Natural Language Processing: Syntactic Parsing

Context Free Grammars (CFG)

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

Simple CFG

Grammar

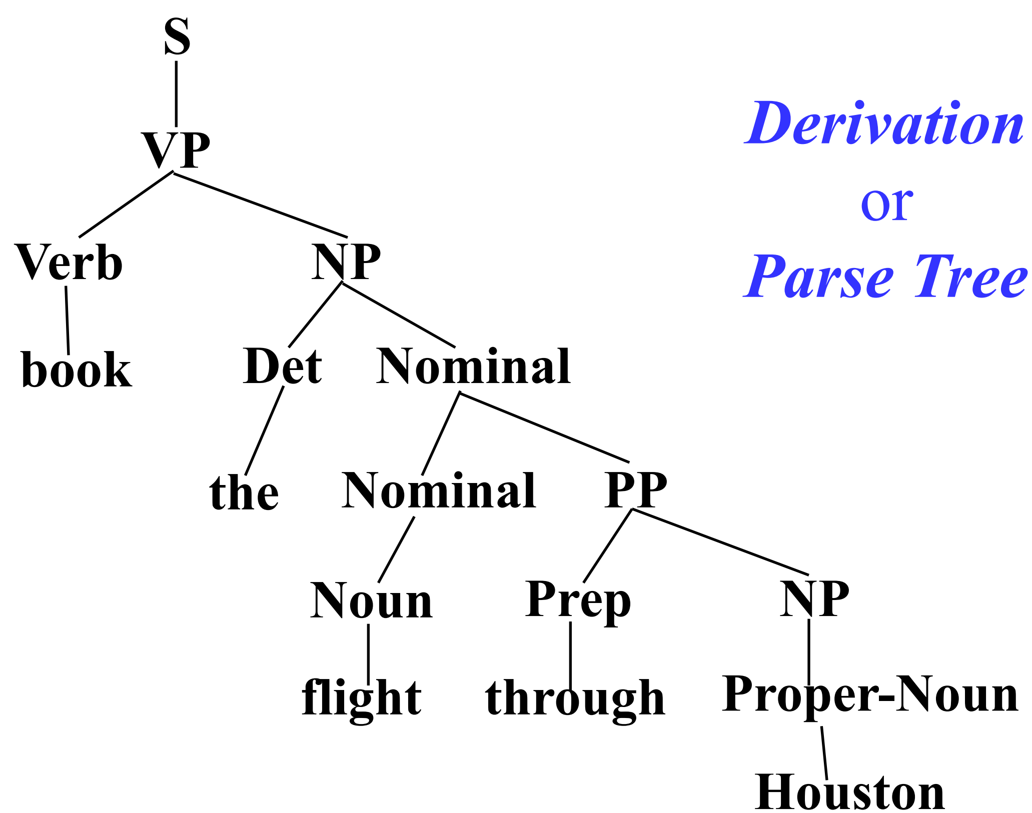
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Sentence Generation

- Sentences are generated by recursively rewriting the start symbol using the productions until only terminals symbols remain.



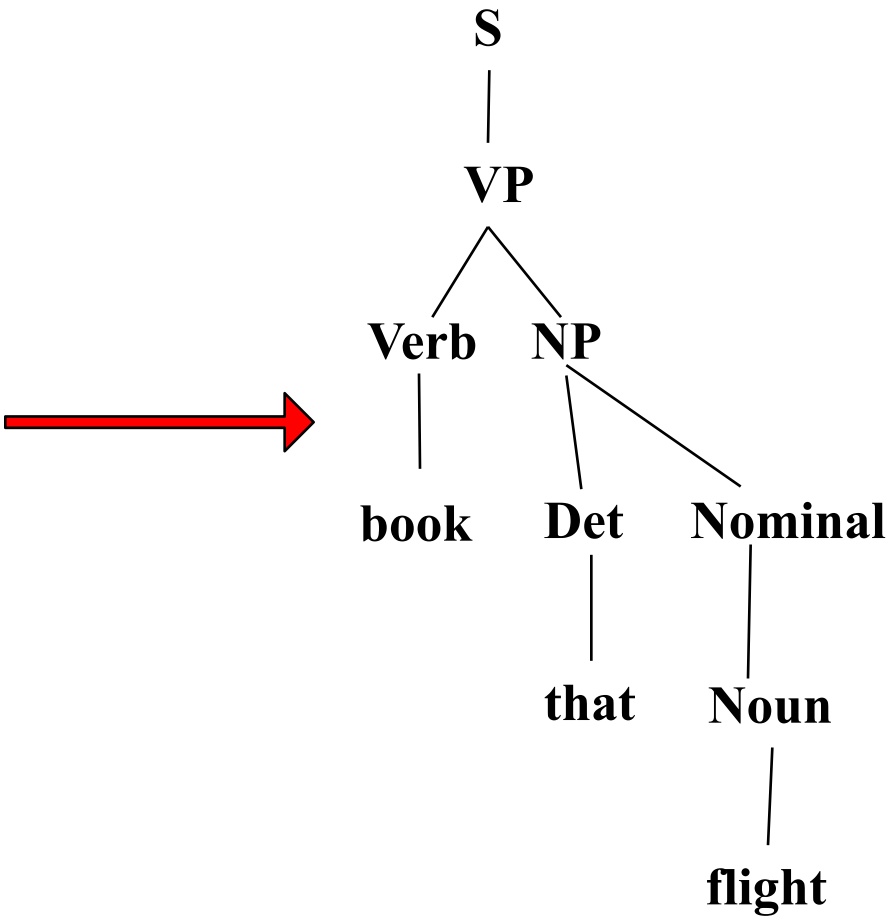
Derivation
or
Parse Tree

Parsing

- Given a string of terminals and a CFG, determine if the string can be generated by the CFG.
  - Also return a parse tree for the string
  - Also return all possible parse trees for the string
- Must search space of derivations for one that derives the given string.
  - **Top-Down Parsing**: Start searching space of derivations for the start symbol.
  - **Bottom-up Parsing**: Start search space of reverse deivations from the terminal symbols in the string.

Parsing Example

book that flight



Simple CFG

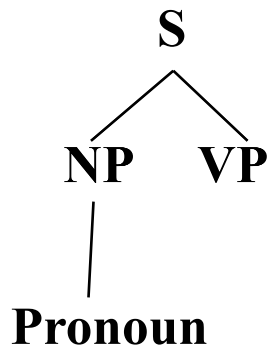
Grammar

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 → VP PP
PP → Prep NP

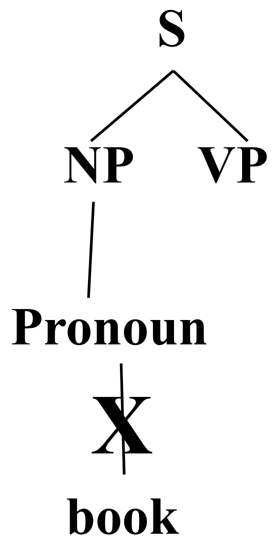
Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

