CSCI 3907/6907

Fall 2019

Lecture 6

Formal Grammars & Syntactic Parsing

Notes

- The due date for 2nd homework is Oct 8
- You should have already formed a group **only** for your **final project**

Syntax and Grammar

- **Syntax**: The way words are arranged in a sentence

- Some syntactic notions we've seen:
  - Regular languages
  - N-gram language models
  - Parts-of-speech tagging

Syntactic Parsing

- Analyzing a sentence to identify certain syntactic features
  - Useful for many applications such as grammar checking

- An important task in NLP to meditate between *surface forms* and *meaning*
    - Relation extraction
    - Semantic role labeling
    - Paraphrase detection

Constituency in English

- A **constituent** is a group of words acting as a single unit or phrase
  - e.g. Noun Phrases *(Harry, Harry the Horse, Three parties from Brooklyn, They, A high-class spot such as Mindy's)*
- Constituents appear in similar syntactic contexts
  - For example, noun phrases can be followed by verbs:
    - Three parties from Brooklyn *arrive* …
    - They *arrive* …
    - A high-class spot such as Mindy's *attracts* …
- The locations the constituents can be placed (preposed or postposed constructions)
    - *On September seventeenth*, I'd like to fly from Atlanta to Denver
    - I'd like to fly *on September seventeenth* from Atlanta to Denver
    - I'd like to fly from Atlanta to Denver *on September seventeenth*

Context-Free Grammars (CFG)

- Constituency in English can be modeled using **Context-Free Grammar** (CFG), also called **Phrase-Structure** Grammar

- Consists of a set of **rules** (productions) and a **lexicon** (words and symbols)
  - Rules express the ways that symbols of the language can be grouped and ordered together

CFG - Example

- Productions for a noun phrase (NP):

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

- Symbols in CFG
  - Terminal symbols → corresponds to words in the language (e.g. flight)
  - Non terminal symbols → clusters or generalizations of these terminals

CFG – Example – Cont.

- Productions for a noun phrase (NP):

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

- The left of the arrow is a single **non-terminal** symbol
- The right of the arrow is an ordered set of non-terminals and **terminals** (words from the lexicon)

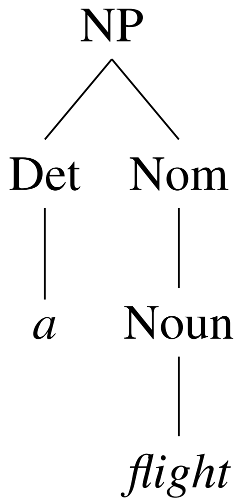
$$\text{Parts-of-speech}\left\{\begin{array}{rcl} Det & \rightarrow & a \\ Det & \rightarrow & the \\ Noun & \rightarrow & flight \end{array}\right\}\text{terminals}$$

CFG as a Generator - Example

– rewrite the symbol on the left with the string of symbols on the right
– Starting with NP :

Derivation
of "a flight" from NP

$NP \rightarrow Det\ Nominal$
Det Nominal
$Nominal \rightarrow Noun$
Det Noun
$Det \rightarrow a$
a Noun
$Noun \rightarrow flight$
a flight



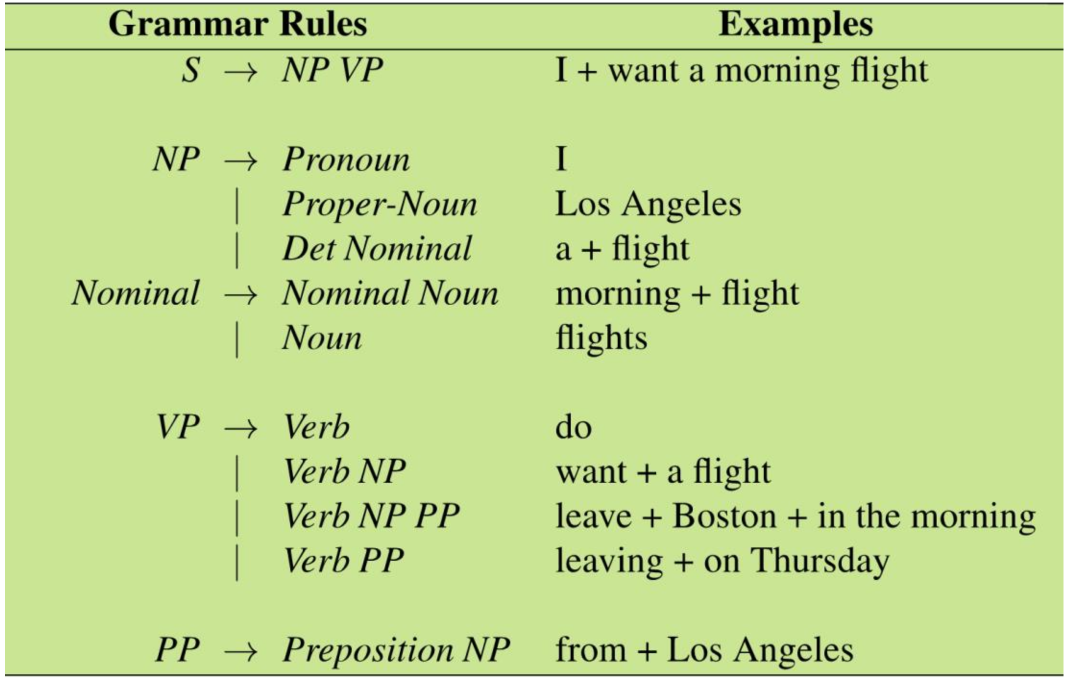
Parse Tree

Formal Language

- The formal language defined by a CFG is the set of all strings (of terminals) that can be derived from the **start symbol**.
  - A grammar must have a designated start symbol, typically called **S**.
  - Rules must be defined to expand the start symbol.

$$S \rightarrow NP\ VP$$

Example: CFG Grammar

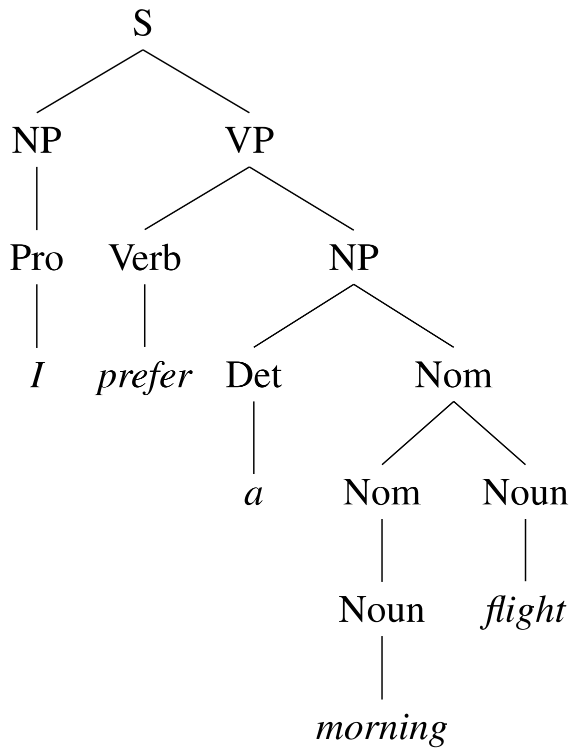


Grammar Rules			Examples
S	$\rightarrow$	*NP VP*	I + want a morning flight
NP	$\rightarrow$	*Pronoun*	I
	$\mid$	*Proper-Noun*	Los Angeles
	$\mid$	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	$\mid$	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	$\mid$	*Verb NP*	want + a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

Example: Lexicon

$$
\begin{array}{rcl}
\mathit{Noun} & \rightarrow & \mathit{flights} \mid \mathit{breeze} \mid \mathit{trip} \mid \mathit{morning} \\
\mathit{Verb} & \rightarrow & \mathit{is} \mid \mathit{prefer} \mid \mathit{like} \mid \mathit{need} \mid \mathit{want} \mid \mathit{fly} \\
\mathit{Adjective} & \rightarrow & \mathit{cheapest} \mid \mathit{non\text{-}stop} \mid \mathit{first} \mid \mathit{latest} \\
 & & \mid \mathit{other} \mid \mathit{direct} \\
\mathit{Pronoun} & \rightarrow & \mathit{me} \mid \mathit{I} \mid \mathit{you} \mid \mathit{it} \\
\mathit{Proper\text{-}Noun} & \rightarrow & \mathit{Alaska} \mid \mathit{Baltimore} \mid \mathit{Los\ Angeles} \\
 & & \mid \mathit{Chicago} \mid \mathit{United} \mid \mathit{American} \\
\mathit{Determiner} & \rightarrow & \mathit{the} \mid \mathit{a} \mid \mathit{an} \mid \mathit{this} \mid \mathit{these} \mid \mathit{that} \\
\mathit{Preposition} & \rightarrow & \mathit{from} \mid \mathit{to} \mid \mathit{on} \mid \mathit{near} \\
\mathit{Conjunction} & \rightarrow & \mathit{and} \mid \mathit{or} \mid \mathit{but}
\end{array}
$$

Bracketed Notation



- Sometimes it's more convenient to represent a parse tree in a compact form called **bracketed notation**:

$$[_{S}\ [_{NP}\ [_{Pro}\ \text{I}]]\ \boxed{[_{VP}\ [_{V}\ \text{prefer}]\ [_{NP}\ [_{Det}\ \text{a}]\ \boxed{[_{Nom}\ [_{N}\ \text{morning}]\ [_{Nom}\ [_{N}\ \text{flight}]]]}]]}]$$

Formal Definition of a CFG

- G = (T, N, S, R)
  - T is a set of **terminal** symbols
  - N is a set of **nonterminal** symbols
  - S is the **start symbol** ($S \in N$)
  - R is a set of **rules/productions** of the form $X \rightarrow \gamma$
    - $X \in N$ and $\gamma \in (N \cup T)^*$

- A grammar G generates a language L

Grammar Rules for English

Some Grammar Rules for English

- A subset of grammar rules and main constituents for English:
  - (1) Sentence constructions
  - (2) The Noun Phrase (NP)
  - (3) The Verb Phrase (VP)
  - (4) Coordination

(1) Sentence Constructions

- **Declarative** structure: a subject noun phrase followed by a verb phrase.

$$S \rightarrow NP\ VP$$

  - I want a flight from Ontario to Chicago
  - The flight should be 11 a.m. tomorrow

- **Imperative** structure: verb phrase, with no subject.

$$S \rightarrow VP$$

  - Show the lowest fare
  - List all flights from 5 to 7 p.m

(1) Sentence Constructions

- **yes-no question** structure: an auxiliary verb followed by a noun phrase, followed by a verb phrase.

$$S \rightarrow Aux\ NP\ VP$$

  - Does any of these flights have stops?
  - Can you give me the same information for United?

- **wh-subject-question** structure: similar to declarative structure, but the noun phrase starts with a wh- word.

$$S \rightarrow Wh\text{-}NP\ VP$$

  - Which airlines fly from Ontario to Chicago?

(1) Sentence Constructions

- **wh-non-subject-question** structure: similar to yes-no structure, but starts with a wh- NP.

$$S \rightarrow \textit{Wh-NP Aux NP VP}$$

  - What flights do you have from Ontario to Chicago?

(2) The Noun Phrase

- NP $\longrightarrow$ Pronoun | ProperNoun | **Det Nominal**
- **Determiners**: Can be
  - a simple lexical determiner (a, an, the)
    - A stop
    - The flights
    - Those flights
  - or a more complex expression:
    - United's flight
    - United's pilots' union
    - Denver's mayor's mother's canceled flight

$$Det \rightarrow NP\ 's$$

(2) The Noun Phrase

- The **nominal** follows the determiner, and can contain pre- and post- head noun modifiers.

$$Nominal \rightarrow Noun$$

- Pre-modifiers: before the head noun, we can have : **cardinal** numbers (one ,two, ..), **ordinal** numbers (first, second, ..), **quantifiers** (some, all, …) , and/or **adjectives**.
  – We can also define Adjective Phrases (AP), which may include adverbs before the adjective
    - A *non-stop* flight
    - The *longest* layover

(2) The Noun Phrase

- Head noun post-modifiers:
  - **Prepositional phrases**:

$$Nominal \rightarrow Nominal\ PP$$

    - All flights *from Cleveland*
    - Arrival *in San Jose* *before 11 p.m*.

(2) The Noun Phrase

- Head noun post-modifiers:
  - **Non-finite clauses:**
    - Gerundive (-ing): a verb phrase that begins with a gerundive (-ing) form
      - Any flight *leaving on Thursday*
      - Flights *arriving within thirty minutes*

$$Nominal \rightarrow Nominal\ GerundVP$$

    - Infinitive forms
      - The last flight *to arrive in Boston*
    - *-ed* and :
      - Which is the aircraft *used by this flight*?

(2) The Noun Phrase

- Head noun post-modifiers:
  - **Relative clauses:** a clause that begins with a relative pronoun (that, who, …)

$$\begin{aligned} Nominal &\rightarrow Nominal\ RelClause \\ RelClause &\rightarrow (who \mid that)\ VP \end{aligned}$$

    - Flights *that leave in the morning*

  - Combining multiple modifiers:
    - All flights *from Boston* *leaving before 5 p.m*

(3) The Verb Phrase

- Some basic VP production rules:

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

- An entire sentence (**sentential complement**) can follow a verb

$$VP \rightarrow Verb\ \ S$$

  – I think *I would like to take the 9:30 a.m flight*
  – Tell me *how to get from the airport to downtown.*

(3) The Verb Phrase

- Another VP can follow a verb, such as the infinitive **VP complements** that can follow some verbs
  - I want *to fly from Denver to Chicago*

- Not every verb is compatible with every verb phrase.
  - "want" can take either an NP complement (*want a flight*), or an infinitive VP complement (*want to book a flight*).
  - "find" cannot take a VP complement, only NP complements (*I found a flight*)
  - Some verbs cannot take any complements, like "disappear"
    - Traditionally called **intransitive** verbs

(3) Verb Phrase

- Subcategorization frames → the possible set of complements
- Define separate types of verbs:

$$\begin{aligned} \textit{Verb-with-NP-complement} &\rightarrow \textit{find} \mid \textit{leave} \mid \textit{repeat} \mid \ldots \\ \textit{Verb-with-S-complement} &\rightarrow \textit{think} \mid \textit{believe} \mid \textit{say} \mid \ldots \\ \textit{Verb-with-Inf-VP-complement} &\rightarrow \textit{want} \mid \textit{try} \mid \textit{need} \mid \ldots \end{aligned}$$

- And modify VP productions accordingly:

$$\begin{aligned} VP &\rightarrow \textit{Verb-with-no-complement} \quad \text{disappear} \\ VP &\rightarrow \textit{Verb-with-NP-comp NP} \quad \text{prefer a morning flight} \\ VP &\rightarrow \textit{Verb-with-S-comp S} \ \text{said there were two flights} \end{aligned}$$

(4) Coordination

- Join multiple phrases using **conjunctions** (and, or, but, …) to form larger phrases of the same type.

$$\begin{aligned} NP &\rightarrow NP \textit{ and } NP \\ VP &\rightarrow VP \textit{ and } VP \end{aligned}$$

  – I need to know *the aircraft* and *the flight number*.
  – Which flights do you have *leaving Denver* and *arriving in San Francisco*?

