CS60075
Natural Language Processing
Autumn 2020

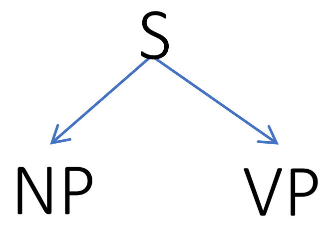
Module 5: Part B
L0 grammar and Phrase Structure Parsing

Sep 30 2020

Formal Grammars of English

Context-free grammars (CFGs)

- Consist of
  - Rules
  - Terminals
  - Non-terminals
  - Start Symbol



- Specifies a set of tree structures that capture constituency and ordering in language

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

Productions of CFG

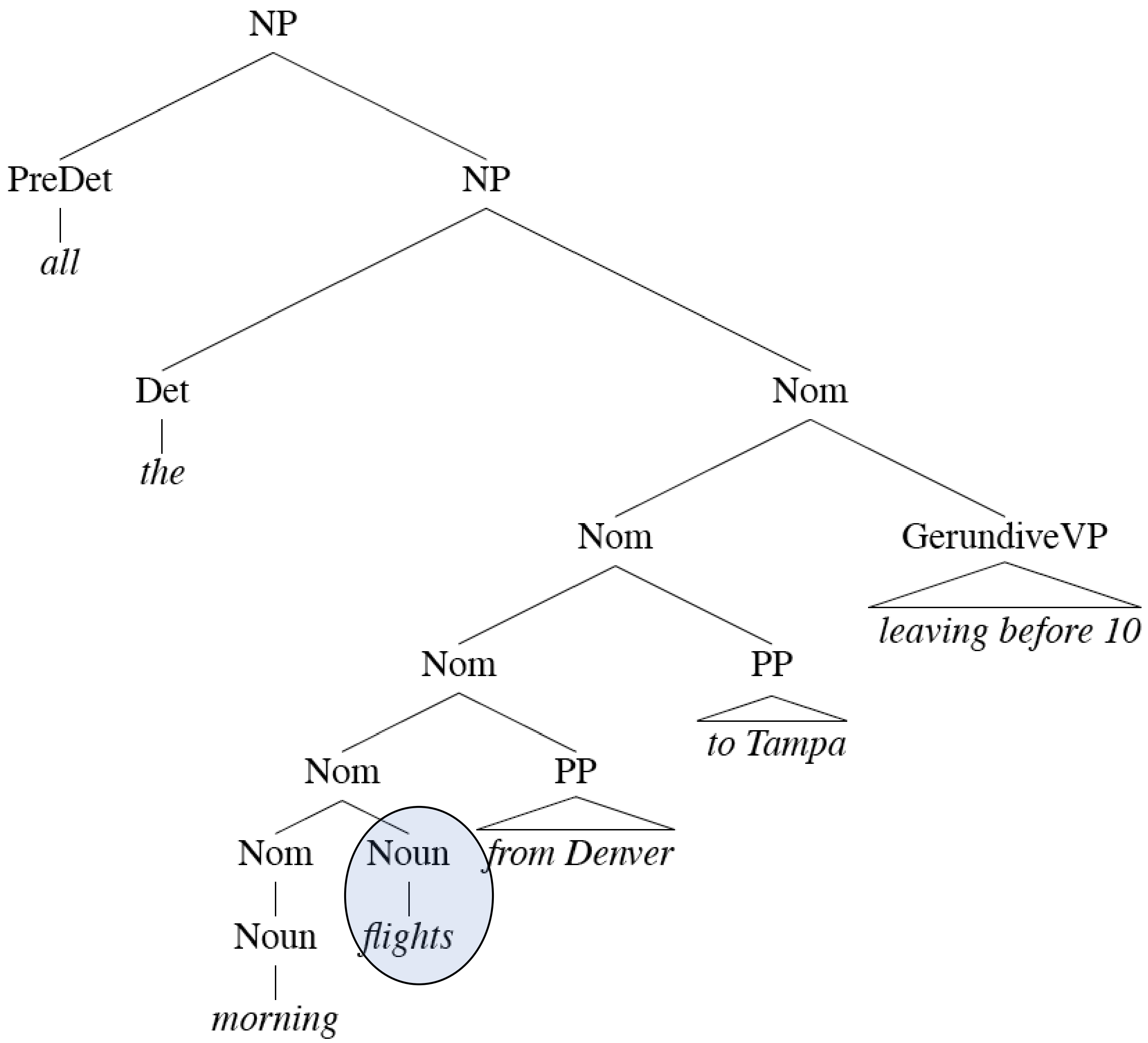
- A CFG can be thought of in two ways:
  - a device for generating sentences

    (Derivation)
  - a device for assigning a structure to a given sentence.

