Announcements

• HW3 due tomorrow
• HW4 assigned tomorrow
• P3 due March 4th
• Midterm is on March 8th
  – Let me know THIS WEEK if you can’t make it
Bottom-up parsing algorithms

Cocke–Younger–Kasami algorithm

And

Chomsky Normal Form
Last Time

• Showed how to blindly use CUP for getting ASTs
• But we never saw HOW the parser works
This Time

• Dip our toe into parsing
  – Approaches to Parsing
  – CFG Transformations
    • Useless non-terminals
    • Chomsky Normal Form: A form of grammar that’s easier to deal with
  – CYK:
    • powerful, heavyweight approach to parsing
Approaches to Parsing

• Top Down / “Goal driven”
  – Start at root of parse tree, grow downward to match the string

• Bottom Up / “Data Driven”
  – Start at terminal, generate subtrees until you get to the start
CYK: A general approach to Parsing (Cocke–Younger–Kasami algorithm)

- Operates in $O(n^3)$
- Works Bottom-Up
- Only takes a grammar in Chomsky Normal Form
  - This will not turn out to be a limitation
Chomsky Normal Form

- All rules must be one of two forms:
  \[
  X \rightarrow t \quad \text{(terminal)}
  \]
  \[
  X \rightarrow A \ B
  \]
- The only rule allowed to derive epsilon is the start S
What CNF buys CYK

Fact that non-terminals come in pairs allows you to think of subtree as a subspan of the input

s = s1  s2  s3  s4
CYK: Dynamic Programming

\[ X \rightarrow t \]
Prods. form the leaves of the parse tree

\[ X \rightarrow A \, B \]
Form binary nodes
Running CYK...

– Track every viable subtree from leaf to root. Here are all the subspans for a string of 6 terminals

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1,6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,5</td>
<td>2,6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,4</td>
<td>2,5</td>
<td>3,6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,3</td>
<td>2,4</td>
<td>3,5</td>
<td>4,6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,2</td>
<td>2,3</td>
<td>3,4</td>
<td>4,5</td>
<td>5,6</td>
<td></td>
</tr>
<tr>
<td>1,1</td>
<td>2,2</td>
<td>3,3</td>
<td>4,4</td>
<td>5,5</td>
<td>6,6</td>
</tr>
</tbody>
</table>
CYK Example

F
1,6

W
1,5 2,6

X
1,4 2,5 3,6

N
1,3 2,4 3,5 4,6

Z
1,2 2,3 3,4 4,5 5,6

I,N
L I,N C I,N R

id ( id , id )

F → I W
F → I Y
W → L X
X → N R
Y → L R
N → id
N → I Z
Z → C N
I → id
L → ( )
R → ,
C → ,

11
CYK Example

F → I W
F → I Y
W → L X
X → N R
Y → L R
N → id
N → I Z
Z → C N
I → id
L → ( 
R → )
C → ,
CYK Example

\[
\begin{align*}
F & \rightarrow \text{I W} \\
F & \rightarrow \text{I Y} \\
W & \rightarrow \text{L X} \\
X & \rightarrow \text{N R} \\
Y & \rightarrow \text{L R} \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{I Z} \\
Z & \rightarrow \text{C N} \\
I & \rightarrow \text{id} \\
L & \rightarrow \text{(} \\
R & \rightarrow \text{)} \\
C & \rightarrow , \end{align*}
\]
CYK Example

\[
F 
\rightarrow IW
F 
\rightarrow IY
W 
\rightarrow LX
X 
\rightarrow NR
Y 
\rightarrow LR
N 
\rightarrow id
N 
\rightarrow IZ
Z 
\rightarrow CN
I 
\rightarrow id
L 
\rightarrow (
R 
\rightarrow )
C 
\rightarrow ,
\]
CYK Example

F → I W
F → I Y
W → L X
X → N R
Y → L R
N → id
N → I Z
Z → C N
I → id
L → (  
R → )
C → ,
CYK Example

F → I W
F → I Y
W → L X
X → N R
Y → L R
N → id
N → I Z
Z → C N
I → id
L → ( )
R → ,
C → ,
CYK Example

F → IW
F → IY
W → LX
X → NR
Y → LR
N → id
N → IZ
Z → CN
I → id
L → (}
R → ,
C → ,
Cleaning up our grammars

• We want to avoid unnecessary work
  – Remove *useless* rules
Eliminating Useless Nonterminals

