PINOCCHIO

Bringing Reflection to Life with First-Class Interpreters

Toon Verwaest
Camillo Bruni
David Gurtner
Adrian Lienhard
Oscar Nierstrasz

Software Composition Group
University of Bern
Switzerland

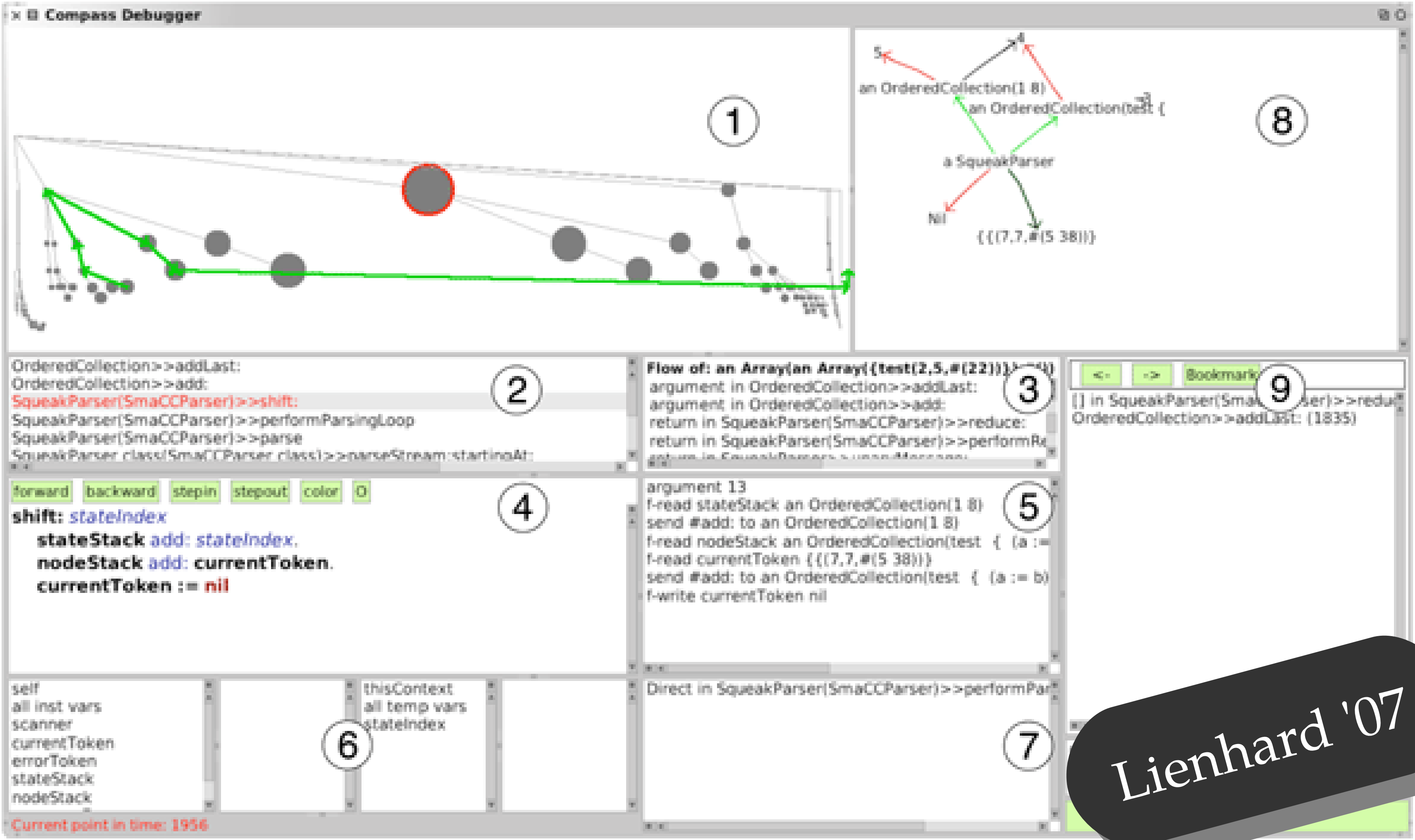
debugging is *hard*

developing debuggers is

even harder

```
System.out.println
```

Object-Flow Debugger



```
OrderedCollection>>addLast:
OrderedCollection>>add:
SqueakParser(SmaCCParser)>>shift:
SqueakParser(SmaCCParser)>>performParsingLoop
SqueakParser(SmaCCParser)>>parse
SqueakParser class(SmaCCParser class)>>parseStream:startingAt:
```

```
Flow of: an Array(an Array({test(2,5,#(22))}
argument in OrderedCollection>>addLast:
argument in OrderedCollection>>add:
return in SqueakParser(SmaCCParser)>>reduce:
return in SqueakParser(SmaCCParser)>>performRe
return in SqueakParser>>unaryMessage:
```

```
