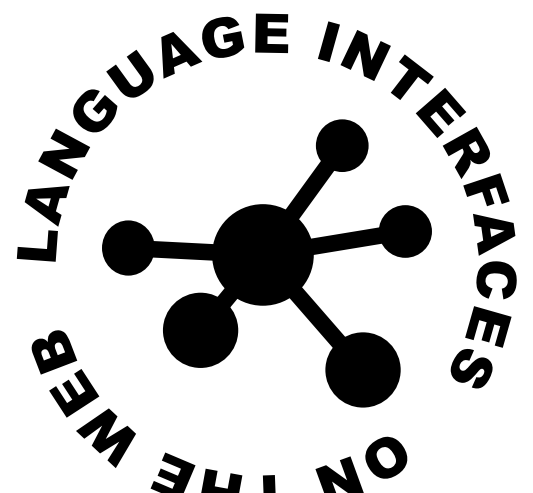


Implementing LionWeb in Rascal

Ulyana Tikhonova

LionWeb: Language { Interfaces, Interoperability, Integration } on the Web

CWI

Centrum Wiskunde & Informatica

Rascal

The one-stop shop for metaprogramming

Design and Implementation of Domain Specific Languages

Context Free Grammars
Concrete Syntax Fragments
Static Analysis
Eclipse Plugins
VScode extensions
Language Server Protocol

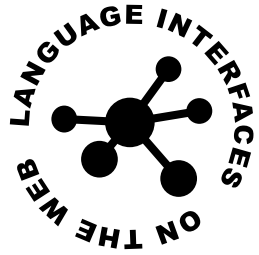
Source Code = Data

Java, C++, Python, Ada, ...
Metrics
Static Analysis
Debugging
Reverse Engineering

Source-to-Source

Pattern Matching
Concrete Syntax Fragments
Generic Type-safe Traversal
Transpilers
Compilers

