Outline

- Recap
  - LR(0) parsing and SLR(1) parsing
  - General/Canonical LR(1) parsing
  - Lookahead LR(1) / LALR(1) parsing
- Summary and homework
LR Parsing: A Schematic View

- Bottom-up parsing
  - Rightmost derivations and right-sentential forms

<table>
<thead>
<tr>
<th>Parsing stack</th>
<th>Input buffer</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>InputString$</td>
<td>lookahead zero or one token, decide S/R</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$StartSymbol</td>
<td>$</td>
<td>accept</td>
</tr>
</tbody>
</table>

Instructor: Dr. Liang Cheng
CSE302: Compiler Design
03/20/07
Finite Automata for Both LR(0) and SLR(1) Parsing

- Finite automata of parsing states
  - LR(0) items are used to identify the parsing states
    - $A \rightarrow \alpha$
      - $A \rightarrow .\alpha$ is an item (initial item)
      - We may about to recognize $A$ by $A \rightarrow \alpha$
      - $A \rightarrow \alpha$ is also an item (complete item)
      - $\alpha$ may be a handle for reduction
    - $A \rightarrow \beta\gamma$
      - $A \rightarrow \beta.\gamma$, $A \rightarrow .\beta\gamma$, and $A \rightarrow \beta\gamma$ are LR(0) items
  - NFA construction
  - Subset construction for NFA to DFA
LR(0) Parser NFA Construction

- \[ A \rightarrow \alpha \cdot X \beta \rightarrow A \rightarrow \alpha X \beta \]
  - Shift action if \( X \) is a terminal

- \[ A \rightarrow \alpha \cdot X \beta \rightarrow X \rightarrow \gamma \]
  - Reduction action if \( X \) is a non-terminal
The LR(0) and SLR(1) Parsing Algorithms

- **LR(0) parsing**
  - If state $s$ contains $A \rightarrow \alpha.X\beta$ where $X$ is a terminal, then shift and the state changes to $s'$ containing $A \rightarrow \alpha.X\beta$
  - If state $s$ contains $A \rightarrow \gamma.$, then reduce by $A \rightarrow \gamma$ (stack ops) and the state changes to $s'$ containing $B \rightarrow \lambda.A.\eta$

- **SLR(1) parsing**
  - If state $s$ contains $A \rightarrow \alpha.X\beta$ where $X$ is a terminal, and the lookahead token is $X$, then shift and the state changes to $s'$ containing $A \rightarrow \alpha.X\beta$
  - If state $s$ contains $A \rightarrow \gamma.$, and the lookahead token is in FOLLOW($A$), then reduce by $A \rightarrow \gamma$ (stack ops) and the state changes to $s'$ containing $B \rightarrow \lambda.A.\eta$

- **Examples**
  - $S \rightarrow (S)S | \varepsilon$ Input: ()
  - $A \rightarrow (A) | a$ Input: ((a))
Limits of LR(0) and SLR(1) Parsing

- LR(0) parsing cannot handle a grammar that in its DFA there is a state \( s \)
  - \( s \) contains a shift item \( A \rightarrow \alpha \cdot \lambda \beta \) and a complete item \( B \rightarrow \delta \).
  - \( s \) contains two complete items \( A \rightarrow \gamma \). and \( B \rightarrow \delta \).

- SLR(1) parsing cannot handle a grammar that in its DFA there is a state \( s \)
  - \( s \) contains a shift item \( A \rightarrow \alpha \cdot \lambda \beta \) with \( \lambda \) a terminal and a complete item \( B \rightarrow \delta \). with \( \lambda \) in \( \text{Follow}(B) \)
  - \( s \) contains two complete items \( A \rightarrow \gamma \). and \( B \rightarrow \delta \). with a nonempty \( \text{Follow}(A) \cap \text{Follow}(B) \)
Observations

- SLR(1) parsing is more powerful than LR(0) parsing due to its consideration of lookaheads in the parsing process.
Another Example

- Another example
  - stmt → call-stmt | assign-stmt
  - call-stmt → identifier
  - assign-stmt → var=expr
  - var → identifier
  - expr → var | number

- Is this an SLR(1) grammar?

- An equivalent grammar
  - S → id | V=E
  - V → id
  - E → V | n
Observations

- SLR(1) parsing is more powerful than LR(0) parsing due to its consideration of lookaheads in the parsing process
  - However, the lookaheads are not used in the finite automata construction
- The limit of SLR(1) parsing can be improved if its NFA/DFA construction does not ignore lookaheads
Outline

- Recap
  - LR(0) parsing and SLR(1) parsing
- General/Canonical LR(1) parsing
- Lookahead LR(1) / LALR(1) parsing
- Summary and homework
Finite Automata of Parsing States

