Chart Parsing: the CYK Algorithm

Informatics 2A: Lecture 18

John Longley

30 October 2011

Grammar Restructuring

Deterministic parsing (e.g., LL(1)) aims to address a limited amount of local ambiguity – the problem of not being able to decide uniquely which grammar rule to use next in a left-to-right analysis of the input string.

By re-structuring the grammar, the parser can make a unique decision, based on a limited amount of look-ahead.

Recursive Descent parsing demands grammar restructuring, in order to eliminate left-recursive rules that can get it into a hopeless loop.

Left Recursion

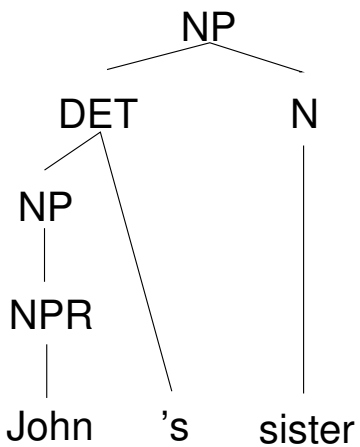
But grammars for natural human languages should be revealing, re-structuring the grammar may destroy this. (Indirectly) left-recursive rules are needed in English.

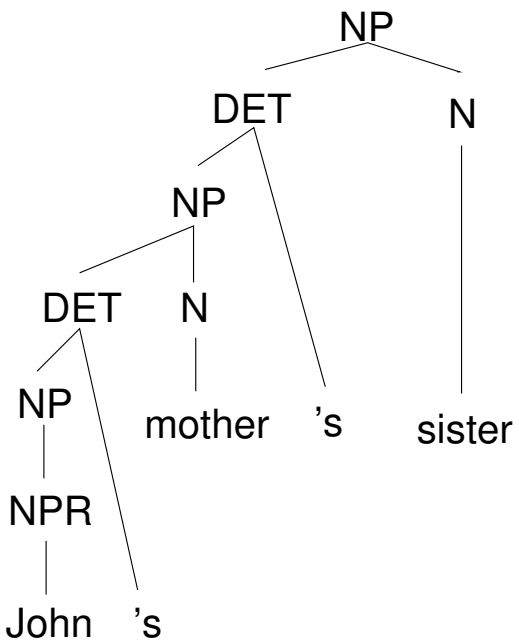
$NP \rightarrow DET\ N$
$NP \rightarrow NPR$
$DET \rightarrow NP\ 's$

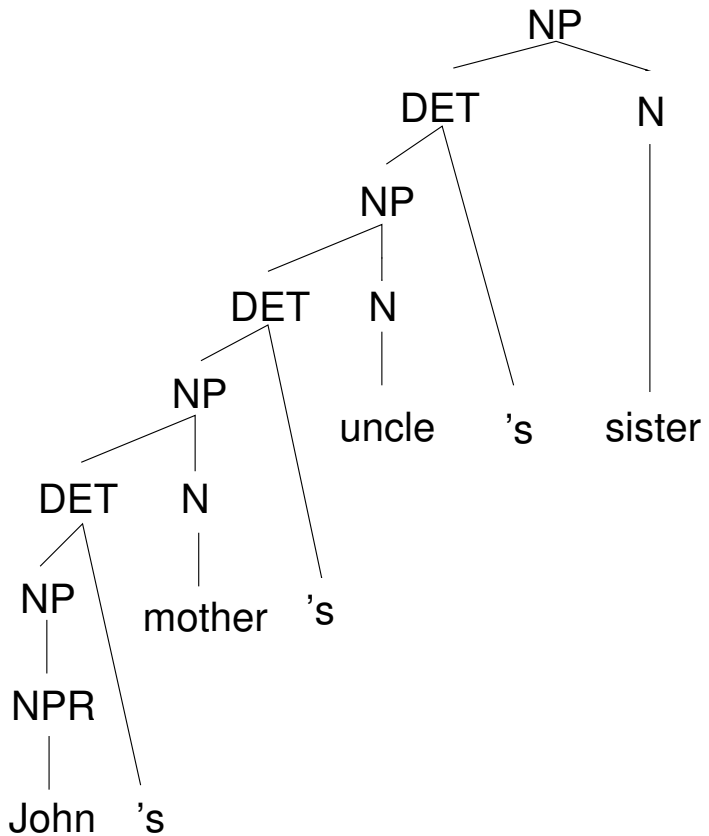
These rules generate NPs with possesive modifiers such as:

John's sister
John's mother's sister
John's mother's uncle's sister
John's mother's uncle's sister's niece

Left Recursion







We don't want to re-structure our grammar rules just to be able to use a particular approach to parsing. Need an alternative.

Problems with Parsing as Search

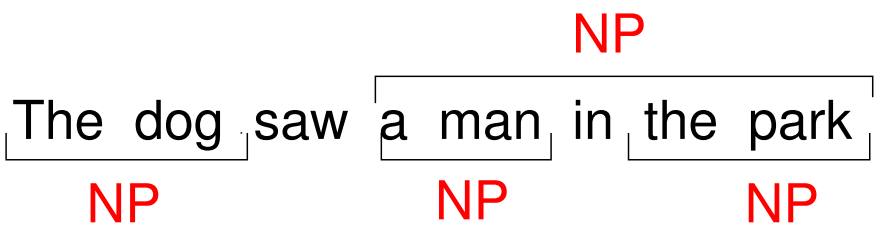
1. A **top-down parser** will do badly if there are many different rules for the same LHS; hopeless for rewriting parts of speech (preterminals) with words (terminals).
2. **A bottom-up parser** does a lot of useless work: locally possible, but globally impossible; inefficient when there is great lexical ambiguity.
3. Both strategies do repeated work by re-analyzing the same sub-string many times!

The next lectures will look at other ways of handling ambiguity:

- Chart parsing: using the parser alone;
- Probabilistic Grammars: using both grammar and parser.

Dynamic Programming

With a CFG, a parser should be able to avoid re-analyzing sub-strings because the analysis of any sub-string is independent of the rest of the parse.



The parser's exploration of its search space can exploit this independence if the parser uses dynamic programming.

Dynamic programming is the basis for all chart parsing algorithms.

Parsing as Dynamic Programming

- Given a problem, systematically fill a table of solutions to sub-problems: this is called memoization.
- Once solutions to all sub-problems have been accumulated, solve the overall problem by composing them.
- For parsing, the sub-problems are analyses of sub-strings and correspond to constituents that have been found.
- Sub-trees are stored in a chart (aka well-formed substring table), which is a record of all the substructures that have ever been built during the parse.

Solves **re-parsing problem**: sub-trees are looked up, not re-parsed!
Solves **ambiguity problem**: chart implicitly stores all parses!

Depicting a Chart

A chart can be depicted as a matrix:

- Rows and columns of the matrix correspond to the start and end positions of a span (ie, starting right before the first word, ending right after the final one);
- A cell in the matrix corresponds to the sub-string that starts at the row index and ends at the column index.
- It can contain information about the type of constituent (or constituents) that span(s) the substring, pointers to its sub-constituents, and/or predictions about what constituents might follow the substring.

CYK Algorithm

CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.

- Assumes that the grammar is in Chomsky Normal Form: rules all have form $A \rightarrow BC$ or $A \rightarrow w$.
- Conversion to CNF can be done automatically.

1. *INF-VP* $\rightarrow$ *to VP*
2. *NP* $\rightarrow$ *Pronoun*
3. $S \rightarrow$ *Aux NP VP*

CYK Algorithm

CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.

- Assumes that the grammar is in Chomsky Normal Form: rules all have form $A \rightarrow BC$ or $A \rightarrow w$.
- Conversion to CNF can be done automatically.

1. *INF-VP* $\rightarrow$ *to VP* *INF-VP* $\rightarrow$ *TO VP*
   TO $\rightarrow$ *to*
2. *NP* $\rightarrow$ *Pronoun*
3. *S* $\rightarrow$ *Aux NP VP*

CYK Algorithm

CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.

