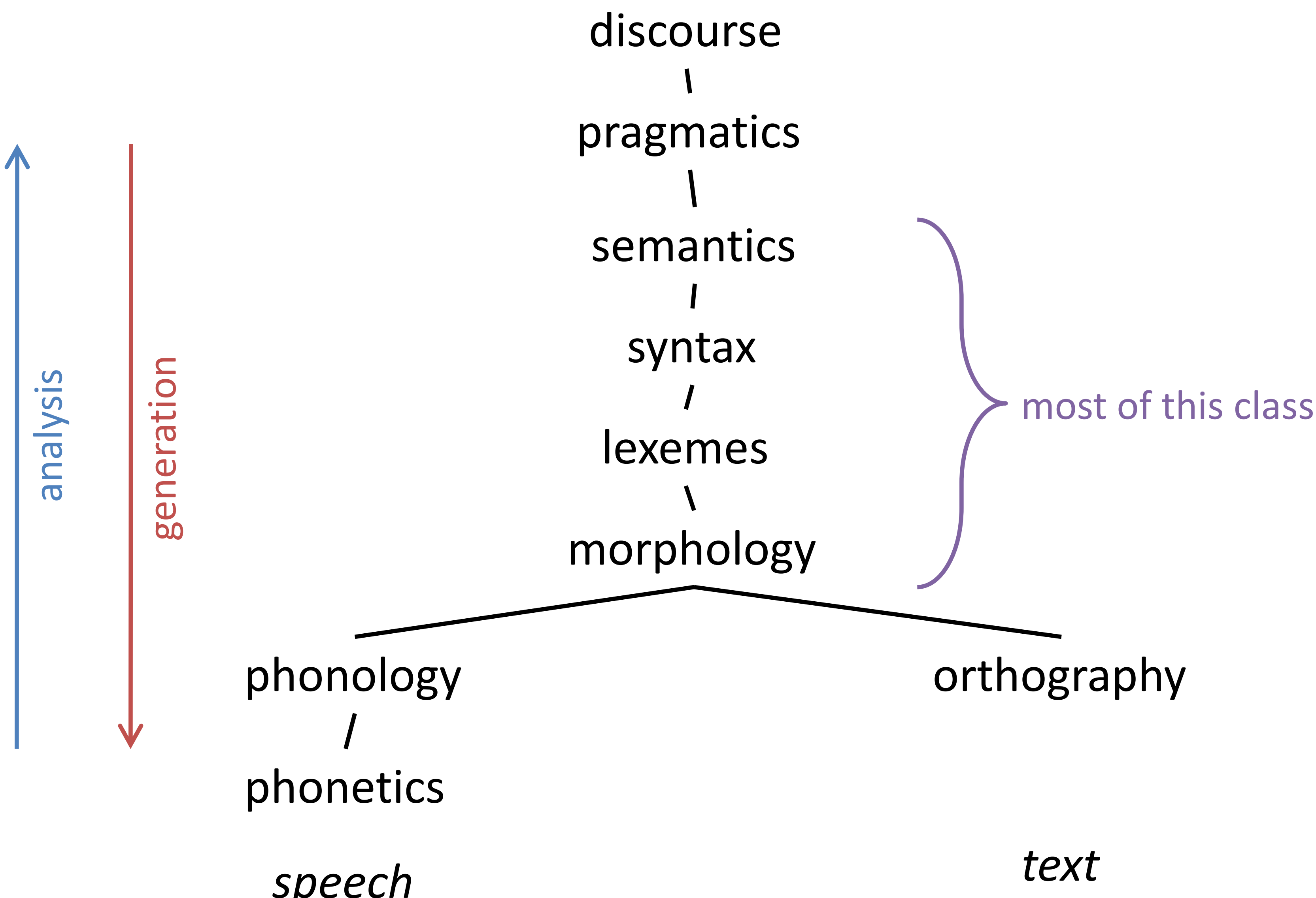
Algorithms for Natural Language Processing

Lecture 12:
Context-Free Recognition

Levels of Linguistic Representation



Context-Free Grammars

- Using grammars
  - Recognition
  - Parsing
- Parsing algorithms
  - Top down
  - Bottom up
- CNF
- CKY Algorithm
  - Cocke-Younger-Kasami

Parsing vs Word Matching

- Consider
  - The student who was taught by David won the prize
- Who won the prize?
- String matching
  - ”David won the prize.”
- Parsing based
  - ((The student (who was taught by David)) won the prize)
  - “The student won the prize”

Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols (a.k.a. variables), N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$

where

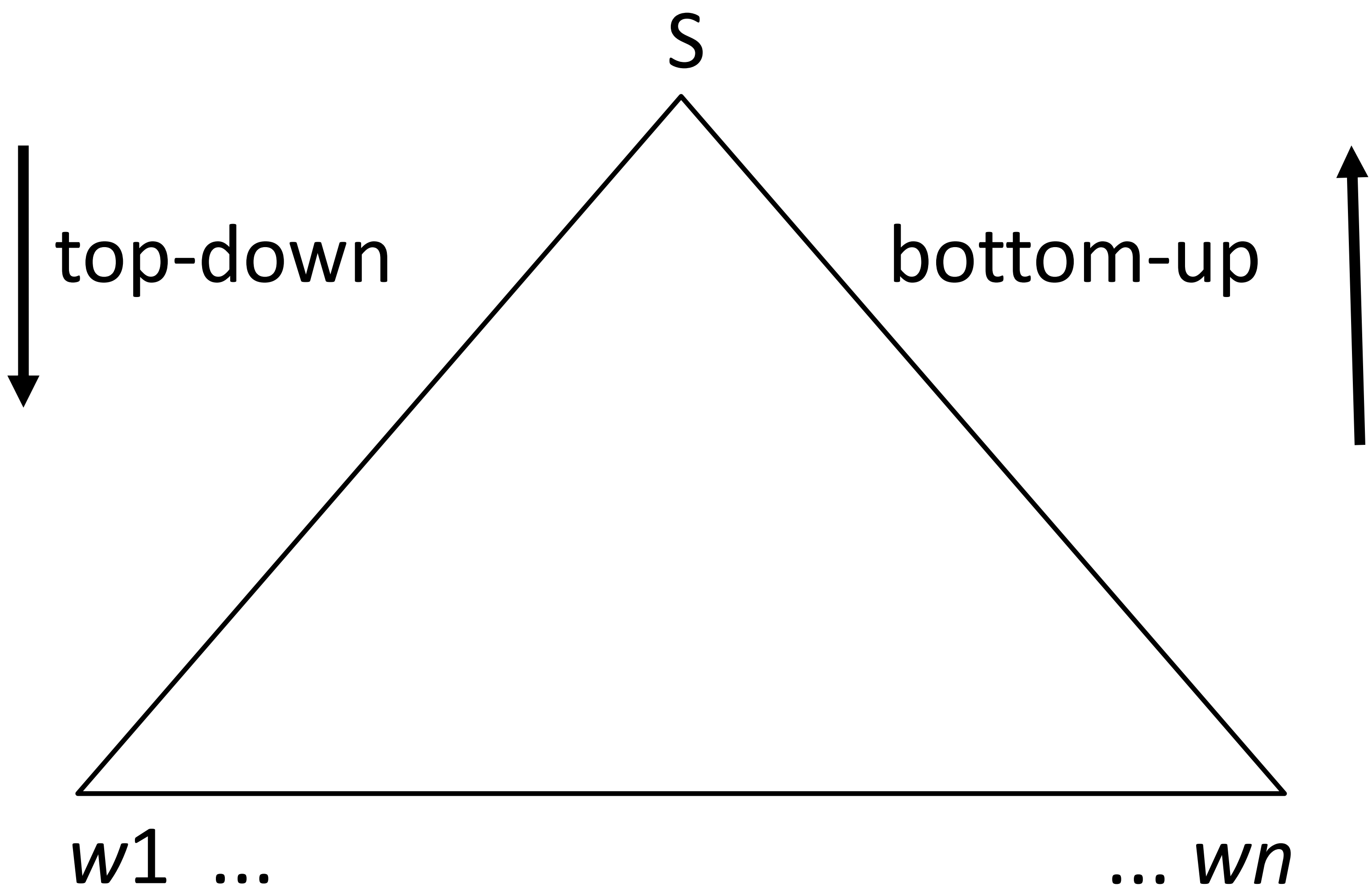
$X \in N$

$\alpha \in (N \cup \Sigma)^*$

Two Related Problems

- Input: sentence ***w*** = (*w*1, ..., *wn*) and CFG *G*
- Output (recognition): true iff ***w*** ∈ Language(*G*)
- Output (parsing): one or more derivations for ***w***, under *G*

Parsing as Search



Implementing Recognizers as Search

```
Agenda = { state0 }
while(Agenda not empty)
    s = pop a state from Agenda
    if s is a success-state return s // valid parse tree
    else if s is not a failure-state:
        generate new states from s
        push new states onto Agenda

return nil // no parse!
```

Example Grammar and Lexicon

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

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

Recursive Descent (A Top-Down Parser)

Start state: (S, 0)

Scan: From (w_{j+1} β, j), you can get to (β, $j + 1$).

