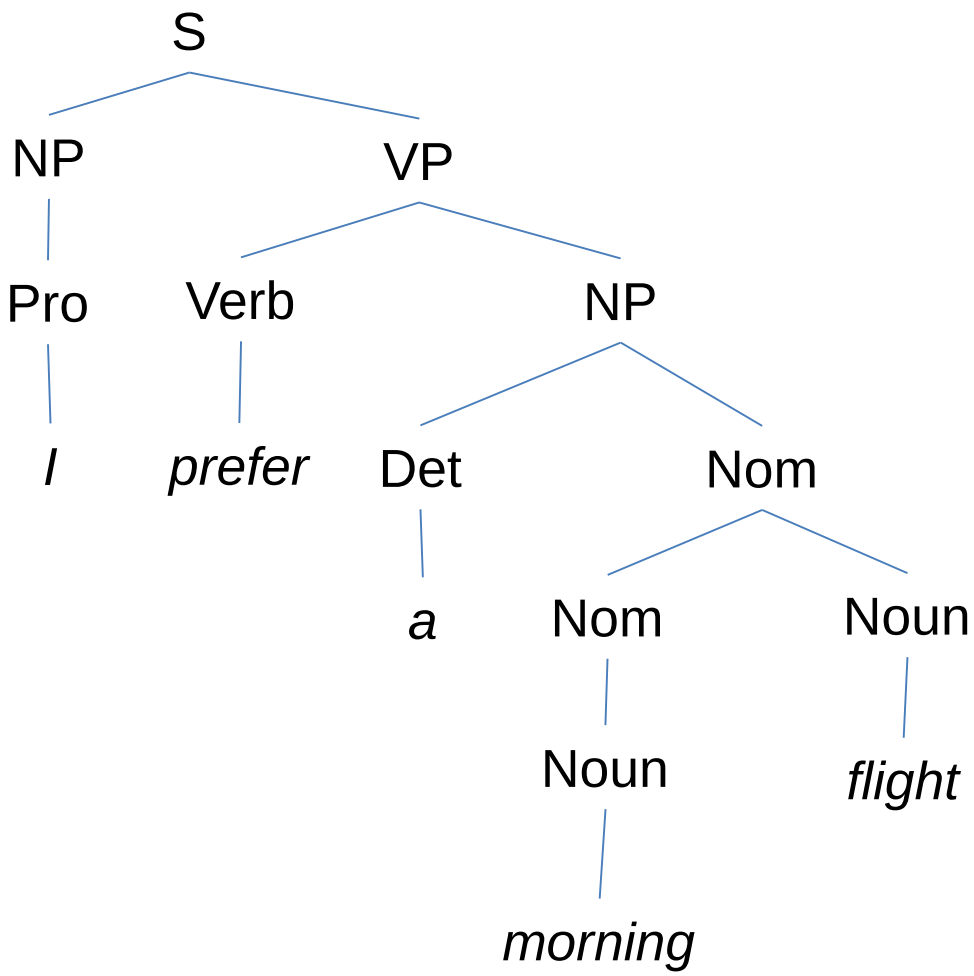
- Goal-directed
  - Start from ‘S’
- Rewrite the goal(s) with the RHS of relevant rules, e.g,
  - S → NP VP,
  - VP → Verb NP
- Parsing is finished when the rewriting **generates** the whole string



Top-down parsing

Top-down parsing: Example

I prefer a morning flight.



1. S → NP VP

2. NP → Pro
   Pro → *I*

3. VP → Verb NP
   Verb → *prefer*

4. NP → Det Nom
   Det → *a*

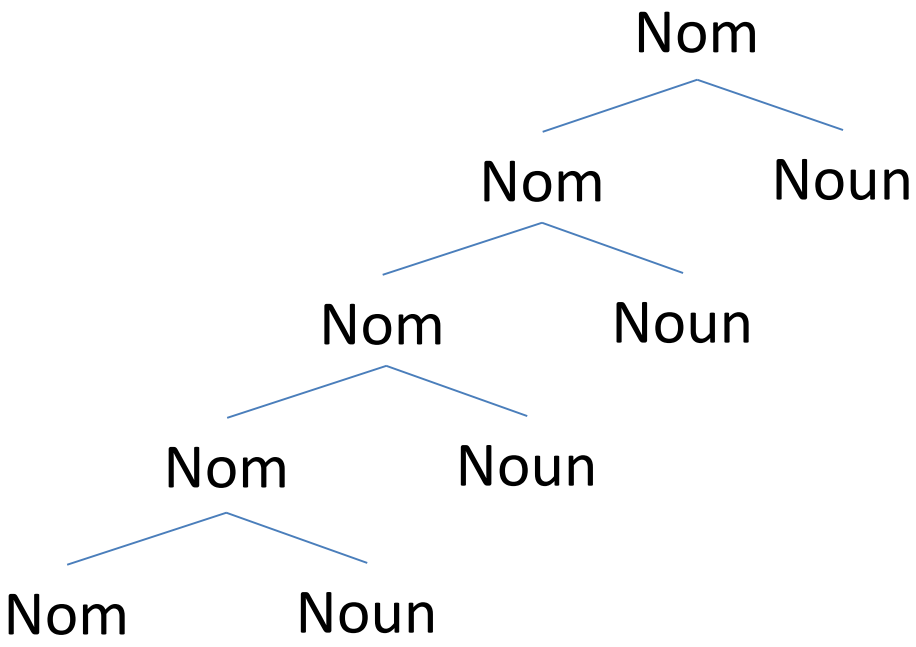
5. Nom → Nom Noun
   Noun → *flight*

6. Nom → Noun
   Noun → *morning*

Top-down parsing: Issues (1/2)

- If a goal can be written in several ways, there is a choice of which rule to apply
  - Example: NP → Pro, NP → Det Nom
- Search problem
  - Search methods: Depth-first, breadth-first, goal-ordering, etc.
- Need grammar-driven control for optimization
  - May waste lots of time in trying rules that are inconsistent with the input string
  - Example: left recursive rules (e.g. Nom → Nom Noun)

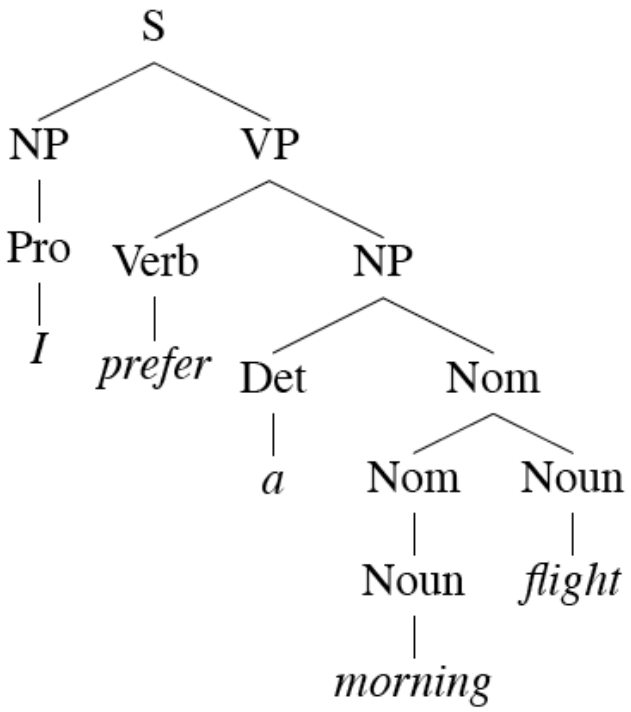
Top-down parsing: left recursive rules



Example: Southeast Asia Public Interest Research Group

Top-down parsing: Issues (2/2)

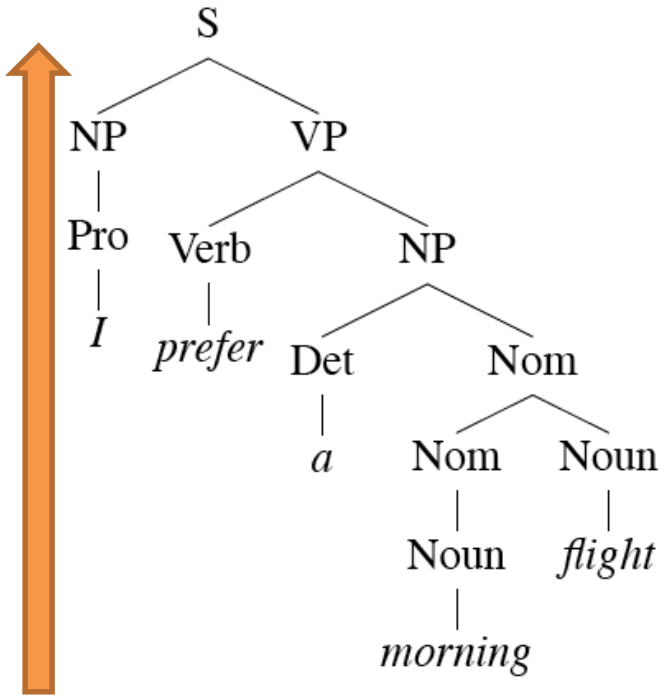
- A part-of-speech like Noun can be replaced with any noun
  - Trying out all rules of a part-of-speech is time consuming
- In practice, part-of-speech tags are predetermined



1. S → NP VP
2. NP → Pro
   Pro → *I*
3. VP → Verb NP
   Verb → *prefer*
4. NP → Det Nom
   Det → *a*
5. Nom → Nom Noun
   Noun → *flight*
6. Nom → Noun
   Noun → *morning*

