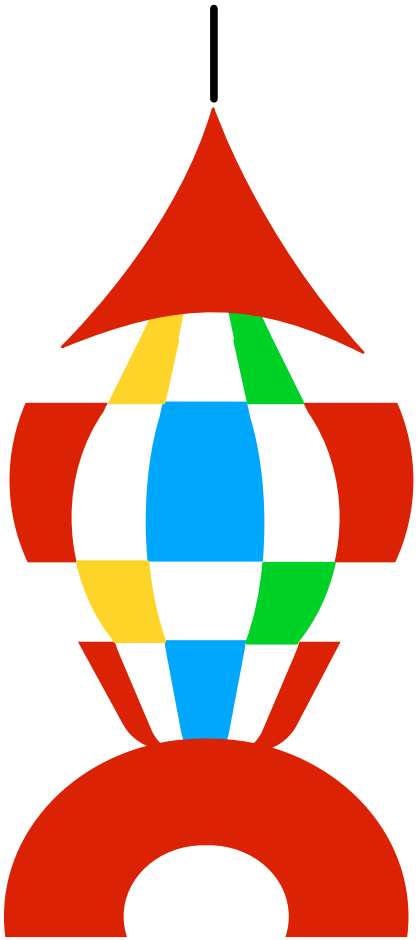
Kris Gybels

Roel Wuyts

Maja D'Hondt

Stéphane Ducasse



ESUG 2005
Brussels

# Inter-Language Reflection

# Executive Summary

Inter-Language Reflection = Linguistic Symbiosis + Reflection

SOUL ⇿ Smalltalk

Agora ⇿ Java

# Meta-Programming: A Review

Programming Tools

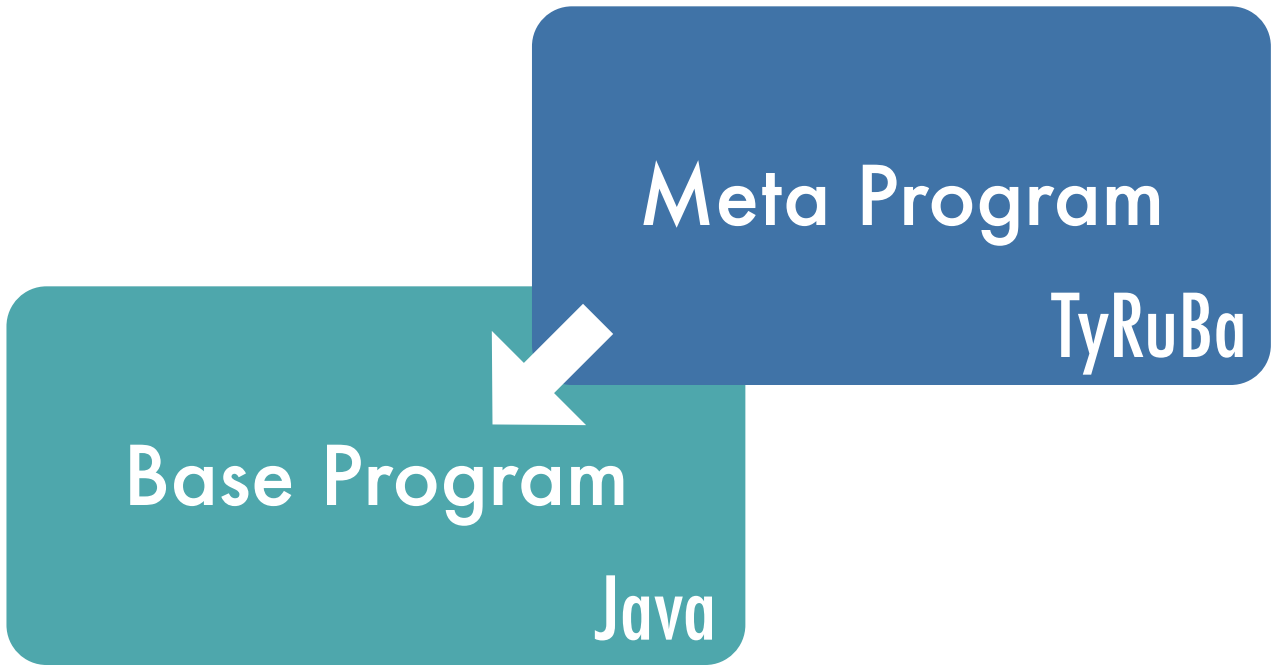
▼

Meta Programming

=

"Programs about programs"

# Different Base and Meta Languages




# No Causal Connection

```
class Leaf extends Node {
   void accept(Visitor v) {
      v.visitLeaf(this);
   }
}

class Visitor {
   void visitLeaf(Leaf l);
   void visitBranch(Branch b);
}
```

```
class(Leaf, Node).
class(Branch, Node).
class(Visitor, Object).
method(Leaf,accept, ...).
method(Node,visitLeaf, ...).
...
```

```
doubleDispatch(Class,Method) :-
  ...
visitor(Class) :-
  ...
```

# Reflection

```
