Compilers
Simple Precedence
Simple Precedence

• Less restricted than Operator precedence
  – We may have adjacent not-terminals on rhs

• More accurate than Operator Precedence
  – All syntax errors are found
  – However, unlike LL(1) and LR(1), illegal inputs may be shifted onto the stack before the parser recognizes the error

• Precedence relations are defined between all symbols (i.e., both terminals and non-terminals)

• Restrictions:
  – no production right side is $\in$
  – Rules cannot have the same rhs.
For all symbols of the grammar, define the relations $\preceq$, $\succeq$ and $\equiv$

- $X \preceq Y$ means that $X$ yields precedence to $Y$
- $X \equiv Y$ means that $X$ has the same precedence as $Y$
- $X \succeq Y$ means that $X$ takes precedence over $Y$

Similar to OP, we will use these relations to find the correct handle
Parsing:

- Let $X$ be the top most symbol in stack and $b$ be the current token,
- Look up their precedence relation, and decide what to do next:
  - If $X \equiv b$, then shift $b$ into the parse stack
  - If $X \prec b$, then shift $\prec$ and then shift $b$ into the parse stack
  - If $X \succ b$, then find the top most $\prec$ relation of the parse stack; the string between this relation and the top of the stack is the handle (the handle should match (exactly) with the RHS of at least one grammar rule); let $top$ be the top symbol of the stack after deleting the handle; let $LHS$ be the left hand of the rule whose rhs has matched the handle; look up the precedence relation between $top$ and $LHS$ and perform the following:
Parsing (Cont.):

- If $\text{top} \equiv \text{LHS}$, then shift $\text{LHS}$ into the parse stack
- If $\text{top} < \text{LHS}$, then shift $<$ and then shift $\text{LHS}$ into the parse stack
- If $\text{top} > \text{LHS}$, a syntax error has occurred!
- In fact, no symbol can have a higher precedence than a non-terminal (Why?)
### Simple Precedence

#### Parse Table

<table>
<thead>
<tr>
<th>STACK</th>
<th>INPUT</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>( c c ) $</td>
<td>$ &lt; c$, shift</td>
</tr>
<tr>
<td>$ &lt; (</td>
<td>c c ) $</td>
<td>( &lt; c, shift</td>
</tr>
<tr>
<td>$ &lt; ( &lt; c</td>
<td>c ) $</td>
<td>c &gt; c, reduce, handle is c</td>
</tr>
<tr>
<td>$ &lt; ( S</td>
<td>c ) $</td>
<td>S &lt; c, shift</td>
</tr>
<tr>
<td>$ &lt; ( S &lt; c</td>
<td>c ) $</td>
<td>c &gt; ), reduce, handle is c</td>
</tr>
<tr>
<td>$ &lt; ( S S</td>
<td>) $</td>
<td>S = ), shift</td>
</tr>
<tr>
<td>$ S</td>
<td>$</td>
<td>&gt; $, reduce, handle is ( S S ) accept</td>
</tr>
</tbody>
</table>

These should be deleted; because, nothing can be greater than a non-terminal

<table>
<thead>
<tr>
<th>S</th>
<th>(</th>
<th>c</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>&lt;</td>
<td>=</td>
<td>&lt;</td>
</tr>
<tr>
<td>(</td>
<td>=</td>
<td>&lt;</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>&gt;</td>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>c</td>
<td>&gt;</td>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>$</td>
<td>=</td>
<td>&lt;</td>
<td></td>
</tr>
</tbody>
</table>

Parse Table

\[ S \rightarrow ( S S ) | c \]
Producing the parse table

• Head(A) = \{X \mid A \Rightarrow^+ X\alpha\}
• Tail(A) = \{X \mid A \Rightarrow^+ \alpha X\}

• X \equiv Y \text{ iff } \exists U \rightarrow \alpha X Y \beta

• X \triangleleft Y \text{ iff } \exists U \rightarrow \alpha X B \beta \text{ and } Y \in \text{Head}(B)

• X \triangleright Y \text{ iff } \exists U \rightarrow \alpha B Y \beta \text{ and } X \in \text{Tail}(B) \text{ or } \
  \exists U \rightarrow \alpha A B \beta \text{ and } X \in \text{Tail}(A) \text{ and } Y \in \text{Head}(B)
Example:

- Head (E) = \{E, T, F, id, ()\}
- Head (T) = \{T, F, id, ()\}
- Head (F) = \{id, ()\}

- Tail (E) = \{T, F, id, ))\}
- Tail (T) = \{F, id, ))\}
- Tail (F) = \{id, ))\}

1-2 \ E \rightarrow \ E + T \ | \ T
3-4 \ T \rightarrow T * F \ | \ F
5-6 \ F \rightarrow (E) \ | \ id
Problem with Left and Right recursions

- If $\exists U \rightarrow U \gamma$ and $\exists U \rightarrow \alpha X U \beta$ then there would be a problem in the parsing table

  - $X \equiv U$ and $X \preccurlyeq U$

- The problem can be resolved by introducing a new non-terminal $W$, and a new rule

  \[ W \rightarrow U, \text{ and change the initial rule to } U \rightarrow \alpha X W \beta \]

- Also, if $\exists U \rightarrow \gamma U$ and $\exists U \rightarrow \alpha U X \beta$ then there would be a problem in the parsing table

  - $U \equiv X$ and $U \succneq X$

- Again, the problem can be resolved by introducing a new non-terminal $W$, and a new rule

  \[ W \rightarrow U, \text{ and change the initial rule to } U \rightarrow \alpha W X \beta \]
Example:

- E → E + T and F → ( E ); then
  - ( E and ( E
- W introduce a new non-terminal E₁, and a new rule
  E₁ → E, and change the initial rule to F → ( E₁ )
- Also, T → T * F and E → E + T; then
  - + T and + T
- Again, the problem can be resolved by introducing a new non-terminal W, and a new rule

  W → T, and change the initial rule to E → E + E₁
Example (Cont.)

• Now, there is a new problem! :-(

- Rules number 2 and 8 have the same rhs

- This problem is resolved by changing rule number 2 into $E \rightarrow T_1$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>$E \rightarrow E + T_1 \mid T$</td>
</tr>
<tr>
<td>3-4</td>
<td>$T \rightarrow T * F \mid F$</td>
</tr>
<tr>
<td>5-6</td>
<td>$F \rightarrow (E_1) \mid \text{id}$</td>
</tr>
<tr>
<td>7</td>
<td>$E_1 \rightarrow E$</td>
</tr>
<tr>
<td>8</td>
<td>$T_1 \rightarrow T$</td>
</tr>
</tbody>
</table>

---

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>$E \rightarrow E + T_1 \mid T_1$</td>
</tr>
<tr>
<td>3-4</td>
<td>$T \rightarrow T * F \mid F$</td>
</tr>
<tr>
<td>5-6</td>
<td>$F \rightarrow (E_1) \mid \text{id}$</td>
</tr>
<tr>
<td>7</td>
<td>$E_1 \rightarrow E$</td>
</tr>
<tr>
<td>8</td>
<td>$T_1 \rightarrow T$</td>
</tr>
</tbody>
</table>