AI6122 Text Data Management & Analysis

Topic: Parsing

Syntax

- **The way words are arranged together**
  - Implicit knowledge of your native language that you had mastered by the time you were young without explicit instruction
  - Not the kind of stuff you were later taught in “grammar” school

- Why should we care?
  - Syntax is the key concept for many NLP tasks
    - E.g. information extraction, grammar checking
  - An **approximation** would be still useful

Key notions

- Constituency
  - Groups of words that behave as **a single unit** or **phrase**
  - E.g., a few words form a noun phrase

- Grammatical relations
  - Syntactic relations between constituents
  - E.g., Subject – verb, verb – object

- Subcategorization
  - Certain kinds of relation between words and phrases
    - transitive words, which can take a direct object
    - intransitive words which cannot take a direct object

Constituent

- Groups of words that behave as a single unit or phrase
  - Can all appear in similar syntactic environments

- Example: Noun phrases (NPs) can occur before verbs
  - Three example sentences starting with NPs
    - <u>three parties from Brooklyn</u> *arrive* …
    - <u>a high-class spot such as Mindy's</u> *attracts* …
    - <u>the Broadway coppers</u> *love* …

  - While the whole noun phrase can occur before a verb, it is not true of each of the individual words that make up a noun phrase

Constituent

- Evidence for constituency
  - Stick together through pre-/post-posed constructions (example using a prepositional phrase)
    - <u>On September seventeenth</u>, I'd like to fly from Atlanta to Denver.
    - I'd like to fly <u>on September seventeenth</u> from Atlanta to Denver.
    - I'd like to fly from Atlanta to Denver <u>on September seventeenth</u>.
  - Individual words making of the phrase cannot be placed differently.
    - *<u>On</u>, I'd like to fly from Atlanta to Denver <u>September seventeenth</u>.
- Non-standard constructions (or constituents)
  - I gave John a watch and <u>Mary a doll</u>.
  - *I gave John a watch <u>Mary</u> and <u>a doll</u>.

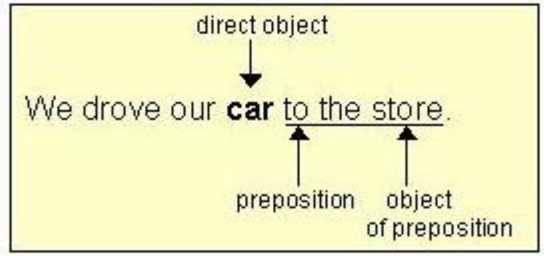
For your information only (details in textbook)

- Noun phrase (NP), verb phrase (VP), prepositional phrase (PP), adjective phrase

- Declarative, imperative, clause
  - Declarative sentences have a subject noun phrase followed by a verb phrase. “I prefer a morning flight”
  - Imperative sentences begin with a verb and have no subject. “Show me the code”.
- Determiner, nominal, quantifier
  - Noun phrases can begin with determiners or quantifier. “a bus stop” “a non-stop flight”
- Auxiliaries, passive, perfect, progressive
  - Auxiliaries include modal verb can could shall, perfect auxiliary have, progressive auxiliary be and passive auxiliary be
- Coordination, conjunction
  - A coordinate noun phrase can consist of two noun phrases by a conjunction: “course code and course title”

Grammatical relations

- Syntactic relations **between constituents**
  - Subject-verb: I love him.
  - Verb-object: I love him.
  - Verb-PrepositionalPhrase: I study at the library.
  - Noun-PrepositionalPhrase: The history of Singapore
  - Preposition-object



https://webapps.towson.edu/ows/prepositions.htm

Verb phrases (VPs): Sample patterns

- Verb
  - disappear
  - *prefer
- Verb Object
  - prefer a morning flight
  - *disappear a morning flight
- Verb IObject DObject (indirect object, direct object)
  - give John a watch
  - *prefer John a watch
- Verb Object PrepositionalPhrase(PP)
  - leave Boston in the morning
- Verb PP
  - leave on Thursday

Subcategorization

- 'Subcategories' of different types of verbs, by the constituents complement them

  - Verb "disappear"
  - Verb NP "prefer a morning flight"
  - Verb NP NP "give John a watch"
  - Verb NP PP "leave Boston in the morning"
  - Verb PP "leave on Thursday"

- Subcategorization frame
  - A possible set of complements for a verb
    - e.g. 'give': {}, {NP}, {NP, NP}, {NP, PPto}, {PP}

  - Verb: function, complements: arguments
    - e.g. def buy(Person: buyer, Thing: product)

Syntactic structure

- Why do we learn about syntax in NLP course?
  - To identify and understand the syntactic structures of sentences

- **Parse tree**
  - Structural representation of grammatical relations expressed in a string

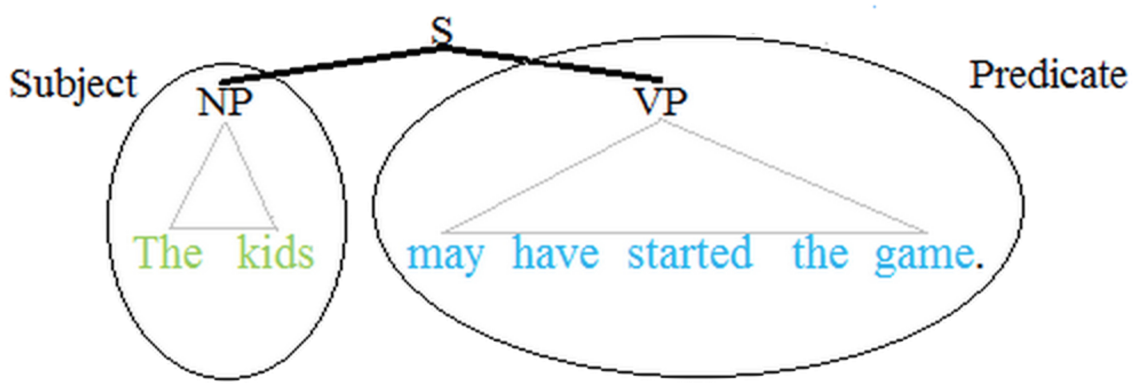
- Representations
  - Phrase structure
  - Dependency structure
  - Predicate-argument structure

Syntactic ≠ Semantic

- iPhone7 is nice but expensive
- iPhone7 is expensive but nice

Phrase Structure

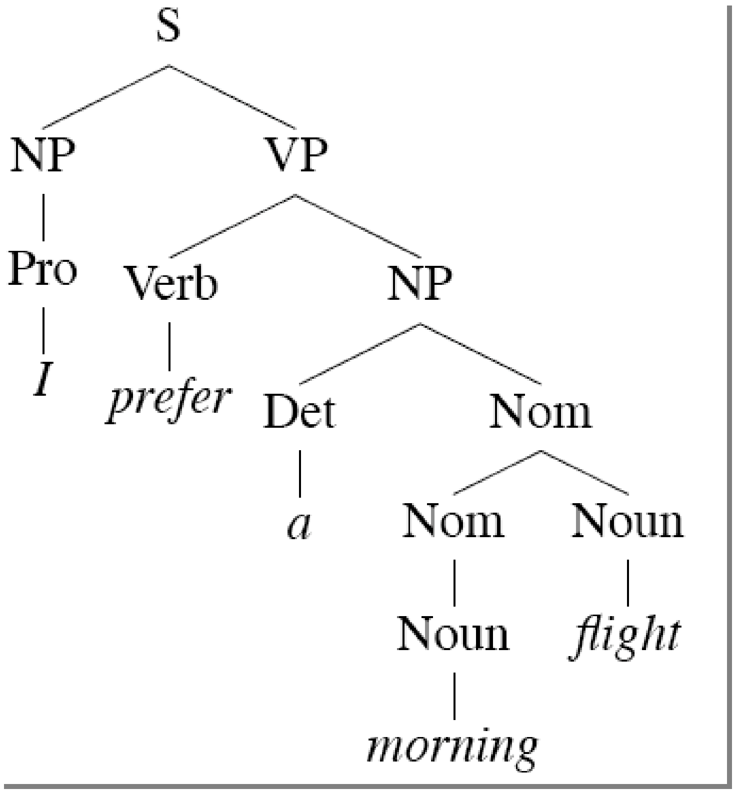
- Phrase structure rules are used to break down a natural language sentence into its constituent parts (also known as syntactic categories), including both lexical categories (parts of speech) and phrasal categories
  - Split a sentence (S) into subject (NP) and predicate (VP)
  - Split subject and predicate into smaller parts
  - Continue until terminal nodes are reached



Phrase Structure

- Organizes words into nested constituents
  - Example sentence: “I prefer a morning flight”

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



Nominal: (i) noun phrase without determiner; and
(ii) the noun that modifies the head noun in a noun phrase.

Exercise: Phrase structure

- Give the phrase structures of the two sentences:
  - (S1) John gives a present to Mary.
  - (S2) John gives Mary a present.

- POS tags:
  - E.g. ProperNoun: John, Mary
  - E.g. Verb, Det, Noun, Prep

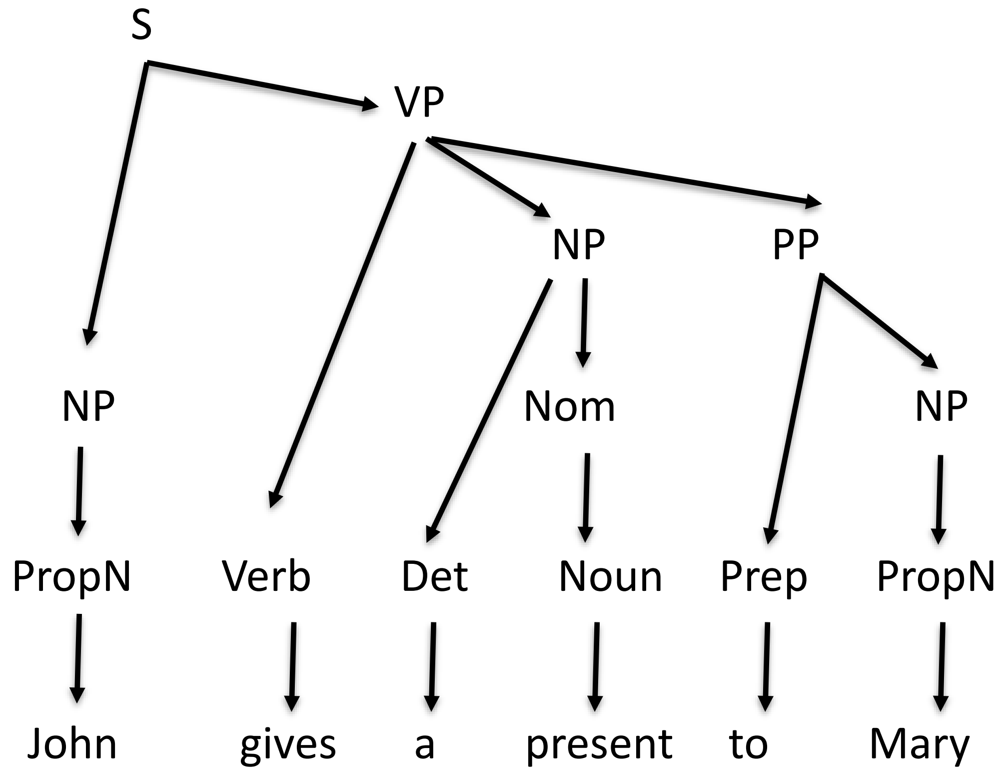
- Example syntactic tags
  - E.g. S, NP, VP, PP_{XX}

Exercise: Phrase structure

- Give the phrase structures of the two sentences:
  - (S1) John gives a present to Mary.
  - (S2) John gives Mary a present.

- POS tags:
  - E.g. ProperNoun: John, Mary
  - E.g. Verb, Det, Noun, Prep

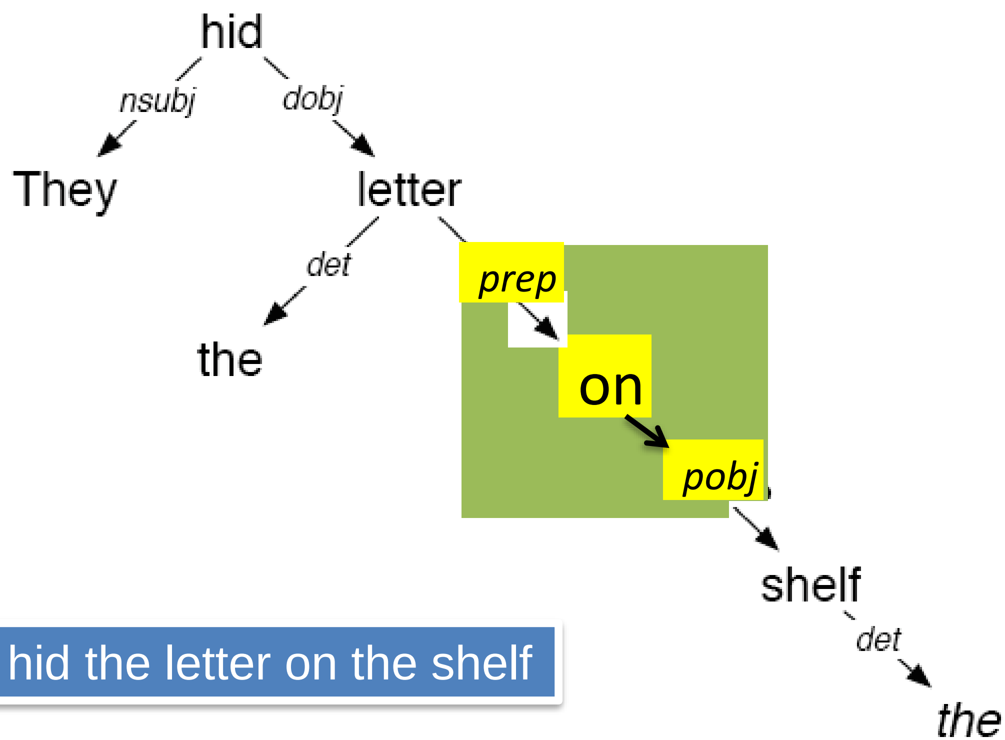
- Example syntactic tags
  - E.g. S, NP, VP, PP_{XX}



Dependency structure

- Represents grammatical (dependency) relations between pairs of words: head, dependent

- Head: Grammatically most important word in a phrase
  - Verb of VP
  - Noun of NP
  - Prep of PP
  - Adj of AdjP
  - …



They hid the the letter on the shelf

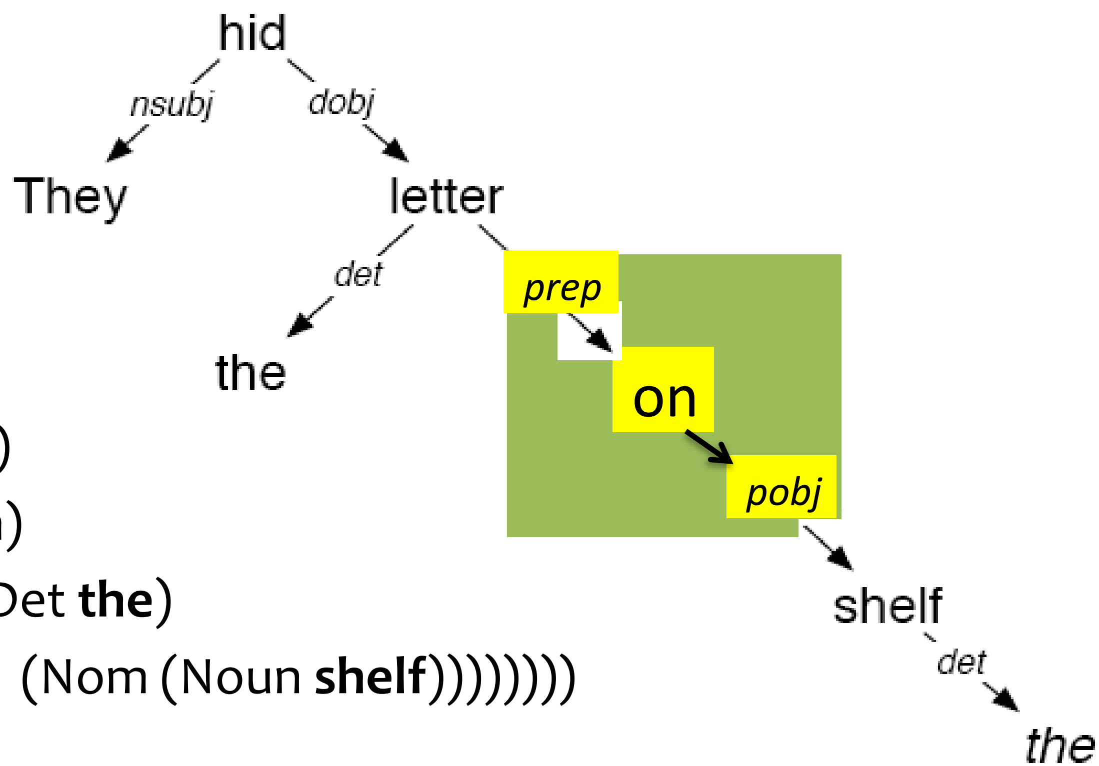
Dependency Relations

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

http://universaldependencies.org/u/dep/index.html

Phrase structure vs. Dependency structure

(S
(NP (Pro **They**))
(VP
(Verb **hid**)
(NP (Det **the**)
(Nom (Noun **letter**)
(PP (Prep **on**)
(NP (Det **the**)
(Nom (Noun **shelf**)))))))))

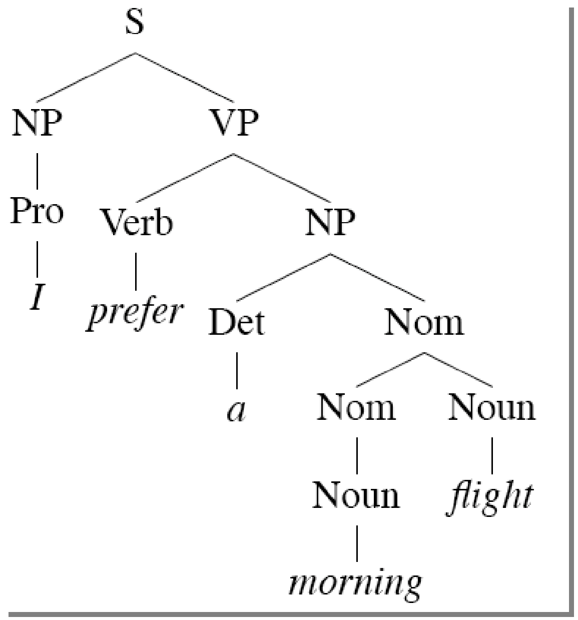


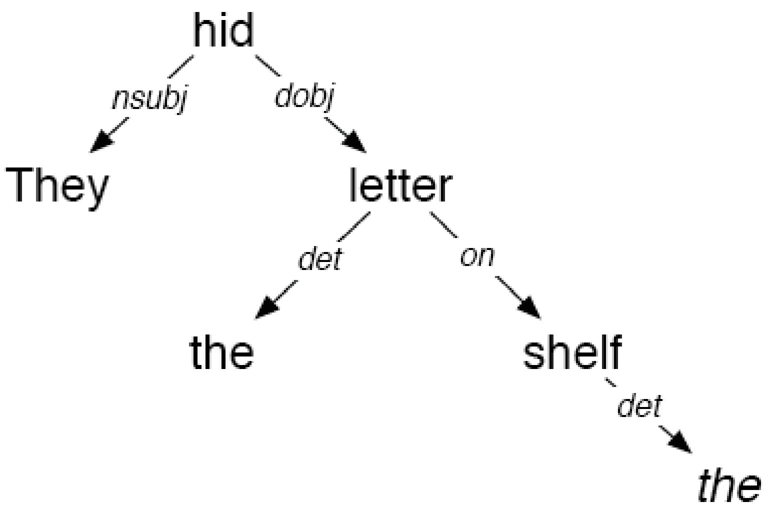
They hid the letter on the shelf

Phrase vs. dependency structure

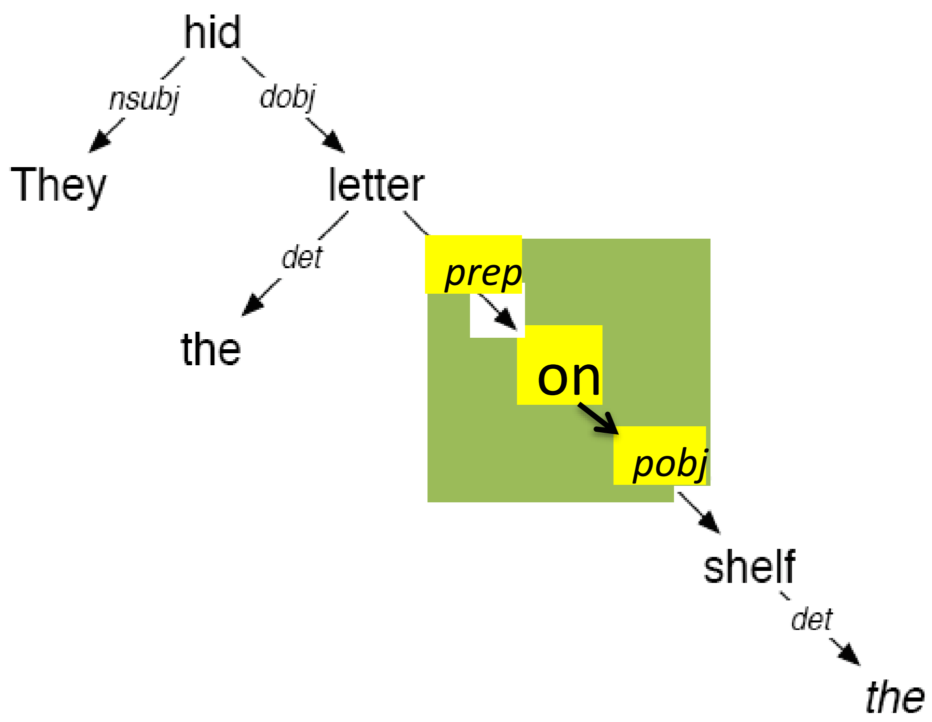
- Constituents are specified
- Grammatical relations are implied
  - Can be deduced from phrase structure