- Assumes that the grammar is in Chomsky Normal Form: rules all have form $A \rightarrow BC$ or $A \rightarrow w$.
- Conversion to CNF can be done automatically.

1.	$INF\text{-}VP \rightarrow$ *to VP*	$INF\text{-}VP \rightarrow TO\ VP$ $TO \rightarrow to$
2.	$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
3.	$S \rightarrow Aux\ NP\ VP$	

CYK Algorithm

CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.

- Assumes that the grammar is in Chomsky Normal Form: rules all have form $A \rightarrow BC$ or $A \rightarrow w$.
- Conversion to CNF can be done automatically.

1.	$INF\text{-}VP \rightarrow \ to\ VP$	$INF\text{-}VP \rightarrow TO\ VP$ $TO \rightarrow to$
2.	$NP \rightarrow Pronoun$	$NP \rightarrow I \vert she \vert me$
3.	$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$ $X1 \rightarrow \ Aux\ NP$

CYK: an example

Let's look at a simple example before we explain the general case.

Grammar Rules in CNF

NP	$\rightarrow$	Det Nom
Nom	$\rightarrow$	*book* | *orange* | AP Nom
AP	$\rightarrow$	*heavy* | *orange* | AdvD A
A	$\rightarrow$	*heavy* | *orange*
Det	$\rightarrow$	*a*
AdvD	$\rightarrow$	*very*

(N.B. Converting to CNF sometimes breeds duplication!)

Now let's parse: *a very heavy orange book*

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a					
1	very					
2	heavy					
3	orange					
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very					
2	heavy					
3	orange					
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD			
2	heavy					
3	orange					
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD			
2	heavy			A,AP		
3	orange					
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD	AP		
2	heavy			A,AP		
3	orange					
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD	AP		
2	heavy			A,AP		
3	orange				Nom,A,AP	
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD	AP		
2	heavy			A,AP	Nom	
3	orange				Nom,A,AP	
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det				
1	very		AdvD	AP	Nom	
2	heavy			A,AP	Nom	
3	orange				Nom,A,AP	
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	
1	very		AdvD	AP	Nom	
2	heavy			A,AP	Nom	
3	orange				Nom,A,AP	
4	book					

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	
1	very		AdvD	AP	Nom	
2	heavy			A,AP	Nom	
3	orange				Nom,A,AP	
4	book					Nom

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	
1	very		AdvD	AP	Nom	
2	heavy			A,AP	Nom	
3	orange				Nom,A,AP	Nom
4	book					Nom

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	
1	very		AdvD	AP	Nom	
2	heavy			A,AP	Nom	Nom
3	orange				Nom,A,AP	Nom
4	book					Nom

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	
1	very		AdvD	AP	Nom	Nom
2	heavy			A,AP	Nom	Nom
3	orange				Nom,A,AP	Nom
4	book					Nom

Filling out the CYK chart

$_0$ a $_1$ very $_2$ heavy $_3$ orange $_4$ book $_5$

		1 *a*	2 *very*	3 *heavy*	4 *orange*	5 *book*
0	a	Det			NP	NP
1	very		AdvD	AP	Nom	Nom
2	heavy			A,AP	Nom	Nom
3	orange				Nom,A,AP	Nom
4	book					Nom

CYK: The general algorithm

function CKY-Parse(*words*, *grammar*) **returns** *table* **for**

$j \leftarrow$**from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$**from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k]$
          $C \in table[k,j]\}$

CYK: The general algorithm

function CKY-Parse(*words*, *grammar*) **returns** *table* **for**

j ←**from** 1 **to** LENGTH(*words*) **do** — loop over the columns

  table[*j* − 1, *j*] ← {*A* | *A* → *words*[*j*] ∈ *grammar*} — fill bottom cell

  for *i* ←**from** *j* − 2 **downto** 0 **do** — fill row *i* in column *j*

    for *k* ← *i* + 1 **to** *j* − 1 **do** — loop over split locations between *i* and *j*

      table[*i*, *j*] ← *table*[*i*, *j*] ∪
        {*A* | *A* → *BC* ∈ *grammar*,
          B ∈ *table*[*i*, *k*]
          C ∈ *table*[*k*, *j*]}

