COMP 520 - Compilers

Lecture 12 (Tue Feb 23, 2016)

miniJava Contextual Analysis

- Please pick up from front of room
  - WA3 graded assignments
    - Total Points: 14, Average 11.2, Median 11, StdDev 2.2
Topics

• Review WA3

• miniJava Contextual Analysis
  – PA3 walkthrough
  – miniJava Identification and Type checking strategies

• Quick look at Triangle contextual analysis
Identification

- **IdTables**
  - enter(String s, Decl d)
    - associate s with Decl d
  - Decl retrieve(String s)
    - yields decl or null

- **Specific id tables**
  - is s a class name?
  - is s a member of class X?

- **Scoped id table**
  - what declaration is associated with s in the current scope?
  - is s already declared in the current scope?
  - is s already declared in a scope at level > k?
  - enter a new <name,Decl> at the current level
  - enter or exit a scope

<table>
<thead>
<tr>
<th>string</th>
<th>Decl</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>class names</td>
<td>ClassDecl</td>
<td>1</td>
</tr>
<tr>
<td>member names</td>
<td>MemberDecl</td>
<td>2</td>
</tr>
<tr>
<td>parameter names</td>
<td>ParameterDecl</td>
<td>3</td>
</tr>
<tr>
<td>local var names</td>
<td>LocalDecl</td>
<td>4+</td>
</tr>
</tbody>
</table>
Identification

- How to perform identification
  - Declarations need to be entered
    - ClassDecl, MemberDecl, LocalDecl
  - All Identifiers need to be linked to their controlling declaration
    - add a Declaration attribute to the Identifier class
    - work out a correct order to visit different parts of the AST to ensure all applicable declarations will have been seen before visiting an Identifier
    - link each identifier in the AST to its declaration using the appropriate idTable(s)
  - What constructs need identification?
    - Basically all
      - Statements, Expressions, References
      - but also Declarations
Identification

• Special challenges
  – Access and Visibility restrictions of MemberDecls
    • Non-static members are not always accessible
    • private members are not always accessible
    • need a “context” for a reference to make a judgment

  – References
    • example
      – x.y.z
    • what needs to be checked at each node of the Reference ast?
Identification

• Special challenges
  – Scope of variable in a VarDecl
    • e.g.
      
      ```java
      int x = 3 + x;
      ```
    • is erroneous!

  – predefined classes
    • System, etc.
Type Checking

- Relatively simple
  - Create a type attribute in every Expression node, or maybe in every node
  - The type rules for predefined functions are relatively simple
    - $\texttt{+}, \texttt{-}, \texttt{*} \; \texttt{etc} : \text{Int} \times \text{Int} \rightarrow \text{Int}$
    - $\texttt{==} : \alpha \times \alpha \rightarrow \text{Boolean}$
    - $\texttt{index} : \text{Array}(\alpha) \times \text{Int} \rightarrow \alpha$
    - $\texttt{assign} : \alpha \times \alpha \rightarrow \text{Stmt}$
  - A single upwards pass suffices for miniJava type checking

- Study the type related classes in the AST
  - Type, TypeKind, ArrayType, Classtype
  - create an equality function between arbitrary instances of Type
  - provided identification has completed successfully!
    - e.g.

$$A \ x = \text{new} \ A();$$
Type Checking

• Additional types
  – Error type
    • Error type is \textit{equal} to any type
    • limits propagation of errors
    • gives most useful continuation of type checking after an error
  – Unsupported type
    • Unsupported type is \textit{not equal} to any type
    • therefore a value of type unsupported is not type correct in any operation
    • predefined name String can have unsupported type