CKY Parsing

Ling 571
Deep Processing Techniques for NLP
January 12, 2011

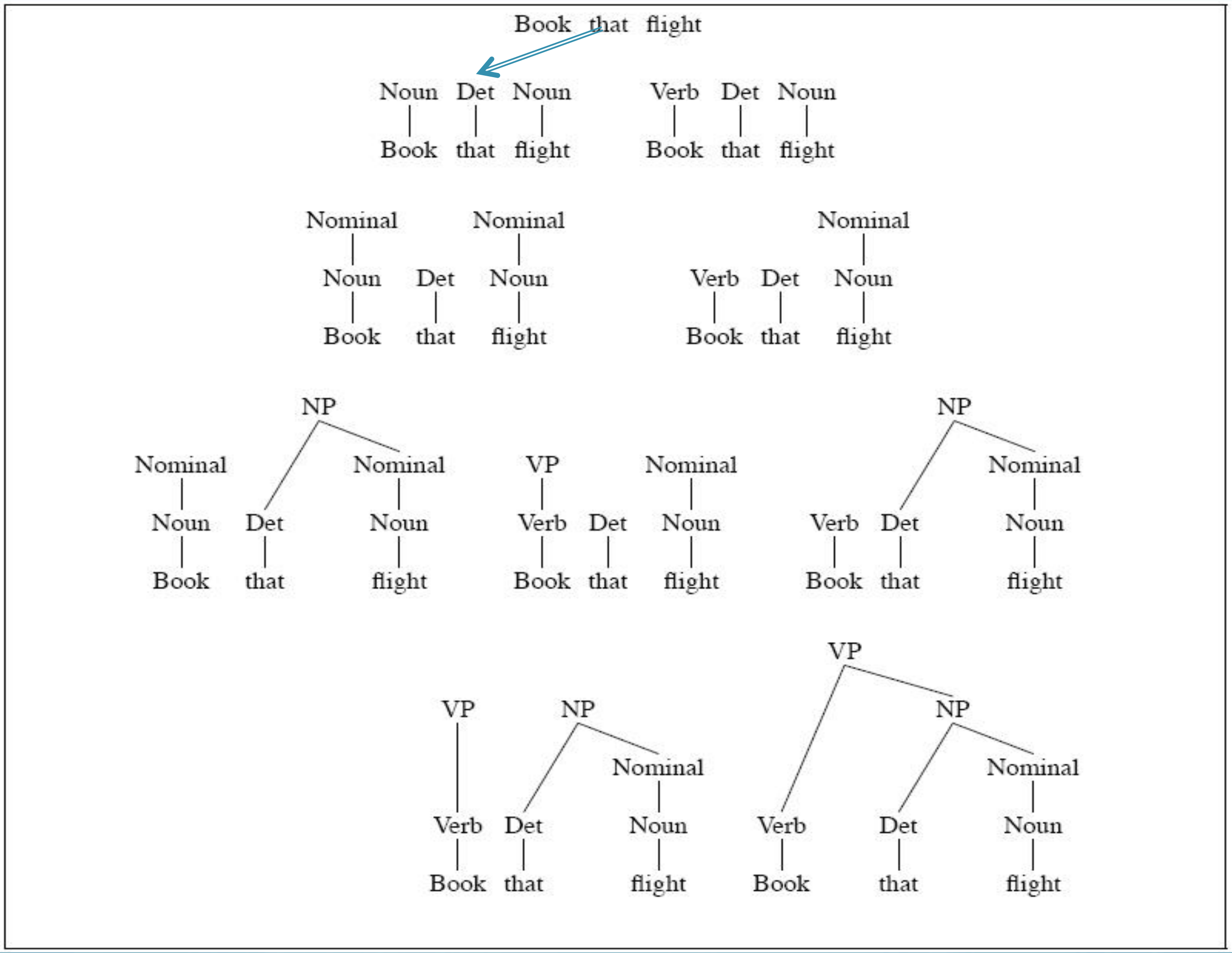
Roadmap

- Motivation:
  - Parsing (In) efficiency
- Dynamic Programming
- Cocke-Kasami-Younger Parsing Algorithm
  - Chomsky Normal Form
    - Conversion
  - CKY Algorithm
    - Parsing by tabulation

