Configuration and CFSM

Xiaokang Qiu
Purdue University
Parsing using an LR(0) parser

• How to construct an LR(0) parser?
• How to determine the states and the goto/action tables?
• Basic idea: a state keeps track, simultaneously, of all possible productions that *could be matched* given what it’s seen so far. When it sees a full production, match it.
Terminology for LR parsers

• Configuration: a production augmented with a “•”

• \( A \to X_1 \ldots X_i \cdot X_{i+1} \ldots X_j \)

• The “•” marks the point to which the production has been recognized. In this case, we have recognized \( X_1 \ldots X_i \)

• Configuration set: all the configurations that can apply at a given point during the parse:

  • \( A \to B \cdot CD \)
  • \( A \to B \cdot GH \)
  • \( T \to B \cdot Z \)

• Idea: every configuration in a configuration set is a production that we could be in the process of matching
Configuration closure set

• Include all the configurations necessary to recognize the next symbol after the •

• For each configuration in set:
  • If next symbol is terminal, no new configuration added
  • If next symbol is non-terminal X, for each production of the form X → α, add configuration X → •α

 closure0({S → • E $}) =
{ S → • E $
 E → • E + T
 E → • T
 T → • ID
 T → • (E) }
Successor configuration set

- Starting with the initial configuration set
- \( s_0 = \text{closure}_0(\{S \rightarrow \cdot \alpha \}$)
- an LR(0) parser will find the successor given the next symbol \( X \)
- \( X \) can be either a terminal (the next token from the scanner) or a non-terminal (the result of applying a reduction)
- Determining the successor \( s' = \text{go}_0(s, X) \):
  - For each configuration in \( s \) of the form \( A \rightarrow \beta \cdot X \gamma \) add \( A \rightarrow \beta X \cdot \gamma \) to \( t \)
  - \( s' = \text{closure}_0(t) \)
CFSM

• CFSM = Characteristic Finite State Machine

• Nodes are configuration sets (starting from s0)

• Arcs are go_to relationships
Building the goto table

- We can just read this off from the CFSM
Building the action table

• Given the configuration set $s$:
  • We shift if the next token matches a terminal after the • in some configuration
  • $A \rightarrow \alpha \cdot a \beta \in s$ and $a \in V_t$, else error
  • We reduce production $P$ if the • is at the end of a production
  • $B \rightarrow \alpha \cdot \in s$ where production $P$ is $B \rightarrow \alpha$
  • Extra actions:
    • shift if goto table transitions between states on a non-terminal
    • accept if we have matched the goal production
<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift</td>
</tr>
<tr>
<td>1</td>
<td>Reduce 2</td>
</tr>
<tr>
<td>2</td>
<td>Shift</td>
</tr>
<tr>
<td>3</td>
<td>Accept</td>
</tr>
</tbody>
</table>