Ambiguous, Informal, and Unsound: Metaprogramming for Naturalness

Toni Mattis, Patrick Rein, Robert Hirschfeld
Software Architecture Group
Hasso Plattner Institute, University of Potsdam, Germany

META’19  20 Oct. 2019, Athens, Greece
Rectangle >> drawOn: aCanvas
   self visible ifTrue:
       [aCanvas paint: self bounds
           color: self color]

 Rectangle >> drawOn: aCanvas
 self visible ifTrue: [aCanvas paint: self bounds
 color: self color]

Penguin >> slipOn: anIceFloe
 self clumsy ifTrue: [
 anIceFloe paint: self bounds
     color: self color]
Systems runtimes, tools

Humans organization, future self
Primary notation
Behaviorally significant

Secondary notation
Perceptually significant

Verbal Cues

Visual Cues

Rectangle >> drawOn: aCanvas
self visible ifTrue:
[aCanvas]
paint: self bounds
color: self color]
[message send] [block]
**Primary notation**
Behaviorally significant

**Secondary notation**
Perceptually significant

- Formal, unambiguous
- Soundness guarantees
- Accessible through Metaprogramming

- Informal, ambiguous
- Used inconsistently
- Subject to (approximate) interpretation
Primary notation
Behaviorally significant

Rectangle >> drawOn: aCanvas
self visible ifTrue:
   aCanvas paint: self bounds
color: self color

Secondary notation
Perceptually significant

Rectangle >> drawOn: aCanvas
self visible ifTrue:
   aCanvas paint: self bounds
   color: self color

» Accessible through Metaprogramming

» Subject to (approximate) interpretation
Examples of Secondary Notation

Names

- aCanvas
- canvas

Comments

- “Re-draw when invalidated”

Values

- 404
- ‘error.log’

Space

Alignment

Order

Co-occurrence
Machine Learning (ML) Example: Word Embedding

ML / Natural Language Processing (NLP)

Canvas

Rectangle

Bounds x y

Color

Paint

Canvas (0.12, 0.94, ...)

Vector space
Features

ML/NLP Model

Application

ML/NLP Model

Features

Meta-objects

class

method

method

Compilation

Debugging

Analysis

Refactoring

...
Features

ML/NLP Model

Application

Recommender
Metrics
Bug-finding
...

Compilation
Debugging
Analysis
Refactoring
...

Meta-objects

mapping

class

method

method

Mattis, Rein, Hirschfeld | META, Athens, 2019 | Software Architecture Group, HPI Potsdam
Meta-objects
Reified representations of a program’s (formal) elements

Rectangle

m := Rectangle methodDict at: #drawOn:
m parseTree body first

a Class
a CompiledMethod
The first statement of m

formal representation

Rectangle

class c

name

methods

m1

name
drawOn:
m2
Representing Meta-objects

Formal representation

ML representation

Rectangle

class c

name

methods

m1

name

drawOn:

(0.41, 0.68, ...)

m2

m1
A Framework for “Meaning”

word2vec, topic models, …

:Model

:Meaning

model

(subject)

{ m : Method }

model meaningOf: m

Meta-object
Comparing Meanings

\[ m_1 \text{ meaning} \leftrightarrow m_2 \text{ meaning} \]

\begin{itemize}
  \item \textit{m1 : Method}
  \item \textit{m2 : Method}
\end{itemize}

\begin{itemize}
  \item Meta-object
  \item subject
\end{itemize}

\[ m_1 \text{ meaning} \prec m_2 \text{ meaning} \]
Comparing Meanings

Test Prioritization

```
{  
  drawOn: 
    bounds
}
```

```
Rectangle
```

```
RectangleTest
  testColor
  testDrawing
  testSize
```
Comparing Meanings

Test Prioritization

\[
m_1 \text{ meaning } \leftrightarrow m_2 \text{ meaning}
\]

Rectangle

\[
\text{drawOn:}
\]

\[
\text{bounds}
\]

RectangleTest

1. testDrawing

2. testColor

3. testSize
Replace `self tests` in the `run` method with call to:

```
TestSuite >> prioritizedTestsGiven: aMethodChange
| changeMeaning |
changeMeaning := aMethodChange newMethod meaning.
^ self tests sorted: [:testMethod1 :testMethod2 |
  (testMethod1 meaning < - changeMeaning)
  > (testMethod2 meaning < - changeMeaning)]
```
Composition

» Represent **classes** as composition of name, fields, doc-string, and methods

» Represent **changes** as composition of modified meta-objects

m1 meaning `composeWith: m2 meaning`

Meaning `composedFrom: {m1. m2. ... }`
Destructuring Composition

» Not all models allow **immediate** composition (e.g. addition in vector space vs. maximum-likelihood)
Use Cases

» Test Prioritization
» Refactoring [s. Paper]
» Metrics/Linting
  › Intra-class similarity ~ cohesion
  › Inter-class similarity ~ coupling

» Code Completion
  › Incomplete code without valid meta-object representation can have a representation in the model

» ...

Implementation: Scope

model := LDAModel trainOn: #package alpha: 0.05 beta: ...
model do: ['The LDA model is valid here']

Model >> do: aBlock
  ^ActiveModel value: self during: aBlock
  Dynamic variable

CompiledMethod >> meaning
  ^ActiveModel value meaningOfMethod: self

Class >> meaning
  ^ActiveModel value meaningOfClass: self

model meaningOf: anObject

More explicit?
Implementation: Minimal Core

» A Model only needs to...
  › represent features
    (lexical tokens in identifiers, strings, symbols)
  › provide composition and comparison

» Default implementations (via double dispatch)
  › Methods as composition of identifiers
  › Classes as composition of name, fields, and methods
  › ...

Meta-objects?

Smalltalk
Same life-cycle

Java/.NET/C[++]
Conclusion

- Programs are “natural” artifacts of human communication
- “Natural” properties are usable through ML but not reified in metaprogramming, yet
- We explored designs to expose “ML/NLP knowledge” at meta-object level

- How do we want to use secondary notation & meaning through metaprogramming?