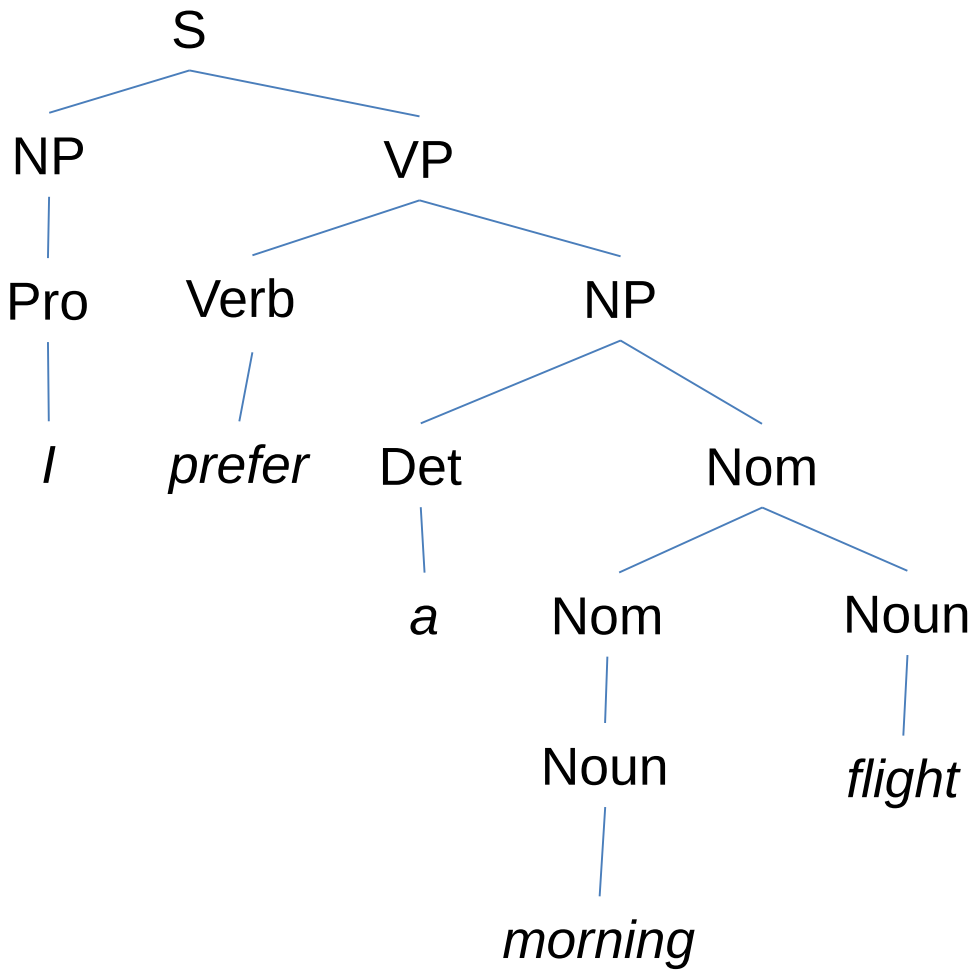
Bottom-up parsing

- Data-directed
  - Start with words
- If a substring matches the RHS of a rule, replace the substring with the LHS
  - Example: Nom → Noun (“morning”)
  - Example: Nom → Nom Noun (“*morning flight*”)
- Parsing is finished when the whole string is replaced with a goal



Bottom-up approach: Example

I prefer a morning flight.



6. S → NP VP

5. Pro → *I*
   NP → Pro

4. Verb → *prefer*
   VP → Verb NP

3. Det → *a*
   NP → Det Nom

2. Noun → *flight*
   Nom → Nom Noun

1. Noun → *morning*
   Nom → Noun

Parsing strategies

Top-down parsing

- Goal-directed
  - Start from ‘S’
- Rewrite the goal(s) with the RHS of relevant rules
  - Example: S → NP VP
- Parsing is finished when the rewriting generates the whole string

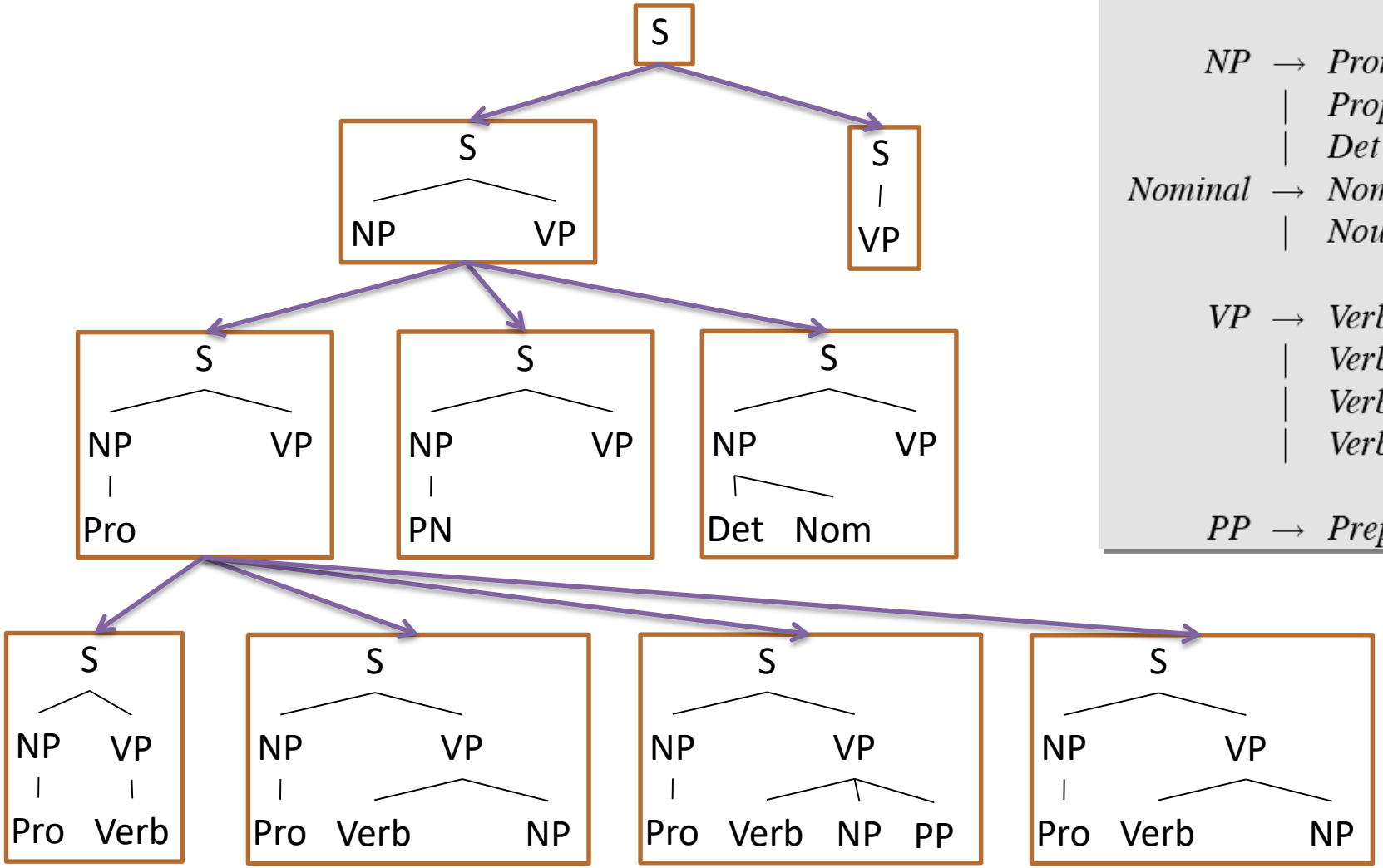
Bottom-up parsing

- Data-directed
  - Start with words
- If a substring matches the RHS of a rule, replace the substring with the LHS
  - Example: Nom → Noun
  - Nom → Nom Noun (“*morning flight*”)
- Parsing is finished when the whole string is replaced with a goal

Top-down vs. Bottom-up parsing

- Top-down
  - Waste lots of time in trying inconsistent rule applications
    - The application of the rules does not lead the generation of the given string
  - Never explore subtrees that cannot find a place in some S-rooted tree
    - All trees are generated starting with S.
- Bottom-up
  - Never suggest trees that are not grounded in the input string
    - All trees are generated based on input string
  - Trees that have no hope of leading to an S are generated
    - Some trees cannot proceed further to reach S

Parsing as search (e.g. I prefer a morning flight)



Grammar Rules

S	→	*NP VP* | *VP*
NP	→	*Pronoun*
	|	*Proper-Noun*
	|	*Det Nominal*
Nominal	→	*Nominal Noun*
	|	*Noun*
VP	→	*Verb*
	|	*Verb NP*
	|	*Verb NP PP*
	|	*Verb PP*
PP	→	*Preposition NP*

