Compiler Practical 2013

Context Analysis

Berthold Hoffmann (B. Gersdorf, T. Röfer)
hof@informatik.uni-bremen.de
Cartesium 2.48

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Universität Bremen

Structure

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

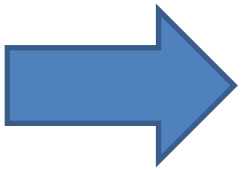
1. Context Analysis
2. Administration of Declarations
3. Extensions of the Syntax Tree
4. Task: Allowing Several Classes in a Program

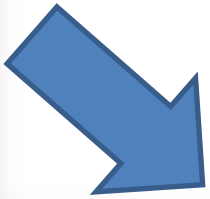
Context Analysis (1)

```
CLASS Main
  METHODS
    METHOD main
      VARIABLES
        c : Integer
      BEGIN
        READ
          c
        WHILE
          NEQ
            c
            MINUS
              1 : _Integer
          DO
            WRITE
              c
            READ
              c
```

LOOPc -s

LOOPc -c





LOOPc -i

```
CLASS Main
  METHODS
    METHOD main
      VARIABLES
        c (1) : Integer
      BEGIN
        READ
          c : REF Integer
        WHILE
          NEQ : _Boolean
            UNBOX : _Integer
              DEREF : Integer
                c : REF Integer
            MINUS : _Integer
              1 : _Integer
          DO
            WRITE
              UNBOX : _Integer
                DEREF : Integer
                  c : REF Integer
            READ
              c : REF Integer
```

```
Line4, Column 13: Integer is predefined
Line6, Column 14: c ist defined in Line 4, Column 9
Line7, Column 15: c ist defined in Line 4, Column 9
Line8, Column 19: c ist defined in Line 4, Column 9
Line9, Column 18: c ist defined in Line 4, Column 9
```

Context Analysis (2)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- 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?

Context Analysis (3)

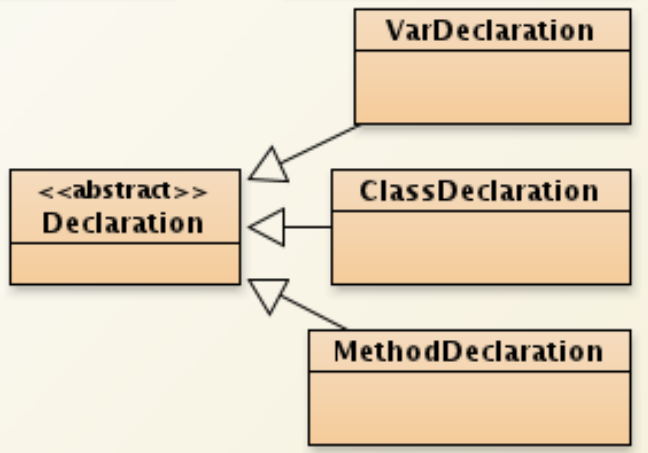
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- 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: *ClassDeclaration*

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

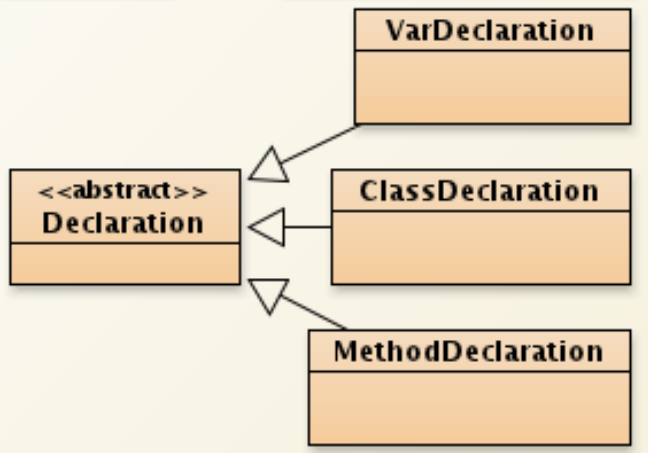
- *ClassDeclaration* contains *MethodDeclarations* and *VarDeclarations*
- Stores the *declarations valid in a class*