[] in SqueakParser(SmaCCParser)>>redu
OrderedCollection>>addLast: (1835)
```

```
shift: stateIndex
    stateStack add: stateIndex.
    nodeStack add: currentToken.
    currentToken := nil
```

```
argument 13
f-read stateStack an OrderedCollection(1 8)
send #add: to an OrderedCollection(1 8)
f-read nodeStack an OrderedCollection(test  (  (a :=
f-read currentToken {{(7,7,#(5 38))}
send #add: to an OrderedCollection(test  (  (a := b)
f-write currentToken nil
```

```
self
all inst vars
scanner
currentToken
errorToken
stateStack
nodeStack
```

```
thisContext
all temp vars
stateIndex
```

```
Direct in SqueakParser(SmaCCParser)>>performPar
```

Current point in time: 1956

Lienhard '07

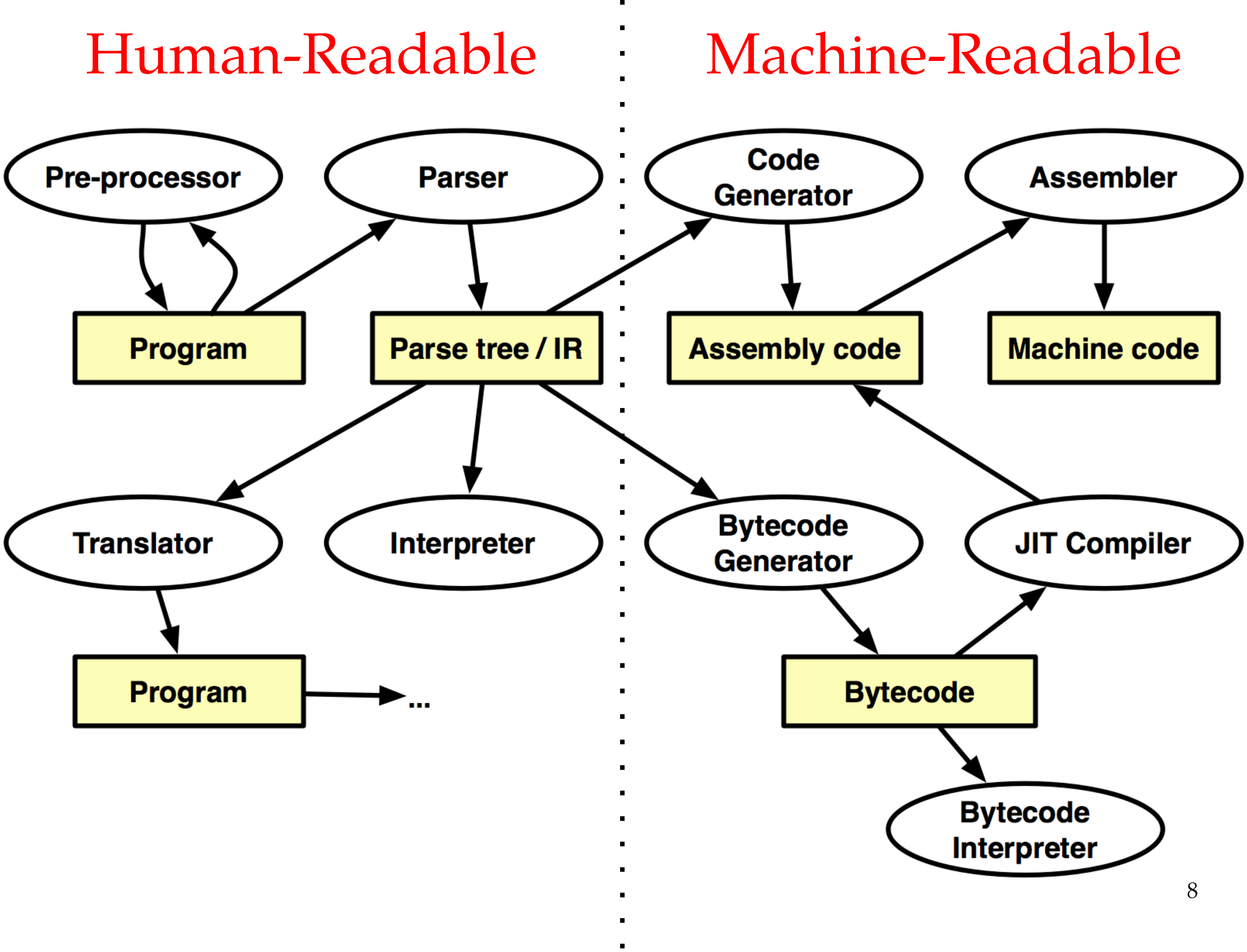
modifications to the Virtual Machine

A programming language

is a notational system for describing computation

in a machine-readable and human-readable form.