- All non-terminals can be conjoined in this manner, so we can define a general rule:

$$X \rightarrow X \textit{ and } X$$

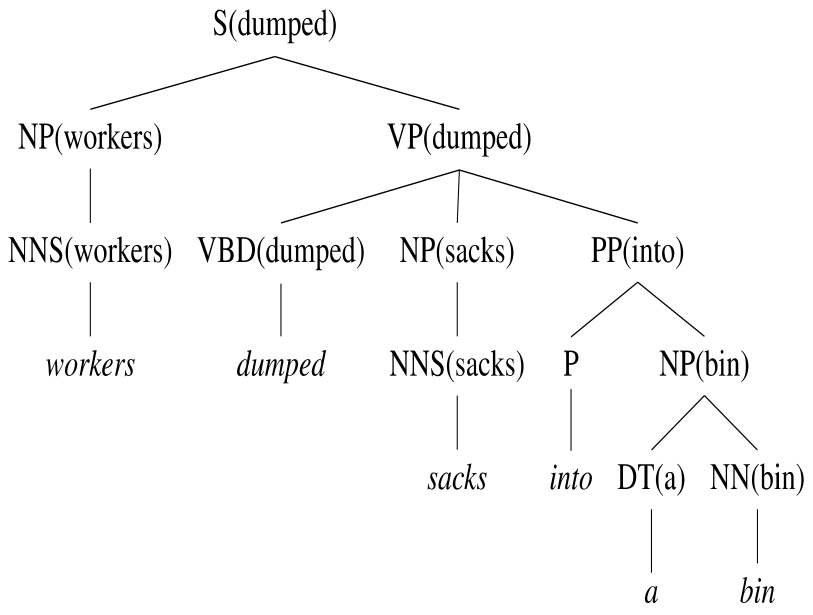
Heads and Head Finding

- Syntactic constituents can be associated with a lexical **head**

- A head → a specific word in the phrase that is grammatically the most important:
  - A noun is the head of an NP (e.g. the **dog** walking by)
  - A verb is the head of VP (e.g. **walks** up the street)

Lexicalized Parse Trees

- Each constituent is annotated with a lexical head



Identifying Head Words

- CFG rules can be augmented to identify one right-hand constituent as the head child
  - Choosing a head child node can be complicated in practice

- Alternatively, head words can be identified dynamically in the context of specific sentences

Head Finding

- A set of simple rules can be used to annotate nodes in parse trees with their appropriate head words

- For example, to find the head child of VP:
    - If the VP contains a V, the head child is the leftmost V;
    - Else, if the VP contains a MD, the head child is the leftmost MD;
    - Else, if the VP contains a VP, the head child is the leftmost VP;
    - Else, the head is the leftmost child.

- Apply the rules bottom up to annotate with lexical heads (The headword of a constituent is the headword of its head child).

TreeBanks

Treebanks

- A **treebank** → an annotated corpus of sentences and their corresponding parse trees

- The **Penn Treebank** project has produced treebanks for various corpora including the Brown Corpus and the Wall Street Journal for English
  - As well as some treebanks for few other languages.

Penn TreeBank (PTB) Format

“The flight should arrive at 11 a.m tomorrow”

```
((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 )))))
```

PTB Format

“The flight should arrive at 11 a.m tomorrow”

PTB POS Tags

```
((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 )))))
```

Traces of Syntactic Movement

“We would have ^ to wait until we have collected on those assets, ” he said ^.

```
( (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) ))
     (. .) ))
```

PTB Grammar

- We can infer the Context-Free Grammar of the language represented by the PTB corpus

- The grammar is relatively flat, resulting in numerous long rules. For example:

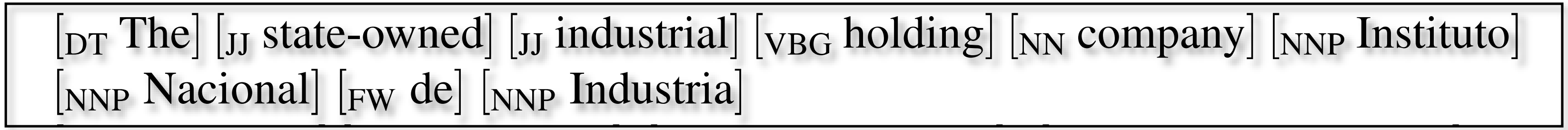
$$\texttt{VP} \rightarrow \texttt{VBP PP PP PP PP PP ADVP PP}$$

This mostly happens because *we go **from** football **in** the fall **to** lifting **in** the winter **to** football **again in** the spring.*

PTB Grammar – Cont.

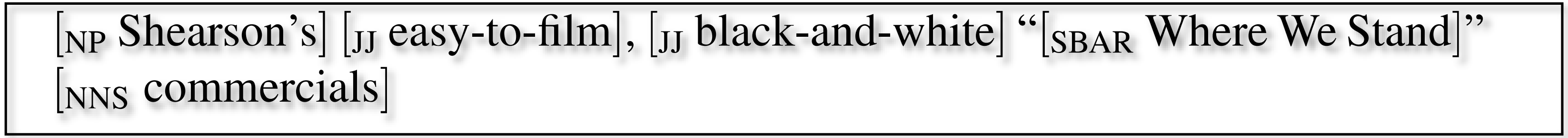
- Examples of NP productions:

$$\text{NP} \rightarrow \text{DT JJ JJ VBG NN NNP NNP FW NNP}$$



[$_{\text{DT}}$ The] [$_{\text{JJ}}$ state-owned] [$_{\text{JJ}}$ industrial] [$_{\text{VBG}}$ holding] [$_{\text{NN}}$ company] [$_{\text{NNP}}$ Instituto] [$_{\text{NNP}}$ Nacional] [$_{\text{FW}}$ de] [$_{\text{NNP}}$ Industria]

$$\text{NP} \rightarrow \text{NP JJ , JJ `` SBAR '' NNS}$$



[$_{\text{NP}}$ Shearson's] [$_{\text{JJ}}$ easy-to-film], [$_{\text{JJ}}$ black-and-white] "[$_{\text{SBAR}}$ Where We Stand]" [$_{\text{NNS}}$ commercials]

Grammar Equivalence

- Two formal grammars are equivalent if they produce the same set of strings
  - **Strong equivalence**: the two grammars generate the same set of strings *and* assign the same parse tree to each sentence (merely renaming of the non-terminal symbols)
  - **Weak equivalence**: the two grammars generate the same set of strings but do not assign the same parse tree to each sentence
    - A $\rightarrow$ B C D can be represented as the following two rules
      - A $\rightarrow$ B X
      - X $\rightarrow$ C D

Chomsky Normal Form (CNF)

Normal Form

- A grammar is said to be in **Chomsky Normal Form** (CNF) if it is ε-free and all productions are of the form $A \rightarrow B\,C$ or $A \rightarrow a$
  - *The right-hand side is either two non-terminals or one terminal.*

- CNF grammars have binary parse trees

- Any CFG can be converted to a weakly equivalennt CNF grammar

Normal Form - Example

- The following PTB rules:

$$\text{VP} \rightarrow \text{VBD NP PP}$$
$$\text{VP} \rightarrow \text{VBD NP PP PP}$$
$$\text{VP} \rightarrow \text{VBD NP PP PP PP}$$
$$\text{VP} \rightarrow \text{VBD NP PP PP PP PP}$$

…

- Can be converted to:

$$\text{VP} \rightarrow \text{VBD NP}$$
$$\text{NP} \rightarrow \text{NP PP}$$

Converting a CFG to CNF

- Three situations:
  - Rules that mix terminals and non-terminals in the right-hand side
  - Rules with a single non-terminal in the right-hand side (unit productions)
  - Rules with more than 2 non-terminals in the right-hand side

Converting a CFG to CNF

- Rules that mix terminals and non-terminals in the right-hand side → introduce a new dummy non-terminal that only covers that terminal in the rule

$$INF\text{-}VP \rightarrow to\ VP \quad \Longrightarrow \quad \begin{array}{l} INF\text{-}VP \rightarrow TO\ VP \\ TO \rightarrow to \end{array}$$

Converting a CFG to CNF

- Rules with a single non-terminal in the right-hand side (unit productions) → eliminate unit productions by replacing them with all possible non-unit production rules

$$\begin{array}{l} \text{VP} \rightarrow \text{INF-VP} \\ \text{INF-VP} \rightarrow \text{TO VP} \end{array} \quad \Longrightarrow \quad \text{VP} \rightarrow \text{TO VP}$$

Converting a CFG to CNF

- Rules with more than 2 non-terminals in the right-hand side → introduce new non-terminals that spread the longer sequence over several new rules

- The choice of what to replace is random

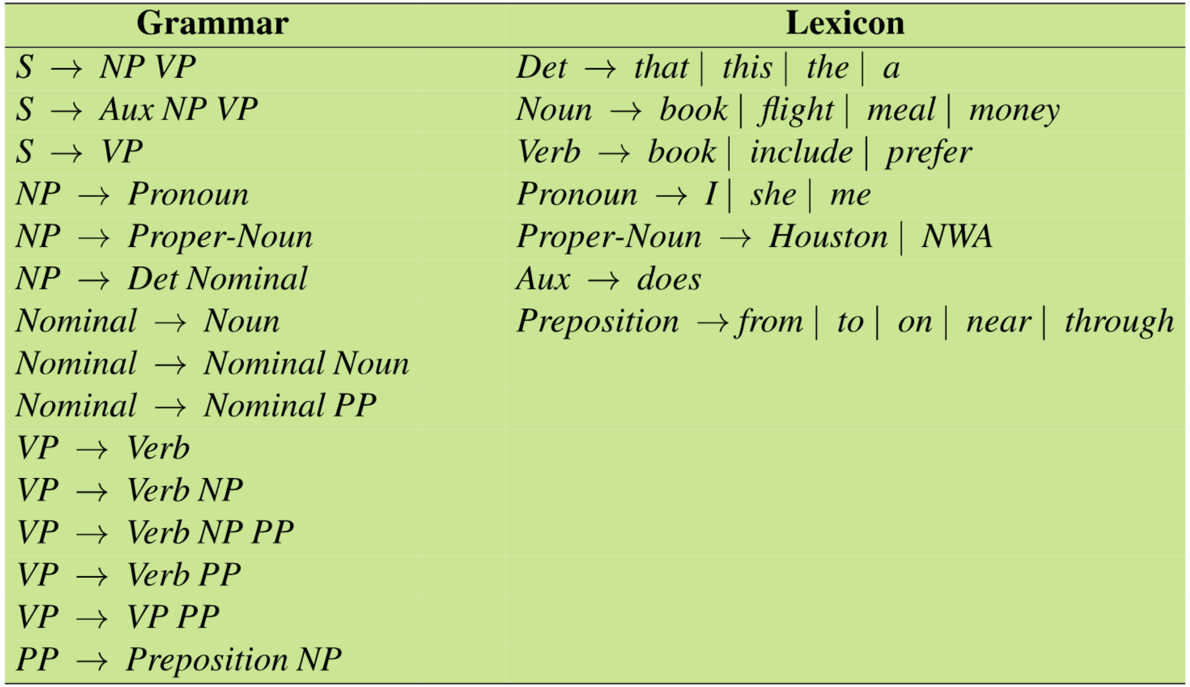
$$S \rightarrow Aux\ NP\ VP \quad \Longrightarrow \quad \begin{array}{rcl} S & \rightarrow & X1\ VP \\ X1 & \rightarrow & Aux\ NP \end{array}$$

Converting a CFG to CNF - Summary

1. Copy all conforming rules to the new grammar unchanged
2. Convert terminals within rules to dummy non-terminals
3. Convert unit-productions
4. Make all rules binary and add them to new grammar

Example

Convert the following Grammar to CNF



Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Syntactic Parsing

Syntactic Parsing

- The process of generating (possibly multiple) parse-trees given a sentence and a formal grammar

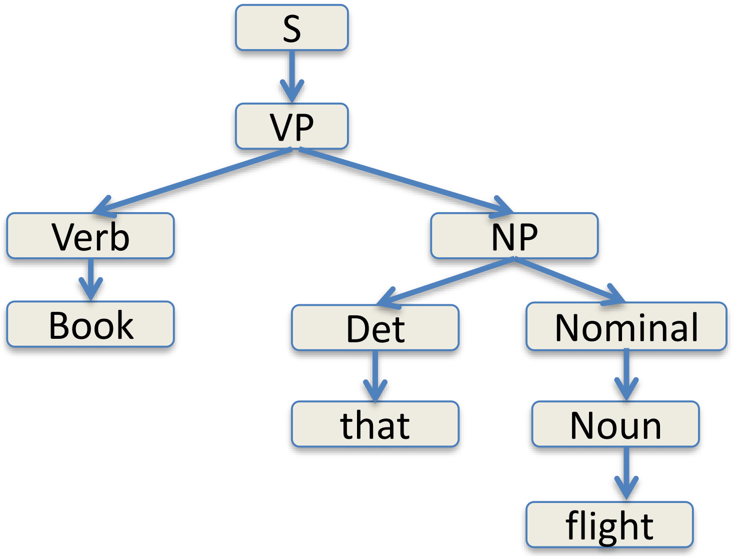
- Parsing with CFGs : all possible parse trees that span the entire input sentence and have **S** as the root

Syntactic Parsing as Search

- Searching through the space of possible parse trees to find the correct parse tree for a given sentence

