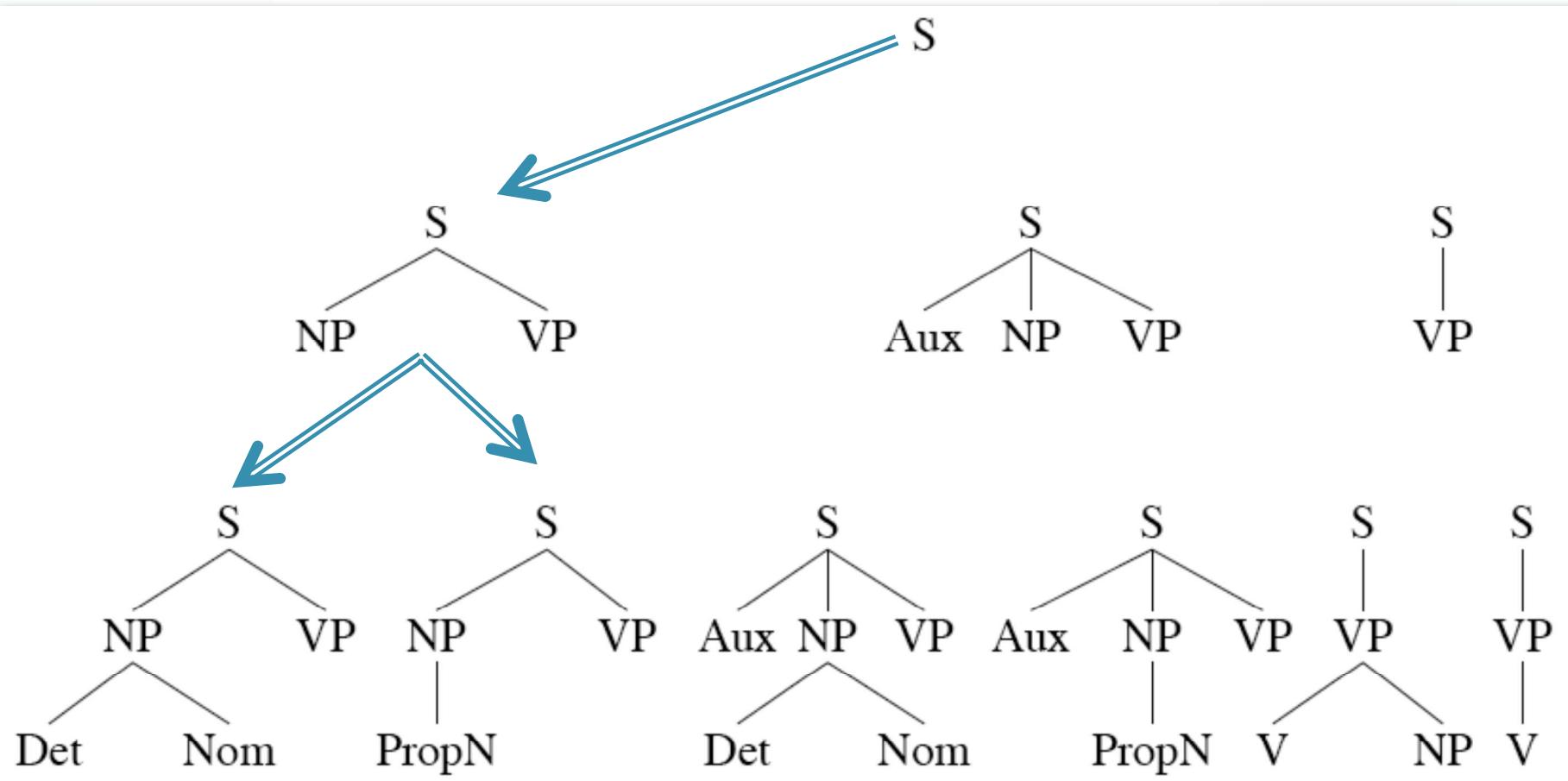


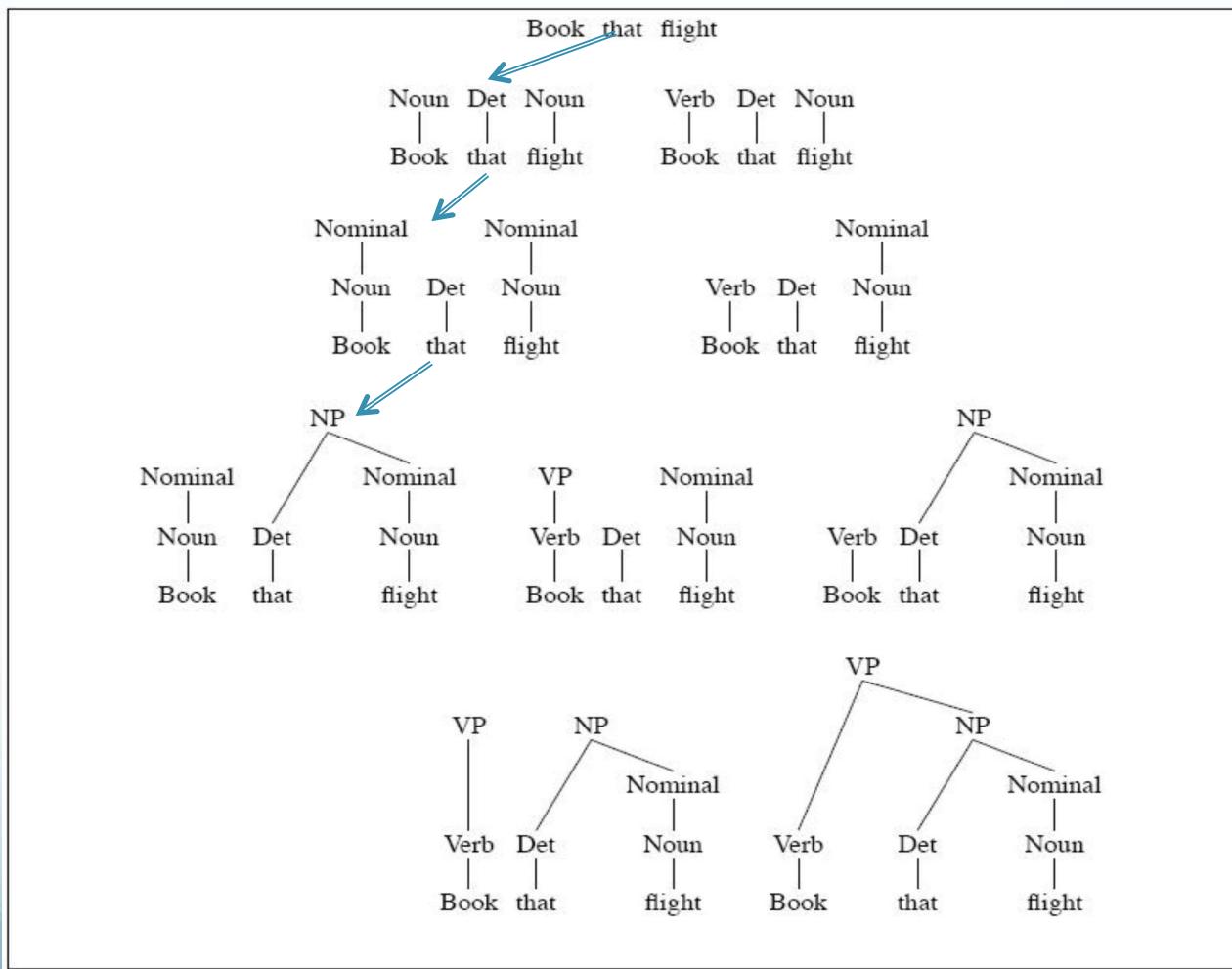
# Roadmap

- Motivation:
  - Inefficiencies of parsing-as-search
- Strategy: Dynamic Programming
- Chomsky Normal Form
  - Weak and strong equivalence
- CKY parsing algorithm

# Top-down parsing (DFS)



# Bottom-Up Search



# Parsing Challenges

- Ambiguity
