Parsing

(These slides are modified from Ray Mooney's slides.)

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 for ATIS English

Grammar

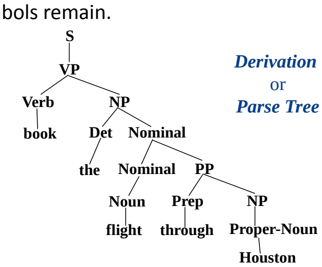
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

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.

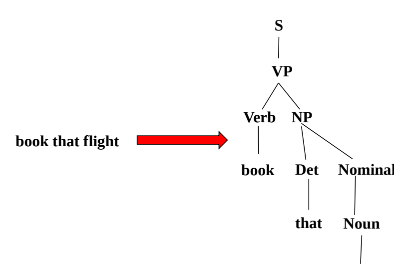


Derivation
or
Parse Tree

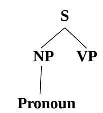
Parsing

- Given a string of non-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.

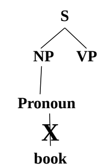
Parsing Example



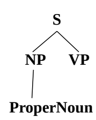
Top Down Parsing



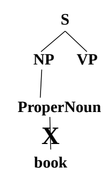
Top Down Parsing



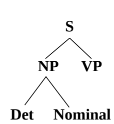
Top Down Parsing



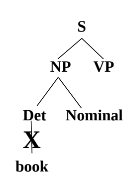
Top Down Parsing



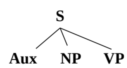
Top Down Parsing



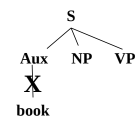
Top Down Parsing



Top Down Parsing

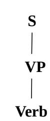


Top Down Parsing

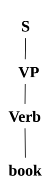


Top Down Parsing

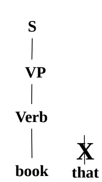
Top Down Parsing



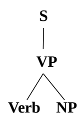
Top Down Parsing



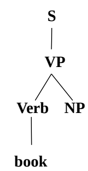
Top Down Parsing



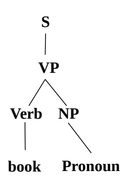
Top Down Parsing



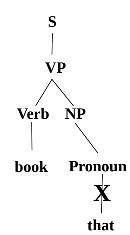
Top Down Parsing



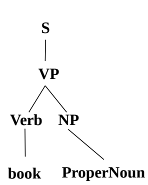
Top Down Parsing



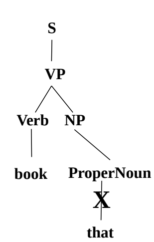
Top Down Parsing



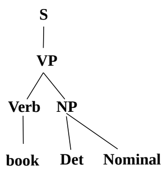
Top Down Parsing



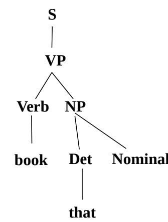
Top Down Parsing



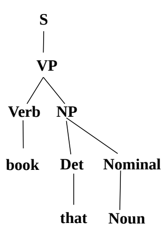
Top Down Parsing



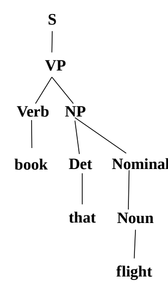
Top Down Parsing



Top Down Parsing



Top Down Parsing

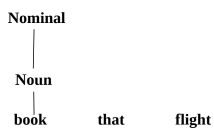


Bottom Up Parsing

book that flight

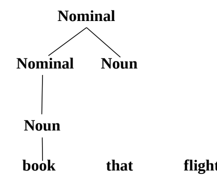
Bottom Up Parsing

Bottom Up Parsing



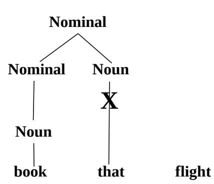
31

Bottom Up Parsing



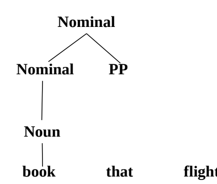
32

Bottom Up Parsing



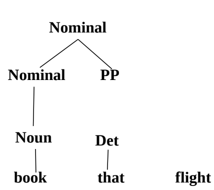
33

Bottom Up Parsing