MyOwnClass class>>compile:
  …
MyOwnClass class>>selectors
  …
```

```
MyOwnClass>>makeNewMethod: unarySelector
  MyOwnClass compile: unarySelector asString
 ^ MyOwnClass selectors size
```

# Introducing Inter-Language Reflection

**Multi-Paradigm Meta Programming**

Different Languages
No Causal Connection

TyRuBa ⇿ Java

**Reflection**

Single Language
Causal Connection

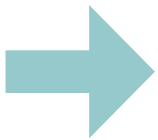
Smalltalk

▼

**Inter-Language Reflection**

Different Languages
Causal Connection

MyOwnClass



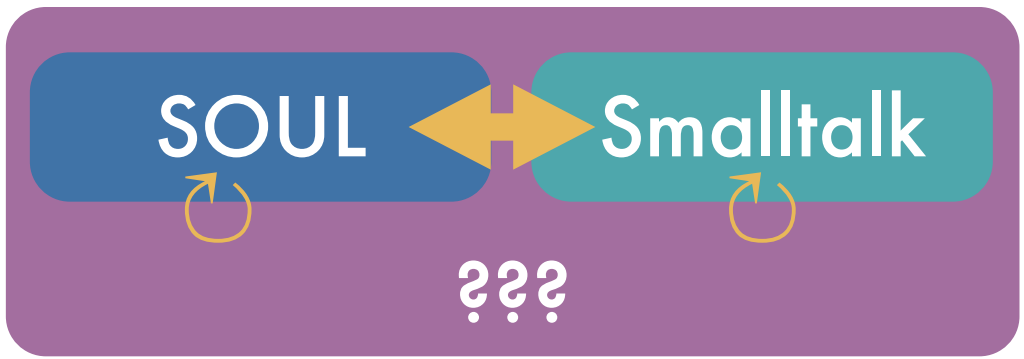
```
class(?c) if
  var(?c),
  member(?c, [ Class allInstances ]).
class(?c) if
  nonvar(?c)
  [ ?c isKindOf: Class ]
```

```
sends(?c, ?rec, ?sends) if
  class(?c),
  method(?c, ?m),
  sendsTo(?m, ?rec, ?sends).

generateEmptyMethod(?class, ?name) if
  emptyMethodSource(?name, ?source),
  [(?class compile: ?source) = nil]
```

