LL Parsing, part 2
CS 440: Programming Languages and Translators, Spring 2019
Lecture 13, Wed Mar 6

Notation

- \( a, b, \ldots \in T \) (a and b stand for members of T; a, b, … are members of T) Was \( \Sigma \) for regular expressions.
- \( x, y, \ldots \in T^* \)
- \( A, B, \ldots \in V \)
- \( W, X, \ldots \in T \mid V \)
- \( \alpha, \beta, \ldots \in (T \mid V)^* \)  Sentential form

Review – \( LL(k) \) Parser

- A top-down parser looks for a parse tree for an inputted terminal string; it begins with start symbol and repeatedly replaces a nonterminal with a rhs of one of its rules.
- An \( LL(k) \) parser always replaces the leftmost nonterminal, so it tries to find the leftmost derivation for the input. It gets to look at \( k \) of the upcoming characters to figure out which rule to use. Typically \( k = 1 \). Recursive descent parsing was an example.
- Table-driven \( LL \) parsers use a table to predict which rule to use.
  - Given nonterminal \( A \) to replace and next character \( a \) of lookahead, prediction table(\( A, a \)) specifies the \( A \to \alpha \) rule to use.
    - If the table entry is empty, we have a parse error (didn't expect to see an \( a \)).
    - When we build the table, if a table entry tries to contain > 1 entry, the grammar is not \( LL(1) \).
- Two kinds of legal next character \( a \) to look at.
  - Rule \( A \to \alpha \) applies if \( a \) is a first character that can be derived from \( \alpha \) \( (A \to \alpha \to^* a \gamma) \).
    - Note \( \gamma \) isn't important, it just stands for whatever form happens to be there.
  - Rule \( A \to \epsilon \) applies if \( a \) is a character that can follow a use of \( A \) \( (S \to \gamma A \beta \to^* \gamma A a \gamma' \to \gamma \epsilon a \gamma') \).
- Needed a couple of utility functions
  - \( First(\alpha) = \) set of first characters that can be derived from \( \alpha \). (Plus, include \( \epsilon \) if \( \alpha \to^* \epsilon \) is possible.)
  - \( Follow(\alpha) \) is the set of characters that can follow just after a derivation of \( \alpha \). (\( \epsilon \) is never in \( Follow(\alpha) \).)
    - If \( \alpha \to^* a \gamma \) can occur, then \( a \in First(\alpha) \).
    - If \( \alpha \to^* B a \gamma \to^* \epsilon a \gamma \) can occur (note we need \( B \to^* \epsilon \)) then \( a \in First(\alpha) \) and \( a \in Follow(B) \).
      - \( \alpha \to^* B a \gamma \to^* \epsilon a \gamma = a \gamma \)
    - If \( S \to^* \gamma B a \gamma' \) can occur, then \( a \in Follow(B) \).
  - Each grammar rule generates some number of \( First \) and \( Follow \) relationships, depending on the form of the rule.
    - \( A \to a \gamma \) generates the relation \( a \in First(A) \).
• Similarly, \( A \rightarrow \epsilon \) generates \( \epsilon \in \text{First}(A) \).

• \( A \rightarrow \epsilon \) also causes each use of \( A \) to generate a \textit{Follow} relationship:
  • \( B \rightarrow \gamma A \beta \) causes \( \text{First}(\beta) \subseteq \text{Follow}(A) \).
    • If \( \gamma \rightarrow^* \epsilon \) is possible (e.g. \( B \rightarrow A \beta \) or \( B \rightarrow C A \beta \) where \( C \rightarrow \epsilon \)), then \( A \rightarrow \epsilon \) also causes \( \text{First}(\beta) \subseteq \text{First}(B) \).

• Whether or not \( A \rightarrow^* \epsilon \),
  • \( B \rightarrow \gamma A \) causes \( \text{Follow}(B) \subseteq \text{Follow}(A) \)

\textbf{Example: List expressions}

• Here's the syntax for the list expressions we've seen before:
  1. \( E \rightarrow \text{id} \) // Expression
  2. \( E \rightarrow (E Es) \) // Parenthesized list of \( \geq \) one E's
  3. \( Es \rightarrow , E Es \) // , E followed by more E's
  4. \( Es \rightarrow \epsilon \) // empty ends list of E's

• And now the inferences from the rules:
  Rule 1:
  • \( \text{id} \in \text{First}(E) \)

  Rule 2:
  • \( ( \in \text{First}(E) \)
  • \( ) \in \text{Follow}(Es) \)
  • \( \text{First}(Es) - \epsilon \subseteq \text{Follow}(E) \)
  • if \( Es \rightarrow^* \epsilon \) then \( ) \in \text{Follow}(E) \)

