LR Parsing - The Items

Lecture 10
Sections 4.5, 4.7

Robb T. Koether

Hampden-Sydney College

Mon, Feb 16, 2009
Outline

1. LR Parsers
2. LR(0) Items
3. Building the PDA
4. Assignment
A bottom-up parser follows a rightmost derivation from the bottom up.

Such parsers typically use an LR algorithm and are called LR parsers.

L means process tokens from Left to right.

R means follow a Rightmost derivation.
Furthermore, in LR parsing, the production is applied only after the pattern has been matched.

In LL (predictive) parsing, the production was selected, and then the tokens were matched to it.
Example (Rightmost Derivations)

Let the grammar be

\[
\begin{align*}
E & \rightarrow E + T | T \\
T & \rightarrow T \times F | F \\
F & \rightarrow (E) | \text{id} | \text{num}
\end{align*}
\]
Example

Example (Rightmost Derivation of $(\text{id} + \text{num}) \ast \text{id}$)

\[
E \Rightarrow T
\Rightarrow T \ast F
\Rightarrow T \ast \text{id}
\Rightarrow F \ast \text{id}
\Rightarrow (E) \ast \text{id}
\Rightarrow (E + T) \ast \text{id}
\Rightarrow (E + F) \ast \text{id}
\Rightarrow (E + \text{num}) \ast \text{id}
\Rightarrow (T + \text{num}) \ast \text{id}
\Rightarrow (F + \text{num}) \ast \text{id}
\Rightarrow (\text{id} + \text{num}) \ast \text{id}
\]
An LR parser uses a PDA.
The parser controls the PDA with
- A parse table (transition function).
- An input buffer.
- A stack of “states.”
The parser performs three types of operation.

- **Shift** a token from the input buffer to the stack.
- **Reduce** the content of the stack by applying a production.
- **Go to** a new state.
LR(0) Items

Definition (LR(0) item)

An LR(0) item is a production with a special marker (●) marking a position within the string on the right side of the production.

- To build an LR parse table, we must first find the LR(0) items.
- LR(0) parsing is also called SLR parsing (Simple LR).
Example (LR(0) Items)

- If the production is

\[ E \rightarrow E + T, \]

then the possible LR(0) items are

- \([E \rightarrow \bullet E + T]\)
- \([E \rightarrow E \bullet + T]\)
- \([E \rightarrow E + \bullet T]\)
- \([E \rightarrow E + T \bullet]\)
The interpretation of \([A \rightarrow \alpha \cdot \beta]\) is

“We have processed \(\alpha\) and we might process \(\beta\) next.”

Whether we do actually process \(\beta\) will be borne out by the subsequent tokens.
We will build a PDA whose states are sets of LR(0) items.

First we augment the grammar with a new start symbol \( S' \).

\[
S' \rightarrow S
\]

This guarantees that the start symbol will not recurse.
States of the PDA

- The initial state is called $I_0$ (item 0).
- State $I_0$ is the closure of the set
  \[
  \{[S' \to \bullet S]\}. 
  \]

- To form the closure of a set of items
  - For each item $[A \to \alpha \bullet B\beta]$ in the set and for each production $B \to \gamma$ in the grammar, add the item $[B \to \bullet \gamma]$ to the set.
  - Continue in this manner until there is no further change.
- We will call $[B \to \bullet \gamma]$ an initial $B$-item.
Example (LR Parsing)

Continuing with our example, the augmented grammar is

\[
E' \rightarrow E \\
E \rightarrow E + T \mid T \\
T \rightarrow T \ast F \mid F \\
F \rightarrow (E) \mid \text{id} \mid \text{num}
\]
Example (The PDA States)

- The state $I_0$ consists of the items in the closure of item $[E' \rightarrow \bullet E]$.

$$I_0 = \{ [E' \rightarrow \bullet E], [E \rightarrow \bullet E + T], [E \rightarrow \bullet T],
[T \rightarrow \bullet T * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)],
[F \rightarrow \bullet \text{id}], [F \rightarrow \bullet \text{num}] \}$$
Transitions

- There will be a transition from one state to another state for each grammar symbol that immediately follows the marker \( \bullet \) in an item in that state.
- If the item \( [A \rightarrow \alpha \bullet X \beta] \) is in the state, then
  - A transition from that state occurs when the symbol \( X \) is processed.
  - The transition is to the state that is the closure of the item \( [A \rightarrow \alpha X \bullet \beta] \).
Example

Example (Building the PDA)

- Thus, from the state $I_0$, there will be transitions for the symbols $E$, $T$, $F$, $($, $\text{id}$, and $\text{num}$.
- For example, on processing $E$, the items

$$[E' \rightarrow \bullet E] \text{ and } [E \rightarrow \bullet E + T]$$

become

$$[E' \rightarrow E \bullet] \text{ and } [E \rightarrow E \bullet + T].$$
Example (Building the PDA)

- Let state $I_1$ be the closure of these items.

$$I_1 = \{[E' \rightarrow E \bullet], [E \rightarrow E \bullet + T]\}.$$

- Then the PDA has the transition

\[ I_0 \xrightarrow{E} I_1 \]
Example (Building the PDA)

- Similarly we determine the other transitions from $I_0$.
- **Process** $T$:

  \[ I_2 = \{ [E \to T \bullet], [T \to T \bullet \ast F] \} \].

- **Process** $F$:

  \[ I_3 = \{ [T \to F \bullet] \} \].
Example

Example (Building the PDA)

- **Process (**:  
  \[ I_4 = \{[F \rightarrow ( \bullet E)], [E \rightarrow \bullet E + T], [E \rightarrow \bullet T], [T \rightarrow \bullet T * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet id], [F \rightarrow \bullet num] \}. \]

- **Process id:**  
  \[ I_5 = \{[F \rightarrow id \bullet]\}. \]

- **Process num:**  
  \[ I_6 = \{[F \rightarrow num \bullet]\}. \]
Example

Example (Building the PDA)

- Now find the transitions from states $I_1$ through $I_6$ to other states, and so on, until no new states appear.
Example (Building the PDA)

\[
I_7 = \{ [E \to E + \bullet T], [T \to \bullet T \ast F], [T \to \bullet F], [F \to \bullet (E) ], [F \to \bullet id], [F \to \bullet num] \}
\]

\[
I_8 = \{ [T \to T \ast \bullet F], [F \to \bullet (E) ], [F \to \bullet id], [F \to \bullet num] \}
\]

\[
I_9 = \{ [F \to (E \bullet )], [E \to E \bullet + T] \}
\]

\[
I_{10} = \{ [E \to E + T \bullet], [T \to T \bullet \ast F] \}
\]

\[
I_{11} = \{ [T \to T \ast F \bullet] \}
\]

\[
I_{12} = \{ [F \to (E) \bullet] \}
\]
Assignment

Homework

- The grammar

\[ R \rightarrow R \cup R \mid RR \mid R^* \mid (R) \mid a \mid b \]

generates all regular expressions on the alphabet \( \Sigma = \{a, b\} \).

- Write the LR(0) items for this grammar.