Are functional languages a good way to represent productive meta–models?

Sébastien Mosser

Rainbow Team, I3S Lab, CNRS
University of Nice, France
http://rainbow.i3s.unice.fr

4th European Lisp Workshop
ECOOP’07, 07/30/2007, Berlin, Germany
Outline

1. Once upon a time . . .

2. The ADORE project, part I

3. About models transformations

4. Proof of concept: Arithmetic Expressions

5. The ADORE project, part II

6. . . . They lived happily ever after

Sébastien Mosser
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models?
Outline

1. Once upon a time . . .

2. The ADORE project, part I

3. About models transformations

4. Proof of concept : Arithmetic Expressions

5. The ADORE project, part II

6. . . . They lived happily ever after

Sébastien Mosser
CNRS, University of Nice, France
Are functional languages a good way to represent productive meta–models?
Once upon a time . . .

in a (not so) far away community

- **Model Driven Development**
  - Work on *models & meta-model*
  - Different levels of abstraction
- Towards productivity
  - From independance (*PIM*)
  - to specificity (*PSM*)
  - or technological domains bridges

Keystone : Model Transformations

Sébastien Mosser

Are functional languages a good way to represent productive meta–models?
Example: CommonLisp programs

- Meta–Model:
  - LISP language
- PIM\(^1\):
  - LISP program
- PSM\(^2\):
  - Executable code

```
(defun fac (n)
  (if (= n 1)
      n
      (* n (fac (- n 1))))
```

\(^1\)Platform Independent Model
\(^2\)Platform Specific Model

Sébastien Mosser
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models?
Challenges

- Model transformations
  - ATL: Pattern selection
  - Kermeta: Mop
  - ...

- Productivity: "Push the button!"
  - But you need such a button!
  - Define meta-models
  - Define languages syntax
  - Express translations
  - Express transformations
  - ...

Sébastien Mosser, CNRS, University of Nice, France
Outline

1. Once upon a time . . .
2. The ADORE project, part I
3. About models transformations
4. Proof of concept: Arithmetic Expressions
5. The ADORE project, part II
6. . . . They lived happily ever after

Sébastien Mosser  
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models?
ADORE : Aspects & Distributed ORchEstrations

Web Services
- Set of norms & protocols (HTTP, SOAP, XML, ...)
- Expose atomic business code in an interoperable way

Web Services Orchestrations
- Bounded language to express collaborations between services
- An orchestration is a Web Service

ADORE Challenges
- Cope with unexpected evolutions of systems
- Merge Orchestrations on demand

http://rainbow.i3s.unice.fr/adore
Illustration: An enterprise network

Are functional languages a good way to represent productive meta–models?
Are functional languages a good way to represent productive meta–models?
Are functional languages a good way to represent productive meta–models?
Once upon a time . . .

Adore, I

Transformations

Arithmetic Expressions

Adore, II

Happy end

The approach : a logic–based merge engine

Are functional languages a good way to represent productive meta–models ?
Models, tools & platforms

Languages:
- BPEL
- Boa

Merge:
- PROLOG

Execution:
- C#

Display:
- GraphViz

... 

Keystone: OMSM

Warning: Projections of Models!
## Outline

1. Once upon a time . . .

2. The ADORE project, part I

3. About models transformations

4. Proof of concept : Arithmetic Expressions

5. The ADORE project, part II

6. . . . They lived happily ever after
Challenges about Model Transformations

- **User:**
  - DSL as entry point
  - Model ???
  - ⇒ black box

- **Developer:**
  - Meta–Model as entry point
  - DSL to express model
  - ⇒ white box

**In a nutshell ...**

- User doesn’t want to know how system works!
- Developer wants to know:
  - How can I express transformations?
  - How should models looks like?
  - How can I reach multiple targets?

“Compile once, transform everywhere”!
Possible solutions

Do it by hand!

Use existing tools
cf. doc p.743

Use a simple approach
Tree-driven evaluation

Are functional languages a good way to represent productive meta-models?
Let’s go using Lisp

1. DSL $\rightarrow$ LISP : Compilation
2. A transformation $\equiv$ A set of functions
3. Model Transformation $\equiv$ Evaluation
Once upon a time . . .