— *Louden*


Human-Readable
Machine-Readable
Pre-processor
Parser
Code Generator
Assembler
Program
Parse tree / IR
Assembly code
Machine code
Translator
Interpreter
Bytecode Generator
JIT Compiler
Program
...
Bytecode
Bytecode Interpreter

What if we

could build a

specialized debugger

in *just a few hours?*

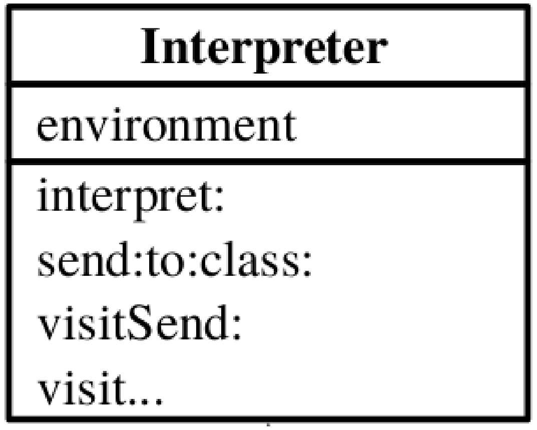
modify the interpretation

in the language itself

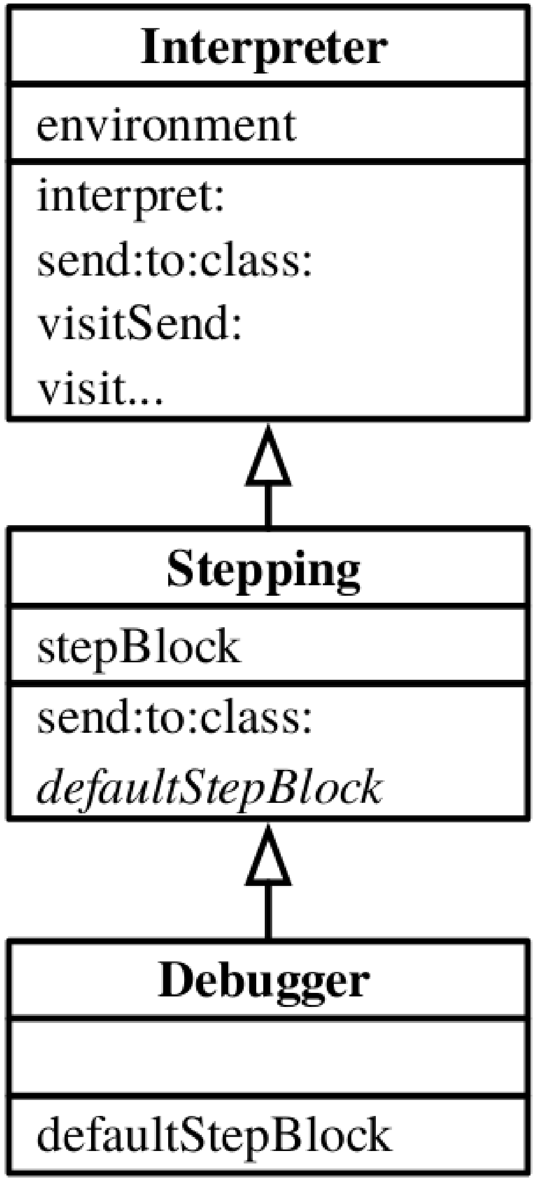
in terms of

the source code

PINOCCHIO


Interpreter
environment
interpret:
send:to:class:
visitSend:
visit...

