Compiler Practical 2013
Context Analysis

Berthold Hoffmann (B. Gersdorf, T. Röfer)
hof@informatik.uni-bremen.de
Cartesium 2.48
1. Context Analysis
2. Administration of Declarations
3. Extensions of the Syntax Tree
4. Task: Allowing Several Classes in a Program
Context Analysis (1)

CLASS Main
METHODS
METHOD main
VARIABLES
c : Integer
BEGIN
READ c
WHILE NEQ c MINUS 1 : Integer
DO
WRITE UNBOX DEREF c : REF Integer
MINUS 1 : Integer
READ c : REF Integer
END

Line 4, Column 13: Integer is predefined
Line 6, Column 14: c ist defined in Line 4, Column 9
Line 7, Column 15: c ist defined in Line 4, Column 9
Line 8, Column 19: c ist defined in Line 4, Column 9
Line 9, Column 18: c ist defined in Line 4, Column 9
Context Analysis (2)

• Identification
  – Where has the identifier been declared?
  – Which type does it have?
  – Visibility (simultaneous, local, global, hiding)

• Type analysis
  – Which type do the operands in expressions have?
  – Are expressions type-correct?
• Extensing the syntax tree
  – (Un)boxing and de-referencing variables
  – *Insert SELF*
  – Simplifies code generation (later)

• Determining (relative) addresses
  – Local variables relative to stack frame
  – Attributes relative to start address of the object
Declarations: \textit{ClassDeclaration}

- \textit{ClassDeclaration} contains \textit{MethodDeclarations} and \textit{VarDeclarations}
- Stores the \textit{declarations valid in a class}
- Knows the size of the objects of a class (\textit{objectSize})
- Contains methods for type comparison
- Contains every predefined type as a class attribute
Method-, VarDeclaration

• **MethodDeclaration**
  – Contains *VarDeclarations* and *Statements*
  – Declares *SELF*

• **VarDeclaration**
  – Can distinguish attributes and local variables (*isAttribute*)
  – Contains the type of the attribute and local variable, resp.
  – Contains the relative address of the attribute and local variable, resp.
  – Does not generate code
Name Spaces (LOOP)

• LOOP has a common name space for classes, variables, attributes, and methods

• Visibility levels in LOOP
  – Classes
  – Methods and attributes of the actual class
    • This level can also be entered with the dot operator
  – Local variables of the actual method
Example for Declaration Levels

CLASS Main IS
  METHOD main IS
    c : Integer;
    BEGIN
    READ c;
    WHILE c # -1 DO
      WRITE c;
      READ c;
    END WHILE
  END METHOD
END CLASS

"Integer" → ClassDeclaration
"Main" → ClassDeclaration
"main" → MethodDeclaration
"_self" → VarDeclaration
"c" → VarDeclaration
• Layers of Hashtables, which associate identifiers with declarations

• \textit{enter}() generates a new level, \textit{leave}() removes it

• \textit{add}(...) inserts a declaration on top level
  – If the identifier has already been declared on top level, an error is signalled

• \textit{resolve}(...) delivers the declaration of an identifier
  – \textit{resolveType}(...) and \textit{resolveVarOrMethod}(...) check the kind of the declaration

• Contains a reference to the actual class as well
Extending the Syntax Tree: Boxing

• a:=1;

```
Assignment

BoxExpression
  type: intClass
  lValue: false

VarOrCall
  type: intClass
  lValue: true

LiteralExpression
  type: intType
  lValue: false

LiteralExpression
  type: intType
  lValue: false

Assignment

VarOrCall
  type: intClass
  lValue: true
```
Dereferencing

- `b: Integer;`

... 

`a:=b;`
Unboxing and Dereferencing

- $-a$

```
UnaryExpression
  type: intType
  lValue: false

UnBoxExpression
  type: intType
  lValue: false

DeRefExpression
  type: intClass
  lValue: false

VarOrCall
  type: intClass
  lValue: true
```

```
UnaryExpression
  type: null
  lValue: false

VarOrCall
  type: intClass
  lValue: true
```

```
VarOrCall
  type: intClass
  lValue: true
```
**Insert SELF**

- METHOD a IS
  BEGIN
    IF ...
    ELSE a;

```plaintext
VarOrCall
  type: null
  lValue: true

AccessExpression
  type: voidType
  lValue: false

DeRefExpression
  type: (Main)
  lValue: false

VarOrCall
  type: (Main.a)
  lValue: false
  identifier: “a”

VarOrCall
  type: (Ref Main)
  lValue: true
  identifier: “_self”
```
Inserting SELF

• `VarOrCall.contextAnalysis(...)` is not called for the right-hand side of an `AccessExpression`
  – Thus a `SELF`. Is inserted here, if the identifier is an attribute or method

• For the right-hand side of an `AccessExpression`,
  `VarOrCall.contextAnalysisForMember(...)` is called instead
  – Also for the left-hand side (in `VarOrCall.contextAnalysis(...)`)
• `Program.classes` replaces `Program.theClass`

• Simultaneous visibility `program ::= { classdecl }`
  – Methods may access attributes and methods of other classes
  – These classes have to be known before method bodies are checked

• Enter `Integer` und `Boolean` in `Program.classes`
  – Then inheritance can be easier introduced, later
  – Add an attribute `_value : _Integer` oder `_value : _Boolean` instead of setting `ClassDeclaration.(int|bool)Class.objectSize` manually