Parsing as search

1 S

2 S
NP VP

3 S
NP VP
Pro
I

4 S
NP VP
Pro Verb NP
I *prefer*

5 S
NP VP
Pro Verb NP
I *prefer* Det Nom
a

6 S
NP VP
Pro Verb NP
I *prefer* Det Nom
a Nom Noun
flight

7 S
NP VP
Pro Verb NP
I *prefer* Det Nom
a Nom Noun
Noun *flight*
morning

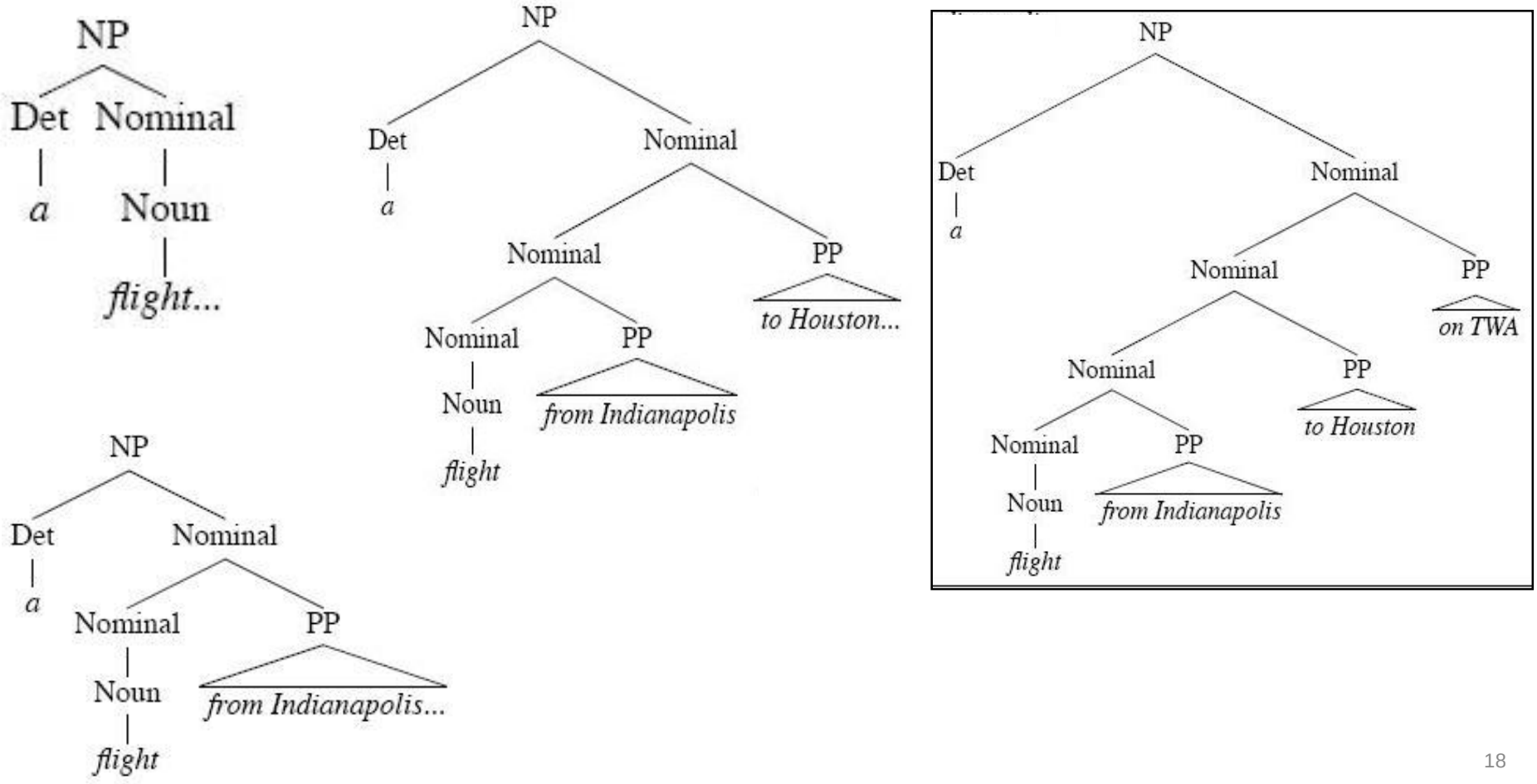
Exercise: Parsing as search

- Draw a search space for parsing "I sleep" in a top-down parsing based on L0 grammar

Grammar Rules		
S	$\rightarrow$	$NP\ VP$
NP	$\rightarrow$	$Pronoun$
	|	$Proper\text{-}Noun$
	|	$Det\ Nominal$
$Nominal$	$\rightarrow$	$Nominal\ Noun$
	|	$Noun$
VP	$\rightarrow$	$Verb$
	|	$Verb\ NP$
	|	$Verb\ NP\ PP$
	|	$Verb\ PP$
PP	$\rightarrow$	$Preposition\ NP$

Repeated work in parsing-as-search

- Parsing: “a flight from Indianapolis to Houston on TWA” (Figure 13.7)

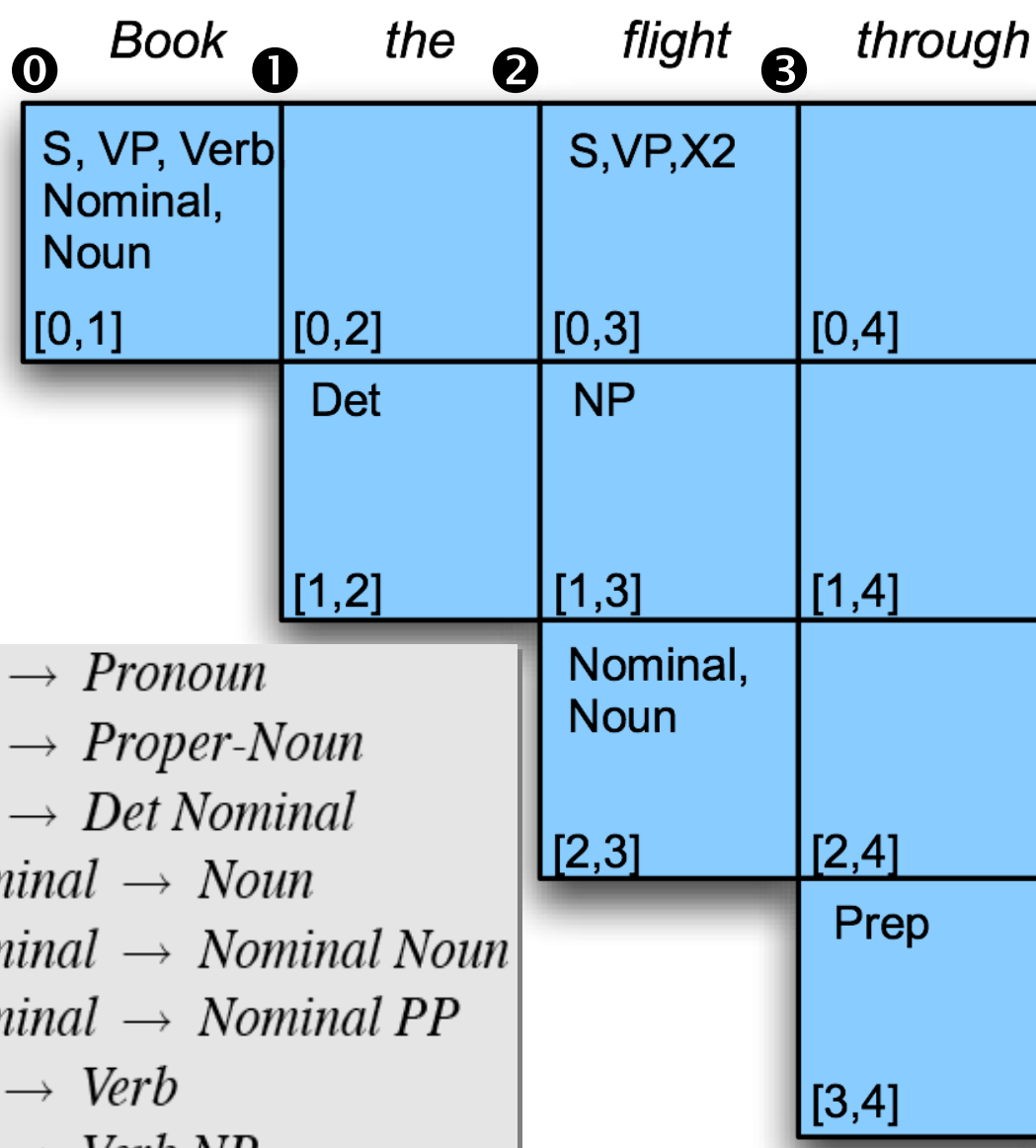


Dynamic programming parsing methods

- Parsing-as-search may have exponentially many parse states
- Dynamic programming solves the problem of doing repeated work
  - Memorization (remembering solved subproblems)
- Bottom-up approach: **CKY parsing**
  - Parse table
- Top-down approach: **Earley parsing**
  - Dotted rules

Parse table

- Cell $[i, j]$ contains the syntactic structures of the substring from the $(i + 1)$-th word to the j-th word

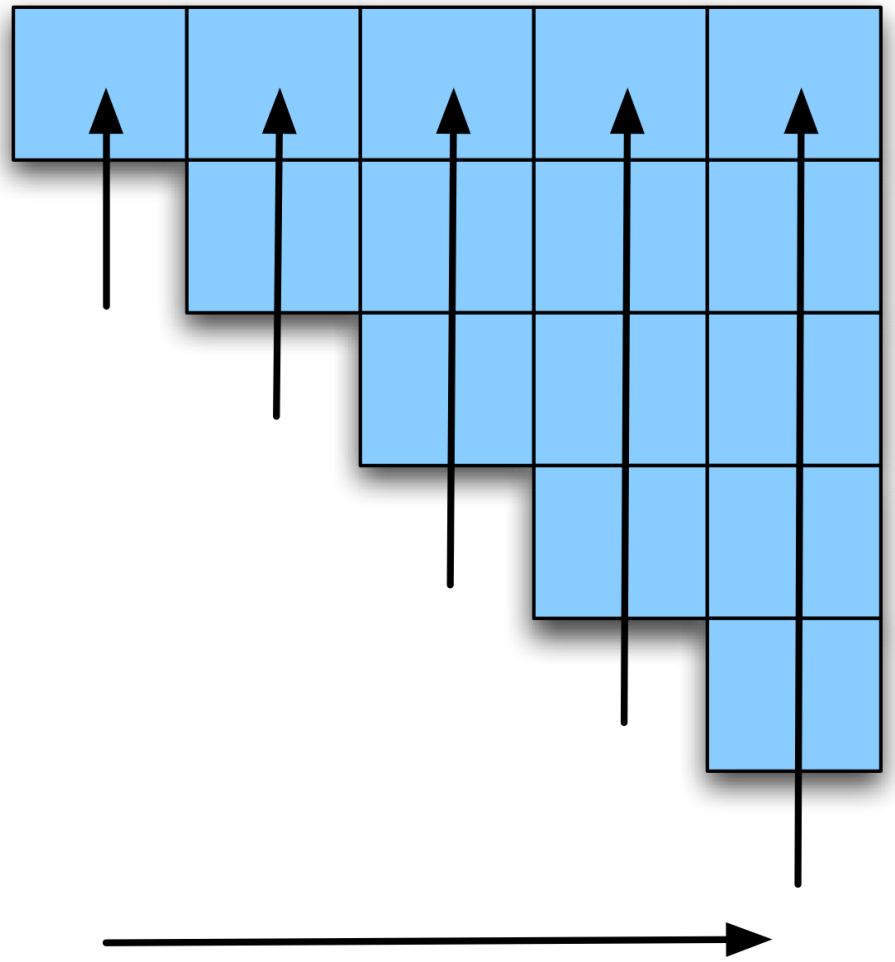


$\mathscr{L}_1$ Grammar
$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$

Parse table: Parsing sequence

Book	*the*	*flight*	*through*	*Houston*
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]



