Recitation #9 – Worksheet

SLR Parser Construction
Given the grammar \( G = (\{\text{SheepNoise}\}, \{\text{baa}\}, \text{SheepNoise}, P) \), where \( P \) is:

\[
\text{SheepNoise} \rightarrow \text{SheepNoise} \text{ baa} \\
\text{SheepNoise} \rightarrow \text{baa}
\]

For brevity you can use the short-hand notation:

\[
\text{SN} \rightarrow \text{SN} \text{ baa} \\
\text{SN} \rightarrow \text{baa}
\]

1. Write down the Augmented Grammar \( G' \)

\[
G' = (\{S, \text{SN}\}, \{\text{baa}\}, S, P), \text{ where } P
\]

\[
S \rightarrow \text{SN} \\
\text{SN} \rightarrow \text{SN} \text{ baa} \\
\text{SN} \rightarrow \text{baa}
\]

2. What are the LR(0) items for production \( \text{SN} \rightarrow \text{SN} \text{ baa} \)?

\[
\{ \text{SN} \rightarrow \text{SN} \text{ baa}, \text{SN} \rightarrow \text{SN} \text{ baa}, \text{SN} \rightarrow \text{SN} \text{ baa} \}
\]

3. Calculate \( \text{CLOSURE}(I) \) for \( I = \{S \rightarrow \text{SN}, \text{SN} \rightarrow \text{SN} \text{ baa}\} \). What are the Kernel Items?

- \( \text{CLOSURE}([S \rightarrow \text{SN}, \text{SN} \rightarrow \text{SN} \text{ baa}]) = [S \rightarrow \text{SN}, \text{SN} \rightarrow \text{SN} \text{ baa}] \)
- Because \( S \rightarrow \text{SN} \) is in CLOSURE, add \( \text{SN} \rightarrow \text{SN} \text{ baa} \) and \( \text{SN} \rightarrow \text{baa} \).
- Nothing to be done for \( \text{SN} \rightarrow \text{SN} \text{ baa} \) and \( \text{SN} \rightarrow \text{baa} \), because terminal follows after dot.
- Nothing to be done for \( \text{SN} \rightarrow \text{SN} \text{ baa} \), because rules deriving from \( \text{SN} \) were already added.
- \( \text{CLOSURE}([S \rightarrow \text{SN}, \text{SN} \rightarrow \text{SN} \text{ baa}]) = [S \rightarrow \text{SN}, \text{SN} \rightarrow \text{SN} \text{ baa}]
\]

Kernel Items

Nonkernel Items
4. Calculate $\text{GOTO}(I, X)$ where $I = \{S \rightarrow SN, SN \rightarrow SN \cdot baa\}$ and $X = baa$.
   - $\text{GOTO}(\{S \rightarrow SN, SN \rightarrow SN \cdot baa\}, baa)$
   - “baa” is immediately to the right of the dot for $SN \rightarrow SN \cdot baa$
   - Calculate $\text{CLOSURE}(\{SN \rightarrow SN \cdot baa\}) = \text{GOTO}(\{S \rightarrow SN, SN \rightarrow SN \cdot baa\}, baa) = \{SN \rightarrow SN \cdot baa\}$

5. Construct the **Canonical LR(0) Collection** using the following algorithm:

   ```
   void items($G'$) {
      C = CLOSURE([S' -> S]);
      repeat
         for ( each set of items $I$ in C )
            for ( each grammar symbol $X$ )
               if ( $\text{GOTO}(I, X)$ is not empty and not in C )
                  add $\text{GOTO}(I, X)$ to C;
         until no new sets of items are added to C on a round;
   }
   ```

   - $C = \text{CLOSURE}(\{S \rightarrow SN\})$
     - $\rightarrow$ Add $I_0 = \{S \rightarrow SN, SN \rightarrow SN \cdot baa, SN \rightarrow baa\}$ to C
   - **Round 1:** For set $I_0$
     - Grammar symbol $S$:
       - $\text{GOTO}(I_0, S) = \emptyset$, because $S$ is not to the right of any dot
     - Grammar symbol $SN$:
       - $\text{GOTO}(I_0, SN) = CLOSURE(\{S \rightarrow SN \cdot baa\}) = \{S \rightarrow SN \cdot baa\}$
       - $\rightarrow$ Add $I_1 = \{S \rightarrow SN \cdot baa, SN \rightarrow baa\}$ to C
     - Grammar symbol “baa”:
       - $\text{GOTO}(I_0, baa) = CLOSURE(\{SN \rightarrow baa\}) = \{SN \rightarrow baa\}$
       - $\rightarrow$ Add $I_2 = \{SN \rightarrow baa\}$ to C
   - **Round 2:** For set $I_1$
     - Grammar symbol $S$:
       - $\text{GOTO}(I_1, S) = \emptyset$, because $S$ is not to the right of any dot
     - Grammar symbol $SN$:
GOTO(I₁, SN) = ∅, because SN is not to the right of any dot

- Grammar symbol “baa”:
  GOTO(I₁, baa) =
  CLOSURE({SN → SN baa•}) = {SN → SN baa•}
  → Add I₁ = {SN → SN baa•} to C

- Round 2: For set I₂
  - GOTO(I₂, S ) = ∅
  - GOTO(I₂, SN ) = ∅
  - GOTO(I₂, baa ) = ∅

- Round 3: For set I₃
  - GOTO(I₃, S ) = ∅
  - GOTO(I₃, SN ) = ∅
  - GOTO(I₃, baa ) = ∅

- C = {I₀, I₁, I₂, I₃} = {{S → SN, SN → SN baa, SN → baa},
  {S → SN•, SN → SN• baa}, {SN → baa•}, {SN → SN baa•}}