- Knows the size of the objects of a class (*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)

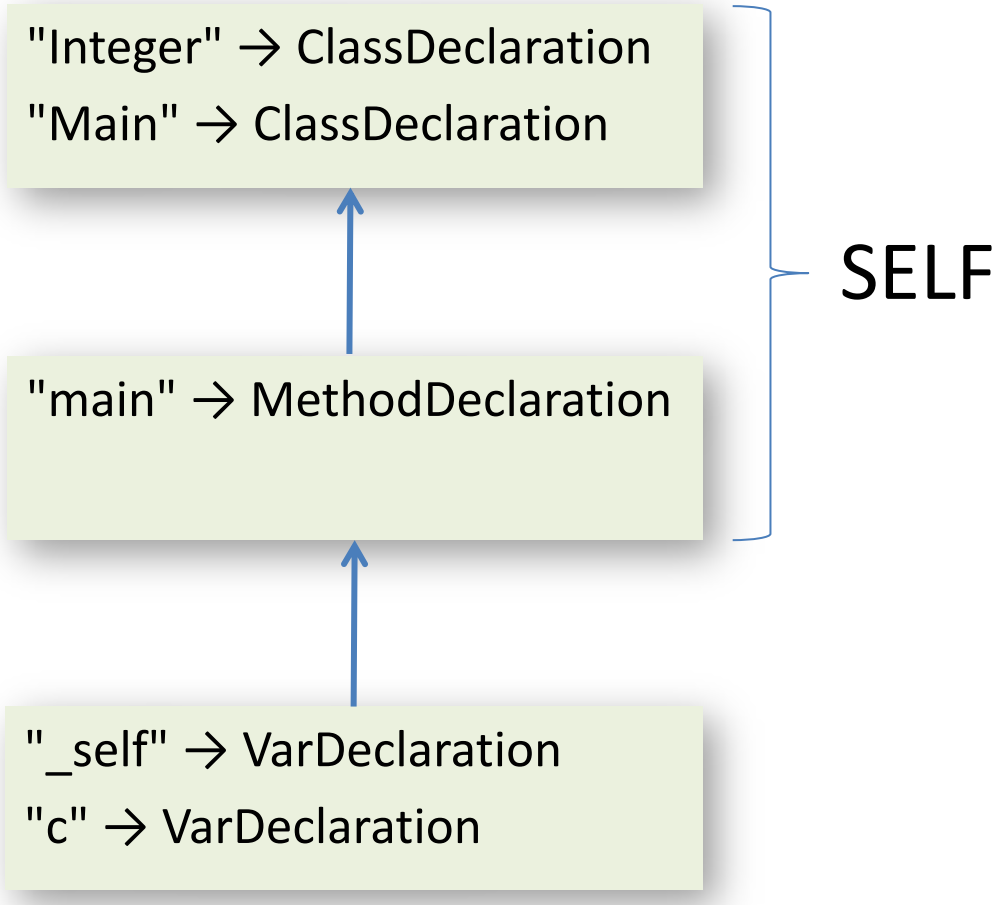
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- 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

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

```
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
```