  Rule 3:
  • \( , \in \text{First}(Es) \)
  • \( \text{First}(Es) - \{\epsilon\} \subseteq \text{Follow}(E) \)
  • if \( Es \rightarrow^* \epsilon \) then \( \text{Follow}(Es) \subseteq \text{Follow}(E) \)

  Rule 4:
  • \( \epsilon \in \text{First}(Es) \)
  • \( Es \rightarrow^* \epsilon \) is true

• We can calculate the \textit{First} and \textit{Follow} sets by repeatedly passing through the inferences above. Below, the first pass through the inferences is shown i

  , stopping when no new information is created during a pass.
  1. Rule 1: \( \text{id} \in \text{First}(E) \)
  2. Rule 2: \( ( \in \text{First}(E) \)
  3. Rule 2: \( ) \in \text{Follow}(Es) \)
  4. Rule 3: \( , \in \text{First}(Es) \)
  5. Rule 4: \( \epsilon \in \text{First}(Es) \)
6. From $\epsilon \in \textit{First}(Es)$: $Es \rightarrow^* \epsilon$ is true

7. Rule 2 (and 3): $\textit{First}(Es) - \epsilon = \{ , \} \subseteq \textit{Follow}(E)$

8. Rule 2: $Es \rightarrow^* \epsilon$, so $\) \in \textit{Follow}(E)$

9. Rule 3: $Es \rightarrow^* \epsilon$, so $\textit{Follow}(Es) = \{ \} \subseteq \textit{Follow}(E)$

Below, the numbers say during what pass we added the information to the entry:

<table>
<thead>
<tr>
<th>$\textit{First}$</th>
<th>$\textit{Follow}$</th>
<th>$\rightarrow^* \epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$Es$</td>
<td>$E$</td>
</tr>
<tr>
<td>1: $\text{id}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: (</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: )</td>
<td>1: )</td>
<td></td>
</tr>
<tr>
<td>1: ,</td>
<td>2: ,</td>
<td></td>
</tr>
<tr>
<td>1: $\epsilon$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

And here's the prediction table for the parser:

<table>
<thead>
<tr>
<th>Nonterminal</th>
<th>$\text{id}$</th>
<th>(</th>
<th>)</th>
<th>,</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$E \rightarrow \text{id}$</td>
<td>$E \rightarrow ( E Es )$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Es$</td>
<td></td>
<td>$Es \rightarrow \epsilon$</td>
<td>$Es \rightarrow , E Es$</td>
<td></td>
</tr>
</tbody>
</table>

Example: Problems with $E \rightarrow E + E | E * E | \text{id}$

- $\textit{First}, E + E | E * E | \text{id}$ is not LL(1)

- After factoring out the initial $E$ and adding a unique start symbol, we get
  - $S \rightarrow E \$, $E \rightarrow E T$, $T \rightarrow + E T | * E T | \epsilon$, $E \rightarrow \text{id}$

- Below, we analyze the rules for $\textit{First}, \textit{Follow}, \rightarrow^* \epsilon$ properties. The inferences are named (a), (b), … for reference further along.

**Inferences**

- $S \rightarrow E \$
  - (a) $\$ \in \textit{Follow}(E)$
  - (b) $\textit{First}(E) \subseteq \textit{First}(S)$

- $E \rightarrow \text{id}$
  - (c) $\text{id} \in \textit{First}(E)$

- $E \rightarrow E T$
  - (useless) $\textit{First}(E) \subseteq \textit{First}(E)$
  - (d) $\textit{First}(T) \subseteq \textit{Follow}(E)$
Follow(E) ⊆ Follow(T)

T → + ET

Follow(T) ⊆ Follow(T)

T → * ET

Follow(T) ⊆ Follow(T)

T → ε

ε ∈ First(T)

First(E) ⊆ First(S) {id} ⊆ First(S)

First(T) - ε ⊆ Follow(E) {+, *} ⊆ Follow(E) [along with {\$}]

Since T →* ε, we apply (g) Follow(T) ⊆ Follow(E) but this doesn't change Follow(E)

A third pass does not change any of the sets, so we're done.

Summary

First(S) = {id} Follow(S) = {\$}

First(E) = {id} Follow(E) = {+, *, \$}

First(T) = {+, *, ε} Follow(T) = {\$} True: T →* ε

Problems

The grammar is not LL: Since id ∈ First(E), we can't distinguish between E → ET and E → id.

Having the rules T → + ET and T → * ET makes + and * of equal precedence.