- Repeated substructure
- Recursion

# Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V

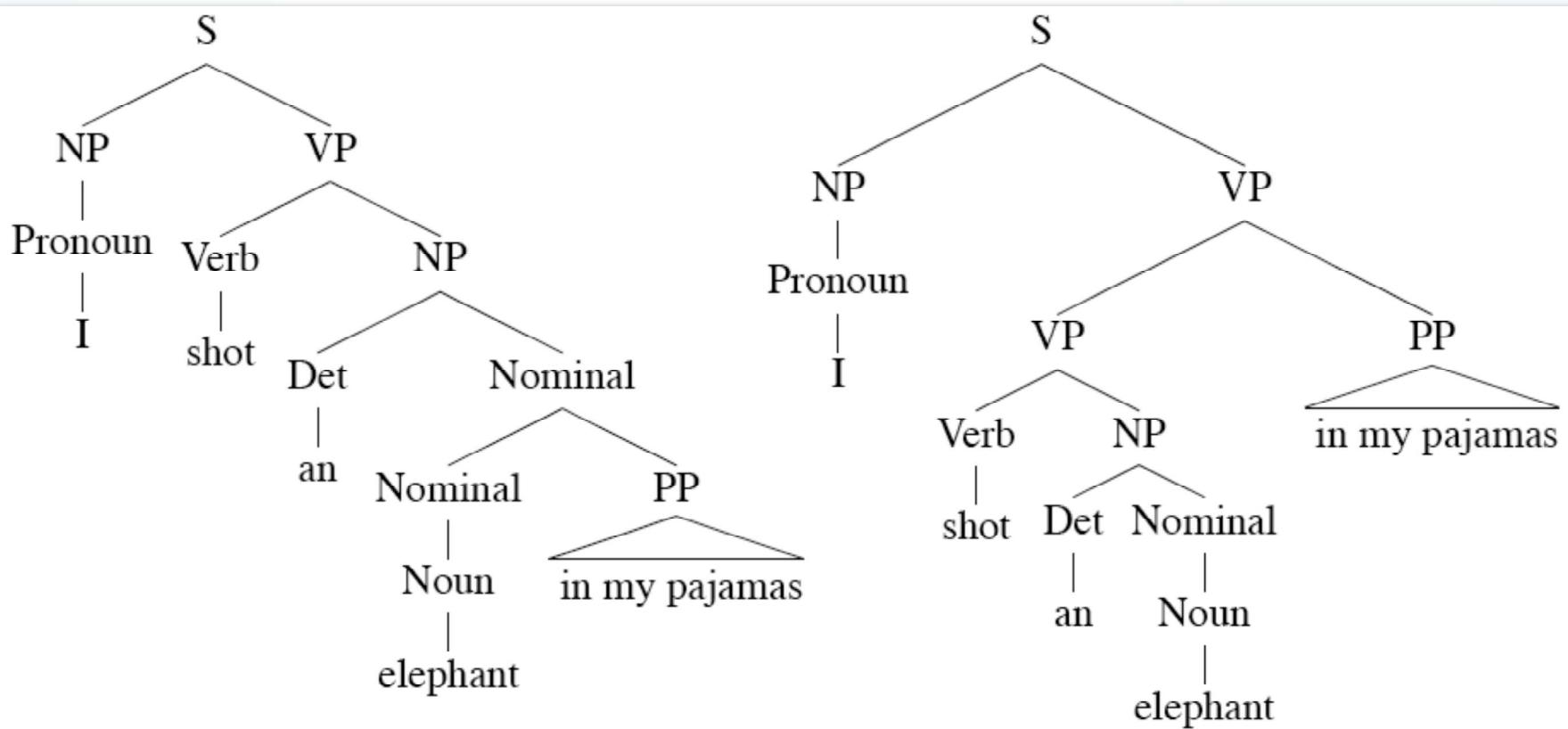
# Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
  - Structural ambiguity: Main types:
    - Attachment ambiguity
      - Constituent can attach in multiple places
        - *I shot an elephant in my pyjamas.*

# Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
  - Structural ambiguity: Main types:
    - Attachment ambiguity
      - Constituent can attach in multiple places
        - *I shot an elephant in my pyjamas.*
    - Coordination ambiguity
      - Different constituents can be conjoined
        - *Old men and women*

# Ambiguity



# Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information

# Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))

# Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic

# Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic
      - Pragmatic
        - E.g., elephants and pyjamas
    - Alternatively, keep all

# Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic
      - Pragmatic
        - E.g., elephants and pyjamas
    - Alternatively, keep all
  - Local ambiguity:
    - Ambiguity in subtree, resolved globally

# Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch
  - No static backtracking strategy can avoid

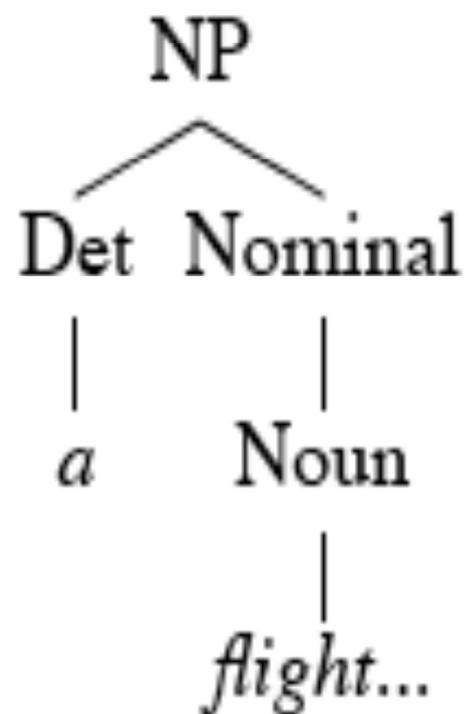
# Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch
  - No static backtracking strategy can avoid
- Efficient parsing techniques require storage of shared substructure
  - Typically with dynamic programming