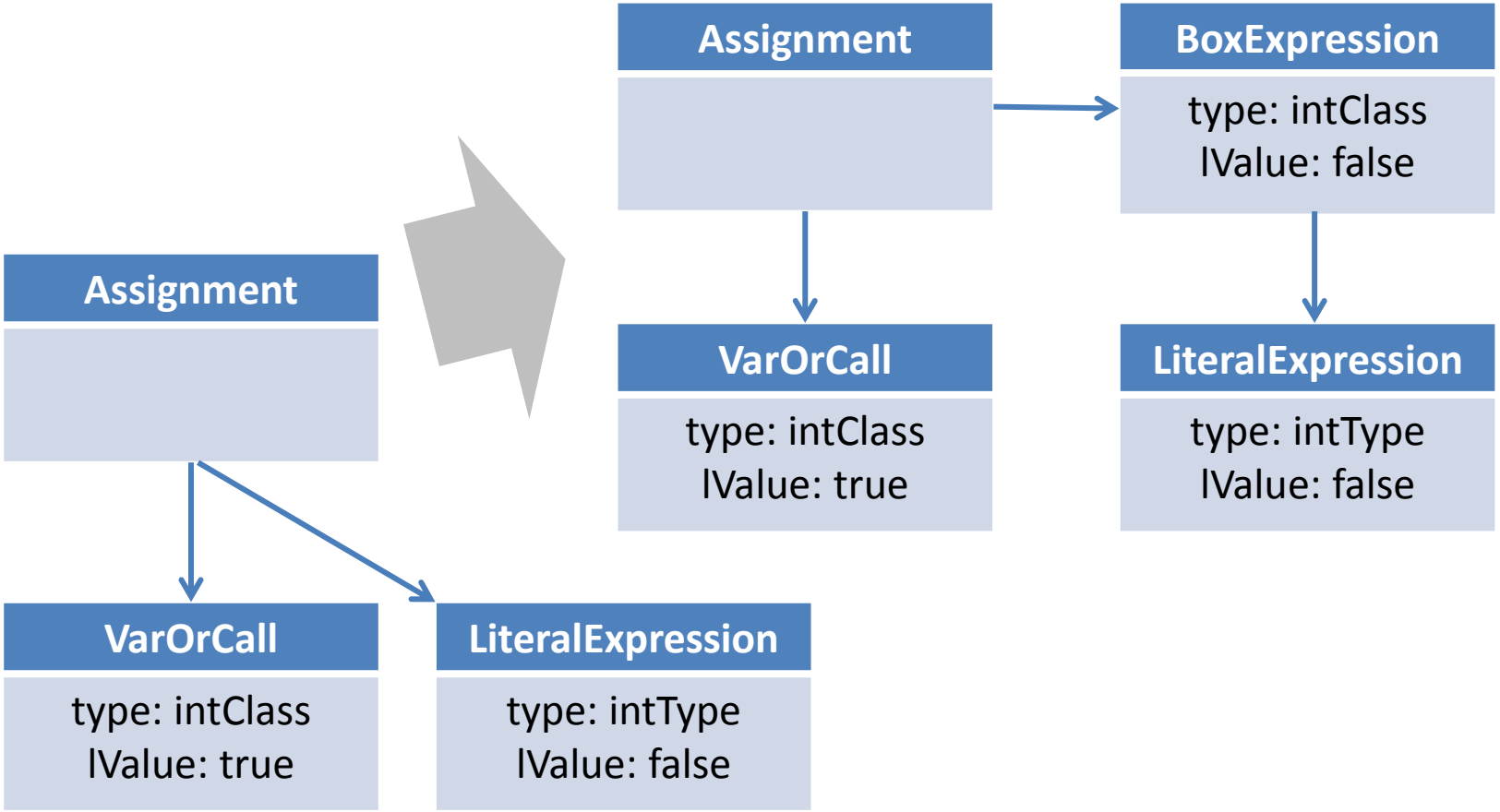
Declaration Sequences: *Declarations*

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Layers of Hashtables, which associate identifiers with declarations
- *enter*() generates a new level, *leave*() removes it
- *add*(...) inserts a declaration on top level
  - If the identifier has already been declared on top level, an error is signalled
- *resolve*(...) delivers the declaration of an identifier
  - *resolveType*(...) and *resolveVarOrMethod*(...) check the kind of the declaration
- Contains a reference to the actual class as well

Extending the Syntax Tree: Boxing

- a:=1;