6. Draw the resulting LR(0) Automaton.
  - Because S → SN• is in I₁, add arc GOTO(I₁, $) = ACCEPT
7. Construct the **SLR-Parsing Table** using the following algorithm

1. Construct \( G' = \{I_0, I_1, \ldots, I_n\} \), the collection of sets of LR(0) items for \( G' \).
2. State \( i \) is constructed from \( I_i \). The parsing actions for state \( i \) are determined as follows:
   a. If \([A \rightarrow \alpha a]\) is in \( I_i \) and \( \text{GOTO}(I_i, a) = I_j \), then set \( \text{ACTION}[i, a] \) to “shift \( j \).” Here \( a \) must be a terminal.
   b. If \([A \rightarrow \alpha]\) is in \( I_i \), then set \( \text{ACTION}[i, a] \) to “reduce \( A \rightarrow \alpha \)” for all \( a \) in \( \text{FOLLOW}(A) \); here \( A \) may not be \( S' \).
   c. If \([S' \rightarrow \alpha]\) is in \( I_i \), then set \( \text{ACTION}[i, S'] \) to “accept.”

If any conflicting actions result from the above rules, we say the grammar is not SLR(1). The algorithm fails to produce a parser in this case.

<table>
<thead>
<tr>
<th>STATE</th>
<th>ACTION</th>
<th>$</th>
<th>GOTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift 2</td>
<td>$</td>
<td>0, S</td>
</tr>
<tr>
<td>1</td>
<td>Shift 3</td>
<td>Accept</td>
<td>1, S</td>
</tr>
<tr>
<td>2</td>
<td>Reduce ( SN \rightarrow baa )</td>
<td>Reduce ( SN \rightarrow baa )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reduce ( SN \rightarrow SN baa )</td>
<td>Reduce ( SN \rightarrow SN baa )</td>
<td></td>
</tr>
</tbody>
</table>

- Calculate FOLLOW sets:
  - \( \text{FOLLOW}(S) = \{ S \} \)
  - \( \text{FOLLOW}(SN) = \{ S, baa \} \)
- GOTO columns can be taken from Canonical LR(0) collection or automaton
- Calculation of ACTION cells:
  - For state 0:
    - Rule (a) applies because \( SN \rightarrow baa \) is part of \( I_0 \rightarrow \text{GOTO}(I_0, baa) = I_2 \rightarrow \text{ACTION}(0, baa) = \text{"Shift 2"} \)
  - For state 1:
    - Rule (a) applies because \( SN \rightarrow SN baa \) is part of \( I_1 \rightarrow \text{GOTO}(I_1, baa) = I_3 \rightarrow \text{ACTION}(1, baa) = \text{"Shift 3"} \)
    - Rule (c) applies because \( S \rightarrow SN \) is part of \( I_1 \rightarrow \text{ACTION}(1, S) = \text{ACCEPT} \)
  - For state 2:
    - Rule (b) applies because \( SN \rightarrow baa \) is part of \( I_2 \rightarrow \text{FOLLOW}(SN) = \{ S, baa \} \rightarrow \text{ACTION}(2, S) = \text{ACTION}(2, baa) = \text{"Reduce \( SN \rightarrow baa \)"} \)
  - For state 3:
    - Rule (b) applies because \( SN \rightarrow SN baa \) is part of \( I_3 \rightarrow \text{FOLLOW}(SN) = \{ S, baa \} \rightarrow \text{ACTION}(3, S) = \text{ACTION}(3, baa) = \text{"Reduce \( SN \rightarrow SN baa \)"} \)
8. Is this grammar SLR?
   Yes, because there are no shift/reduce or reduce/reduce conflicts in the parsing table.

9. Show me the actions of the resulting parser for the input string “baa baa baa baa” (bbbb)

<table>
<thead>
<tr>
<th></th>
<th>Stack</th>
<th>Input</th>
<th>Action</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$0$</td>
<td>bbbb$</td>
<td>Shift 2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$0$ b 2</td>
<td>bbb$</td>
<td>Reduce $SN \rightarrow baa$</td>
<td>$SN \rightarrow baa$</td>
</tr>
<tr>
<td>3</td>
<td>$0$ $SN$ 1</td>
<td>bbb$</td>
<td>Shift 3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$0$ $SN$ 1 b 3</td>
<td>bb$</td>
<td>Reduce $SN \rightarrow SN baa$</td>
<td>$SN \rightarrow SN baa$</td>
</tr>
<tr>
<td>5</td>
<td>$0$ $SN$ 1</td>
<td>bb$</td>
<td>Shift 3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$0$ $SN$ 1 b 3</td>
<td>b$</td>
<td>Reduce $SN \rightarrow SN baa$</td>
<td>$SN \rightarrow SN baa$</td>
</tr>
<tr>
<td>7</td>
<td>$0$ $SN$ 1</td>
<td>b$</td>
<td>Shift 3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$0$ $SN$ 1 b 3</td>
<td>$   $</td>
<td>Reduce $SN \rightarrow SN baa$</td>
<td>$SN \rightarrow SN baa$</td>
</tr>
<tr>
<td>9</td>
<td>$0$ $SN$ 1</td>
<td>$   $</td>
<td>Accept</td>
<td></td>
</tr>
</tbody>
</table>

- At line (1) the SLR parser is in state 0, the initial state with no grammar symbols, and with “baa” the first input symbol
- ACTION(0, baa) = “Shift 2” meaning shift by pushing “baa” followed by state 2 onto the stack
- “baa” becomes the new input symbol in line (2).
- ACTION(2, baa) = “Reduce $SN \rightarrow baa$”. The state 2 and “baa” are then popped off the stack.
- $SN$ is pushed onto the stack.
- Since state 0 is exposed and GOTO(0, $SN$) = 1, push state 1 onto the stack
- ...