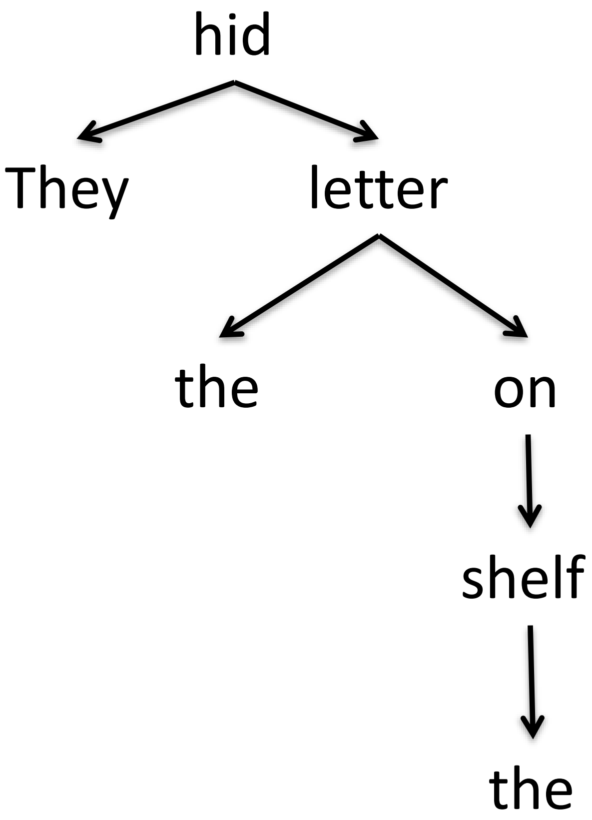
- Grammatical relations are specified
- A subtree corresponds to a constituent
  - Syntactic tag is implied by the root of subtree



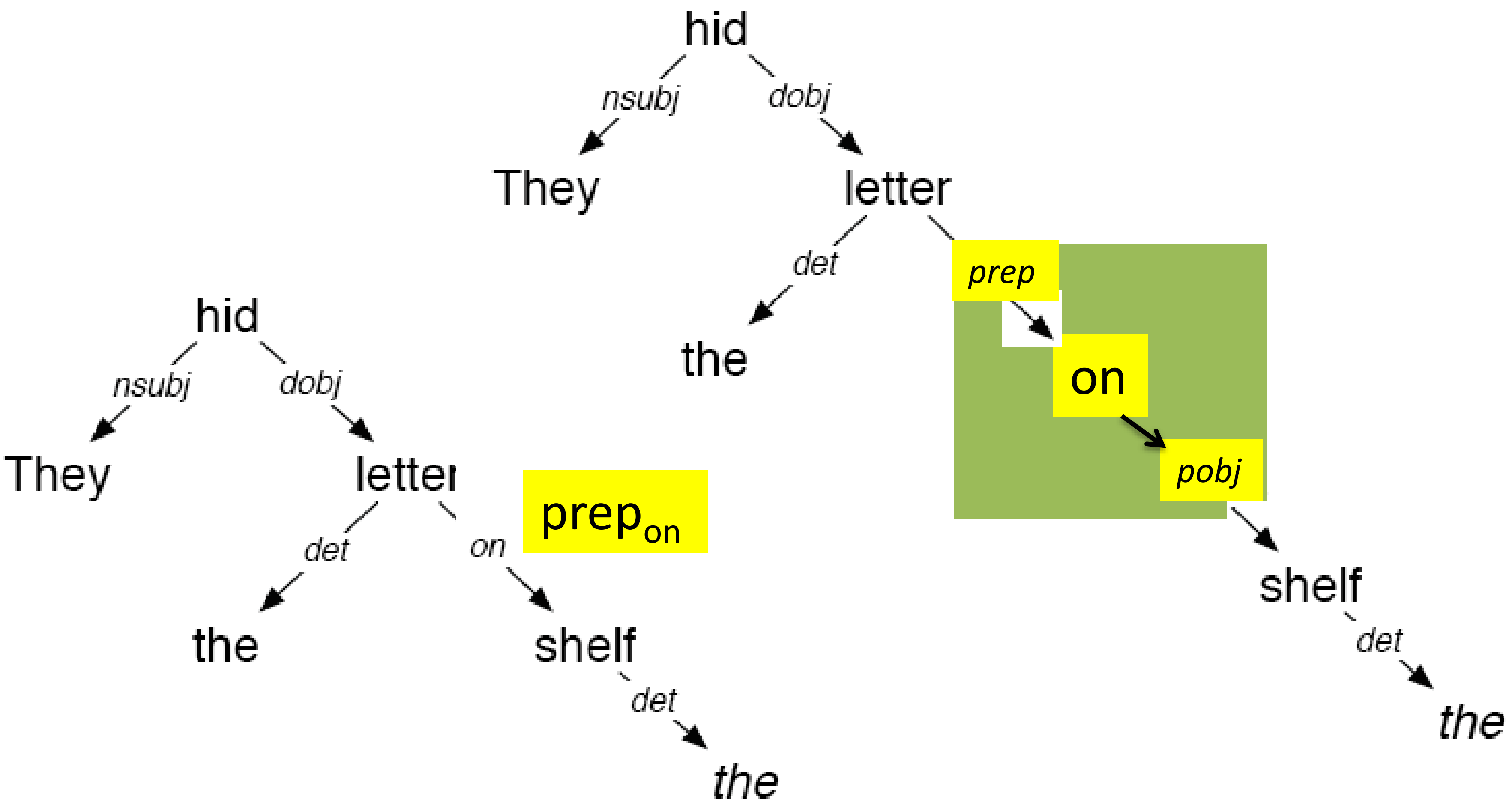


Typed dependency vs. untyped dependency





Typed dependency vs. Collapsed typed dependency



Preposition word followed by its object

Exercise: Dependency structure

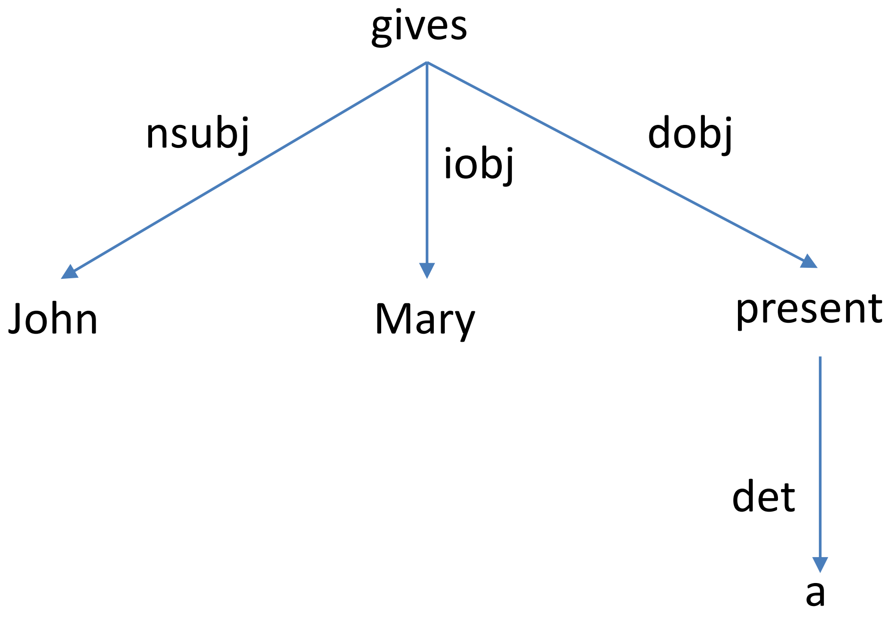
- Give the dependency structures of the two sentences:
  - (S1) John gives a present to Mary.
  - (S2) John gives Mary a present.

- Example dependency relations
  - E.g. nsubj, iobj, dobj, det, prep_{XX}

Exercise: Dependency structure

- Give the dependency structures of the two sentences:
  - (S1) John gives a present to Mary.
  - (S2) John gives Mary a present.

- Example dependency relations
  - E.g. nsubj, iobj, dobj, det, prep_{XX}



Predicate-argument structure

- Review: Subcategorization frame
  - Verb: function/predicate, complements: arguments
- Generalize to all types of words that have complements
  - E.g. Preposition: predicate, object: argument
  - E.g. They hid **the$_1$** letter on **the$_2$** shelf.

Relation type	Predicate	Argument 1	Argument 2
verb_arg12	hid	They	letter
det_arg1	the$_1$	letter	
prep_arg12	on	letter	shelf
det_arg1	the$_2$	shelf	

Dependency structure vs. Predicate-argument structure

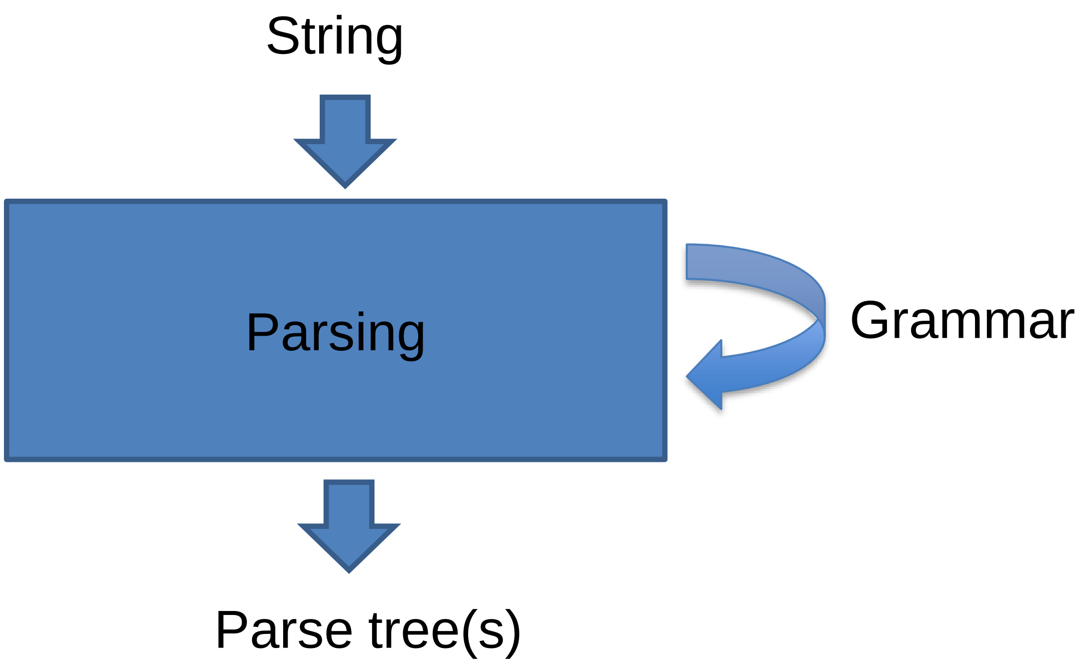
- Sentence: They hid the letter on the shelf

Dependency structure
root(ROOT-0, hid-2) nsubj(hid-2, They-1) dobj(hid-2, letter-4) det(letter-4, the-3) prep_on(letter-4, shelf-7) det(shelf-7, the-6)

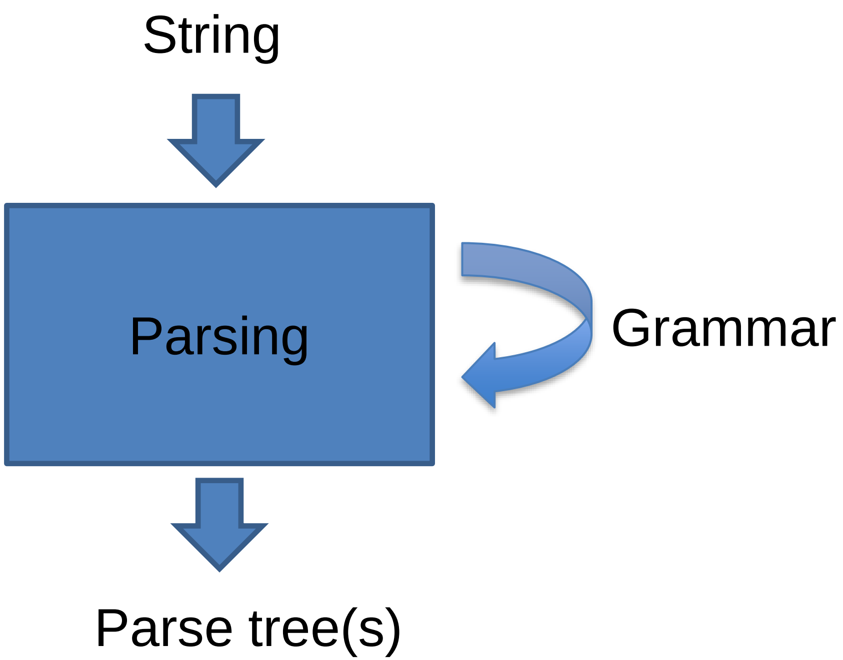
Predicate-argument structure
verb_arg12(hid-2, They-1, letter-4) det_arg1(the-3, letter-4) prep_arg12(on-6, letter-4, shelf-7) det_arg1(the-6, shelf-7)

Syntactic parsing

- The process of taking a string and a grammar and returning parse tree(s) for that string



Syntactic parsing in the course

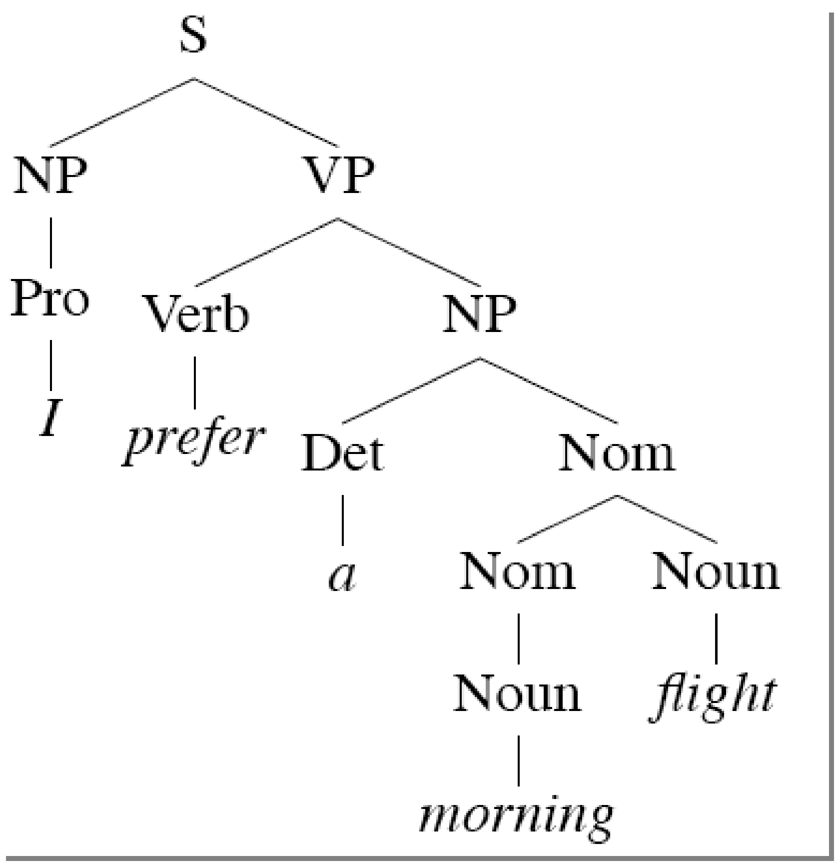


- G = (T, N, S, R)
  - T: a set of terminals (e.g. ‘flight’)
  - N: a set of non-terminals (e.g. Noun)
  - S: the start symbol, a non-terminal
  - R: rules of the form X →γ
    - X: a non-terminal
    - γ: a sequence of terminals and non-terminals

- Examples on next slide

Context-free grammar (1/2)

- 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



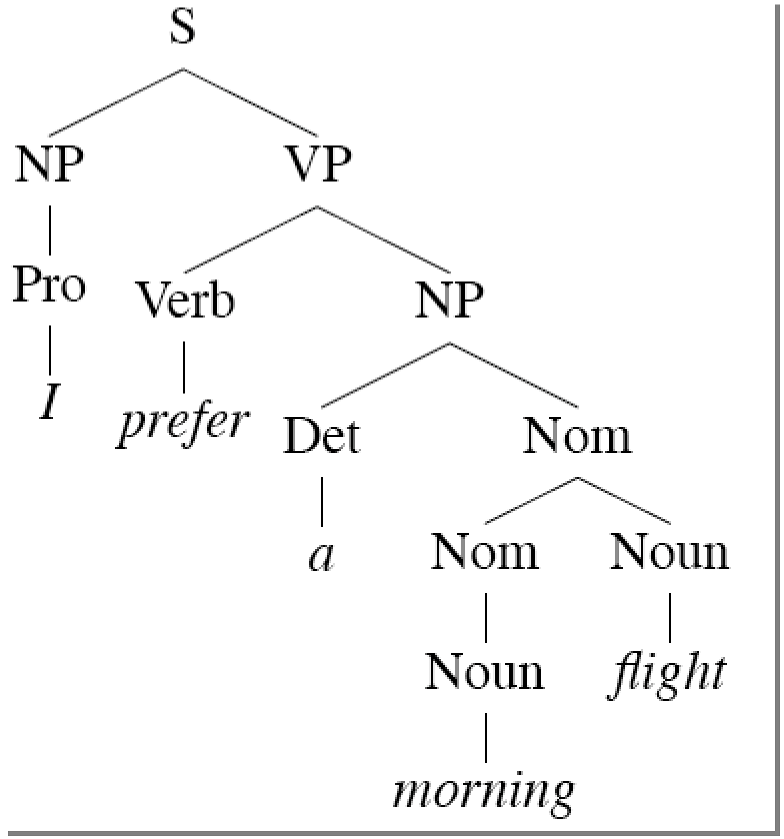
$$
\begin{array}{rcl}
NP & \rightarrow & Det\ Nominal \\
NP & \rightarrow & ProperNoun \\
Nominal & \rightarrow & Noun \mid Nominal\ Noun
\end{array}
$$

Context-free grammar (2/2)

- A grammar G **generates** a language L
  - **A language is a set of sentences**

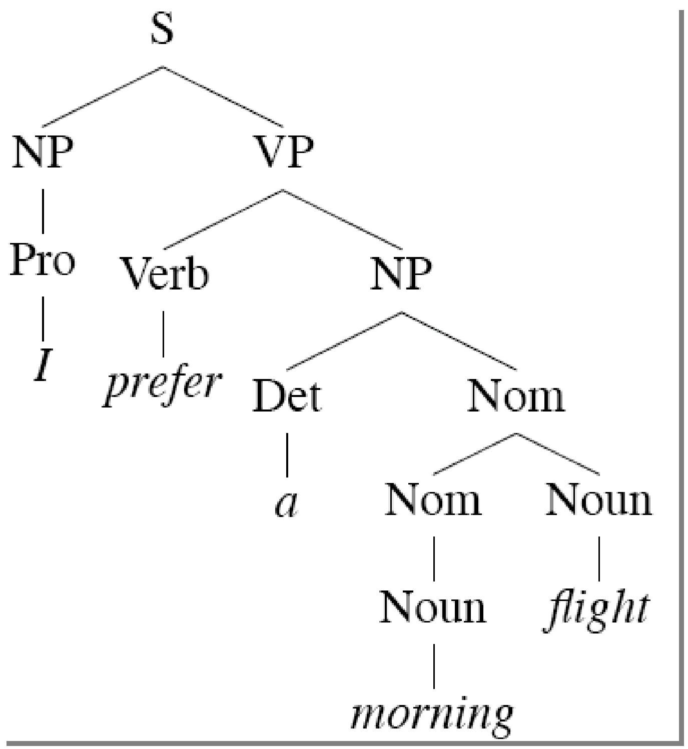
- G = (T, N, S, R)
  - T: **words or tokens**
  - N: **POS tags, syntactic tags**
  - S: the start symbol
  - R: rules of the form X →γ
    - X: a non-terminal
    - γ: the sequence of X's children

$$NP \rightarrow Det\ Nominal$$
$$NP \rightarrow ProperNoun$$
$$Nominal \rightarrow Noun \mid Nominal\ Noun$$