- Some rules for noun phrases:

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

Noun Phrases



Nominals

- Contain the head and any pre- and post- modifiers of the head.
  - Pre-
    - Quantifiers, cardinals, ordinals...
      - *Three cars*
    - Adjectives
      - *large cars*

Postmodifiers

- Three kinds
  - Prepositional phrases
    - *From Seattle*
  - Non-finite clauses
    - *Arriving before noon*
  - Relative clauses
    - *That serve breakfast*
- Same general (recursive) rules to handle these
  - *Nominal → Nominal PP*
  - *Nominal → Nominal GerundVP*
  - *Nominal → Nominal RelClause*

HINDI

Nominal → PP Nominal

Nominal → Nominal GerundVP

Nominal → Nominal RelClause

Verb Phrases

- English *VP*s consist of a verb (the head) along with 0 or more *following* constituents which we'll call *arguments*.

VP	$\rightarrow$	$Verb$	disappear
VP	$\rightarrow$	$Verb\ NP$	prefer a morning flight
VP	$\rightarrow$	$Verb\ NP\ PP$	leave Boston in the morning
VP	$\rightarrow$	$Verb\ PP$	leaving on Thursday

Subcategorization

- Even though there are many valid VP rules in English, not all verbs are allowed to participate in all those VP rules.
- We can *subcategorize* the verbs in a language according to the sets of VP rules that they participate in.
- This is just an elaboration on the traditional notion of transitive/intransitive.
- Modern grammars have many such classes

Subcategorization

- Sneeze: John sneezed
- Find: Please find [a flight to NY]$_{NP}$
- Give: Give [me]$_{NP}$[a cheaper fare]$_{NP}$
- Help: Can you help [me]$_{NP}$[with a flight]$_{PP}$
- Prefer: I prefer [to leave earlier]$_{TO\text{-}VP}$
- Told: I was told [United has a flight]$_{S}$
- …

Generative Grammar

- The use of formal languages to model Generative natural languages is called ***generative grammar*** since the language is defined by the set of possible sentences "generated" by the grammar.
- You can view these rules as either analysis or synthesis engines
  - Generate strings in the language
  - Reject strings not in the language
  - Assign structures (trees) to strings in the language