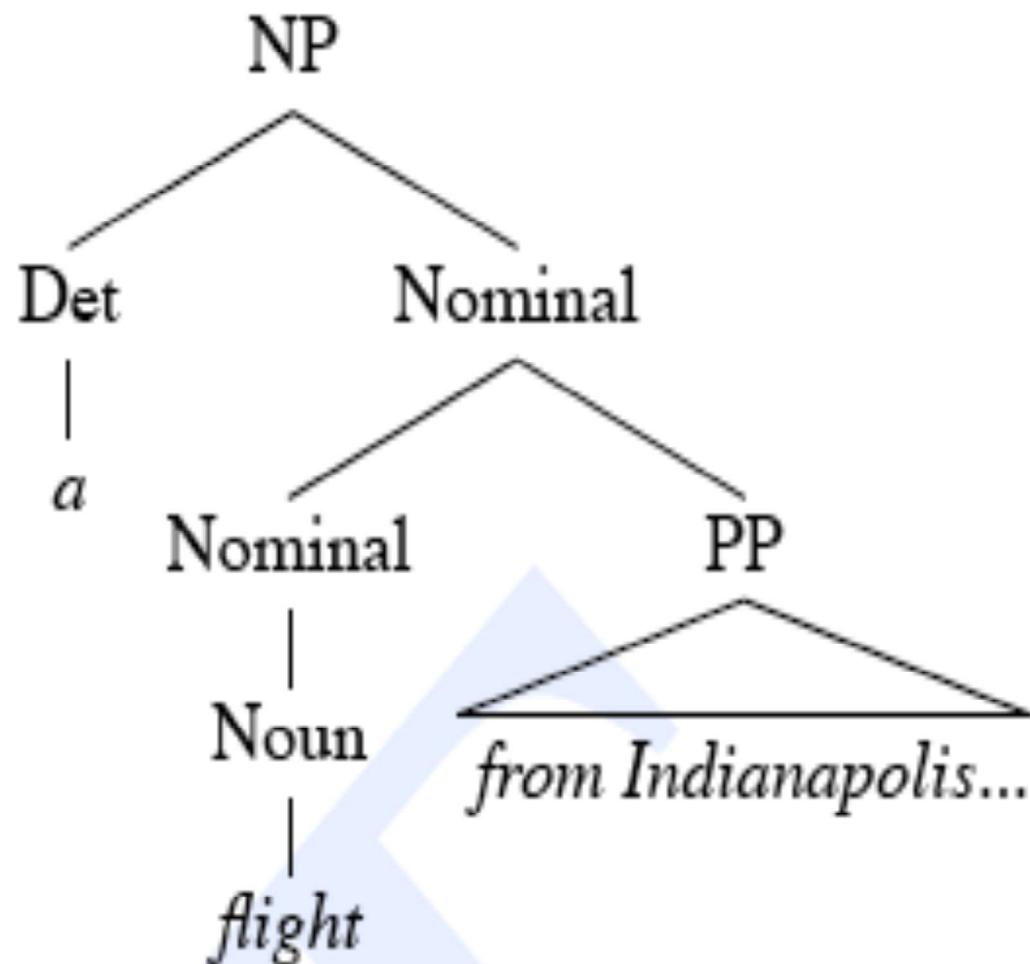
# Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch
  - No static backtracking strategy can avoid
- Efficient parsing techniques require storage of shared substructure
  - Typically with dynamic programming
- Example: *a flight from Indianapolis to Houston on TWA*