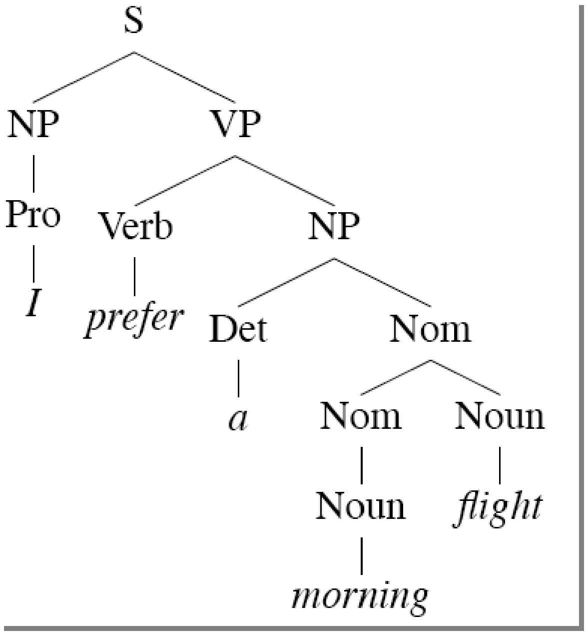
Exercise: Context free grammar

- Give a context free grammar to generate the parse tree



Exercise: Context free grammar

- Give a context free grammar to generate the parse tree
  - T: {I, prefer, a, morning, flight}
  - N: {VP, VP, Pro, Verb, Det, Nom, Noun}
  - S: S
  - R:
    - S -> NP VP
    - NP -> Pro
    - VP -> Verb NP
    - Verb -> prefer
    - NP -> Det Nom
    - Nom -> Nom Noun | Noun
    - Noun -> morning | flight
    - Det -> a
    - Pro -> I



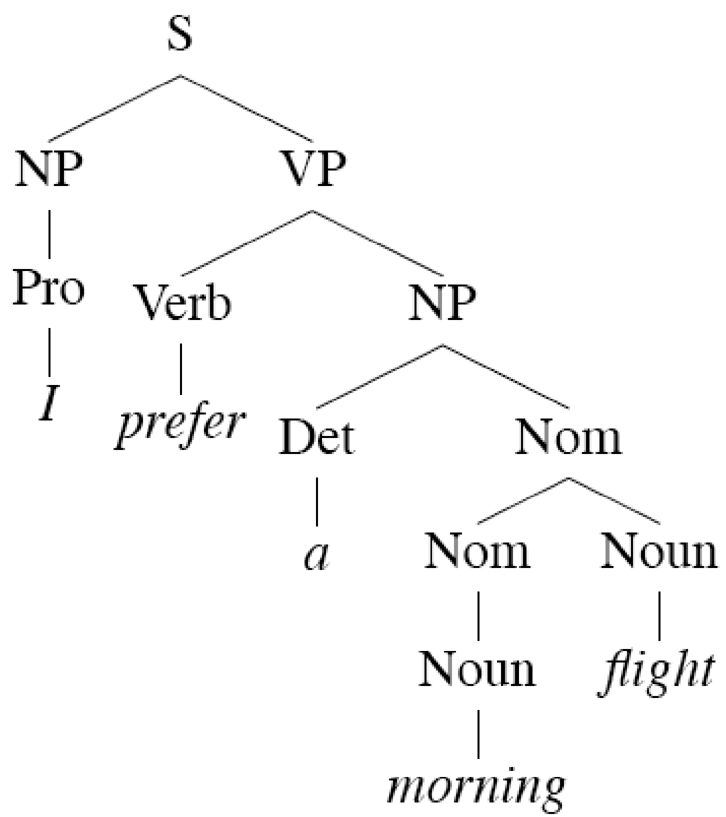
L0 grammar

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

Derivation

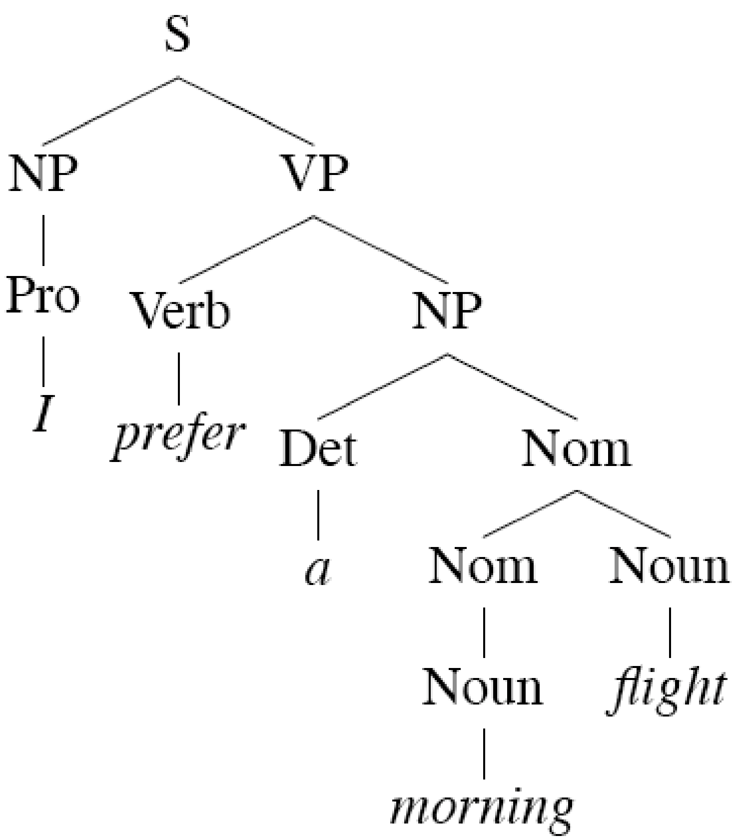
- A sequence of rules applied to a string that accounts for that string

$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
\\
NP & \rightarrow & Pronoun \\
 & | & Proper\text{-}Noun \\
 & | & Det\ Nominal \\
Nominal & \rightarrow & Nominal\ Noun \\
 & | & Noun \\
\\
VP & \rightarrow & Verb \\
 & | & Verb\ NP \\
 & | & Verb\ NP\ PP \\
 & | & Verb\ PP \\
\\
PP & \rightarrow & Preposition\ NP
\end{array}
$$



I prefer a morning flight.

Derivation (2)



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

Summary: CFG & phrase structure

- G = (T, N, S, R)
  - Terminals: words
  - Non-terminals: constituent names (e.g. NP, VP)
  - Start symbol: S (sentence) or NP (noun phrase)
  - Rules: e.g. subcategorization frames for verbs

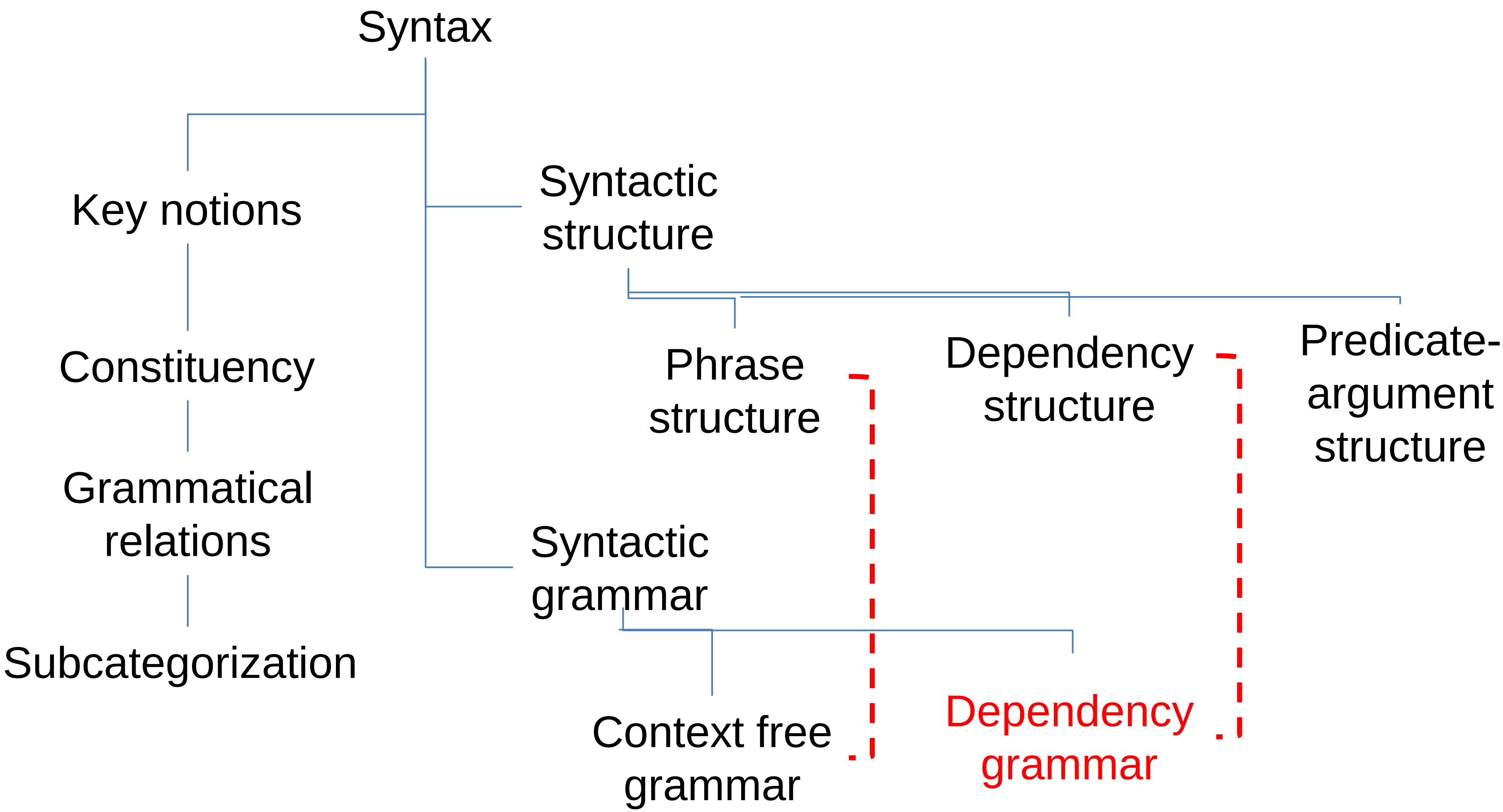
- Derivation will generate the phrase structure of an input

Example grammar formalisms

- Context free grammar (CFG)
- Combinatory categorial grammar (CCG)
- Dependency grammar (DG)

- HPSG: head-driven phrase structure grammar
- LFG: lexical functional grammar
- TAG: tree-adjoining grammar

Summary


Syntax
Key notions
Constituency
Grammatical relations
Subcategorization
Syntactic structure
Phrase structure
Dependency structure
Predicate-argument structure
Syntactic grammar
Context free grammar
Dependency grammar

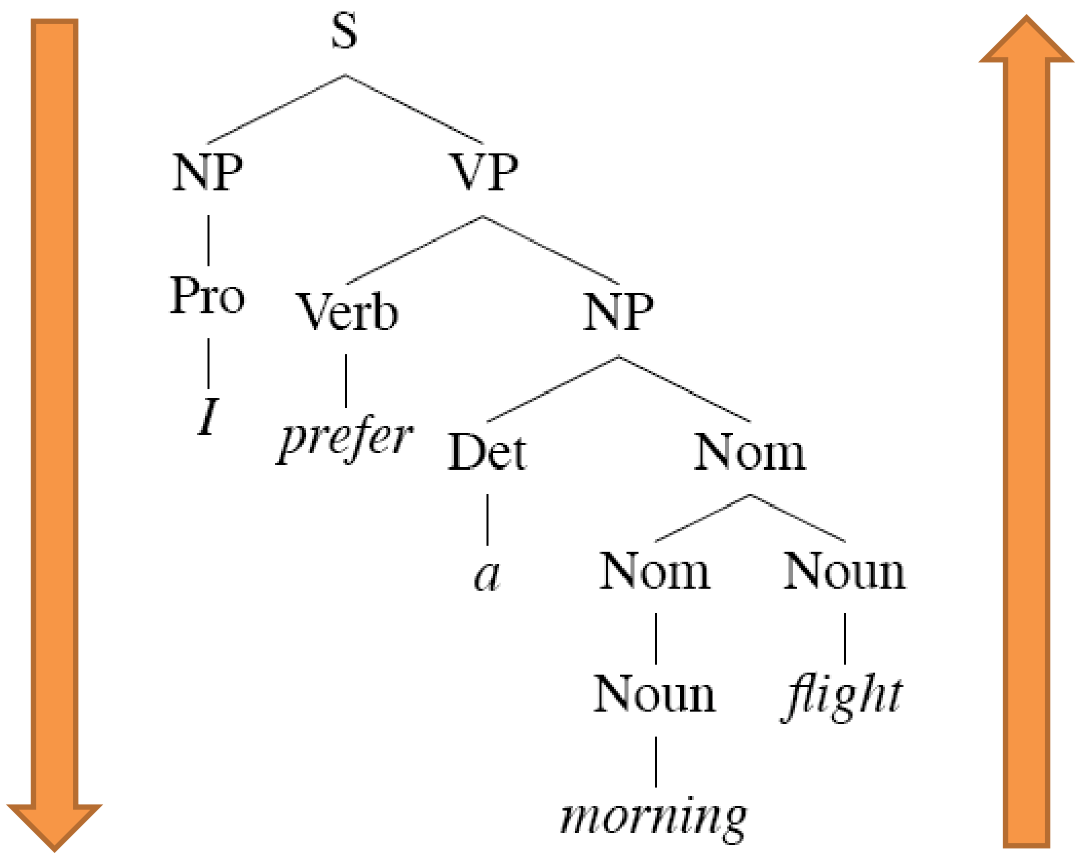
We focus on CFG!

- Classic approach
  - Still widely used for practical parsing systems

- Theoretically well-studied
  - Equivalent to Backus-Naur Form (BNF)
  - SQL is formally defined in BNF

Parsing strategies

Top-down parsing



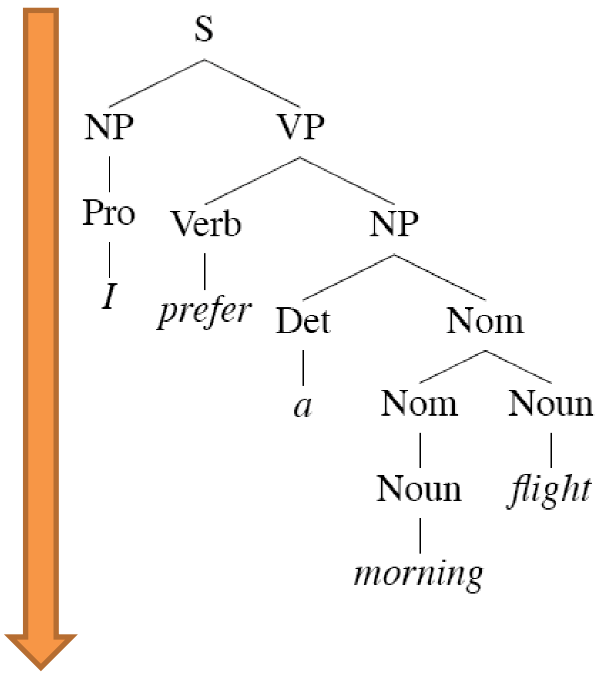
Bottom-up parsing

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

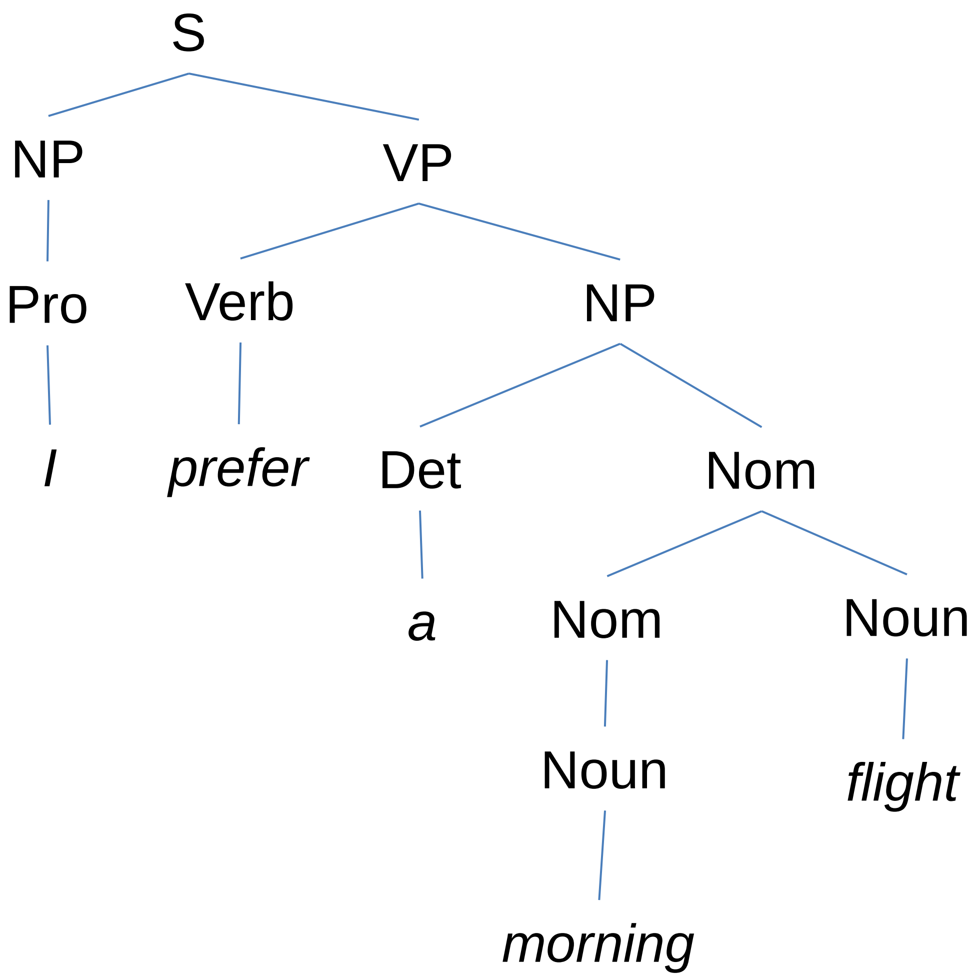
- Parsing is finished when the rewriting generates the whole string



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*

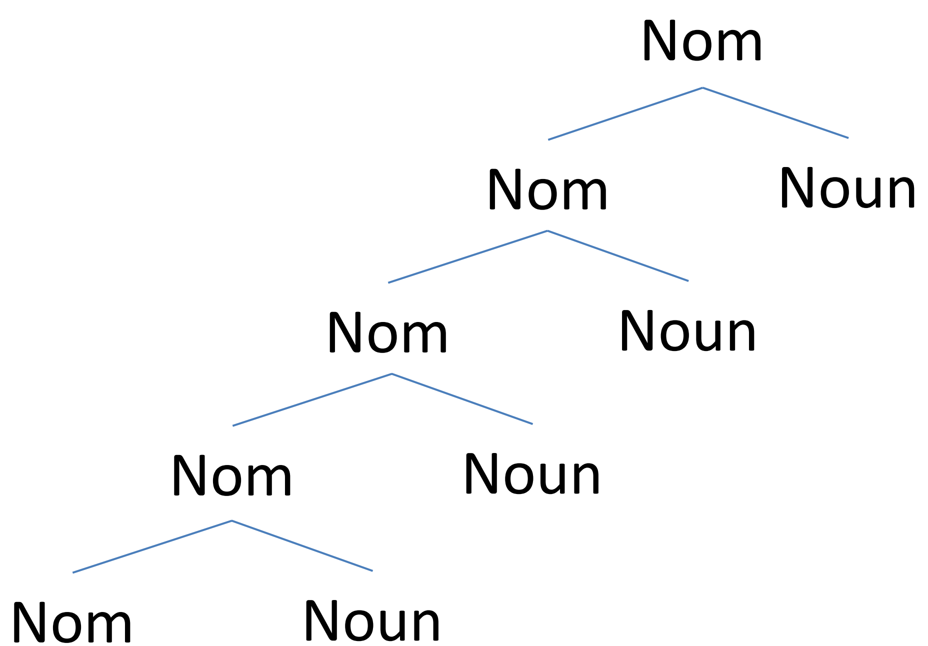
5. Nom → Nom Noun
   Noun → *flight*

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

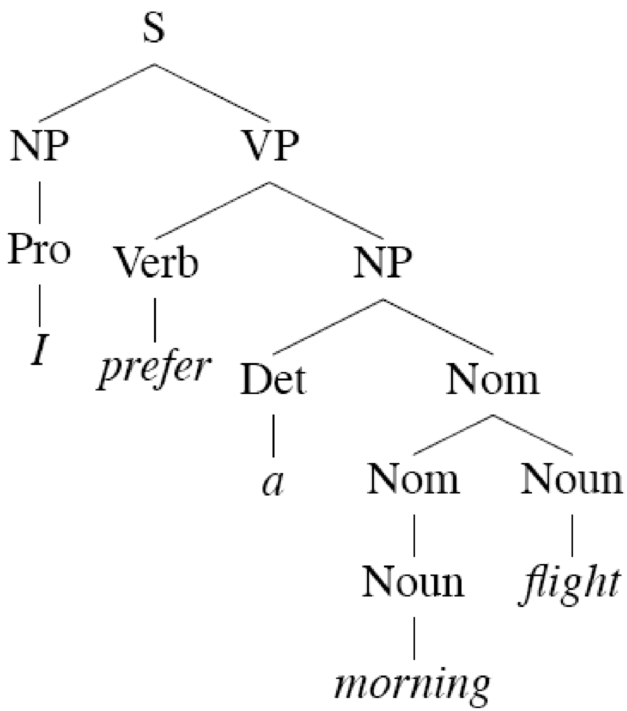


Example: Southeast Asia Public Interest Research Group

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

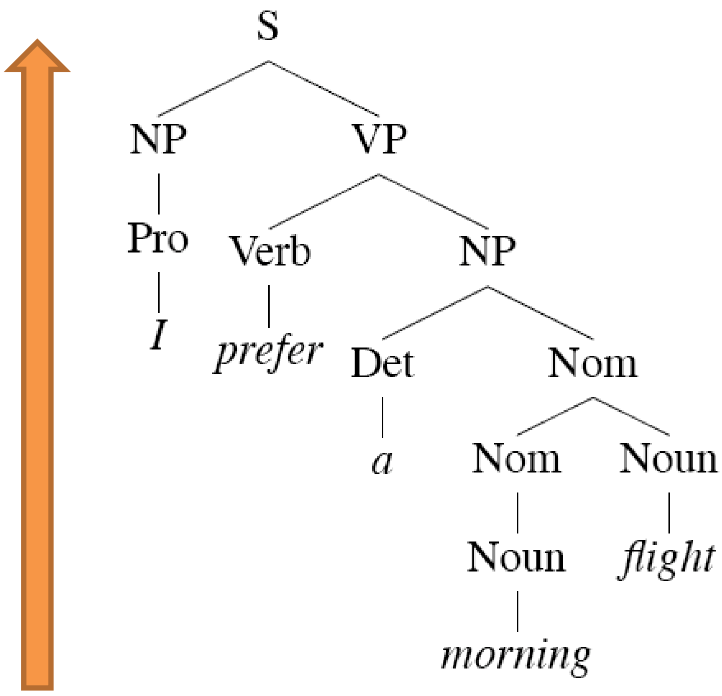
- In practice, **part-of-speech tags are predetermined**