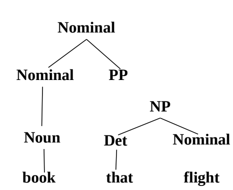
34

Bottom Up Parsing



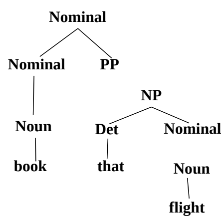
35

Bottom Up Parsing



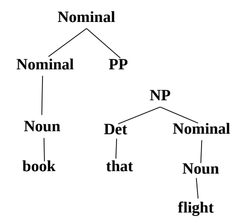
36

Bottom Up Parsing



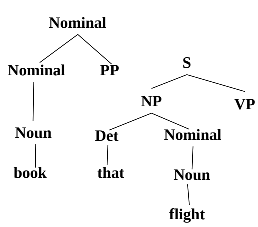
37

Bottom Up Parsing



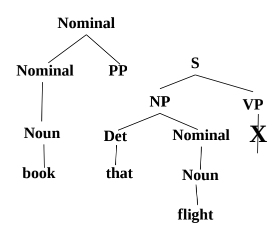
38

Bottom Up Parsing



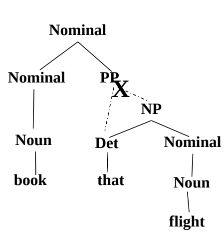
39

Bottom Up Parsing



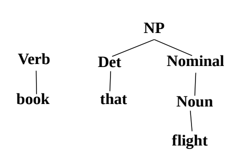
40

Bottom Up Parsing



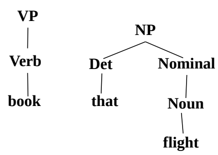
41

Bottom Up Parsing



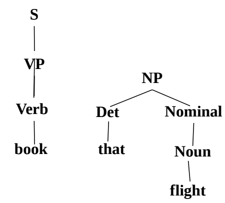
42

Bottom Up Parsing



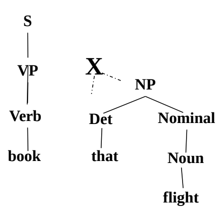
43

Bottom Up Parsing



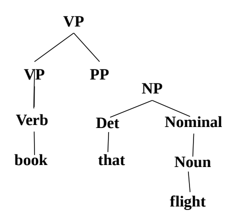
44

Bottom Up Parsing



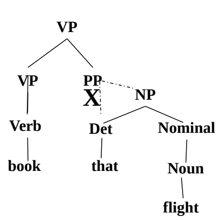
45

Bottom Up Parsing



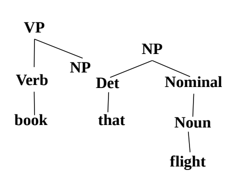
46

Bottom Up Parsing



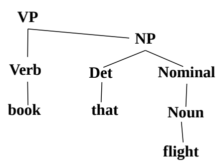
47

Bottom Up Parsing



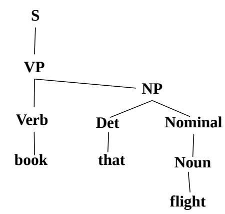
48

Bottom Up Parsing



49

Bottom Up Parsing



50

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.

51

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.

52

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.

53

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).

54

ATIS English Grammar Conversion

Original Grammar	Chomsky Normal Form
S → NP VP	S → NP VP
S → Aux NP VP	S → X1 VP
	X1 → Aux NP
S → VP	S → book | include | prefer
	S → Verb NP
	S → VP PP
NP → Pronoun	NP → I | he | she | me
NP → Proper-Noun	NP → Houston | NWA
NP → Det Nominal	NP → Det Nominal
Nominal → Noun	Nominal → book | flight | meal | money
Nominal → Nominal Noun	Nominal → Nominal Noun
Nominal → Nominal PP	Nominal → Nominal PP
VP → Verb	VP → book | include | prefer
VP → Verb NP	VP → Verb NP
VP → VP PP	VP → VP PP
PP → Prep NP	PP → Prep NP

CKY Parser

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

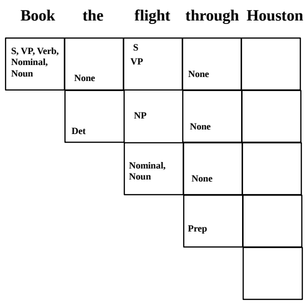
CKY Parser

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

CKY Parser

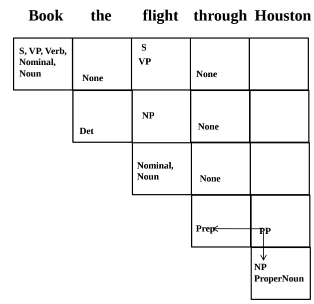
Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP		
	Det	NP		
		Nominal, Noun		

