

Declaration Processing Fundamentals

Attribute Descriptor Objects

VariableAttributes

Variabletype => A Type Reference

Variableaddr => An Address

TypeAttributes

Thisstype => A Type Reference

Type Descriptor Objects

IntegerTypeDescriptor

Size => *a positive number*

ArrayTypeDescriptor

Size => *a positive number*

ElementType => *a type reference*

Bounds => *a range descriptor*

RecordTypeDescriptor

Size => *a positive number*

Fields => *a symbol table*

Type Checking (and Code Generation) using and Abstract Syntax Tree

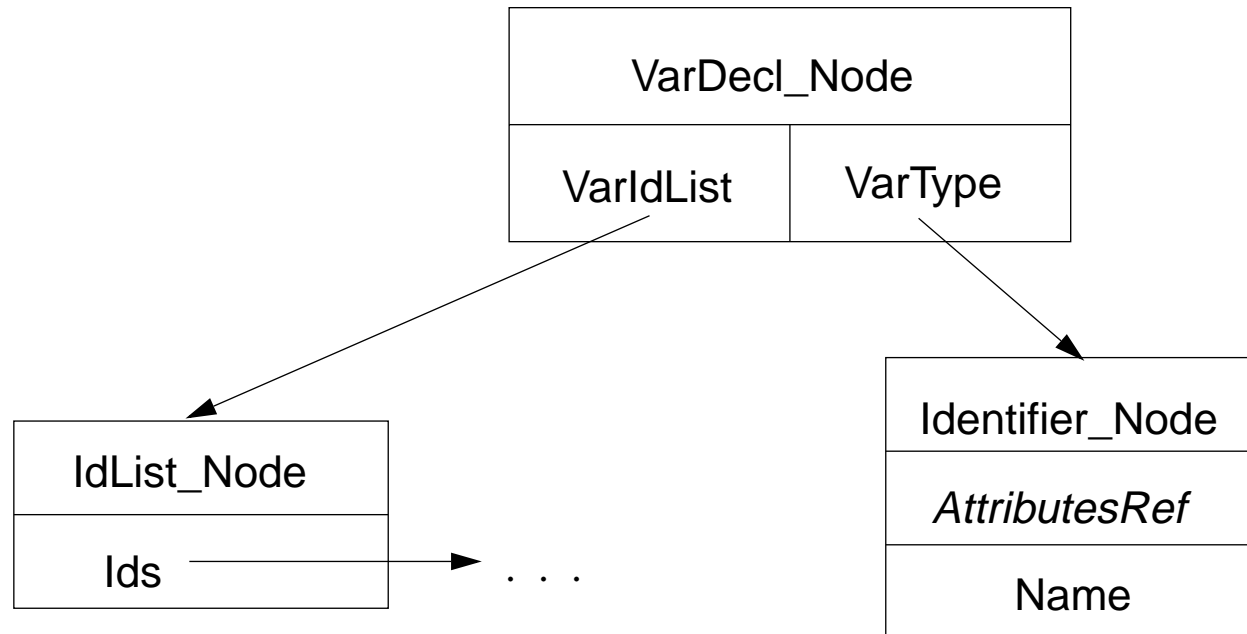
PROGRAM_NODE.SEMANTICS()

1. Declarations.Semantics()
2. Statements.Semantics()

PROGRAM_NODE.CODEGEN()

1. Declarations.CodeGen()
2. Statements.CodeGen()

Simple Variable Declarations



`VARDECL_NODE.SEMANTICS()`

1. `VarIdList.DeclSemantics (VarType.TypeSemantics())`

IDENTIFIER_NODE.TYPESEMANTICS()

1. this_node.Semantics ()
2. if AttributesRef indicates that Name is a type name
3. then return the value of AttributesRef.ThisType()
4. else Produce an error message indicating that Name cannot be interpreted as a type
5. return ErrorType