Predict: If Z → γ, then from (Z β, j), you can get to (γβ, j).

Final state: (ε, n)

Example Grammar and Lexicon

Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

Shift-Reduce (A Bottom-Up Parser)

- Start state: $(\varepsilon, 0)$
- **Shift**: From (α, j), you can get to $(\alpha\ w_{j+1}, j + 1)$.
- **Reduce**: If $Z \rightarrow \gamma$, then from $(\alpha\gamma, j)$ you can get to $(\alpha\ Z, j)$.
- Final state: (S, n)

Simple Grammar

- S -> NP VP
- VP -> V NP
- NP -> John
- NP -> Delta
- V -> flies

Context-Free Grammars in Chomsky Normal Form

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols (a.k.a. variables), N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$

where

$X \in N$

$\alpha \in N,N \cup \Sigma$

Convert CFGs to CNF

- For each rule
  - X → A B C
- Rewrite as
  - X → A X2
  - X2 → B C
- Introducing a new non-terminal

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

Figure 13.8 $\mathscr{L}_1$ Grammar and its conversion to CNF. Note that although they aren't shown here all the original lexical entries from $\mathscr{L}_1$ carry over unchanged as well.

CKY Algorithm

for $i = 1 \ldots n$

  $C[i\text{-}1, i] = \{ V \mid V \rightarrow wi \}$

for $\ell = 2 \ldots n$ // width

  for $i = 0 \ldots n - \ell$ // left boundary

    $k = i + \ell$ // right boundary

      for $j = i + 1 \ldots k - 1$ // midpoint

        $C[i, k] = C[i, k] \cup$

          $\{ V \mid V \rightarrow YZ, Y \in C[i, j], Z \in C[j, k] \}$

return true if $S \in C[0, n]$

CKY Algorithm: Chart

book					
	this				
		flight			
			through		
				Houston	

CKY Algorithm: Chart

	Noun				
book					
	this				
		flight			
			through		
				Houston	

CKY Algorithm: Chart

	Noun, Verb				
book					
	this				
		flight			
			through		
				Houston	

CKY Algorithm: Chart

	Noun, Verb				
book		Det			
	this		Noun		
		flight		Prep	
			through		PNoun
				Houston	

CKY Algorithm: Chart

	Noun, Verb				
book		Det			
	this		Noun		
		flight		Prep	
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det			
	this		Noun		
		flight		Prep	
			through		PNoun NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP		
	this		Noun		
		flight		Prep	
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP		
	this		Noun		
		flight		Prep	
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP		
	this		Noun	-	
		flight		Prep	
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP	-	
	this		Noun	-	
		flight		Prep	
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP	-	
	this		Noun	-	
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP	-	
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-			
book		Det	NP	-	NP
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-	VP		
book		Det	NP	-	NP
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-	VP,S		
book		Det	NP	-	NP
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-	VP,S	-	
book		Det	NP	-	NP
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm: Chart

	Noun, Verb	-	VP,S	-	S
book		Det	NP	-	NP
	this		Noun	-	-
		flight		Prep	PP
			through		PNoun, NP
				Houston	

CKY Algorithm

```
for i = 1 ... n
        C[i-1, i] = { V | V → wi }
for ℓ = 2 ... n // width
        for i = 0 ... n - ℓ // left boundary
                k = i + ℓ  // right boundary
                        for j = i + 1 ... k – 1 // midpoint
                                C[i, k] = C[i, k] ∪
                                        { V | V → YZ, Y ∈ C[i, j], Z ∈ C[j, k] }
return true if S ∈ C[0, n]
```

CKY Equations

$$C[i-1, i, w_i] = \text{TRUE}$$

$$C[i-1, i, V] = \begin{cases} \text{TRUE} & \text{if } V \rightarrow w_i \\ \text{FALSE} & \text{otherwise} \end{cases}$$

$$C[i, j, V] = \begin{cases} \text{TRUE} & \text{if } \exists j, Y, Z \text{ such that} \\ & V \rightarrow Y\,Z \\ & \quad \text{and } C[i, k, Y] \\ & \quad \text{and } C[k, j, Z] \\ & \quad \text{and } i < k < j \\ \text{FALSE} & \text{otherwise} \end{cases}$$

$$\text{goal} = C[0, n, S]$$

CKY Complexity

- CKY worst case is O(n^3 . G)
- Best is worst case
- (Others better in average case)

CFG Grammars

- Parsing and Recognition
- Bottom up and Top down
- CKY (for CNF)