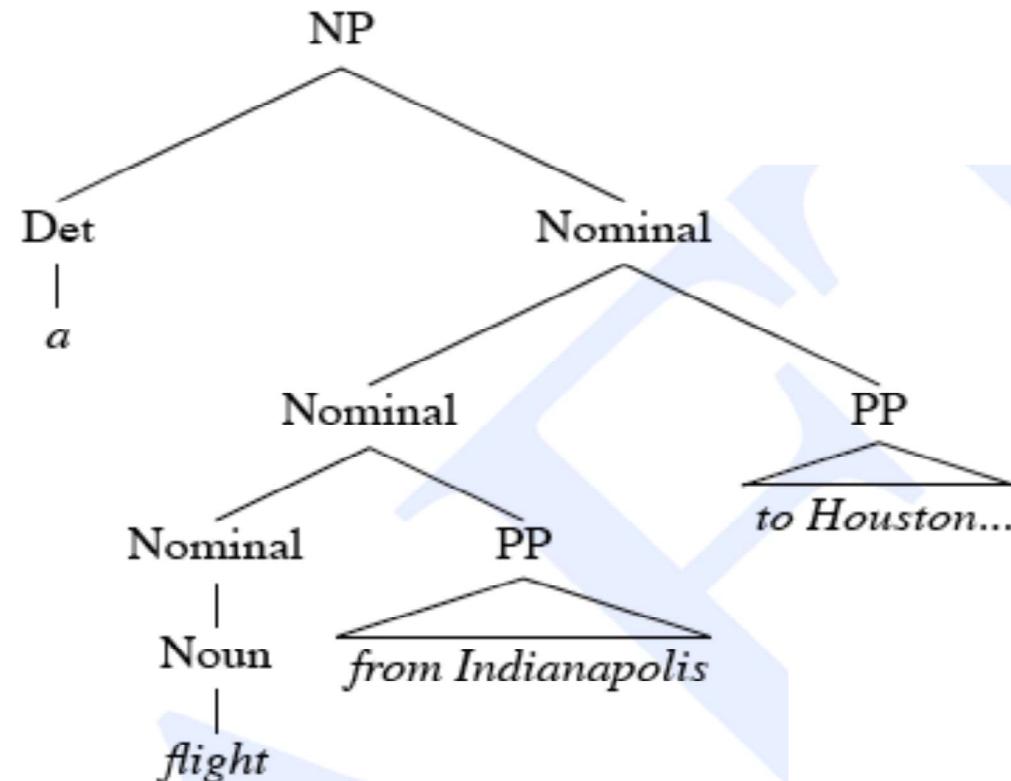
# Shared Sub-Problems



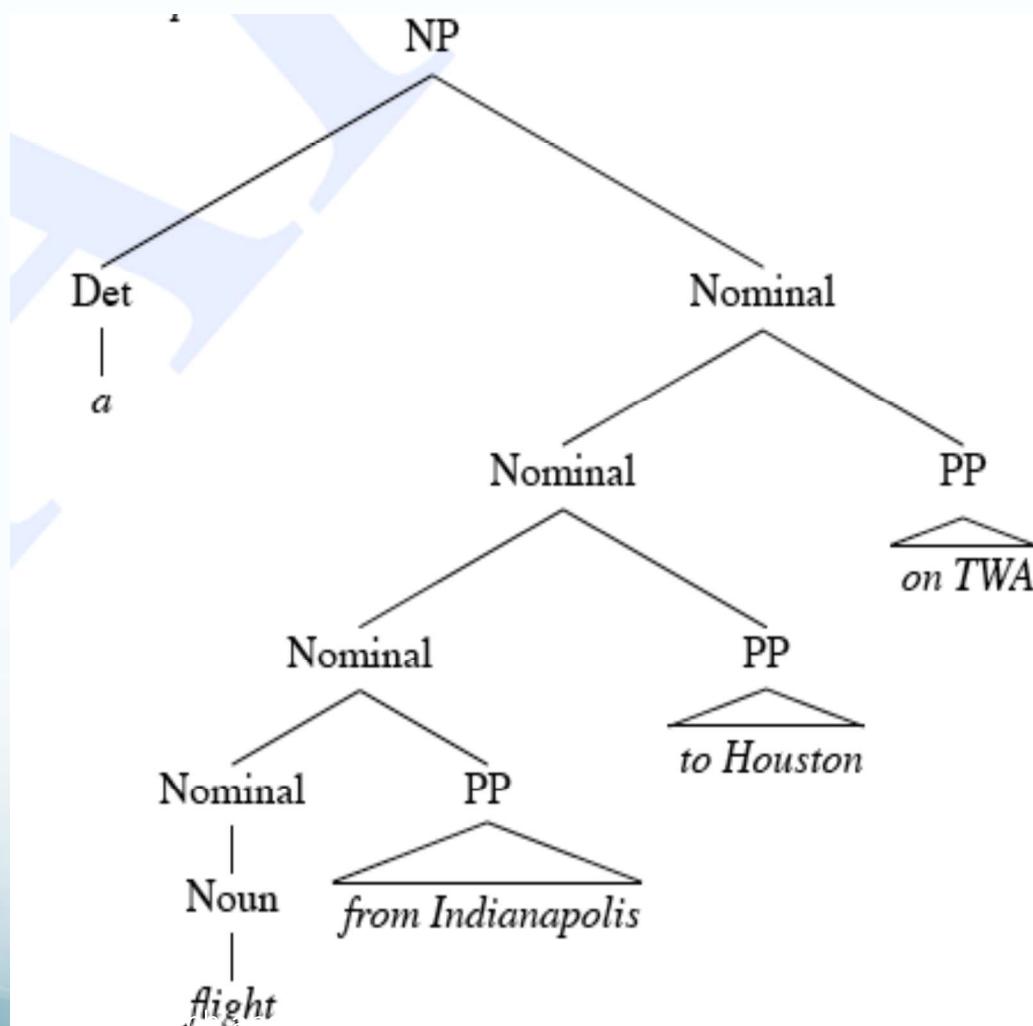
# Shared Sub-Problems



# Shared Sub-Problems



# Shared Sub-Problems



# Recursion

- Many grammars have recursive rules
  - E.g.,  $S \rightarrow S \text{ Conj } S$
- In search approaches, recursion is problematic
  - Can yield infinite searches
    - Esp., top-down

# Dynamic Programming

- Challenge: Repeated substructure → Repeated work

# Dynamic Programming

- Challenge: Repeated substructure → Repeated work
- Insight:
  - Global parse composed of parse substructures
  - Can record parses of substructures

# Dynamic Programming

- Challenge: Repeated substructure → Repeated work
- Insight:
  - Global parse composed of parse substructures
  - Can record parses of substructures
- Dynamic programming avoids repeated work by tabulating solutions to subproblems
  - Here, stores subtrees

# Parsing w/Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
  - Polynomial time in input length
    - Typically cubic ( $n^3$ ) or less

# Parsing w/Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
  - Polynomial time in input length
    - Typically cubic ( $n^3$ ) or less
- Several different implementations
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley algorithm
  - Chart parsing

# Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - $A \rightarrow B C$ , or
    - $A \rightarrow a$

# Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - $A \rightarrow B C$ , or
    - $A \rightarrow a$
- However, most of our grammars are not of this form
  - E.g.,  $S \rightarrow \text{Wh-NP Aux NP VP}$

# Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - $A \rightarrow B C$ , or
    - $A \rightarrow a$
- However, most of our grammars are not of this form
  - E.g.,  $S \rightarrow Wh\text{-}NP \text{ Aux NP VP}$
- Need a general conversion procedure
  - Any arbitrary grammar can be converted to CNF

# CNF Conversion

- Three main conditions:

# CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP → to VP

# CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP → to VP
  - Unit productions:
    - A → B

# CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP → to VP
  - Unit productions:
    - $A \rightarrow B$
  - Long productions:
    - $A \rightarrow B C D$

# CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP → to VP

# CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP → to VP
    - INF-VP → TO VP; TO → to

# CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP → to VP
    - INF-VP → TO VP; TO → to
- Unit productions:
  - Rewrite RHS with RHS of all derivable non-unit productions
    - If  $A \xrightarrow{*} B$  and  $B \rightarrow w$ , then add  $A \rightarrow w$

# CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - $S \rightarrow \text{Aux NP VP}$

# CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - $S \rightarrow \text{Aux NP VP}$
  - $S \rightarrow X_1 \text{ VP}; X_1 \rightarrow \text{Aux NP}$

# CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - $S \rightarrow \text{Aux NP VP}$
  - $S \rightarrow X_1 \text{ VP}; X_1 \rightarrow \text{Aux NP}$
- For all non-conforming rules,
  - Convert terminals to dummy non-terminals
  - Convert unit productions
  - Binarize all resulting rules

### $\mathcal{L}_1$ Grammar

$S \rightarrow NP VP$

$S \rightarrow Aux NP VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb NP$

$VP \rightarrow Verb NP PP$

$VP \rightarrow Verb PP$

$VP \rightarrow VP PP$

$PP \rightarrow Preposition NP$

### $\mathcal{L}_1$ in CNF

**$\mathcal{L}_1$  Grammar** $S \rightarrow NP VP$  $S \rightarrow Aux NP VP$  $S \rightarrow VP$  $NP \rightarrow Pronoun$  $NP \rightarrow Proper-Noun$  $NP \rightarrow Det Nominal$  $Nominal \rightarrow Noun$  $Nominal \rightarrow Nominal Noun$  $Nominal \rightarrow Nominal PP$  $VP \rightarrow Verb$  $VP \rightarrow Verb NP$  $VP \rightarrow Verb NP PP$  $VP \rightarrow Verb PP$  $VP \rightarrow VP PP$  $PP \rightarrow Preposition NP$  **$\mathcal{L}_1$  in CNF** $S \rightarrow NP VP$

**$\mathcal{L}_1$  Grammar** $S \rightarrow NP\ VP$  $S \rightarrow Aux\ NP\ VP$  $S \rightarrow VP$  $NP \rightarrow Pronoun$  $NP \rightarrow Proper-Noun$  $NP \rightarrow Det\ Nominal$  $Nominal \rightarrow Noun$  $Nominal \rightarrow Nominal\ Noun$  $Nominal \rightarrow Nominal\ PP$  $VP \rightarrow Verb$  $VP \rightarrow Verb\ NP$  $VP \rightarrow Verb\ NP\ PP$  $VP \rightarrow Verb\ PP$  $VP \rightarrow VP\ PP$  $PP \rightarrow Preposition\ NP$  **$\mathcal{L}_1$  in CNF** $S \rightarrow NP\ VP$  $S \rightarrow X1\ VP$  $X1 \rightarrow Aux\ NP$

$\mathcal{L}_1$ Grammar	$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
$S \rightarrow VP$	$X1 \rightarrow Aux\ NP$
	$S \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	
$NP \rightarrow Proper-Noun$	
$NP \rightarrow Det\ Nominal$	
$Nominal \rightarrow Noun$	
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

$\mathcal{L}_1$ Grammar	$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$ $X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$ $S \rightarrow Verb\ NP$ $S \rightarrow X2\ PP$ $S \rightarrow Verb\ PP$ $S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	
$NP \rightarrow Proper-Noun$	
$NP \rightarrow Det\ Nominal$	
$Nominal \rightarrow Noun$	
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

$\mathcal{L}_1$ Grammar	$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
$S \rightarrow VP$	$X1 \rightarrow Aux\ NP$
$NP \rightarrow Pronoun$	$S \rightarrow book \mid include \mid prefer$
$NP \rightarrow Proper-Noun$	$S \rightarrow Verb\ NP$
$NP \rightarrow Det\ Nominal$	$S \rightarrow X2\ PP$
$Nominal \rightarrow Noun$	$S \rightarrow Verb\ PP$
$Nominal \rightarrow Nominal\ Noun$	$S \rightarrow VP\ PP$
$Nominal \rightarrow Nominal\ PP$	$NP \rightarrow I \mid she \mid me$
$VP \rightarrow Verb$	$NP \rightarrow TWA \mid Houston$
$VP \rightarrow Verb\ NP$	$NP \rightarrow Det\ Nominal$
$VP \rightarrow Verb\ NP\ PP$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$VP \rightarrow Verb\ PP$	$Nominal \rightarrow Nominal\ Noun$
$VP \rightarrow VP\ PP$	$Nominal \rightarrow Nominal\ PP$
$PP \rightarrow Preposition\ NP$	$VP \rightarrow book \mid include \mid prefer$
	$VP \rightarrow Verb\ NP$
	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
	$VP \rightarrow Verb\ PP$
	$VP \rightarrow VP\ PP$
	$PP \rightarrow Preposition\ NP$

# CKY Parsing

- Cocke-Kasami-Younger parsing algorithm:
  - (Relatively) efficient bottom-up parsing algorithm based on tabulating substring parses to avoid repeated work