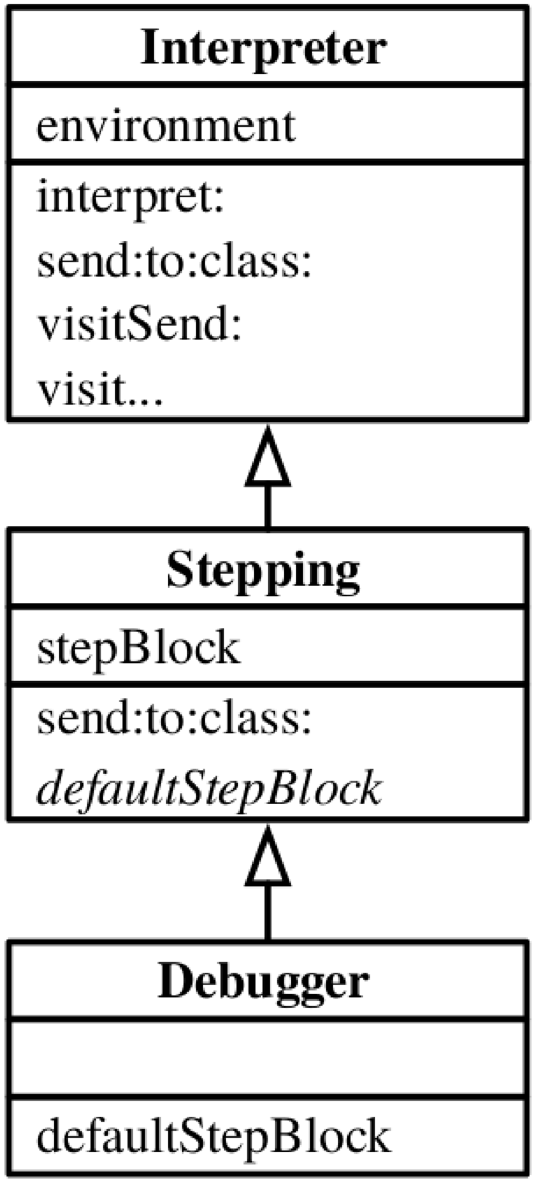
PINOCCHIO


Interpreter
environment
interpret:
send:to:class:
visitSend:
visit...
Stepping
stepBlock
send:to:class:
defaultStepBlock
Debugger
defaultStepBlock

```
Debugger interpret: [ Person new ]
Debugger interpret: [ Person new ]
```

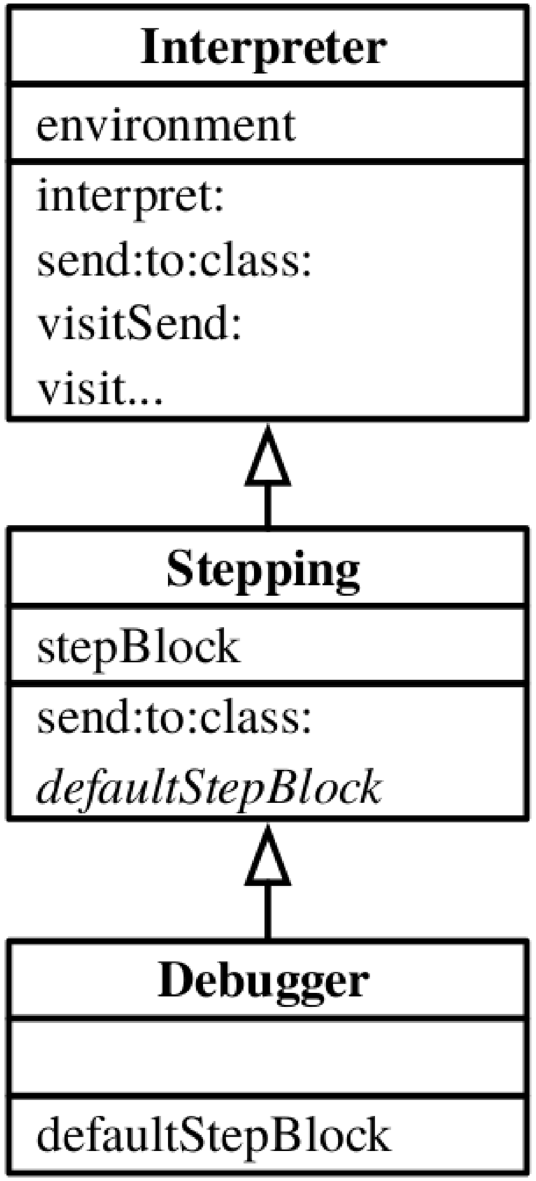
structural reflection

continuous behavioral reflection



```
send: message to: receiver: class: class
    self print:
        receiver class name, '>>', message.

    ^ self debugShellWithAction:[
        super
            send: message
            to: receiver
            class: class ]
```