1. Once upon a time . . .

2. The ADORE project, part I

3. About models transformations

4. Proof of concept : Arithmetic Expressions

5. The ADORE project, part II

6. . . . They lived happily ever after
A simple Meta–Model : Arithmetic Binary Expressions

Sébastien Mosser
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models ?
A simple DSL: Arithmetic Binary Expressions

\[
\text{ROOT} ::= \ <\text{EXPR}\ > \\
\text{EXPR} ::= 0 \mid [1-9]^+[0-9]^* \mid (\ <\text{EXPR}\ > ) \\
 \quad \mid \ <\text{EXPR}\ > + \ <\text{EXPR}\ > \mid <\text{EXPR}\ > - <\text{EXPR}\ > \\
 \quad \mid <\text{EXPR}\ > \ast \ <\text{EXPR}\ > \mid <\text{EXPR}\ > / <\text{EXPR}\ >
\]
Model Example: an expression

- **DSL representation:**

\[ e = \frac{((20 + 32) - (5 \times 4))}{2} \]

- **UML representation:**

![UML Diagram]

Are functional languages a good way to represent productive meta-models?
## Model transformations targets

<table>
<thead>
<tr>
<th>Language</th>
<th>Concrete Syntax describing $e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
<td><code>dc -e &quot;20 32 + 5 4 * - 2 / p&quot;</code></td>
</tr>
<tr>
<td>LISP</td>
<td><code>(format t &quot;~a~%&quot; (/ (- (+ 20 32) (* 5 4)) 2))</code></td>
</tr>
<tr>
<td>PYTHON</td>
<td><code>print ((20+32) - (5*4)) / 2</code></td>
</tr>
</tbody>
</table>

$$e = \frac{(20 + 32) - (5 \times 4)}{2}$$
Transformation algorithm

- Compilation: from DSL to s–expr

\[
\text{(setq e } \\
\text{ (quote (root (divide (minus (plus 20 32) } \\
\text{ (star 5 4)) } \\
\text{ 2)))))}
\]

Another model representation!

- Model transformation: read, eval, execute!

\[
\text{(defun transform (model functions) } \\
\text{(load functions) } \\
\text{(eval model))}
\]
Lisp transformation

- Transformation definition

```
(defun root (exp) (format nil "~a = ~a" exp (eval exp)))
(defun divide (a b) (list (quote /) a b))
(defun minus (a b) (list (quote -) a b))
(defun plus (a b) (list (quote +) a b))
(defun star (a b) (list (quote *) a b))
```

- Transformation session

```
[1]> (load "transfo.l")
;; Loading file transfo.l ...
;; Loaded file transfo.l
T
[2]> (transform e "lisp.l")
;; Loading file lisp.l ...
;; Loaded file lisp.l
"(/ (- (+ 20 32) (* 5 4)) 2) = 16"
[3]> (exit)
```

Sébastien Mosser
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models?
Dc transformation

- **Transformation definition**

  ```lisp
  (defun root (exp) (format nil "dc\-e\-" exp))
  (defun divide (a b) (format nil "~a/~a/" a b))
  (defun minus (a b) (format nil "~a/~a-" a b))
  (defun plus (a b) (format nil "~a/~a+" a b))
  (defun star (a b) (format nil "~a~a*" a b))
  ```

- **Transformation session**

  ```lisp
  [1]> (load "transfo.l")
  ;; Loading file transfo.l ...
  ;; Loaded file transfo.l
  T
  [2]> (transform e "dc.l")
  ;; Loading file dc.l ...
  ;; Loaded file dc.l
  "dc\-e\-"20\-32\+\-5\*\-2\/\p"
  [3]> (exit)
  ```
Outline

1. Once upon a time . . .

2. The ADORE project, part I

3. About models transformations

4. Proof of concept : Arithmetic Expressions

5. The ADORE project, part II

6. . . . They lived happily ever after

Are functional languages a good way to represent productive meta–models?
Once upon a time . . .