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

Sentence Types

- Declaratives: *A plane left.*

  S $\rightarrow$ NP VP

- Imperatives: *Leave!*

  S $\rightarrow$ VP

- Yes-No Questions: *Did the plane leave?*

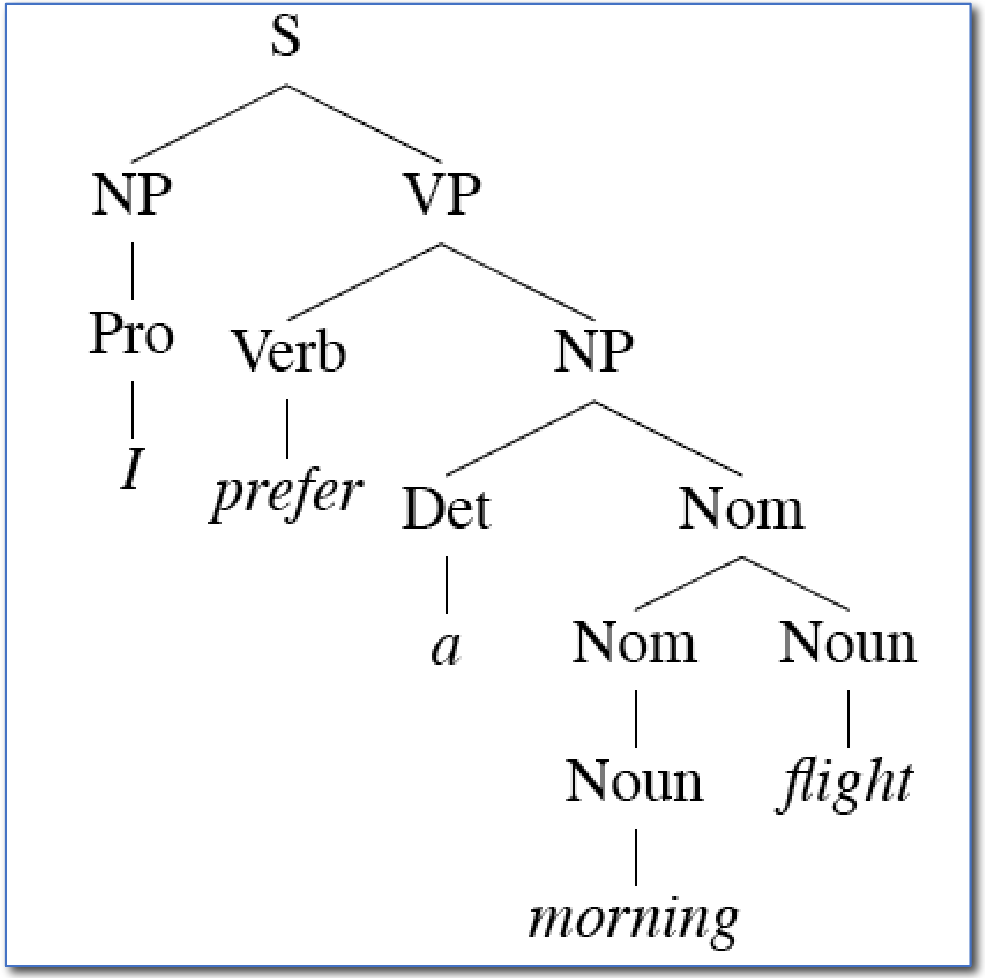
  S $\rightarrow$ Aux NP VP

- WH Questions: *When did the plane leave?*

  S $\rightarrow$ WH-NP Aux NP VP

Derivations

- A *derivation* is a sequence of rules applied to a string that *accounts* for that string
  - Covers all the elements in the string
  - Covers only the elements in the string



Parsing

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

Treebank

- A syntactically annotated corpus where every sentence is paired with a corresponding tree.
- The Penn Treebank project
  - treebanks from the Brown, Switchboard, ATIS, and Wall Street Journal corpora of English
  - treebanks in Arabic and Chinese.
- Others
  - the Prague Dependency Treebank for Czech,
  - the Negra treebank for German, and
  - the Susanne treebank for English
  - Universal Dependencies Treebank

Penn Treebank

- Penn TreeBank is a widely used treebank.

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Most well known part is the Wall Street Journal section of the Penn TreeBank.

- 1 M words from the 1987-1989 Wall Street Journal.

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Figure 11.7 Parsed sentences from the LDC Treebank3 version of the Brown (a) and ATIS (b) corpora.