DSL in LionWeb vs. Rascal

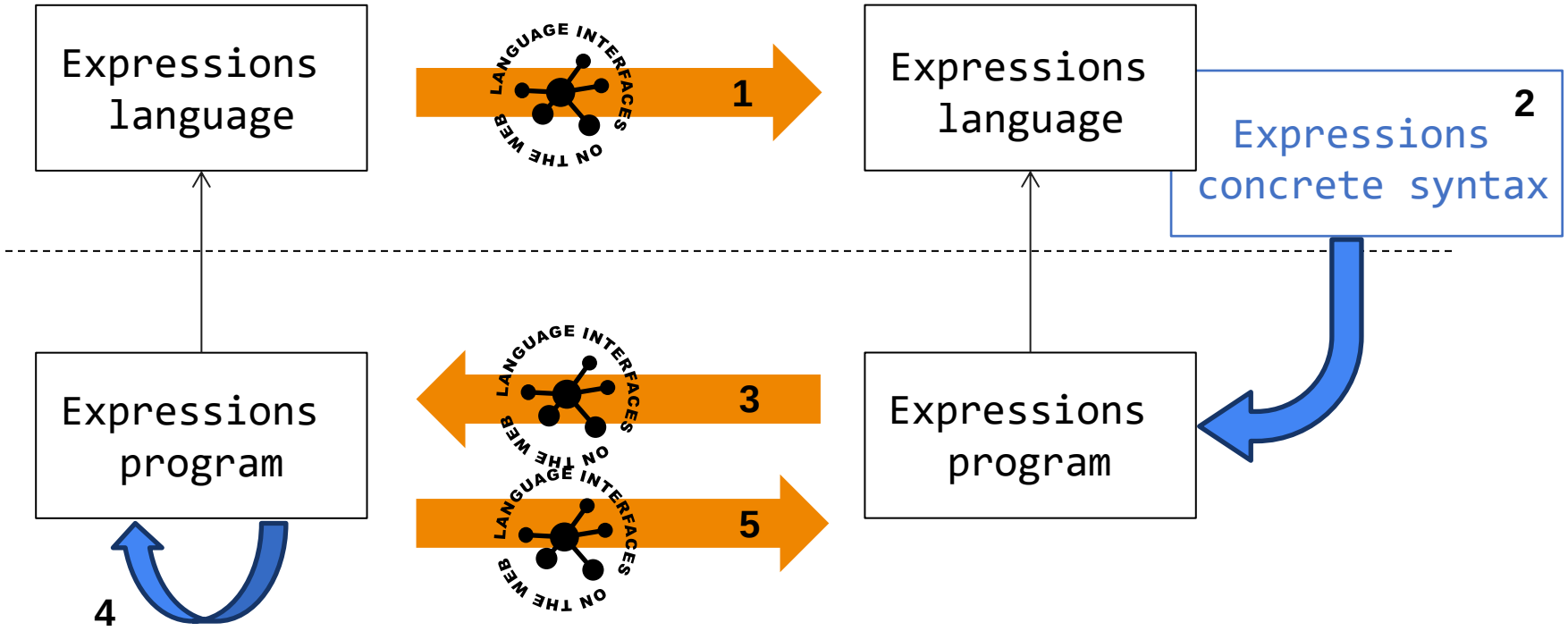


- Everything is a model
- Meta-model
- Objects/nodes
- Cross-referencing
- Inheritance
- Containment
- Interfaces
- Annotations
- Enumerations

- Concrete syntax grammar
- Algebraic data types
- Immutable values
- Model is a pure (parse) tree
- Alternative constructors for ADT
- Constructors have parameters

