CS 536 Announcements for Tuesday, March 29, 2022

Midterm 2
- Wednesday, March 30, 7:30 – 9 pm
- B102 Van Vleck
- bring your student ID

Last Time
- building a predictive parser
- predictive parsing and syntax-directed translation

Today
- static semantic analysis
- name analysis
- exam review

Next Time
- Thursday, March 31 – no lecture
- Tuesday, April 4 – continue name analysis, start type checking

Static Semantic Analysis

Two phases
- name analysis (aka name resolution)
  - for each scope
    - process declarations
  - process statements
- type checking
  - process statements
Why do we need this phase?

**Code generation**
- different operations use different instructions

**Optimization**
- symbol table entry serves to identify which variable is used

**Error checking**

**Semantic error analysis**
For non-trivial programming languages, we run into fundamental undecidability problems:
- does the program halt?
- does the program crash?

Even with simplifying assumptions, sometimes infeasible in practice, as well
- combinations of thread interleavings
- inter-procedureal data analysis

**Goal of static semantic analysis:** catch some obvious errors
- undeclared identifiers
- multiply declared identifiers
- ill-typed terms
Name analysis

Associating IDs with their uses
Need to bind names before we can do type analysis
Questions to consider:
  - What definitions do we need about identifiers?
  - How do we bind definitions and uses together?

Symbol Table

= (structured) dictionary that binds a name to information we need

Each entry in the symbol table stores a set of attributes:
  - kind
  - type
  - nesting level
  - runtime location

Symbol table operations

  -

  -

  -

  -

  -

When do we do these operations?

Implementation considerations
Scoping

**scope** = block of code in which a name is visible/valid

**No scope (flat name scope)**

**Static/most-nested scope**

**Kinds of scoping**

*static*

*dynamiic*

**Scoping issues to consider**

- Can the same name be used in multiple scopes?
- Can the same name be used multiple times in a single scope (if the names are of different kinds)?
- Where does declaration have to occur relative to use?
- How do we match up uses to declarations?
- What are the boundaries of scopes, e.g., are method/function parameters and local variables in the same scope?
**CYK example**

**CFG**

\[ s \rightarrow a \ C \]

\[ a \rightarrow A \ B \]

\[ b \rightarrow D \]

Convert to CNF

Run the CYK algorithm to parse the input: D C C A B C
FIRST and FOLLOW sets

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

FOLLOW(\(a\)) for \(x \rightarrow a \alpha \beta\)
If \(a\) is the start, add EOF
Add FIRST(\(\beta\)) – \{ \epsilon \}
Add FOLLOW(\(x\)) if \(\epsilon\) is in FIRST(\(\beta\)) or \(\beta\) is empty

Building the parse table

for each production \(x \rightarrow \alpha\)
    for each terminal \(T\) in FIRST(\(\alpha\))
        put \(\alpha\) in table[x][T]
    if \(\epsilon\) is in FIRST(\(\alpha\))
        for each terminal \(T\) in FOLLOW(\(x\))
            put \(\alpha\) in table[x][T]
FIRST/FOLLOW Example

**Original CFG**

<table>
<thead>
<tr>
<th>expr</th>
<th>expr + term</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>term * factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>factor</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>factor</th>
<th>INTLIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( expr )</td>
<td></td>
</tr>
</tbody>
</table>

**Transformed CFG**

<table>
<thead>
<tr>
<th>expr</th>
<th>term expr'</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>expr'</th>
<th>+ term expr'</th>
<th>ε</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>factor term'</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>term'</th>
<th>* factor term'</th>
<th>ε</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>factor</th>
<th>INTLIT</th>
<th>( expr )</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>FIRST</th>
<th>FOLLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td></td>
</tr>
<tr>
<td>expr'</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
</tr>
<tr>
<td>term'</td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td></td>
</tr>
</tbody>
</table>

**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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expr'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>term'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example SDT on transformed grammar

CFG:

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

SDT actions:

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

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></td>
<td>term expr'</td>
<td>term expr'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expr'</td>
<td>+</td>
<td>term #1 expr'</td>
<td></td>
<td>\varepsilon</td>
<td>\varepsilon</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td></td>
<td>factor term'</td>
<td>factor term'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>term'</td>
<td>\varepsilon</td>
<td></td>
<td>* factor #2 term'</td>
<td>\varepsilon</td>
<td>\varepsilon</td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td>( expr )</td>
<td>#3 INTLIT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>