# Understanding Inter-Language Reflection?

## Implementation


## Conceptual Model




=

Data Mapping

▼

Protocol Mapping

# Data Mapping in Agora: Example

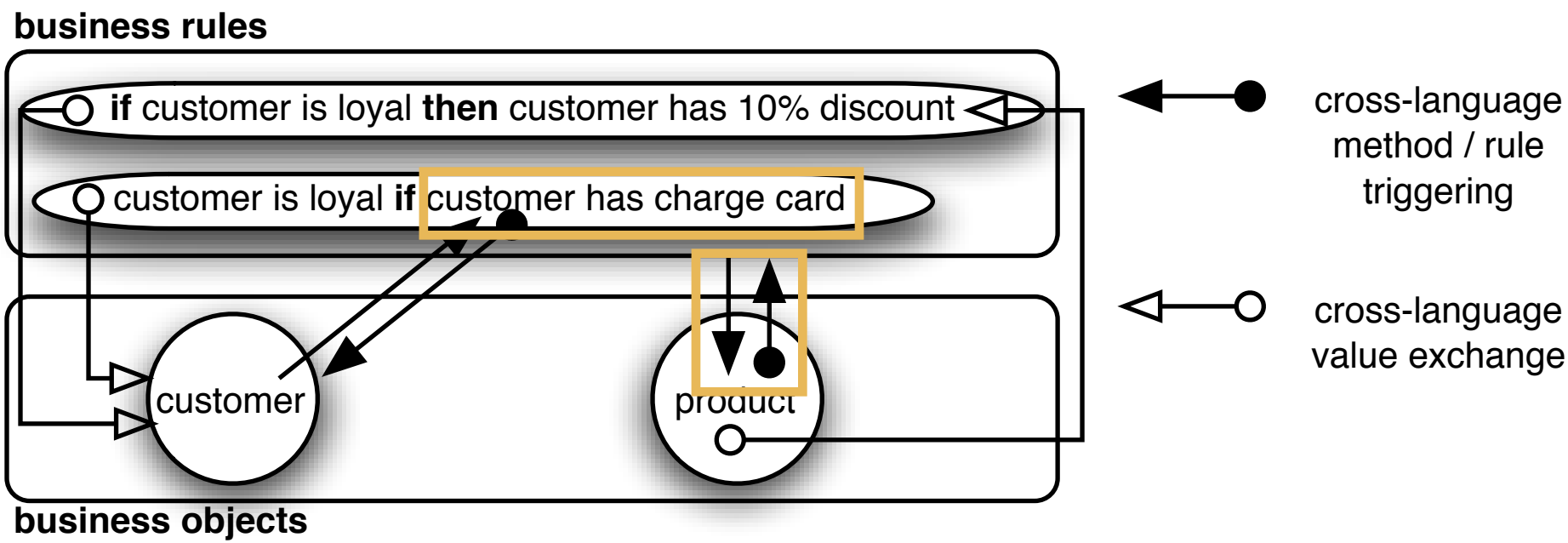
```
frame VARIABLE: ("java.awt.Frame" JAVA) new;

ok VARIABLE: ("java.awt.Button" JAVA) newString: "OK";
```

```
frame addComponent: ok;
```

```
okListener VARIABLE: [
  implements METHOD:
     (1 ARRAY: ("java.awt.event.ActionListener" JAVA));
  replaces METHOD:
     ("java.lang.Object" JAVA);
  actionPerformedActionEvent: e METHOD: {
    ("java.lang.System" JAVA) out printlnString: "Button Pressed!";
    frame setVisibleboolean: false
    }
  ];
```

```
ok addActionListenerActionListener: okListener
```