https://github.com/cwi-swat/rascal-ecore

Demo

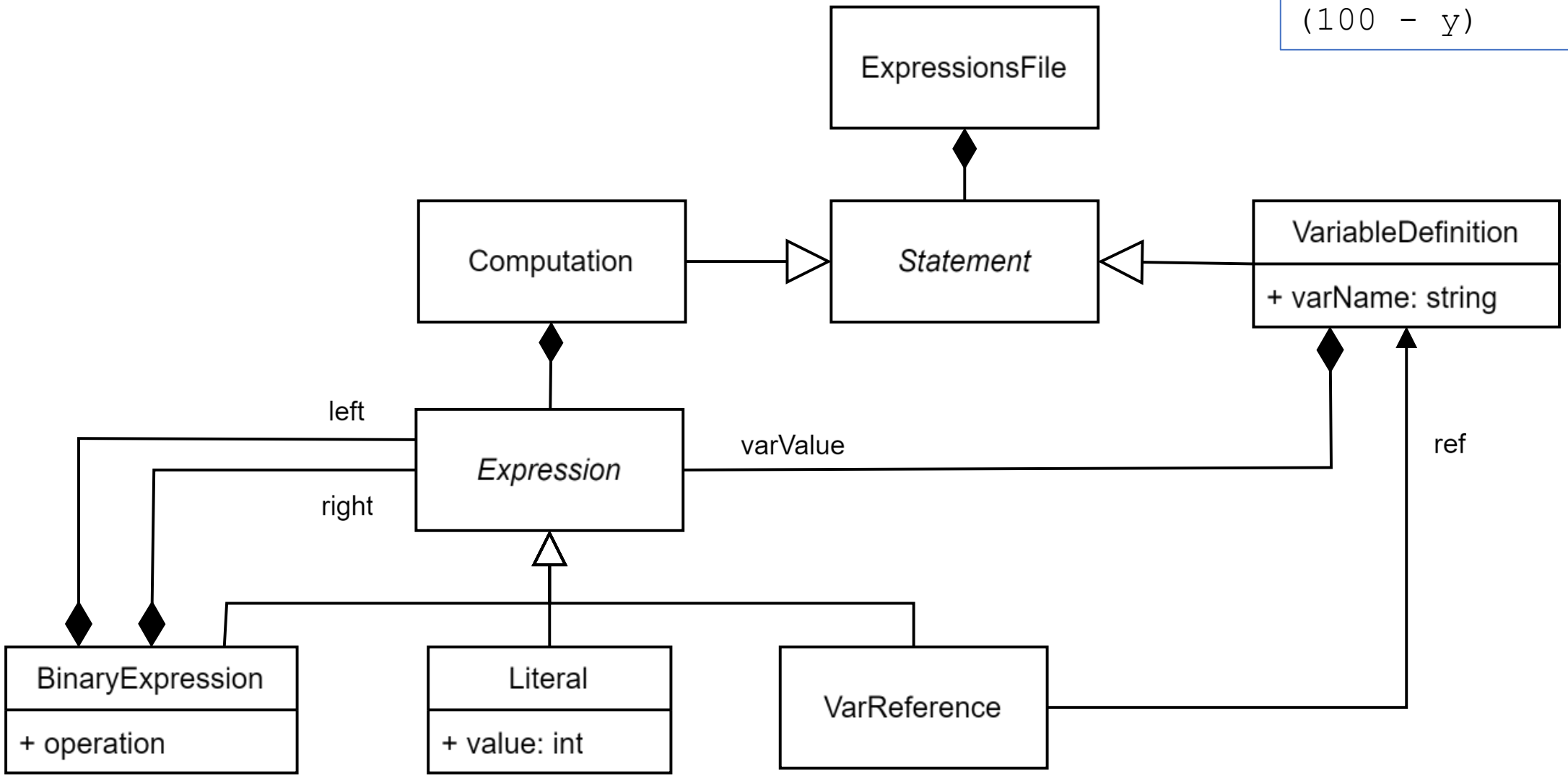


Lionweb translation, DSL agnostic

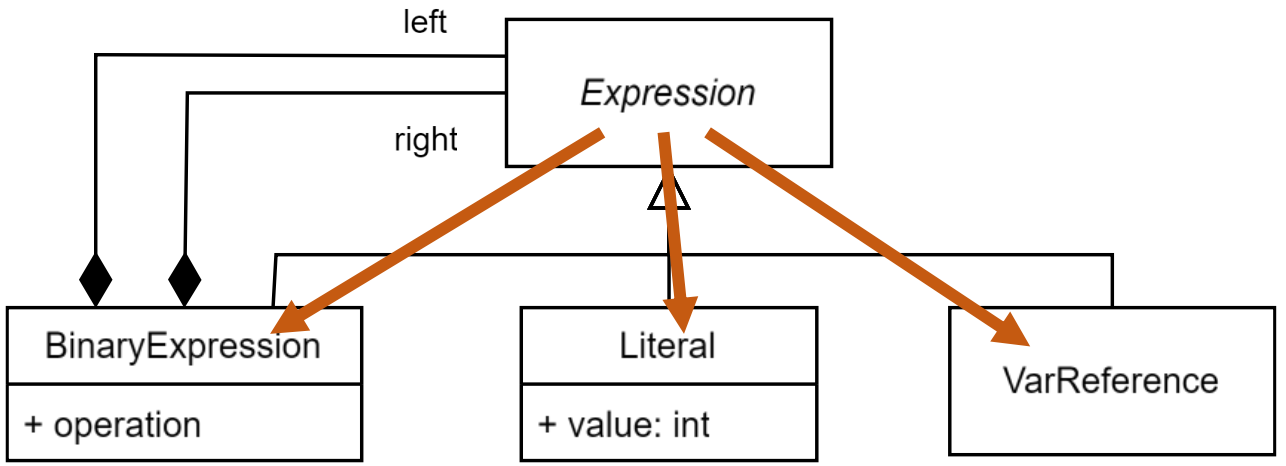
manual change, DSL specific

Expressions language

```
x = 5;
y = 4;

z =(10 + (x - y));
(100 - y)
```

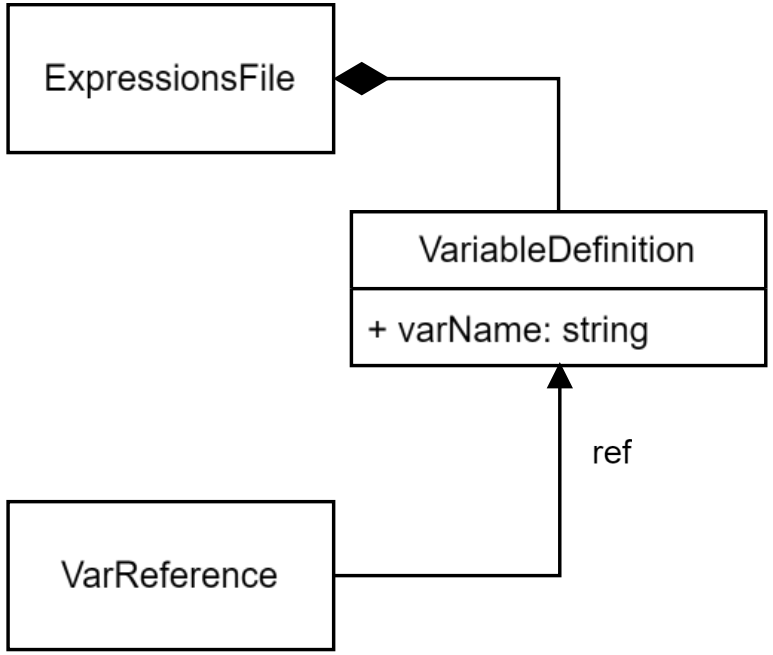


How it works




```
data Expression
  = Expression(Literal \literal
        , int \value = literal.\value)
  | Expression(VarReference \varReference
        , lionweb::pointer::Pointer[VariableDefinition] \ref = varReference.\ref)
  | Expression(BinaryExpression \binaryExpression
        , BinaryOperation \operation = binaryExpression.\operation
