Advanced Reflection: MetaLinks

Marcus Denker, Inria

http://marcusdenker.de

Lecture at VUB Brussels, March 22, 2018
What we know...

- Smalltalk is reflective
- Classes, Methods, Stack-Frames... are Objects
- Reflective API on all Objects
Take home message

• Reflection is based on the meta-class model, thus inherently structural.

• Behavioural reflection limited to:
  • Method lookup upon failure (doesNotUnderstand: message)
  • Current execution reified (thisContext)
Can we do better?

• A more fine-grained reflective mechanism seems to be missing

• Let’s look again at a Method in the Inspector
Inspector on a Method
The AST

- **AST = Abstract Syntax Tree**
- Tree Representation of the Method
- Produced by the Parser (part of the Compiler)
- Used by all tools (refactoring, syntax-highlighting, ...)

Smalltalk compiler parse: 'test ^(1+2)'


The Compiler

- Smalltalk compiler -> Compiler Facade
- Classes define the compiler to use
  - You can override method #compiler
- Behind: Compiler Chain
The Compiler

Source → AST → Annotated AST

RBParser → OCSemanticAnalyzer

Annotated AST → IR → Bytecode

OCASTTranslator/IRBuilder → IRBytecodeGenerator
AST

- RBMethodNode: Root
- RBVariableNode: Variable (read and write)
- RBAssignmentNode: Assignment
- RBMessageNode: A Message (most of them)
- RBReturnNode: Return
AST: Navigation

- To make it easy to find and enumerate nodes, there are some helper methods

- CompiledMethod has: `#sendNodes`, `#variableNodes`, `#assignmentNodes`

- Every AST node has `#nodesDo:` and `#allChildren`
Inspect a simple AST

• A very simple Example

Smalltalk compiler parse: 'test ^(1+2)'
Integration

- Originally just internal to the compiler

- Pharo:
  - send #ast to a method to get the AST
  - Cached for persistency.

\[(\text{Point}>>\#x) \text{ast} == (\text{Point}>>\#x) \text{ast} \rightarrow \text{true}\]
Wouldn’t it be nice..

• With the AST, wouldn’t it be nice if we could use this structure for Behavioural Reflection?

• If we could somehow attach a “arrow to the code” that points to a meta-object

\[ \text{test}^{\wedge (1 + 2)} \]

meta-object for this Send
We have all pieces...

- We have the AST for each method
- It is quite simple
- We have a compiler in the system
- So this should be possible...
The MetaLink

link := MetaLink new
  metaObject: Halt;
  selector: #once;
  control: #before.

• MetaLink points to metaObject
• Defines a selector to call
• And a control attribute: #before, #after, #instead
• Installed on a AST node:

  (Number>>#sin) ast link: link
The MetaLink

- Can be installed on any AST Node

- Methods will be re-compiled on the fly just before next execution
  - Link installation is very fast

- Changing a method removes all links from this method
  - Managing link re-installation has to be done by the user
MetaLink: MetaObject

- MetaObject can be any object
- Even a Block: [Transcript show ‘hello’]
- Install on any Node with #link:
- de-install a link with #uninstall
MetaLink: Selector

- MetaLink defines a message send to the MetaObject
- #selector defines which one
- Default is #value
- Yes, a selector with arguments is supported
  - We can pass information to the meta-object
MetaLink: Argument

- The arguments define which arguments to pass
- We support a number of reifications
Reifications

- Reifications define data to be passed as arguments
- Reify —> Make something into an object that is not one normally
- Example: “All arguments of this message”
Reifications: examples

• All nodes: #object #context #class #node #link

• Sends: #arguments #receiver #selector

• Method: #arguments #selector

• Variable: #value

They are defined as subclasses of class RFReification
Reifications as MetaObject

• We support some special metaObjects:
  • #node  The AST Node we are installed on
  • #object  self at runtime
  • #class  The class the links is installed in
MetaLink: Condition

• We can specify a condition for the MetaLink

• Link is active if the condition evaluates to true

• We can pass reifications as arguments

```smalltalk
link := MetaLink new
    metaObject: Halt;
    selector: #once;

(Number>>#sin) ast link: link.
```
MetaLink: control

- We can specify when to call the meta-object
- We support `#before`, `#after` and `#instead`
- The instead is very simple: last one wins
Example: Log

• We want to just print something to the Transcript

```plaintext
link := MetaLink new
    metaObject: [Transcript show: 'Reached Here'].

(Number>>#sin) ast link: link
```
Recursion Problem

- Before we see more examples: There is a problem

- Imagine we put a MetaLink on some method deep in the System (e.g. `new`, `+`, `do:`).

- Our Meta-Object might use exactly that method, too

Endless Loop!!
Recursion Problem

- Solution: Meta-Level
- We encode the a level in the execution of the system
- Every Link Activation increases the level
- A meta-link is just active for one level. (e.g. 0)

```smalltalk
link := MetaLink new
    metaObject: [ Object new ];
    level: 0.

(Behavior>>#new) ast link: link.
```
Example: Log

- Better use #level: 0

- Nevertheless: be careful! If you add this to method called often it can be very slow.

```smalltalk
link := MetaLink new
    metaObject: [Transcript show: 'Reached Here'];
level: 0.
```
Example: Counter

- In the Browser you can add a “counter” to the AST
- See class ExecutionCounter

```smalltalk
install

    link := MetaLink new
    metaObject: self;
    selector: #increase.

node link: link.
```
Example: Breakpoint

- "Add Breakpoint" in AST (Suggestions) Menu
- See class Breakpoint
- Break Once
- Conditional Break

```metaclass
breakLink
  ^ MetaLink new
    metaObject: Break;
    selector: #break;
    options: options
```
Example: WatchPoint

- Watchpoint: Record Value at a point in the AST
- Example: Watch event in WorldMorph>>#mouseDown:

Click on background
-> value recorded
Example: WatchPoint

- Implementation: class Watchpoint, method install
- example of a #after link with a condition

```
link := MetaLink new
  metaObject: self;
  selector: #addValue:;
  arguments: #(value);
  control: #after;
  condition: [ recording ].
```
Example: Code Coverage

- Small Demo.
- Start with `CoverageDemo` new `openWithSpec`
Example: Code Coverage

- Example of a MetaLink with a #node MetObject

- Meta-Object is the node that the link is installed on

```plaintext
link := MetaLink new
    metaObject: #node;
    selector: #tagExecuted.
```
Interesting Properties

- Cross Cutting
  - One Link can be installed multiple times
  - Over multiple methods and even Classes
  - And across operations (e.g., Send and Assignment) as long as all reifications requested are compatible
- Fully Dynamic: Links can be added and removed at runtime
- Even by the meta-object of another meta-link!
Limitations

- Better use Pharo7 (we are improving it still)
- Still some bugs with #after on MethodNode
- Loops: next execution of a method. Need to restart long running loops (no on-stack replacement).
- Keep in mind: next metaLink taken into account for next method activation
- Take care with long running loops!
Help Wanted

- We are always interested in improvements!
- Pharo7 is under active development.
- Pull Requests Welcome!