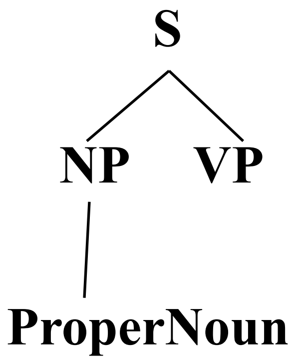
Top Down Parsing



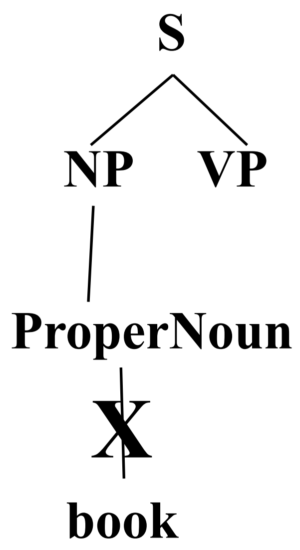
Top Down Parsing



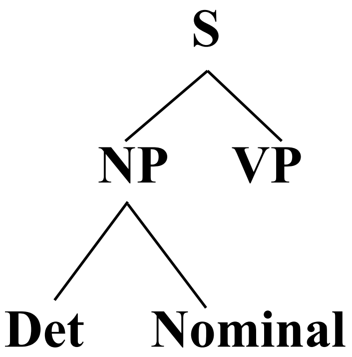
Top Down Parsing



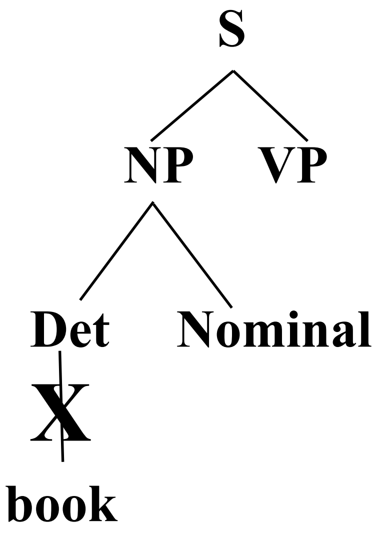
Top Down Parsing



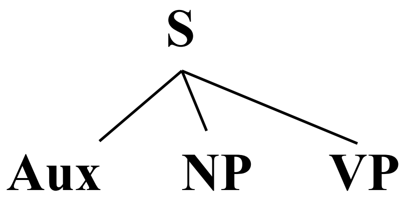
Top Down Parsing



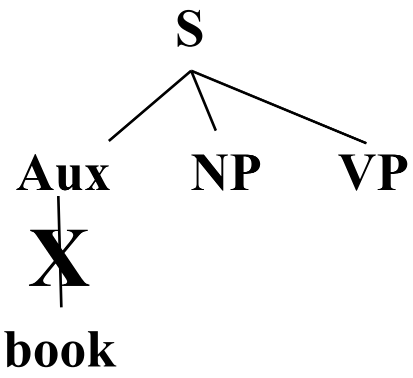
Top Down Parsing



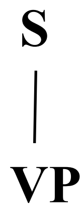
Top Down Parsing



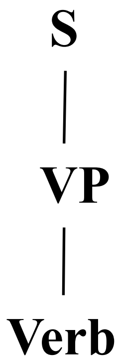
Top Down Parsing



Top Down Parsing

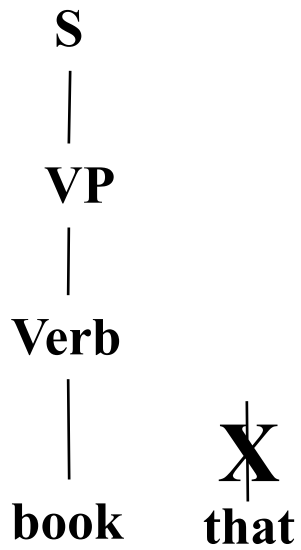


Top Down Parsing

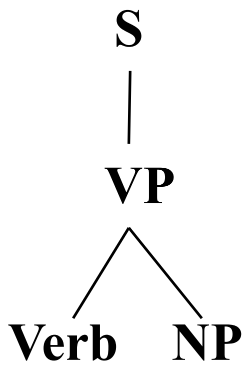


Top Down Parsing

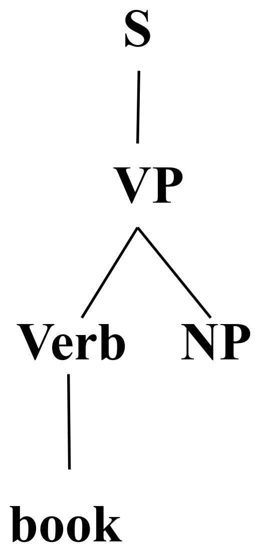
Top Down Parsing



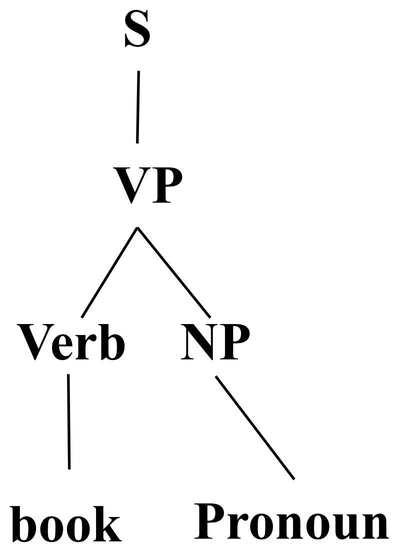
Top Down Parsing



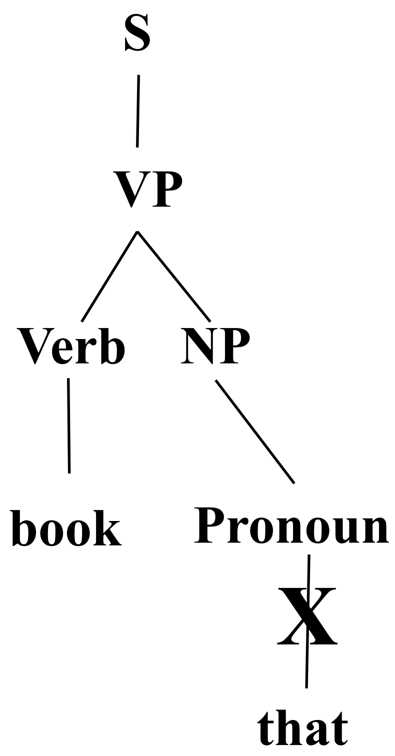
Top Down Parsing



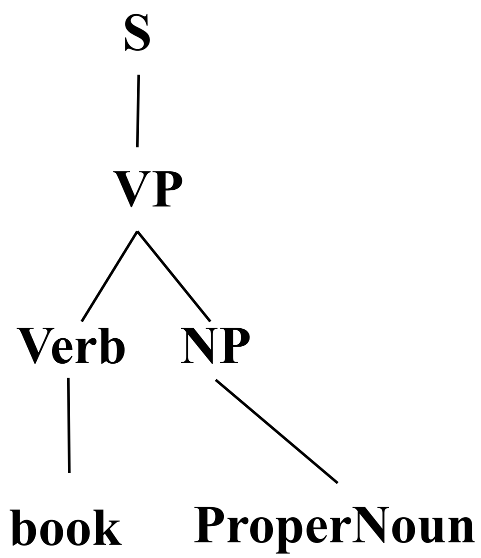
Top Down Parsing



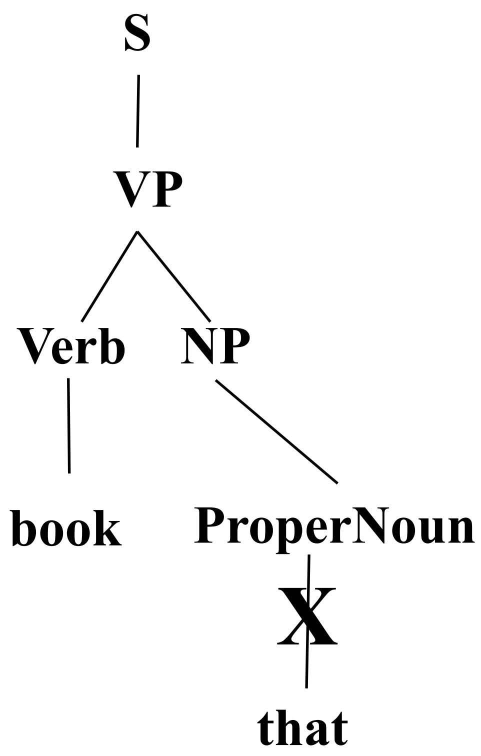
Top Down Parsing



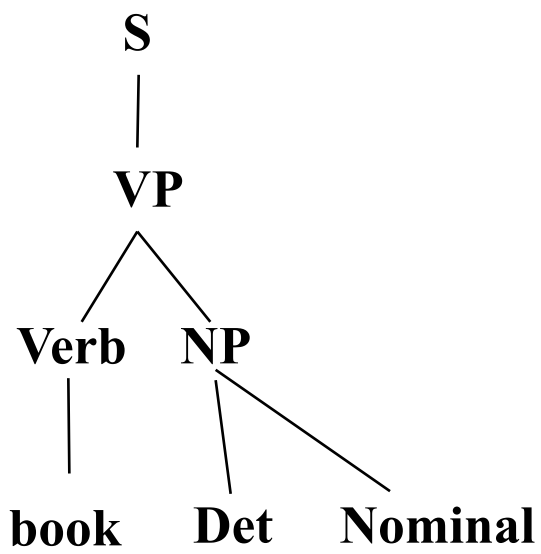
Top Down Parsing



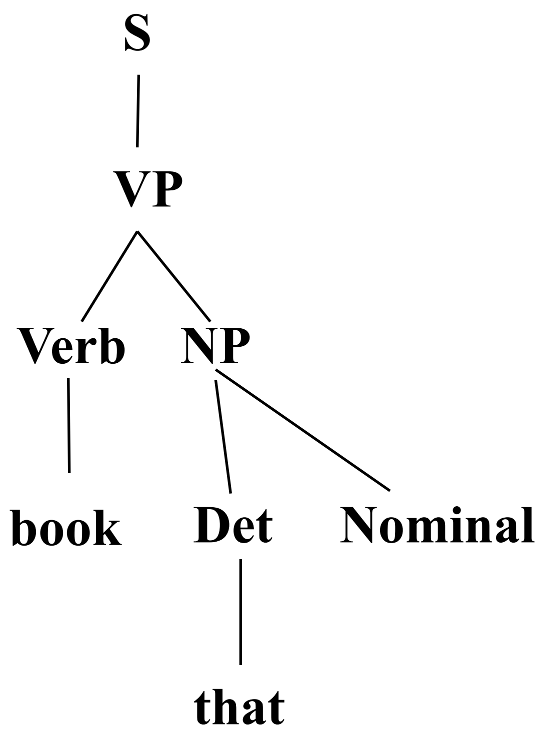
Top Down Parsing



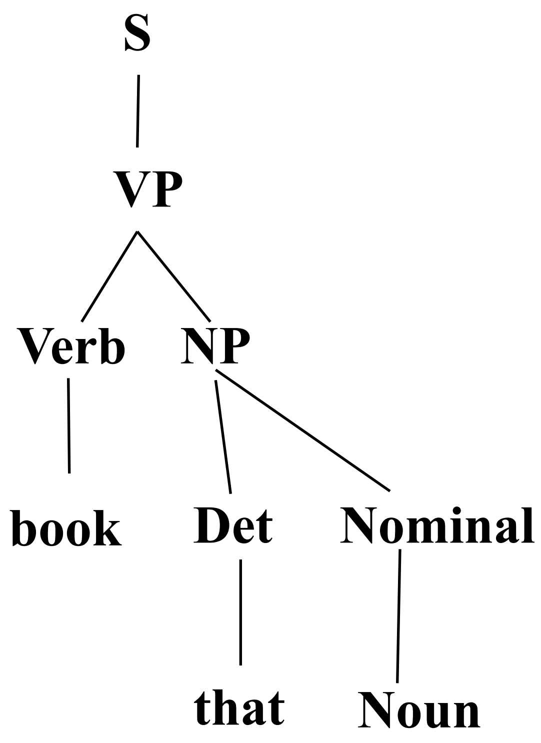
Top Down Parsing



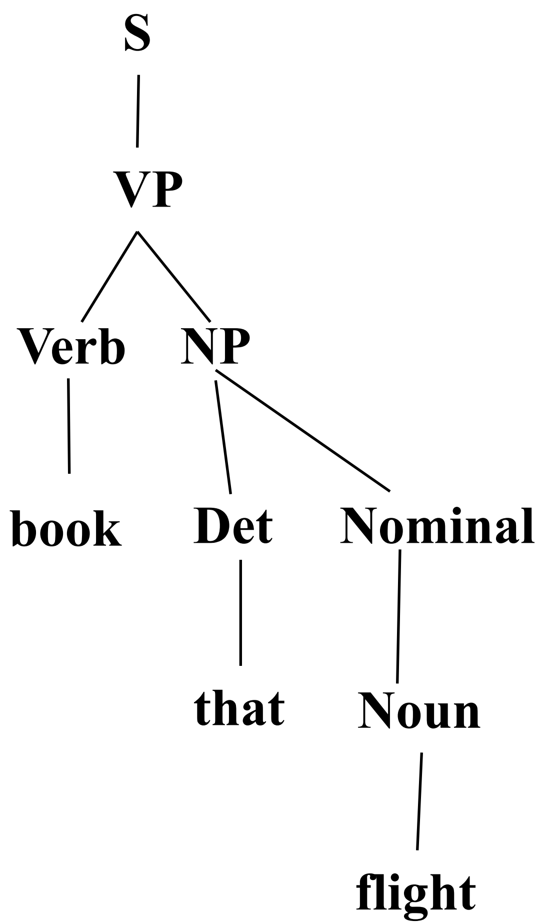