Parse table: A fast method of filling in each cell

0 Book 1	1 the 2	2 flight 3	3 through 4	4 Houston 5
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Assume an RHS has
exactly two items (X → Y Z)

[book$_1$] [the$_2$ flight$_3$ through$_4$ Houston$_5$]
[0,1] [1,5]
[book$_1$ the$_2$] [flight$_3$ through$_4$ Houston$_5$]
[0,2] [2,5]
…
[book$_1$ the$_2$ flight$_3$ through$_4$] [Houston$_5$]
[0,4] [4,5]

[0,5] → [0,k] [k,5] for 0<k<5

Parse table: A fast method of filling in each cell

NOT assume an RHS has exactly two items

[book$_1$] [the$_2$] [flight$_3$] [through$_4$] [Houston$_5$]

[0,1] [1,2] [2,3] [3,4] [4,5]

[book$_1$ the$_2$] [flight$_3$] [through$_4$] [Houston$_5$]

[0,2] [2,3] [3,4] [4,5]

[book$_1$] [the$_2$ flight$_3$] [through$_4$] [Houston$_5$]

[0,1] [1,3] [3,4] [4,5]

…

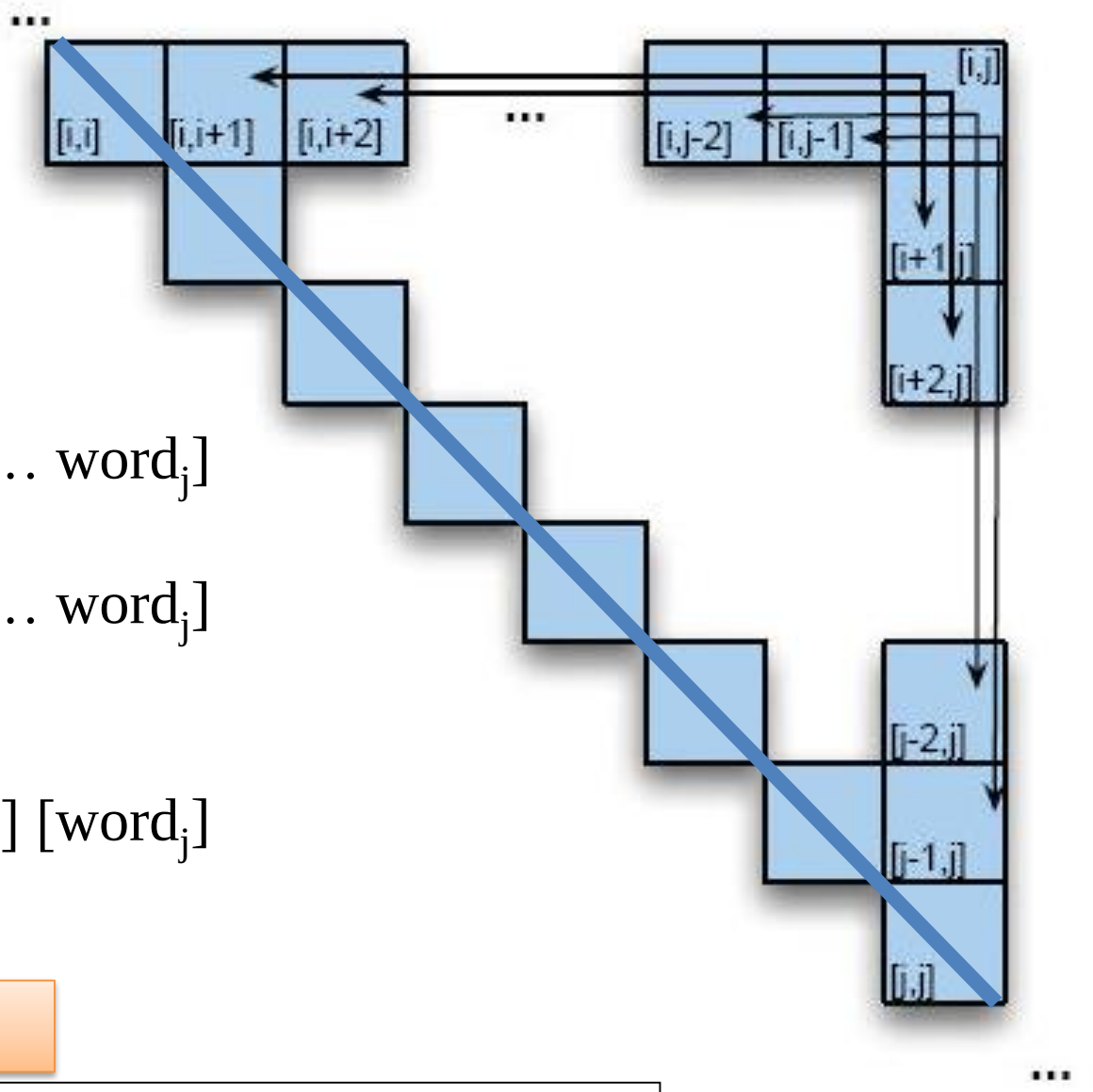
[book$_1$ the$_2$ flight$_3$ through$_4$] [Houston$_5$]

[0,4] [4,5]

[0,5]
all possible
combinations
$2^4 = 16$
cf. 4 binary
combinations

Parse table:

Assume an RHS
has exactly two items



[word$_{i+1}$] [word$_{i+2}$ word$_{i+3}$ … word$_j$]

[i,i+1] [i+1,j]

[word$_{i+1}$ word$_{i+2}$] [word$_{i+3}$ … word$_j$]

[i,i+2] [i+2,j]

…

[word$_{i+1}$ word$_{i+2}$ word$_{i+3}$ …] [word$_j$]

[i,j-1] [j-1,j]

[i,j] → [i,k] [k,j] for i<k<j

How can we create a CFG following the assumption?

CKY parsing

- Requirement
  - All rules should be in Chomsky normal form (CNF)

- CNF rules have the form of either
  - $A \rightarrow B\ C$ or $A \rightarrow w$ (w is a terminal)
  - RHS must have two non-terminals or one terminal

POS Tagging

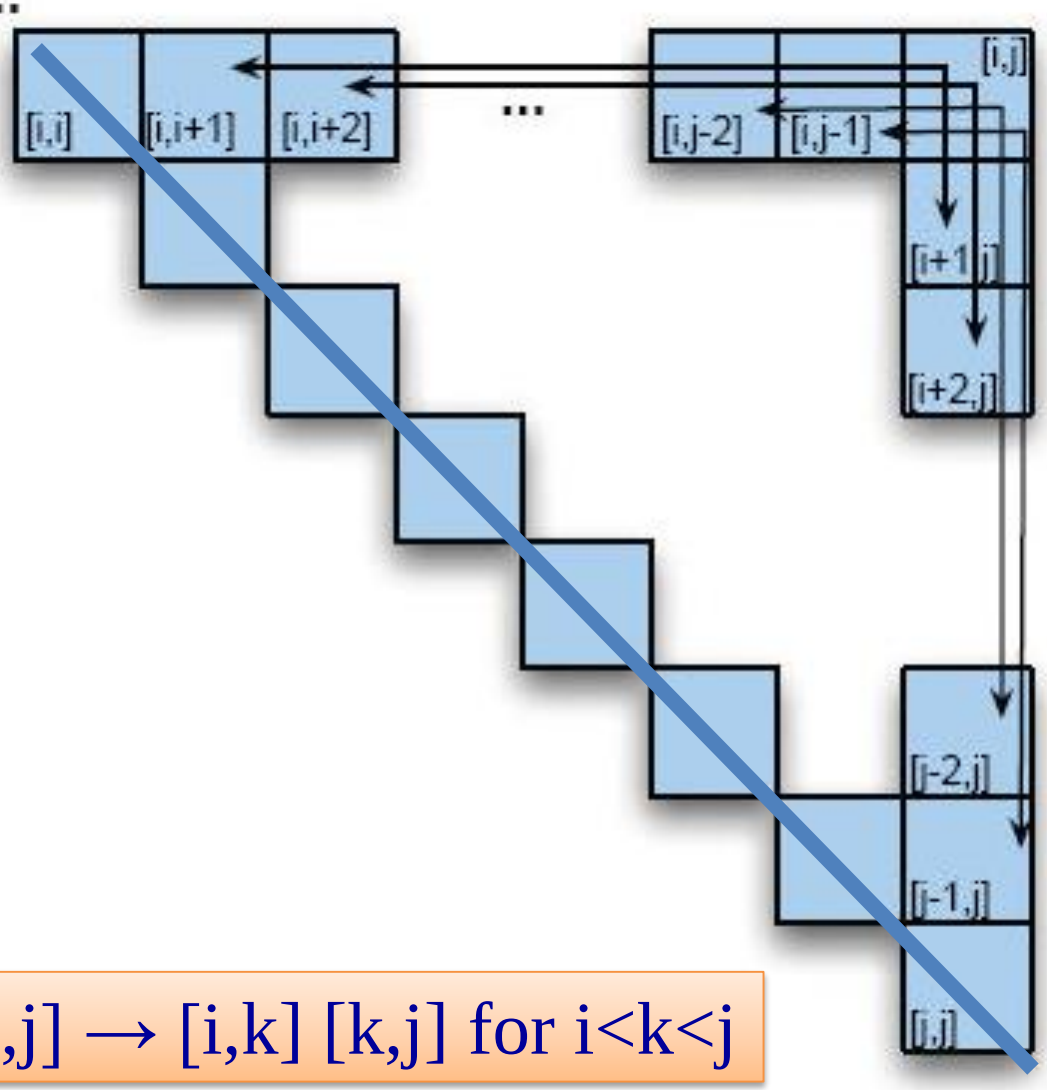
- Conversion to CNF
  - Read Section 13.4.1