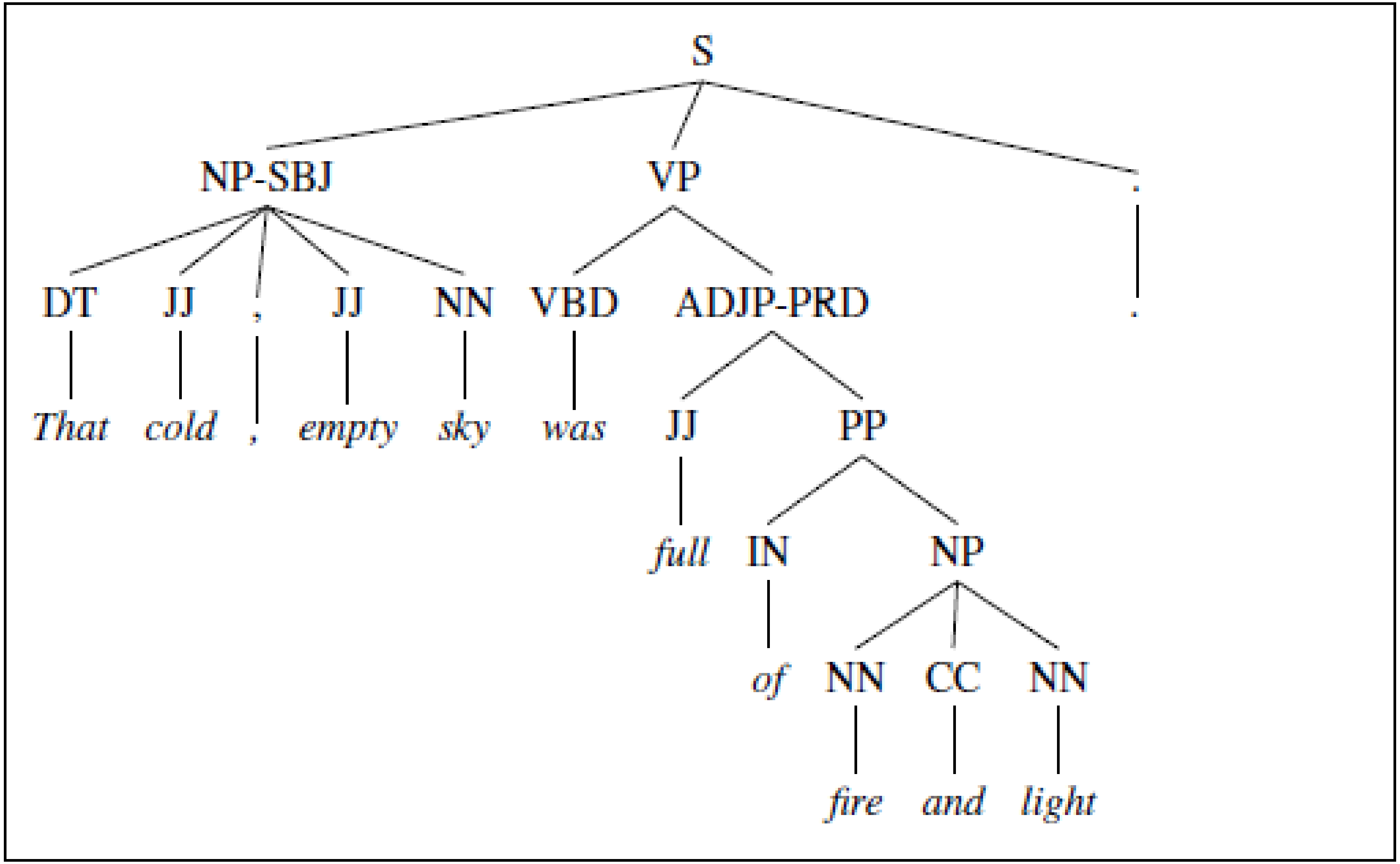


Figure 11.8 The tree corresponding to the Brown corpus sentence in the previous figure.

Treebanks as Grammars

- The sentences in a treebank implicitly constitute a grammar of the language represented by the corpus being annotated.

- Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar
  - The WSJ section gives us about 12k rules

Parsing

- Parsing with CFGs refers to the task of assigning proper trees to input strings
- Proper here means a tree that covers all and only the elements of the input and has an S at the top
- It doesn't mean that the system can select the correct tree from among all the possible trees

Syntactic Analysis (Parsing)

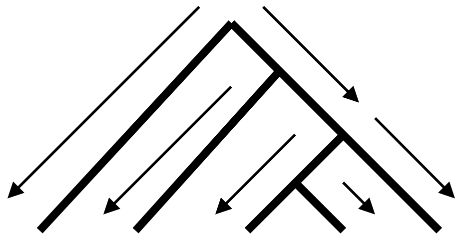
- Automatic methods of finding the syntactic structure for a sentence
  - Symbolic methods: a phrase grammar or another description of the structure of language is required.
    The chart parser.
  - Statistical methods: a text corpus with syntactic structures is needed (a treebank)

Search Framework

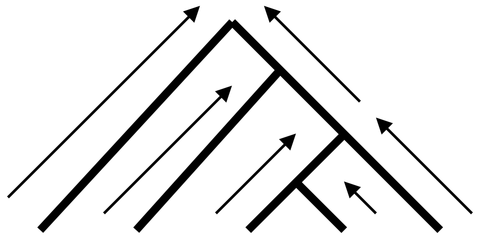
- Think about parsing as a form of search…
  - A search through the space of possible trees given an input sentence and grammar

How to parse

Top-down: Start at the top of the tree with an S node, and work your way down to the words.



Bottom-up: Look for small pieces that you know how to assemble, and work your way up to larger pieces.



Summary

- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.
- But there are problems
  - That can be dealt with adequately, although not elegantly, by staying within the CFG framework.
- There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)
  - LFG, HPSG, Construction grammar, XTAG, etc.

Top-Down Search

- Since we're trying to find trees rooted with an *S* (Sentences)
  - Start with the rules that give us an *S*.
  - Then we can work our way down from there to the words.