Top Down Parsing



Top Down Parsing



Top Down Parsing



Simple CFG

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 → VP PP
PP → Prep NP

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Bottom Up Parsing

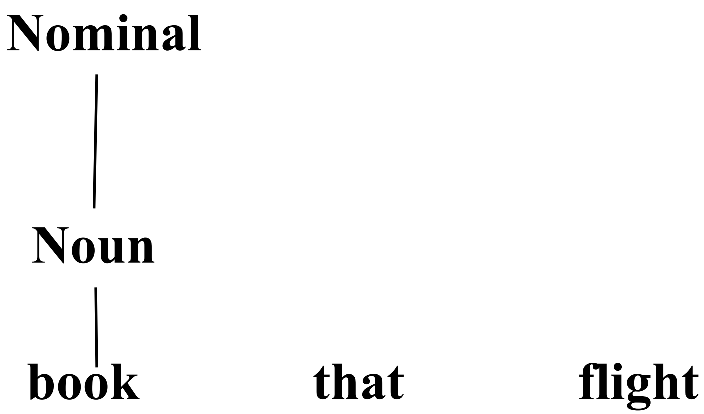
book **that** **flight**

Bottom Up Parsing

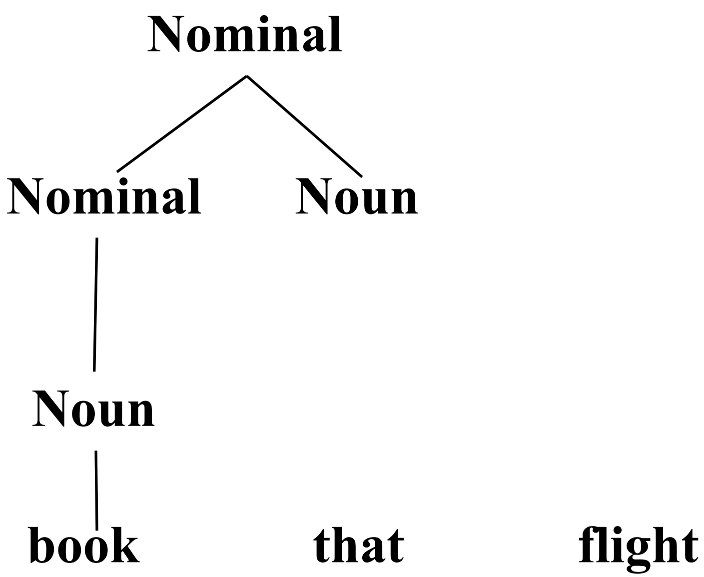
Noun

book **that** **flight**

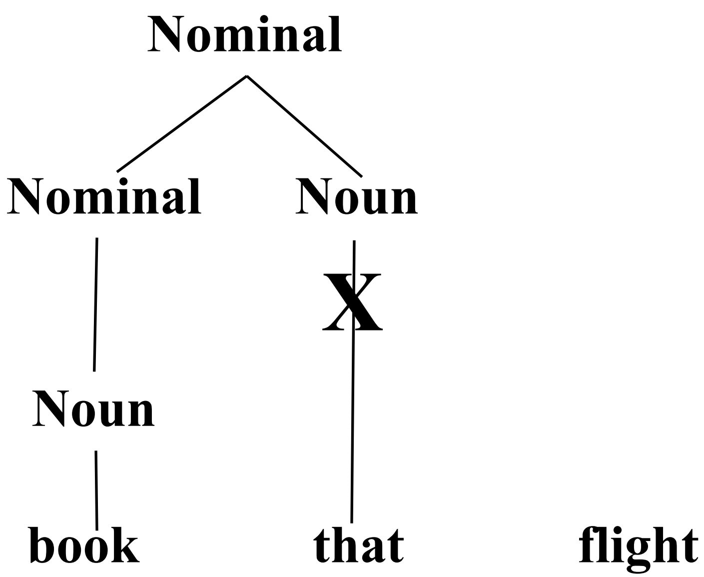
Bottom Up Parsing



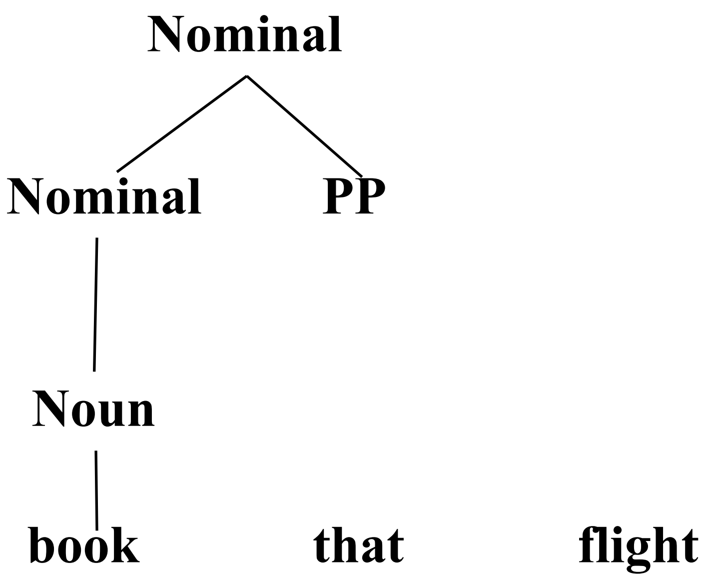
Bottom Up Parsing



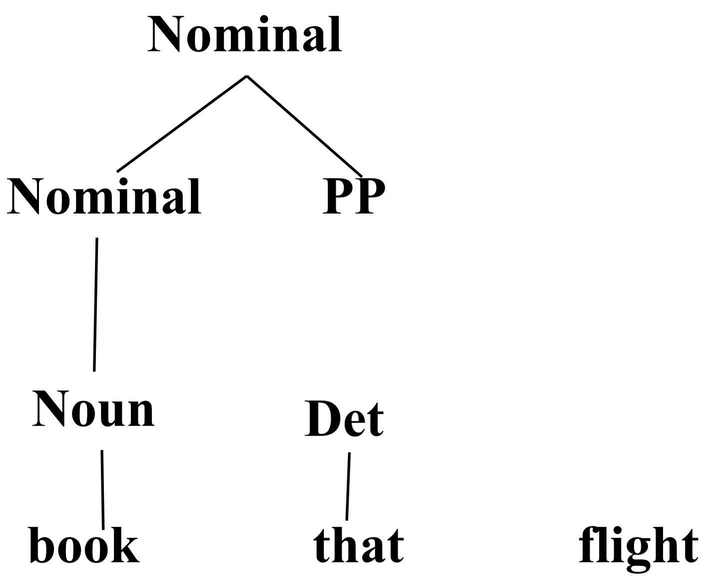
Bottom Up Parsing



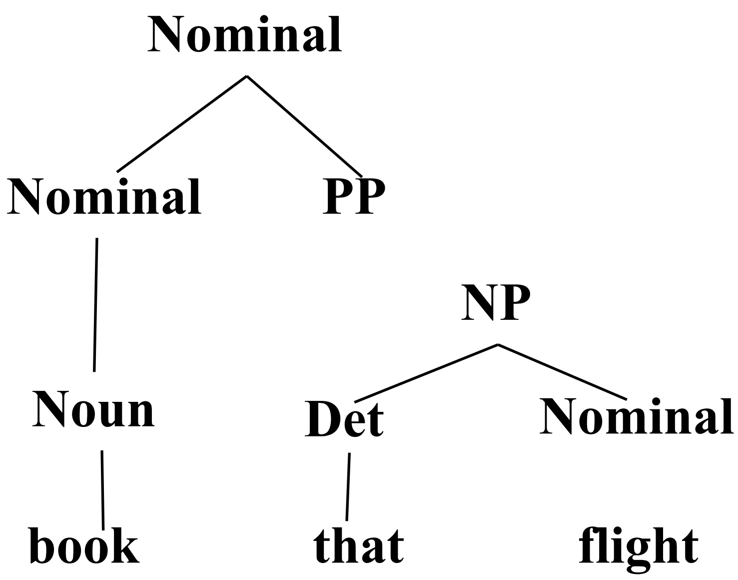
Bottom Up Parsing



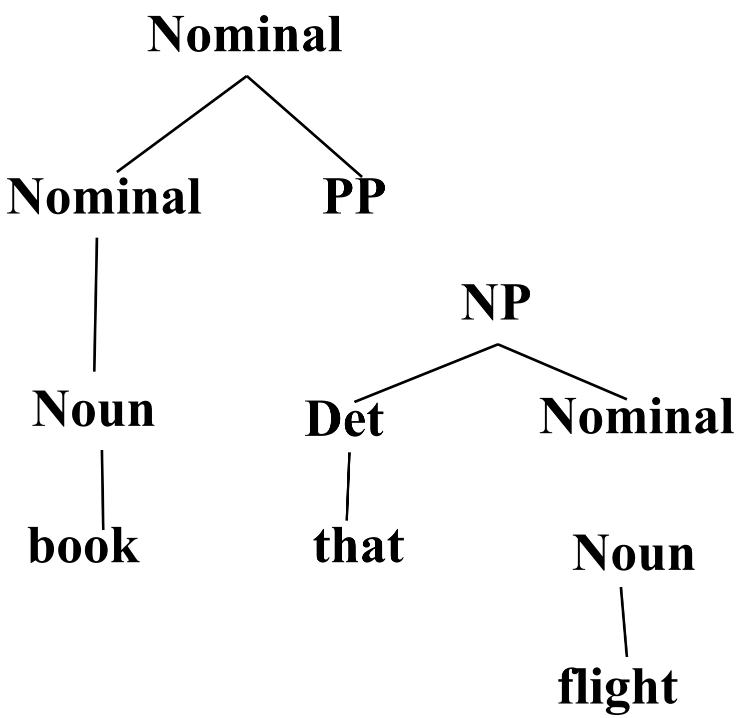
Bottom Up Parsing



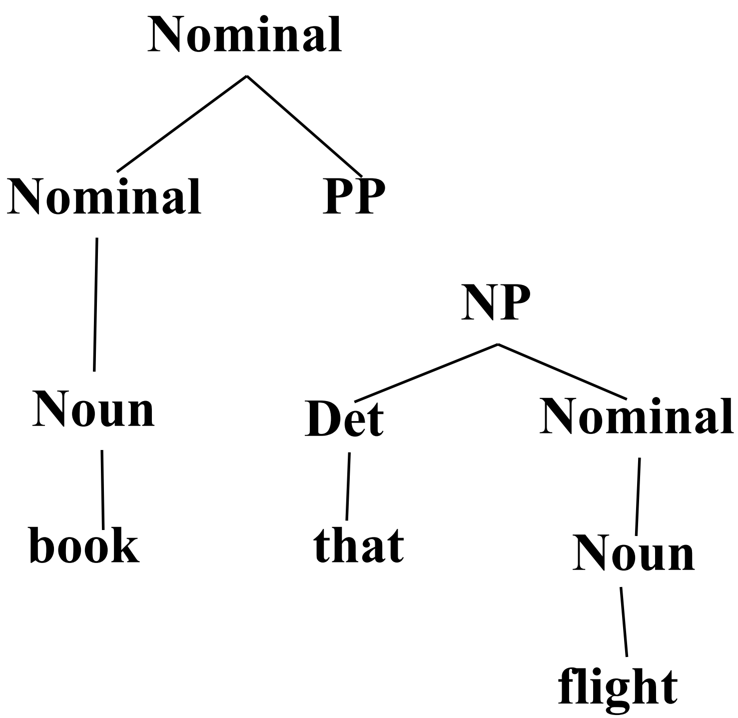
Bottom Up Parsing



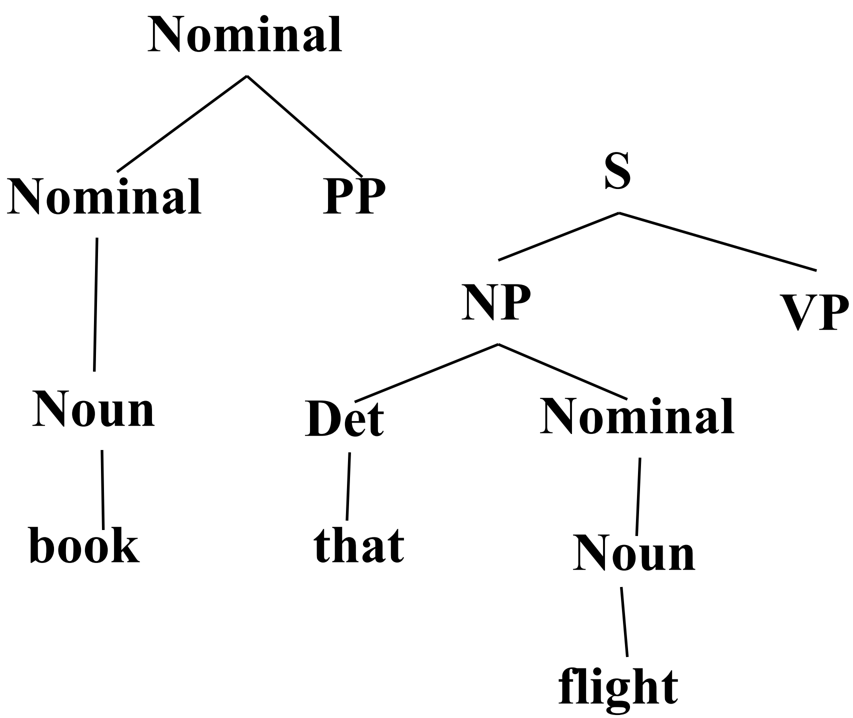
Bottom Up Parsing



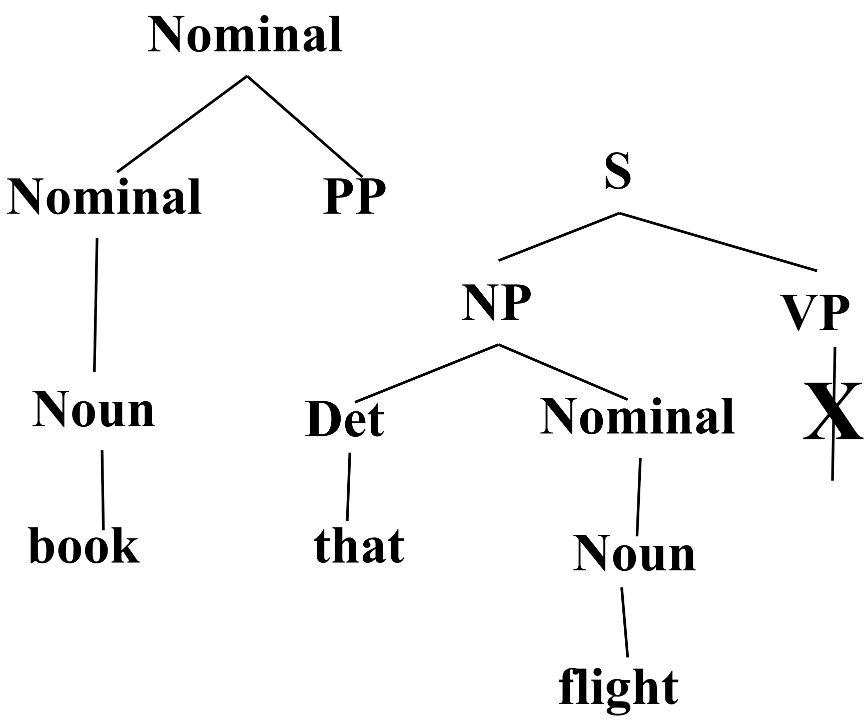
Bottom Up Parsing



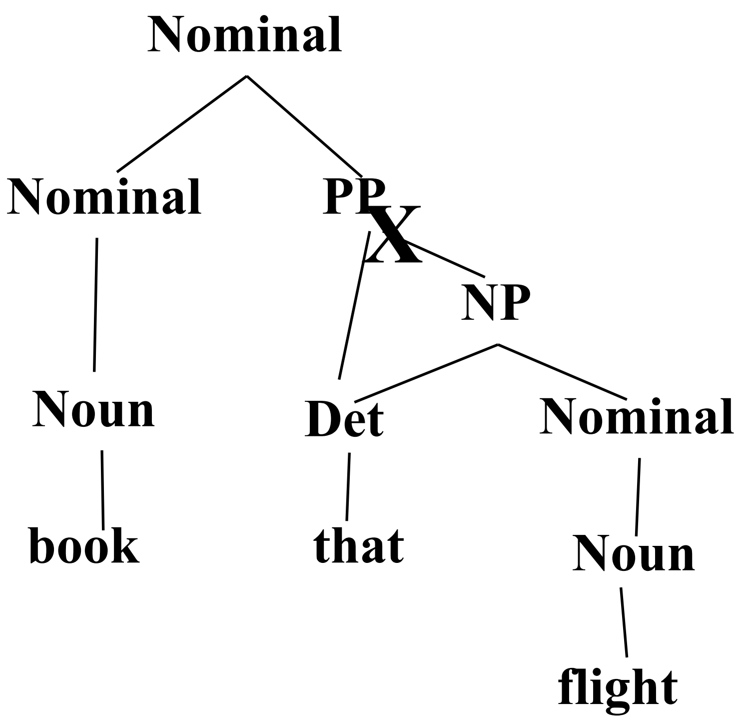
Bottom Up Parsing



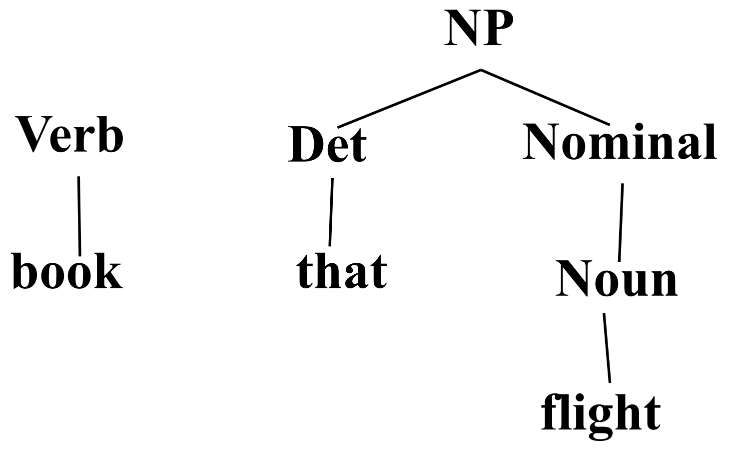
Bottom Up Parsing



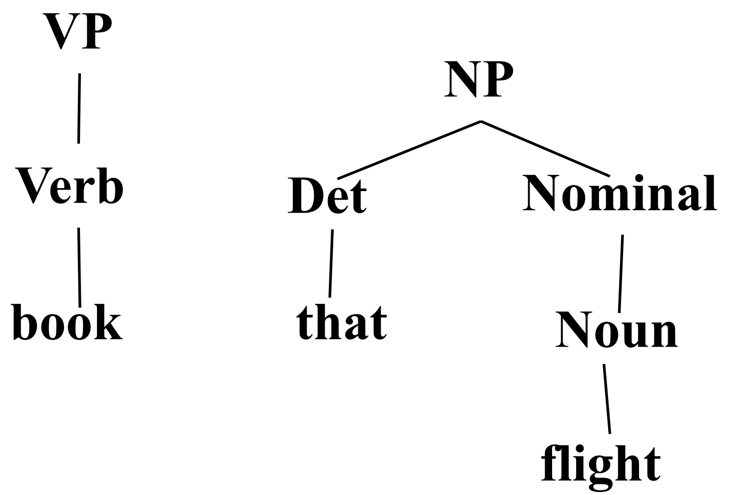
Bottom Up Parsing



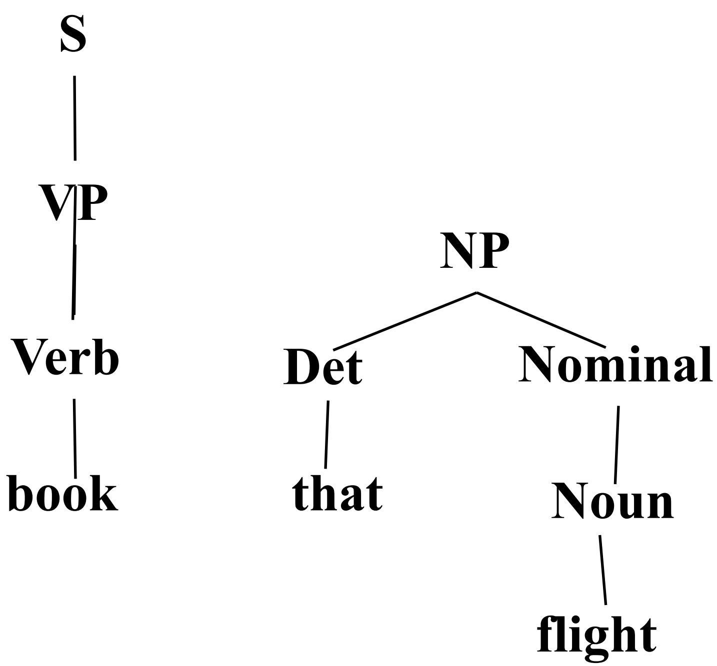
Bottom Up Parsing