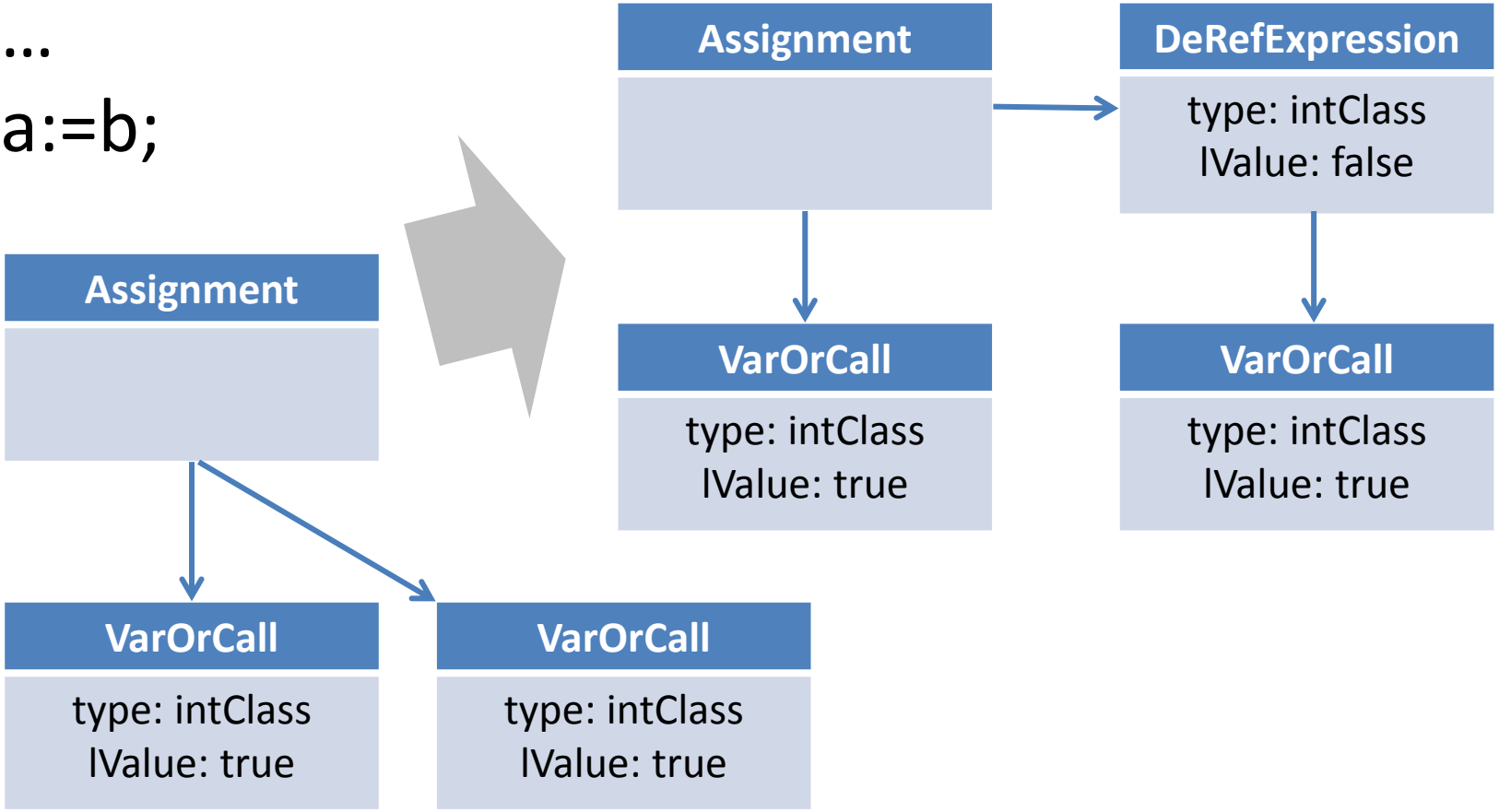


Dereferencing

- b: Integer;