# Data Mapping in SOUL

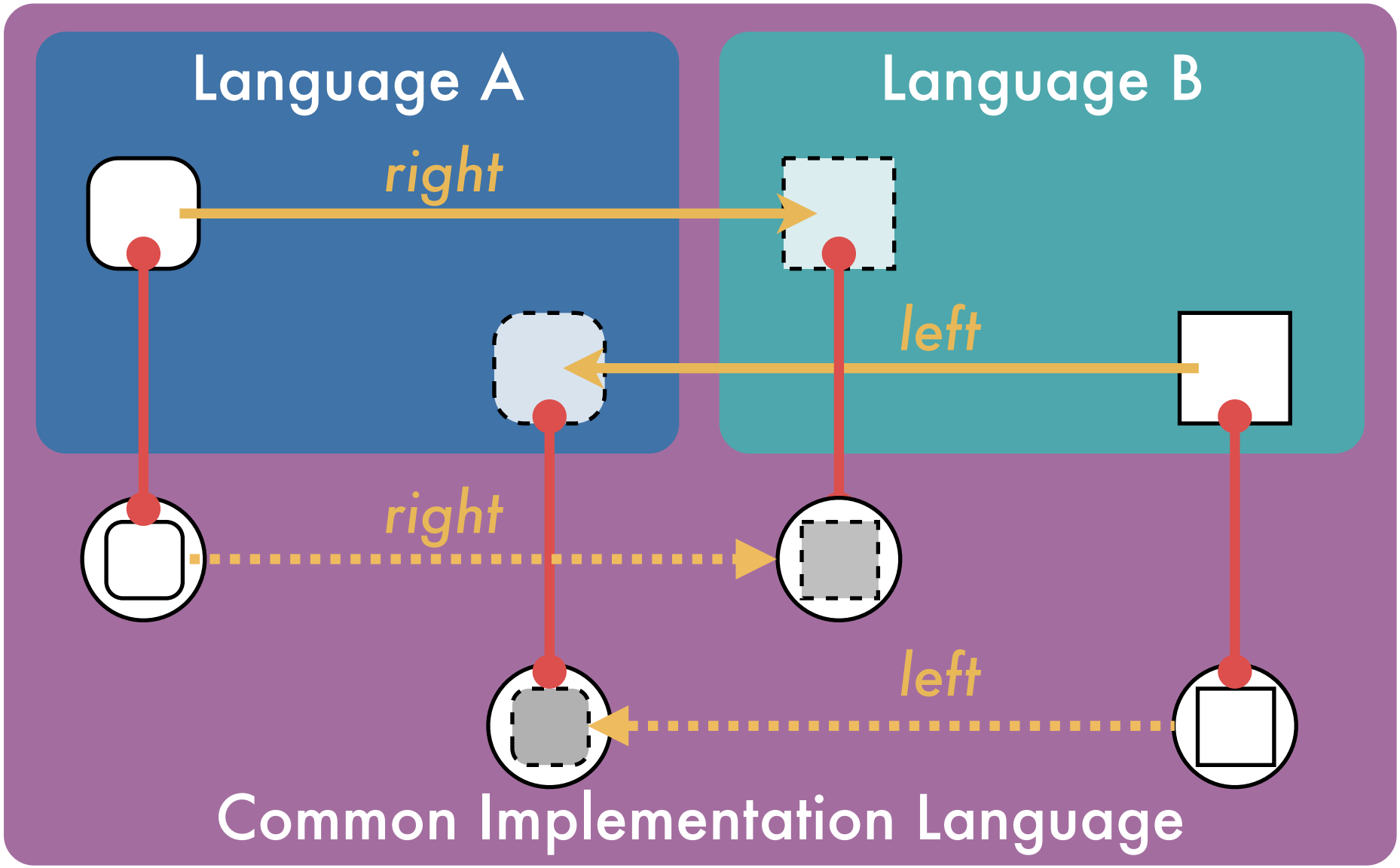



# Data Mapping in SOUL

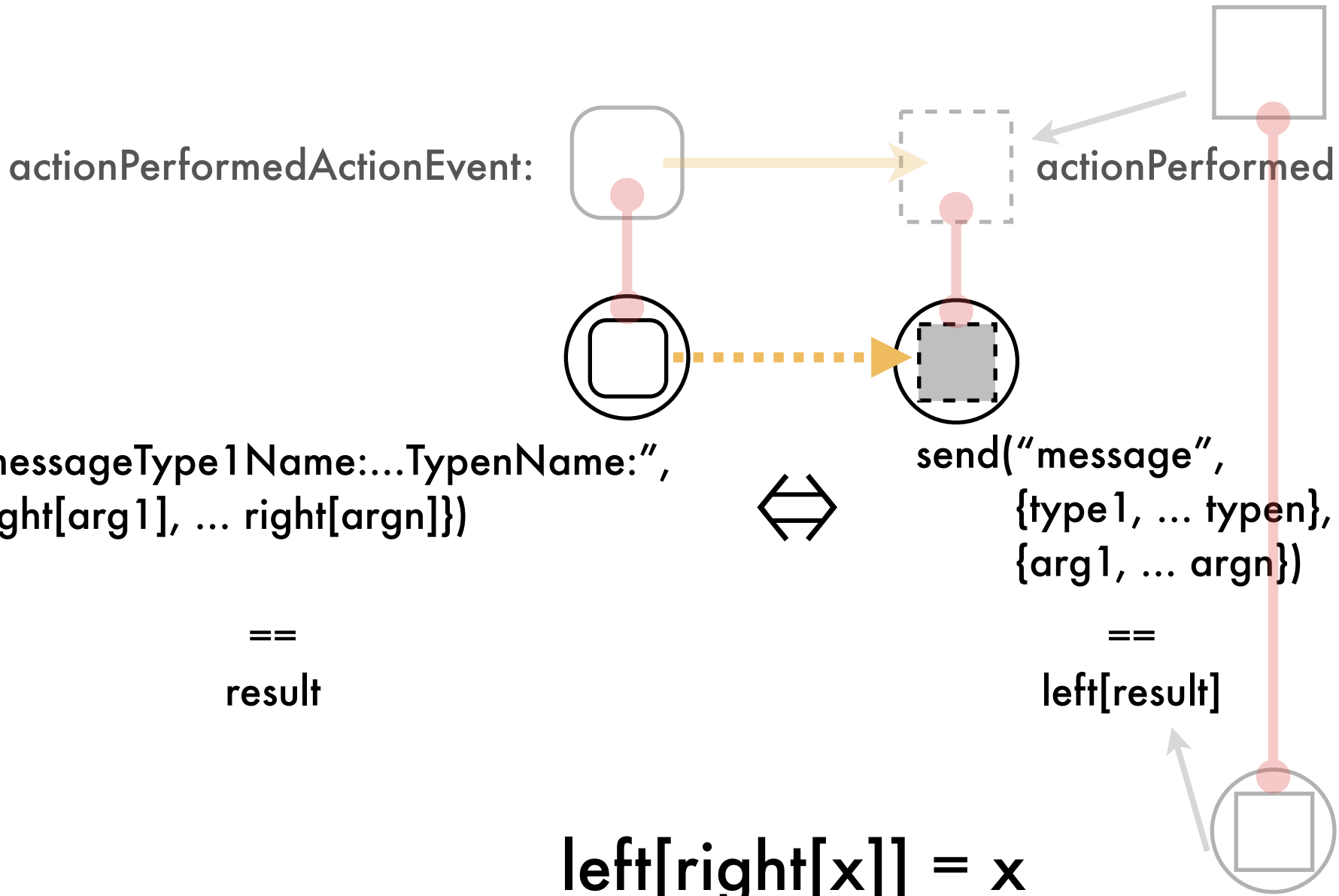
```
Product>>priceFor: aC
| discounts |

discounts :=
  (SOULEvaluator
      evaluate: 'discoun
      withArguments:
        (Array with: #cu
                  with: #p
       valuesForVariable: #discount.
^ price * (100 - discounts max) / 100
```

```
discount(?customer, ?product, 10) if
  loyal(?customer).
discount(?customer, ?product, 15) if
  student(?customer),
  educationalProduct(?product).

loyal(?customer) if
  [ ?customer hasChargeCard ].

educationalProduct(?product) if
  equals(?sectionitems, [ Section education products ]),
  [ ?sectionitems includes: ?product ]
```

