CS 536 Announcements for Thursday, March 24, 2022

Homework 3 has been released

Programming Assignment 3 – due Monday, March 28

Midterm 2 – Wednesday, March 30

Last Time
• review grammar transformations
• building a predictive parser
• FIRST and FOLLOW sets

Today
• FIRST and FOLLOW sets
• building a predictive parser
• predictive parsing and syntax-directed translation

Next Time
• review, static semantic analysis

Recap of where we are

Predictive parser builds the parse tree top-down
• 1 token lookahead
• parse(selector table
• stack tracking current parse tree's frontier

Building the parse table – given production lhs → rhs, determine what terminals would lead us to choose that production

\[ \text{FIRST}(\alpha) = \{ T \mid (T \in \Sigma \land \alpha \Rightarrow^* T\beta) \lor (T = \varepsilon \land \alpha \Rightarrow^* \varepsilon) \} \]

\[ \text{FOLLOW}(a) = \{ T \mid (T \in \Sigma \land s \Rightarrow^* aa\beta \lor T = \text{EOF} \land s \Rightarrow^* aa) \} \]
FIRST and FOLLOW sets

FIRST(α) for $α = y_1 \ y_2 \ldots \ y_k$
Add $\text{FIRST}(y_1) - \{ \varepsilon \}$
If $\varepsilon$ is in $\text{FIRST}(y_1 \ \text{to} \ i-1)$, add $\text{FIRST}(y_i) - \{ \varepsilon \}$
If $\varepsilon$ is in all RHS symbols, add $\varepsilon$

FOLLOW(a) for $x \rightarrow α \ a \ β$
If $a$ is the start, add EOF
Add $\text{FIRST}(β) - \{ \varepsilon \}$
Add $\text{FOLLOW}(x)$ if $\varepsilon$ is in $\text{FIRST}(β)$ or $β$ is empty

Note that

FIRST sets
- only contain alphabet terminals and $ε$
- defined for arbitrary RHS and nonterminals
- constructed by started at the beginning of a production

FOLLOW sets
- only contain alphabet terminals and EOF
- defined for nonterminals only
- constructed by jumping into production

Putting it all together
- Build FIRST sets for each nonterminal
- Build FIRST sets for each production's RHS
- Build FOLLOW sets for each nonterminal
- Use FIRST and FOLLOW sets to fill parse table for each production

Building the parse table

```latex
\begin{verbatim}
for each production $x \rightarrow α$ {
  for each terminal $T$ in $\text{FIRST}(α)$ {
    put $α$ in table[$x$][T]
  }
  if $\varepsilon$ is in $\text{FIRST}(α)$ {
    for each terminal $T$ in $\text{FOLLOW}(x)$ {
      put $α$ in table[$x$][T]
    }
  }
}
\end{verbatim}
```
**Example**

**Original CFG**

```
expr    →  expr + term  
   |  term  

term    →  term * factor  
   |  factor  
factor  →  exponent ^ factor  
   |  exponent  
exponent →  INTLIT  
   |  ( expr )  
```

**Transformed CFG**

```
expr    →  term expr'  
expr'    →  + term expr' | ε  
term    →  factor term'  
term'    →  * factor term' | ε  
factor  →  exponent factor'  
factor'  →  ^ factor | ε  
exponent →  INTLIT | ( expr )  
```

<table>
<thead>
<tr>
<th></th>
<th><strong>FIRST</strong></th>
<th><strong>FOLLOW</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td>INTLIT (</td>
<td></td>
</tr>
<tr>
<td>expr'</td>
<td>+ ε</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td>INTLIT (</td>
<td></td>
</tr>
<tr>
<td>term'</td>
<td>* ε</td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td>INTLIT (</td>
<td></td>
</tr>
<tr>
<td>factor'</td>
<td>^ ε</td>
<td></td>
</tr>
<tr>
<td>exponent</td>
<td>INTLIT (</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>FIRST</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>expr  →  term expr'</td>
<td>INTLIT (</td>
</tr>
<tr>
<td>expr'  →  + term expr'</td>
<td>+</td>
</tr>
<tr>
<td>expr'  →  ε</td>
<td>ε</td>
</tr>
<tr>
<td>term  →  factor term'</td>
<td>INTLIT (</td>
</tr>
<tr>
<td>term'  →  * factor term'</td>
<td>*</td>
</tr>
<tr>
<td>term'  →  ε</td>
<td>ε</td>
</tr>
<tr>
<td>factor  →  exponent factor'</td>
<td>INTLIT (</td>
</tr>
<tr>
<td>factor'  →  ^ factor</td>
<td>^</td>
</tr>
<tr>
<td>factor'  →  ε</td>
<td>ε</td>
</tr>
<tr>
<td>exponent  →  INTLIT</td>
<td>INTLIT</td>
</tr>
<tr>
<td>exponent →  ( expr )</td>
<td>(</td>
</tr>
</tbody>
</table>
Example