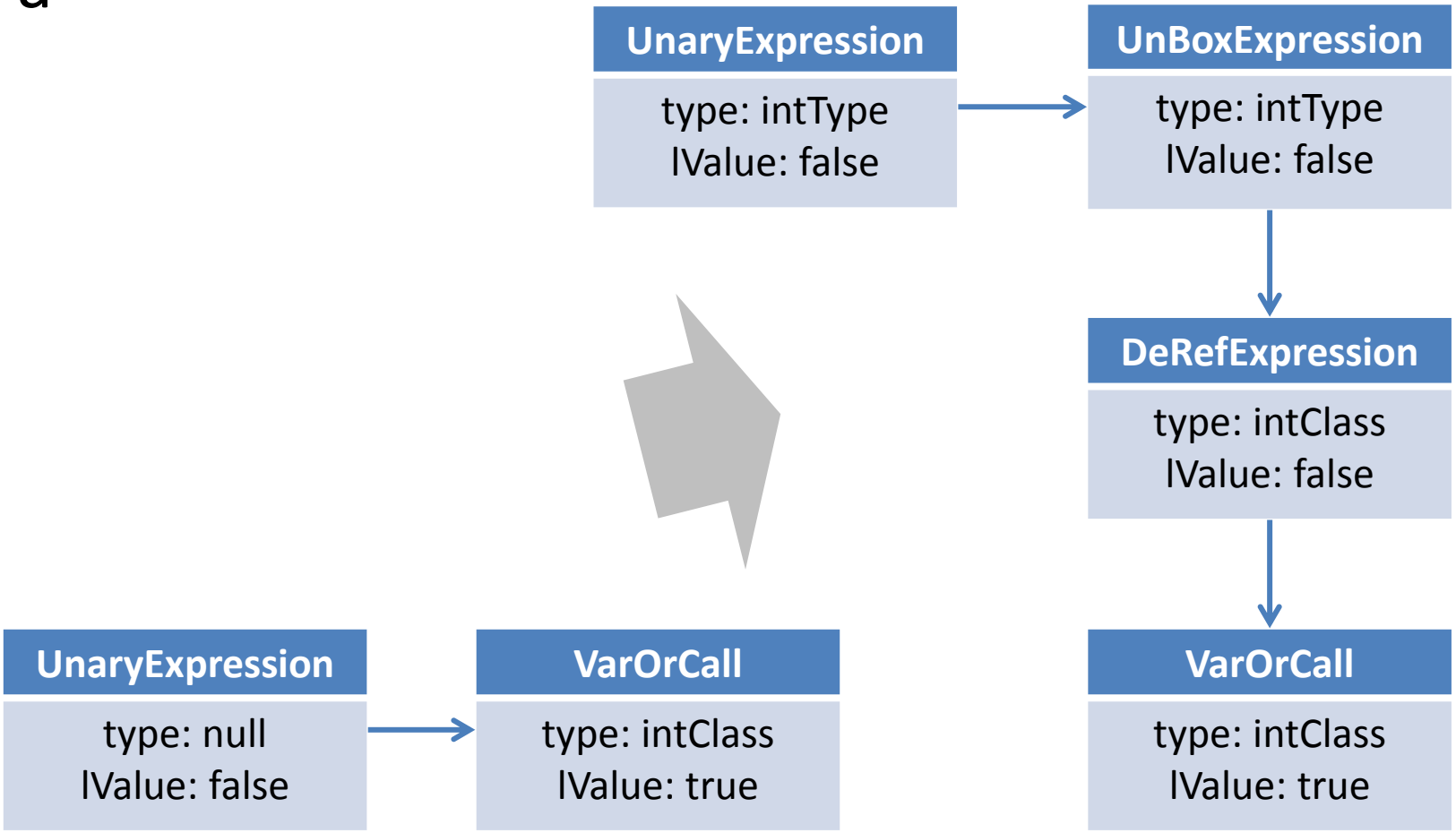
  …

  a:=b;



