

Call Graphs for Live Programming

Implementing Call Tracing in Babylonian/S based on a Survey
of Property Extraction Techniques for Dynamic Analysis

Christian Flach

Software Architecture Group
Hasso Plattner Institute, University of Potsdam

—
HPI Research Symposium 2021

April 19, 2021

Live Programming with Squeak/Smalltalk

Microview

The screenshot displays the Squeak/Smalltalk environment with two browser windows and a visualization of a bouncing atom.

Basic Babylonian Browser: MyClass

- 60Deprecated-Tools-In
- 60Deprecated-Tools-M
- Babylonian-Compiler
- Babylonian-Core
- Babylonian-Core-Annot
- Babylonian-Core-Callgr
- Babylonian-Core-Callgr
- Babylonian-Core-Callgr
- Babylonian-Core-Tracir
- Babylonian-Demo-Morph
- Babylonian-Demo-Tree
- BPSceneObject
- BPMountain
- BPSky
- BPTree
- BPTreeSceneMorph
- MyClass

instance class ?

myExample

add example | add script example

BouncingAtomsMorph new

ified · 1 implementor

Basic Babylonian Browser: AtomMorph

- MorphicExtras-AdditionalIM
- MorphicExtras-AdditionalS
- MorphicExtras-AdditionalW
- MorphicExtras-Books
- MorphicExtras-Demo
- MorphicExtras-EToy-Downl
- MorphicExtras-Exceptions
- MorphicExtras-Flaps
- MorphicExtras-GeeMail
- AbstractMediaEventM
- ZASMCameraMarkMc
- AtomMorph
- BouncingAtomsMorph
- ClockMorph
- Flasher
- FrameRateMorph

instance class ?

randomPositionIn: aRectangle maxVelocity: maxVelocity

add example | add script example

"Give this atom a random position and velocity."

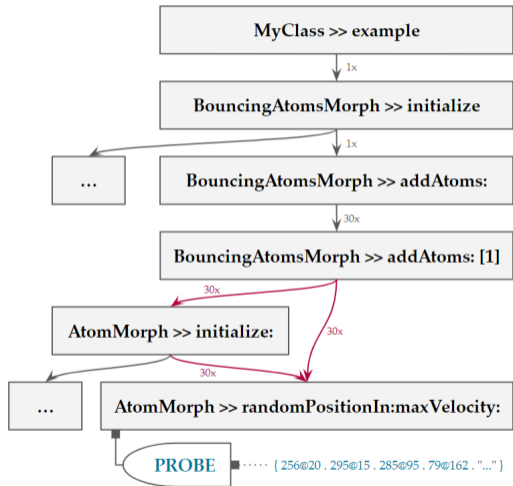
| origin extent |
origin := aRectangle origin.
extent := (aRectangle extent - self bounds extent) rounded.
self position:
 (origin x + extent x atRandom) @
 Q || bounc... 256@203, 295@159, 285@95, 79@162, 288@233, 258@139, X
 set expression
 (origin y + extent y atRandom).
velocity :=
 (maxVelocity - (2 * maxVelocity) atRandom) @
 (maxVelocity - (2 * maxVelocity) atRandom).

cmfcmf 3/16/2021 18:35 · initialization · 1 implementor · only in change set HomeProject ·

The visualization shows a green rectangular area containing several small blue squares, representing the positions of atoms in a simulation.

Selection of Possible Use Cases

Macroview



Possible Use Cases

As part of an example, ...

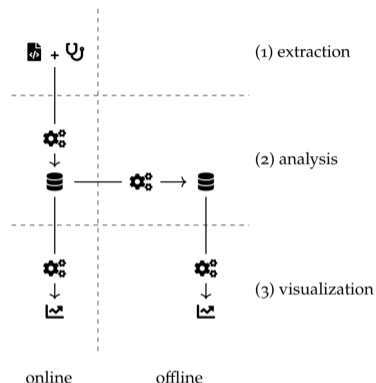
- given two or more procedure/probe executions, how do they relate to each other?
- given a procedure execution, what other procedures/probes are called from it?
- which procedures/probes are executed?

Dynamic Analysis

“the analysis of the **properties** of a **running** program”¹
 (emphasis mine)

Process²³

1. Property Extraction
2. Property Analysis
3. Property Visualization



¹Thomas Ball. “The Concept of Dynamic Analysis”. In: *Software Engineering - ESEC/FSE'99, 7th European Software Engineering Conference, Held Jointly with the 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Toulouse, France, September 1999, Proceedings*. Ed. by Oscar Nierstrasz and Michel Lemoine. Vol. 1687. Lecture Notes in Computer Science. Springer, 1999, pp. 216–234. DOI: 10.1007/3-540-48166-4_14.

²Mireille Ducassé and Jacques Noyé. “Logic Programming Environments: Dynamic Program Analysis and Debugging”. In: *The Journal of Logic Programming* 19/20 (1994), pp. 351–384. DOI: 10.1016/0743-1066(94)90030-2.