- Finite automata for LR(1) parsers
  - LR(1) items are used to identify the parsing states
    - An LR(1) item is a pair consisting of an LR(0) item and a lookahead token
      - \([A \rightarrow \alpha \cdot \beta, a]\)
  - NFA construction: transitions between LR(1) items
    - Non-\(\varepsilon\) transitions
      - Given an LR(1) item \([A \rightarrow \alpha \cdot X \beta, a]\), where \(X\) is any symbol, there is a transition on \(X\) to the item \([A \rightarrow \alpha X \beta, a]\)
    - \(\varepsilon\)-transitions
      - Given an LR(1) item \([A \rightarrow \alpha \cdot B \gamma, a]\), where \(B\) is a nonterminal, there are \(\varepsilon\)-transitions to items \([B \rightarrow \cdot \beta, b]\) for every production \(B \rightarrow \beta\) and for every token \(b\) in \(\text{First}(\gamma a)\)
Grammar:
- \( S' \rightarrow S \)
- \( S \rightarrow \text{id} | V = E \)
- \( V \rightarrow \text{id} \)
- \( E \rightarrow V | n \)

NFA construction: transitions between LR(1) items
- Non-\( \epsilon \)-transitions
  - Given an LR(1) item \([A \rightarrow \alpha.X\beta, a]\), where \( X \) is any symbol, there is a transition on \( X \) to the item \([A \rightarrow \alpha.X\beta, a]\)
- \( \epsilon \)-transitions
  - Given an LR(1) item \([A \rightarrow \alpha.B\gamma, a]\), where \( B \) is a nonterminal, there are \( \epsilon \)-transitions to items \([B \rightarrow .\beta, b]\) for every production \( B \rightarrow \beta \) and for every token \( b \) in First(\( \gamma a \))

Input: \( \text{id}=n \)
LR(1) Parsing Algorithm

- **LR(1) parsing**
  - If state $s$ contains $[A \rightarrow \alpha.X\beta,a]$ where $X$ is a terminal, and the lookahead token is $X$, then shift and the state changes to $s'$ containing $[A \rightarrow \alpha.X\beta,a]$
  - If state $s$ contains $[A \rightarrow \gamma,a]$, and the lookahead token is $a$, then reduce by $A \rightarrow \gamma$ (stack ops) and the state changes to $s'$ containing $[B \rightarrow \lambda.A.\eta,b]$

- **LR(0) parsing**
  - If state $s$ contains $A \rightarrow \alpha.X\beta$ where $X$ is a terminal, then shift and the state changes to $s'$ containing $A \rightarrow \alpha.X\beta$
  - If state $s$ contains $A \rightarrow \gamma$, then reduce by $A \rightarrow \gamma$ (stack ops) and the state changes to $s'$ containing $B \rightarrow \lambda.A.\eta$

- **SLR(1) parsing**
  - If state $s$ contains $A \rightarrow \alpha.X\beta$ where $X$ is a terminal, and the lookahead token is $X$, then shift and the state changes to $s'$ containing $A \rightarrow \alpha.X\beta$
  - If state $s$ contains $A \rightarrow \gamma$, and the lookahead token is in $\text{FOLLOW}(A)$, then reduce by $A \rightarrow \gamma$ (stack ops) and the state changes to $s'$ containing $B \rightarrow \lambda.A.\eta$
Another LR(1) DFA and Parsing Table Construction Examples

- **Grammar**
  - $A' \rightarrow A$
  - $A \rightarrow (A)$
  - $A \rightarrow a$

- **NFA construction: transitions between LR(1) items**
  - **Non-$\epsilon$ transitions**
    - Given an LR(1) item $[A \rightarrow \alpha . X \beta, a]$, where $X$ is any symbol, there is a transition on $X$ to the item $[A \rightarrow \alpha . X \beta, a]$.
  - **$\epsilon$-transitions**
    - Given an LR(1) item $[A \rightarrow \alpha . B \gamma, a]$, where $B$ is a nonterminal, there are $\epsilon$-transitions to items $[B \rightarrow . \beta, b]$ for every production $B \rightarrow \beta$ and for every token $b$ in $\text{First}(\gamma a)$.

- **Compare the LR(1) DFA with the LR(0) DFA**
Outline

- Recap
  - LR(0) parsing and SLR(1) parsing
  - General/Canonical LR(1) parsing
  - Lookahead LR(1) / LALR(1) parsing
- Summary and homework
Two Principles of LALR(1) Parsing

- The **core** of a state in LR(1) DFA is a state in the LR(0) DFA.

- Given two states $s_1$ and $s_2$ in the LR(1) DFA that have the same core. Suppose there is a transition on $X$ from $s_1$ to a state $t_1$. Then there is also a transition on $X$ from $s_2$ to a state $t_2$, and $t_1$ and $t_2$ have the same core.

- Therefore based on LR(1) DFA, we can transform it to a DFA that is identical to the LR(0) DFA, except that each state consists of items with sets of lookaheads.
Constructing LALR(1) DFA

- Identifying all states that have the same core and forming the union of the lookaheads for each LR(0) item
- Linking the new states based on the links in the LR(1) DFA
- An example
  - $A' \rightarrow A$
  - $A \rightarrow (A)$
  - $A \rightarrow a$
Outline

- Recap
- Bottom-up parsing (Section 4.5)
- Summary and homework