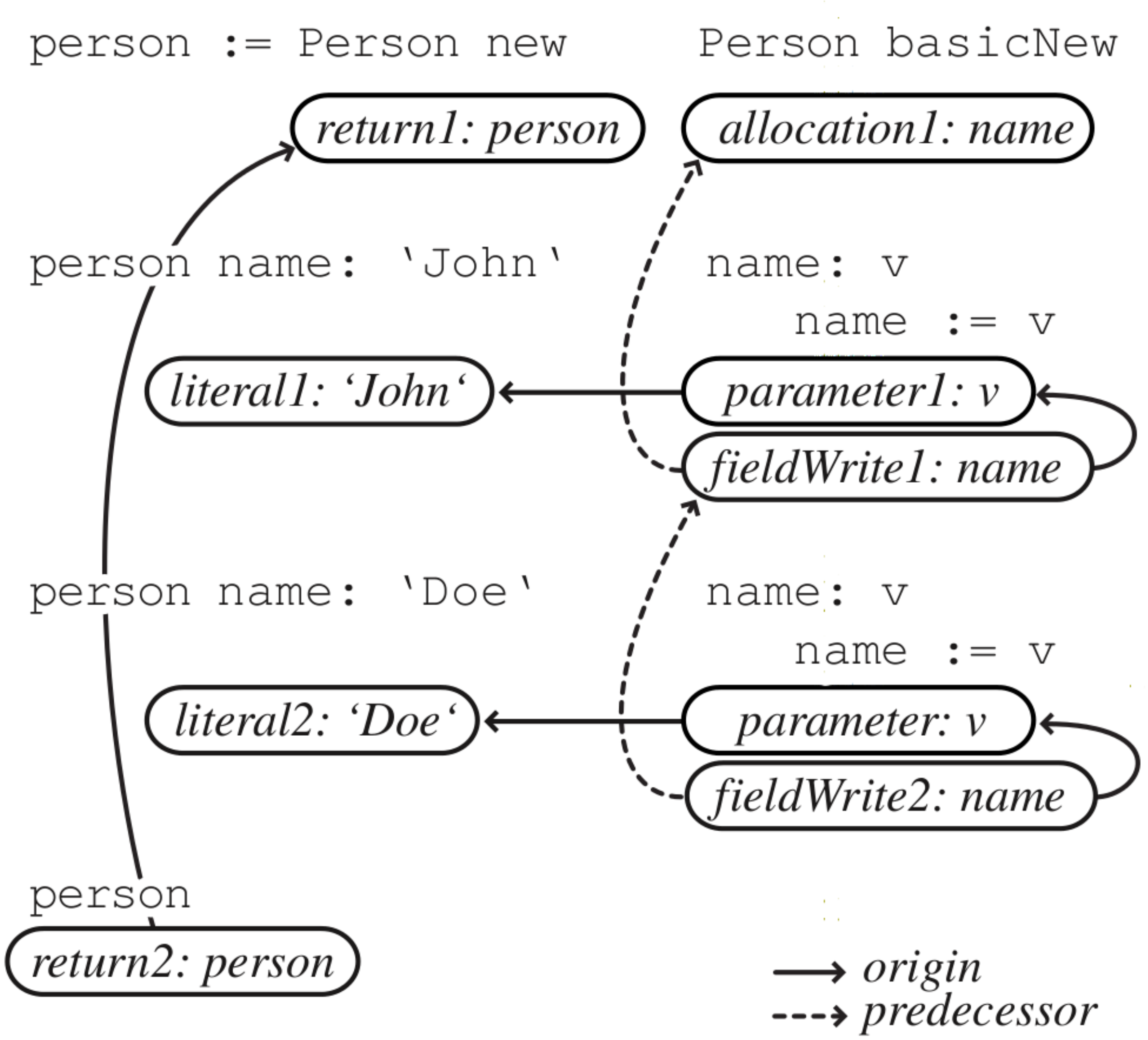


recursive AST visitors

garbage collection

object model

Alias Interpreter



```
AliasInterpreter
              interpret: [
   p := Person new.
   p name: 'John'.
   p name: 'Doe'.
]
```

Alias Interpreter

```
interpretMethod: method
   | result |
   result := super interpretMethod: method.
   ^ (ReturnAlias alias: result)
      environment: environment
```

Performance *(fib)*

Pinocchio	2x slower than Pharo
	2x slower than Ruby 1.9
	2x faster than Python 2.6.4
	5x faster than Ruby 1.8
Metacircular	160x slower Pinocchio

Performance *(fib)*

Pinocchio	2x slower than Pharo
	2x slower than Ruby 1.9
	2x faster than Python 2.6.4
	5x faster than Ruby 1.8
Metacircular	160x slower Pinocchio
Java	160x faster than Ruby 1.8

PINOCCHIO

- recursive AST visitors
- extensible using OO techniques
- implemented practical debuggers

Future work

- performance is not addressed yet