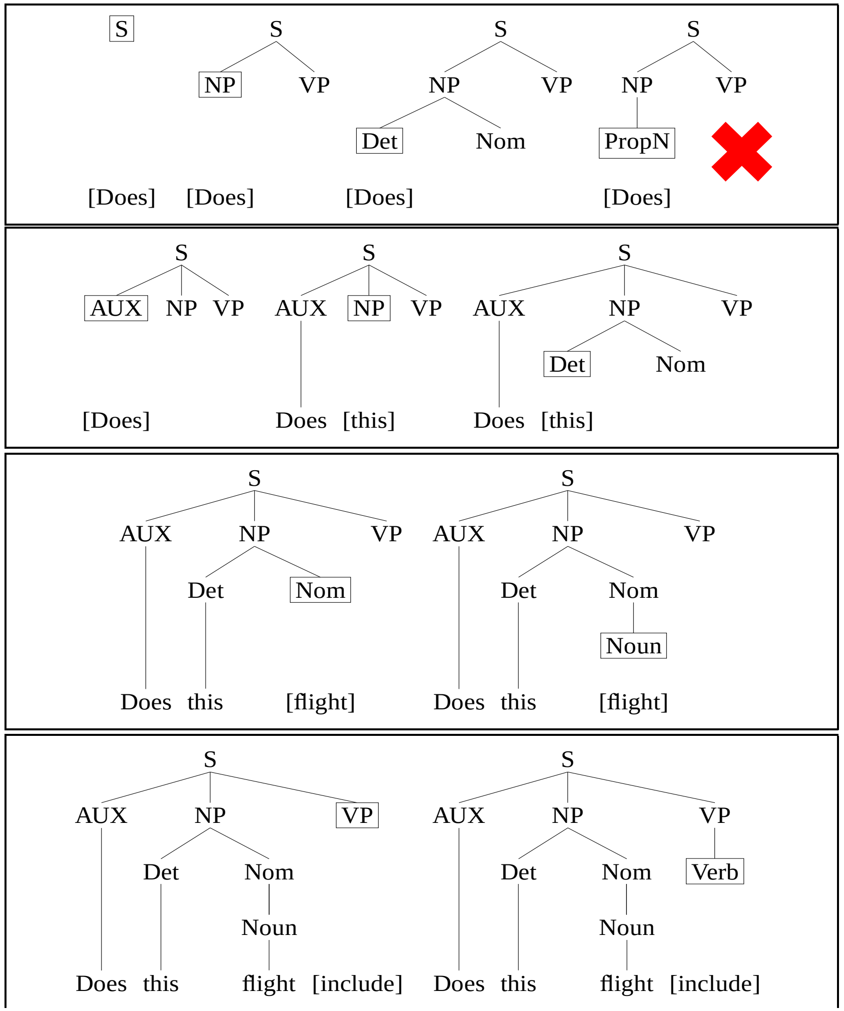
- Two search strategies:
  - Top-Down Parsing
  - Bottom-Up Parsing

- Example: Book that flight $\rightarrow$



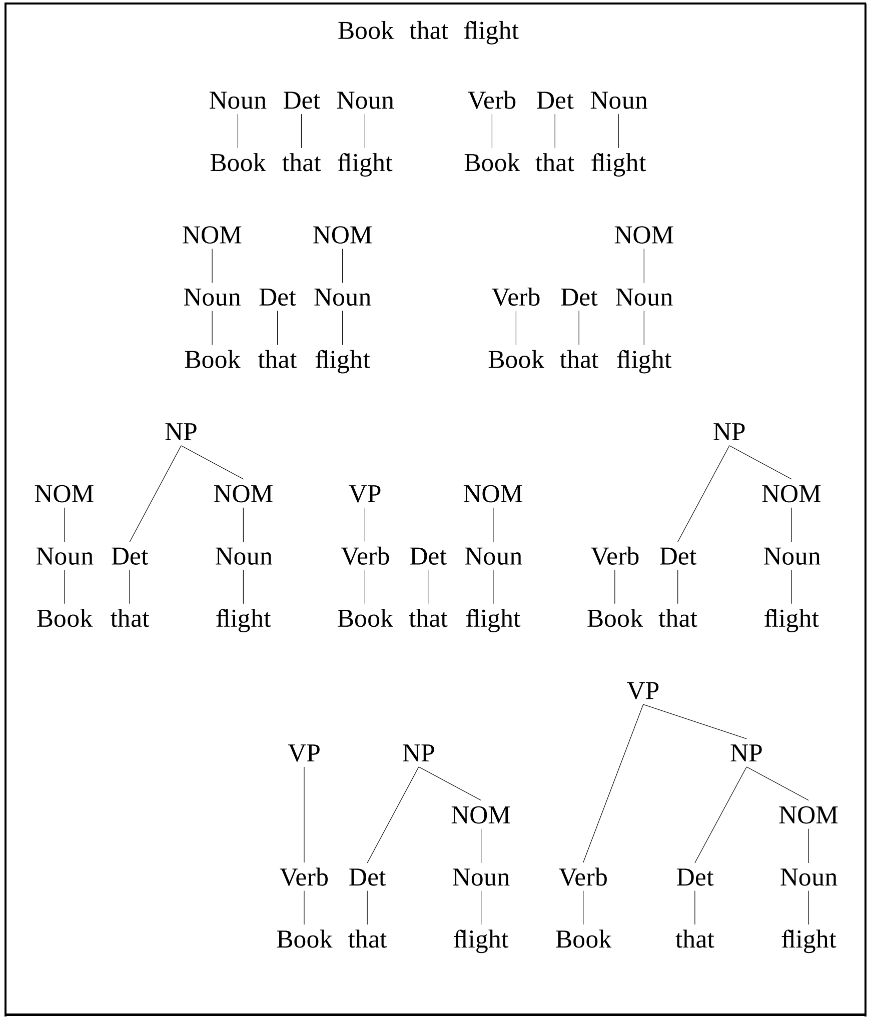
Top-Down Parsing

- *Example: Does this flight include …*
- Start from S, then expand all production rules
- Eliminate rules that don't lead to words in the observed sentence
- The trees are expanded **depth-first** (expanding the most-recently generated nodes) and left-to-right.



Bottom-up Parsing

- *Example: Book that flight*
- Start from the observed words, and combine them upwards
- A successful parse results in a tree rooted at **S** that spans all words in the sentence



Repeated Parsing of Sub-trees

- Both search approaches have drawbacks
  - **local ambiguity** (ambiguity that results from looking at partial input)
  - the naïve approaches described so far result in repeated parsing of sub-trees

- **Dynamic programming** parsing algorithms use a table of partial-parses to efficiently parse ambiguous sentences.
  - Top-down: The **Earley** algorithm (not covered in this class)
  - Bottom-up: The **CKY** algorithm

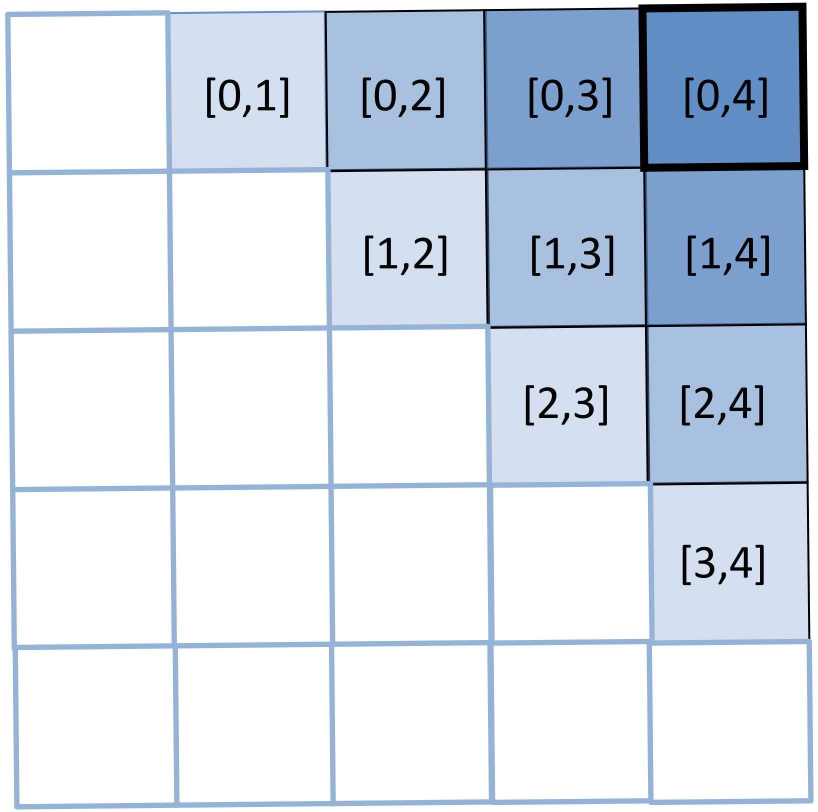
CKY Parsing

- The **CKY** algorithm is a dynamic programming approach to bottom-up syntactic parsing

- Requires grammars in Chomsky Normal Form (CNF).
  - Can be used with any Context-Free Grammar, but it has to be converted to CNF first

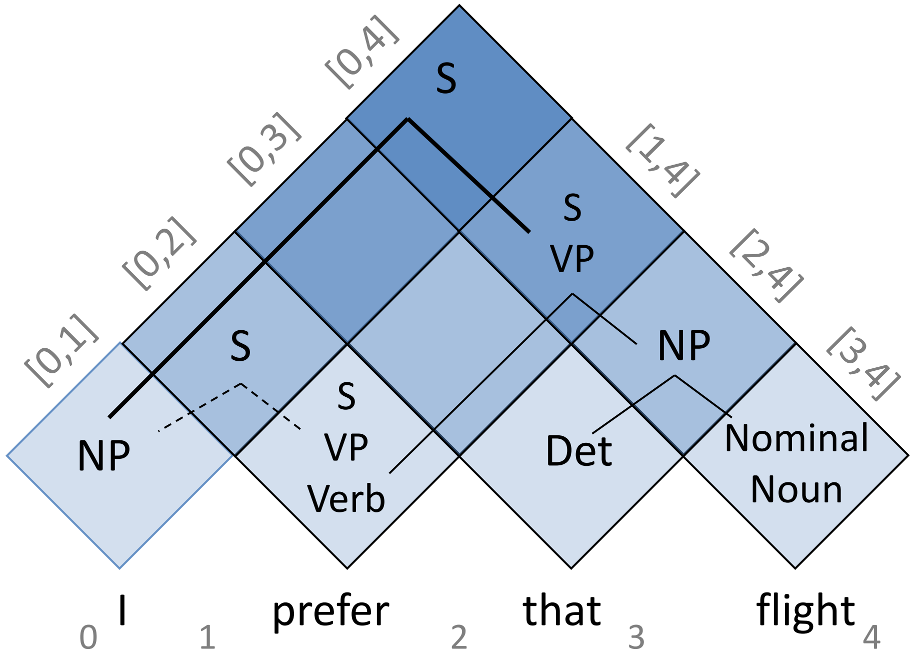
CKY Recognition

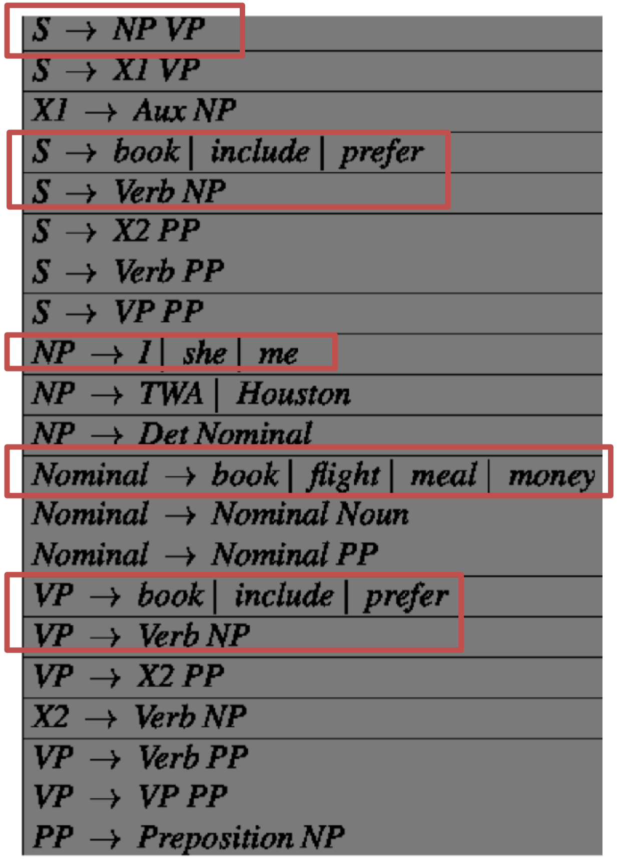
- A two-dimensional matrix is used to encode the structure of a binary tree
- For a sentence of length **n**, we work with the upper right triangle of an **(n+1) x (n+1)** matrix
- Each cell **[i,j]** contains the set of **non-terminals** that span the input from i to j
- The cell that represents the entire parse tree is **[0,n]** (the upper right corner)



Example

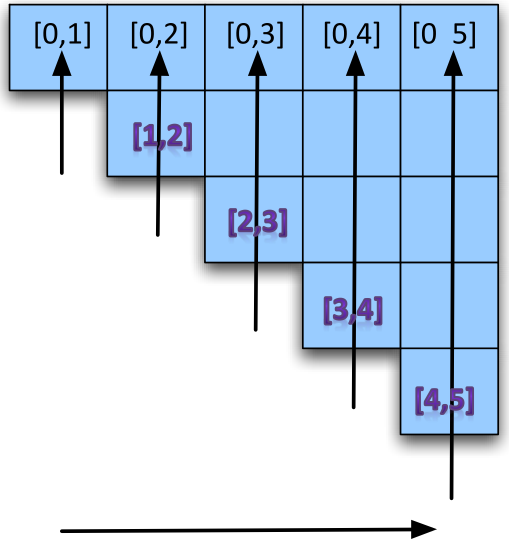
- “I prefer that flight”





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

CKY Recognition Algorithm

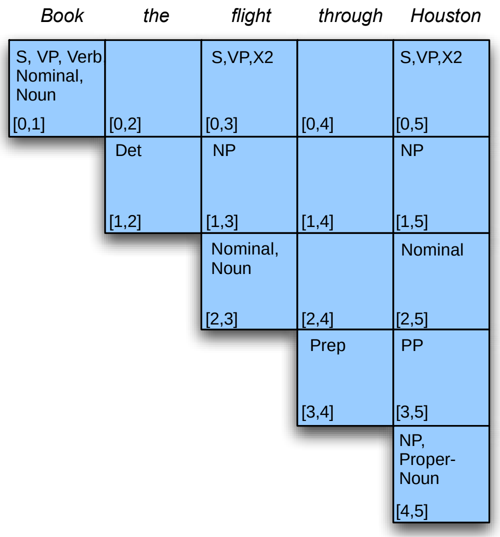


```
function CKY-PARSE(words, grammar) returns table

  for j    from 1 to LENGTH(words) do
      for all { A | A !    words[j]  2  grammar}
              table[j − 1, j]     table[j − 1, j] [  A
      for i     from j − 2 downto 0 do
          for k     i + 1 to j − 1 do
              for all { A | A !    BC  2  grammar and B  2  table[i, k] and C  2  table[k, j]}
                      table[i,j]     table[i,j] [  A
```