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*

Bottom-Up Search



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
- 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 -> B C, or
    - A -> a
- However, most of our grammars are not of this form
  - E.g., S -> Wh-NP Aux NP VP
- Need a general conversion procedure
  - Any arbitrary grammar can be converted to CNF

Grammatical Equivalence

- Weak equivalence:
  - Recognizes same language
  - Yields different structure
- Strong equivalence
  - Recognizes same languages
  - Yields same structure
- CNF is weakly equivalent

CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP -> to VP
  - Unit productions:
    - A -> B
  - Long productions:
    - A -> B C D

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 $A \overset{*}{\Rightarrow} B$ and B -> w, then add A -> w

CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - S -> Aux NP VP
    - S -> X1 VP; X1 -> Aux NP

- For all non-conforming rules,
  - Convert terminals to dummy non-terminals
  - Convert unit productions
  - Binarize all resulting rules

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

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

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

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

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

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

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) x (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
- So,
  - To have a rule A -> 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) x (n+1)
  - Indexes correspond to inter-word positions
    - W.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 $j \leftarrow$ **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**
            $table[i,j] \leftarrow table[i,j] \cup$
                $\{A \mid A \rightarrow BC \in grammar,$
                    $B \in table[i,k],$
                    $C \in table[k,j]\}$

- Is this a parser?

