Transformations Debugging
Transformations

Māris Jukšs, Clark Verbrugge, Hans Vangheluwe

McGill University
Canada
{mjukss,clump}@cs.mcgill.ca

University of Antwerp
Belgium
hans.vangheluwe@uantwerp.be

September 2017
Model Transformation Debugging

- MT stack is complex, heterogeneous
- Formalisms at different levels of abstractions

```
==5?
```

```
S=2
```
Model Transformation Debugging

- MT stack is complex, heterogeneous
- Formalisms at different levels of abstractions
Model Transformation Debugging

• Problems can occur at any level/abstraction
  – Or in layer interactions

• How to build a general debugging design?
Structured View

- General, structured view
- Execution traverses the structure
  - In/out of abstractions
  - Within abstractions
Structured View

• Navigation pointer
  – Coordinate (tuple) to describe control flow position

• “Vertical” dimension
  – hierarchy, abstraction level

• “Horizontal”
  – activity within an abstraction
Structured View

- Navigation pointer
Structured View

- Navigation pointer
Debugger Commands

- Basic ops, in each abstraction
  - Next / Step-over
  - Down / Step-into
  - Up / Step-out
  - Continue
  - Pause / Break
Execution Environment

• Evolution of Navigation Pointer (NP)
  – (IP,LP) changes by commands
  – some symbolic values too

1: x=3
2: foo(x)
3: print x

5: foo(val):
6: val=val+1
Execution Environment

- Evolution of Navigation Pointer (NP)
  - \((IP,LP)\) changes by commands
  - some symbolic values too

1: \(x = 3\)
2: \(\text{foo}(x)\)
3: print \(x\)
5: \(\text{foo}(\text{val})\):
6: \(\text{val} = \text{val} + 1\)
Execution Environment

• Evolution of Navigation Pointer (NP)
  - (IP,LP) changes by commands
  - some symbolic values too

1: x=3
2: foo(x)
3: print x

5: foo(val):
6: val=val+1
Execution Environment

- Evolution of Navigation Pointer (NP)
  - (IP,LP) changes by commands
  - some symbolic values too

\[(1, \text{TOP}) \rightarrow \text{Next} \rightarrow (2, \text{TOP}) \rightarrow \text{Next} \rightarrow (3, \text{TOP}) \rightarrow \text{Next} \rightarrow (\text{NULL}, \text{TOP})\]

1: \(x=3\)
2: \(\text{foo}(x)\)
3: \(\text{print } x\)
5: \(\text{foo}(\text{val}):\)
6: \(\text{val}=\text{val}+1\)
Execution Environment

- Evolution of Navigation Pointer (NP)
  - (IP,LP) changes by commands
  - some symbolic values too

1: x=3
2: foo(x)
3: print x
5: foo(val):
6: val=val+1
Execution Environment

- Evolution of Navigation Pointer (NP)
  - (IP, LP) changes by commands
  - some symbolic values too

1: x = 3
2: foo(x)
3: print x

5: foo(val):
6: val = val + 1
Execution Environment

- Evolution of Navigation Pointer (NP)
  - $(IP,LP)$ changes by commands
  - some symbolic values too

1: $x=3$
2: $\text{foo}(x)$
3: print $x$
5: $\text{foo}(\text{val})$:
6: $\text{val} = \text{val}+1$
Execution Environment

- Evolution of Navigation Pointer (NP)
  - (IP,LP) changes by commands
  - some symbolic values too

1: x=3
2: foo(x)
3: print x
5: foo(val):
6: val=val+1

Breakpoint is just an NP value
Execution Environment

• NP evolution depends on formalism
  - execution semantics
  - layers underneath
• E.g., rule schedule
Graphical Formalism
Pattern Matching & Application

• Match (set constructions)
  – Candidates, bindings
  – Match
  – Matchsets

• Application
  – C(R)UD sets
Interaction

- Debugger itself as model transformation (rules)
- Debugging rules reuse MT syntax
Interaction

- Debugger itself as model transformation (rules)
- Debugging rules reuse MT syntax
Interaction

• When, to what does the LHS apply?
  – Input model
  – MT specification
  – Pattern matching / application
Debugging Semantics

- Scenarios

E.g.: Step over until circle created

Then pause
Debugging Semantics

- Syntactic sugar to reduce tedium
Architecture

System

Debugging Target

MT

Commands

NP

Debugger Controller Statechart

Debugger

Toolbar

Debugging Scenario

Commands
Statechart Controller
Action Code Stepping

Rule: R_moveCar.model
LP att_position
IP: user_line

```python
1: def x():
2:   k=0
3:   for x in range(2):
4:     k+=1
5:   return k
6:
7:   n=2
8:   y=x()
9:   result = True
```
Other Implementations

- AToM3
  - Easy
Other Implementations

- ATL
  - reimplement Statechart controller
  - work at VM level
Conclusions

- Structured view of debugging
  - Flow within, between abstractions
- Debugging language
- Familiar MT syntax
  - Separate control flow
- Reuse problem DSL in queries
- Interactive and batch modes
Thanks

Transformations Debugging Transformations

Māris Jukšs, Clark Verbrugge, Hans Vangheluwe