        , Expression \leftOperand = binaryExpression.\leftOperand
        , Expression \rightOperand = binaryExpression.\rightOperand)
  ;
```

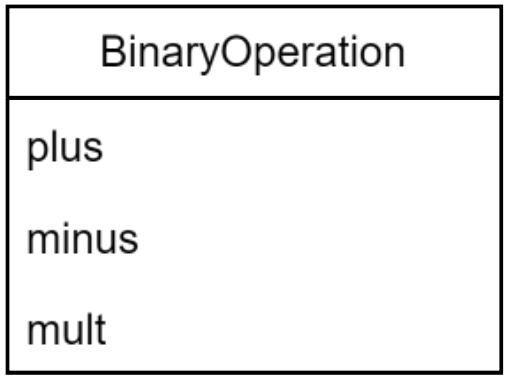
How it works




```
data VarReference
  = VarReference(lionweb::pointer::Pointer[VariableDefinition] \ref = null()
      , lionweb::pointer::Id \uid = "")
  ;

data VariableDefinition
  = VariableDefinition(str \varName = ""
      , list[Expression] \varValue = []
      , lionweb::pointer::Id \uid = "")
  ;
```

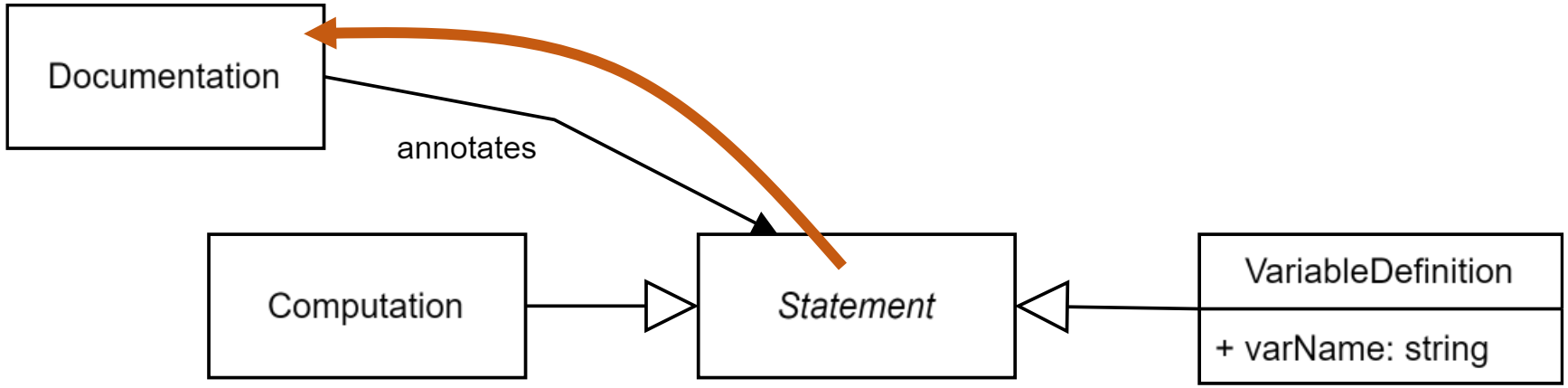
```
alias Id = str;

data Pointer[&T]
    = Pointer(Id uid, str info = "")
    | null();

node resolve(Pointer[&T <: node] pointer,
                    list[node] scope)
```