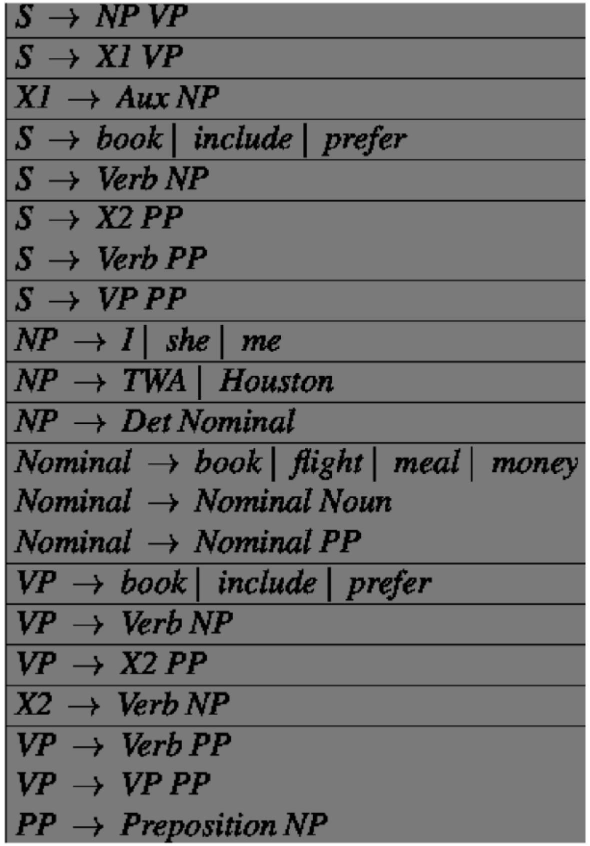
Example

- “Book the flight through **Houston**”



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]

Completed parse tree using CKY



S	→	*NP VP*
S	→	*X1 VP*
X1	→	*Aux NP*
S	→	*book* | *include* | *prefer*
S	→	*Verb NP*
S	→	*X2 PP*
S	→	*Verb PP*
S	→	*VP PP*
NP	→	*I* | *she* | *me*
NP	→	*TWA* | *Houston*
NP	→	*Det Nominal*
Nominal	→	*book* | *flight* | *meal* | *money*
Nominal	→	*Nominal Noun*
Nominal	→	*Nominal PP*
VP	→	*book* | *include* | *prefer*
VP	→	*Verb NP*
VP	→	*X2 PP*
X2	→	*Verb NP*
VP	→	*Verb PP*
VP	→	*VP PP*
PP	→	*Preposition NP*

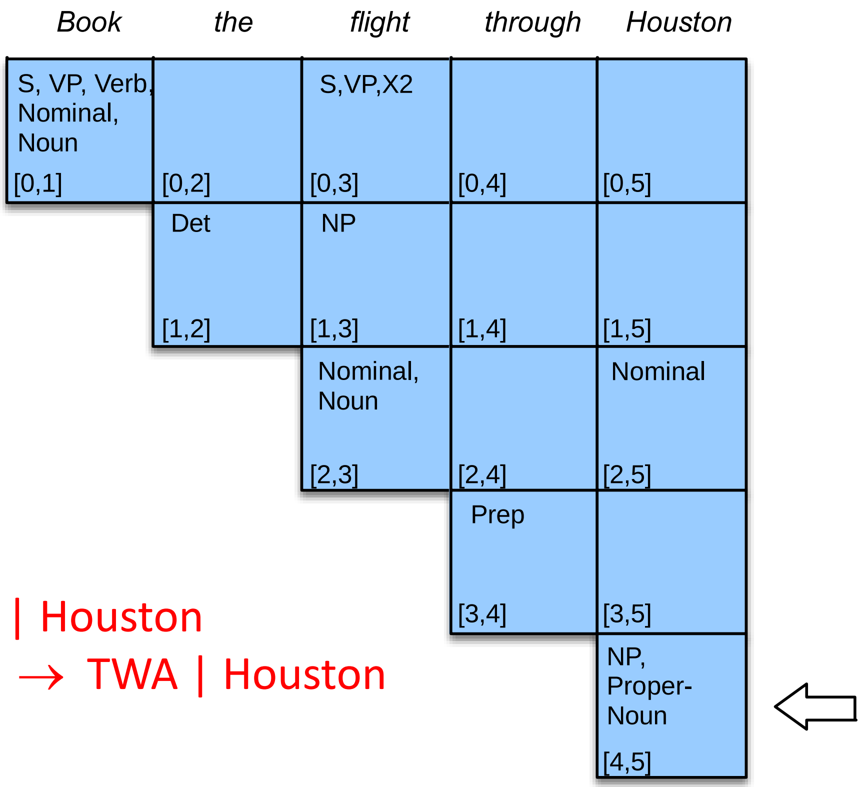
Example

- “Book the flight through **Houston**”
- j=5

$table[j-1, j] \quad table[j-1, j]\ [\ A$

Table[4,5] = {NP, Proper-Noun}

NP $\rightarrow$ TWA | Houston
ProperNoun $\rightarrow$ TWA | Houston



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]	Nominal [2,5]
			Prep [3,4]	[3,5]
				NP, Proper-Noun [4,5]

Example

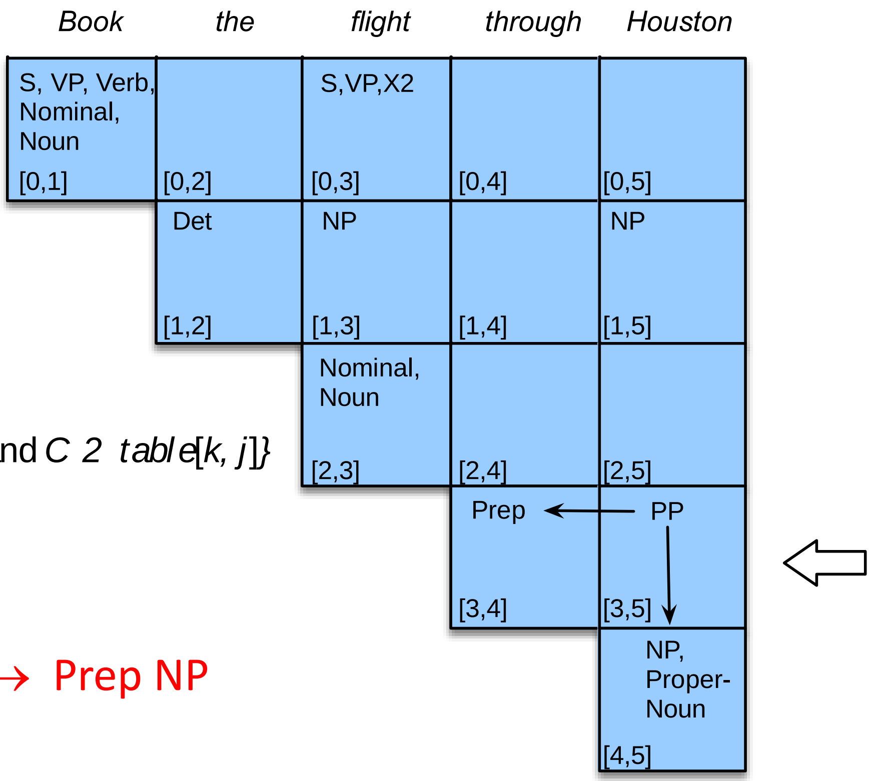
- "Book the flight through **Houston**"
- j=5

i=3, k=4

```
for i   from j − 2 downto 0 do
    for k   i + 1 to j − 1 do
        for all { A | A !   BC 2 grammar and B 2 table[i, k] and C 2 table[k, j]}
                table[i,j]   table[i,j] [ A
```

Table[3,5] = {PP}

PP $\rightarrow$ Prep NP



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]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	[2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Example

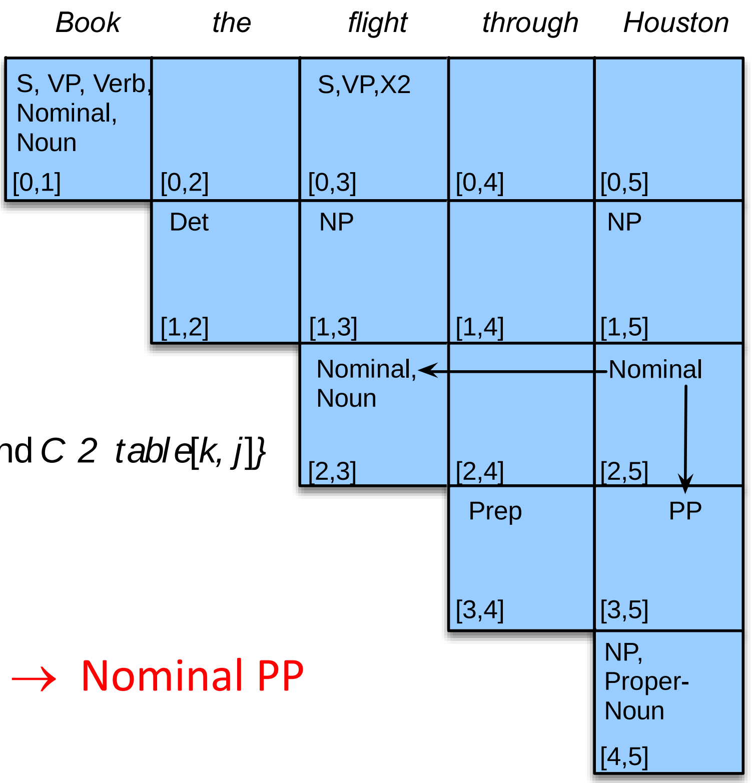
- “Book the flight through **Houston**”
- j=5

i=2, k=3

```
for i    from j − 2 downto 0 do
    for k    i + 1 to j − 1 do
        for all { A | A !    BC 2 grammar and B 2 table[i, k] and C 2 table[k, j]}
                table[i,j]    table[i,j] [  A
```

Table[2,5] = {Nominal}

$\text{Nominal} \rightarrow \text{Nominal PP}$



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]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

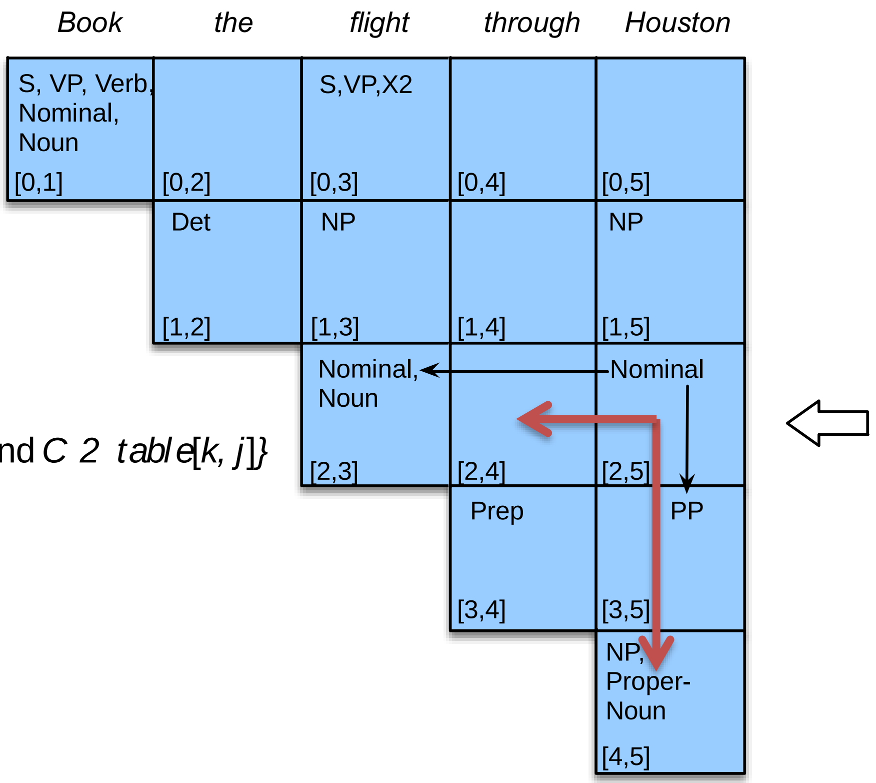
Example

- “Book the flight through **Houston**”
- j=5

i=2, k=4

```
for $i$   from $j-2$ downto 0 do
    for $k$   $i+1$ to $j-1$ do
        for all $\{A \mid A\ !\ \ BC\ 2\ grammar$ and $B\ 2\ table[i,k]$ and $C\ 2\ table[k,j]\}$
                $table[i,j]\ \ \ table[i,j]\ [\ \ A$
```

Table[2,5] = {Nominal}



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]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Example

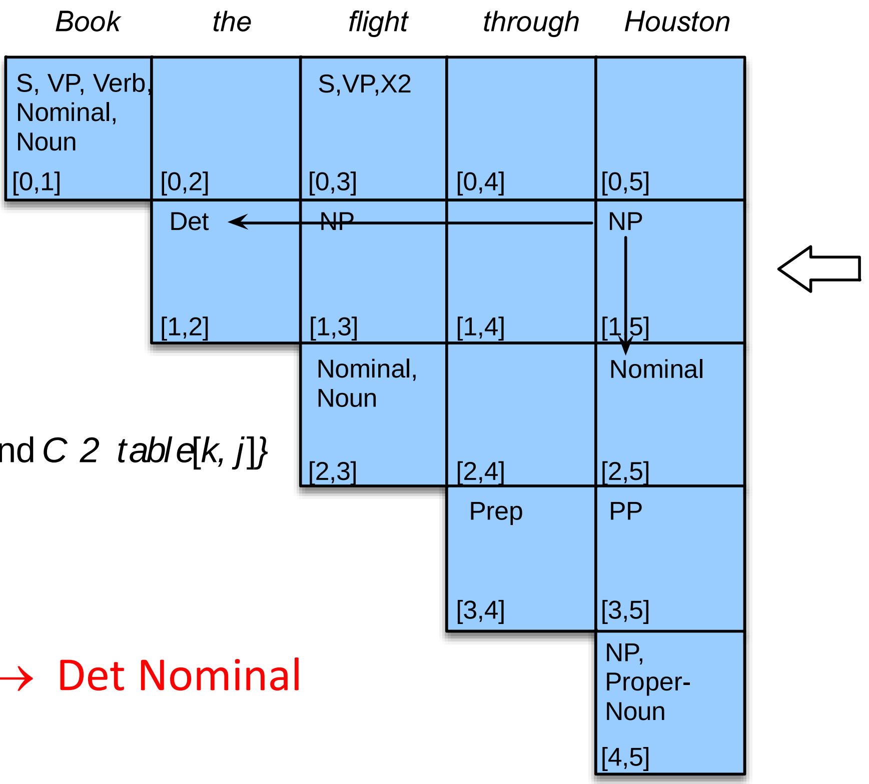
- “Book the flight through **Houston**”
- j=5

i=1, k=2

```
for i     from j − 2 downto 0 do
     for k     i + 1 to j − 1 do
          for all { A | A !     BC 2  grammar and B 2  table[i, k] and C 2  table[k, j]}
                    table[i,j]     table[i,j] [  A
```