Adore, I

Transformations

Arithmetic Expressions

Adore, II

Happy end

ADORE entry point: aBstract Orchestration pAttern

- A simple Dsl expressing behaviour
- more elegant than XML based verbose dialects

```plaintext
pattern AddPartner {

    partner InfoProvider, NewService

    rule InfoProvider::GetInformations -> {
        var legacy, newL, result as adore:list
        ( legacy <- proceed() //
          newL <- NewService::GetNewInformations() ) ;
        result <- Concat(legacy,newL);
        reply result
    }
}
```

Sébastien Mosser CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models?
Once upon a time . . .

Adore, I

Transformations

Arithmetic Expressions

Adore, II

Happy end

./boac.sh -i docs/AddPartner.boa -t scheme

(pattern 'AddPartner
  (partners (list 'InfoProvider 'NewService))
  (rules (list (rule
    (target 'InfoProvider "GetInformations"
    (variables
      (list (variable 'legacy ;; ... )))
    (instructions
      (list
        (instruction 'i236
          (list 'boa:proceed (list))
          (inputs (list))
          (outputs (list (ref_var (;; ...))))
          (conditions (list))) ;; ...))
    (orders (list
      (order 'i236 'i238) ;; ... ))))))

Sébastien Mosser
CNRS, University of Nice, France

Are functional languages a good way to represent productive meta–models ?
(define (partners L)
  (let*
    ((s (apply string-append
              (map
                (lambda (e)
                  (format "~a|" e))
              L)))
     (l (string-length s))
     (ps (substring s 0
           (if (= 0 l)
               0 (- l 1))))))
  (format "partners\|...\|\{Name|\~a\}\|\|;" ps)))
namespace Adore.Workflows
{
  public sealed class AddPartner
  {
    public static AWorkflow GetWorkflow()
    {
      AWorkflow _w = new AWorkflow();
      _w.ActivatePartner("InfoProvider");
      _w.ActivatePartner("NewService");
      Variable legacy = new AdoreList("legacy"); _w.AddVariable(legacy);
      Variable new = new AdoreList("new"); _w.AddVariable(new);
      Variable result = new AdoreList("result"); _w.AddVariable(result);
      Activity i236 = new AdoreProceed();
      _w.AddActivity(i236);
      Activity i237 = new AdoreInvoke("NewService", "GetNewInfo"),
                    new Reference[] {null}
                         new VariableReference(_w, legacy),
                         new VariableReference(_w, new),
                         new VariableReference(_w, result);
      _w.AddActivity(i237);
      Activity i238 = new AdoreAssign(new Reference[] {null})
                  new VariableReference(_w, legacy),
                  new VariableReference(_w, new),
                  new VariableReference(_w, result),"Concat":
                    _w.AddActivity(i238);
      Activity i239 = new AdoreProceed();
      _w.AddActivity(i239);
      _w.AddWait(i236, i238);
      _w.AddWait(i237, i238);
      _w.AddWait(i238, i239);
      return _w;
  }
}

using System;
using Adore.Workflow.Execution;

class Program
{
  static void Main()
  {
    AWorkflow workflow = Adore.Workflows.AddPartner.GetWorkflow();
    workflow.Execute();
  }
}
Outline

1. Once upon a time . . .
2. The ADORE project, part I
3. About models transformations
4. Proof of concept: Arithmetic Expressions
5. The ADORE project, part II
6. . . . They lived happily ever after

Are functional languages a good way to represent productive meta–models?
Summary: Today, we have seen, ... 

- How s-exprs can represent models
- How “evaluation” sounds like “transformation”
- How it works:
  - on a proof of concept
  - on a real application
Open questions

- How about performances & scalability?
- How about evaluation order?
- How about model evolutions & tracability?
Once upon a time . . .
Adore, I
Transformations
Arithmetic Expressions
Adore, II
Happy end

. . . They lived happily ever after

- Object Oriented languages foundations come from Lisp:
  - CLOS, MOP, Dynamic typing, . . .
- Model Driven Development comes from the Object World
- Functional languages can express MDD concepts
- Let’s fill the gap!

Questions ? Remarks ?