1. If a nonterminal cannot derive a sequence of terminal symbols then it is useless
2. If a nonterminal cannot be derived from the start symbol, then it is useless
Eliminate Useless Nonterms

- If a nonterminal cannot derive a sequence of terminal symbols, then it is useless

Mark all terminal symbols
Repeat
  If all symbols on the righthand side of a production are marked
  mark the lefthand side
Until no more non-terminals can be marked
Example:

\[
\begin{align*}
S & \rightarrow X \mid Y \\
X & \rightarrow () \\
Y & \rightarrow (YY)
\end{align*}
\]
Eliminate Useless Nonterms

- If a nonterminal cannot be derived from the start symbol, then it is useless.

Mark the start symbol

Repeat

  If the lefthand side of a production is marked
  mark all righthand non-terminal

Until no more non-terminals can be marked
Example:

S → A B
A → + | - | ε
B → digit | B digit
C → . B
Chomsky Normal Form

• 4 Steps
  – Eliminate epsilon rules
  – Eliminate unit rules
  – Fix productions with terminals on RHS
  – Fix productions with > 2 nonterminals on RHS
Eliminate (Most) Epsilon Productions

• If a nonterminal $A$ immediately derives epsilon
  – Make copies of all rules with $A$ on the RHS and delete all combinations of $A$ in those copies
Example 1

F → id (A)
A → ε
A → N
N → id
N → id , N

F → id (A)
F → id ()
A → N
N → id
N → id , N
Example 2

\[
\begin{align*}
X & \rightarrow A \times A \, y \, A \\
A & \rightarrow \epsilon \\
A & \rightarrow z \\
X & \rightarrow A \times A \, y \, A \\
& \quad | \\
& \quad | \\
& \quad | \\
& \quad | \\
A & \rightarrow z
\end{align*}
\]
Eliminate Unit Productions

• Productions of the form $A \rightarrow B$ are called unit productions

• Place B anywhere A could have appeared and remove the unit production
Example 1

\[ \text{F} \rightarrow \text{id (A)} \]
\[ \text{F} \rightarrow \text{id ( )} \]
\[ \text{A} \rightarrow \text{N} \]
\[ \text{N} \rightarrow \text{id} \]
\[ \text{N} \rightarrow \text{id , N} \]

\[ \text{F} \rightarrow \text{id (N)} \]
\[ \text{F} \rightarrow \text{id ( )} \]
\[ \text{N} \rightarrow \text{id} \]
\[ \text{N} \rightarrow \text{id , N} \]
Fix RHS Terminals

• For productions with Terminals and something else on the RHS
  – For each terminal $t$ add the rule
    $X \rightarrow t$
    Where $X$ is a new non-terminal
  – Replace $t$ with $X$ in the original rules
Example

$F \rightarrow \text{id ( N )}$

$F \rightarrow \text{id ( )}$

$N \rightarrow \text{id}$

$N \rightarrow \text{id, N}$

$F \rightarrow \text{ILNR}$

$F \rightarrow \text{ILR}$

$N \rightarrow \text{id}$

$N \rightarrow \text{ICN}$

$I \rightarrow \text{id}$

$L \rightarrow \text{(}$

$R \rightarrow \text{)}$

$C \rightarrow \text{,}$
Fix RHS Nonterminals

• For productions with > 2 Nonterminals on the RHS
  – Replace all but the first nonterminal with a new nonterminal
  – Add a rule from the new nonterminal to the replaced nonterminal sequence
  – Repeat
Example

F ⟶ I L N R

F ⟶ I W
W ⟶ L N R

F ⟶ I W
W ⟶ L X
X ⟶ N R
Parsing is Tough

• CYK parses an arbitrary CFG, but
  – O(n³)
  – Too slow!

• For special class of grammars
  – O(n)
  – Includes LL(1) and LALR(1)
Classes of Grammars

• LL(1)
  – Scans input from Left-to-right (first L)
  – Builds a Leftmost Derivation (second L)
  – Can peek (1) token ahead of the token being parsed
  – Top-down “predictive parsers”

• LALR(1)
  – Uses special lookahead procedure (LA)
  – Scans input from Left-to-right (second L)
  – Rightmost derivation (R)
  – Can also peek (1) token ahead

• LALR(1) strictly more powerful, much harder to understand
In summary

We talked about how to parse with CYK and Chomsky Normal Form grammars