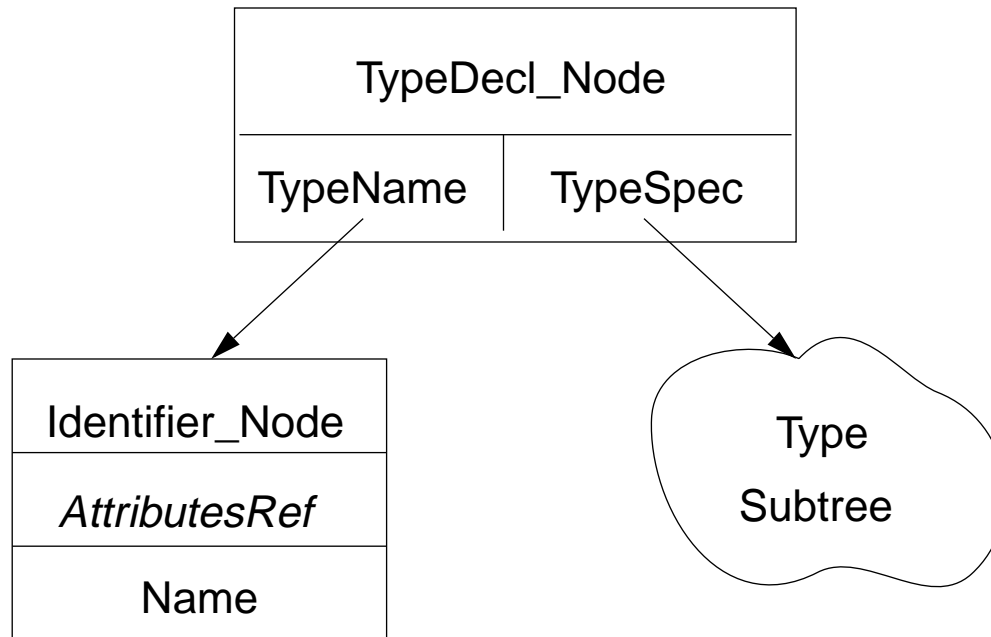
IDENTIFIER_NODE.SEMANTICS()

1. Look Name up in the symbol table
2. if it is found there
3. then Set AttributesRef to the Attributes descriptor associated with Name
4. else Produce an error message indicating that Name has not been declared
5. Set AttributesRef to null

IDLIST_NODE.DECLSEMANTICS(*DeclType*)

1. for each Id on the list referenced by Ids
2. do Enter the Id in the current symbol table
3. if it is already there
4. then Produce an error message indicating a duplicate declaration
5. else Associate a VariableAttributes descriptor with the Id indicating
 - its type is DeclType

