Plan for Today

Predictive parsing as a specific subclass of recursive descent parsing
  – complexity comparisons with general parsing
  – might have to left-factor
  – necessary to remove left-recursion (MiniSVG modification as example)

Working with a partner

PA2 MeggyJava

Predictive Parsing Complexity

LL(k) grammar classes
  – Left-to-right scan
  – Left-most derivation
  – k tokens of lookahead

Comparing complexity
  – O(N^3) for general case algorithms, where N is the number of tokens in the stream
  – O(N) for predictive parsing

Requirements for LL(1), for all productions of nonterminal A
  – None of the FIRST(rhs) for A production rules can overlap
  – If nullable(A) then FOLLOW(A) must not overlap with FIRST(rhs) for any A->rhs
Constructing the Predictive Parser Table

**Algorithm**

for each $X \rightarrow \gamma$

for each $T$ in FIRST($\gamma$)

\[ \text{table}[X,T] = X \rightarrow \gamma \]

if $\gamma$ is nullable

for each $T$ in FOLLOW($X$)

\[ \text{table}[X,T] = X \rightarrow \gamma \]

---

**Left Refactoring**

**First requirement for LL(1)**

– None of the FIRST(rhs) for a production rules can overlap

**General Case**

\[
\begin{align*}
S & \rightarrow \alpha \beta_1 \\
S & \rightarrow \alpha \beta_2
\end{align*}
\]

\[ \text{Left refactor} \]

\[
\begin{align*}
S & \rightarrow \alpha S' \\
S' & \rightarrow \beta_1 | \beta_2
\end{align*}
\]
Predictive Parser Table for Modified MiniSVG Grammar

Algorithm
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

(1) svg -> SVG_START elem_list SVG_END EOF
(2a & b) elem_list -> elem_list elem | epsilon
(3) elem -> RECT_START ... ELEM_END
(4) elem -> CIRCLE_START ... ELEM_END
(5) elem -> LINE_START ... ELEM_END

Example Parse Tree for Modified MiniSVG (PROBLEM!)
Working with a partner

Why?
– No programmer is an island.
– The MeggyJava compiler is a significant course project.
– Working with a partner SHOULD encourage planning ahead.
– Two heads are better than one.

Each of you is ultimately responsible for making the compiler work
– Come see me if you are having problems.

Partner Report and Evaluation

Joint pieces submitted with PA3 through PA7
– The plan for who is to do what and who actually did what.
– What is the plan for testing? What was the reality?
– A timeline including a meeting schedule.

Separate pieces emailed to mstrout@cs.colostate.edu by each programming assignment deadline (PA3 – PA7)
– Approximately ½ page
– Organizational strategies that are working well.
– Specifics on how the work was divided.
– How could the division of work be improved?
– How could the interaction between partners be improved?
Grading the Partner Report and Evaluation

5% of total course grade, part of the project grade

Will be posted AFTER PA7 has been submitted

Grading criteria
– Were all of the aspects on the previous slide addressed in a thoughtful manner.
– How well do the reports match.
– Clarity and succinctness of the writing.
– A subjective evaluation of how well each partner attempted to make the group programming experience work.

PA2: MeggyJava

How do we run a MeggyJava example?

How does the opt_args file work?

How is PA2 going to be graded?
– Follow directions!! More points will be taken off next time if you don’t.
– Use subversion or we will have to create a usage regiment.