# CKY Parsing

- Cocke-Kasami-Younger parsing algorithm:
  - (Relatively) efficient bottom-up parsing algorithm based on tabulating substring parses to avoid repeated work
  - Approach:
    - Use a CNF grammar
    - Build an  $(n+1) \times (n+1)$  matrix to store subtrees
      - Upper triangular portion
    - Incrementally build parse spanning whole input string

# Dynamic Programming in CKY

- Key idea:
  - For a parse spanning substring  $[i,j]$  , there exists some k such there are parses spanning  $[i,k]$  and  $[k,j]$ 
    - We can construct parses for whole sentence by building up from these stored partial parses

# Dynamic Programming in CKY

- Key idea:
  - For a parse spanning substring  $[i,j]$  , there exists some k such there are parses spanning  $[i,k]$  and  $[k,j]$ 
    - We can construct parses for whole sentence by building up from these stored partial parses
- So,
  - To have a rule  $A \rightarrow B C$  in  $[i,j]$ ,
    - We must have B in  $[i,k]$  and C in  $[k,j]$ , for some  $i < k < j$ 
      - CNF grammar forces this for all  $j > i+1$

# CKY

- Given an input string S of length n,
  - Build table  $(n+1) \times (n+1)$
  - Indexes correspond to inter-word positions
    - E.g., 0 Book 1 That 2 Flight 3

# CKY

- Given an input string  $S$  of length  $n$ ,
  - Build table  $(n+1) \times (n+1)$
  - Indexes correspond to inter-word positions
    - E.g., 0 Book 1 That 2 Flight 3
- Cells  $[i,j]$  contain sets of non-terminals of ALL constituents spanning  $i,j$ 
  - $[j-1,j]$  contains pre-terminals
  - If  $[0,n]$  contains Start, the input is recognized

# CKY Algorithm

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

# Is this a parser?

# CKY Parsing

- Table fills:
  - Column-by-column
  - Left-to-right
  - Bottom-to-top
- Why?

# CKY Parsing

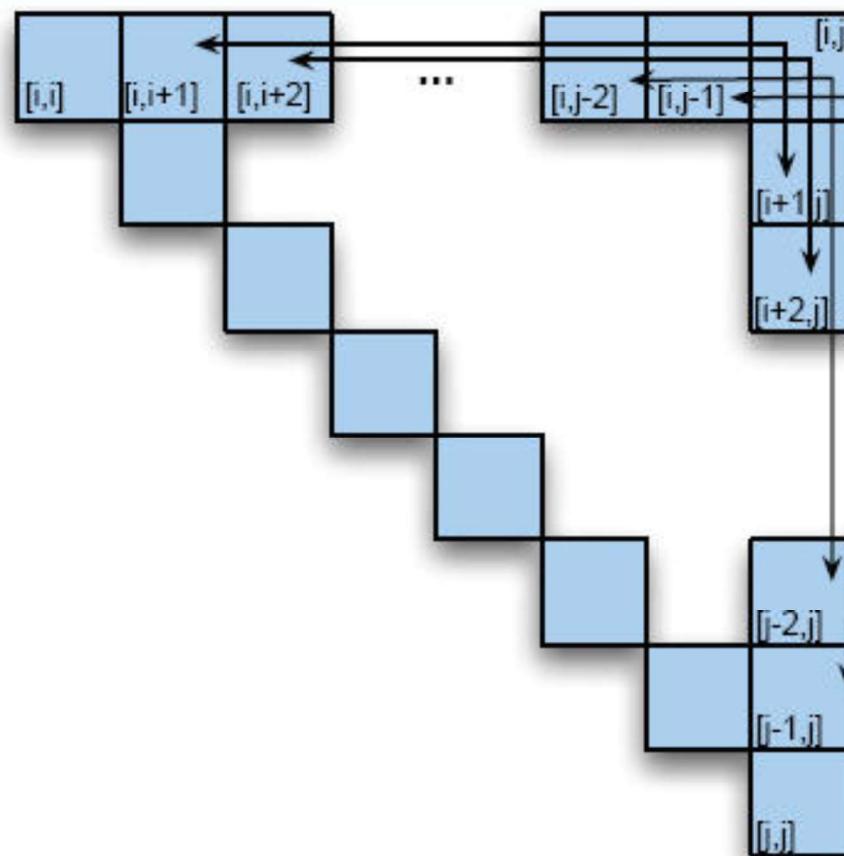
- Table fills:
  - Column-by-column
  - Left-to-right
  - Bottom-to-top
- Why?
  - Necessary info available (below and left)
  - Allows online sentence analysis
    - Works across input string as it arrives

# CKY Table

- Book the flight through Houston

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

# Filling CKY cell



0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	through	Houston
NN, VB, Nominal, VP, S [0,1]				

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	through	Houston
NN, VB, Nominal, VP, S [0,1]				
	Det [1,2]			

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	Through	Houston
NN, VB, Nominal, VP, S [0,1]	[0,2]			
	Det [1,2]			

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	through	Houston
NN, VB, Nominal, VP, S [0,1]	[0,2]			
	Det [1,2]			
		NN, Nominal [2,3]		

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	through	Houston
NN, VB, Nominal, VP, S [0,1]	[0,2]			
	Det [1,2]	NP [1,3]		
		NN, Nominal [2,3]		