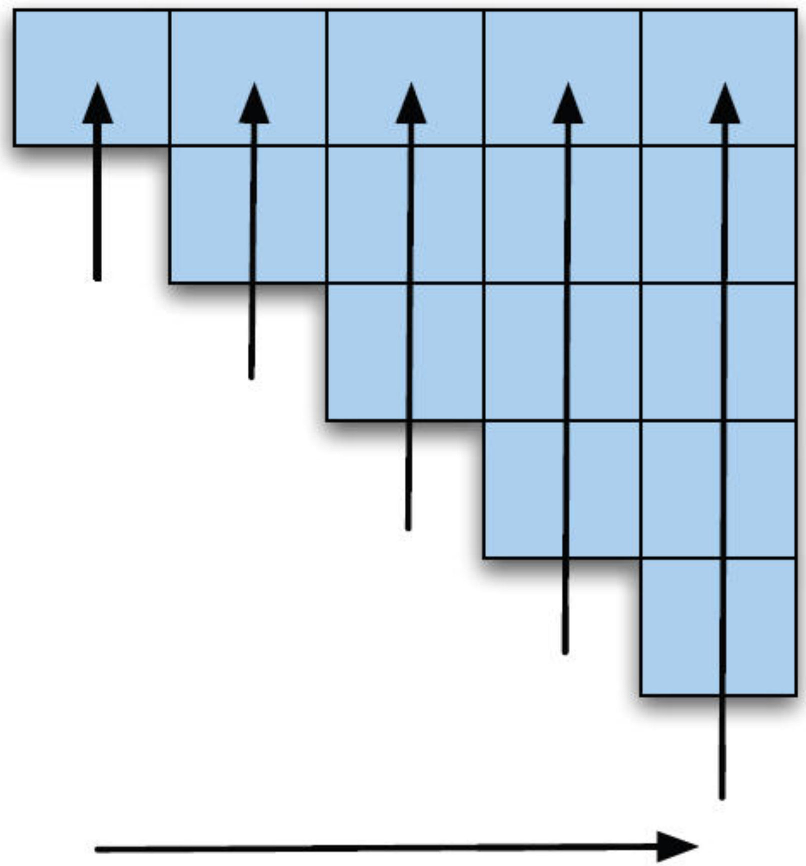
CKY Parsing

- 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

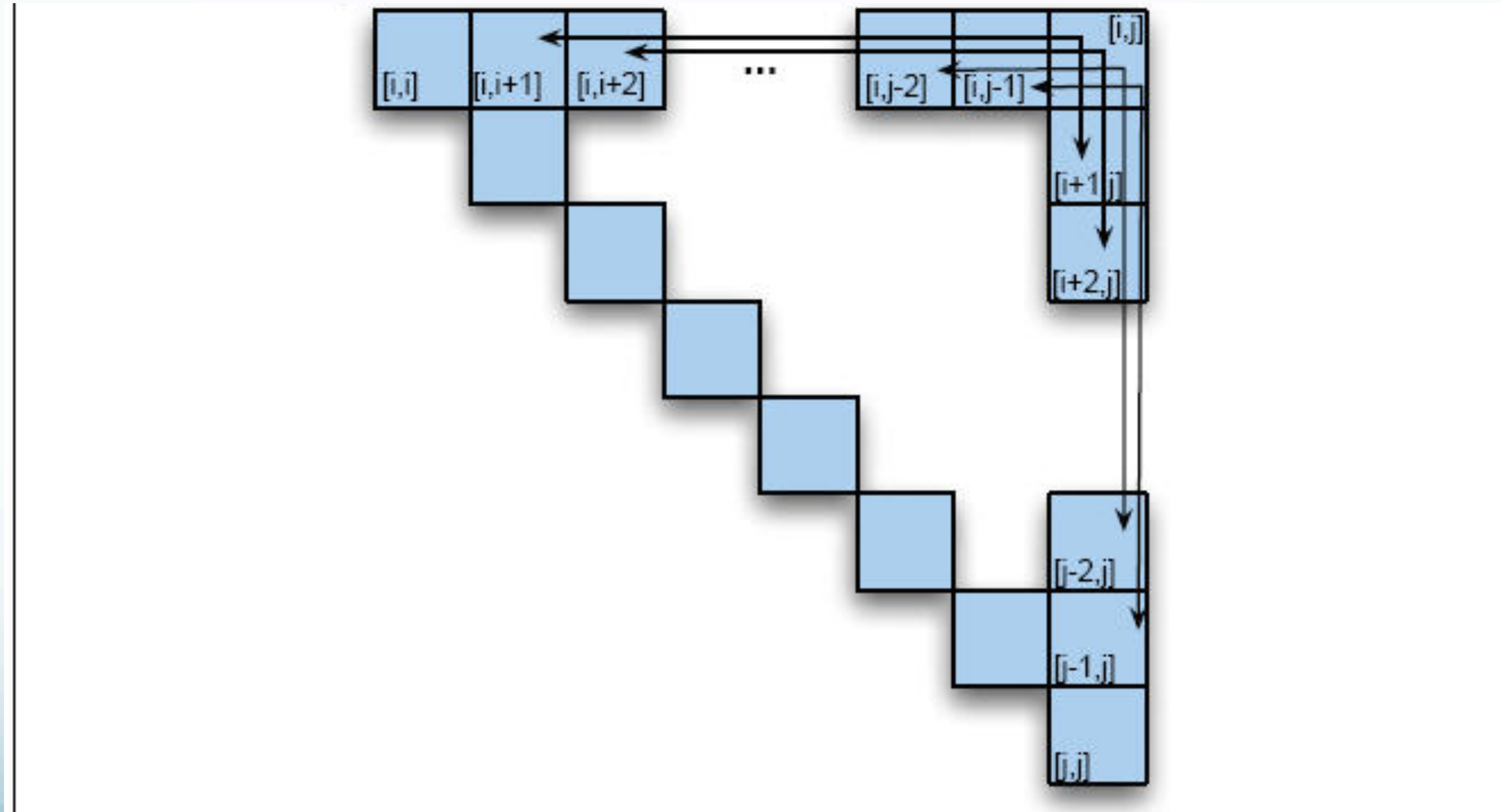
CKY Table

- Book the flight through Houston