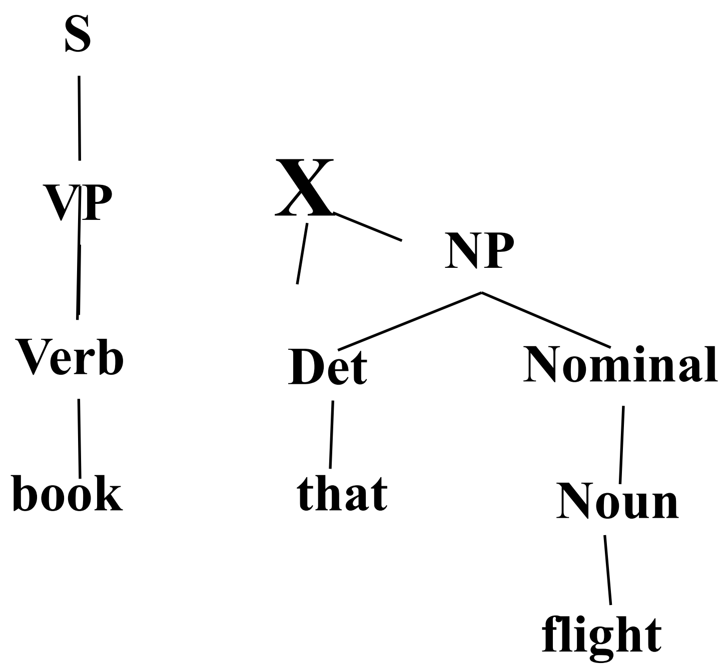
Bottom Up Parsing



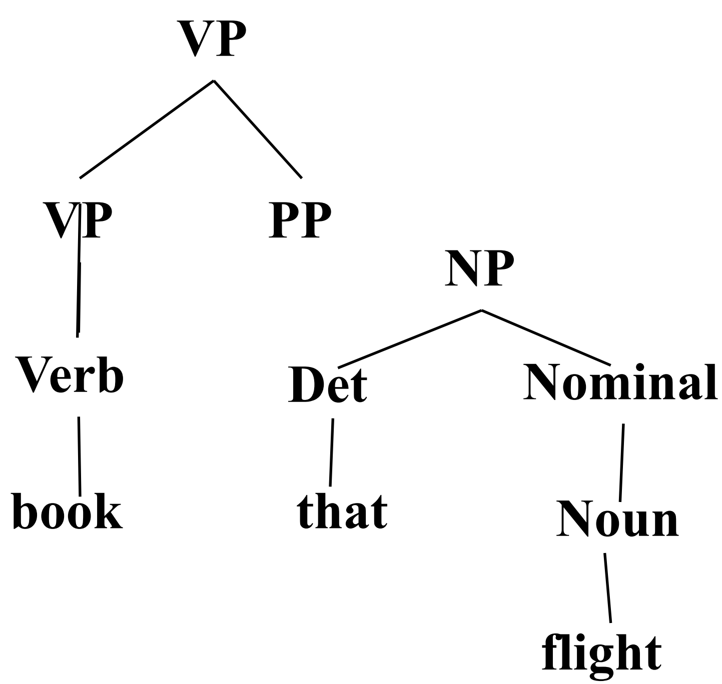
Bottom Up Parsing



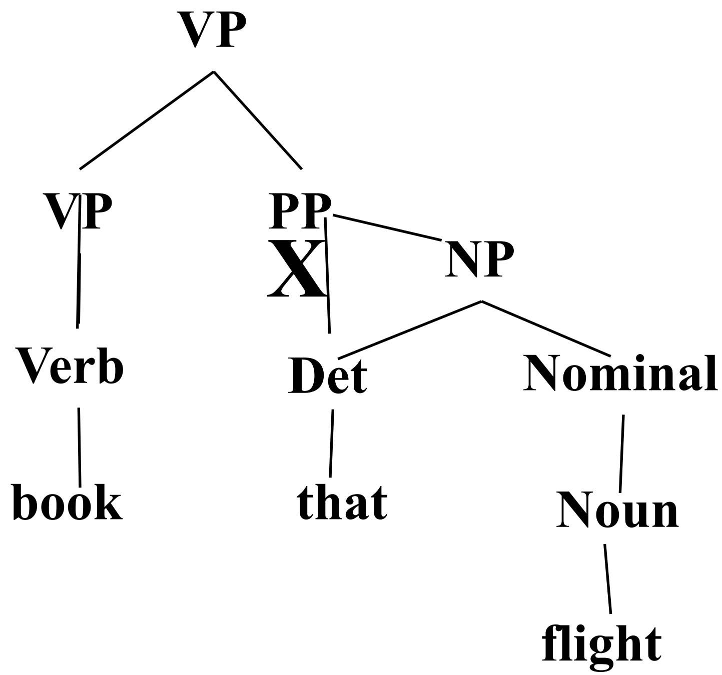
Bottom Up Parsing



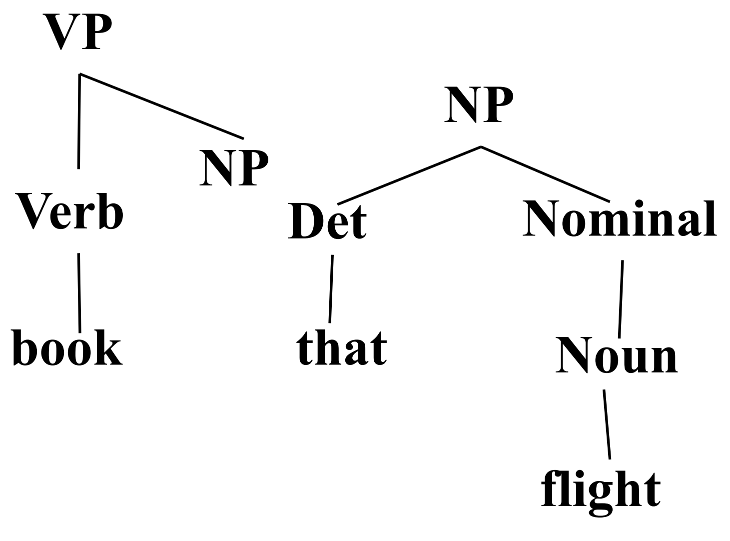
Bottom Up Parsing



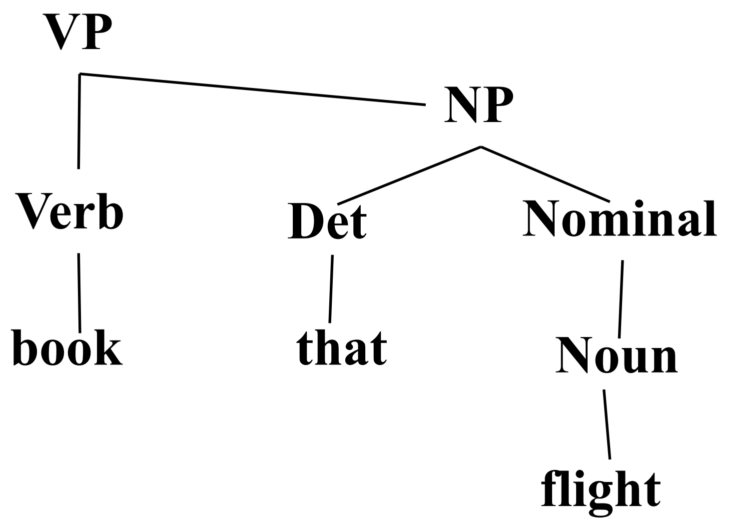
Bottom Up Parsing



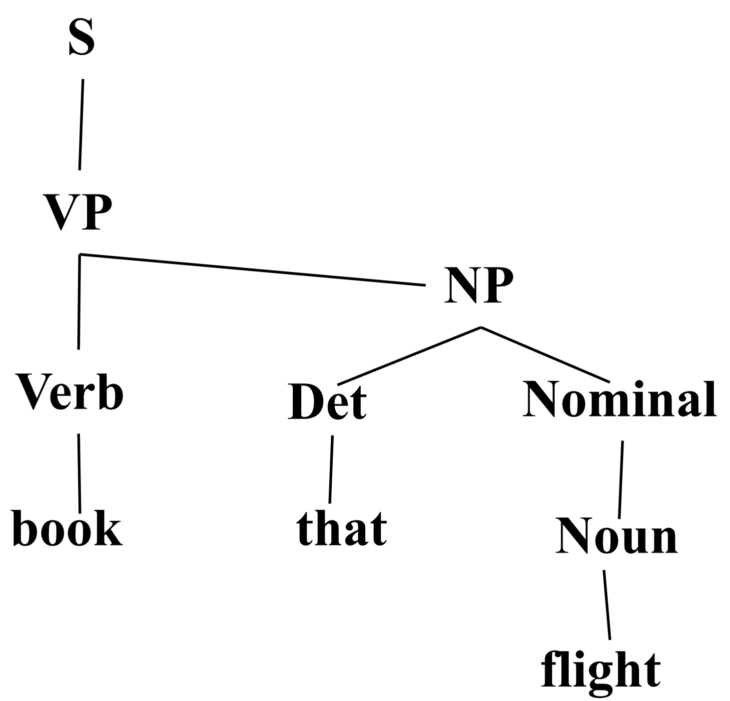
Bottom Up Parsing



Bottom Up Parsing



Bottom Up Parsing



Top Down vs. Bottom Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where n is the length of the input string.

Dynamic Programming Parsing Methods

- **CKY** (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and requires first normalizing the grammar.
- **Earley parser** is based on top-down parsing and does not require normalizing grammar but is more complex.
- More generally, **chart parsers** retain completed phrases in a chart and can combine top-down and bottom-up search.

CKY

- First grammar must be converted to **Chomsky normal form (CNF)** in which productions must have either exactly 2 non-terminal symbols on the RHS or 1 terminal symbol (lexicon rules).
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart).