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*

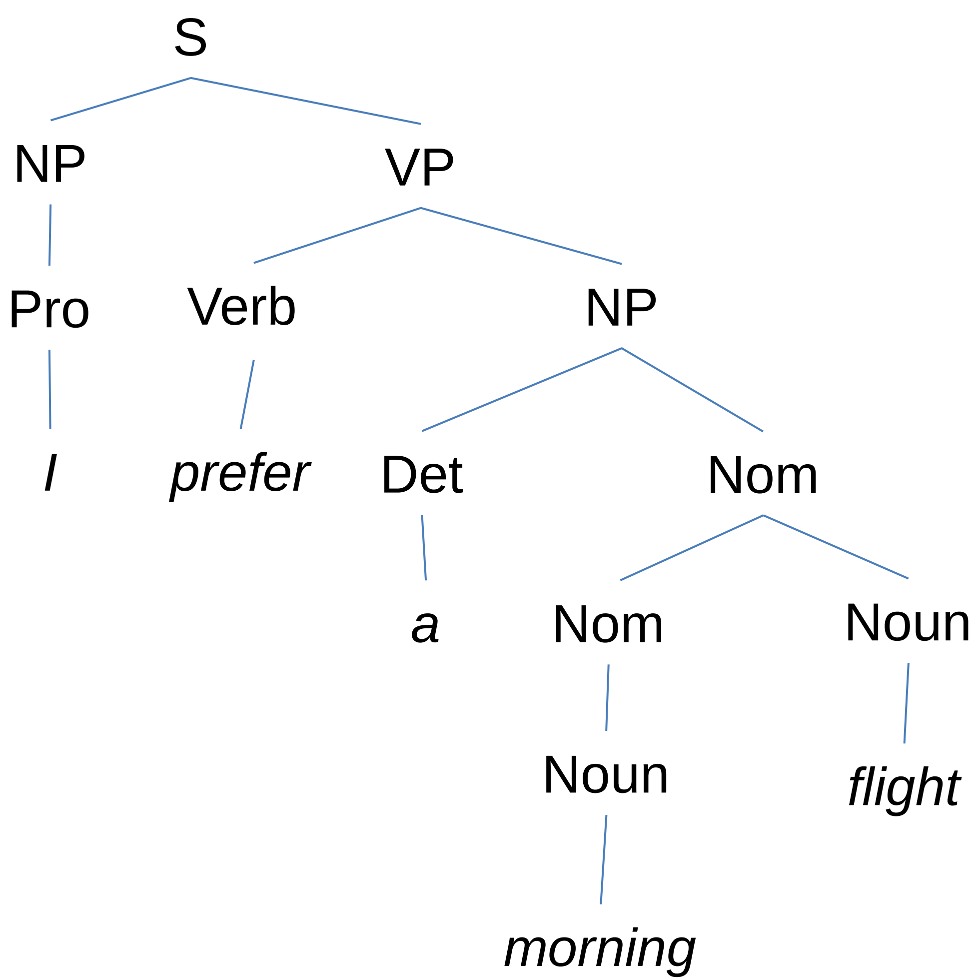
Bottom-up parsing

- 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

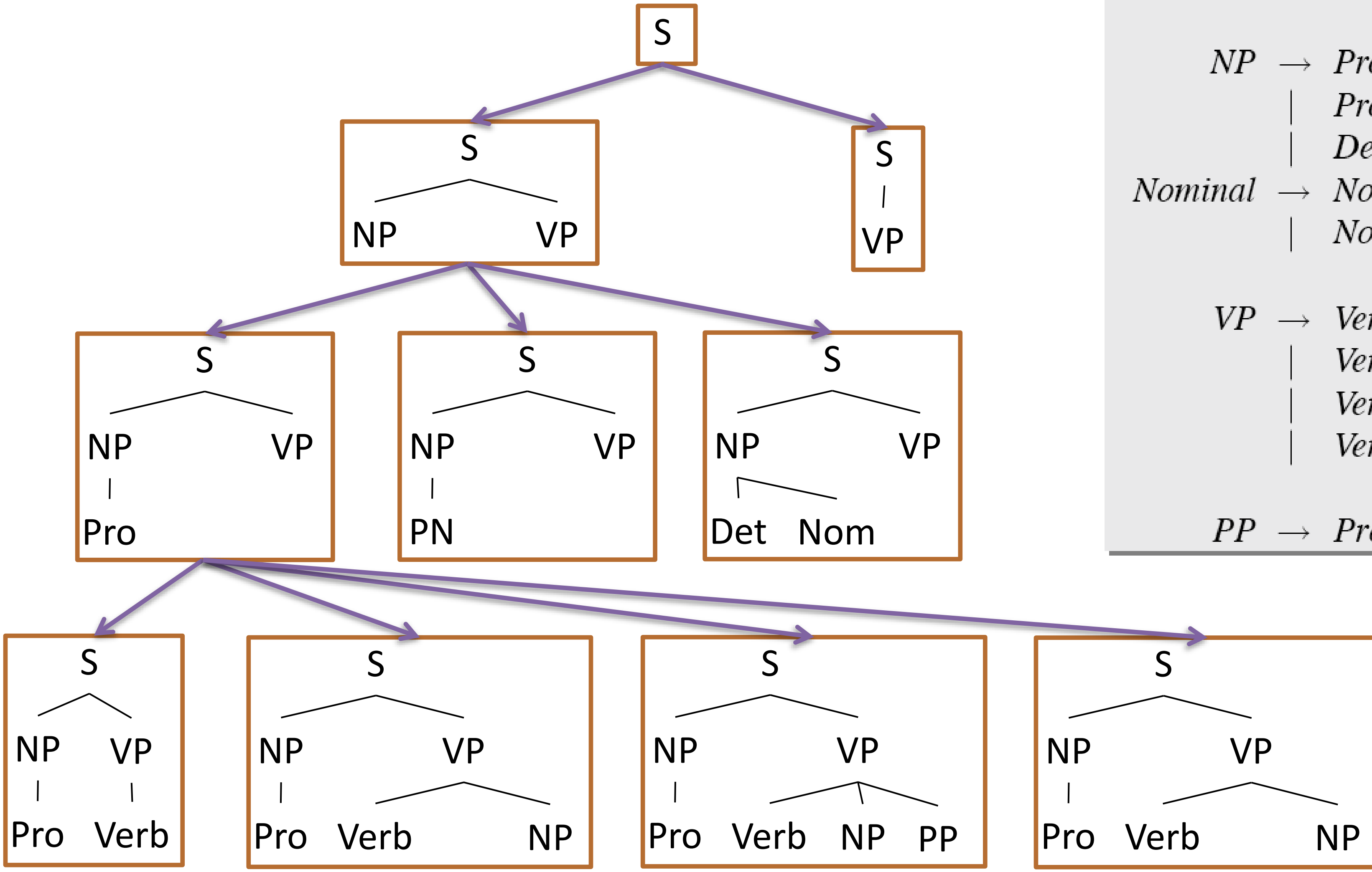
Bottom-up parsing

- 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

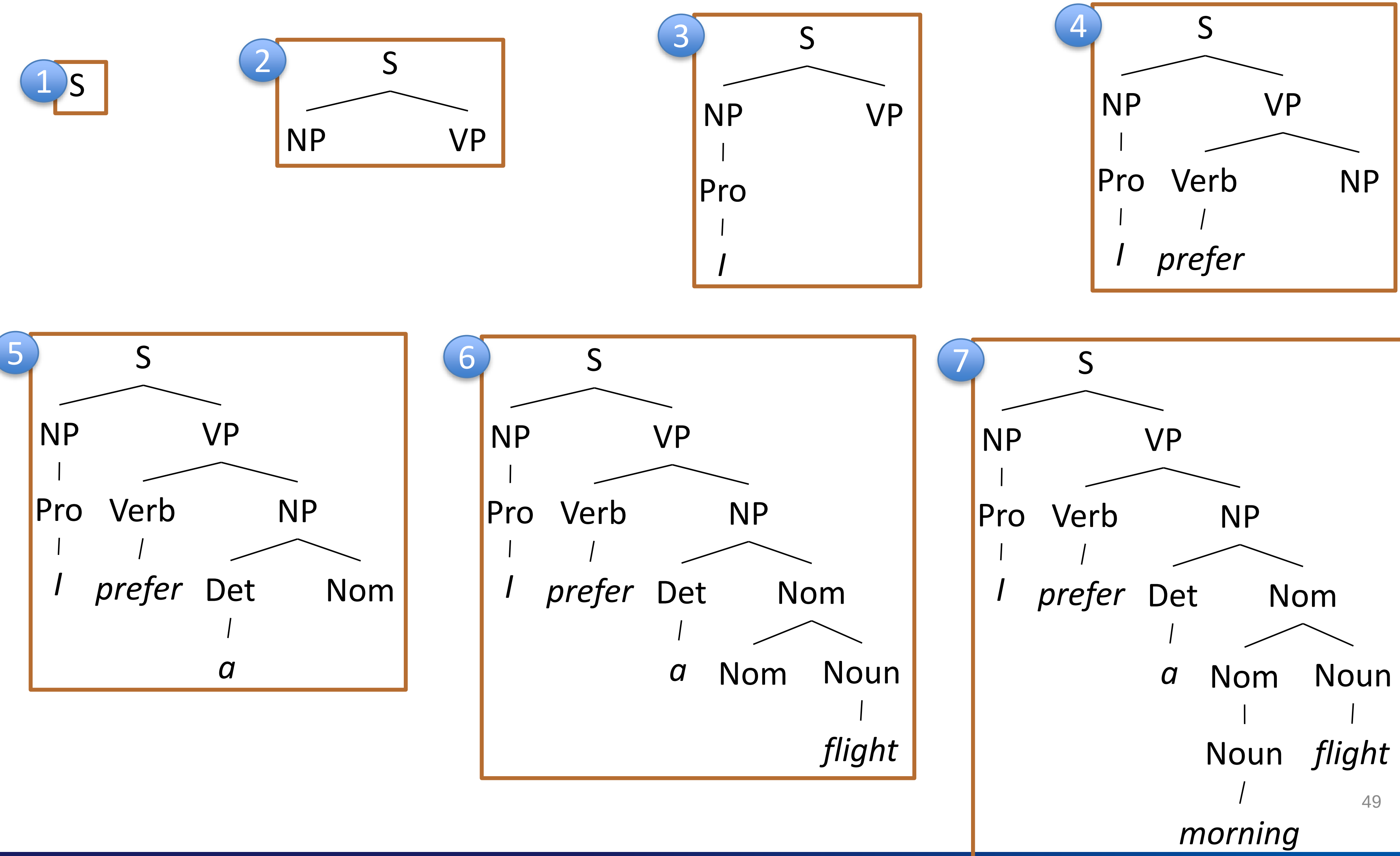
Parsing as search (e.g. I prefer a morning flight)



Grammar Rules

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

Parsing as search



Exercise: Parsing as search

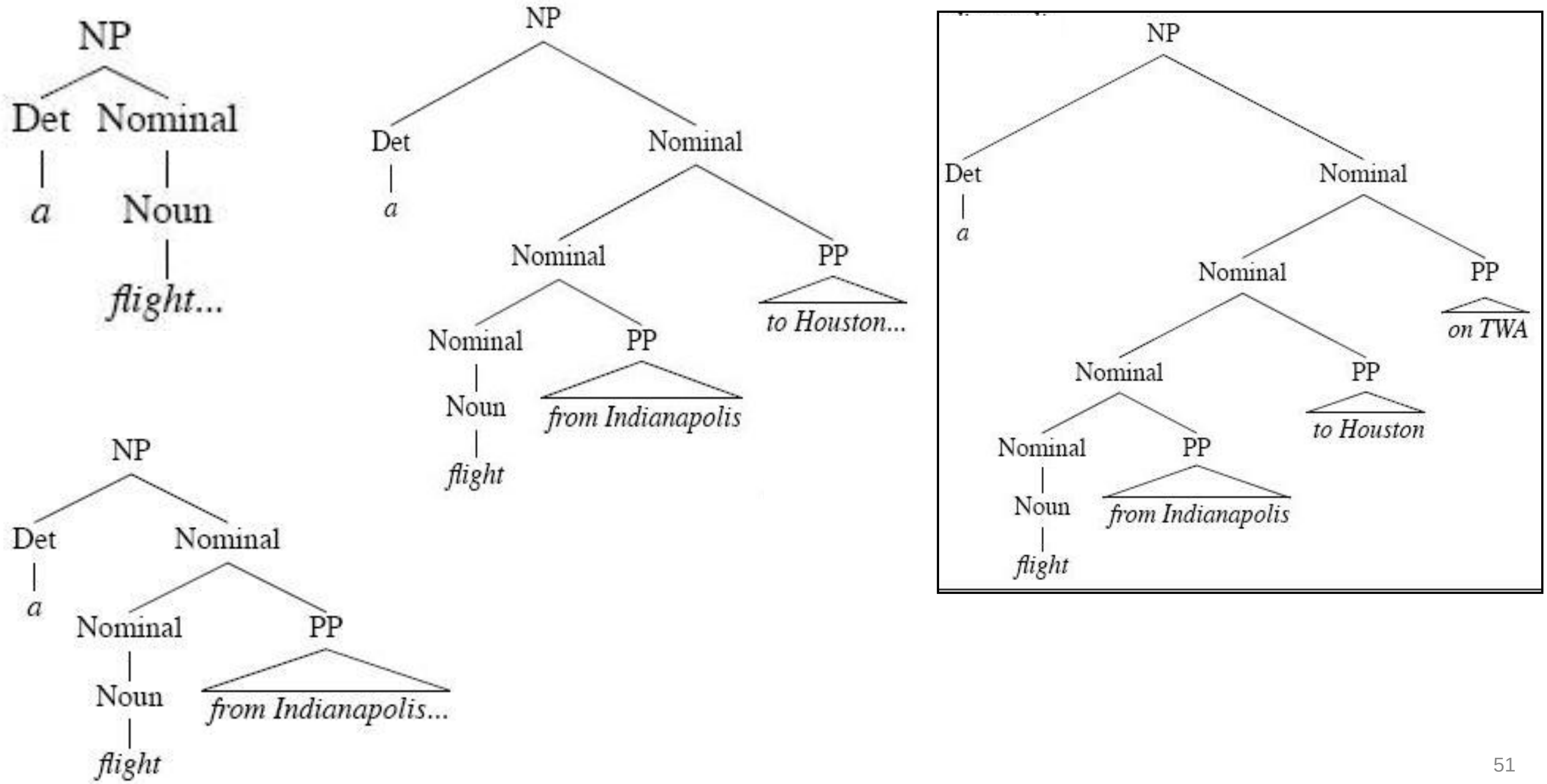
- Draw a search space for parsing "I sleep" in a top-down parsing based on L0 grammar

Grammar Rules

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

Repeated work in parsing-as-search

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

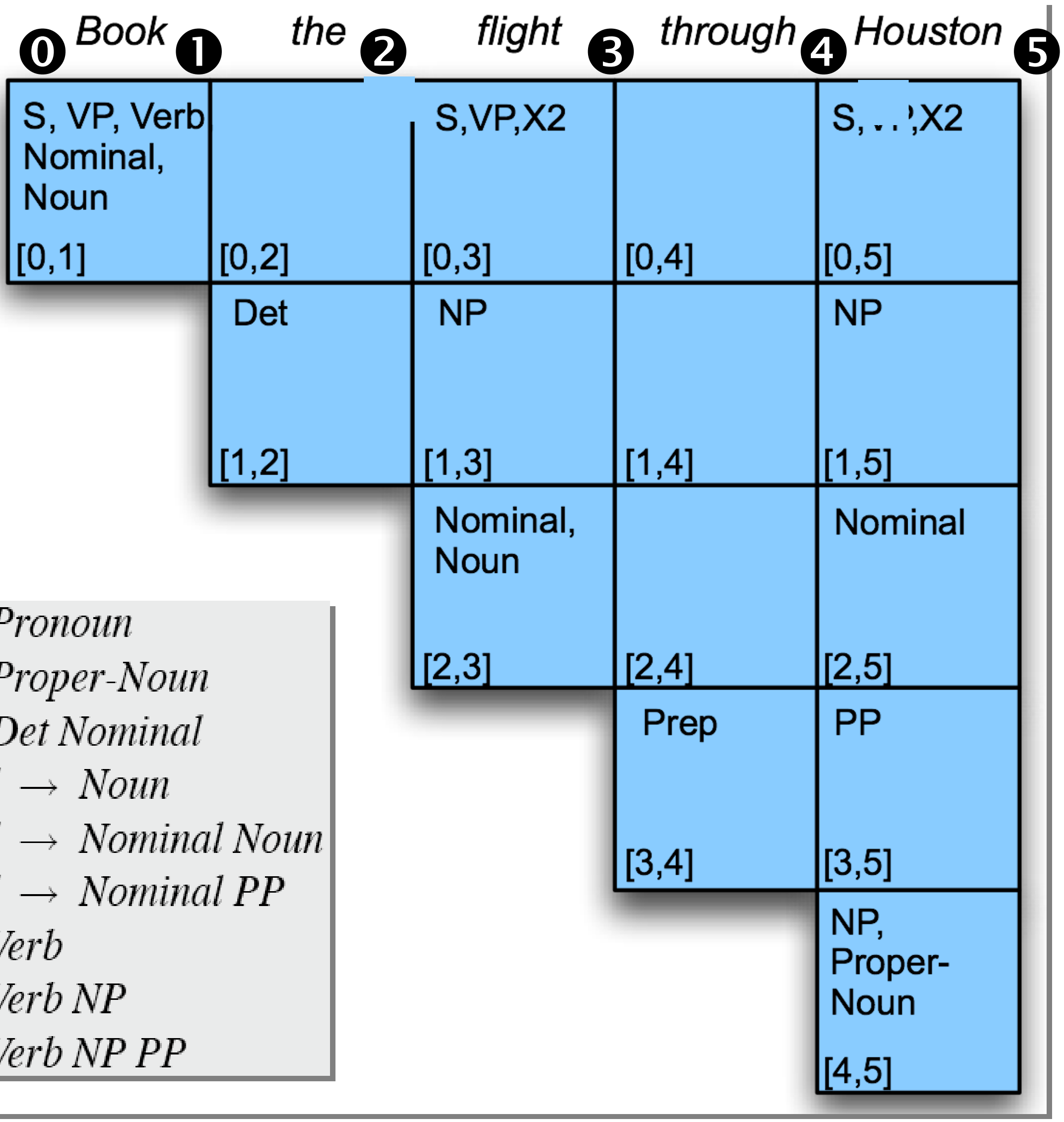


Dynamic programming parsing methods

- 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

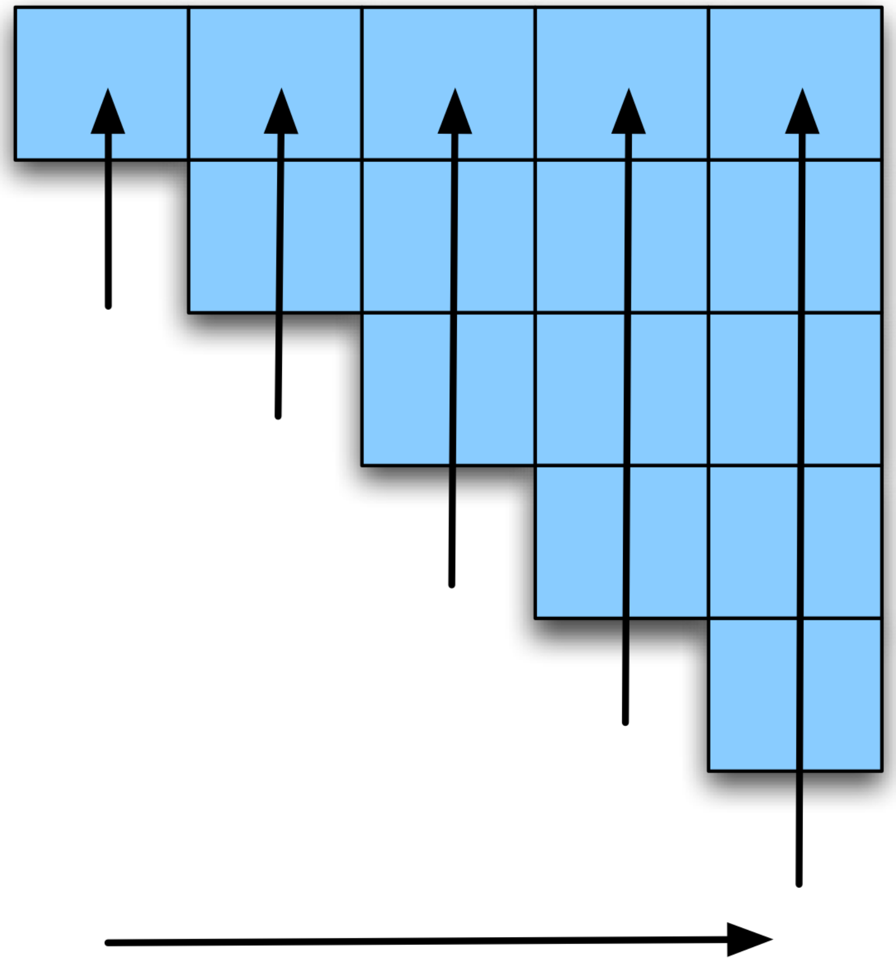


$\mathscr{L}_1$ **Grammar**		
S	→	*NP VP*
S	→	*Aux NP VP*
S	→	*VP*
VP	→	*Verb PP*
VP	→	*VP PP*
PP	→	*Preposition NP*

