CSE443 Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Team status updates
Items

- "How does a shift-reduce parser know when to shift and when to reduce?" [p 242]
- "...by maintaining states to keep track of where we are in a parse."
- Each state is a set of items.
- An item is a grammar rule annotated with a dot, •, somewhere on the RHS.
Rules and items

\[
\begin{array}{|c|}
\hline
A \rightarrow X Y Z \\
\hline
A \rightarrow \bullet X Y Z \\
\hline
A \rightarrow X \bullet Y Z \\
\hline
A \rightarrow X Y \bullet Z \\
\hline
A \rightarrow X Y Z \bullet \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
A \rightarrow \varepsilon \\
\hline
A \rightarrow \bullet \\
\hline
\end{array}
\]

The \( \bullet \) shows where in a rule we might be during a parse.
Building the finite control for a bottom-up parser

- Build a finite state machine, whose states are sets of items
- Build a table (M) incorporating shift/reduce decisions
Augment grammar

Given a grammar

\[ G = (N, T, P, S) \]

we augment to a grammar

\[ G' = (N \cup \{S'\}, T, P \cup \{S' \rightarrow S\}, S'), \quad \text{where } S' \notin N \]

\[ G' \] has exactly one rule with \( S' \) on left.
We need two operations to build our finite state machine:

\[ \text{CLOSURE}(I) \]

\[ \text{GOTO}(I,X) \]
I is a set of items

\textbf{CLOSURE(I)} fixed point construction

\[ \text{CLOSURE}_0(I) = I \]

\text{repeat } \{
\quad \text{CLOSURE}_{i+1}(I) = \text{CLOSURE}_i(I) \cup \{ B \rightarrow \gamma \mid A \rightarrow \alpha \beta \in \text{CLOSURE}_i(I) \text{ and } B \rightarrow \gamma \in \mathcal{P} \} \}

\text{until } \text{CLOSURE}_{i+1}(I) = \text{CLOSURE}_i(I) \}
CLOSURE(I)

- I is a set of items
- CLOSURE(I) fixed point construction

Intuition: an item like $A \rightarrow X \cdot YZ$ conveys that we've already seen $X$, and we're expecting to see a $Y$ followed by a $Z$.

The closure of this item is all the other items that are relevant at this point in the parse.

For example, if $Y \rightarrow RST$ is a production, then $Y \rightarrow \cdot RST$ is in the closure because if the next thing in the input can derive from $Y$, it can derive from $R$. 
**GOTO(I, X)**

- **GOTO(I, X)** is the closure of the set of items $A \rightarrow \alpha X \beta$ s.t. $A \rightarrow \alpha X \beta \in I$

- **GOTO(I, X)** construction for $G'$ (figure 4.32):

  set-of-items $\text{CLOSURE}(I) \{$
  
  $J = I$

  repeat \{ 
  
  for each item $A \rightarrow \alpha B \beta \in J$

  for each production $B \rightarrow \gamma \in P$

  if $B \rightarrow \gamma$ not already in $J$

  add $B \rightarrow \gamma$ to $J$

  \}

  until no more items are added to $J$

  return $J$

  \}
Building the LR(0) automaton

void items(G') {
    C = { CLOSURE( { S' \rightarrow \varepsilon S } ) } 
    repeat {
        for each set of items \( I \in C \) and
        for each grammar symbol \( X \in (NUT) \)
        if ( \( \text{GOTO}(I, X) \) is not empty and not already in \( C \) )
            add \( \text{GOTO}(I, X) \) to \( C \)
    } until no new sets of items are added to \( C \)
}
<table>
<thead>
<tr>
<th>Grammar G</th>
<th>Augmented Grammar G'</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S' \rightarrow E$</td>
<td></td>
</tr>
<tr>
<td>$E \rightarrow E + T$</td>
<td>$E \rightarrow E + T$</td>
</tr>
<tr>
<td>$E \rightarrow T$</td>
<td>$E \rightarrow T$</td>
</tr>
<tr>
<td>$T \rightarrow T \ast F$</td>
<td>$T \rightarrow T \ast F$</td>
</tr>
<tr>
<td>$T \rightarrow F$</td>
<td>$T \rightarrow F$</td>
</tr>
<tr>
<td>$F \rightarrow (E)$</td>
<td>$F \rightarrow (E)$</td>
</tr>
<tr>
<td>$F \rightarrow id$</td>
<td>$F \rightarrow id$</td>
</tr>
</tbody>
</table>
**Compute items(G')**

<table>
<thead>
<tr>
<th>SET OF ITEMS (I)</th>
<th>i</th>
<th>CLOSURE_i(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ S' → • E }</td>
<td>0</td>
<td>{ S' → • E }</td>
</tr>
</tbody>
</table>
### Compute items($G'$)

<table>
<thead>
<tr>
<th>SET OF ITEMS (I)</th>
<th>i</th>
<th>CLOSURE$_i$(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ $S' \rightarrow \bullet E$ }</td>
<td>0</td>
<td>{ $S' \rightarrow \bullet E$ }</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CLOSURE$_0$(I) $\cup$ { $E \rightarrow \bullet E + T$, $E \rightarrow \bullet T$ }</td>
</tr>
</tbody>
</table>

### Grammar Rules

- $S' \rightarrow E$
- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T * F$
- $T \rightarrow F$
- $F \rightarrow (E)$
- $F \rightarrow \text{id}$
Compute items($G'$)

<table>
<thead>
<tr>
<th>SET OF ITEMS (I)</th>
<th>i</th>
<th>CLOSURE$_i$ (I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ $S' \rightarrow \bullet E$ }</td>
<td>0</td>
<td>{ $S' \rightarrow \bullet E$ }</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CLOSURE$_0$ (I) $\cup$ { $E \rightarrow \bullet E + T$ , $E \rightarrow \bullet T$ }</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CLOSURE$_1$ (I) $\cup$ { $T \rightarrow \bullet T \ast F$ , $T \rightarrow \bullet F$ }</td>
</tr>
</tbody>
</table>