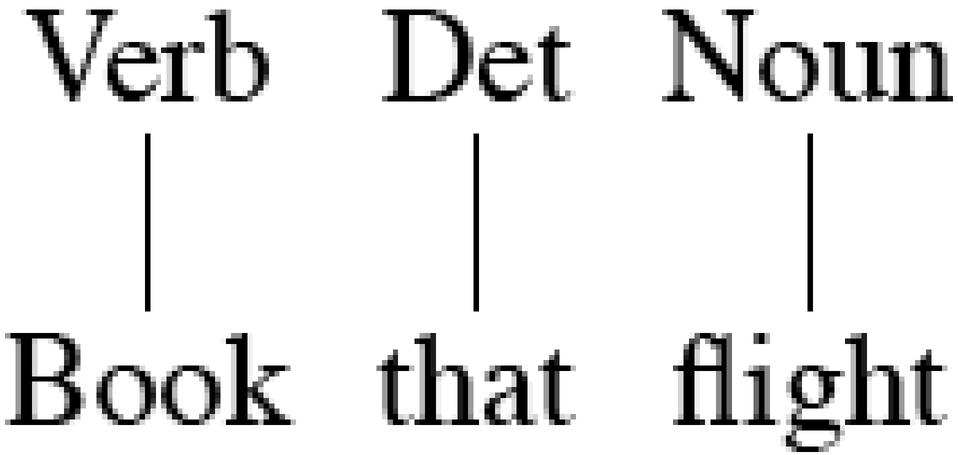
Bottom-Up Parsing

- Of course, we also want trees that cover the input words. So we might also start with trees that link up with the words in the right way.
- Then work your way up from there to larger and larger trees.