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	Through	Houston
NN, VB, Nominal, VP, S [0,1]		S, VP, X2 [0,3]		
	Det [1,2]	NP [1,3]		
		NN, Nominal [2,3]		

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	Through	Houston
NN, VB, Nominal, VP, S [0,1]	[0,2]	S, VP, X2 [0,3]		
	Det [1,2]	NP [1,3]		
		NN, Nominal [2,3]		
			Prep [3,4]	

0 Book 1 the 2 flight 3 through 4 Houston 5

Book	the	Flight	Through	Houston
NN, VB, Nominal, VP, S [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	
	Det [1,2]	NP [1,3]	[1,4]	
		NN, Nominal [2,3]	[2,4]	
			Prep [3,4]	

0 Book 1 the 2 flight 3 through 4 Houston 5

<b>Book</b>	<b>the</b>	<b>Flight</b>	<b>Through</b>	<b>Houston</b>
NN, VB, Nominal, VP, S [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	
	Det [1,2]	NP [1,3]	[1,4]	
		NN, Nominal [2,3]	[2,4]	
			Prep [3,4]	
				NNP, NP [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

<b>Book</b>	<b>the</b>	<b>Flight</b>	<b>Through</b>	<b>Houston</b>
NN, VB, Nominal, VP, S [0,1]		S, VP, X2		
	[0,2]	[0,3]	[0,4]	
	Det [1,2]	NP [1,3]	[1,4]	
		NN, Nominal [2,3]	[2,4]	
			Prep [3,4]	PP [3,5]
				NNP, NP [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

<b>Book</b>	<b>the</b>	<b>Flight</b>	<b>Through</b>	<b>Houston</b>
NN, VB, Nominal, VP, S [0,1]		S, VP, X2		
	[0,2]	[0,3]	[0,4]	
	Det [1,2]	NP [1,3]	[1,4]	
		NN, Nominal [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NNP, NP [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

<b>Book</b>	<b>the</b>	<b>Flight</b>	<b>Through</b>	<b>Houston</b>
NN, VB, Nominal, VP, S [0,1]		S, VP, X2		
	[0,2]	[0,3]	[0,4]	
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		NN, Nominal [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NNP, NP [4,5]

0 Book 1 the 2 flight 3 through 4 Houston 5

<b>Book</b>	<b>the</b>	<b>Flight</b>	<b>Through</b>	<b>Houston</b>
NN, VB, Nominal, VP, S [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]
		NN, Nominal [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NNP, NP [4,5]

# From Recognition to Parsing

- Limitations of current recognition algorithm:

# From Recognition to Parsing

- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS

# From Recognition to Parsing

- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS
  - Stores SETS of non-terminals
    - Can't store multiple rules with same LHS

# From Recognition to Parsing

- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS
  - Stores SETS of non-terminals
    - Can't store multiple rules with same LHS
- Parsing solution:
  - All repeated versions of non-terminals

# From Recognition to Parsing

- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS
  - Stores SETS of non-terminals
    - Can't store multiple rules with same LHS
- Parsing solution:
  - All repeated versions of non-terminals
  - Pair each non-terminal with pointers to cells
    - Backpointers

# From Recognition to Parsing

- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS
  - Stores SETS of non-terminals
    - Can't store multiple rules with same LHS
- Parsing solution:
  - All repeated versions of non-terminals
  - Pair each non-terminal with pointers to cells
    - Backpointers
  - Last step: construct trees from back-pointers in  $[0,n]$

# Filling column 5

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
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]		Nominal [2,5]
			Prep [3,4]	[3,5]
				NP, Proper- Noun [4,5]

*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]		NP  [1,5]
		Nominal, Noun  [2,3]		
			[2,4]	[2,5]
			Prep ←	PP

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

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

*Book*      *the*      *flight*      *through*      *Houston*

S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S <sub>1</sub> , VP, X2 S <sub>2</sub> , VP S <sub>3</sub>
Det [1,2]	NP [1,3]			NP [1,5]
	Nominal, Noun [2,3]	[1,4]		Nominal [2,5]
		[2,4]	Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

# CKY Discussion

- Running time:

$$O(n^3)$$

# CKY Discussion

- Running time:
  - $O(n^3)$  where  $n$  is the length of the input string

# CKY Discussion

- Running time:
  - $O(n^3)$  where  $n$  is the length of the input string
  - Inner loop grows as square of # of non-terminals
- Expressiveness:

# CKY Discussion

- Running time:
  - $O(n^3)$  where  $n$  is the length of the input string
  - Inner loop grows as square of # of non-terminals
- Expressiveness:
  - As implemented, requires CNF
    - Weakly equivalent to original grammar
    - Doesn't capture full original structure
      - Back-conversion?

# CKY Discussion

- Running time:
  - $O(n^3)$  where  $n$  is the length of the input string
  - Inner loop grows as square of # of non-terminals
- Expressiveness:
  - As implemented, requires CNF
    - Weakly equivalent to original grammar
    - Doesn't capture full original structure
      - Back-conversion?
        - Can do binarization, terminal conversion
        - Unit non-terminals require change in CKY