- $S' \rightarrow E$
- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T \ast F$
- $T \rightarrow F$
- $F \rightarrow (E)$
- $F \rightarrow id$
### Compute items($G'$)

<table>
<thead>
<tr>
<th>SET OF ITEMS (I)</th>
<th>i</th>
<th>CLOSURE$_i$(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ S' \to \bullet E }$</td>
<td>0</td>
<td>${ S' \to \bullet E }$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$\text{CLOSURE}_0(I) \cup { E \to \bullet E + T, E \to \bullet T }$</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$\text{CLOSURE}_1(I) \cup { T \to \bullet T \ast F, T \to \bullet F }$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$\text{CLOSURE}_2(I) \cup { F \to \bullet (E), F \to \bullet \text{id} }$</td>
</tr>
</tbody>
</table>
## Compute items($G'$)

<table>
<thead>
<tr>
<th>SET OF ITEMS (I)</th>
<th>i</th>
<th>CLOSURE&lt;sub&gt;i&lt;/sub&gt;(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ S' -&gt; • E }</td>
<td>0</td>
<td>{ S' -&gt; • E }</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CLOSURE&lt;sub&gt;0&lt;/sub&gt;(I) ∪ { E -&gt; • E + T, E -&gt; • T }</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CLOSURE&lt;sub&gt;1&lt;/sub&gt;(I) ∪ { T -&gt; • T * F, T -&gt; • F }</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CLOSURE&lt;sub&gt;2&lt;/sub&gt;(I) ∪ { F -&gt; • ( E ), F -&gt; • id }</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CLOSURE&lt;sub&gt;3&lt;/sub&gt;(I) ∪ ∅</td>
</tr>
</tbody>
</table>
Terminology

- Kernel items: $S' \rightarrow \bullet S$ and all items with $\bullet$ not at left edge

- Non-kernel items: all items with $\bullet$ at left edge, except $S' \rightarrow \bullet S$
This gives us the first state of the finite state machine, $I_0$.

<table>
<thead>
<tr>
<th>$S' \rightarrow \ast \ E$</th>
<th>kernel item</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \rightarrow \ast \ E + T$</td>
<td>non-kernel items are computed from CLOSURE(kernel), and therefore do not need to be explicitly stored</td>
</tr>
<tr>
<td>$E \rightarrow \ast \ T$</td>
<td></td>
</tr>
<tr>
<td>$T \rightarrow \ast \ T \ast \ F$</td>
<td></td>
</tr>
<tr>
<td>$T \rightarrow \ast \ F$</td>
<td></td>
</tr>
<tr>
<td>$F \rightarrow \ast \ ( \ E \ )$</td>
<td></td>
</tr>
<tr>
<td>$F \rightarrow \ast \ id$</td>
<td></td>
</tr>
</tbody>
</table>
Next we compute \( \text{GOTO}(I_0, X) \forall X \in N \cup T \)

\[ N \cup T = \{ E, T, F, +, *, (, ), \text{id} \} \]

N.B. - augmented start symbol \( S' \) can be ignored

\[
\text{GOTO}(I_0, E) = \text{CLOSURE} ( \{ S' \to E \circ, E \to E \circ + T \} ) \\
= \{ S' \to E \circ, E \to E \circ + T \} \\
\]

N.B. there is no non-terminal after the \( \circ \), so no new items are added by CLOSURE operation

\[
I_1 \begin{array}{c}
S' \to E \circ \\
E \to E \circ + T
\end{array}
\]

only kernel items
\[ \text{GOTO}(I_0, T) = \text{CLOSURE}( \{ E \rightarrow T \odot, T \rightarrow T \odot \ast F \} ) \]

\[ = \{ E \rightarrow T \odot, T \rightarrow T \odot \ast F \} \]

N.B. there is no non-terminal after the \( \odot \), so no new items are added by CLOSURE operation

**only kernel items**
\[ \text{GOTO}(I_0, F) = \text{CLOSURE} \left( \{ T \rightarrow F \, \bullet \} \right) \]

\[ = \{ T \rightarrow F \, \bullet \} \]

N.B. there is no non-terminal after the \( \bullet \), so no new items are added by CLOSURE operation.
GOTO(I₀, '(') = CLOSURE( { F → ( • E ) } )

= { F → ( • E ) } ∪ { E → • E + T, E → • T } ∪ { T → • T * F, T → • F } ∪ { F → • ( E ), F → • id }

---

N.B. there is a non-terminal after the •, so new items are added by CLOSURE operation.
$\text{GOTO}(I_0, \text{id}) = \text{CLOSURE}(\{ F \rightarrow \text{id} \cdot \})$

$= \{ F \rightarrow \text{id} \cdot \}$

\[ \text{N.B. there is no non-terminal after the } \cdot, \text{ so no new items are added by CLOSURE operation} \]

\[ \text{only kernel items} \]

\[ \text{Is} \]

$F \rightarrow \text{id} \cdot$

$\text{GOTO}(I_0, \text{'} ) = \text{GOTO}(I_0, + ) = \text{GOTO}(I_0, * ) = \text{GOTO}(I_0, $ ) = \emptyset $
The finite state machine as at this point.

EXERCISE: complete the machine by computing GOTO(I_k,X) until no new states are added.
The finite state machine as at this point.

EXERCISE: complete the machine by computing GOTO(I_k,X) until no new states are added.
Compute GOTO(I_1, X) for each X in \{ +, *, '(', ')', id, E, T, F, $ \}