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

From CYK Recognizer to CYK Parser

- So far, we just have a chart recognizer, a way of determining whether a string belongs to the given language.
- Changing this to a parser requires recording which existing constituents were combined to make each new constituent.
- This requires another field to record the one or more ways in which a constituent spanning (i,j) can be made from constituents spanning (i,k) and (k,j). (More clearly displayed in graph representation, see next lecture.)
- In any case, for a fixed grammar, the CYK algorithm runs in time $O(n^3)$ on an input string of n tokens.
- The algorithm identifies all possible parses.
- Can also draw CYK-style charts for non-CNF grammars. But runtime will in general be worse than $O(n^3)$ in this case.

Second example

Grammar Rules in CNF

$S \rightarrow NP\ VP$	$Nominal \rightarrow book \mid flight \mid money$
$S \rightarrow X1\ VP$	$Nominal \rightarrow Nominal\ noun$
$X1 \rightarrow Aux\ VP$	$Nominal \rightarrow Nominal\ PP$
$S \rightarrow book \mid include \mid prefer$	$VP \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$	$VPVerb \rightarrow NP$
$S \rightarrow X2$	$VP \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$	$X2 \rightarrow Verb\ NP$
$S \rightarrow VP\ PP$	$VP \rightarrow Verb\ NP$
$NP \rightarrow TWA \mid Houston$	$VP \rightarrow VP\ PP$
$NP \rightarrow Det\ Nominal$	$PP \rightarrow Preposition\ NP$
$Verb \rightarrow book \mid include \mid prefer$	$Noun \rightarrow book \mid flight \mid money$

Let's parse *Book the flight through Houston*!

Second example

Grammar Rules in CNF

$S \rightarrow NP\ VP$	$Nominal \rightarrow book \mid flight \mid money$
$S \rightarrow X1\ VP$	$Nominal \rightarrow Nominal\ noun$
$X1 \rightarrow Aux\ VP$	$Nominal \rightarrow Nominal\ PP$
$S \rightarrow book \mid include \mid prefer$	$VP \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$	$VPVerb \rightarrow NP$
$S \rightarrow X2$	$VP \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$	$X2 \rightarrow Verb\ NP$
$S \rightarrow VP\ PP$	$VP \rightarrow Verb\ NP$
$NP \rightarrow TWA \mid Houston$	$VP \rightarrow VP\ PP$
$NP \rightarrow Det\ Nominal$	$PP \rightarrow Preposition\ NP$
$Verb \rightarrow book \mid include \mid prefer$	$Noun \rightarrow book \mid flight \mid money$

Let's parse *Book the flight through Houston*!

Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]				

Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]				
	Det [1, 2]			

Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]				
	Det [1, 2]			
		Nominal, Noun [2, 3]		

Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]				
	Det [1, 2]			
		Nominal, Noun [2, 3]		
			Prep [3, 4]	

Second example

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

Second example

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

Second example

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

Second example

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

Second example

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

Second example

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

Second example

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

Second example

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

Second example

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

Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	[0, 2]	S, VP, X2 [0, 3]	[0, 4]	
	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]