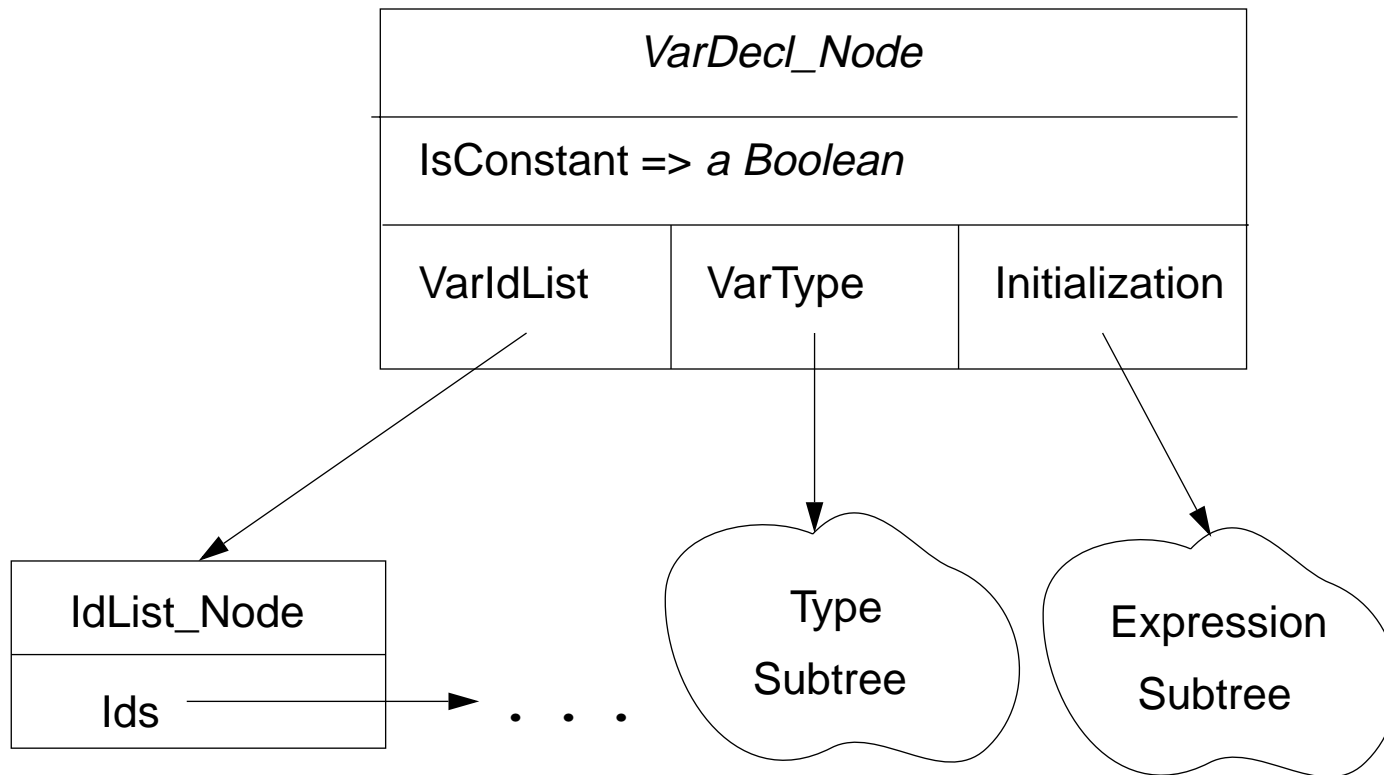
Type Definitions and Declarations



`TYPEDECL_NODE.SEMANTICS()`

1. Enter `TypeName.Name()` into the symbol table
2. Create a `TypeAttributes` descriptor