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*

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	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]

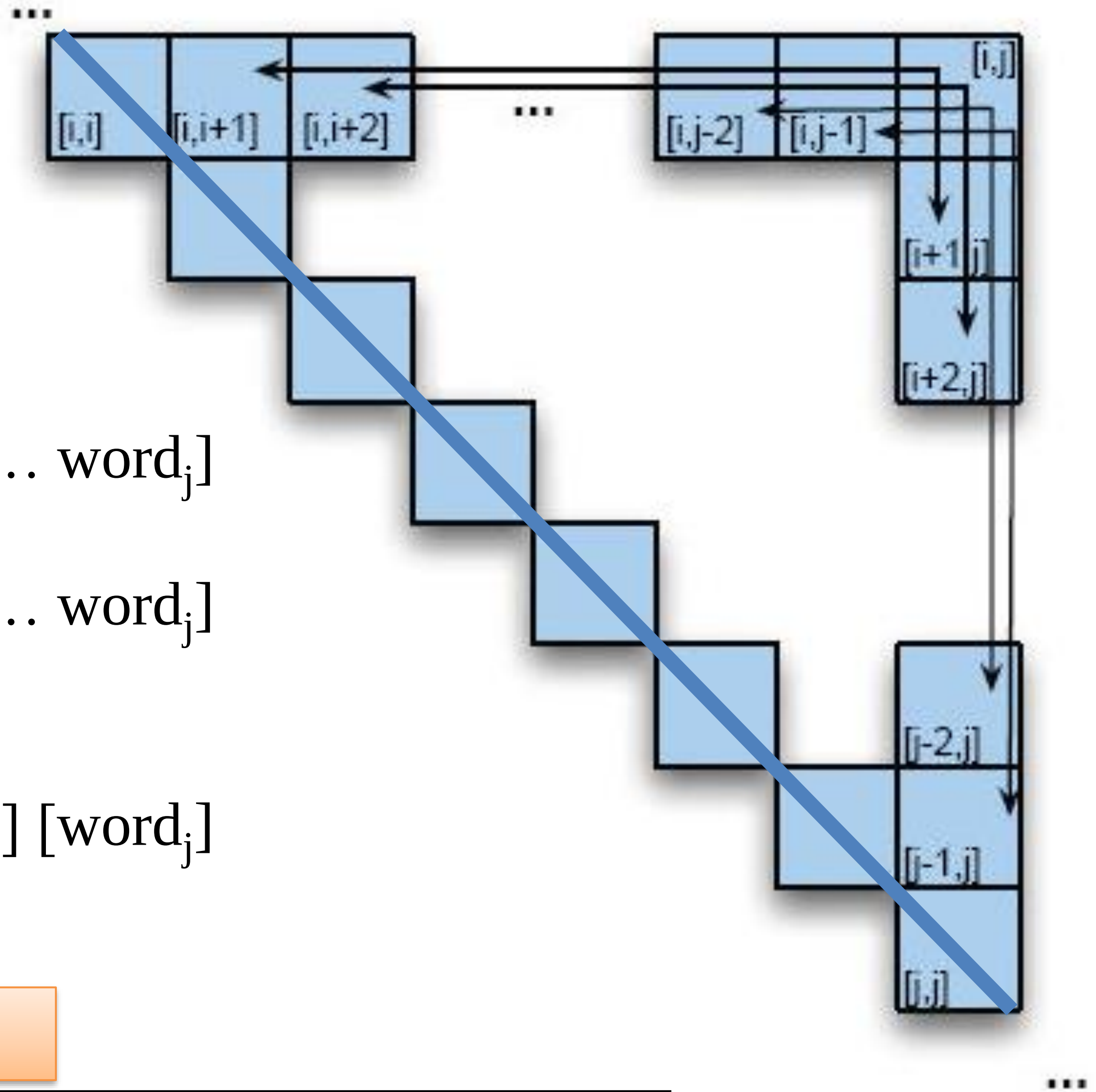
Assume an RHS has
exactly two items (X → Y Z)

[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:

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) POS Tagging
  - RHS must have two non-terminals or one terminal

- Conversion to CNF
  - Example on next slide

CNF grammar: example

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

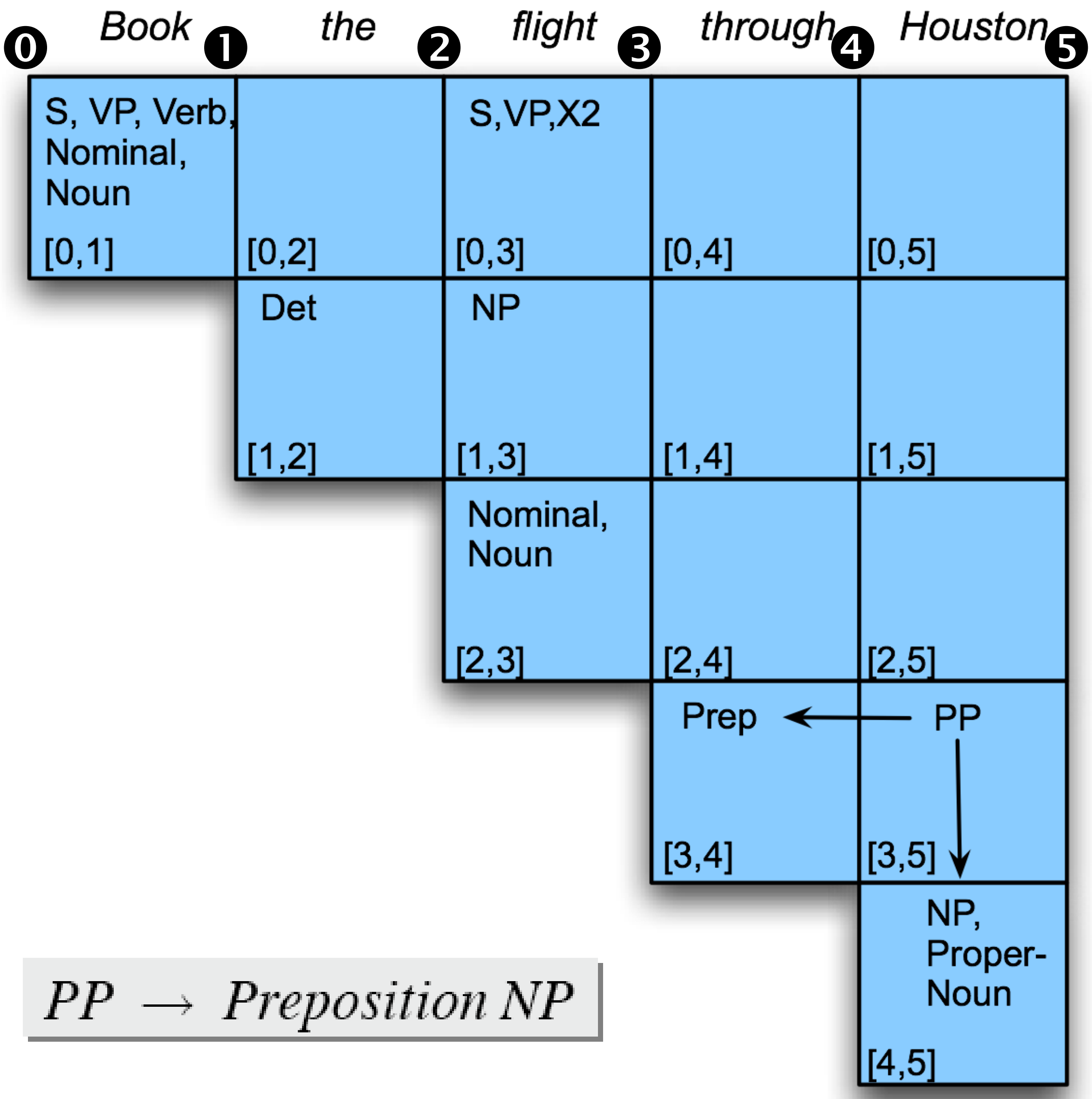
Example : filling the 5th column

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

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	[2,5]
			Prep [3,4]	[3.5]
				NP, Proper-Noun [4,5]

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

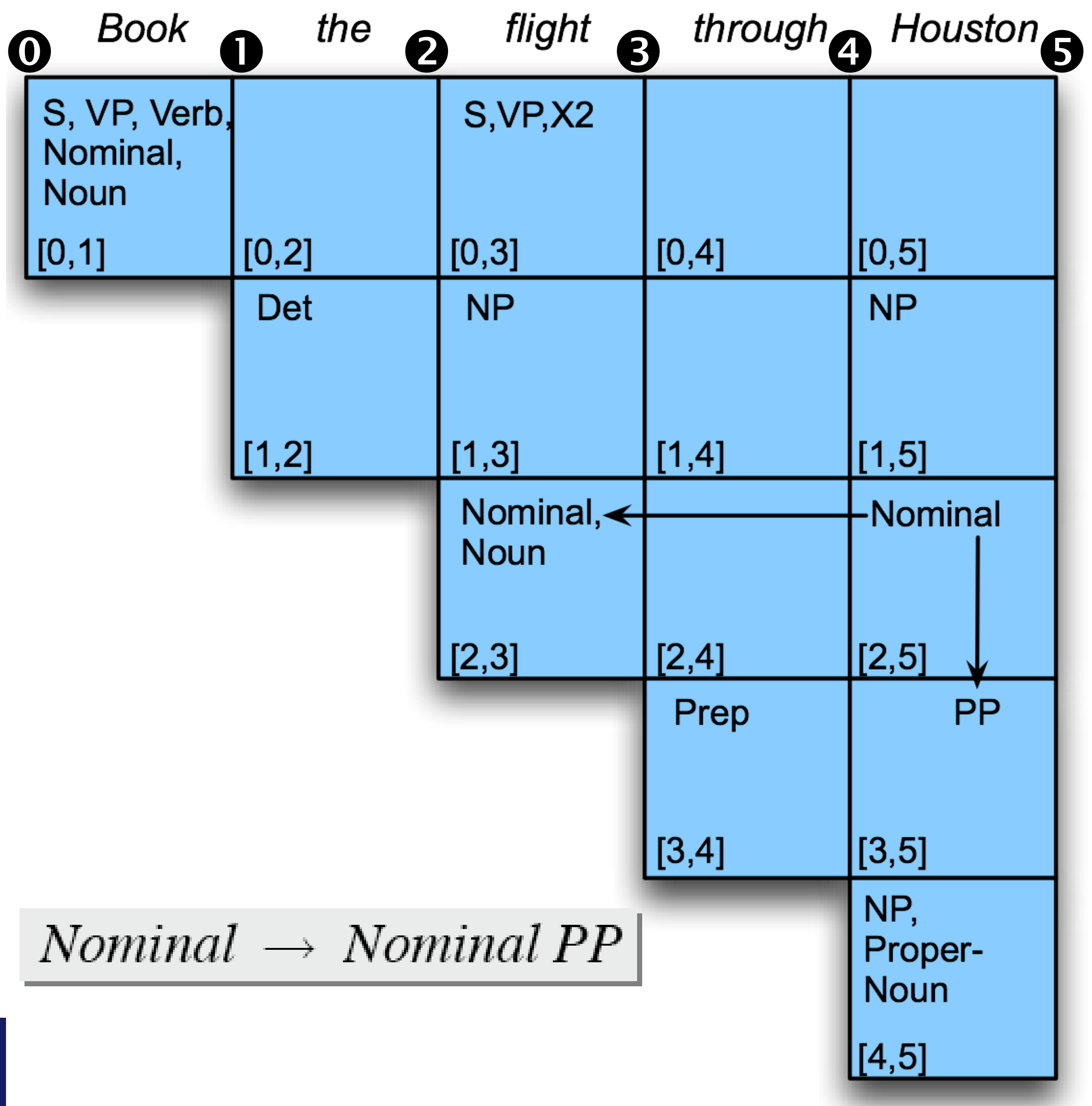
Example : filling the 5th column



$PP \rightarrow Preposition\ NP$

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

Example : filling the 5th column



$$Nominal \rightarrow Nominal\ PP$$

$S \rightarrow NP\ VP$

$S \rightarrow X1\ VP$

$X1 \rightarrow Aux\ NP$

$S \rightarrow book \mid include \mid prefer$

$S \rightarrow Verb\ NP$

$S \rightarrow X2\ PP$

$S \rightarrow Verb\ PP$

$S \rightarrow VP\ PP$

$NP \rightarrow I \mid she \mid me$

$NP \rightarrow TWA \mid Houston$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow book \mid flight \mid meal \mid$

$Nominal \rightarrow Nominal\ Noun$

$Nominal \rightarrow Nominal\ PP$

$VP \rightarrow book \mid include \mid prefer$

$VP \rightarrow Verb\ NP$

$VP \rightarrow X2\ PP$

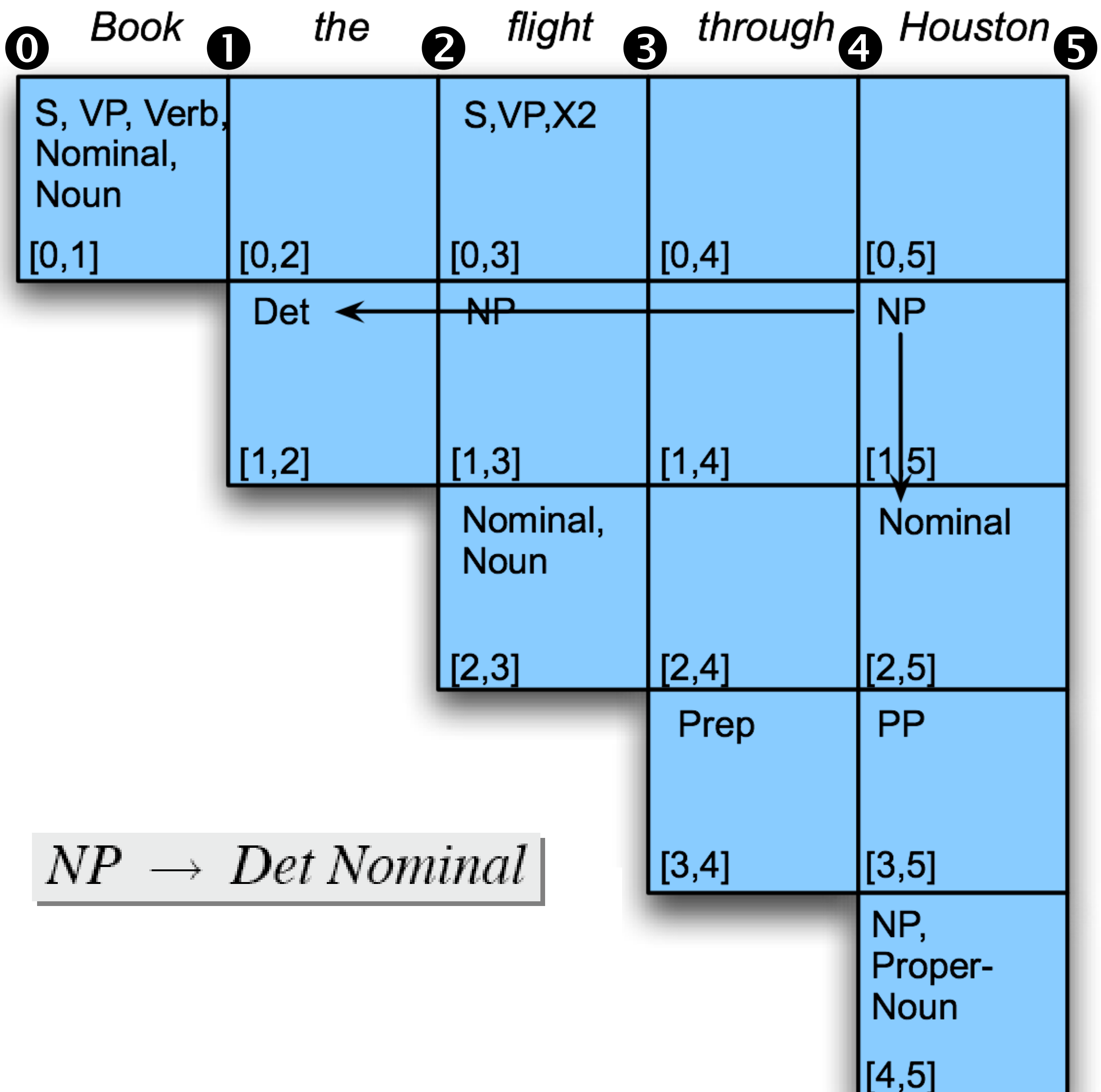
$X2 \rightarrow Verb\ NP$

$VP \rightarrow Verb\ PP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Preposition\ NP$

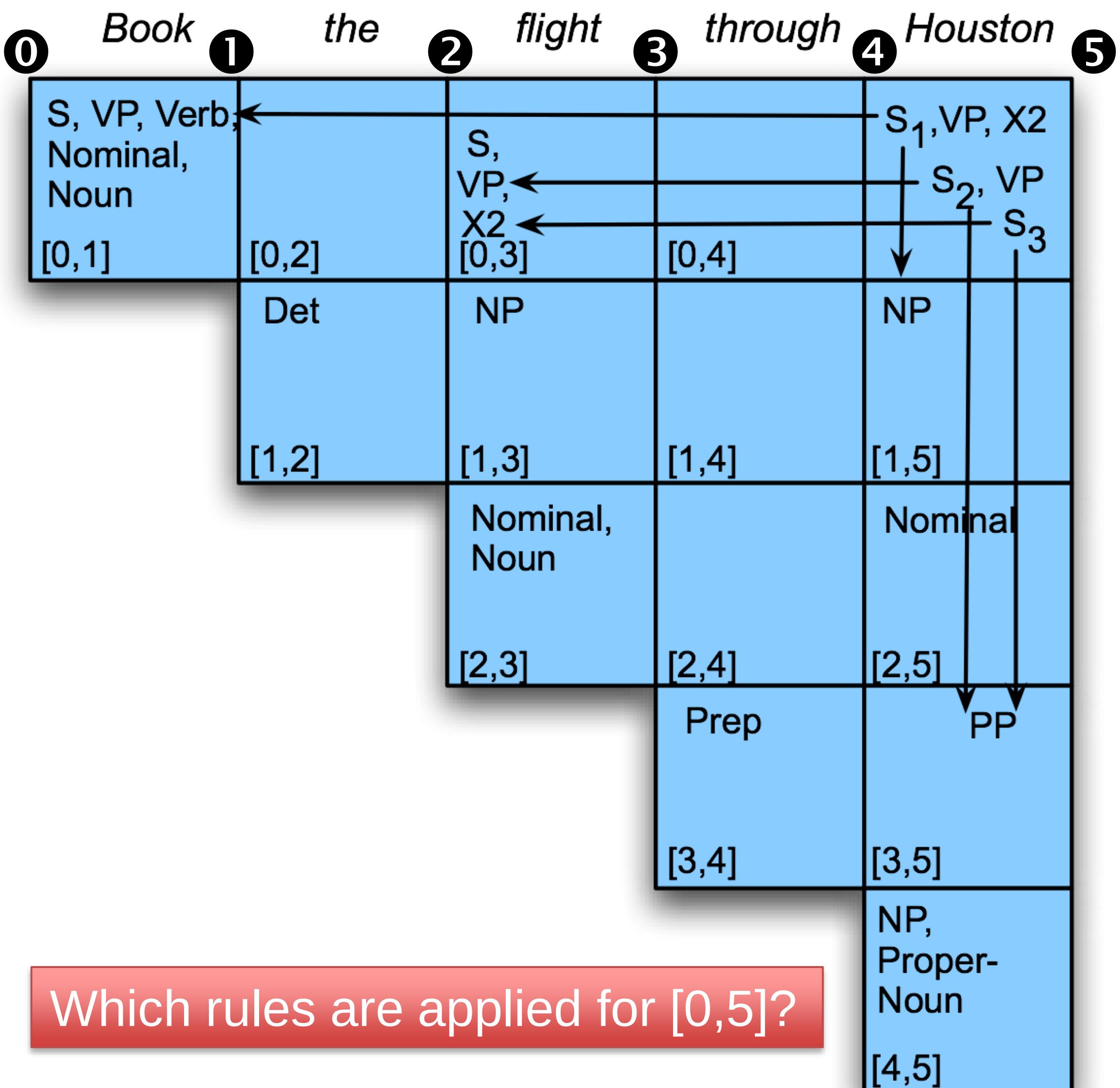
Example : filling the 5th column



$NP \rightarrow Det\ Nominal$

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

Exercise: CKY parsing



$S \rightarrow NP\ VP$

$S \rightarrow X1\ VP$

$X1 \rightarrow Aux\ NP$

$S \rightarrow book \mid include \mid prefer$

$S \rightarrow Verb\ NP$

$S \rightarrow X2\ PP$

$S \rightarrow Verb\ PP$

$S \rightarrow VP\ PP$

$NP \rightarrow I \mid she \mid me$

$NP \rightarrow TWA \mid Houston$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow book \mid flight \mid meal \mid$

$Nominal \rightarrow Nominal\ Noun$

$Nominal \rightarrow Nominal\ PP$

$VP \rightarrow book \mid include \mid prefer$

$VP \rightarrow Verb\ NP$

$VP \rightarrow X2\ PP$

$X2 \rightarrow Verb\ NP$

$VP \rightarrow Verb\ PP$

$VP \rightarrow VP\ PP$

$PP \rightarrow Preposition\ NP$

Which rules are applied for [0,5]?