Table[1,5] = {NP}

NP $\rightarrow$ Det Nominal



Example

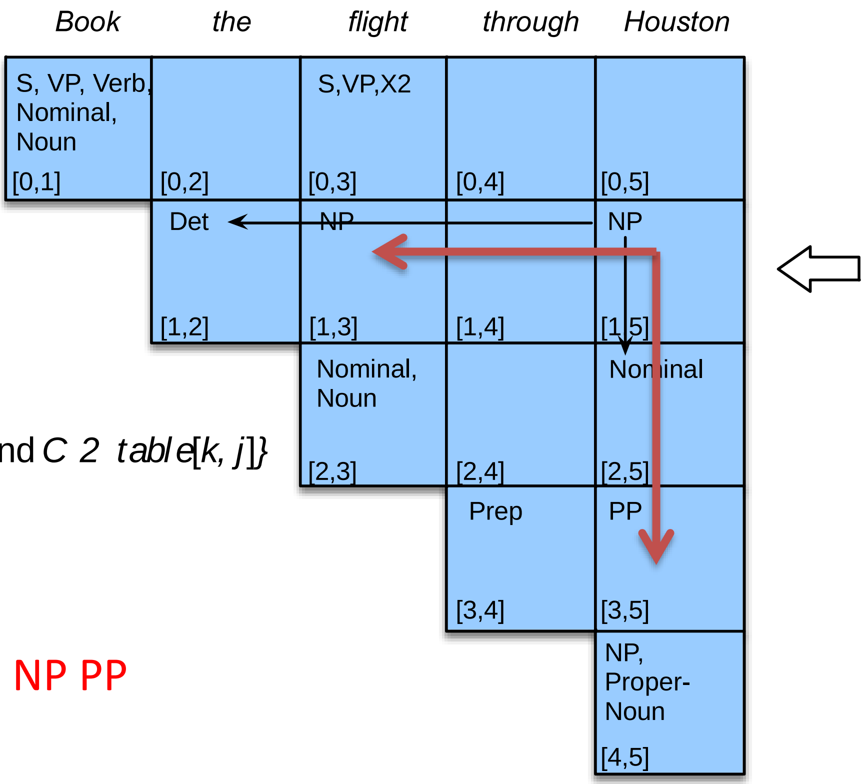
- “Book the flight through **Houston**”
- j=5

i=1, k=3

```
for i   from j − 2 downto 0 do
    for k   i + 1 to j − 1 do
        for all { A | A !   BC 2 grammar and B 2 table[i, k] and C 2 table[k, j]}
                table[i,j]   table[i,j] [ A
```

Table[1,5] = {NP}

$? \rightarrow$ NP PP



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]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Example

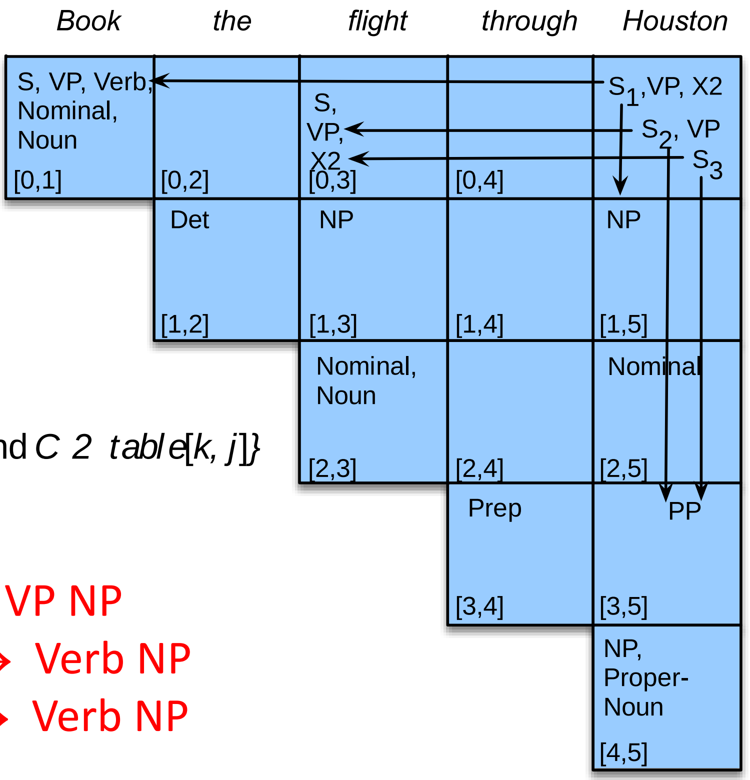
- “Book the flight through **Houston**”
- j=5

i=0, k=1

```
for i      from j − 2 downto 0 do
     for k     i + 1 to j − 1 do
          for all { A | A !    BC 2  grammar and B 2  table[i, k] and C 2  table[k, j]}
                    table[i,j]     table[i,j] [   A
```

Table[0,5] = {S_1, VP, X2}

$S \rightarrow$ VP NP
VP $\rightarrow$ Verb NP
X2 $\rightarrow$ Verb NP



Example

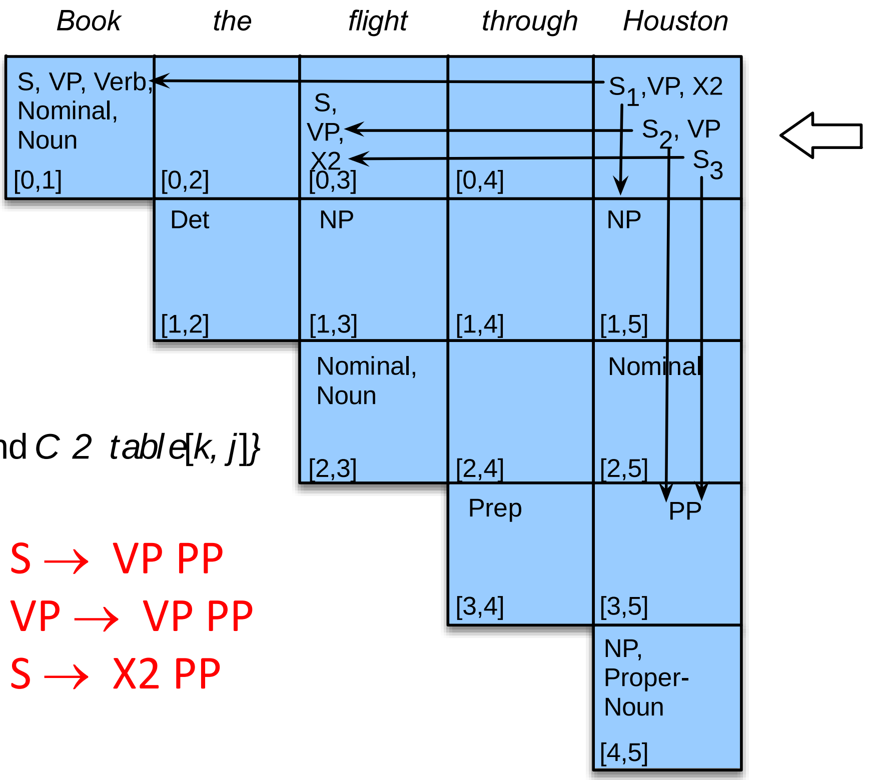
- “Book the flight through **Houston**”
- j=5

i=0, k=3

```
for i     from j − 2 downto 0 do
    for k     i + 1 to j − 1 do
        for all { A | A !    BC 2  grammar and B 2  table[i, k] and C 2  table[k, j]}
                    table[i,j]     table[i,j] [  A
```

Table[0,5] = {S_1, VP, X2, S_2, VP, S_3}

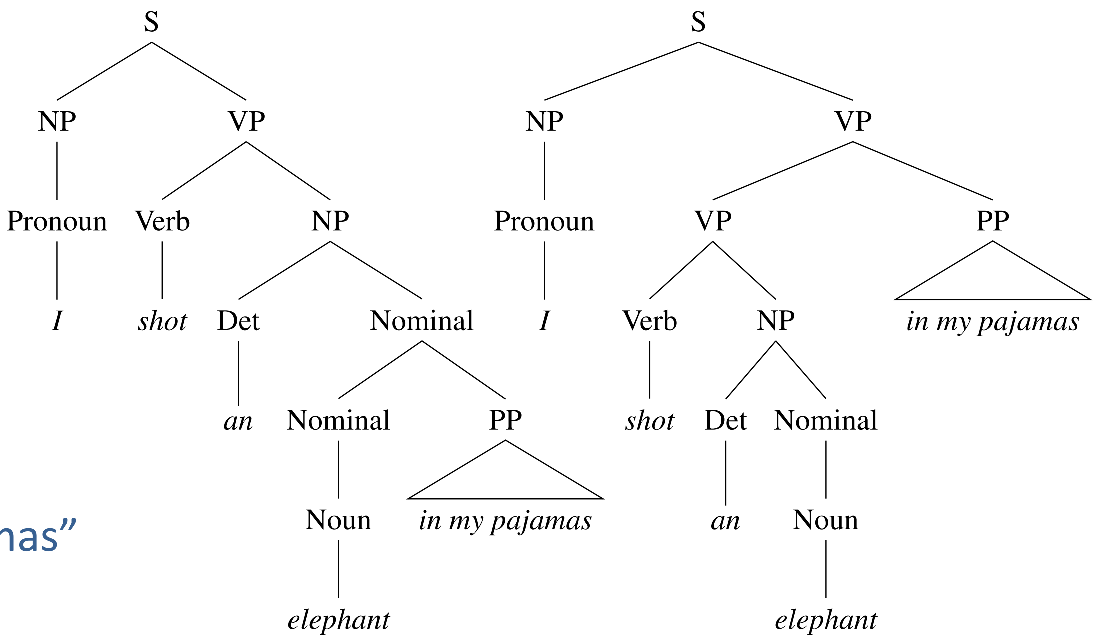
$S \rightarrow VP\ PP$
$VP \rightarrow VP\ PP$
$S \rightarrow X2\ PP$



Structural Ambiguity

Structural Ambiguity

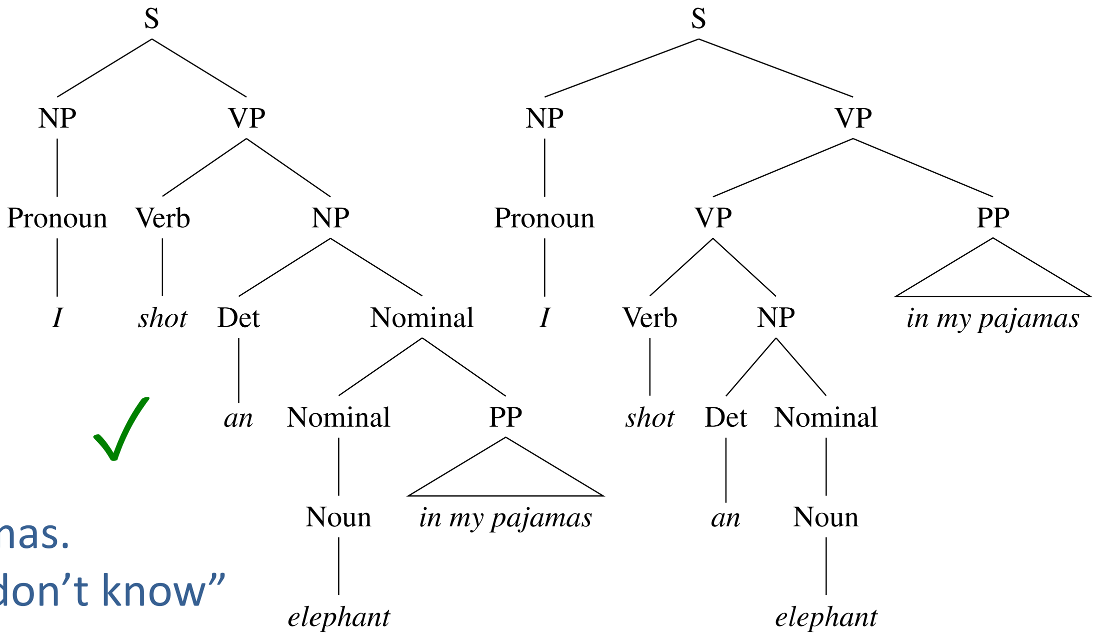
- When a grammar can generate multiple parse trees from a sentence.



“I shot an elephant in my pajamas”

Structural Ambiguity

- When a grammar can generate multiple parse trees from a sentence.