to associate with this name indicating:
 - the type it references `TypeSpec.TypeSemantics()`

Variable Declarations Revisited



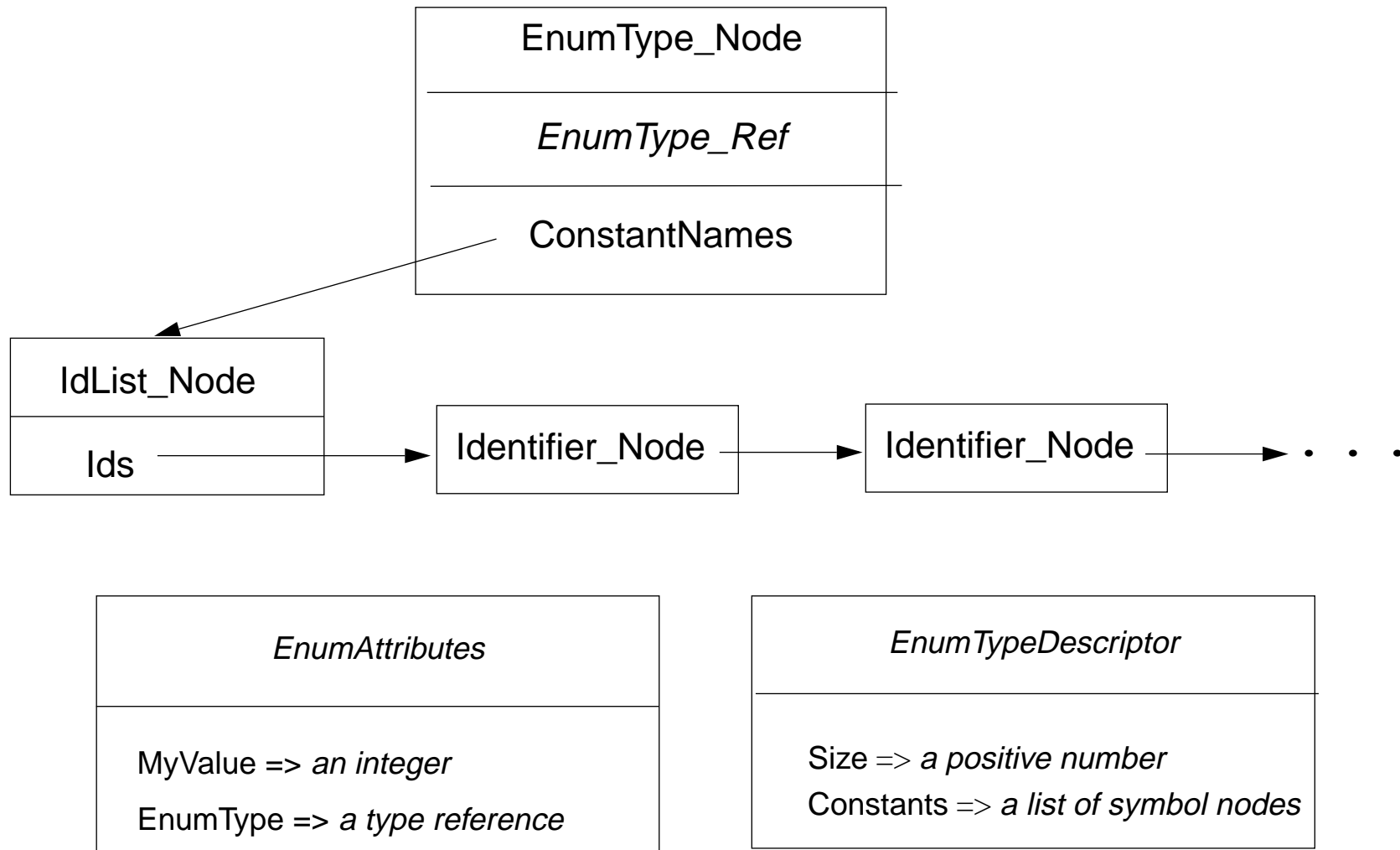
VARDECL_NODE.SEMANTICS()