Grammar Conversion

Original 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 → VP PP
PP → Prep NP

Chomsky Normal Form

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → VP PP
PP → Prep NP

Simple CFG

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 → VP PP
PP → Prep NP

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

CKY Parser

i=	Book j= 1	the 2	flight 3	through 4	Houston 5
0					
1					
2					
3					
4					

Cell[*i*,*j*] contains all constituents (non-terminals) covering words *i* +1 through *j*

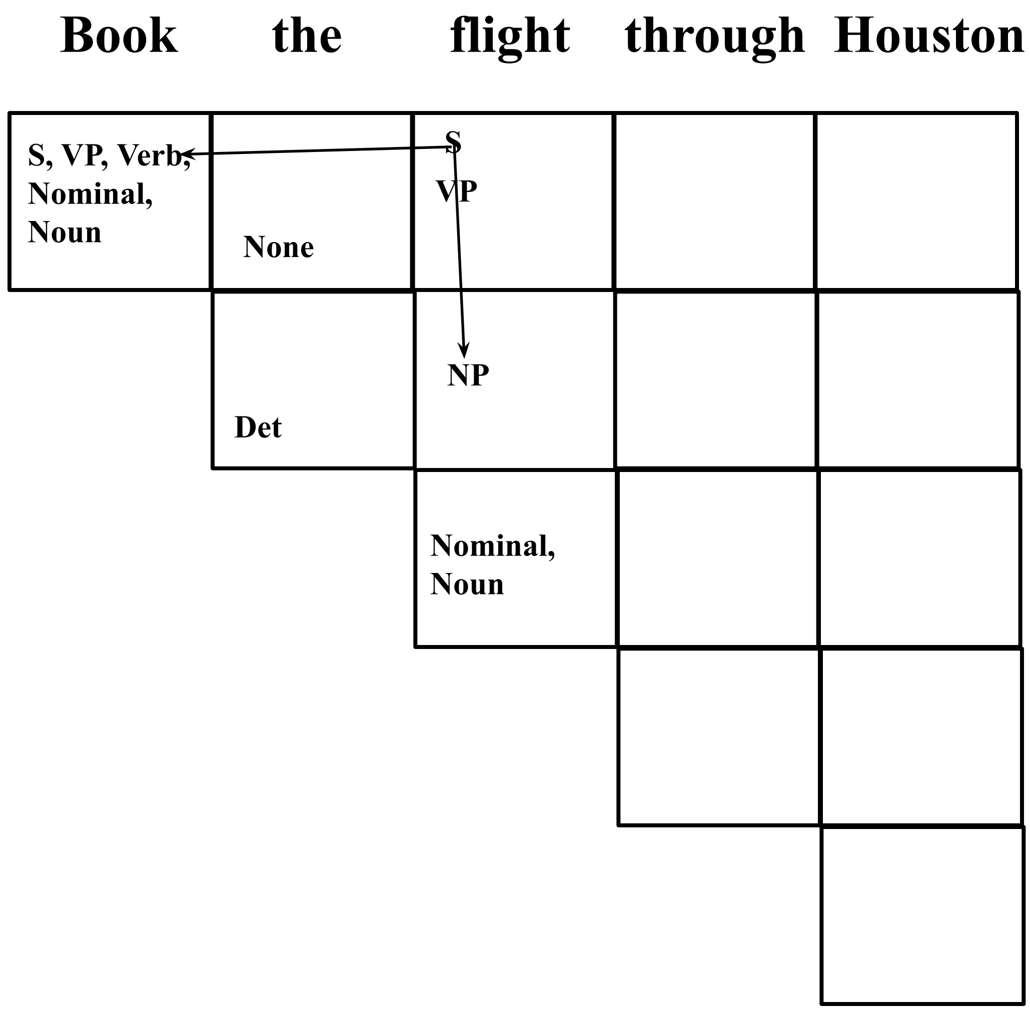
CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None			
	Det	NP		
		Nominal, Noun		

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	VP		
	Det	NP		
		Nominal, Noun		

CKY Parser


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
None
S
VP
Det
NP
Nominal, Noun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP		
	Det	NP		
		Nominal, Noun		

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	
			Prep	

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	
			Prep	PP
				NP ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