# Protocol Mappings




# Agora ⇿ Java

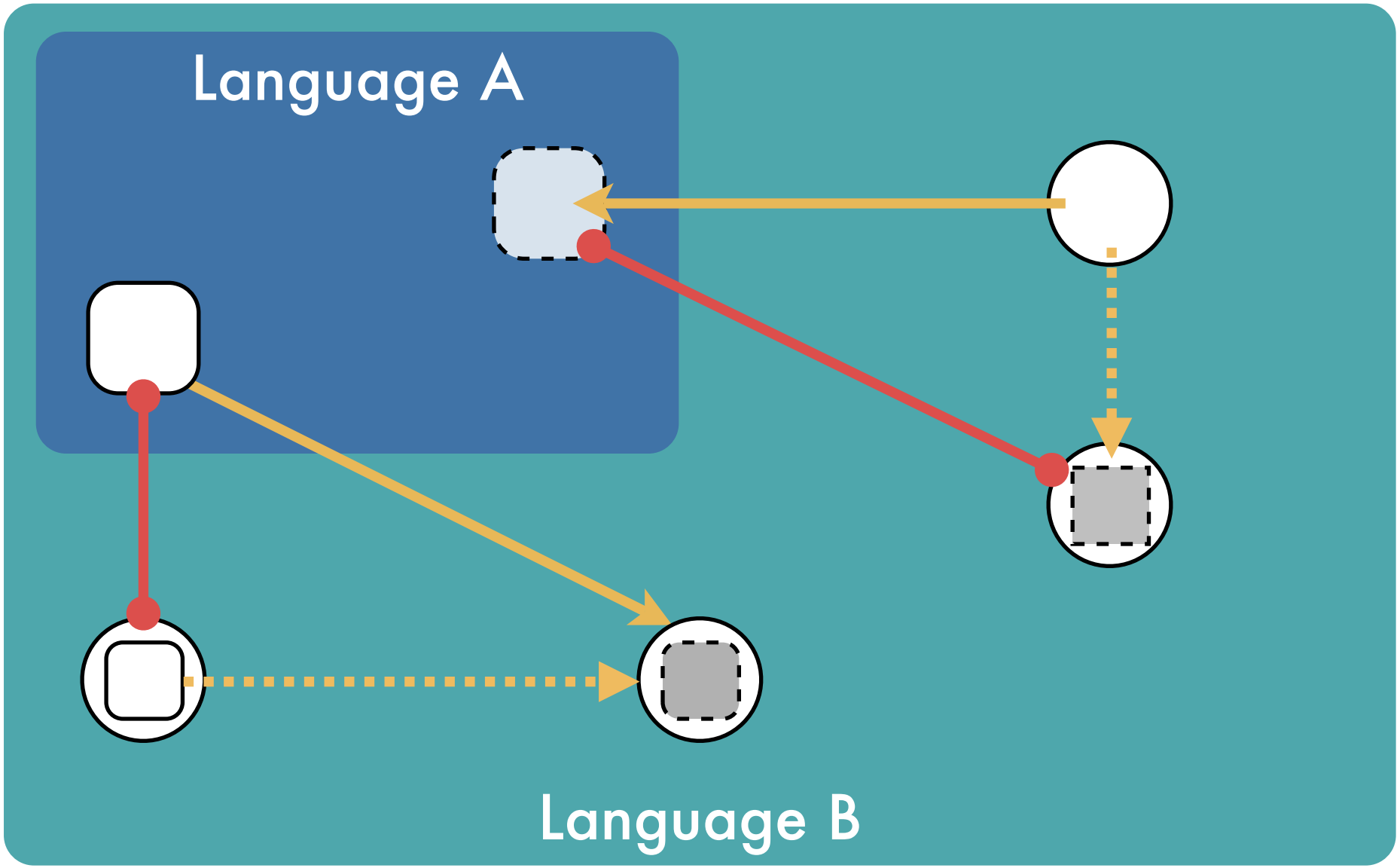



left[right[x]] = x

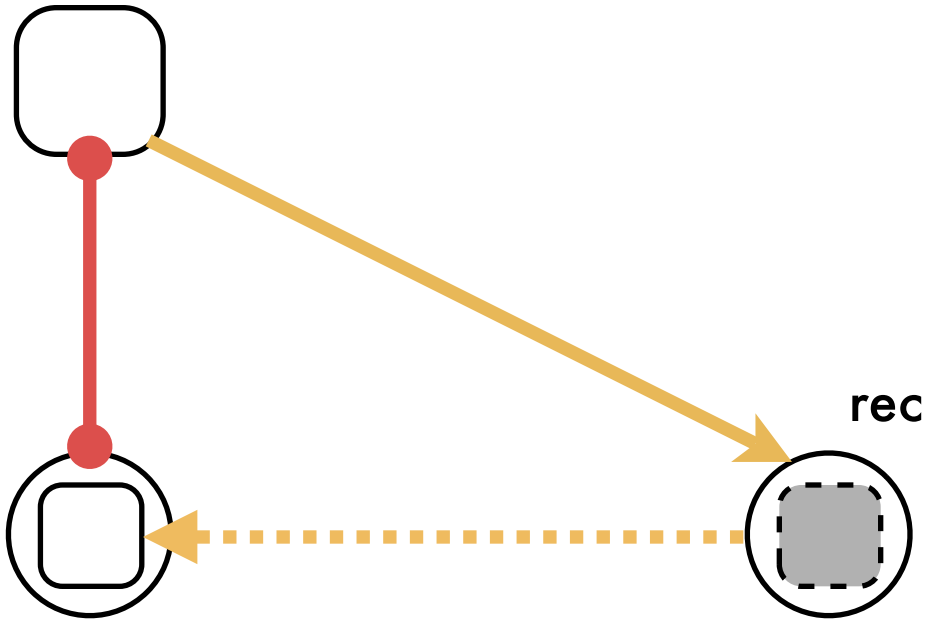
# SOUL ↔ Smalltalk

lo1 unify: lo2 ⇔ right[lo*1*] send: #= withArguments: { right[*lo2*] }

# Implementation Mapping




# Implementation Protocol Mapping



receiver.name(argument1, … argumentn)

==