CNF grammar: example

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

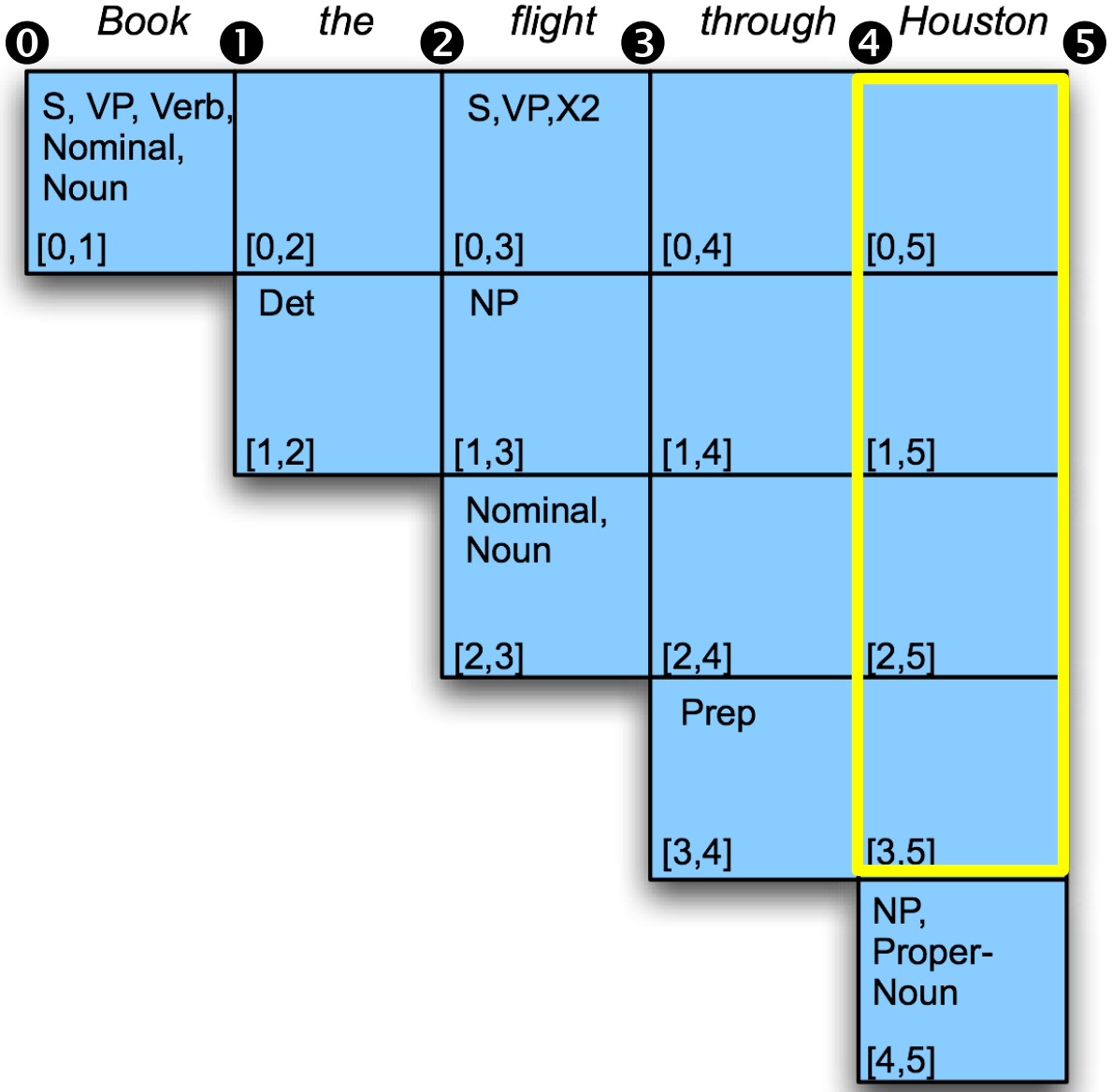
CKY parsing for [i,j] cell

- CNF rule for non-terminal
  - $A \rightarrow B\ C$
- $W_{i+1}, W_{i+2}, \ldots, W_j$
  - $[W_{i+1}], [W_{i+2}, \ldots, W_j]$
    - $[i,j] \rightarrow [i,i+1]\ \ [i+1,j]$
  - $[W_{i+1}, W_{i+2}], [\ldots, W_j]$
    - $[i,j] \rightarrow [i,i+2]\ \ [i+2,j]$
  - …
  - $[W_{i+1}, W_{i+2}, \ldots, W_{j-1}], [W_j]$
    - $[i,j] \rightarrow [i,j-1]\ \ [j-1,j]$



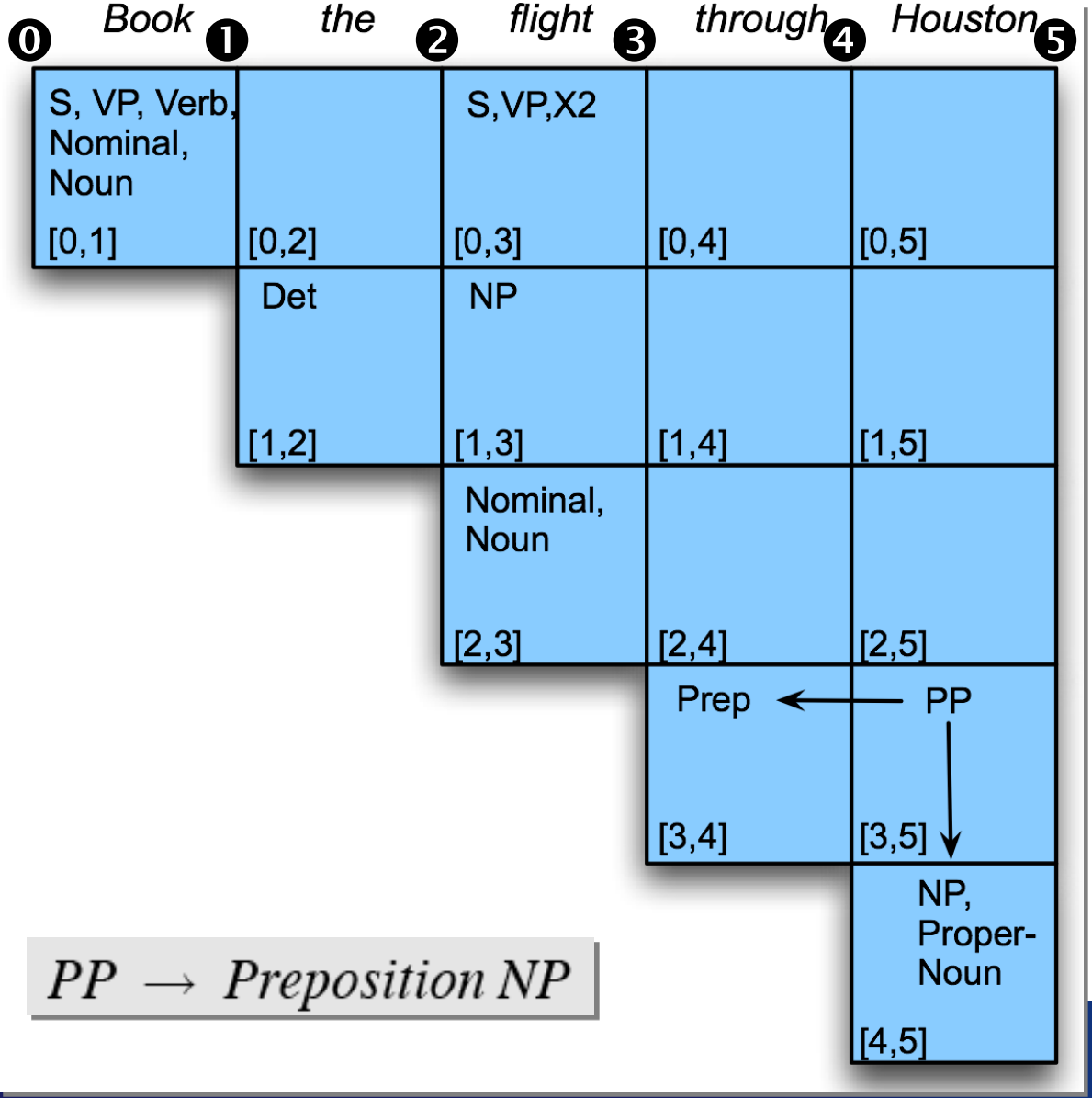
$[i,j] \rightarrow [i,k]\ [k,j]$ for $i<k<j$