✓

"I shot an elephant in my pajamas.
How he got into my pajamas I don't know"

Structural Ambiguity

- **Attachment ambiguity**: when a constituent can be attached to the tree in more than one place.
  - "We saw the Eiffel Tower flying to Paris"

- **Coordination ambiguity**: ambiguity that arises from conjunctions, such as *and*.
  - "Nation-wide television and radio"

Disambiguation

- Out of all the valid parse trees generated by the Grammar, which is the correct one?
  - Or rather, which one is most **likely** to be the correct one?
  - As usual, we need a probabilistic model …
    - Probabilistic Formal Grammar
    - A probabilistic parser
    - A Treebank to train the model

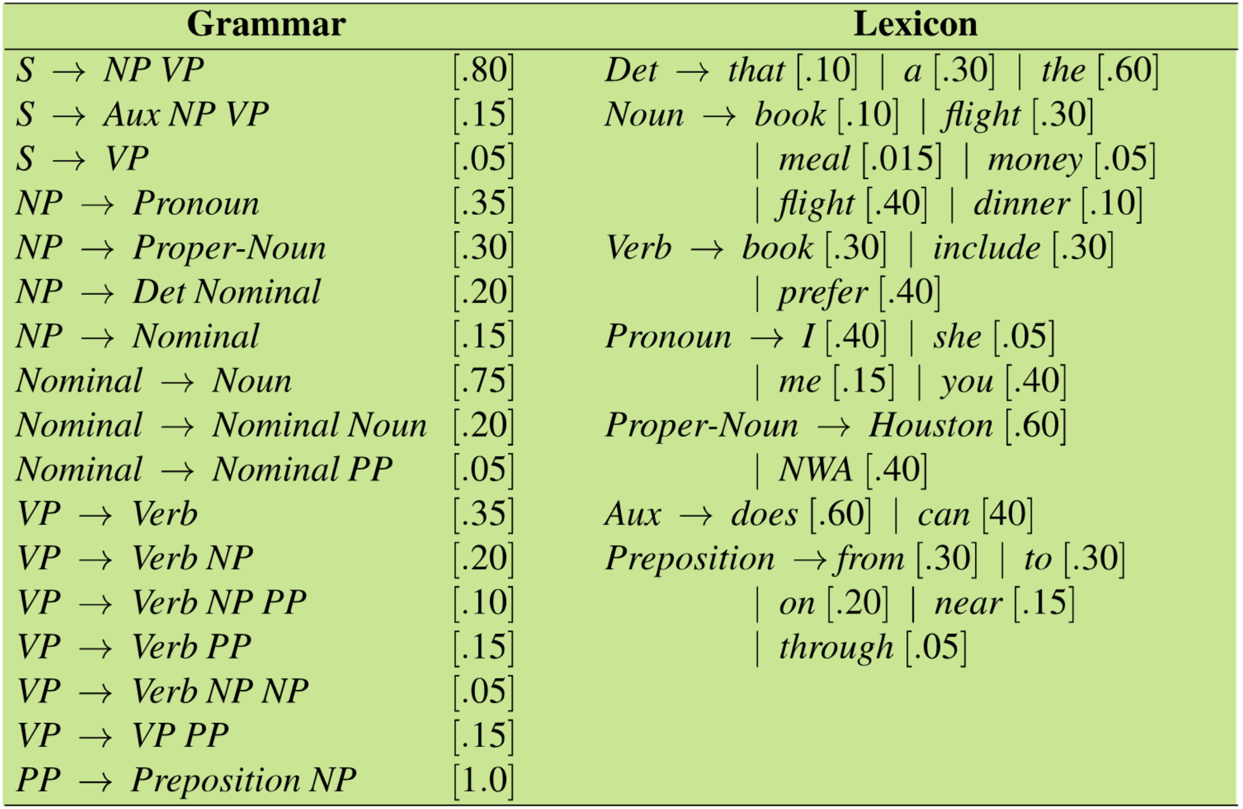
Probabilistic CFG (PCFG)

- G = (T, N, S, R, P)
  - T is a set of terminal symbols
  - N is a set of nonterminal symbols
  - S is the start symbol (S ∈ N)
  - R is a set of rules/productions of the form X $\rightarrow \gamma$
  - P is a probability function
    - P: R $\rightarrow$ [0,1]
    - $\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$

Probabilities of all expansion of each non-terminal sum to 1

- A grammar G generates a language model L : $\sum_{g \in T^*} P(g) = 1$

Example



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* [.015] | *money* [.05]
NP → *Pronoun*	[.35]	| *flight* [.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]	

Using PCFGs

- P(*t*) – The probability of a tree *t* is the product of the probabilities of the rules used to generate it

Disambiguation

- P(*s*) – The probability of the string *s* is the sum of the probabilities of the trees which have that string as their yield

$$P(s) = \Sigma_j P(s, t) \quad \text{, where } t \text{ is a parse of } s$$

Language Modeling

PCFGs for Disambiguation

- Which parse tree is most likely?

$$P(T,S) = P(T)P(S|T)$$

T → a parse tree
S → a sentence

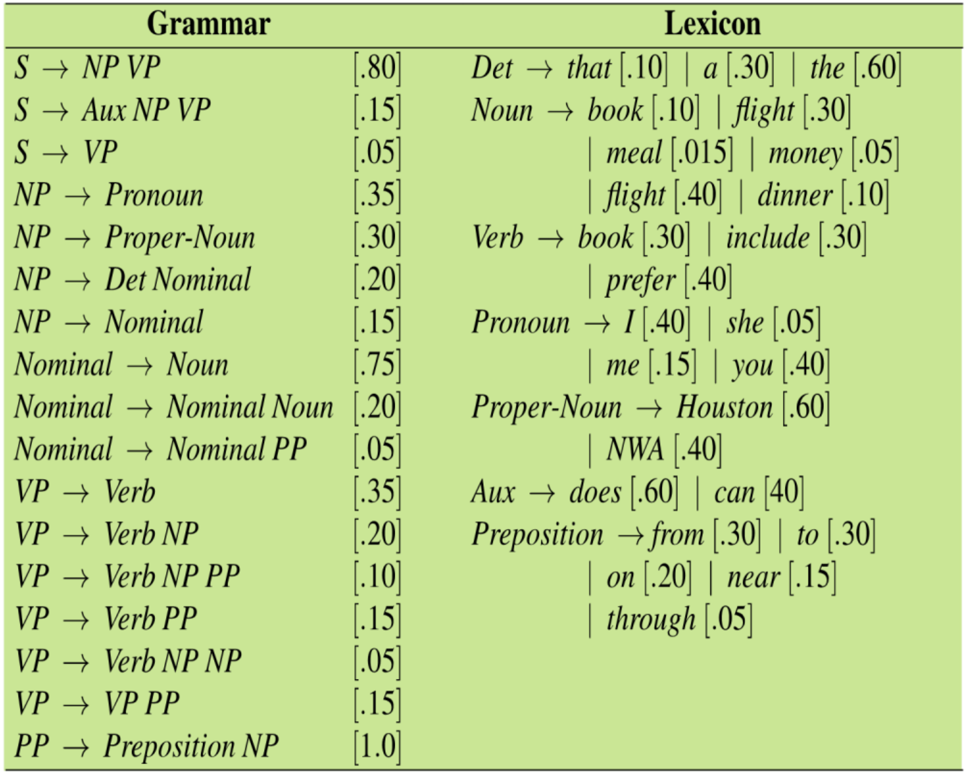
  - Since a parse tree include the words as leaves, P(S|T) = 1.

$$P(T,S) = P(T)P(S|T) = P(T)$$

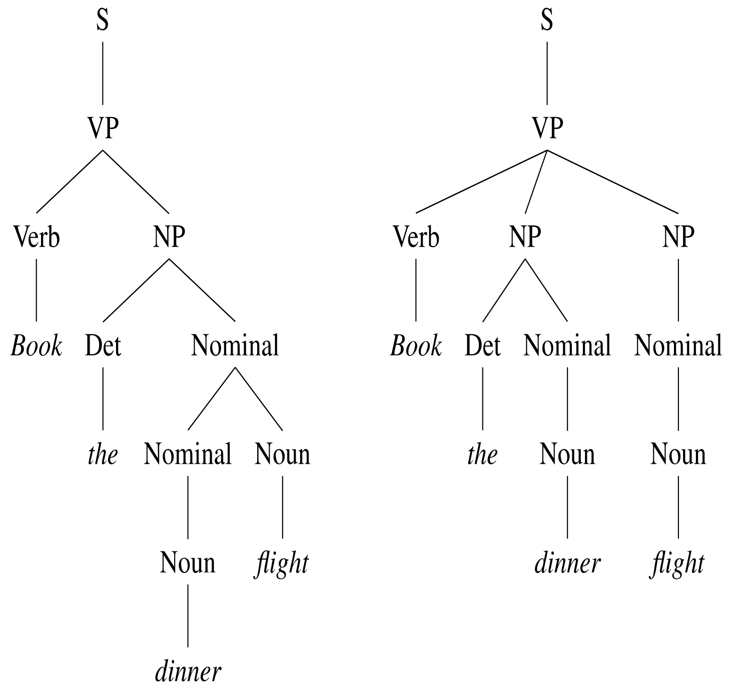
  - Find the most likely parse tree:

$$\hat{T}(S) = \underset{T\,s.t.\,S=\text{yield}(T)}{\text{argmax}} P(T)$$

Which Parse Tree is More Likely



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* [.015] | *money* [.05]
$NP \rightarrow Pronoun$	[.35]	| *flight* [.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]	



$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Probabilistic CKY Parsing

- Given a grammar of **V** non-terminals and a sentence of length **n**, create a 3-dimensional (n+1) x (n+1) x V matrix
  - store probabilities of the various constituents.
  - **Backpointers** to reconstruct the most likely tree.

Example

Find the most likely parse tree for:

"book the flight through Houston"

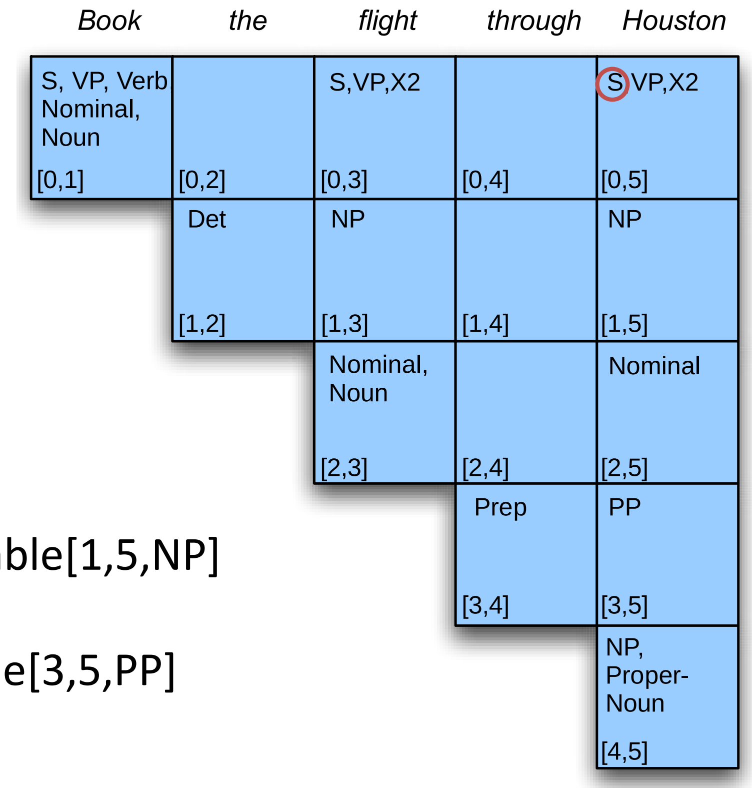
Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.015] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.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]	

Example

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]

$$\max \begin{cases} S \rightarrow VP\ NP \\ P(S \rightarrow VP\ NP) * P(VP \rightarrow Book) * Table[1,3,NP] \\ S \rightarrow Verb\ NP \\ P(S \rightarrow Verb\ NP) * P(VP \rightarrow Book) * Table[1,3,NP] \end{cases}$$

Example



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]

$$\max \begin{cases} S \rightarrow VP\ NP \\ P(S \rightarrow VP\ NP) * P(VP \rightarrow Book) * Table[1,5,NP] \\ S \rightarrow VP\ PP \\ P(S \rightarrow VP\ PP) * Table\ [0,3,VP] * Table[3,5,PP] \end{cases}$$

Learning Rule Probabilities

- **Counting from a Treebank**

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- **No Treebank?** Expectation Maximization …
  - Start with equal probabilities for all rules in a CFG
  - Parse corpus
  - Re-estimate probabilities
  - Repeat until convergence

Evaluation

- Evaluating parse trees at the sentence level is rather harsh
  - Parses are often partially correct, especially for longer sentences
  - So we need a fine-grained evaluation metric

- Measure how much the constituents in a generated parse (hypothesis) resemble hand-annotated constituents from a treebank (reference).
  - A hypothesis parse constituent C_h is labeled correct if there is a reference constituent C_r with the same starting point, end point, and non-terminal symbol

$$\textbf{labeled recall:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of total constituents in hypothesis parse of } s}$$

Problems with PCFG

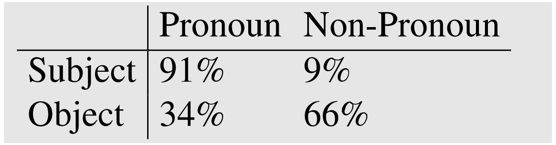
1. Poor independence assumption (structural dependencies)

1. Lack of lexical conditioning (lexical dependencies)

(1) Problems with PCFG

- **Independence assumption between sub-trees**
  - The probability of a particular rule is independent of the rest of the tree
  - Can result in poor probability estimates

  - The choice of how a node expands can depend on the location of the node in the parse tree
    - In English, NPs that are syntactic subjects are more likely to be pronouns that NPs that are syntactic objects



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

(1) Problems with PCFG – Cont.

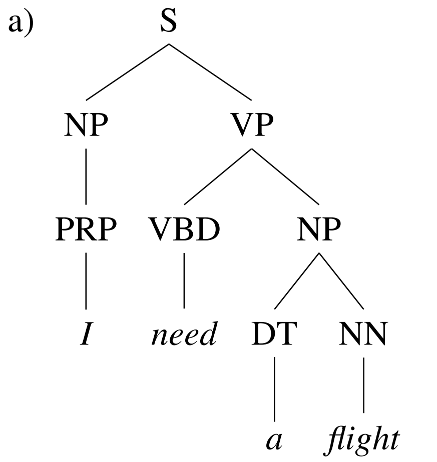
- **Independence assumption between sub-trees**
  - Yet PCFGs only include overall probabilities of each NP production

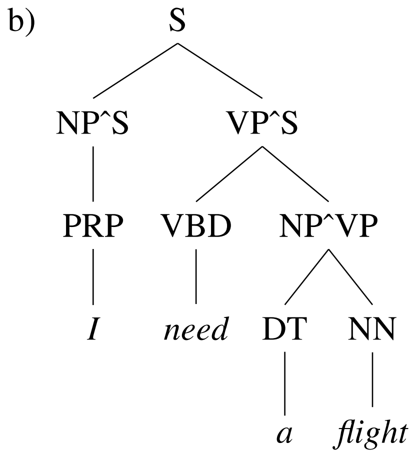
$$\begin{array}{llll} NP & \rightarrow & DT\ NN & .28 \\ NP & \rightarrow & PRP & .25 \end{array}$$

  - This can be resolved to *some extent* by
    - adding **parent annotations**
    - modifying probabilities accordingly.