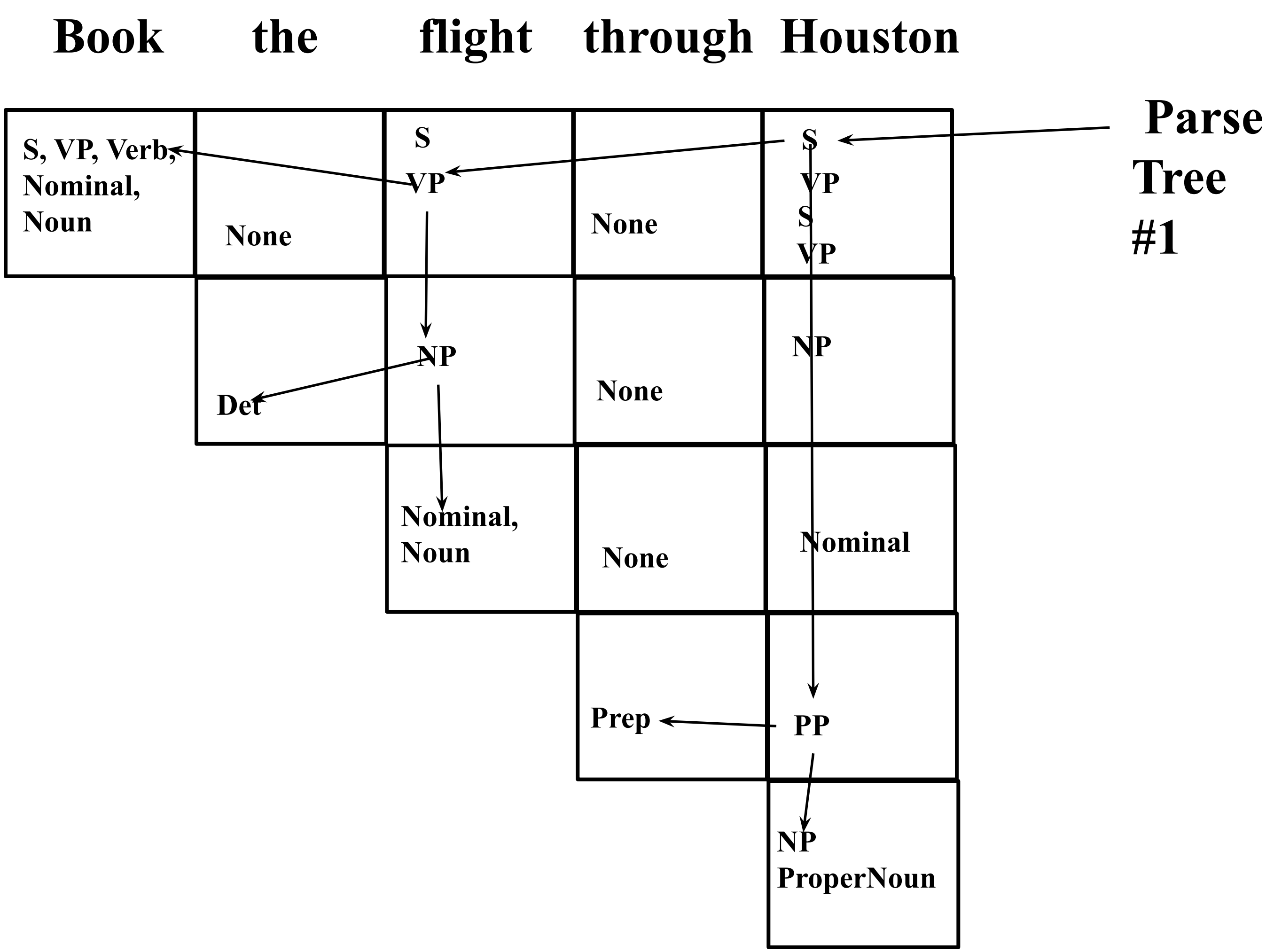
CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	VP S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

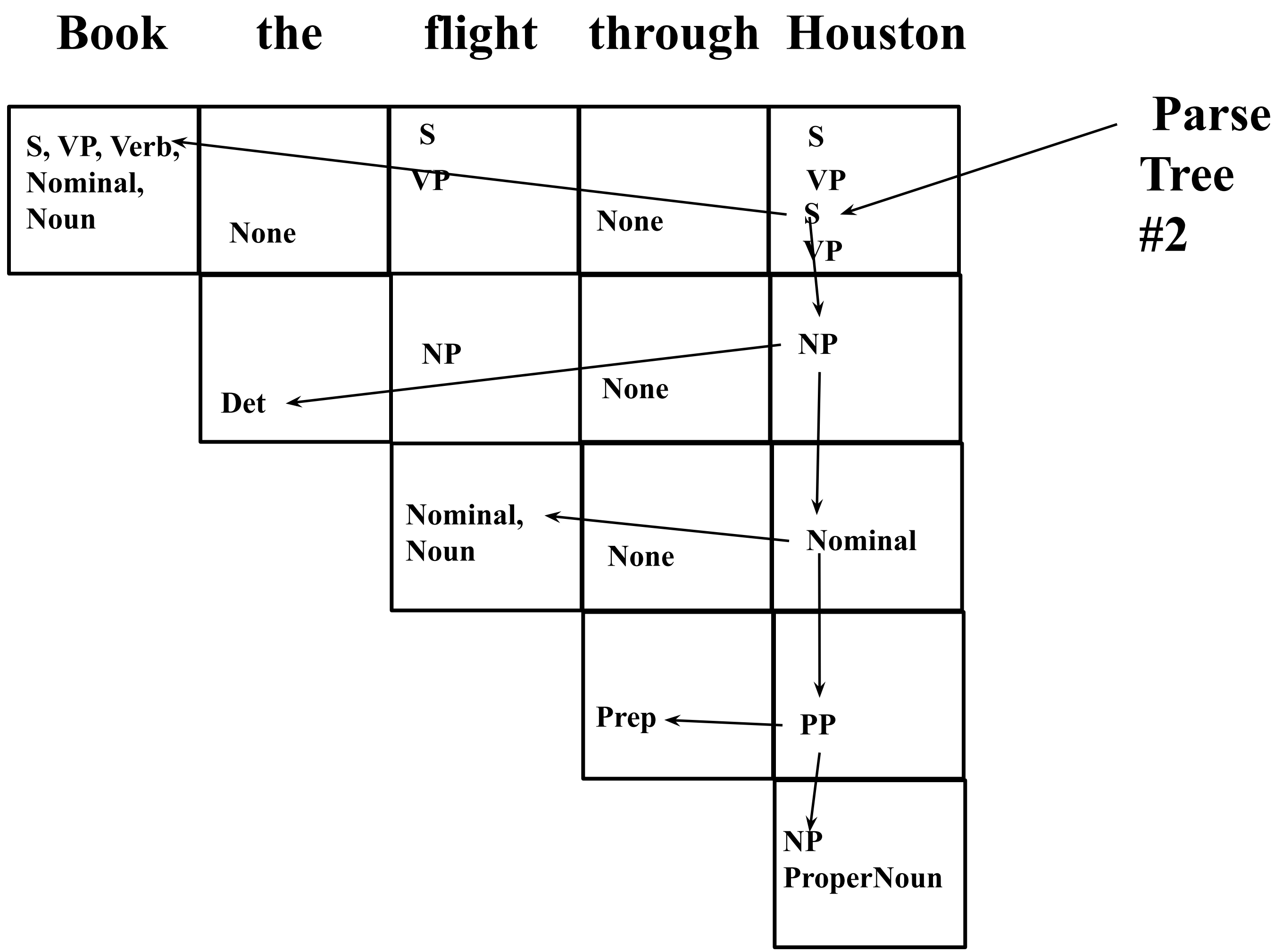
CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	S VP S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
None
S
VP
None
S
VP
S
VP
Parse Tree #1
Det
NP
None
NP
Nominal, Noun
None
Nominal
Prep
PP
NP
ProperNoun

CKY Parser



Book the flight through Houston
S, VP, Verb, Nominal, Noun
None
S
VP
None
S
VP
S
VP
Det
NP
None
NP
Nominal, Noun
None
Nominal
Prep
PP
NP
ProperNoun
Parse Tree #2

Complexity of CKY (recognition)

- There are $(n(n+1)/2) = O(n^2)$ cells
- Filling each cell requires looking at every possible split point between the two non-terminals needed to introduce a new phrase.
- There are $O(n)$ possible split points.
- Total time complexity is $O(n^3)$

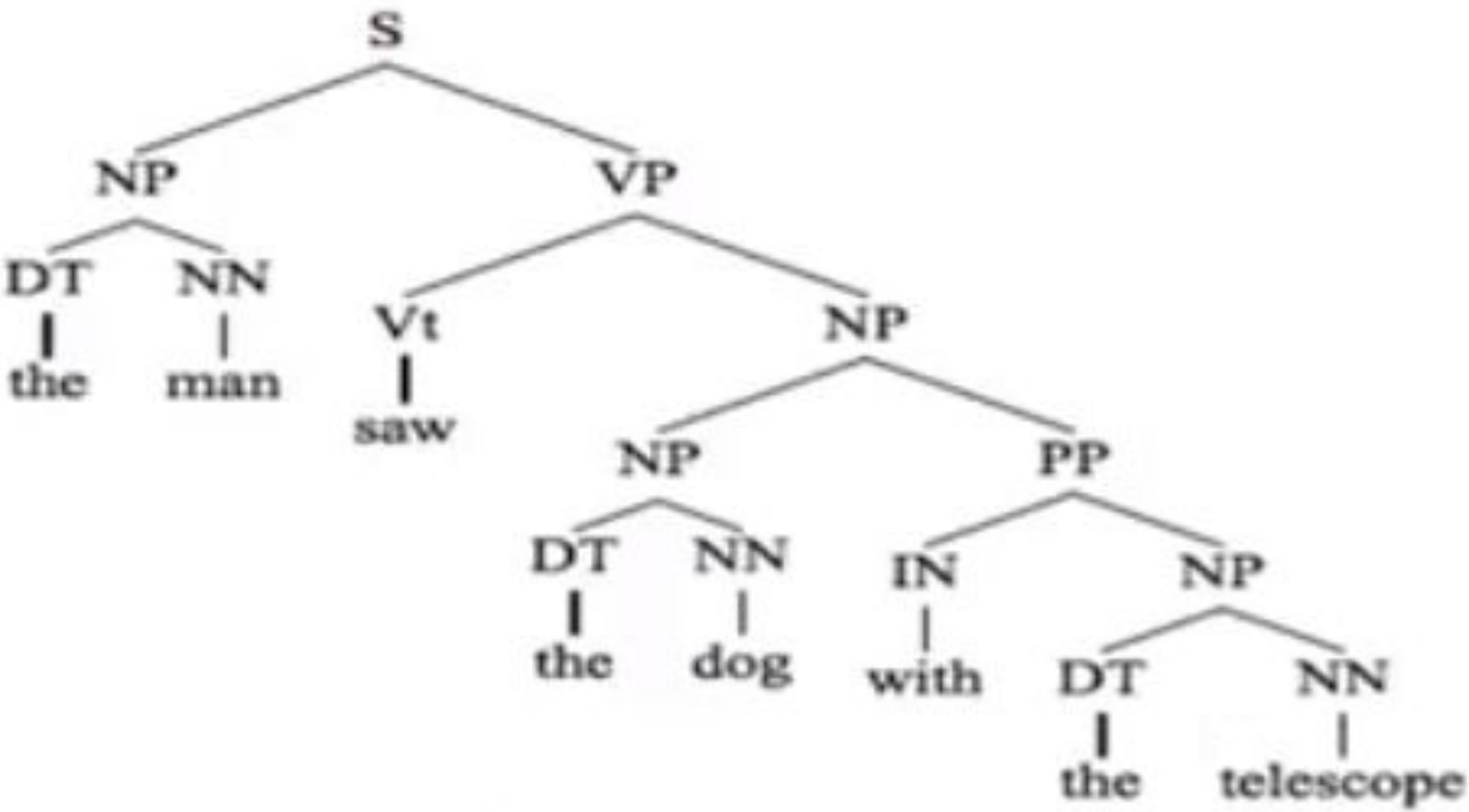
Complexity of CKY (all parses)

- Previous analysis assumes the number of phrase labels in each cell is fixed by the size of the grammar.
- If compute all derivations for each non-terminal, the number of cell entries can expand combinatorially.
- Since the number of parses can be exponential, so is the complexity of finding all parse trees.