³Margaret-Anne D. Storey. “Theories, Tools and Research Methods in Program Comprehension: Past, Present and Future”. In: *Software Quality Journal* 14.3 (2006), pp. 187–208. DOI: 10.1007/s11219-006-9216-4.

Approach

- Exploratory search for “dynamic analysis” and related topics (“debugger”, “malware analysis”, “program comprehension”, etc.), including
- Well-known publications for Squeak/Smalltalk
- *Workshop on Dynamic Analysis* (WODA, 2003 to 2017)

Corpus

- ~40 surveys/survey-like publications, ~20 tool frameworks, ~60 tools, ~90 other publications, ~20 Smalltalk papers
- 13 publications that explicitly list more than two Property Extraction techniques

1. Program Level

- 1.1 Changing Input, Observing Output
- 1.2 Source Code Rewriting
- 1.3 Bytecode/Machine Code Rewriting

2. Runtime/Interpreter Level

- 2.1 Analysis Interfaces Provided by the Runtime/Interpreter
- 2.2 Runtime/Interpreter Modification

3. Operating System Level

4. System/Processor Software-based

- 4.1 Processor Tracing
- 4.2 Hardware Performance Counters
- 4.3 Hardware Sensors

5. Virtualization and Emulation

- 5.1 System Virtual Machine
- 5.2 Processor Emulation
- 5.3 System Emulation

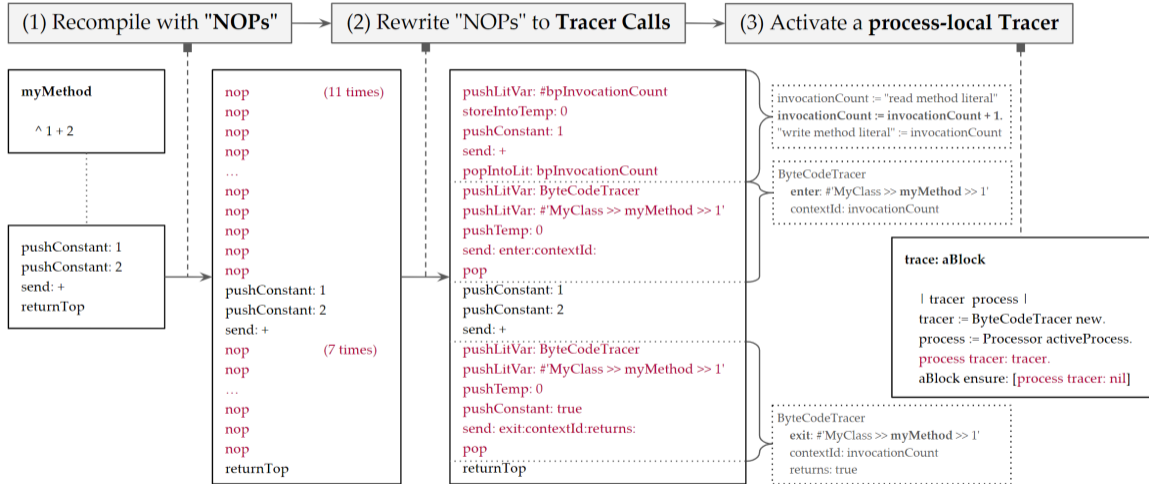
6. System/Processor Hardware-based

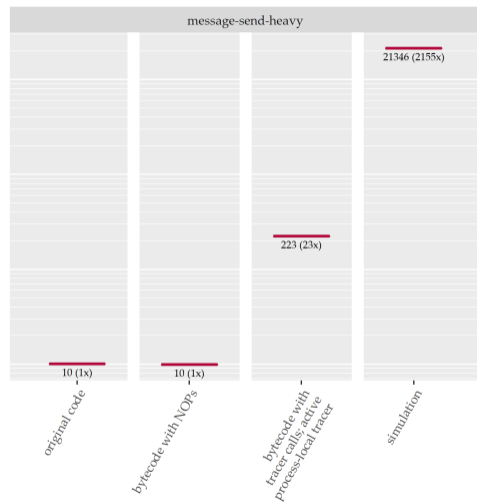
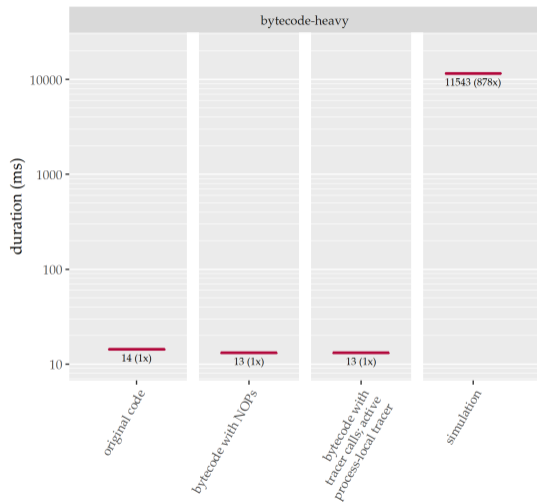
- 6.1 Hardware Processor Debugging
- 6.2 System/Processor Hardware Instrumentation