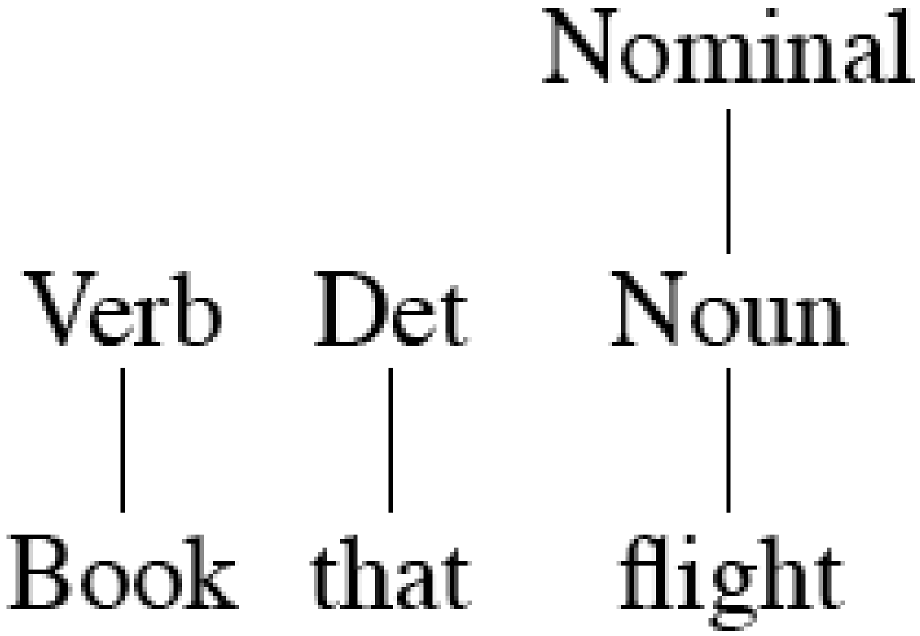
Bottom-Up Search

Book that flight

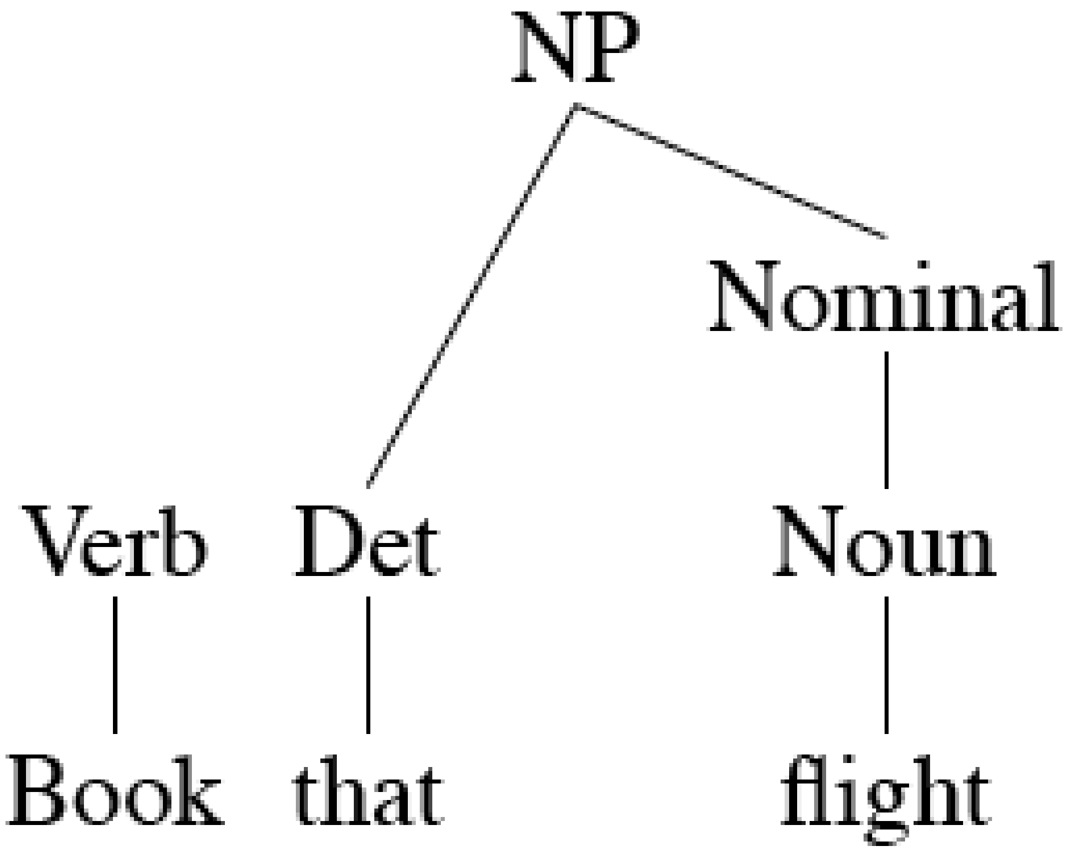
Bottom-Up Search



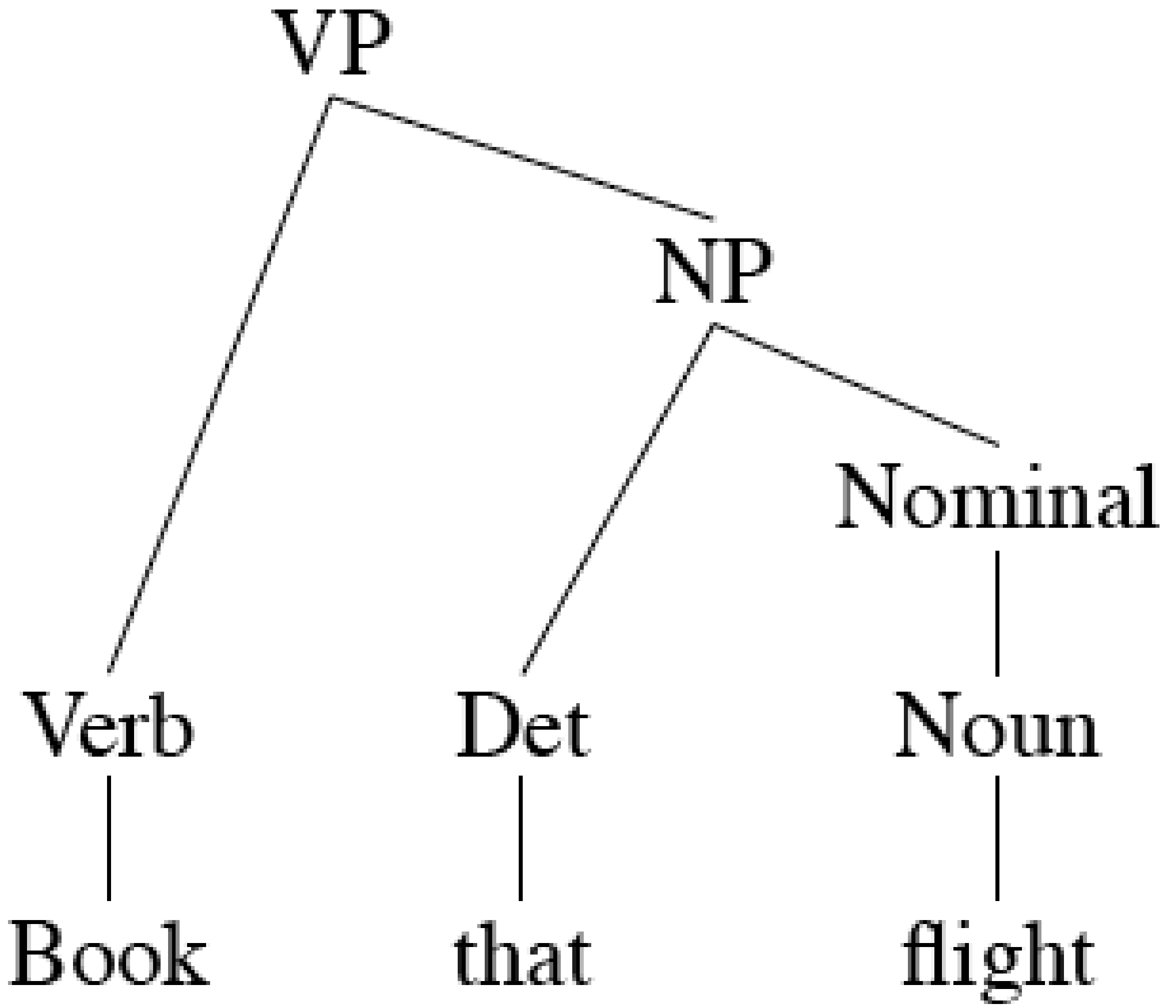
Bottom-Up Search



Bottom-Up Search



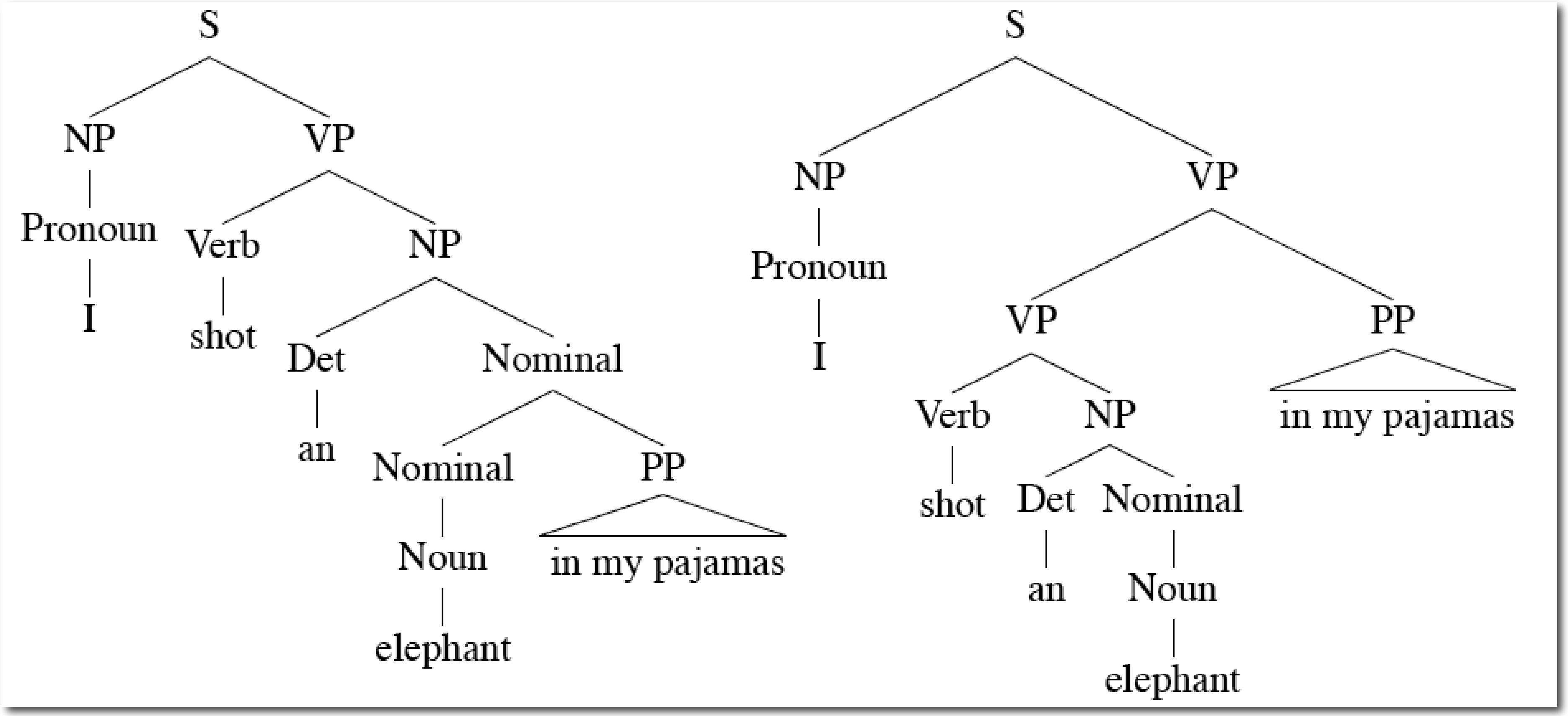
Bottom-Up Search



Issues

- Ambiguity
- Shared subproblems

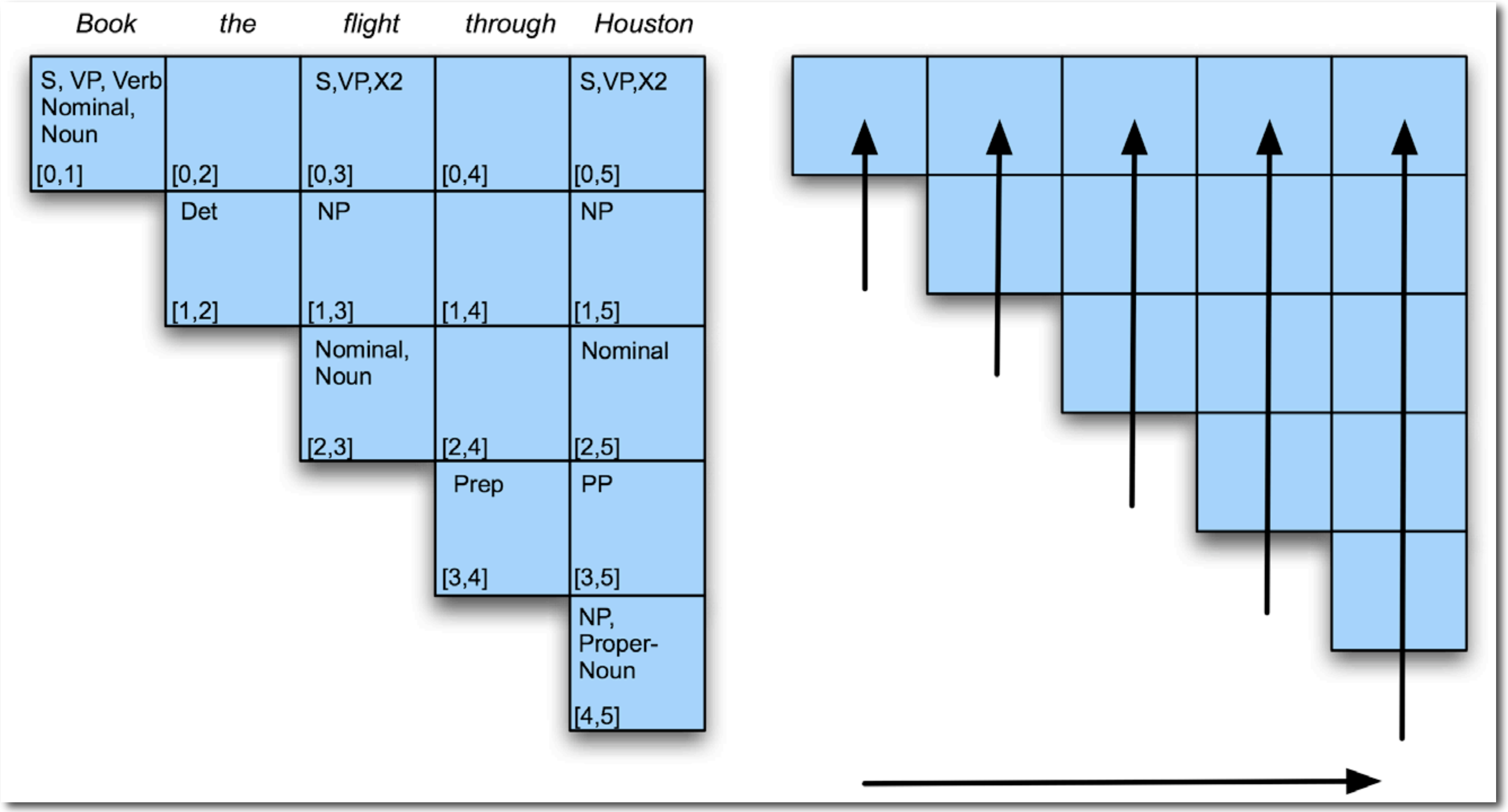
Ambiguity


S
NP
VP
Pronoun
Verb
NP
I
shot
Det
Nominal
an
Nominal
PP
Noun
in my pajamas
elephant
S
NP
VP
Pronoun
VP
PP
I
Verb
NP
in my pajamas
shot
Det
Nominal
an
Noun
elephant

Dynamic Programming

- DP search methods fill tables with partial results and thereby
  - Avoid doing avoidable repeated work
  - Solve exponential problems in polynomial time (ok, not really)
  - Efficiently store ambiguous structures with shared sub-parts.
- We’ll cover one approach that corresponds to a bottom-up strategy
  - CKY

CKY Algorithm



CKY Algorithm

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

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

$table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in gram$

for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**

for $k \leftarrow i+1$ **to** $j-1$ **do**

$table[i,j] \leftarrow table[i,j] \cup$

$\{A \mid A \rightarrow BC \in grammar,$

$B \in table[i,k],$

$C \in table[k,j]\}$

Looping over the columns

Filling the bottom cell

Filling row i in column j

Looping over the possible split locations between i and j.

Check the grammar for rules that link the constituents in [i,k] with those in [k,j]. For each rule found store the LHS of the rule in cell [i,j].