Plan for Today

Building the LR Parsing Table for LR(1)
- First and Follow sets
- building the table

Debugging shift/reduce and reduce/reduce errors

FIRST and FOLLOW sets

FIRST( gamma )
- gamma is a string of terminals and nonterminals
- FIRST(gamma) is any nonterminals that can start a string derived from gamma

nullable(X)
- X is a nonterminal
- nullable(X) is true if X can derive the empty string

FOLLOW( X )
- X is a nonterminal
- FOLLOW(X) is all terminals that can immediately follow X

Building the LR Parse Table for LR(1), Grammar 3.23 in book

<table>
<thead>
<tr>
<th>Symbol</th>
<th>FIRST</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>g2</td>
</tr>
<tr>
<td>T</td>
<td>g1</td>
</tr>
<tr>
<td>x</td>
<td>s3</td>
</tr>
<tr>
<td>+</td>
<td>x</td>
</tr>
<tr>
<td>T</td>
<td>r2</td>
</tr>
<tr>
<td>E</td>
<td>r3</td>
</tr>
<tr>
<td>S</td>
<td>g2</td>
</tr>
<tr>
<td>x</td>
<td>g5</td>
</tr>
</tbody>
</table>

Example Ambiguous Grammars: SableCC errors

Productions
stm = exp ;
exp =
  {minus_rule} exp minus exp
| {num_rule} num

Verifying identifiers.

Productions
stm = exp ;
exp =
  {minus_rule} [left]:exp minus [right]:exp
| {num_rule} num

shift/reduce conflict in state [stack: PExp TMinus PExp] on TMinus in {
  [PExp = PExp * TMinus PExp ] (shift),
  [PExp = PExp TMinus PExp * ] followed by TMinus (reduce)