CFG

\[\begin{align*}
  s & \rightarrow a C \mid b a \\
  a & \rightarrow A B \mid C s \\
  b & \rightarrow D \mid \varepsilon
\end{align*}\]

FIRST and FOLLOW sets

<table>
<thead>
<tr>
<th></th>
<th>FIRST sets</th>
<th>FOLLOW sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s \rightarrow a C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s \rightarrow b a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a \rightarrow A B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a \rightarrow C s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b \rightarrow D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b \rightarrow \varepsilon</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parse table

for each production \(x \rightarrow \alpha\)
  for each terminal \(T\) in \(\text{FIRST}(\alpha)\)
    put \(\alpha\) in \(\text{table}[x][T]\)
  if \(\varepsilon\) is in \(\text{FIRST}(\alpha)\)
    for each terminal \(T\) in \(\text{FOLLOW}(x)\)
      put \(\alpha\) in \(\text{table}[x][T]\)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parsing and syntax-directed translation

Instead of building parse tree, give parser a second, semantic stack

SDT rules are converted to actions

Augment the parsing algorithm

- number the actions
- when RHS of production is pushed onto symbol stack, include the actions
- when action is the top of symbol stack, pop & perform the action

Placing the action numbers in the productions

- action numbers go
  - after their corresponding non-terminal
  - before their corresponding terminal

Note: translations are popped from the semantic stack right-to-left
Example

CFG:

expr → expr + term #1
  | term

term → term * factor #2
  | factor

factor → #3 INTLIT
  | ( expr )

Translation goal: evaluate the expression, e.g., $5 + 2*3$ produces 11

#1 : termTrans = pop()
  exprTrans = pop()
  push(exprTrans + termTrans)

#2 : factorTrans = pop()
  termTrans = pop()
  push(termTrans * factorTrans)

#3 : push(INTLIT.val)

Converting to LL(1)

CFG:

expr → expr + term #1
  | term

term → term * factor #2
  | factor

factor → #3 INTLIT
  | ( expr )
Example SDT on transformed grammar

CFG:

expr → term expr'
expr' → + term #1 expr' | ε
term → factor term'
term' → * factor #2 term' | ε
factor → #3 INTLIT | ( expr )

SDT actions:

#1 : termTrans = pop()
    exprTrans = pop()
    push(exprTrans + termTrans)

#2 : factorTrans = pop()
    termTrans = pop()
    push(termTrans * factorTrans)

#3 : push(INTLIT.val)

Parse table

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>INTLIT</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td></td>
<td>term expr'</td>
<td>term expr'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expr'</td>
<td>+ term #1 expr'</td>
<td></td>
<td>ε</td>
<td>ε</td>
<td></td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td>factor term'</td>
<td>factor term'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term'</td>
<td>ε</td>
<td>* factor #2 term'</td>
<td>ε</td>
<td>ε</td>
<td></td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td>( expr )</td>
<td>#3 INTLIT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What about ASTs?

Push and pop AST nodes on the semantic stack

Keep references to nodes that we pop

CFG:

\[
\begin{align*}
\text{expr} & \rightarrow \text{term} \text{expr}' \\
\text{expr}' & \rightarrow + \text{term} \ #1 \text{expr'} | \epsilon \\
\text{term} & \rightarrow \text{factor} \text{term}' \\
\text{term}' & \rightarrow * \text{factor} \ #2 \text{term'} | \epsilon \\
\text{factor} & \rightarrow \#3 \text{INTLIT} | ( \text{expr} )
\end{align*}
\]

SDT actions:

\[
\begin{align*}
#1 & : \text{termTrans} = \text{pop()} \\
& \quad \text{exprTrans} = \text{pop()} \\
& \quad \text{push(} \\
#2 & : \text{factorTrans} = \text{pop()} \\
& \quad \text{termTrans} = \text{pop()} \\
& \quad \text{push(} \\
#3 & : \text{push(}
\end{align*}
\]