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



Filling CKY cell



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

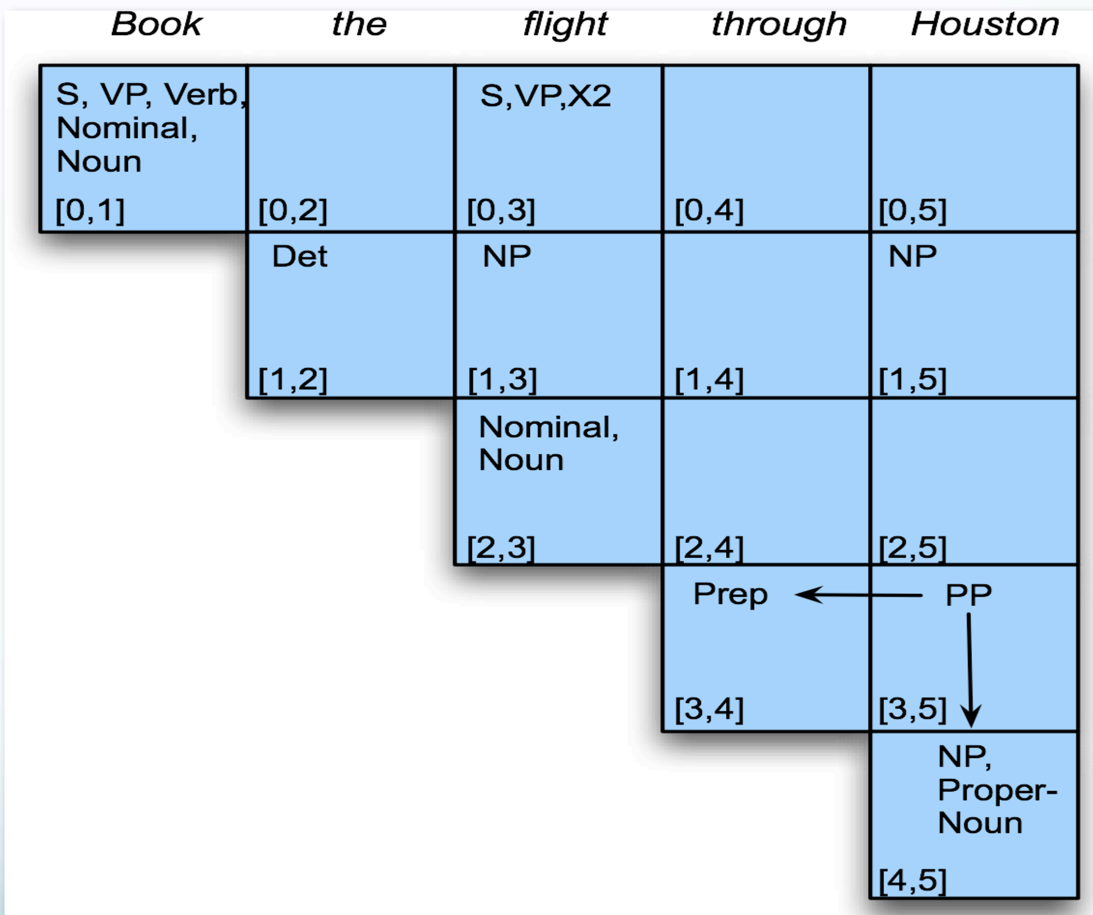
From Recognition to Parsing

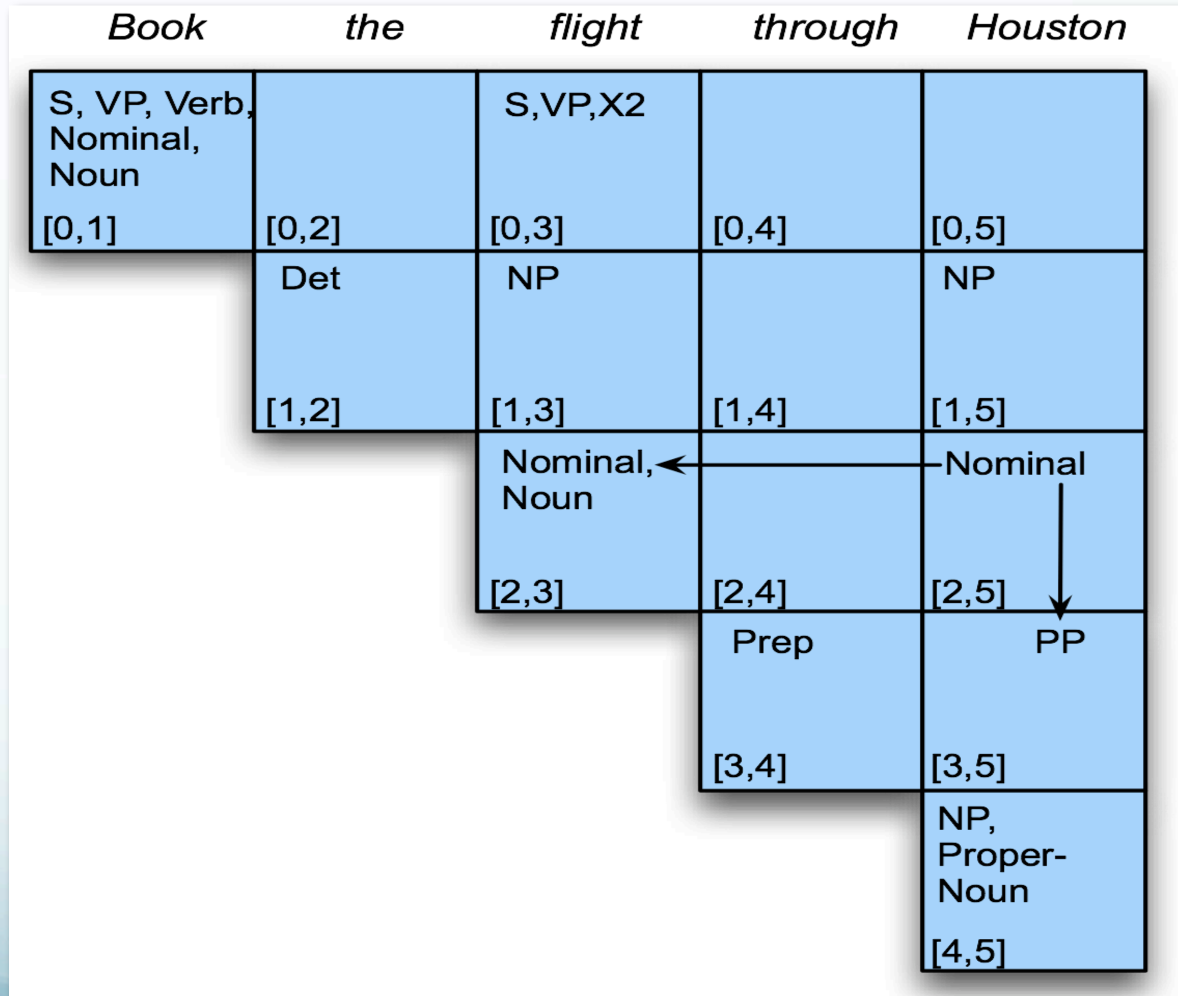
- Limitations of current recognition algorithm:
  - Only stores non-terminals in cell
    - Not rules or cells corresponding to RHS
  - Stores SETS of non-terminals
    - Can't store multiple rules with same LHS