1. Local Variables: Decltype, Inittype
2. Set Decltype To Vartype.TypeSemantics()
3. if Initialization Is Not A Null Pointer
4. then Set Inittype To Initialization.type semantics ()
5. Check That Assignable (Inittype, Decltype) Is True
6. else • Initialization is null
7. Check That Isconstant Is False
8. • Isconstant was set when the AST was built
9. Idlist.declsemantics (Decltype, Isconstant)

IDLIST_NODE.DECLSEMANTICS(*Decltype, Isconstant*)

1. for Each Id On The List Referenced By Ids
2. do Enter The Id In The Current Symbol Table
3. if Isconstant
4. then Associate A Constattributes Descriptor With It Indicating:
5. • Its Type Is Decltype
6. • Its Value Is Defined By Initialization (A Pointer To An Expression Tree)
7. else • It Is A Variable
8. Associate A Variableattributes Descriptor With It Indicating:
9. • Its Type Is Decltype
- Any initialization will be handled by the code generation pass

Enumeration Types

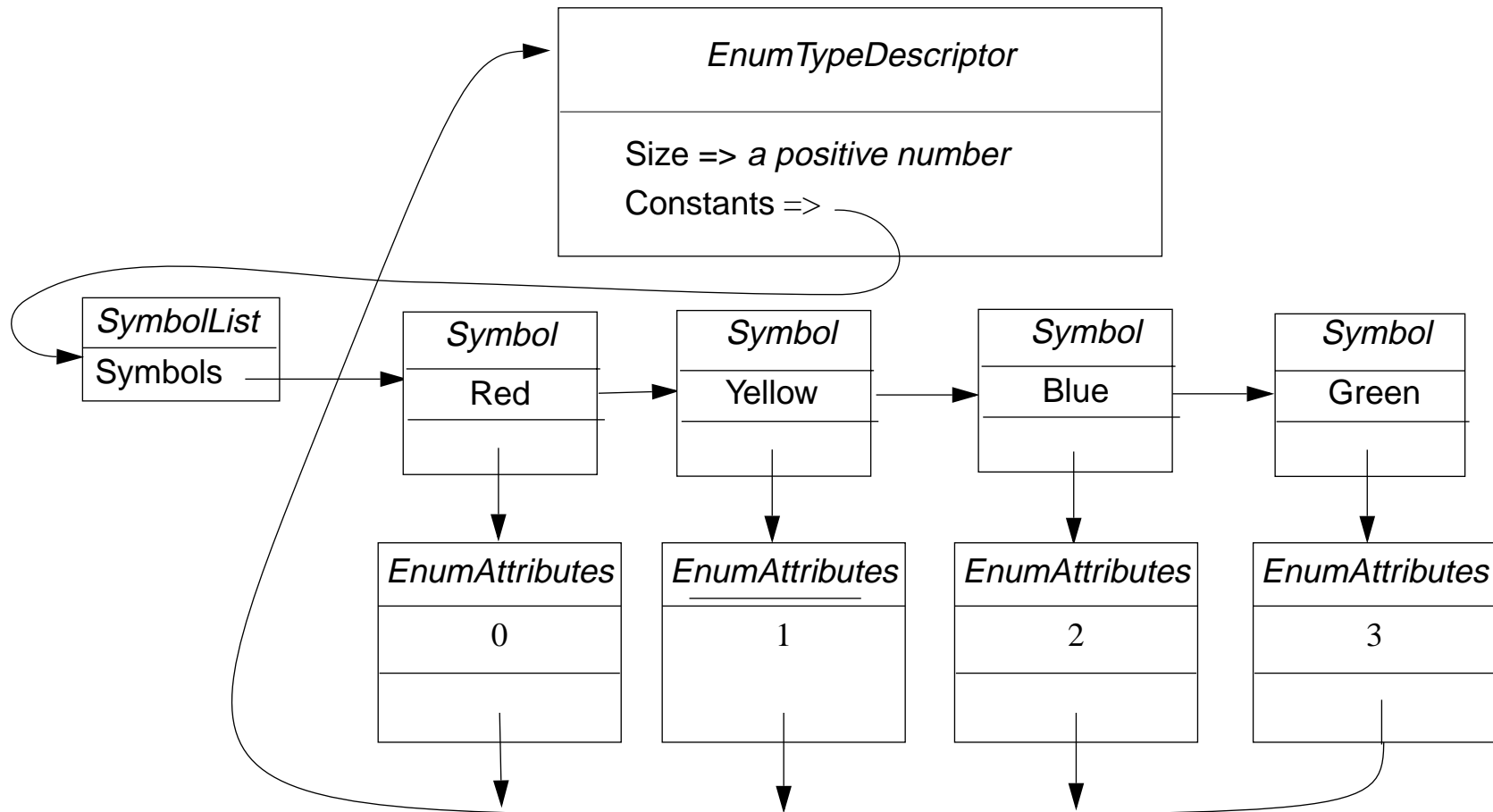


Defining an Enumeration Type

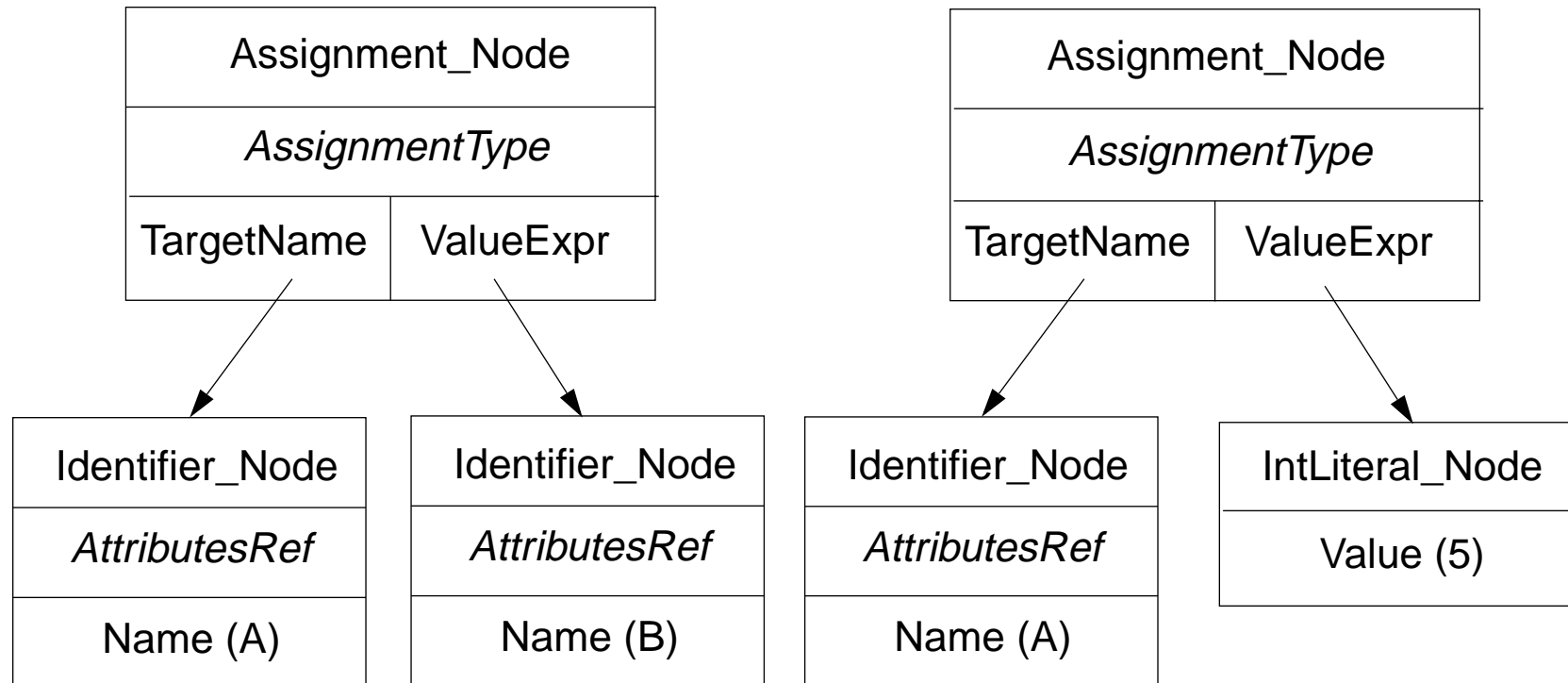
ENUMTYPE_NODE.TYPESEMANTICS())