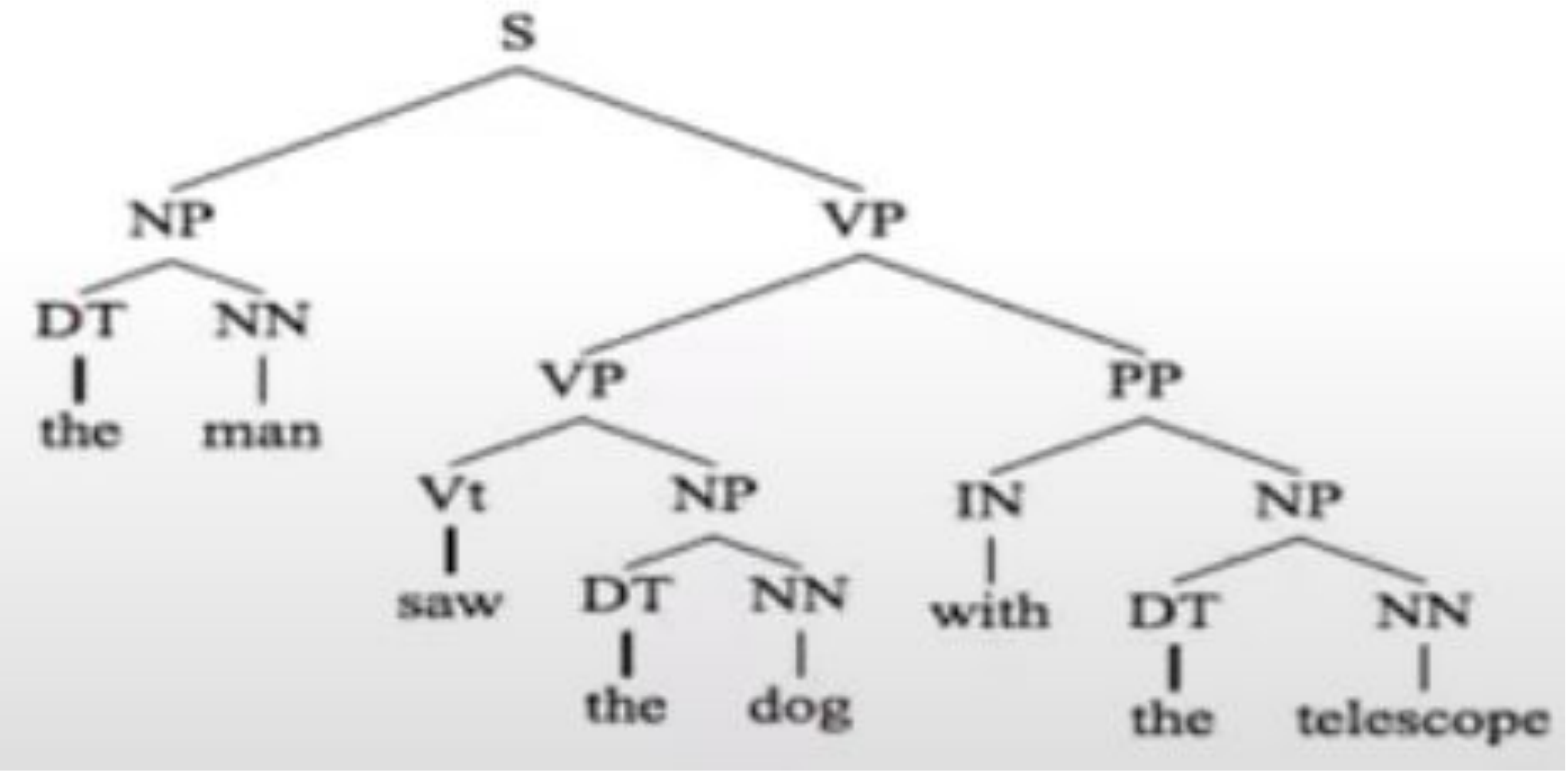
Syntactic Ambiguity

- Just produces all possible parse trees.
- Does not address the important issue of ambiguity resolution.
- CKY does not tell which parse is more probable in case if multiple parse tree is generated through CKY.

Syntactic Ambiguity



Syntactic Ambiguity



Issues with CFGs

- Addressing some grammatical constraints requires complex CFGs that do no compactly encode the given regularities.
- Some aspects of natural language syntax may not be captured at all by CFGs and require context-sensitivity (productions with more than one symbol on the LHS).

Agreement

- Subjects must agree with their verbs on person and number.
  - I am cold.   You are cold. He is cold.
  - * I are cold   * You is cold.   *He am cold.
- Requires separate productions for each combination.
  - S → NP1stPersonSing VP1stPersonSing
  - S → NP2ndPersonSing VP2ndPersonSing
  - NP1stPersonSing → …
  - VP1stPersonSing → …
  - NP2ndPersonSing → …
  - VP2ndPersonSing → …

Probabilistic Context-free grammars

- It’s an extension of simple Context Free Grammars, where in addition of CFG, each rule is also assigned some probability.
- From a given non-terminal on the left hand side, the probability which is generating anything should adapt to one.
- Example, If there are five possibility for NP then probability for all five NP should be one.

Probabilistic Context-free grammars

PCFG: $G = (T, N, S, R, P)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$

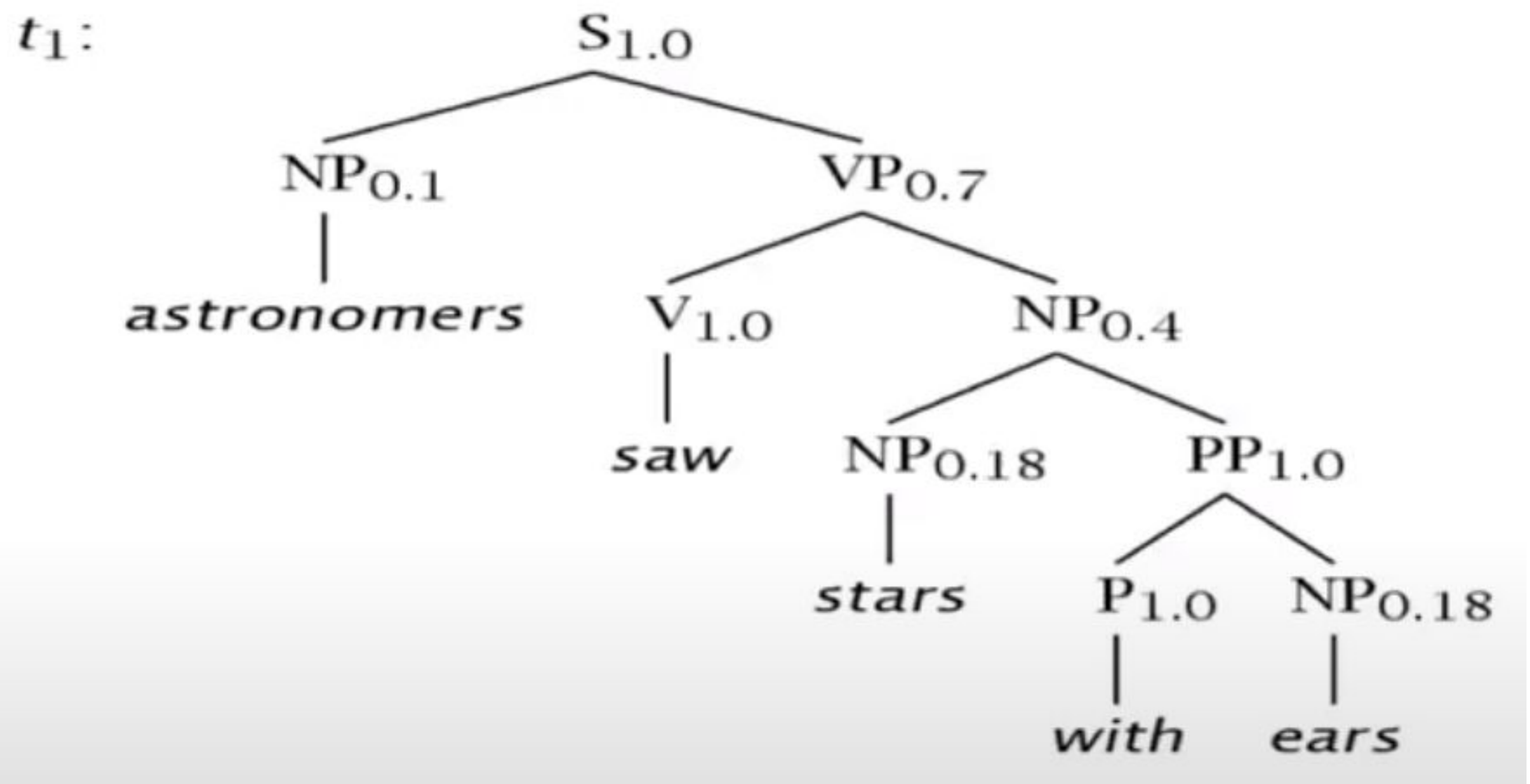
- $P(R)$ gives the probability of each rule.

