CSEN 1003: Compilers

Tutorial 5 - Predictive LL(1) Parsing

1/3/2020 - 4/3/2020

Today's Plan

1. Non-deterministic Parsers
2. LL(1) Grammars
3. LL(1) Parsers
4. Recap

Basic Top-Down Parser

- A basic top-down parser searches for a derivation from the start variable S to the input program.

Basic Top-Down Parser

- A basic top-down parser searches for a derivation from the start variable S to the input program.

Example

$$S \rightarrow SS+ \mid SS* \mid \mathsf{a}$$

Input: aa+

Predictive Parsing

- A predictive parser always chooses the right rule to apply.

Predictive Parsing

- A predictive parser always chooses the right rule to apply.
- This is possible for certain classes of CFGs.

Predictive Parsing

- A predictive parser always chooses the right rule to apply.
- This is possible for certain classes of CFGs.
- Today we will look into one such class called $LL(1)$ Grammars.

Predictive Parsing

- A predictive parser always chooses the right rule to apply.
- This is possible for certain classes of CFGs.
- Today we will look into one such class called $LL(1)$ Grammars.
- LL(1) stands for:
  - Left to right input scanning.
  - Left most derivation.
  - 1 symbol of lookahead.

Today's Plan

First and Follow - Towards Deterministic Parsers

- First and Follow allows the parser to choose the correct rule to apply.

Definition

Let α be a sentential form of G.

$$First(\alpha) = \{a \mid a \in \Sigma \text{ and } \alpha \overset{*}{\Rightarrow} a\beta\}$$

$$\text{If } \alpha \overset{*}{\Rightarrow} \varepsilon, \text{ then } \varepsilon \in First(\alpha).$$

First and Follow - Towards Deterministic Parsers

- First and Follow allows the parser to choose the correct rule to apply.

Definition

Let α be a sentential form of G.

$$First(\alpha) = \{a \mid a \in \Sigma \text{ and } \alpha \stackrel{*}{\Rightarrow} a\beta\}$$

$$\text{If } \alpha \stackrel{*}{\Rightarrow} \varepsilon, \text{ then } \varepsilon \in First(\alpha).$$

Example

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & \textbf{id}\ A \mid \textbf{num} \\ B & \rightarrow & CA \\ C & \rightarrow & \textbf{0}C \mid \textbf{1} \end{array}$$

First

To Compute $First(A)$

1. If A is a terminal, $First(A) = \{A\}$.
2. If $A \rightarrow \varepsilon$, then $\varepsilon \in First(A)$.
3. If $A \rightarrow Y_1 Y_2 \ldots Y_n$, then $First(Y_1 Y_2 \ldots Y_n) \subseteq First(A)$ where:
   a. $First(Y_1 Y_2 \ldots Y_n) = First(Y_1)$ if $\varepsilon \notin First(Y_1)$.
   b. If $\varepsilon \in First(Y_1)$, then $First(Y_1 Y_2 \ldots Y_n)$ contains $First(Y_1) - \{\varepsilon\}$ as well as $First(Y_2, \ldots, Y_n)$.
   c. If $\varepsilon \in First(Y_i)\ \forall i. 1 \leq i \leq n$, then $\varepsilon \in First(Y_1 Y_2 \ldots Y_n)$.

Follow

Definition

Let A be a variable for a grammar with S as the start variable.

$$Follow(A) = \{a \in \Sigma \mid S \overset{*}{\Rightarrow} \alpha A a \beta\}.$$

$$\$ \in Follow(S) \text{ if } S \text{ is the start variable.}$$

Example

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & \textbf{id}\ A\ \textbf{id} \mid \textbf{num} \\ B & \rightarrow & CA \\ C & \rightarrow & \mathbf{0}C \mid \mathbf{1} \end{array}$$

Follow

To Compute Follow

1. If S is the start variable, then $\$ \in Follow(S)$.
2. If $A \rightarrow \alpha B \beta$, then $First(\beta) - \{\varepsilon\} \subseteq Follow(B)$.
3. If $A \rightarrow \alpha B$, or
   $A \rightarrow \alpha B \beta$ and $\varepsilon \in First(\beta)$, then $Follow(A) \subseteq Follow(B)$.

LL(1) Parsing Table

- An LL(1) parsing table M is a table where the rows are variables, and the columns are terminals $\cup\{\$\}$.

		b	\$
A			

- $A \rightarrow \alpha \in M[A, b]$ if:
  1. $b \in First(\alpha)$; or
  2. $\varepsilon \in First(\alpha)$ and $b \in Follow(A)$.

LL(1) Parsing Table

- An LL(1) parsing table M is a table where the rows are variables, and the columns are terminals $\cup\{\$\}$.

		b	\$
A			

- $A \rightarrow \alpha \in M[A, b]$ if:
  1. $b \in First(\alpha)$; or
  2. $\varepsilon \in First(\alpha)$ and $b \in Follow(A)$.
- A grammar is $LL(1)$ if each entry in the parsing table has maximum one rule.

Example

Example

Construct the LL(1) parsing table for the following grammar.

$$
\begin{array}{rcl}
S & \rightarrow & \mathrm{a}S' \\
S' & \rightarrow & SS'' \mid \varepsilon \\
S'' & \rightarrow & +S' \mid *S'
\end{array}
$$

Today's Plan

1. Non-deterministic Parsers
2. LL(1) Grammars
3. **LL(1) Parsers**
4. Recap

Build an LL(1) in 3 Steps!

1. Construct the PDA for Grammar G.
2. Construct the LL(1) Parsing Table M.
3. If the top of the stack is A and the input head is pointing at b, use $M[A, b]$ to choose a transition.
   If $M[A, b] = \varnothing$, then output an error.

If we output the chosen rules, we can build a parse tree!

One Last Example

Example

Is the following Grammar LL(1)?

$$
\begin{array}{lcl}
S & \rightarrow & SAB \mid SBC \mid \varepsilon \\
A & \rightarrow & \mathtt{a}A\mathtt{a} \mid \varepsilon \\
B & \rightarrow & \mathtt{b}B \mid \varepsilon \\
C & \rightarrow & \mathtt{c}C \mid \varepsilon
\end{array}
$$

Today's Plan

Points To Take Home!

1. LL(1) Grammars.
2. LL(1) Parsers.

After the Midterm: Bottom Up Parsing!