Example : filling the 5th column



$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

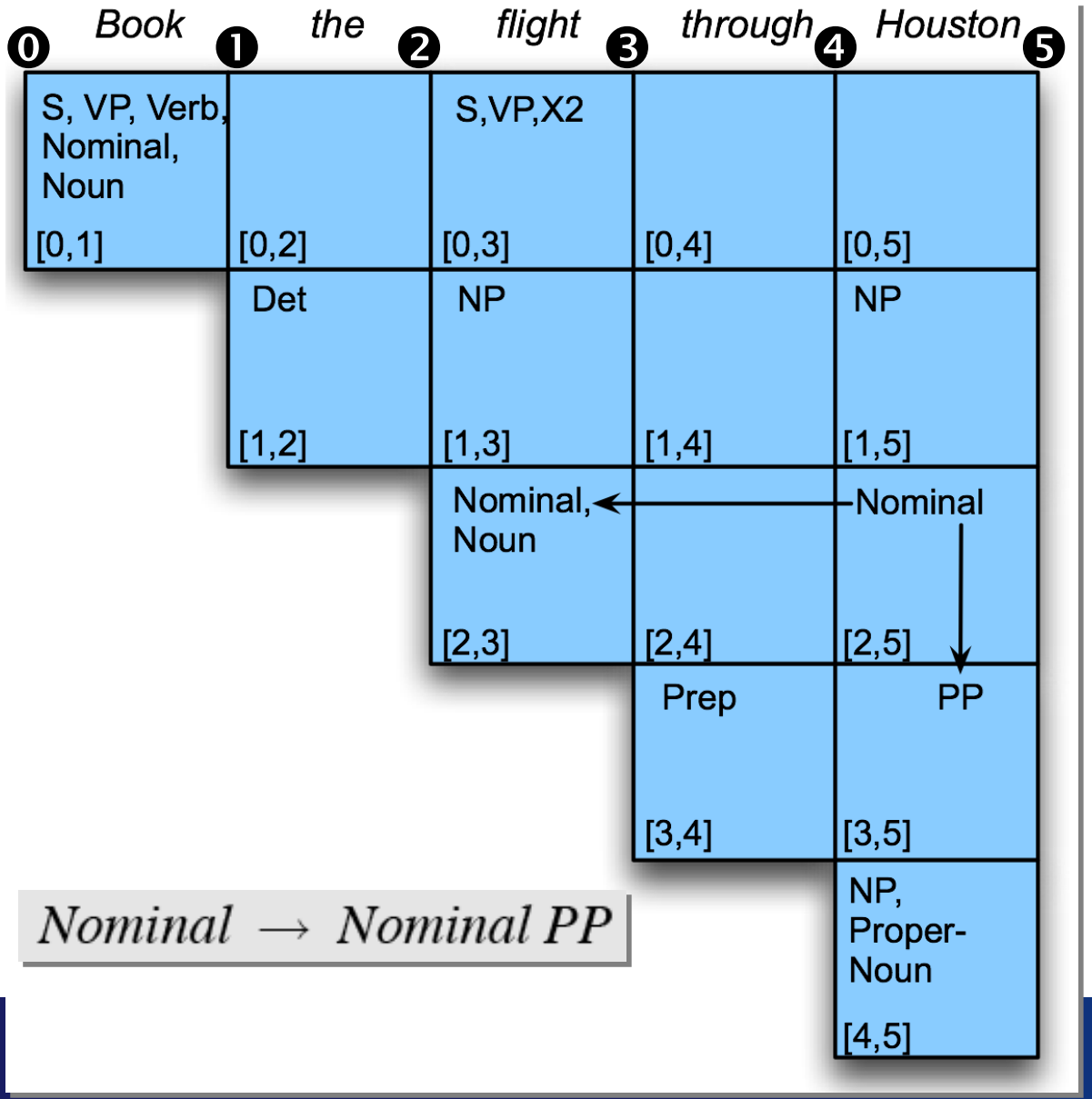
Example : filling the 5th column



$PP \rightarrow Preposition\ NP$

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

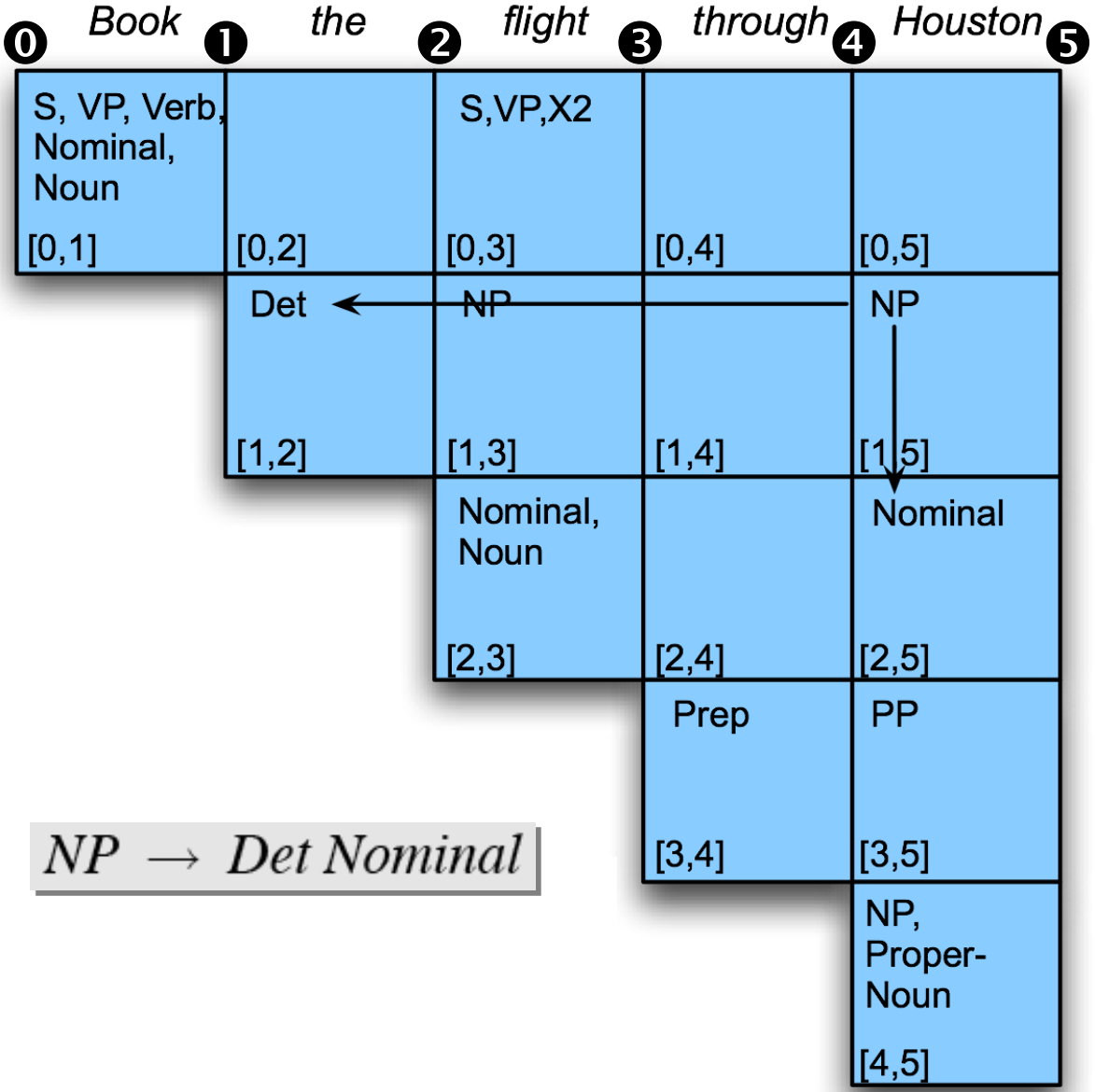
Example : filling the 5th column



$Nominal \rightarrow Nominal\ PP$

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

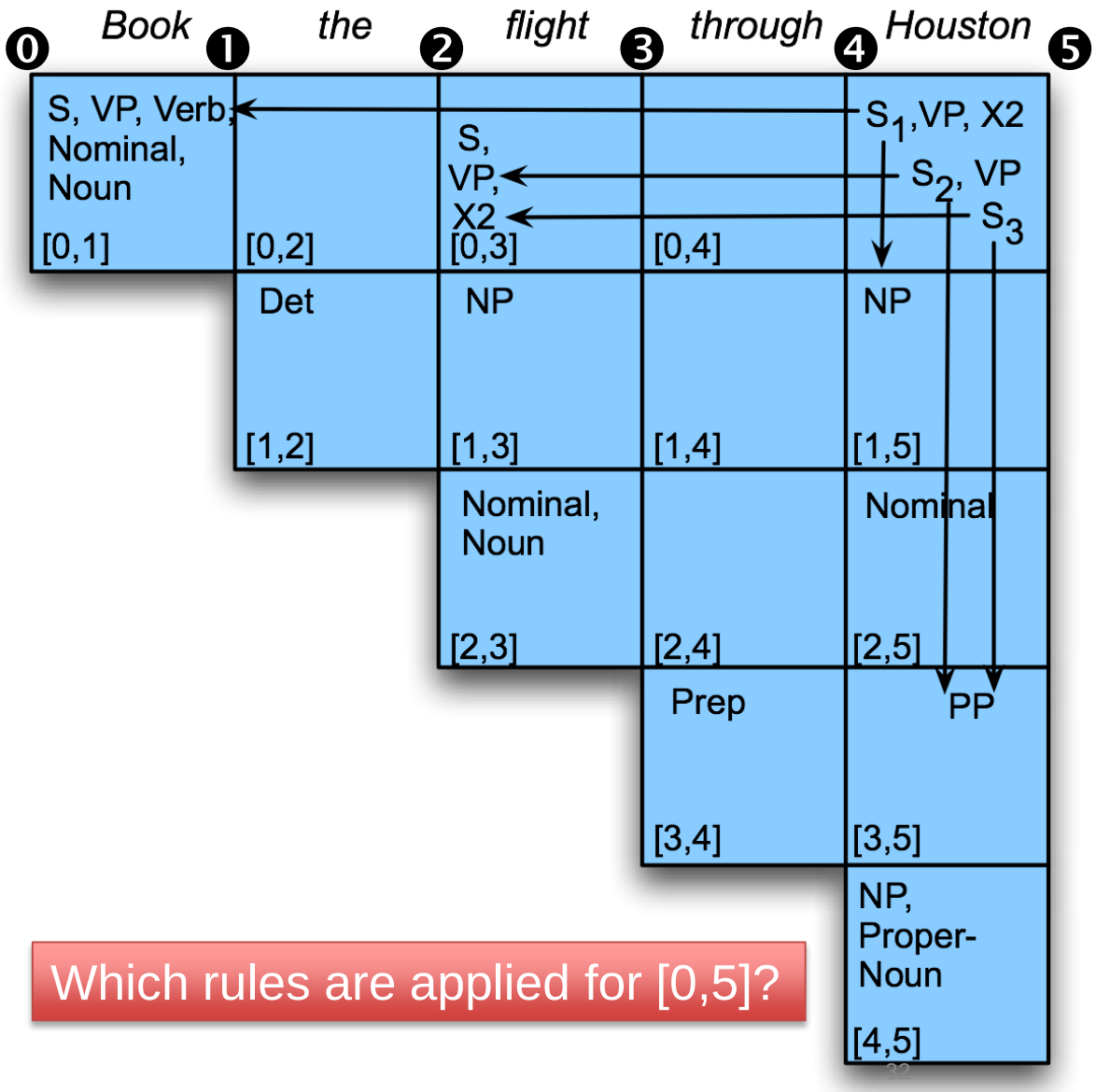
Example : filling the 5th column



$NP \rightarrow Det\ Nominal$

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Exercise: CKY parsing



Which rules are applied for [0,5]?

