CS536

Intro to Parsing
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 Nonterminals
    • CNF: 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

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

• All rules must be one of two forms:
  \[ X \rightarrow t \]
  \[ X \rightarrow A B \]

• The only rule allowed to derive epsilon is the start rule, in which case it’s forbidden on the RHS of any rule
What CNF buys CYK

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

\[ s = s_1 \quad s_2 \quad s_3 \quad s_4 \]
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
CYK Example

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

\[
F \rightarrow \text{I W} \\
F \rightarrow \text{I Y} \\
W \rightarrow \text{L X} \\
X \rightarrow \text{N R} \\
Y \rightarrow \text{L R} \\
N \rightarrow \text{id} \\
N \rightarrow \text{I Z} \\
Z \rightarrow \text{C N} \\
I \rightarrow \text{id} \\
L \rightarrow ( \\
R \rightarrow ) \\
C \rightarrow ,
\]

id ( id , id )
CYK Example

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

```
1,6
2,6
3,6
3,5
4,5
id ( id , id )
```

```
F
W
X
N
Z
I, N
L
I
C
N
R
```
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 → ,

id ( id , id )
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 → ,
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Right Hand Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>→ I W</td>
</tr>
<tr>
<td>F</td>
<td>→ I Y</td>
</tr>
<tr>
<td>W</td>
<td>→ L X</td>
</tr>
<tr>
<td>X</td>
<td>→ N R</td>
</tr>
<tr>
<td>Y</td>
<td>→ L R</td>
</tr>
<tr>
<td>N</td>
<td>→ id</td>
</tr>
<tr>
<td>N</td>
<td>→ I Z</td>
</tr>
<tr>
<td>Z</td>
<td>→ C N</td>
</tr>
<tr>
<td>I</td>
<td>→ id</td>
</tr>
<tr>
<td>L</td>
<td>→ (</td>
</tr>
<tr>
<td>R</td>
<td>→ )</td>
</tr>
<tr>
<td>C</td>
<td>→ ,</td>
</tr>
</tbody>
</table>
Cleaning up our grammars

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

1. If a nonterminal cannot derive a terminal symbol 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 terminal symbol, then it is useless

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

\[
\begin{align*}
S & \rightarrow X \mid Y \\
X & \rightarrow () \\
Y & \rightarrow (Y Y)
\end{align*}
\]
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:

\[
\begin{align*}
S & \rightarrow \quad A \ B \\
A & \rightarrow \quad + \ | \ - \ | \ \varepsilon \\
B & \rightarrow \quad \text{digit \ | \ B digit} \\
C & \rightarrow \quad . \ B
\end{align*}
\]
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

\[
\begin{align*}
F & \rightarrow \text{id} (A) \\
A & \rightarrow \varepsilon \\
A & \rightarrow N \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{id} , N \\
\end{align*}
\]
Example 2

\[
\begin{align*}
X & \rightarrow A \times A \ y A \\
A & \rightarrow \varepsilon \\
A & \rightarrow z
\end{align*}
\]
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

\[
\begin{align*}
F & \rightarrow \text{id (A)} \\
F & \rightarrow \text{id ( )} \\
A & \rightarrow N \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{id , N} \\
\end{align*}
\]

\[
\begin{align*}
F & \rightarrow \text{id (N)} \\
F & \rightarrow \text{id ( )} \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{id , N} \\
\end{align*}
\]
Fix RHS Terminals

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

\[
\begin{align*}
F & \rightarrow \text{id (N)} \\
F & \rightarrow \text{id ()} \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{id, N}
\end{align*}
\]

\[
\begin{align*}
F & \rightarrow \text{ILNR} \\
F & \rightarrow \text{ILR} \\
N & \rightarrow \text{id} \\
N & \rightarrow \text{ICN} \\
I & \rightarrow \text{id} \\
L & \rightarrow ( \\
R & \rightarrow ) \\
C & \rightarrow ,
\end{align*}
\]
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
  – $O(n^3)$
  – 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 CNF