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**
      $table[i,j] \leftarrow table[i,j] \cup \{A \mid A \rightarrow BC \in grammar, B \in table[i,k], C \in table[k,j]\}$

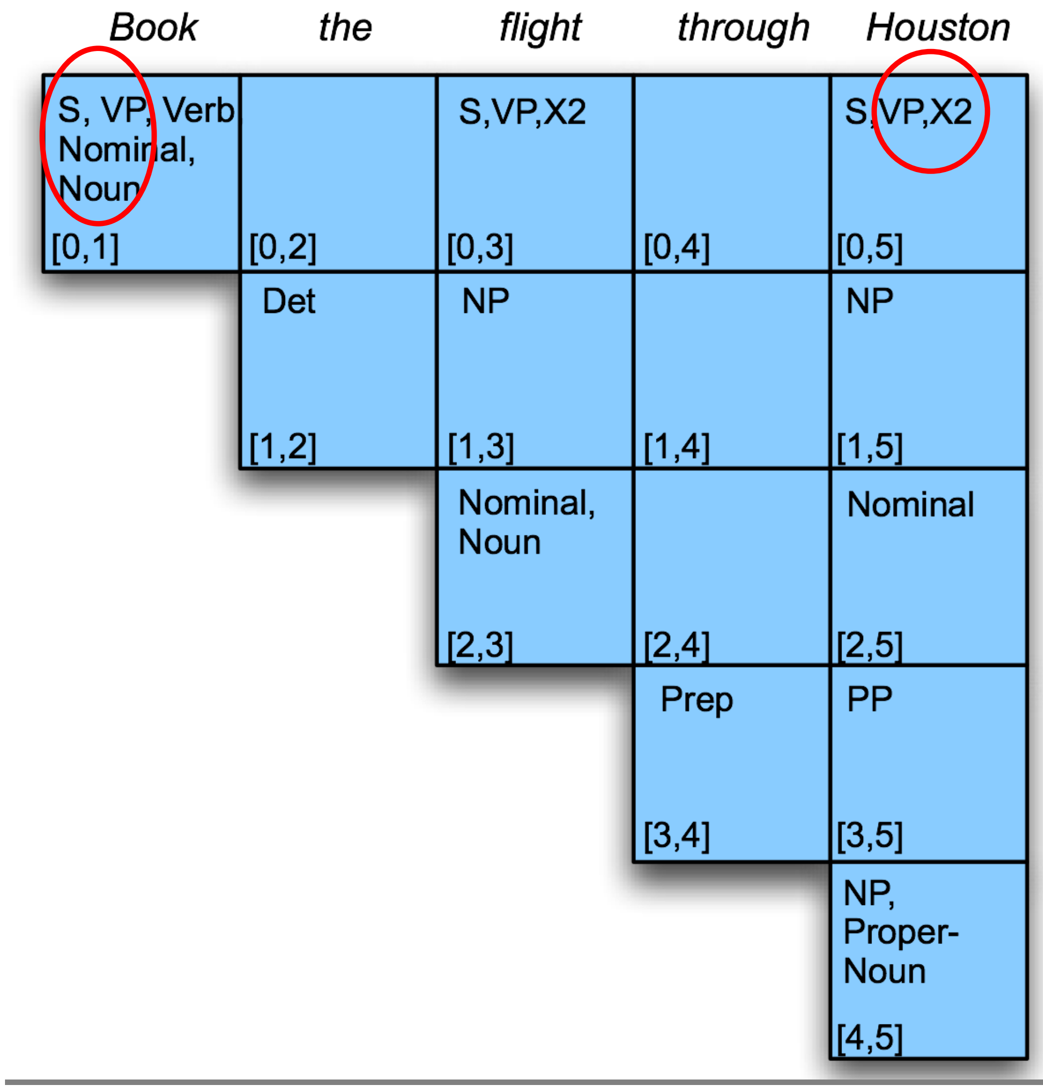
CKY parsing: summary

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

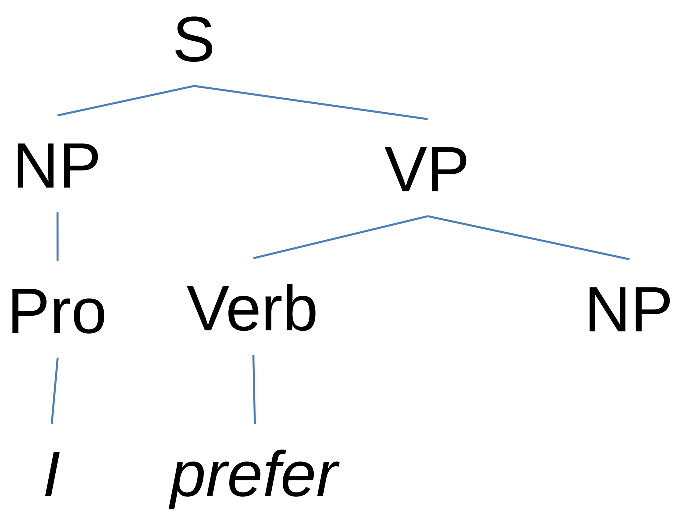
- Bottom-up approach
  - Process from [j-1,j] to [0,j]
  - This assures us that whenever we're filling a cell, the parts needed to fill it are already in the table (to the left and below)
  - It's somewhat natural in that it processes the input, from left to right, a word at a time

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



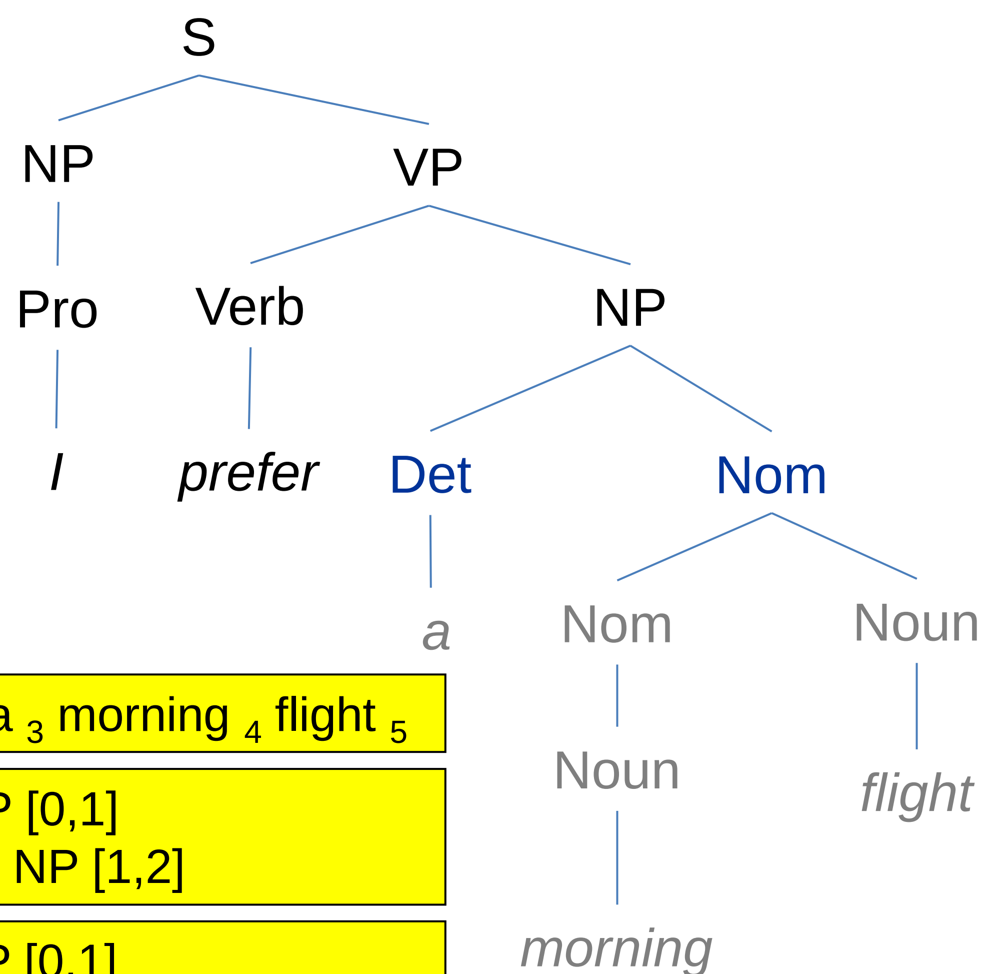
1. S → NP VP
2. NP → Pro
   Pro → *I*
3. VP → Verb NP
   Verb → *prefer*

$_0$ 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

$\mathscr{L}_1$ **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	$Verb \rightarrow book \bullet$	[0,1]	Scanner
S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
S17	$S \rightarrow VP \bullet$	[0,1]	Completer
S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

$\mathscr{L}_1$ **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 “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

$\mathscr{L}_1$ **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	$Nominal \rightarrow Nominal \bullet Noun$	[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

Top-down vs. Bottom-up parsing

Review

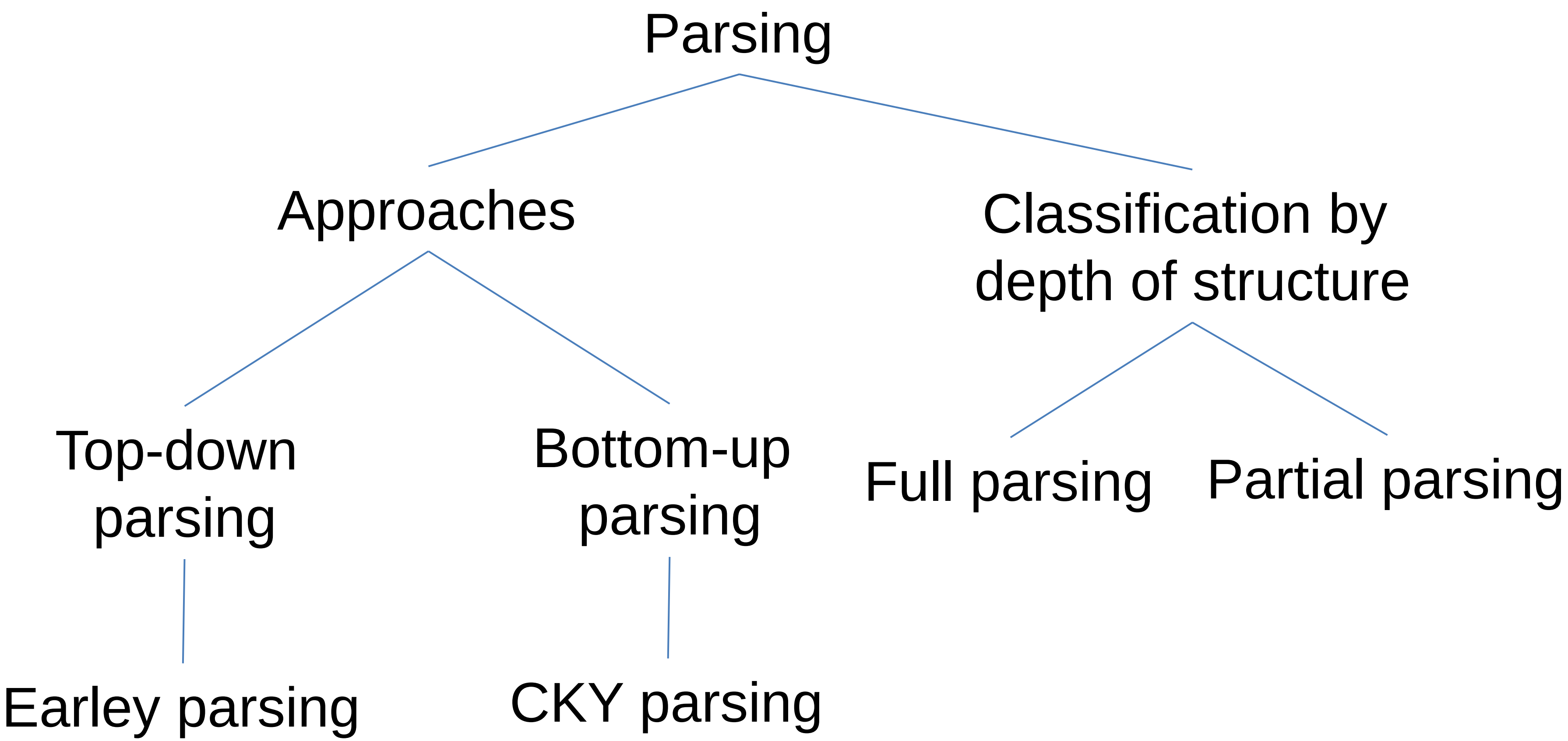
- Top-down
  - Waste lots of time in trying inconsistent rule applications. The application of the rules does not lead the generation of the given string
  - Never explore subtrees that cannot find a place in some S-rooted tree. All trees are generated starting with S.
- Bottom-up
  - Never suggest trees 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

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.

Summary


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

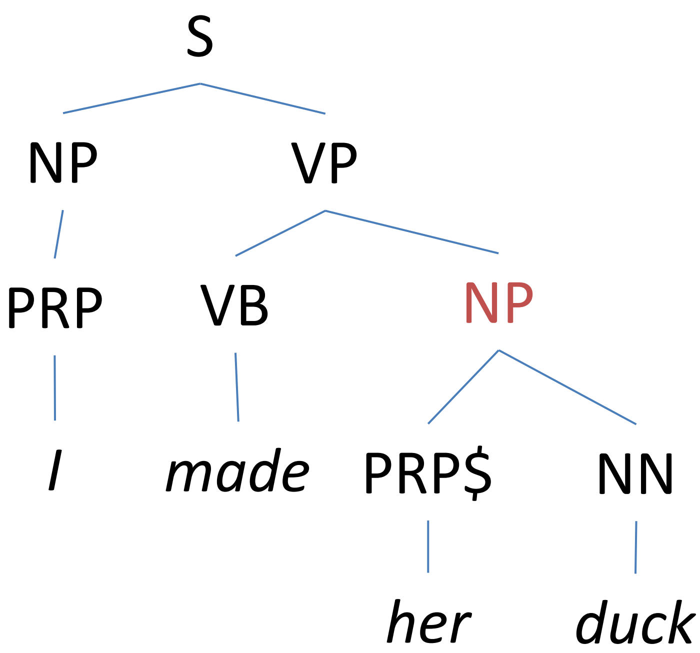
Ambiguity is Pervasive

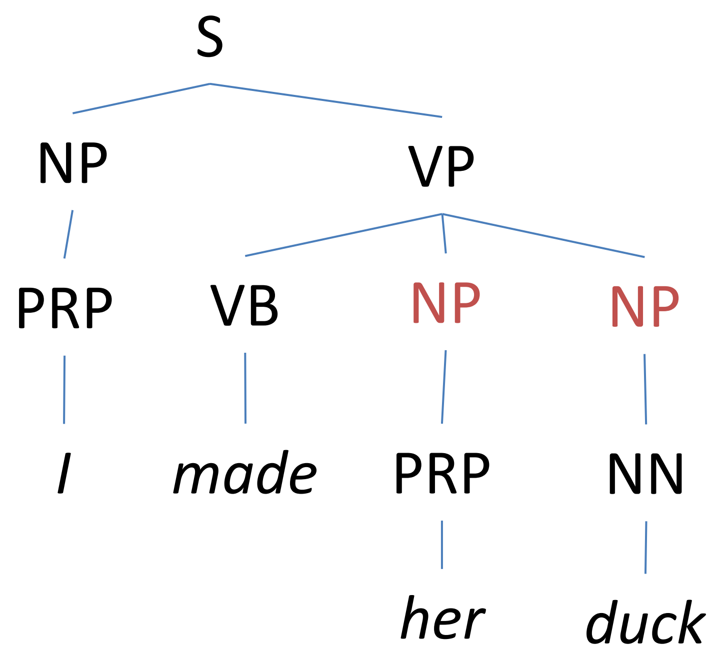
Review

- Find at least 5 meanings of this sentence: **I made her duck**

- Possible meanings
  - I cooked waterfowl for her
  - I cooked waterfowl belonging to her
  - I created the (plaster?) duck she owns
  - I caused her to quickly lower her head and body
  - I waved my magic wand and turned her into undifferentiated waterfowl

Ambiguity Resolution by Syntactic Structures





I cooked waterfowl belonging to her
I created the duck she owns

I cooked waterfowl for her benefit
I waved my magic wand and turned her into undifferentiated waterfowl

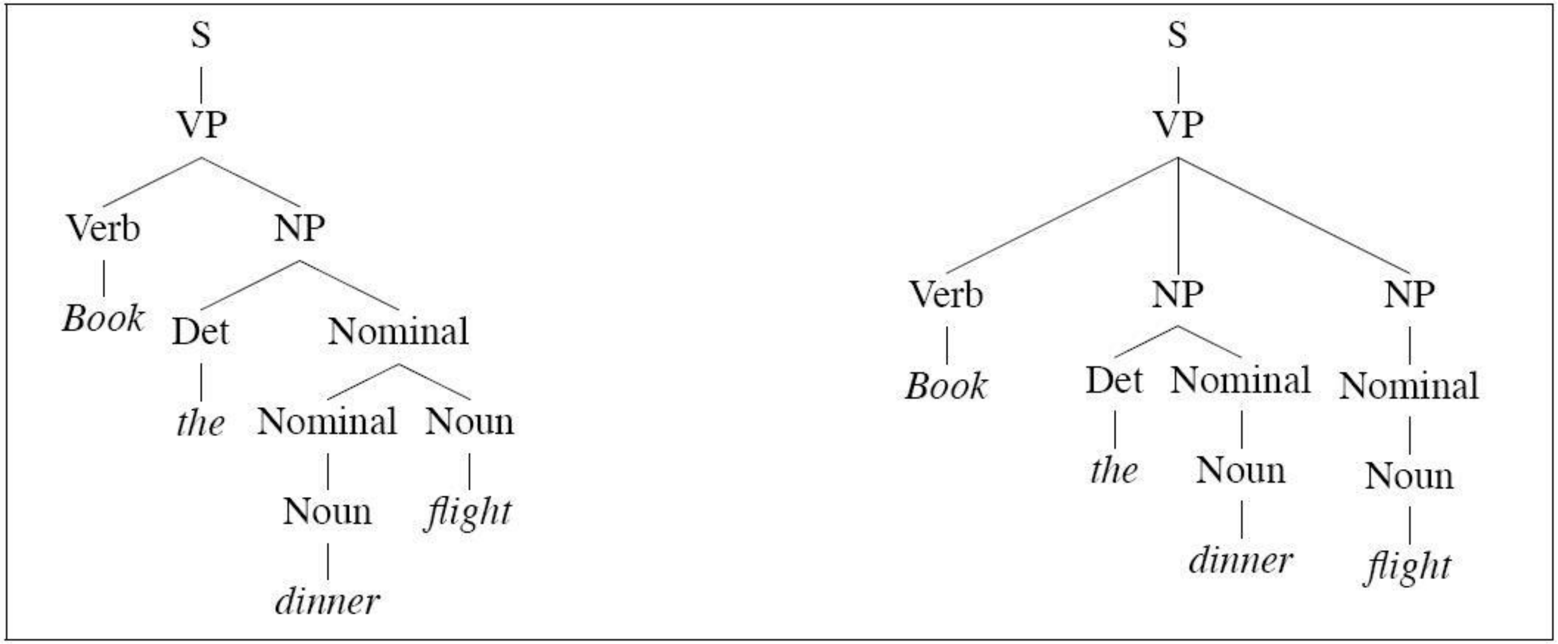
Exercise: Attachment Ambiguity

- How many distinct phrase structures may the following sentence have due to the prepositional phrase attachment ambiguities?