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

CKY algorithm

0 Book	1 the	2 flight	3 through	4 Houston 5
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

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

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k],$
          $C \in table[k,j]\}$

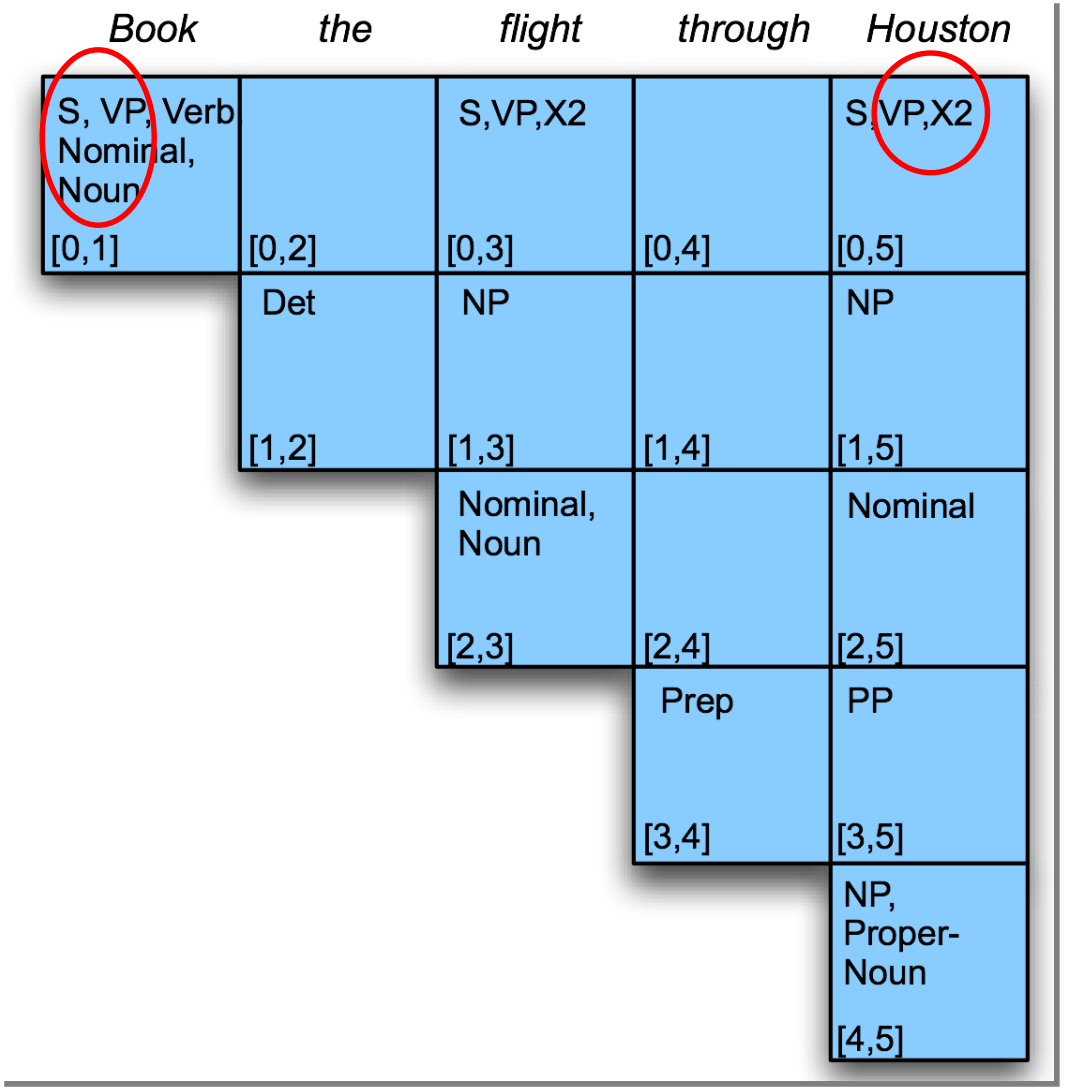
CKY parsing: summary

- Requirement: CNF grammar
  - Binarization: [i,j] → [i,k] [k,j] for $i < k < j$

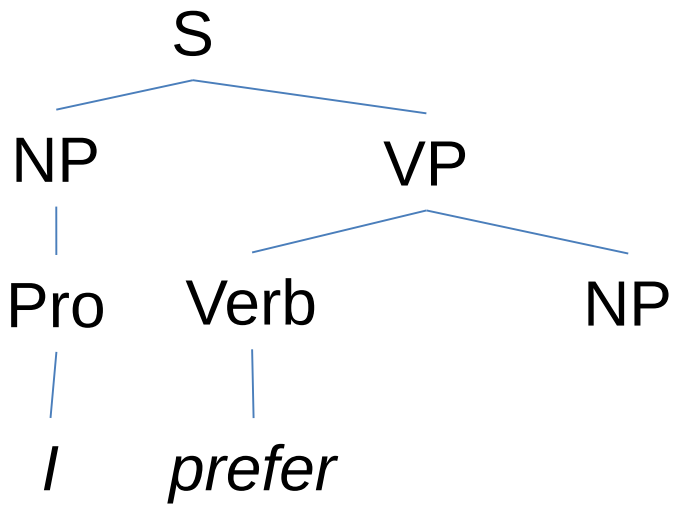
- Bottom-up approach
  - Process from [j-1,j] to [0,j]
  - This assures us that whenever we're filling a cell, the parts needed to fill it are already in the table (to the left and below)
  - It's somewhat natural in that it processes the input, from left to right, a word at a time

CKY parsing: Issue

- Trees that have no hope of leading to an S are generated
  - To avoid this we can switch to a top-down control strategy
  - Or we can add some kind of filtering that blocks constituents where they cannot happen in a final analysis.



Top-down parsing: intermediate state example



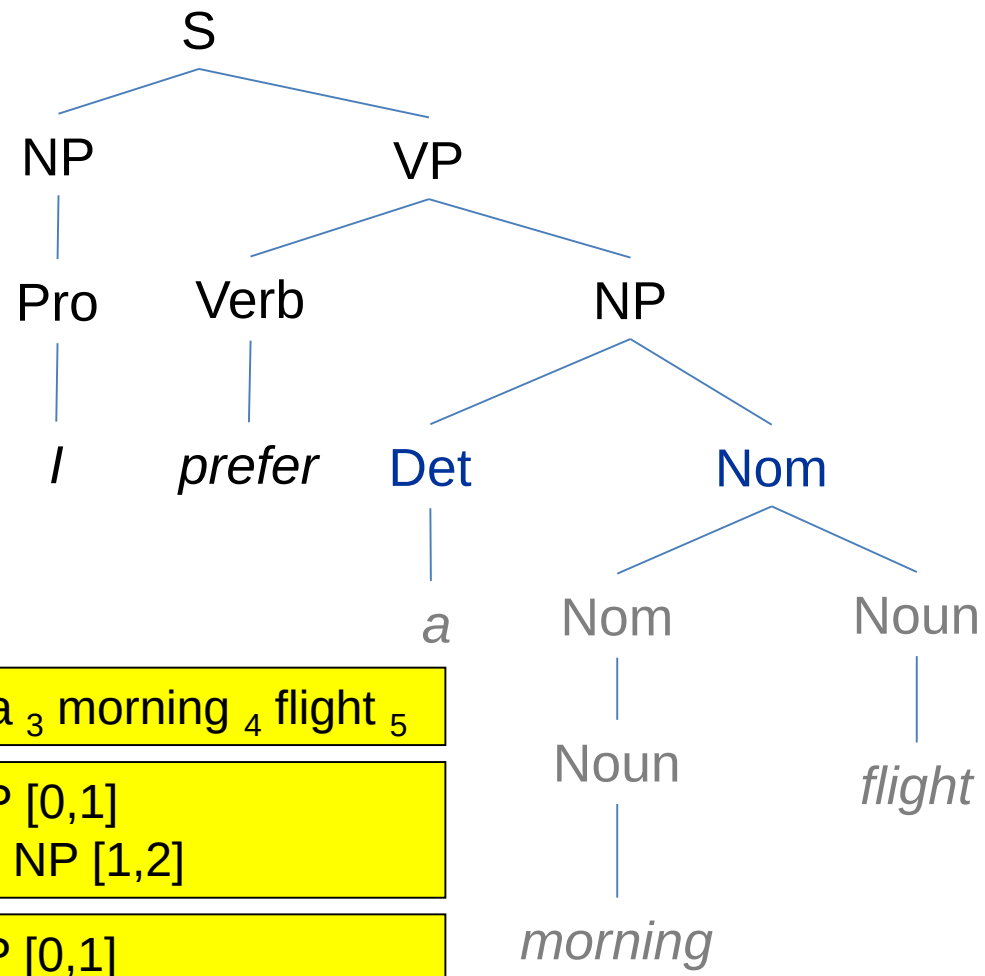
1. S → NP VP
2. NP → Pro
   Pro → *I*
3. VP → Verb NP
   Verb → *prefer*

$_0$ I $_1$ prefer $_2$ a $_3$ morning $_4$ flight $_5$

Dotted rules

S → NP • VP [0,1]
VP → Verb • NP [1,2]