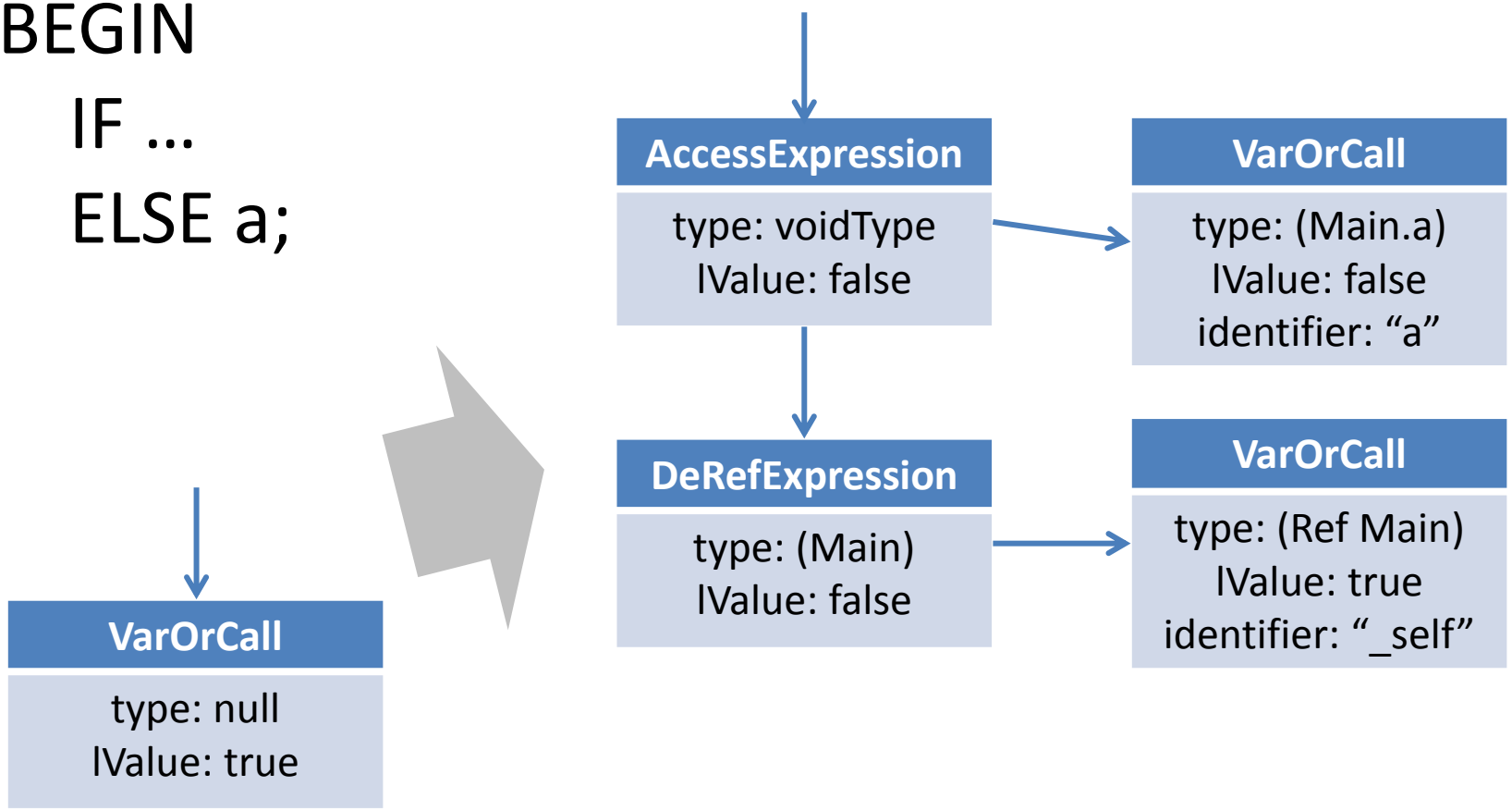
Unboxing and Dereferencing

- -a



Insert SELF

- METHOD a IS
  BEGIN
    IF …
    ELSE a;



Inserting SELF

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- *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(...)*)

Task: Programs with Several Classes

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- *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

program ::= { *classdecl* }

- 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

10 %