How it works



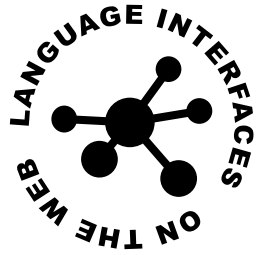

```
data BinaryOperation
  = plus()
  | mult()
  | minus()
  ;
```

How it works




```
data Statement
  = Statement(Computation \computation
      , Expression \expr = computation.\expr
      , list[Documentation] \annoDocumentation = computation.\annoDocumentation
      , lionweb::pointer::Id \uid = computation.\uid)
  | Statement(VariableDefinition \variableDefinition
      , str \varName = variableDefinition.\varName
      , list[Expression] \varValue = variableDefinition.\varValue
      , list[Documentation] \annoDocumentation = variableDefinition.\annoDocumentation
      , lionweb::pointer::Id \uid = variableDefinition.\uid)
  ;

data Documentation
  = ...
```

DSL in LionWeb vs. Rascal



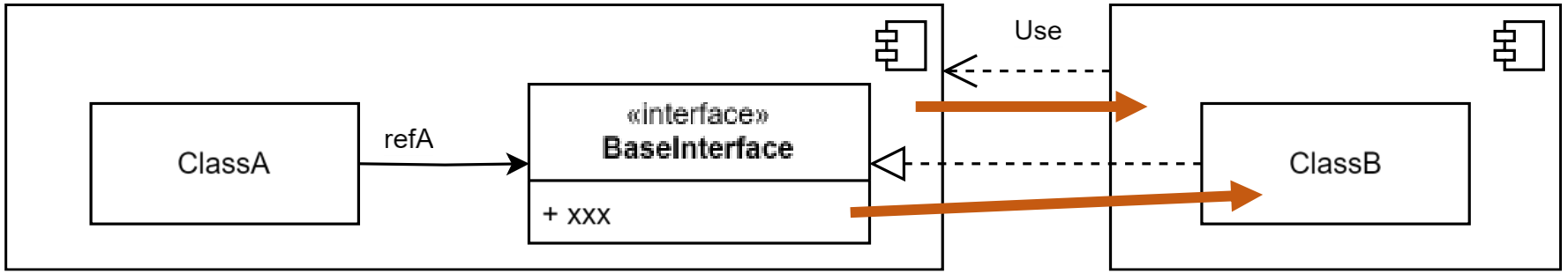
- Everything is a model
- Meta-model
- Objects/nodes
- Cross-referencing
- Inheritance
- Containment
- Interfaces
- Annotations
- Enumerations

- Concrete syntax grammar
- Algebraic data types
- Immutable values
- Model is a pure (parse) tree
- Alternative constructors for ADT
- Constructors have parameters

https://github.com/UlyanaTikhonova/lionweb-rascal

Challenges

- Language modularity and extension mechanism
  - Interfaces in LionWeb



- Delta protocol
  - How to preserve layout in a textual program?
- Generate LionCore language for an arbitrary Rascal grammar

What it means

QD0D

ProxyHands