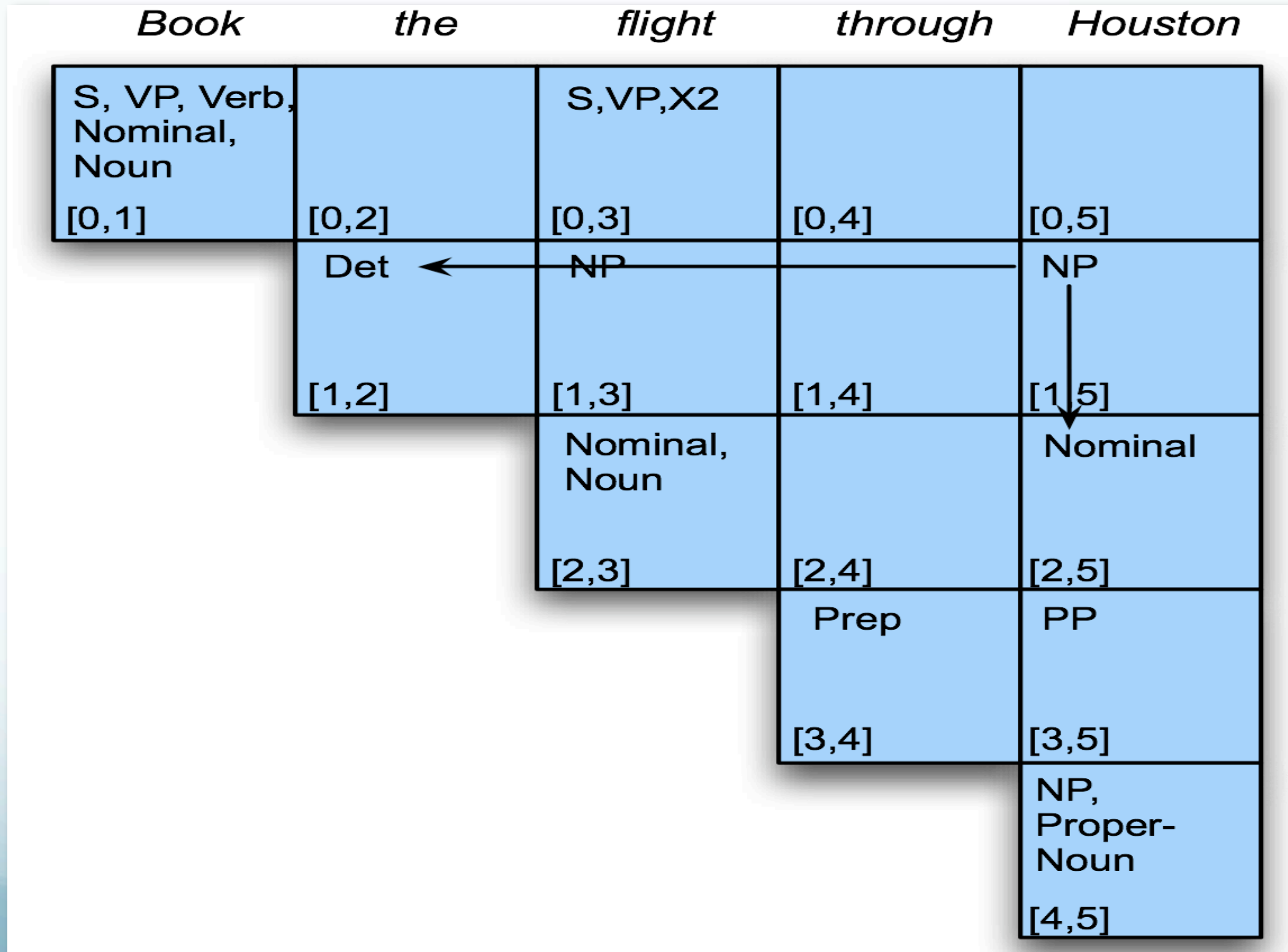
- Parsing solution:
  - All repeated versions of non-terminals
  - Pair each non-terminal with pointers to cells
    - Backpointers
  - Last step: construct trees from back-pointers in [0,n]