Parent Annotation

- Example:
  - NPs with S parents (like subjects) are marked ***NPˆS***
  - NPs with VP parents (like objects) are marked ***NPˆVP***.
- Equivalent to **subcategorization**
- Results in larger grammars and may result in overfitting, but relatively simple to implement:
  - only split a rule **if**
    - subcategories are frequent enough
    - **and** the split results in improving performance in a development set





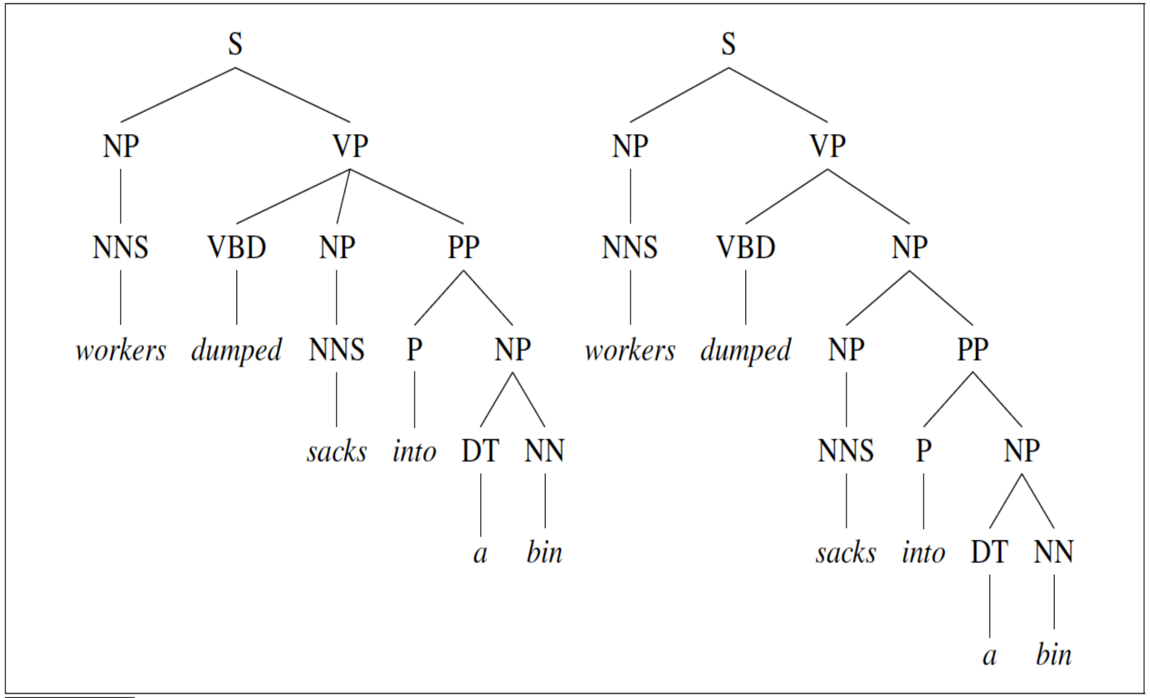
(2) Problems with PCFGs

- **Lack of sensitivity to the words in the parse tree**
  - The parse probabilities include the probability of a word given parts-of-speech

  - Words can be useful for resolving attachment and coordination ambiguities
    - “Dogs in houses and cats” – *cats* is more likely to be conjoined with *dogs* than *houses*.
    - [dogs in houses]and[cats] **vs.** [dogs]in[houses and cats].

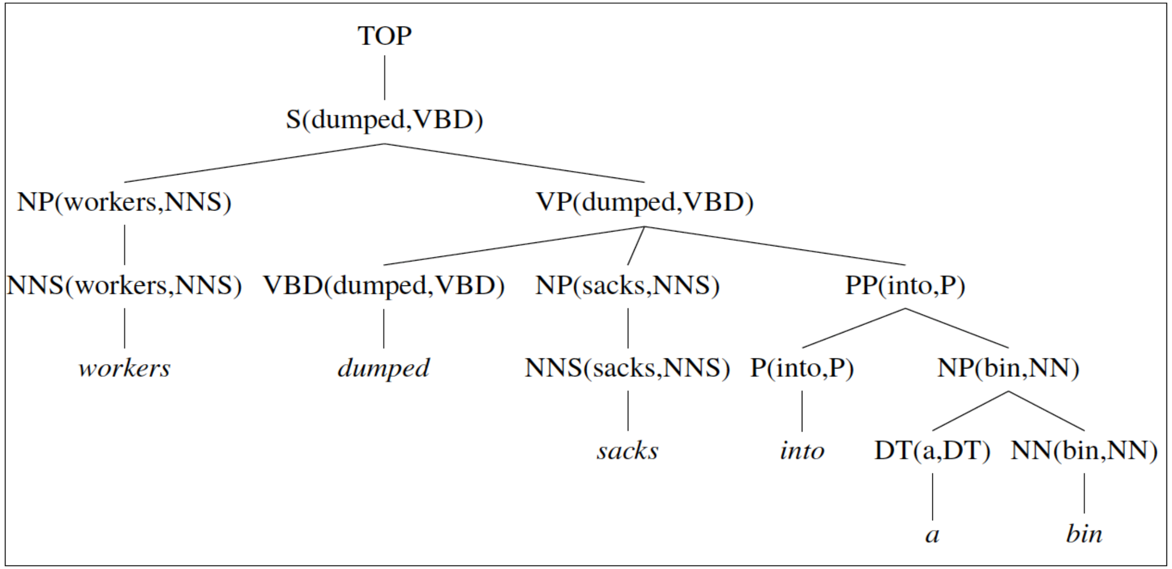
(2) Problems with PCFGs – Cont.

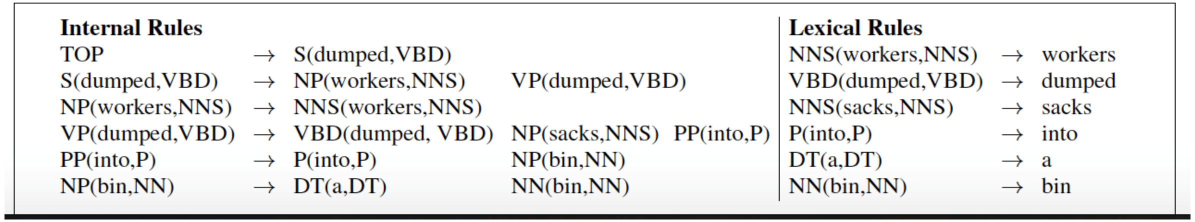
- Workers dumped sacks ***into a bin***



(2) Problems with PCFGs – Cont.

- **Lexicalized PCFGs**
  - Annotate non-terminals with lexical heads
  - Results in large fine-grained rules whose probabilities cannot be estimated directly with MLE





Internal Rules				
TOP	→	S(dumped,VBD)		
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)	
NP(workers,NNS)	→	NNS(workers,NNS)		
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS)	PP(into,P)
PP(into,P)	→	P(into,P)	NP(bin,NN)	
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)	

Lexical Rules		
NNS(workers,NNS)	→	workers
VBD(dumped,VBD)	→	dumped
NNS(sacks,NNS)	→	sacks
P(into,P)	→	into
DT(a,DT)	→	a
NN(bin,NN)	→	bin

(2) Problems with PCFGs – Cont.

- **Lexicalized PCFGs**
  - Additional independence assumptions are introduced to work around that (see **Collins** parser as an example)

Partial Parsing

Partial Parsing

- Partial (or shallow) parse may be sufficient for some tasks
  - E.g. information extraction, information retrieval

- **Chunking** → the process of identifying **non-overlapping** segments of texts that correspond to major constituent types
  - noun phrases, verb phrases, adjective phrases, and prepositional phrases

[$_{NP}$ The morning flight] [$_{PP}$ from] [$_{NP}$ Denver] [$_{VP}$ has arrived.]

Chunking

- Definition of chunks may vary by application

- The following guidelines generally apply:
  - The phrases are non-recursive: they do not include smaller constituents of the same type
    - Only need to identify phrase boundaries and phrase type

  - Post-head modifiers are generally excluded
    - No attachment ambiguities

Chunking as Classification

- Identify phrase boundaries and types in a single classification task using **IOB** tagging
  - Tags for beginning (**B**) and inside (**I**) of each chunk type.
  - Tag **O** for tokens outside any chunk.

The	*morning*	*flight*	*from*	*Denver*	*has*	*arrived*
B_NP	I_NP	I_NP	B_PP	B_NP	B_VP	I_VP

all main phrase types

The	*morning*	*flight*	*from*	*Denver*	*has*	*arrived.*
B_NP	I_NP	I_NP	O	B_NP	O	O

If we're only interested in NP's

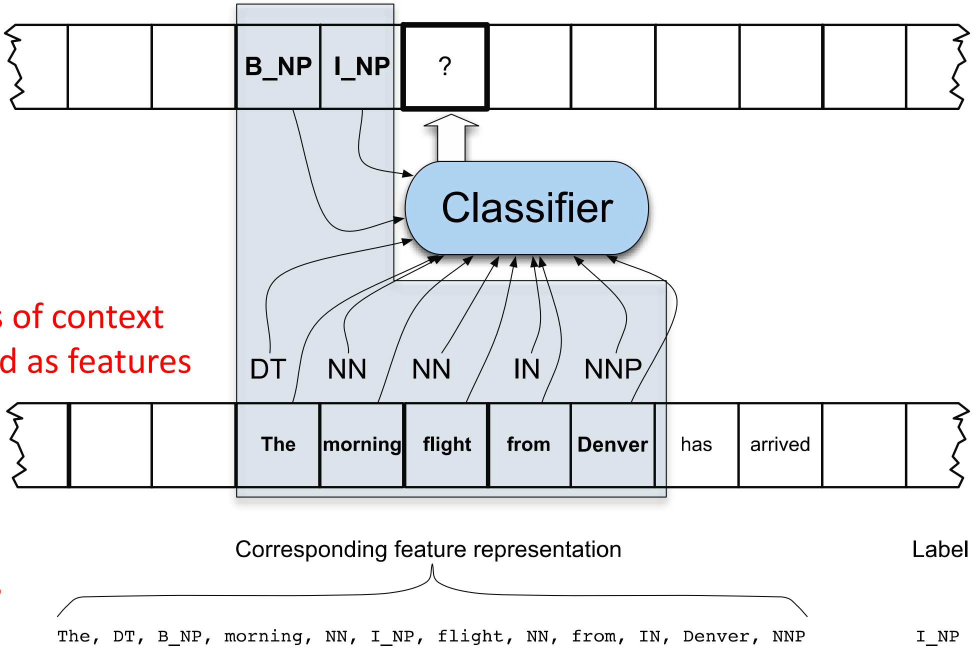
Supervised Training

- Given a training set with labeled chunks (can be derived from a Treebank), we can use any sequence classification model

previous two tags used as features

POS tags of context words used as features

Context words used as features.


B_NP
I_NP
?
Classifier
DT
NN
NN
IN
NNP
The
morning
flight
from
Denver
has
arrived
Corresponding feature representation
Label
The, DT, B_NP, morning, NN, I_NP, flight, NN, from, IN, Denver, NNP
I_NP

Evaluation of Chunking Systems

- Since chunking models identify spans of text with labels, word level accuracy is not a suitable measure.
  - Number of identified chunks may vary …

- **Precision** and **Recall** are used to evaluate the system

$$\textbf{Precision:} = \frac{\text{Number of correct chunks given by system}}{\text{Total number of chunks given by system}}$$

$$\textbf{Recall:} = \frac{\text{Number of correct chunks given by system}}{\text{Total number of actual chunks in the text}}$$

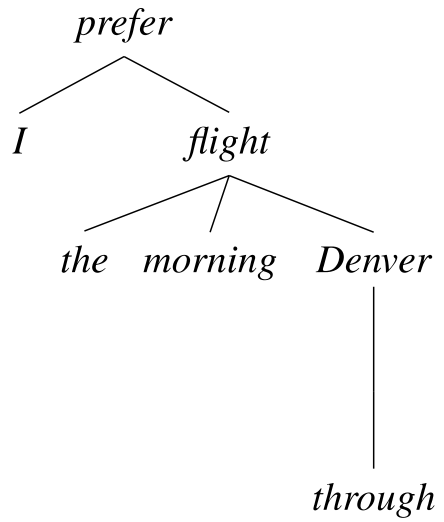
Supervised Learning – Classical Approach

- Identify the problem (e.g. chunking, POS tagging, … etc)
- Do we have human annotated data?
- Cleaning and Preprocessing (e.g. remove all e-mails or noisy characters, tokenization, lemmatization, …. etc)
- Input and output representation – feature extraction and output representation
- What type of algorithms work best?
- How do we evaluate? Intrinsic or extrinsic evaluation

Dependency Parsing

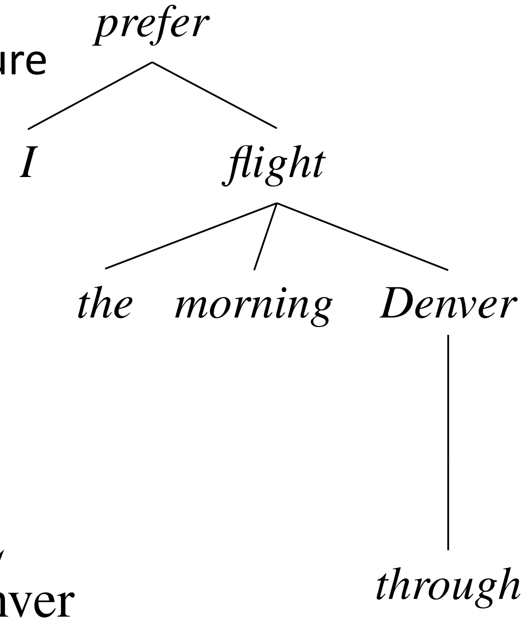
Dependency Grammar

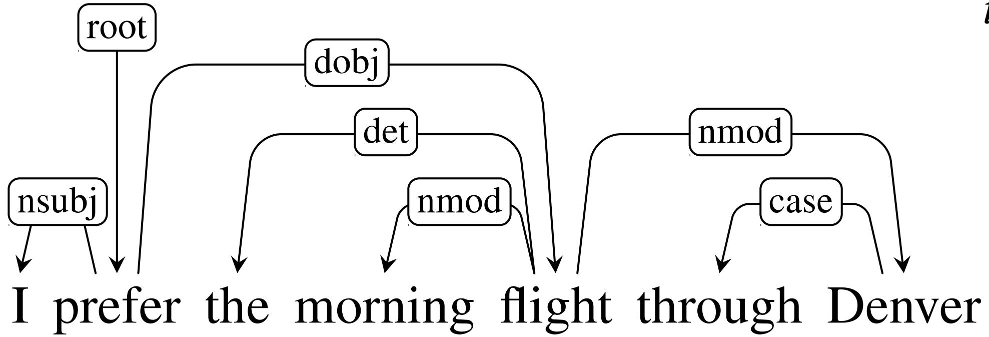
- Important in contemporary speech and language processing systems
- Consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies
    - The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