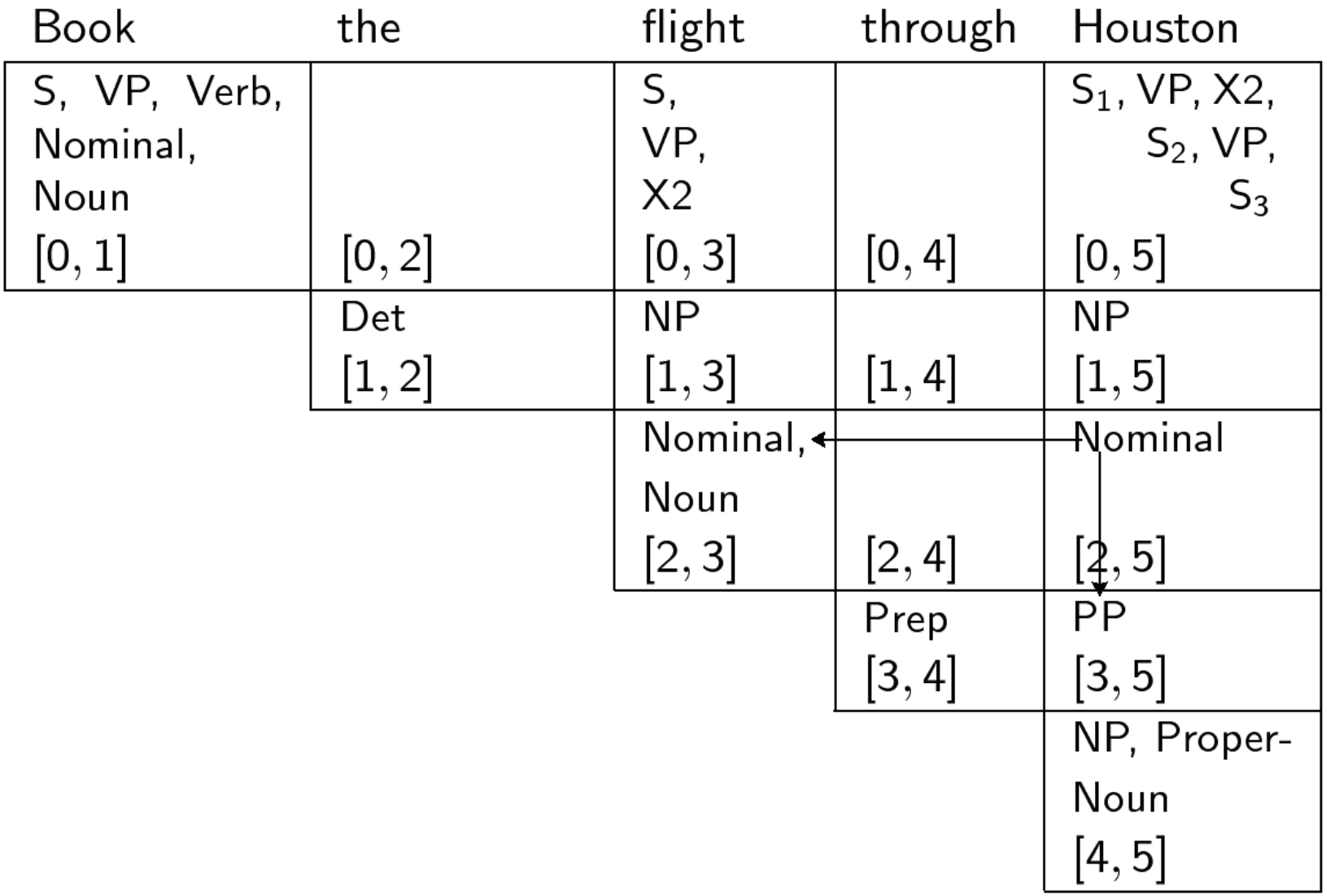
Second example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	 [0, 2]	S, VP, X2 [0, 3]	 [0, 4]	S_1, VP, X2, S_2, VP, S_3 [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]

Visualizing the Chart

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0, 1]	 [0, 2]	S, VP, X2 [0, 3]	 [0, 4]	S_1, VP, X2, S_2, VP, S_3 [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]

Visualizing the Chart



Summary

- Parsing as search is inefficient (typically exponential time).
- Alternative: use dynamic programming and memoize sub-analysis in a chart to avoid duplicate work.
- The chart can be visualized as as a matrix.
- The CYK algorithm builds a chart in $O(n^3)$ time. The basic version gives just a recognizer, but it can be made into a parser if more info is recorded in the chart.

Reading: J&M (2nd ed), Chapter. 13, Sections 13.3–13.4
NLTK Book, Chapter. 8 (*Analyzing Sentence Structure*), Section 8.4

Next lecture: the Earley parser or dynamic programming for top-down parsing