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

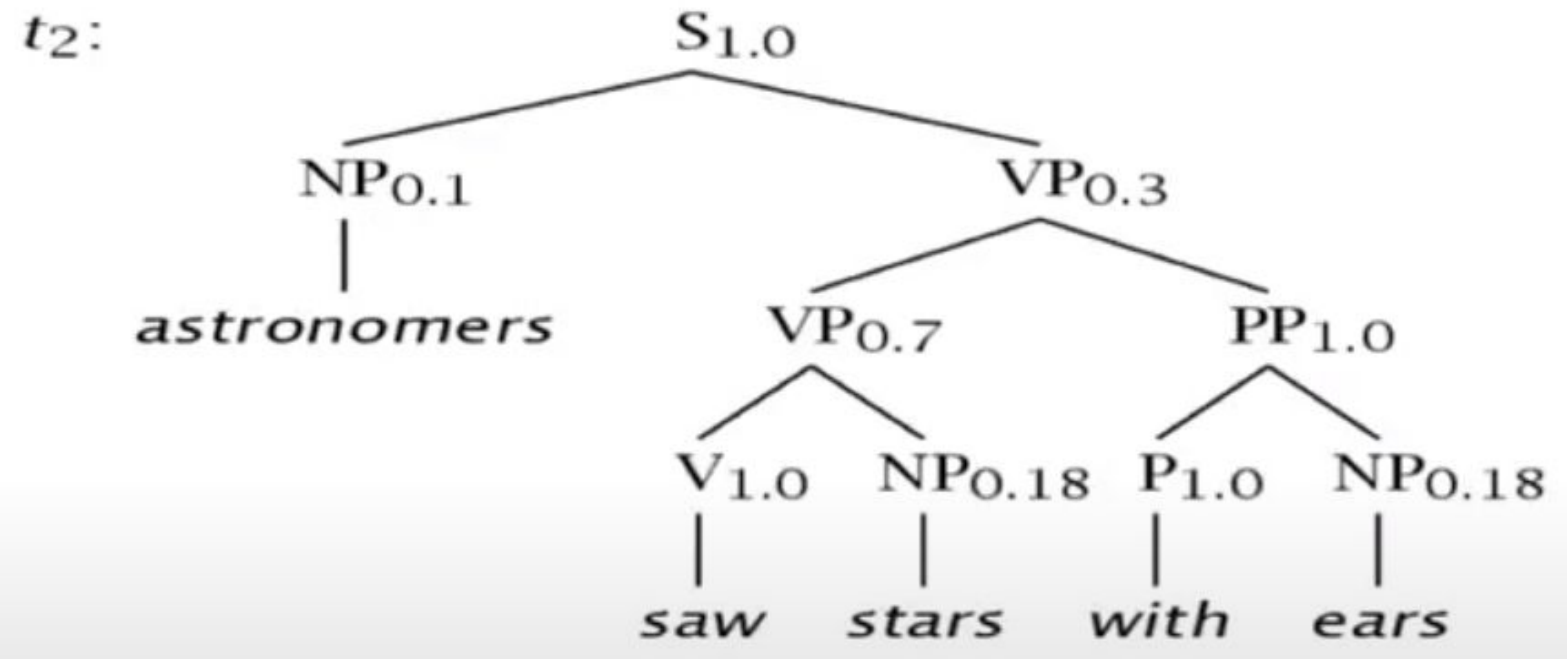
A Simple PCFG (in CNF)

S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	*astronomers*	0.1
VP	→	VP PP	0.3	NP	→	*ears*	0.18
PP	→	P NP	1.0	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescope*	0.1

Example Tree (t1)



Example Tree (t2)

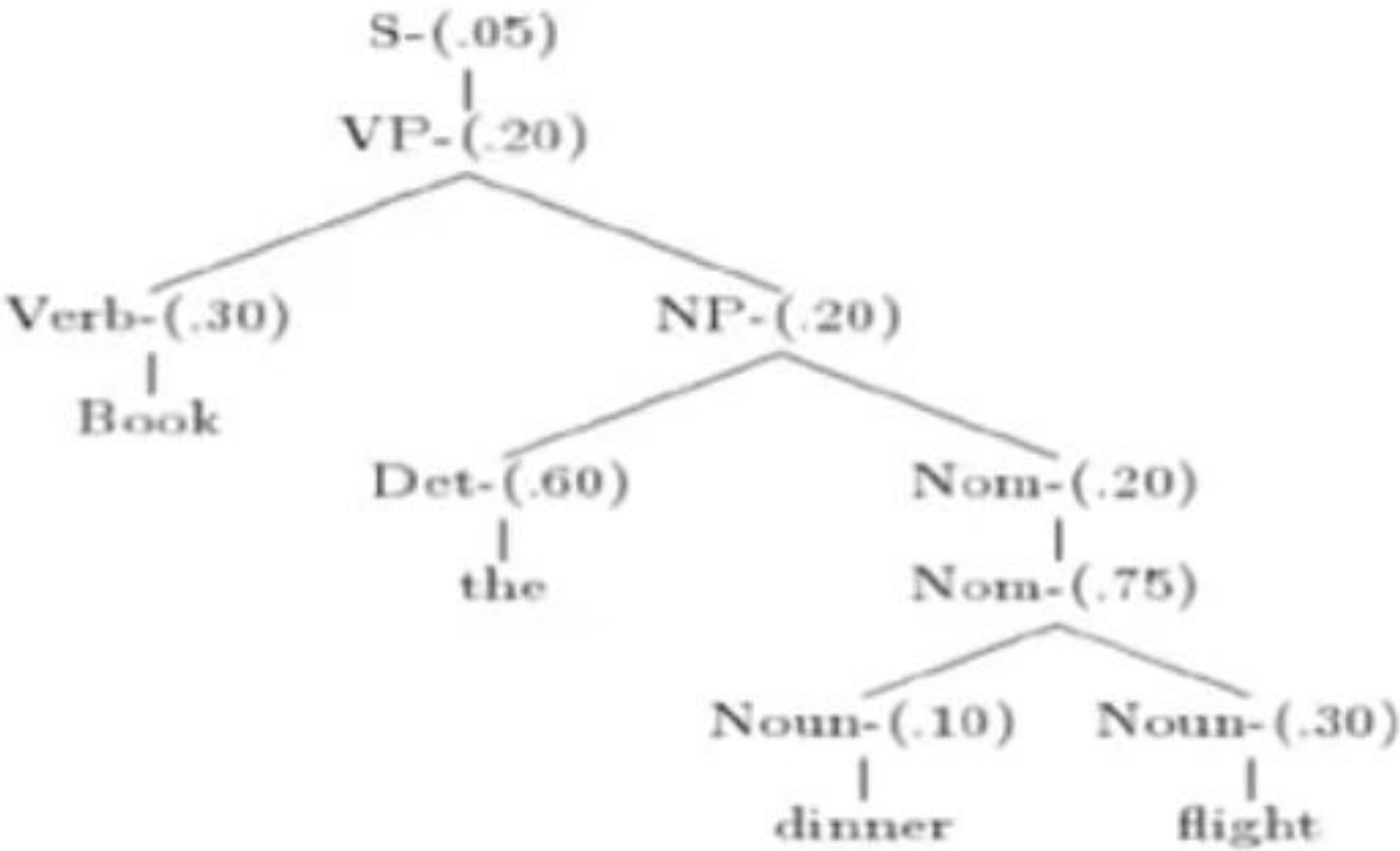


Probability of trees and strings

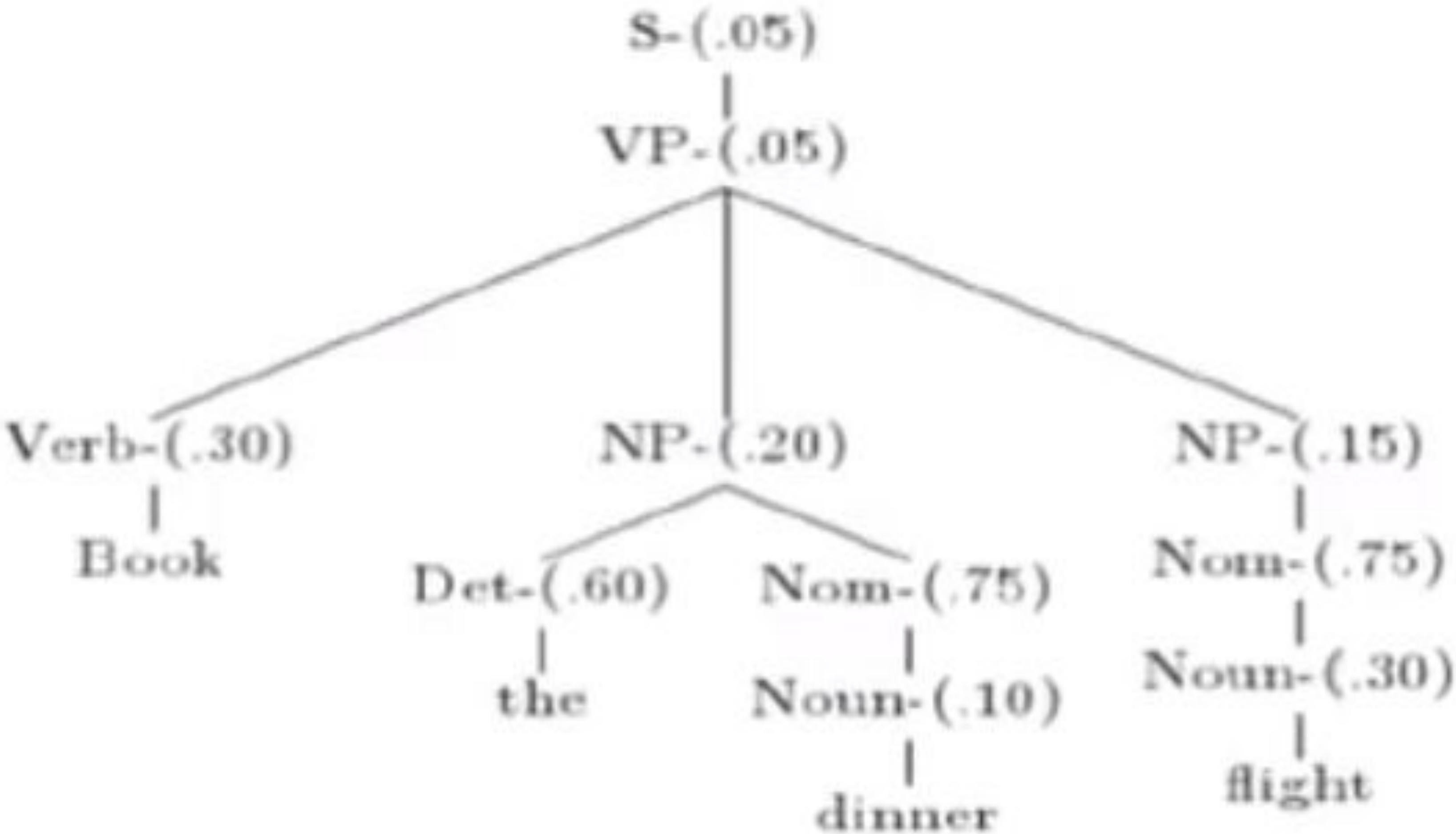
- $P(t)$: The probability of tree is the product of the probabilities of the rules used to generate it
- $P(w_{1n})$: The probability of the string is the sum of the probabilities of the trees which have that string as their yield

$$
\begin{aligned}
w_{15} &= \textit{astronomers saw stars with ears} \\
P(t_1) &= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 \\
&\quad * 1.0 * 1.0 * 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) \\
&= 0.0009072 + 0.0006804 \\
&= 0.0015876
\end{aligned}
$$

Book the dinner flight



Book the dinner flight



Book the dinner flight

Probabilities

- Parse tree 1: $.05 \times .20 \times .30 \times .20 \times .60 \times .20 \times .75 \times .10 \times .30 = 1.62 \times 10^{-6}$
- Parse tree 2: $.05 \times .05 \times .30 \times .20 \times .60 \times .75 \times .10 \times .15 \times .75 \times .30 = 2.28 \times 10^{-7}$

Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse.
- The probability estimates area based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability

Features of PCFGs

- In practice, a PCFG is a worse language model for English than an n-gram model
- All else being equal, the probability of a smaller tree is greater than a larger tree

Currently we can use PCFGs to check, what are all the probabilities for different parse trees.