Top-down parsing: Next state



1. S → NP VP

2. NP → Pro
   Pro → *I*

3. VP → Verb NP
   Verb → *prefer*

4. NP → Det Nom
   Det → a

5. Nom → Nom Noun
   Noun → *flight*

6. Nom → Noun
   Noun → *morning*

$_0$ I $_1$ prefer $_2$ a $_3$ morning $_4$ flight $_5$

S → NP • VP [0,1]
VP → Verb • NP [1,2]

S → NP • VP [0,1]
VP → Verb • NP [1,2]
NP → • Det Nom [2,2]

Earley algorithm: States

$_0$ I $_1$ prefer $_2$ a $_3$ morning $_4$ flight $_5$

- NP → • Det Nom [2,2]
  - A Det is predicted at position 2
- VP → Verb • NP [1,2]
  - A VP is in progress; the Verb goes from 1 to 2
- VP → Verb NP • [1,4]
  - A VP has been found, starting at 1 and ending at 4
- S → α • [0,N]
  - Parsing is finished

Earley algorithm: How it works

1. Predict all the states you can upfront [**Predictor**]

2. Read a word [**Scanner**]
   - Extend states based on matches [**Completer**]
   - Generate new predictions
   - Repeat step 2

3. When you're out of words, look at the chart to see if you have a winner

Earley algorithm: Example (1) $_0$ Book $_1$ that $_2$ flight $_3$

S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor

$\mathcal{L}_1$ Grammar
$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Note that given a grammar, these entries are the same for all inputs; they can be pre-loaded.

Earley algorithm: Example (2) $_0$ Book $_1$ that $_2$ flight $_3$

S12	*Verb* → *book* •	[0,1]	Scanner
S13	*VP* → *Verb* •	[0,1]	Completer
S14	*VP* → *Verb* • *NP*	[0,1]	Completer
S15	*VP* → *Verb* • *NP PP*	[0,1]	Completer
S16	*VP* → *Verb* • *PP*	[0,1]	Completer
S17	*S* → *VP* •	[0,1]	Completer
S18	*VP* → *VP* • *PP*	[0,1]	Completer
S19	*NP* → • *Pronoun*	[1,1]	Predictor
S20	*NP* → • *Proper-Noun*	[1,1]	Predictor
S21	*NP* → • *Det Nominal*	[1,1]	Predictor
S22	*PP* → • *Prep NP*	[1,1]	Predictor

$\mathcal{L}_1$ Grammar
S → *NP VP*
S → *Aux NP VP*
S → *VP*
NP → *Pronoun*
NP → *Proper-Noun*
NP → *Det Nominal*
Nominal → *Noun*
Nominal → *Nominal Noun*
Nominal → *Nominal PP*
VP → *Verb*
VP → *Verb NP*
VP → *Verb NP PP*
VP → *Verb PP*
VP → *VP PP*
PP → *Preposition NP*

Note that "Noun → book• [,] can be also scanned

Earley algorithm: Example (3) $_0$ Book $_1$ that $_2$ flight $_3$

S23	$Det \rightarrow that \bullet$	[1,2]	Scanner
S24	$NP \rightarrow Det \bullet Nominal$	[1,2]	Completer
S25	$Nominal \rightarrow \bullet Noun$	[2,2]	Predictor
S26	$Nominal \rightarrow \bullet Nominal\ Noun$	[2,2]	Predictor
S27	$Nominal \rightarrow \bullet Nominal\ PP$	[2,2]	Predictor
S28	$Noun \rightarrow flight \bullet$	[2,3]	Scanner
S29	$Nominal \rightarrow Noun \bullet$	[2,3]	Completer
S30	$NP \rightarrow Det\ Nominal \bullet$	[1,3]	Completer
S31	$Nominal \rightarrow Nominal \bullet Noun$	[2,3]	Completer
S32	$Nominal \rightarrow Nominal \bullet PP$	[2,3]	Completer
S33	$VP \rightarrow Verb\ NP \bullet$	[0,3]	Completer
S34	$VP \rightarrow Verb\ NP \bullet PP$	[0,3]	Completer
S35	$PP \rightarrow \bullet Prep\ NP$	[3,3]	Predictor
S36	$S \rightarrow VP \bullet$	[0,3]	Completer
S37	$VP \rightarrow VP \bullet PP$	[0,3]	Completer

$\mathcal{L}_1$ Grammar
$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Earley algorithm: Summary (Pseudo codes: Section 13.4.2)

- Top-down approach
  - Breadth-first search
  - State representation
    - (compare with the cells of parse tree in CKY algorithm: subtrees)

- Waste lots of time in trying inconsistent rule applications

S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
S33	*VP* → *Verb NP* •	[0,3]	Completer
S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
S35	*PP* → • *Prep NP*	[3,3]	Predictor
S36	*S* → *VP* •	[0,3]	Completer
S37	*VP* → *VP* • *PP*	[0,3]	Completer

Top-down vs. Bottom-up parsing

Review

- Top-down
  - Waste lots of time in trying inconsistent rule applications
    - The application of the rules does not lead the generation of the given string
  - Never explore subtrees that cannot find a place in some S-rooted tree
    - All trees are generated starting with S.
- Bottom-up
  - Never suggest trees that are not grounded in the input string
    - All trees are generated based on input string
  - Trees that have no hope of leading to an S are generated
    - Some trees cannot proceed further to reach S

Soundness and completeness

- A parser is *sound* if every parse result it returns is valid/correct
- A parser *terminates* if it is guaranteed to not go off into an infinite loop
- A parser is *complete* if for any given grammar and string,
  - It is sound
  - It produces all valid parse results for the string, and
  - It terminates
- For practical purposes, we settle for sound and terminating, but incomplete parsers
  - E.g. *k*-best parse results

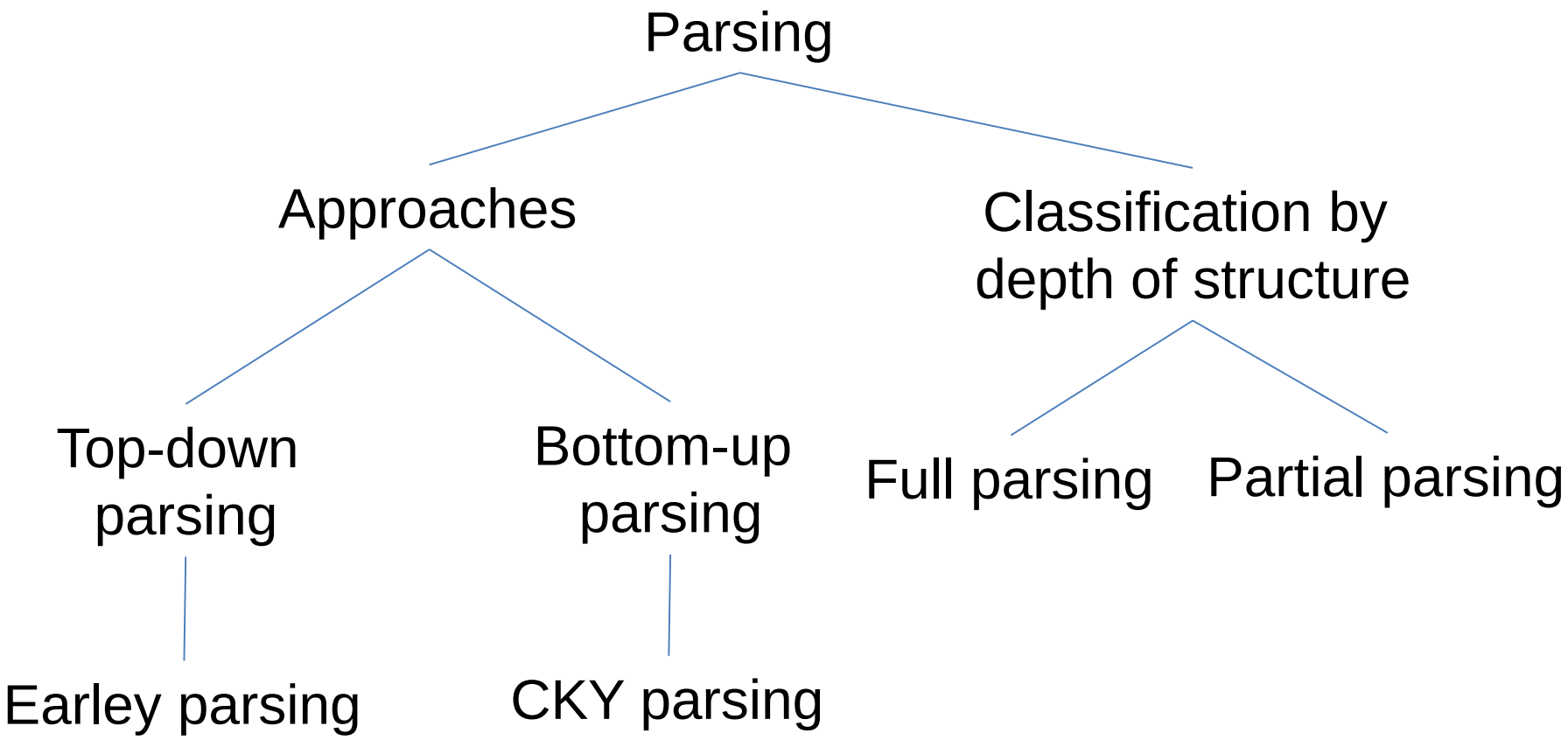
Full parsing vs. Partial parsing

- Full parsing
  - Identify the complete syntactic structure of a sentence
  - Oftentimes, parsing is the most time-consuming part

- Partial parsing
  - Identify parts of the syntactic structure of a sentence
  - Not all applications require full syntactic structures. Example: only Noun phrases need to be extracted in some applications but not the full syntactic structure of the sentence.

- Read Section 13.5

Summary


Parsing
Approaches
Classification by depth of structure
Top-down parsing
Bottom-up parsing
Full parsing
Partial parsing
Earley parsing
CKY parsing