CKY Parser



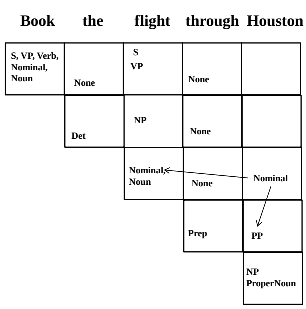
61

CKY Parser



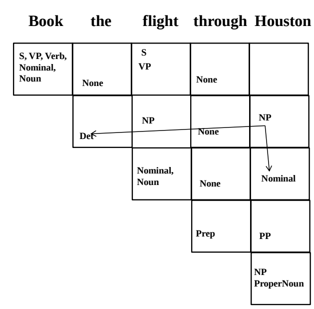
62

CKY Parser



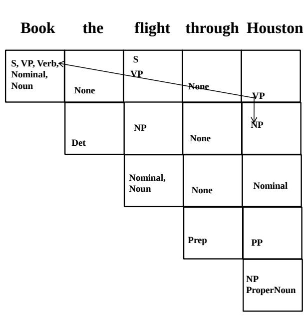
63

CKY Parser



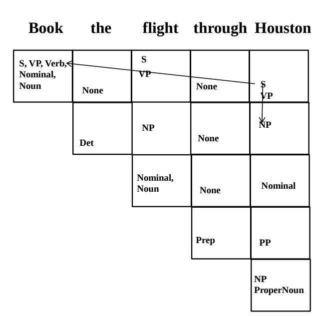
64

CKY Parser



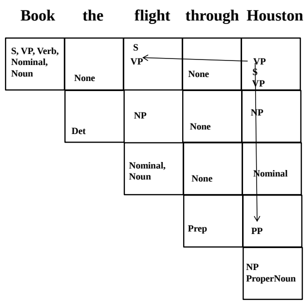
65

CKY Parser

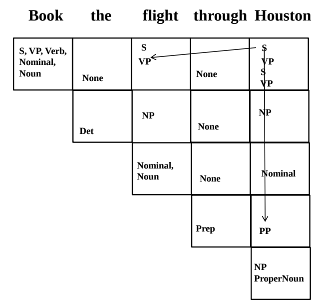


66

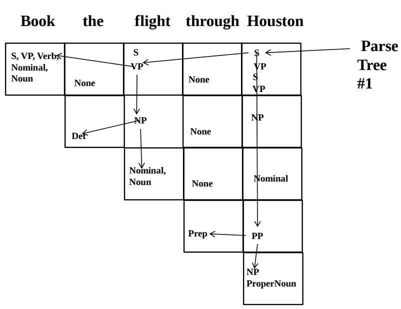
CKY Parser



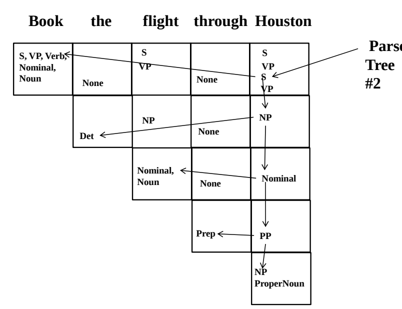
CKY Parser



CKY Parser



CKY Parser



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.

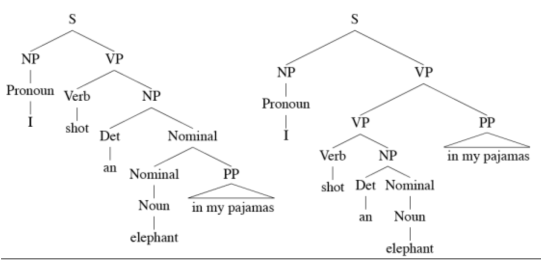
Effect of CNF on Parse Trees

- Parse trees are for CNF grammar not the original grammar.
- A post-process can repair the parse tree to return a parse tree for the original grammar.

Syntactic Ambiguity

- Just produces all possible parse trees.
- Does not address the important issue of ambiguity resolution.

Ambiguity



Examples taken from Jurafsky and Martin

Shallow Parsing (=Phrase Chunking)

(These slides are modified from Ray Mooney's slides.)

Shallow Parsing (also known as **Phrase Chunking**)

- Find all non-recursive noun phrases (NPs) and verb phrases (VPs) in a sentence.
  - [NP I] [VP ate] [NP the spaghetti] [PP with] [NP meatballs].
  - [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September]

Phrase Chunking as **Sequence Tagging!**

- Tag individual words with one of 3 tags
  - B (Begin) word starts new target phrase
  - I (Inside) word is part of target phrase but not the first word
  - O (Outside) word is not part of target phrase
- Sample for NP chunking
  - He reckons the current account deficit will narrow to only # 1.8 billion in September.

Begin **Inside** **Outside**

Evaluating Chunking

- Per token accuracy does not evaluate finding correct full chunks. Instead use:

$$\text{Precision} = \frac{\text{Number of correct chunks found}}{\text{Total number of chunks found}}$$

$$\text{Recall} = \frac{\text{Number of correct chunks found}}{\text{Total number of actual chunks}}$$

- Take harmonic mean to produce a single evaluation metric called F measure.

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \qquad F_1 = \frac{1}{(\frac{1}{P} + \frac{1}{R})/2} = \frac{2PR}{P + R}$$

79

Current Chunking Results

- Best system for NP chunking: F_1=96%
- Typical results for finding range of chunk types (CONLL 2000 shared task: NP, VP, PP, ADV, SBAR, ADJP) is F_1=92–94%

80