Filling column 5

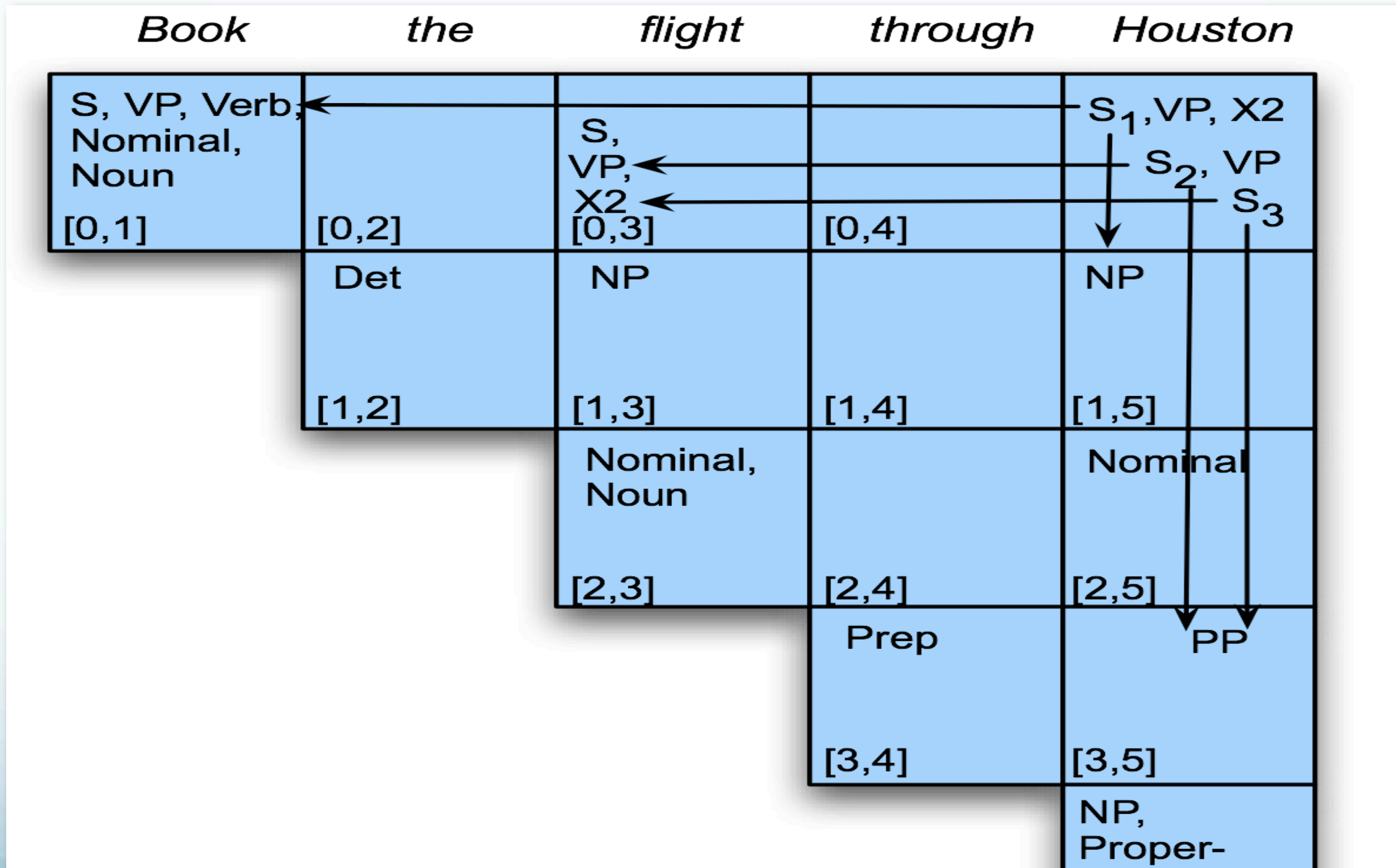
Book	*the*	*flight*	*through*	*Houston*
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	[3,5]
				NP, Proper-Noun [4,5]


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]


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]



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]


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
[0,1]
[0,2]
S, VP, X2
[0,3]
[0,4]
S1,VP, X2
S2, VP
S3
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-

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 Discussions

- 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 Discussions

- 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

Parsing Efficiently

- With arbitrary grammars
  - Earley algorithm
    - Top-down search
    - Dynamic programming
      - Tabulated partial solutions
    - Some bottom-up constraints