**Constructing SLR Parsing Tables**

- **SLR** = “Simple LR” - weakest, but easiest to implement, LR method
- **SLR grammar** is one for which an SLR parser can be constructed
- **LR(0) item** is a grammar production with a dot somewhere in the RHS. E.g., for $A \rightarrow XYZ$
  - $A \rightarrow \bullet XYZ$
  - $A \rightarrow X\bullet YZ$
  - $A \rightarrow X\bullet YZ$
  - $A \rightarrow XYZ$

---

**Building the Canonical LR(0) Collection**

- Augment grammar with a new production $S' \rightarrow S$ where $S$ is start symbol (serves a similar role to $\$$)
- Initial item is $S' \rightarrow \bullet S$
- Compute closure and goto functions.
- **Closure**: Given a set of items $I$
  1. All items in $I$ are initially in $\text{closure}(I)$
  2. If $A \rightarrow \alpha B\beta$ is in $\text{closure}(I)$ and $B \rightarrow \gamma$ is a production, add $B \rightarrow \gamma$ to $\text{closure}(I)$, if it is not already there.
  3. Repeat until no new items are generated
Building the Canonical LR(0) Collection

Example: \[ E \rightarrow E + T \mid T \]
\[ T \rightarrow T \ast F \mid F \]
\[ F \rightarrow (E) \mid \text{id} \]

Augmented: \[ E' \rightarrow E \]
\[ E' \rightarrow E + T \mid T \]
\[ T \rightarrow T \ast F \mid F \]
\[ F \rightarrow (E) \mid \text{id} \]

Building the Canonical LR(0) Collection: \textit{Closure}

Initial \textit{closure}(I): \{ [E \rightarrow \varepsilon E] \}

Matches: \[ A \rightarrow \alpha \beta \beta \quad (\alpha = \beta = \varepsilon) \]

Add: \[ E \rightarrow \varepsilon E + T \]
\[ E \rightarrow \varepsilon T \]

New \textit{closure}(I):
\[
\{ [E \rightarrow \varepsilon E], [E \rightarrow \varepsilon E + T], [E \rightarrow \varepsilon T] \}
\]

Final \textit{closure}(I):
\[
\{ [E \rightarrow \varepsilon E], [E \rightarrow \varepsilon E + T], [E \rightarrow \varepsilon T],
\quad [T \rightarrow \varepsilon T \ast F], [T \rightarrow \varepsilon F], [F \rightarrow \varepsilon E],
\quad [F \rightarrow \varepsilon \text{id}] \}
\]

Building the Canonical LR(0) Collection: \textit{Goto}

- \textit{Goto}(I, X) takes a set of items I and a symbol X and returns the closure of the set of all items \[ A \rightarrow \alpha \beta \] such that \[ A \rightarrow \alpha \beta \] is in I. (Like \( \in \text{-closure(move(a))} \) in Subset Construction)

- Intuitively: “If you have item \[ A \rightarrow \alpha \beta \] and you see symbol X, move the dot to the right of the X”.

Building the Canonical LR(0) Collection: \textit{Goto}

- E.g., \( I = \{ [E \rightarrow E_1], [E \rightarrow E_2 + T] \} \)

- Matches: \[ A \rightarrow \alpha \beta \]

- So add \[ [E \rightarrow E + T] \] to \textit{goto}(I, +)

- \textit{goto}(I, +) = \{ [E \rightarrow E + T] \}

- Then add closure of \[ [E \rightarrow E + T] \]:

\[
\text{goto}(I, +) = \{ [E \rightarrow E + T], [T \rightarrow \varepsilon T \ast F], [T \rightarrow \varepsilon F],
\quad [F \rightarrow \varepsilon E], [F \rightarrow \varepsilon \text{id}] \}
\]
Building the Canonical LR(0) Collection: Sets of Items

**procedure** $\text{items}(G')$

**begin**

$C := \{ \text{closure}( \{ [S' \rightarrow \bullet S] \} ) \}$

**repeat**

for each set of items $I$ in $C$ and each grammar symbol $X$ such that $\text{goto}(I, X)$ is not empty and not in $C$

add $\text{goto}(I, X)$ to $C$

**until** no new sets of items can be added to $C$

**end**

**return** $C$

---

Building the Canonical LR(0) Collection: Sets of Items

Compute $C$ for:

- $E' \rightarrow E$
- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T \ast F$
- $T \rightarrow F$
- $F \rightarrow (E) | \text{id}$

---

Sets of Items as DFA

[Diagram showing transitions between states for sets of items]

---

Construcing the SLR Parsing Table

**Input:** An augmented grammar $G'$.

**Output:** The SLR parsing table (action and goto) for $G'$

**Method:**

1. Construct $C = \{ I_0, I_1, ... I_n \}$, 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:
Constructing the SLR Parsing Table

2.  
   (a) If $[A \rightarrow \alpha \beta]$ is in $I$ and $\text{goto}(I, a) = I$, then set action[$i,a$] to “shift $j$”.

   (b) If $[A \rightarrow \alpha]$ then set action[$i,a$] to “reduce $A \rightarrow \alpha$” for all $a$ in FOLLOW($A$), except where $A = S'$.

3. The goto transitions for state $i$ are constructed for all nonterminals $A$ using the rule: If goto($I, A$) = $I$, then $\text{goto}(i, A) = j$.

Constructing the SLR Parsing Table

4. All entries not defined by rules (2) and (3) are made “error”.

5. The initial state of the parser is the one constructed from the set of items containing $[S \rightarrow \epsilon S]$. 