John wrote the book with a pen in the room

Need for Syntactic Disambiguation

- Consider the two parses of “Book the dinner flight”
- The left parse is sensible, while the right is not
  – How about “Can you book John a flight?”



Need for Common Sense

- People usually provide **only useful information** and take the rest for granted. The rest is common-sense: obvious things people know and usually leave unstated

Context-Free Grammar

Review

- 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

$$
\begin{array}{rcl}
NP & \rightarrow & Det\ Nominal \\
NP & \rightarrow & ProperNoun \\
Nominal & \rightarrow & Noun \mid Nominal\ Noun
\end{array}
$$

Probabilistic Context-Free Grammar (PCFG)

- G = (T, N, S, R, P)
  - T: a set of terminals (e.g. ‘boy’)
  - N: a set of non-terminals (e.g. Noun)
  - S: the start symbol, a non-terminal
  - R: rules of the form X →γ
  - P(R) gives the probability of each rule

Grammar	
$S \rightarrow NP\ VP$	[.80]
$S \rightarrow Aux\ NP\ VP$	[.15]
$S \rightarrow VP$	[.05]

$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

PCFG: Example

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] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.15] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flights$ [.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]	

PCFG

- How to learn the probability of rules?

$$\forall X \in N, \sum_{X \to \gamma \in R} P(X \to \gamma) = 1$$

- How to estimate the probability of parse trees with a PCFG?

- Once we have probabilities of possible parse trees, we can select the parse tree with the highest probability as the parse result for a given string

Probability of Rules

- Need for treebanks! $P(X \to \beta \mid X)$

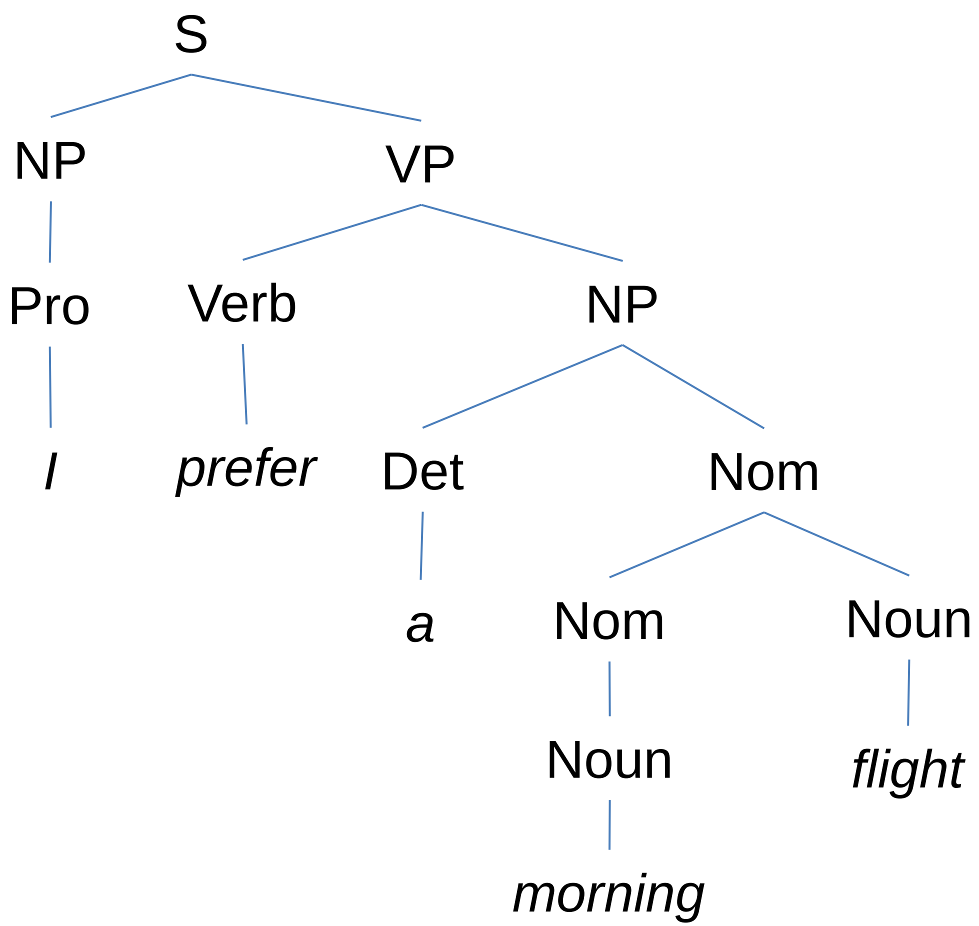
$$P(X \to \beta \mid X) = \frac{\text{count }(X \to \beta)}{\sum_{\gamma} \text{count }(X \to \gamma)} = \frac{\text{count }(X \to \beta)}{\text{count }(X)}$$

$$\forall X \in N, \sum_{X \to \gamma \in R} P(X \to \gamma) = 1$$

Derivation

Review

- A derivation (parse tree) consists of the bag of grammar rules that are in the tree



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 → morning
6. Nom → Noun
   Noun → flight

Probability of Parse Trees

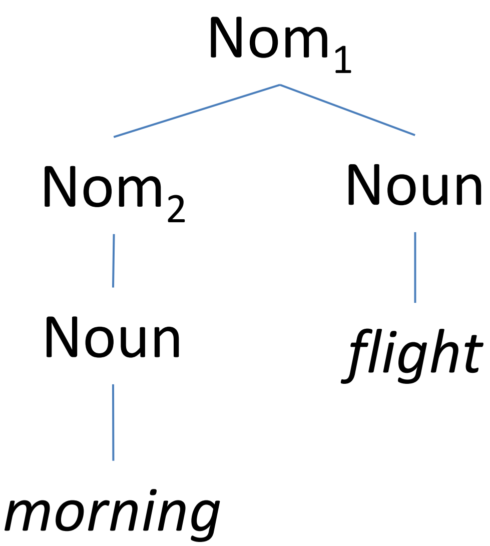
- A derivation (parse tree) consists of the bag of grammar rules that are in the tree
  - The probability of a tree is the product of the probabilities of the rules in the derivation.

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 → morning
6. Nom → Noun
   Noun → flight

$$P(T,S) = \prod_{node \in T} P(rule(n))$$

Probability of Parse Tree: Example

- Noun $\rightarrow$ morning [0.10]
- Noun $\rightarrow$ flight [0.40]
- Nom $\rightarrow$ Noun [0.75]
- Nom $\rightarrow$ Nom Noun [0.20]

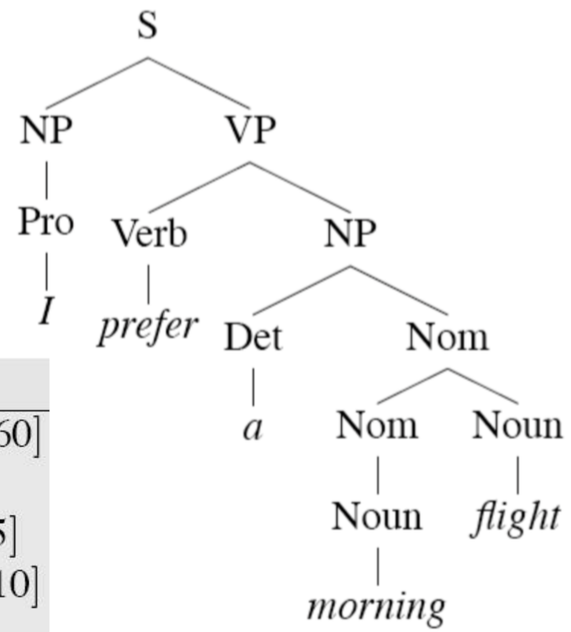


$$P(Nom_2) = 0.75 \times 0.1 = 0.75 \times 10^{-1}$$

$$P(Nom_1) = 0.2 \times 0.75 \times 10^{-1} \times 0.4 = 0.6 \times 10^{-2}$$

Exercise: Probability of Parse Tree

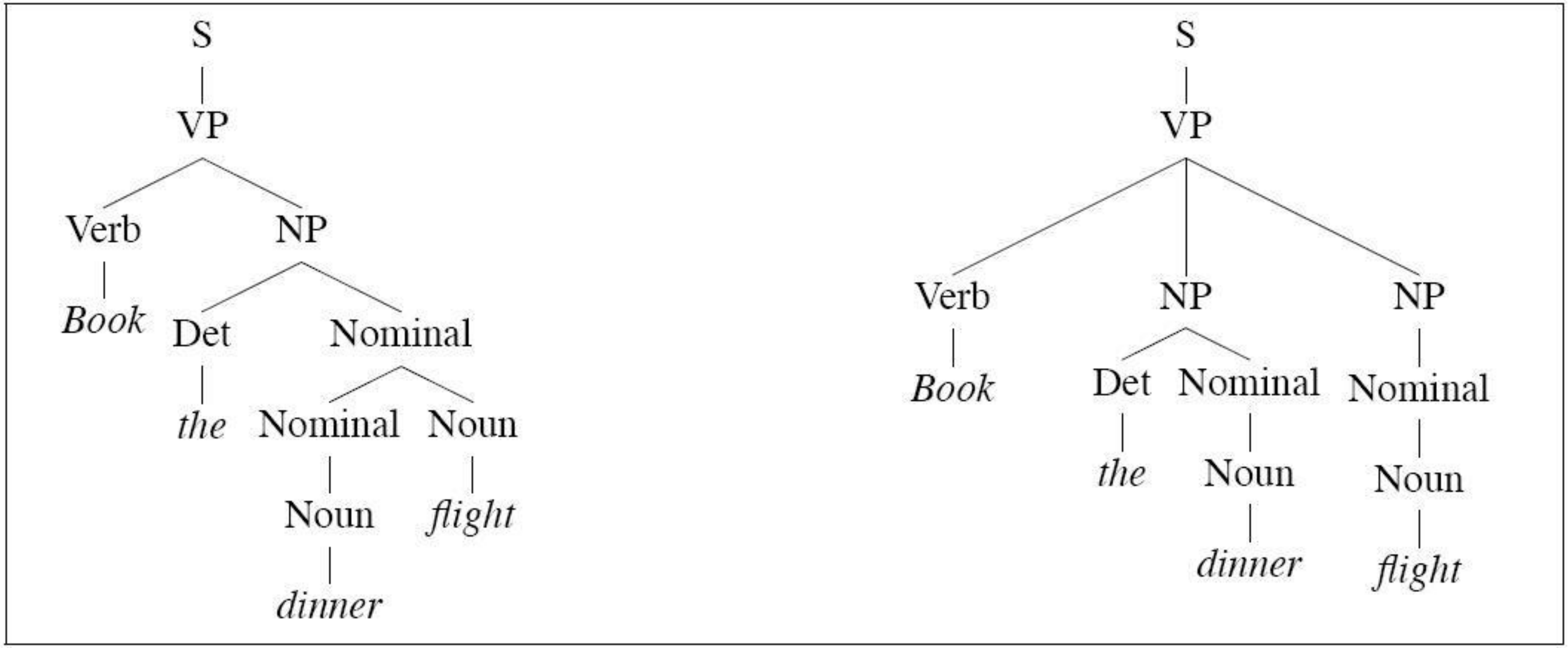
- Calculate the probability of the parse tree below



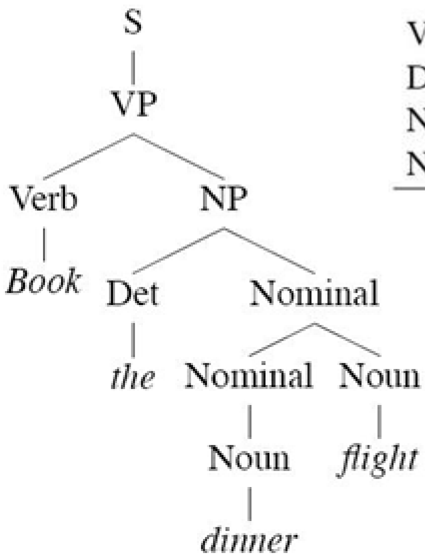
Grammar		Lexicon
S → *NP VP*	[.80]	*Det* → *that* [.10] $\mid$ *a* [.30] $\mid$ *the* [.60]
S → *Aux NP VP*	[.15]	*Noun* → *book* [.10] $\mid$ *flight* [.30]
S → *VP*	[.05]	$\mid$ *meal* [.15] morning [.05]
NP → *Pronoun*	[.35]	$\mid$ *flights* [.40] $\mid$ *dinner* [.10]
NP → *Proper-Noun*	[.30]	*Verb*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

Why Do We Need Probability of Parse Trees?

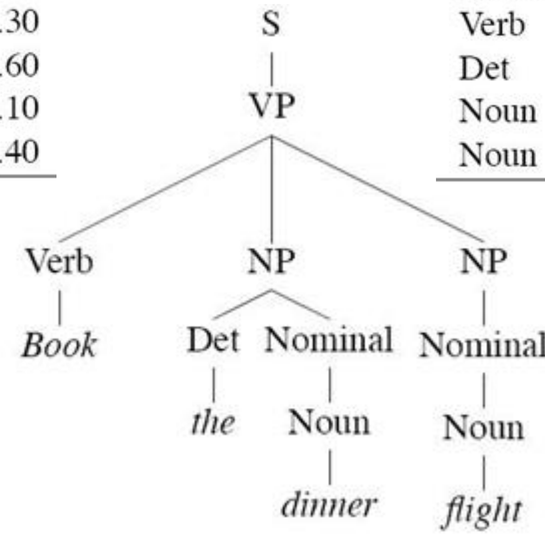
- Once we have probabilities of possible parse trees, we can select the parse tree with the highest probability as the parse result for a given string



Why Do We Need Probability of Parse Trees?



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	→	flights	.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	→	flights	.40

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

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

But How Accurate/General Are These Probabilities?

- Probabilities are bound to specific datasets or corpora and, in general, are not domain-independent

Problems with PCFG

- Doesn't take the actual words (Grammar) into account
  - e.g., verb subcategorization
- Doesn't take into account where in the derivation a rule is used
  - e.g., NPs that are syntactic objects are more likely to be Pronouns.

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

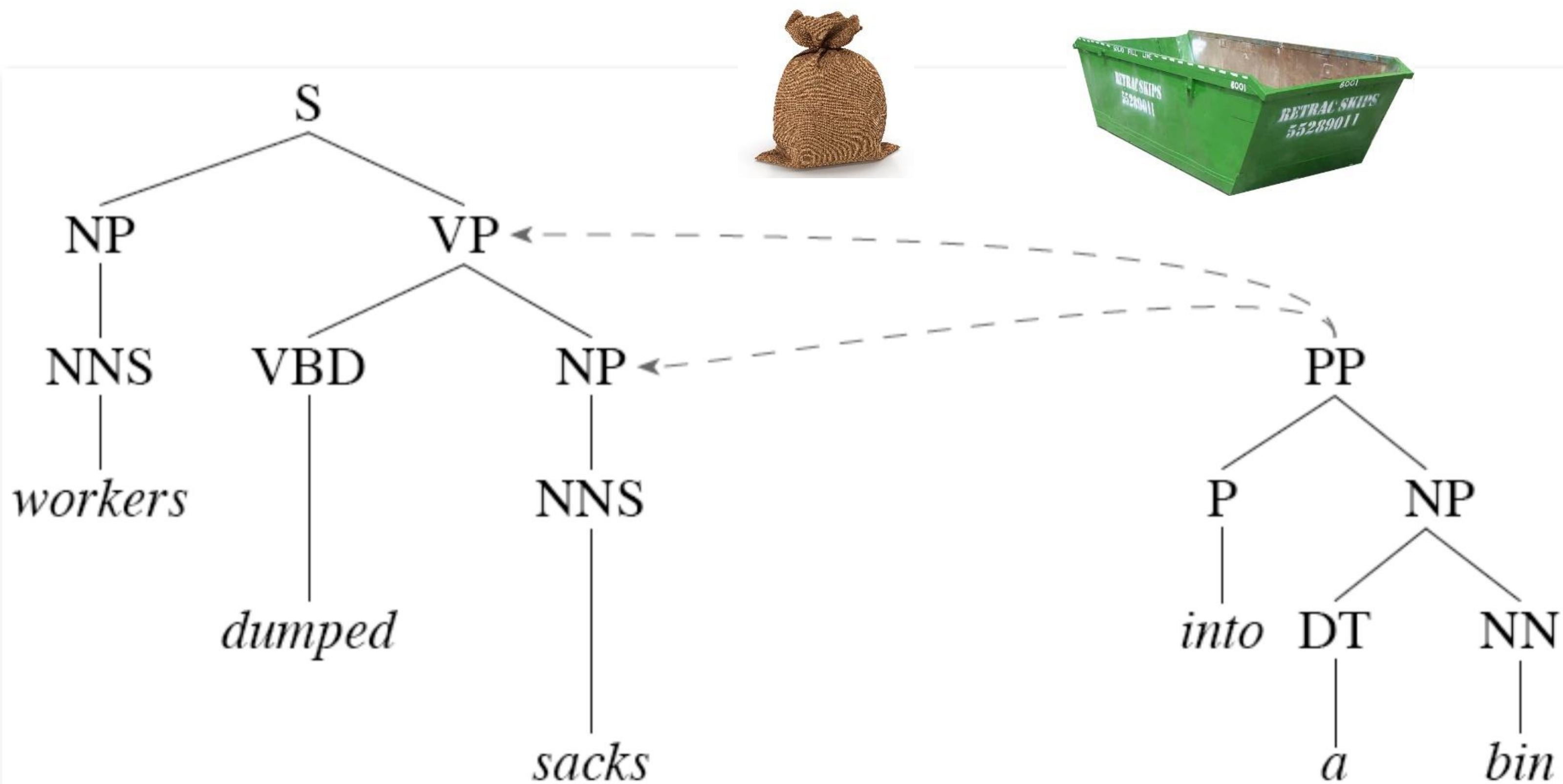
Specific Problems

- Attachment ambiguities
  - Prepositional phrase (PP) attachment
  - Coordination problem

- Structural dependencies between rules

PP Attachment

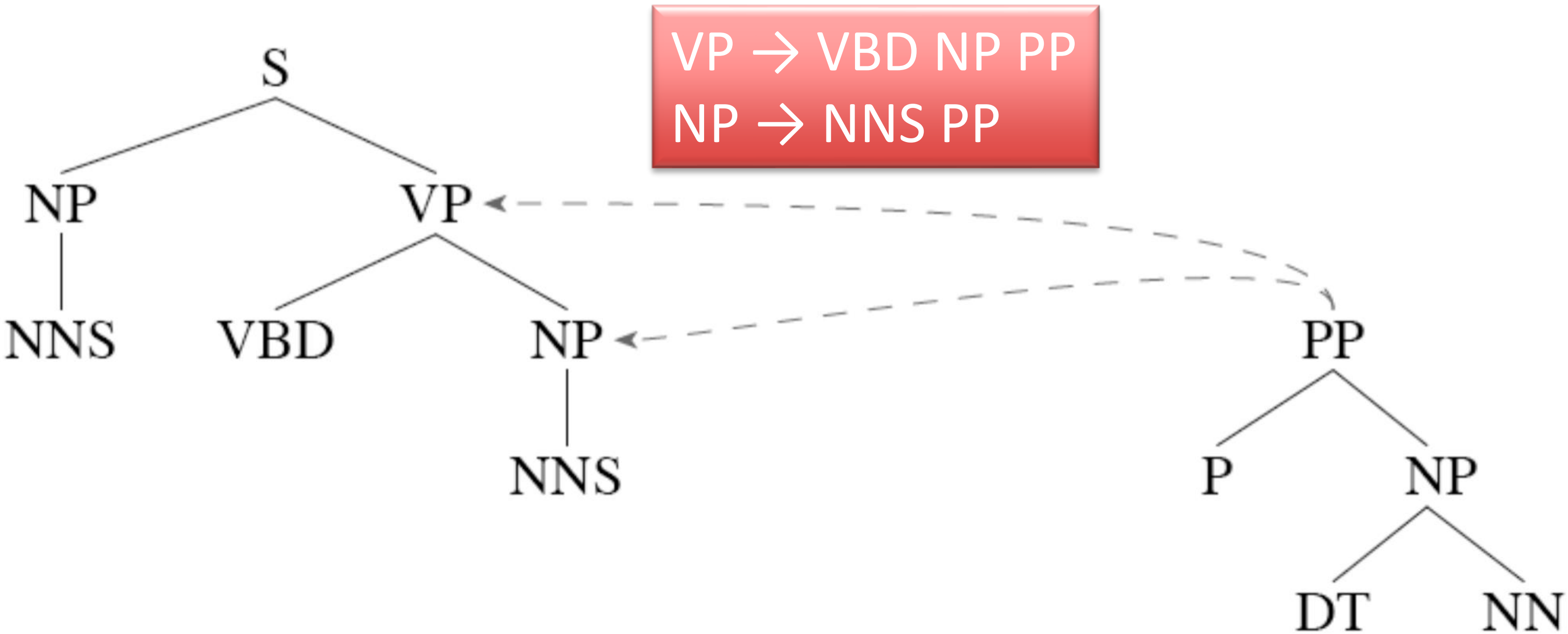
- Example sentence: Workers dumped sacks into a bin.