right[result]

left[receiver].send("nameType1Name:…typenName",
{left[argument1], …left[argumentn]})

==

result

# Conclusion

Multi-Paradigm Meta Programming

Reflection

▶

Inter-Language Reflection

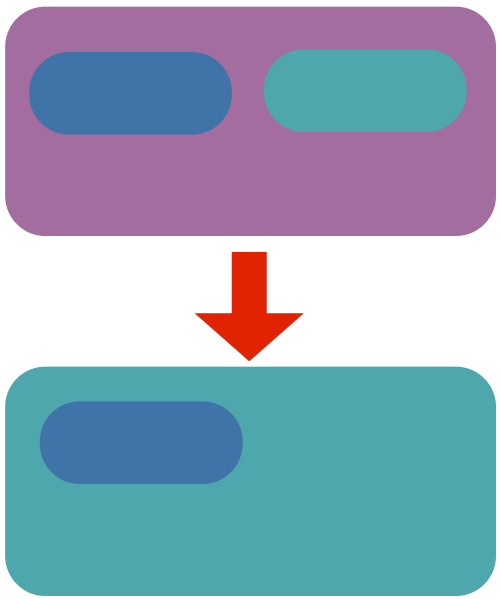
Different Languages
Causal Connection

=

Reflection

+

Linguistic Symbiosis



Data Mapping

▼

Protocol Mapping