Plan for Today

Logistics
  – Midterm
    – Handing it out, extra credit has not been added yet.
    – Discussing stats, but not recording that.
    – If you think you might request a grade change, then leave midterm here.
  – PA2 due Monday

“15 minute” compiler example
  – Lexer: regular expressions, NFA, DFA, and then modified DFA
  – Nullable, FIRST, and FOLLOW

Top-down Predictive Parsing with Code Generation
  – Predictive parsing table for “15 minute” example
  – Nullable, FIRST, and FOLLOW
”15 Minute” Compiler Example

Tomorrow’s Recitation for Tomorrow
- Show example of using it.
- Show Context Free Grammar in Main15min.hs.

Lexical Analysis
- Regular expressions for tokens
- NFA for all tokens
- NFA to DFA, or check that it is a DFA
- Modify DFA for lexer
Predictive Parsing and Code Gen for “15 minute” Compiler

Building the Predictive Parse Table
- Compute Nullable, FIRST, and FOLLOW
- For this grammar, need an EOF token
- Draw the predictive parsing table
- Show correspondence between predictive parsing table and code.

Code generation
- Every time a production in the grammar matches, a string is created.
- The parsing function for each non-terminal returns a string.
- The matchAndGrab... functions for tokens return information about particular tokens.
Class Exercise

Compute nullable, FIRST and FOLLOW for

\[ Z \rightarrow d \mid X Y Z \]
\[ X \rightarrow a \mid Y \]
\[ Y \rightarrow c \mid \varepsilon \]
Constructing the Predictive Parser Table

A predictive parse table has a row for each non-terminal X, and a column for each input token t. Entries table[X,t] contain productions:

for each X -> gamma
    for each t in FIRST(gamma)
        table[X,t] = X->gamma
    if gamma is nullable
        for each t in FOLLOW(X)
            table[X,t] = X->gamma

Compute the predictive parse table for

\[
\begin{align*}
Z & \rightarrow d & | & X, Y, Z \\
X & \rightarrow a & | & Y \\
Y & \rightarrow c & | & \varepsilon \\
\end{align*}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
 & a & c & d \\
\hline
X & X \rightarrow a & X \rightarrow Y & X \rightarrow Y \\
Y & Y \rightarrow \varepsilon & Y \rightarrow \varepsilon & Y \rightarrow \varepsilon \\
Z & Z \rightarrow XYZ & Z \rightarrow XYZ & Z \rightarrow XYZ \\
\hline
\end{array}
\]
One more time

Balanced parentheses grammar 1:

\[ S \rightarrow ( S ) | SS | \varepsilon \]

1. Augment the grammar with EOF/$

2. Construct Nullable, First and Follow

3. Build the predictive parse table, what happens?
One more time, but this time with feeling …

Balanced parentheses grammar 2:

\[ S \rightarrow ( S ) S \mid \varepsilon \]

1. Augment the grammar with EOF/$

2. Construct Nullable, First and Follow

3. Build the predictive parse table

4. Using the predictive parse table, construct the parse tree for

\[ ( ) ( ( ) ) \] $\]

and

\[ ( ) ( ) ( ) \] $\]