7. System External

Call Tracing Implementation

Bytecode Rewriting — Process





(lower is better)

Squeak/Smalltalk (Babylonian/S) Integration

Macroview

```

Ex for MyClass>>#myExample>>#'#bouncing'
  BouncingAtomsMorph #initialize          ctb: 49
    BouncingAtomsMorph #defaultColor     ctb: 49
      BouncingAtomsMorph #defaultColor   ctb: 49
    BouncingAtomsMorph #invalidRect:from  ctb: 173
      BouncingAtomsMorph #invalidRect:from ctb: 173
    BouncingAtomsMorph #invalidRect:from  ctb: 173
      BouncingAtomsMorph #invalidRect:from ctb: 173
    BouncingAtomsMorph #addAtoms:         ctb: 49
      BouncingAtomsMorph #addAtoms: [1]   ctb: 49
        AtomMorph #initialize             ctb: 144
          AtomMorph #defaultColor         ctb: 144
            AtomMorph #defaultColor       ctb: 144
          AtomMorph #randomPositionIn:maxVelocity:
            AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 121
              AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 121
                Probe hit                    256@20
              ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 121
            ^ AtomMorph #initialize         ctb: 144
          ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 122
            AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 122
              AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 122
                Probe hit                    295@15
              ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 122
            BouncingAtomsMorph #addMorphFront: ctb: 144
              BouncingAtomsMorph #addMorphFront: ctb: 144
            BouncingAtomsMorph #addAtoms: [1] ctb: 49
          BouncingAtomsMorph #addAtoms: [1] ctb: 49
            AtomMorph #initialize           ctb: 144
              AtomMorph #defaultColor       ctb: 144
                AtomMorph #defaultColor     ctb: 144
              AtomMorph #randomPositionIn:maxVelocity: ctb: 123
                AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 123
                  AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 123
                    Probe hit                285@95
                  ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 123
                ^ AtomMorph #initialize       ctb: 144
              ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 124
                AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 124
                  AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 124
                    Probe hit                79@162
                  ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 124
                BouncingAtomsMorph #addMorphFront: ctb: 144
                  BouncingAtomsMorph #addMorphFront: ctb: 144
                ^ AtomMorph #initialize       ctb: 144
          ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 124
        ^ AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 124
          AtomMorph #randomPositionIn:maxVelocity: [1] ctb: 124
            Probe hit                        79@162
          ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 124
        BouncingAtomsMorph #addMorphFront: ctb: 144
          BouncingAtomsMorph #addMorphFront: ctb: 144
        ^ AtomMorph #initialize               ctb: 144
      ^ AtomMorph #randomPositionIn:maxVelocity: ctb: 124
    ^ BouncingAtomsMorph #addAtoms: [1]     ctb: 144
  ^ BouncingAtomsMorph #initialize           ctb: 144

```

```

callgraph
annotation set
procedure set
Ex for MyClass>>#'#bouncing>>#'#'bouncing'
P in AtomMorph #randomPositionIn:maxVelocity: (orig
Common ancestor (context-insensitive)
BouncingAtomsMorph #addAtoms: [1]
Unique Invocation Path 1 (context-insensitive)
BouncingAtomsMorph #initialize
BouncingAtomsMorph #addAtoms:
BouncingAtomsMorph #addAtoms: [1]
AtomMorph #initialize
AtomMorph #randomPositionIn:maxVelocity:
P in AtomMorph #randomPositionIn:maxVelocity: (orig
Unique Invocation Path 2 (context-insensitive)
BouncingAtomsMorph #initialize
BouncingAtomsMorph #addAtoms:
BouncingAtomsMorph #addAtoms: [1]
AtomMorph #randomPositionIn:maxVelocity:
P in AtomMorph #randomPositionIn:maxVelocity: (orig
Common ancestor (context-sensitive)
BouncingAtomsMorph #addAtoms: [1]
Invocation 1 (context-sensitive)
BouncingAtomsMorph #initialize
BouncingAtomsMorph #addAtoms:
BouncingAtomsMorph #addAtoms: [1]
AtomMorph #initialize
AtomMorph #randomPositionIn:maxVelocity:
Probe hit
Invocation 2 (context-sensitive)
BouncingAtomsMorph #initialize

```

randomPositionIn: aRectangle maxVelocity: maxVelocity

add example | add script example

"Give this atom a random position and velocity."

| origin extent |

origin := aRectangle origin.

extent := (aRectangle extent - self bounds extent) rounded.

self position:

(origin x + extent x atRandom) @

x

Q || bouncing 157@150, 377@177, 53@121, 38

set expression

(origin y + extent y atRandom).

velocity :=

(maxVelocity - (2 * maxVelocity) atRandom) @

(maxVelocity - (2 * maxVelocity) atRandom).

Call Graphs for Live Programming

Implementing Call Tracing in Babylonian/S based on a Survey of Property Extraction Techniques for Dynamic Analysis