‘Dump’ has a stronger association with ‘into’

PP Attachment

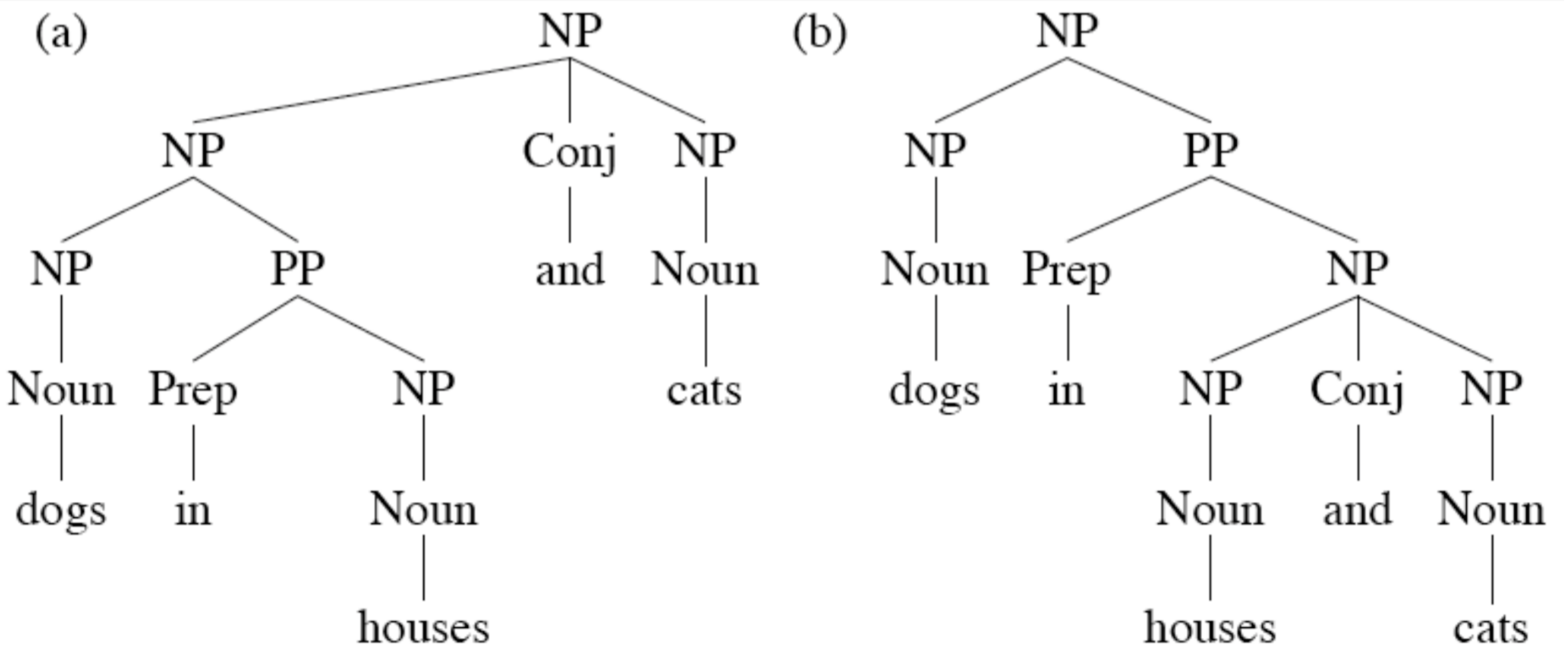
- NNS VBD NNS P DT NN
  - Rule does not consider the actual words in the sentence.
  - So we are not using the actual words here, but only the rules



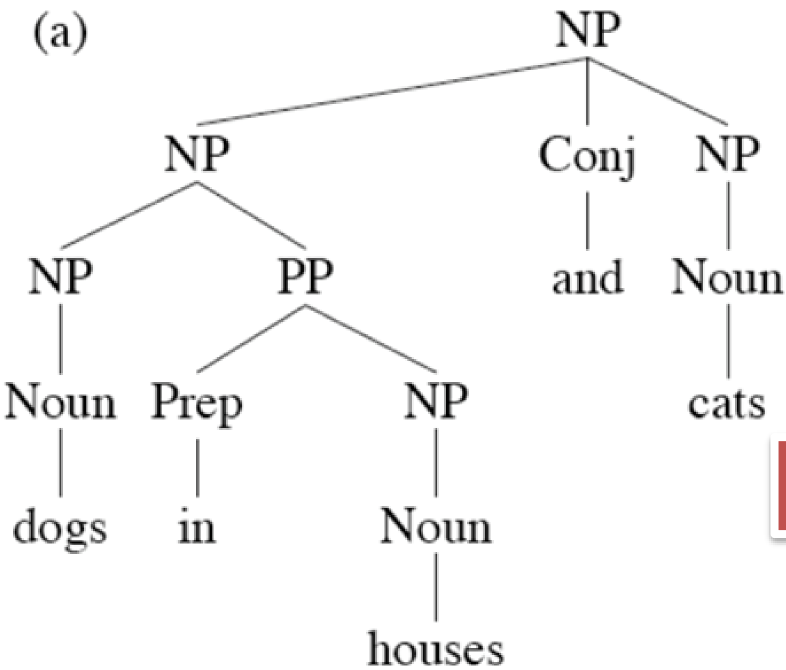
Both rules are valid, and we cannot determine the attachment here.

Coordination Problem

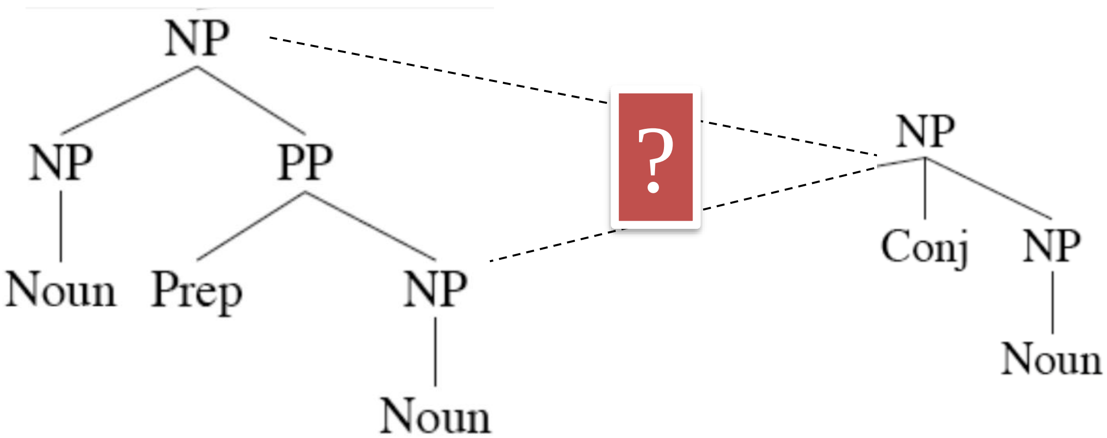
- Most grammars have such (implicit) rules as “X $\rightarrow$ X and X”
  - This leads to massive ambiguity problems.



Coordination Problem



Again the rules do not consider the words



Structural Dependencies Between Rules

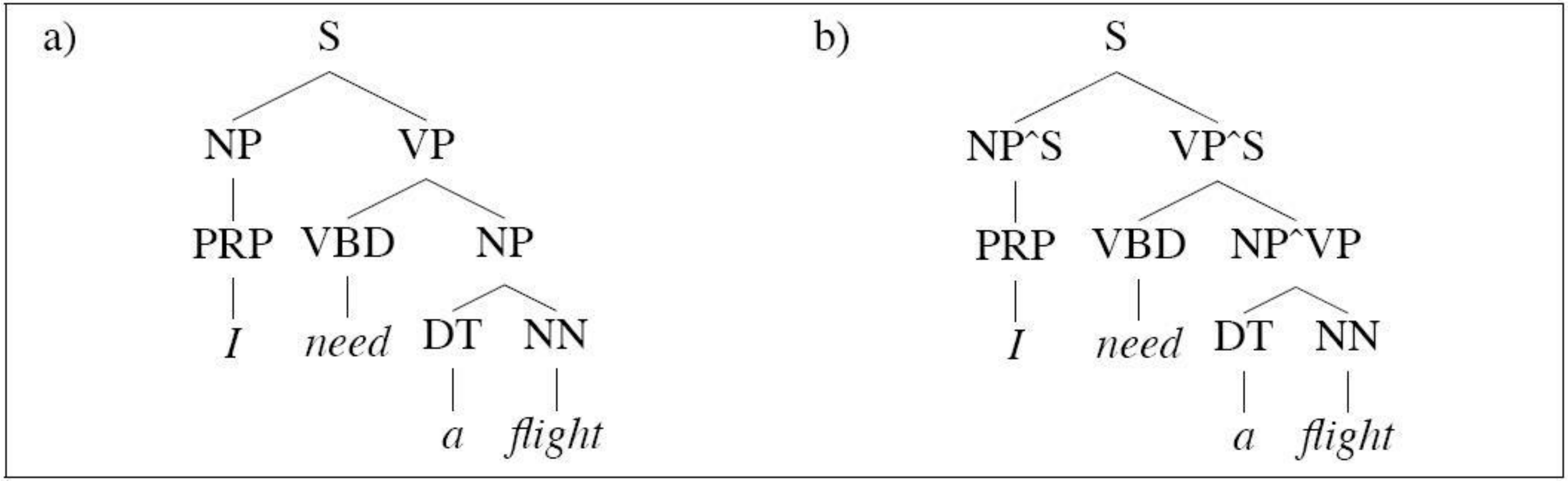
- Example probability for rules of NP
  - NP → DT NN (0.28)
  - NP → Pronoun (0.25)

- Rules that involve NP
  - S → **NP** VP
  - VP → Verb **NP**

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

Improving PCFG: Splitting Non-Terminals

- Encoding contextual dependencies into PCFG symbols
  - NP is a child of S → NP^S
  - NP is a child of VP → NP^VP



Improving PCFG: Splitting Non-Terminals

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

NP^S → Pronoun
NP^VP → Pronoun
NP^PP → Pronoun

NP^S → Det Nominal^NP
NP^VP → Det Nominal^NP
NP^PP → Det Nominal^NP

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

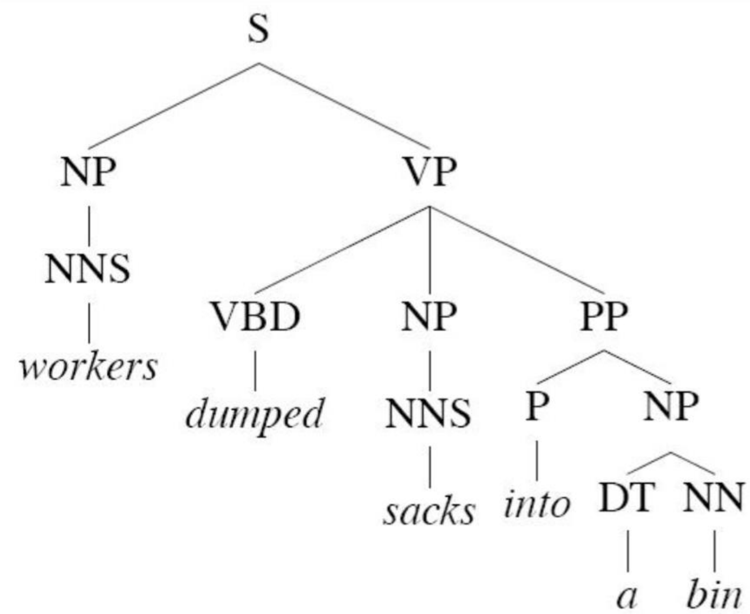
Improving PCFG: Lexicalized PCFG

- How to add lexical information to rules?

- (Review) Lexical head
  - The word in the phrase that is grammatically the most important
  - E.g. N is the head of NP
  - E.g. V is the head of VP

- Now, we put the **lexicon head** into the rules

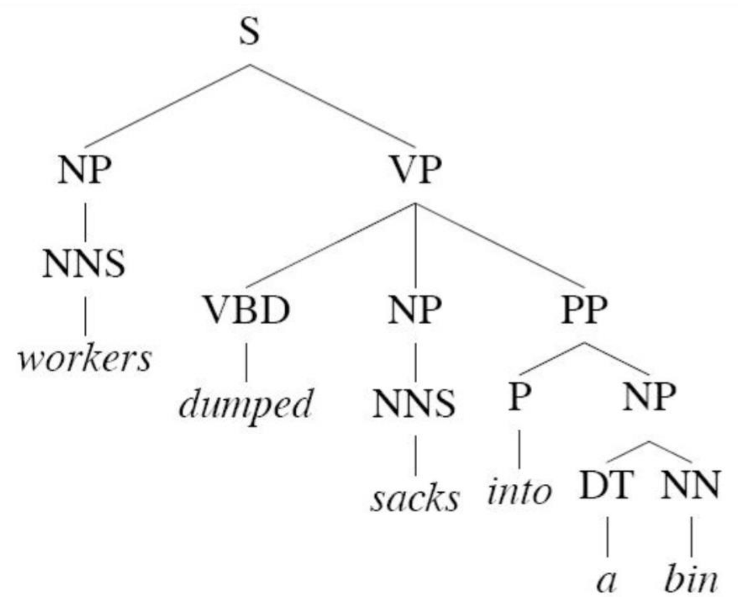
- VP → VBD NP PP
  - VP(**dumped**) → VBD(dumped) NP(sacks) PP(into)
  - VP(**dumped,VBD**) → VBD(dumped,VBD) NP(sacks,NNS) PP(into,P)



Evaluating Parsing Accuracy

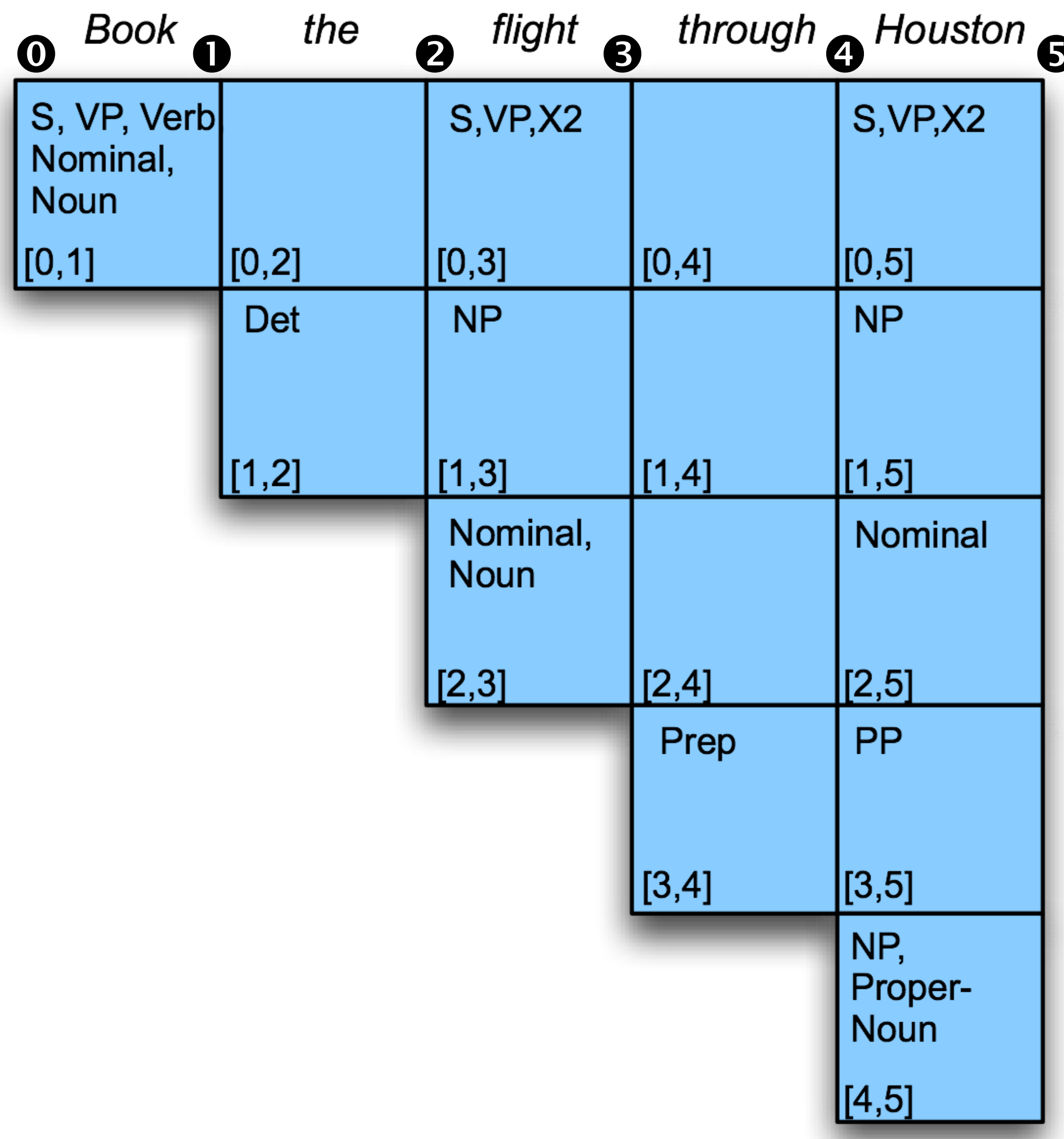
- Sentence-level accuracy
  - But most sentences are not given a completely correct parse by any existing parser

- Constituent-level accuracy
  - Constituent as labeled span: [label, start, finish], e.g. [NP, 0, 1]



Evaluating Parsing Accuracy: Labeled Spans

- [S, 0, 5]
- [VP, 0, 5]
- [Verb, 0, 1]
- [NP, 1, 5]
- [Det, 1, 2]
- [Nominal, 2, 5]
- [Noun, 2, 3]
- [PP, 3, 5]
- [Prep, 3, 4]
- [NP, 4, 5]
- [Noun, 4, 5]



Example

Ground truth

```
(ROOT
  (S
    (INTJ (VB Please))
    (VP (VB repeat)
      (NP (DT that)))
    (. .)))
```

```
(ROOT
  (S
    (ADVP (RB Please))
    (VP (VB repeat)
      (NP (DT that)))
    (. .)))
```

TP = 5,

FP = 2,

FN = 2

(S, 0, 3)
(INTJ, 0, 1) (VB, 0, 1)
(VP, 1, 3) (VB, 1, 2)
(NP, 2, 3) (DT, 2, 3)

(S, 0, 3)
(ADVP, 0, 1) (RB, 0, 1)
(VP, 1, 3) (VB, 1, 2)
(NP, 2, 3) (DT, 2, 3)

Example

(S (NP (PRP I))
(VP (VBP need)
(S (VP (TO to)
(VP (VB fly)
(PP (IN between)
(NP (NNP Philadelphia)
(CC and)
(NNP Atlanta))))))) (. .)))

Ground truth

(S (NP (PRP I))
(VP (VBP need)
(S (VP (TO to)
(VP (VB fly)
(PP (IN between)
(NP
(NP (NNP Philadelphia))
(CC and)
(NP (NNP Atlanta)))))))) (. .)))

(S,0,8) (NP,0,1) (PRP,0,1)
(VP,1,8) (VBP,1,2)
(S,2,8) (VP,2,8) (TO,2,3)
(VP,3,8) (VB,3,4)
(PP,4,8) (IN,4,5)
(NP,5,8) (NNP,5,6)
(CC,6,7) (NNP,7,8)

(S,0,8) (NP,0,1) (PRP,0,1)
(VP,1,8) (VBP,1,2)
(S,2,8) (VP,2,8) (TO,2,3)
(VP,3,8) (VB,3,4)
(PP,4,8) (IN,4,5)
(NP,5,8) (NP,5,6) (NNP,5,6)
(CC,6,7) (NP,7,8) (NNP,7,8)

TP = 16,

FP = 0,

FN = 2

Evaluating Parsing Accuracy: Measures

$$\text{Precision}\,(P) = \frac{TP}{TP + FP}$$

$$\text{Recall}(R) = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2PR}{P + R}$$