1. [Local Variable: Nextvalue, Enumtype_ref]
2. Create An Enumtypedescriptor For The New Enumeration Type With:
3. Constants Pointing To An Empty List Of Symbols
4. Set Enumtype_ref To Point To This Enumtypedescriptor
5. Set Nextvalue To 0
6. for Each Of The Identifiers In The Constantnames List
7. do Enter The Identifier In The Current Symbol Table
8. Associate An Enumattributes Descriptor With It Indicating:
9. Its Type Is The Typedescriptor Pointed To By Enumtype_ref
10. Its Value Is Nextvalue
11. Increment Nextvalue
12. return Enumtype_ref

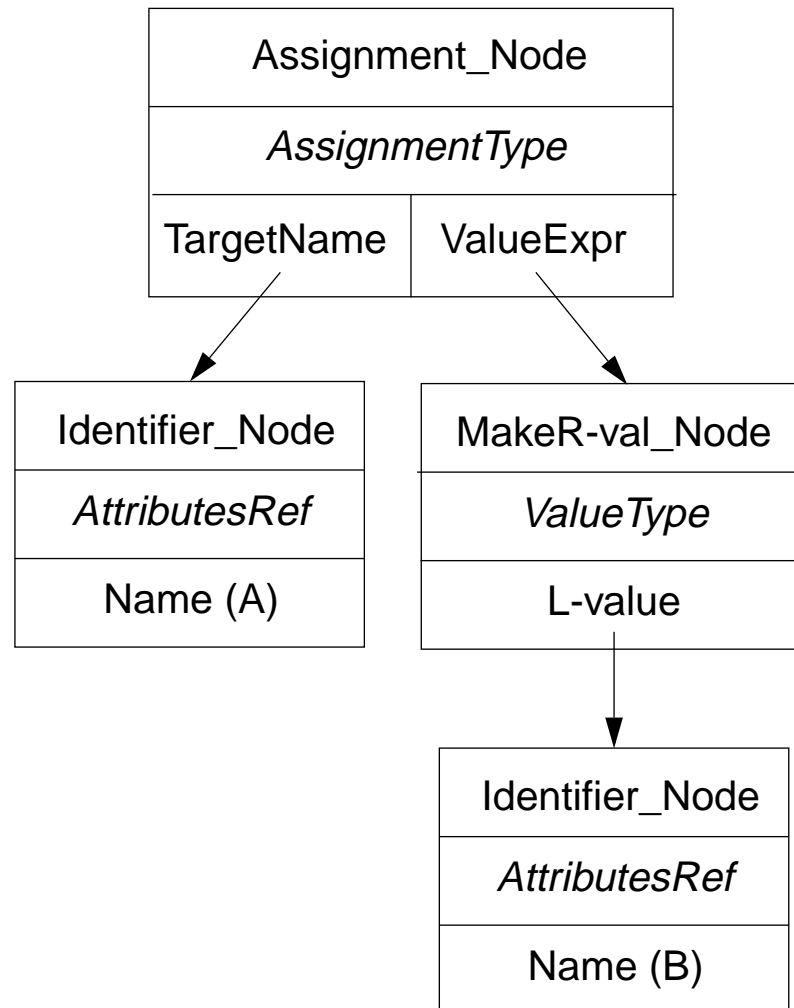
Representation of an Enumeration Type



Processing Assignment Statements



L-values and R-values



ASSIGNMENT_NODE.SEMANTICS()

1. [Local variables: LeftType, RightType]
2. Set LeftType to TargetName.ExprSemantics ()
3. Set RightType to ValueExpr.ExprSemantics ()
4. if RightType is assignable to LeftType
5. then Set AssignmentType to LeftType
6. else Set AssignmentType to ErrorType

IDENTIFIER_NODE.EXPRSEMANTICS ()

1. Call this_node.Semantics ()
2. if AttributesRef indicates that Name is a variable
3. then returnAttributesRef.VariableType
4. else Produce an error message indicating that Name cannot be interpreted as a variable
5. returnErrorType

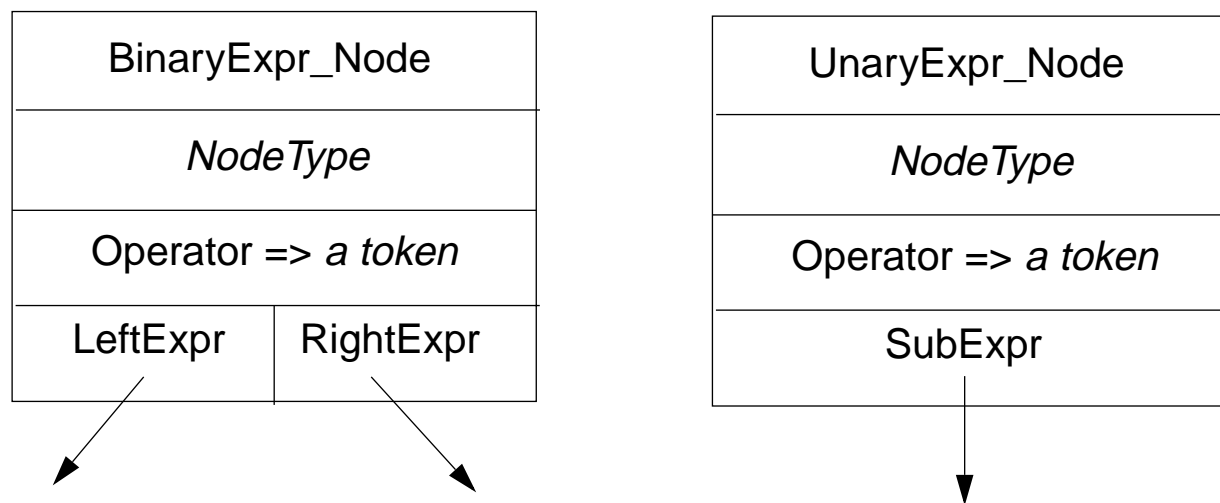
MAKER-VAL_NODE.EXPRSEMANTICS ()

1. Set ValueType to L-value.ExprSemantics ()
2. returnValueType

INTLITERAL_NODE.EXPRSEMANTICS()

1. returnIntegerType

Processing Expressions



BINARYEXPR_NODE.EXPRSEMANTICS ()

1. [Local variables: LeftType, RightType]
2. Set LeftType to LeftExpr.ExprSemantics ()
3. Set RightType to RightExpr.ExprSemantics ()
4. Set NodeType to BinaryResultType (Operator, LeftType, RightType)
5. returnNodeType

UNARYEXPR_NODE.EXPRSEMANTICS ()

1. [Local variable: SubType]
2. Set SubType to SubExpr.ExprSemantics ()
3. Set NodeType to UnaryResultType (Operator, SubType)
4. returnNodeType