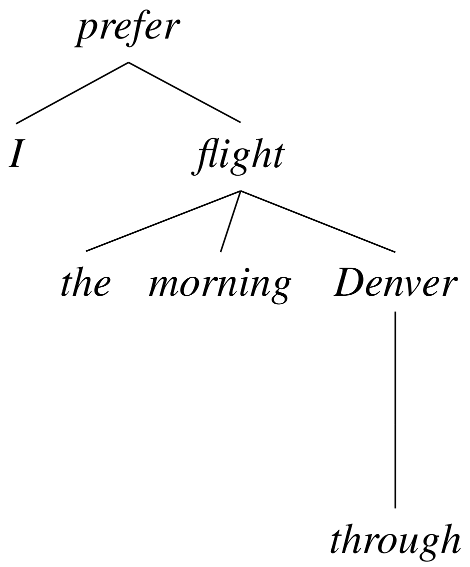
Typed Dependency Structure

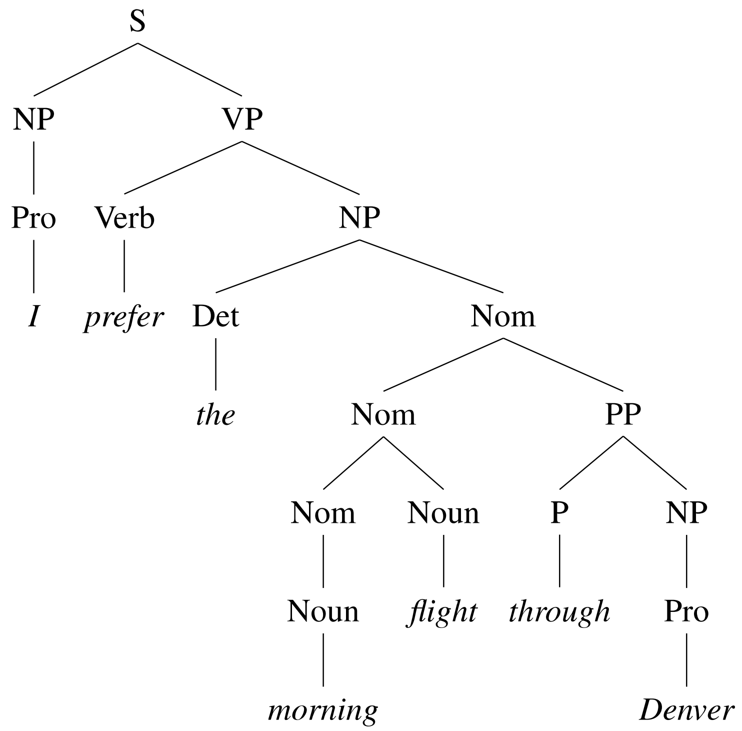
- Directed labeled arcs from heads to dependents
- Express binary grammatical relations between words
- Includes a **root** node, which is the head of the entire structure





Dependency vs. Phrase Structure





Advantages of Dependency Structure

- Encode important semantic relationships that are often buried in complex phrase structures

- Particularly useful for languages with rich morphology and relatively free word order.
  - Phrase structure grammars can get very complex for some languages

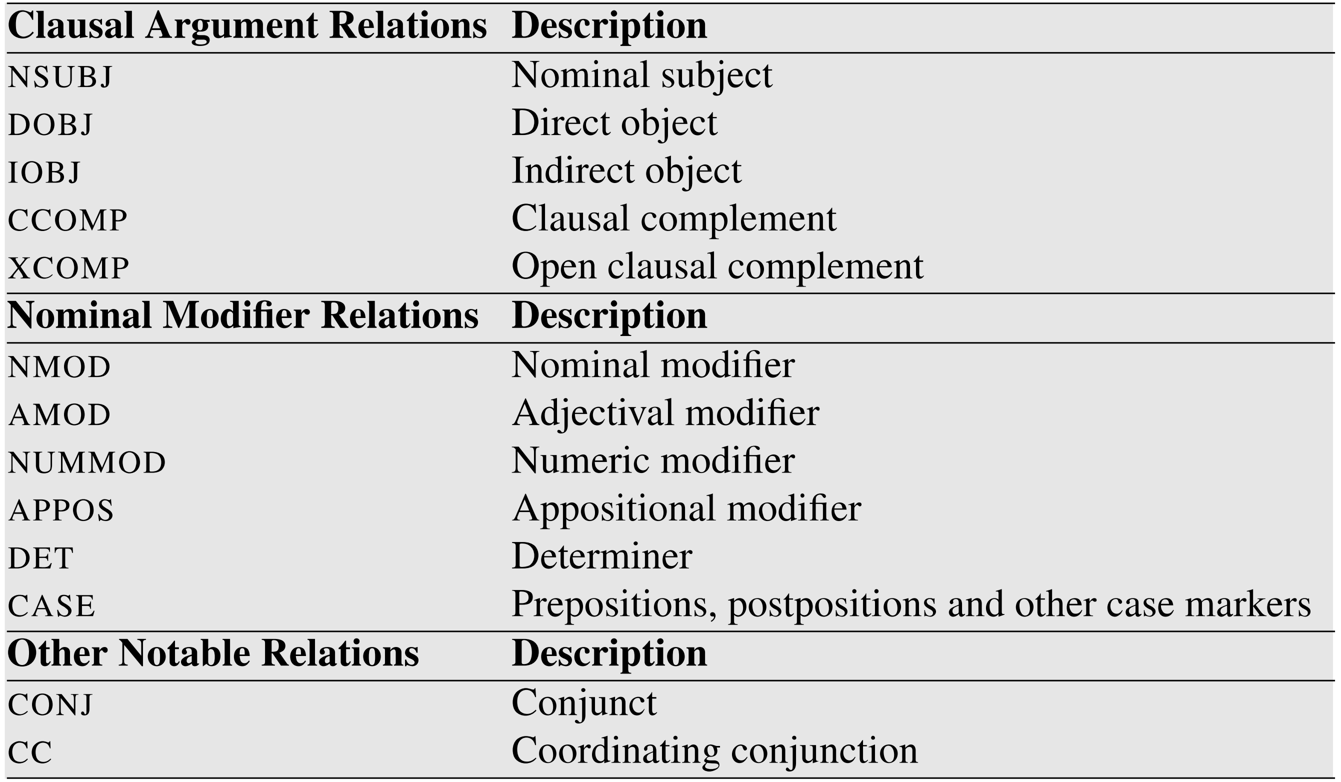
Dependency Relations

- Dependency structures are comprised of **binary** relations based on traditional grammatical relations
  - Subject, direct object, indirect object, etc.

- The arguments of these relations consist of a **head** and a **dependent**

- The **Universal Dependencies** project includes dependency relations that are applicable across languages

Examples of Universal Dependency Relations



Clausal Argument Relations	**Description**
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Dependency Treebanks

- A dependency grammar has a notion of a **head**
  - Officially, CFGs don’t

- But modern linguistic theory and all modern statistical parsers (Charniak, Collins, Stanford, …) do, via hand-written phrasal “**head rules**”:
  - The head of a Noun Phrase is a noun/number/…
  - The head of a Verb Phrase is a verb/modal/….

- The head rules can be used to automatically extract a dependency parse from a CFG parse
  - At least for English. Morphologically rich languages typically require manually annotated treebanks

Formalisms

- Dependency structures are directed graphs G = (V,A)
  - A set of vertices **V**
    - V is typically the set of words in a sentence, but can also consist of stems and affixes for morphologically rich languages
  - A set of arcs **A**; ordered pairs of vertices from V (captures head dependents and grammatical function relationships between elements in V)

- A dependency Tree is a directed graph with the following constraints:
  1. There is a single **root** node with no incoming arcs
  2. All other vertices have exactly one incoming arc
  3. There is a unique path from the root to each vertex in the graph

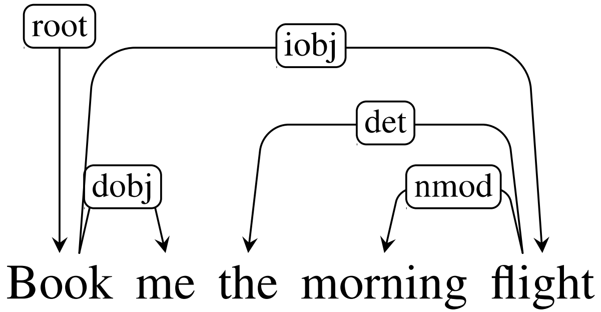
Transition-Based Parsing

- A stack-based parsing approach

- **Configurations** consist of:
  - A stack
  - Input buffer of words
  - A set of relations representing a dependency tree

- The goal of parsing is to find a final configuration where all input words have been accounted for and a dependency tree has been created

Example



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Transition-Based Parsing

- **Start** with an initial configuration:
  - The stack consists of a ROOT node
  - The word list consists of all words in the input
  - An empty set of relations

- In the **final** configuration,
  - the word list should be empty
  - stack only includes the ROOT
  - The set of relations will represent the parse tree

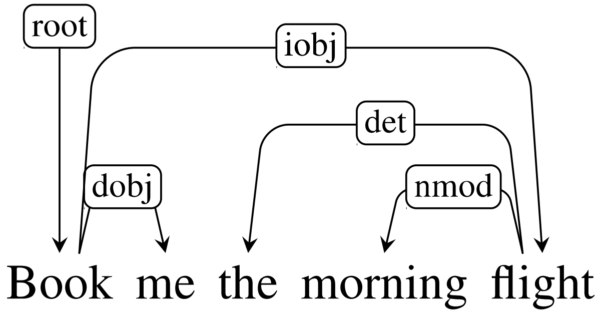
Arc-Standard Approach

- In **arc-standard** dependency parsing, one of three operations may be performed at each step:
  - **LEFTARC**: Assert a head-dependent relation between the word at the top of stack and the word directly beneath it; remove the lower word from the stack
  - **RIGHTARC**: Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack
  - **SHIFT**: Remove the word from the front of the input buffer and push it onto the stack

Arc-Standard Approach

- A straightforward greedy algorithm. Yet effective and simple to implement:
  - Words are examined in a single pass through the input from left to right
  - Transitions only assert relations between elements at the top of the stack
  - Once an element has been assigned a head, it’s removed from the stack and is no longer available for processing

Example



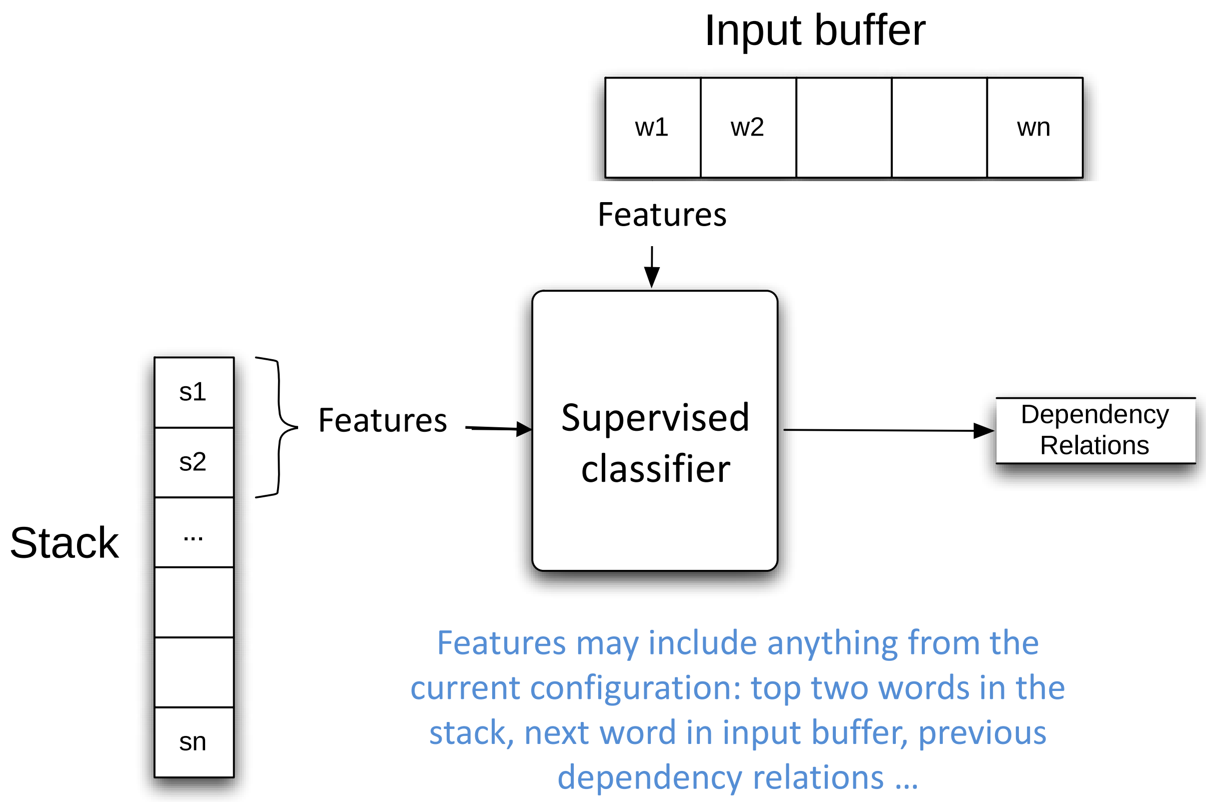
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Transition-Based Parsing

- To produce labeled arcs, we can expand the set of transition operators to include LEFTARC and RIGHTARC for each type of dependency relation.
  - LEFTARC_NSUBJ, RIGHTARC_NSUBJ, LEFTARC_DOBJ, …

- Selecting the transition operator is typically done with **supervised machine learning** methods
  - Train a classifier that predicts the correct transition given the current configuration

Transition-Based Parsing



Features may include anything from the current configuration: top two words in the stack, next word in input buffer, previous dependency relations …

Evaluation

- Unlabled attachment accuracy → percentage of words that are assigned the correct head
- Labeled attachment accuracy → percentage of words that are assigned the correct head with the correct label
- Label accuracy → percentage of words with correct labels (regardless of head)
- Precision and recall for each relation type

Resources

- The Stanford Parser:
  - Java implementation of lexicalized PCFG Constituency and dependency parsers for : English, German, French, Arabic, Chinese, and Spanish.

https://nlp.stanford.edu/software/lex-parser.html