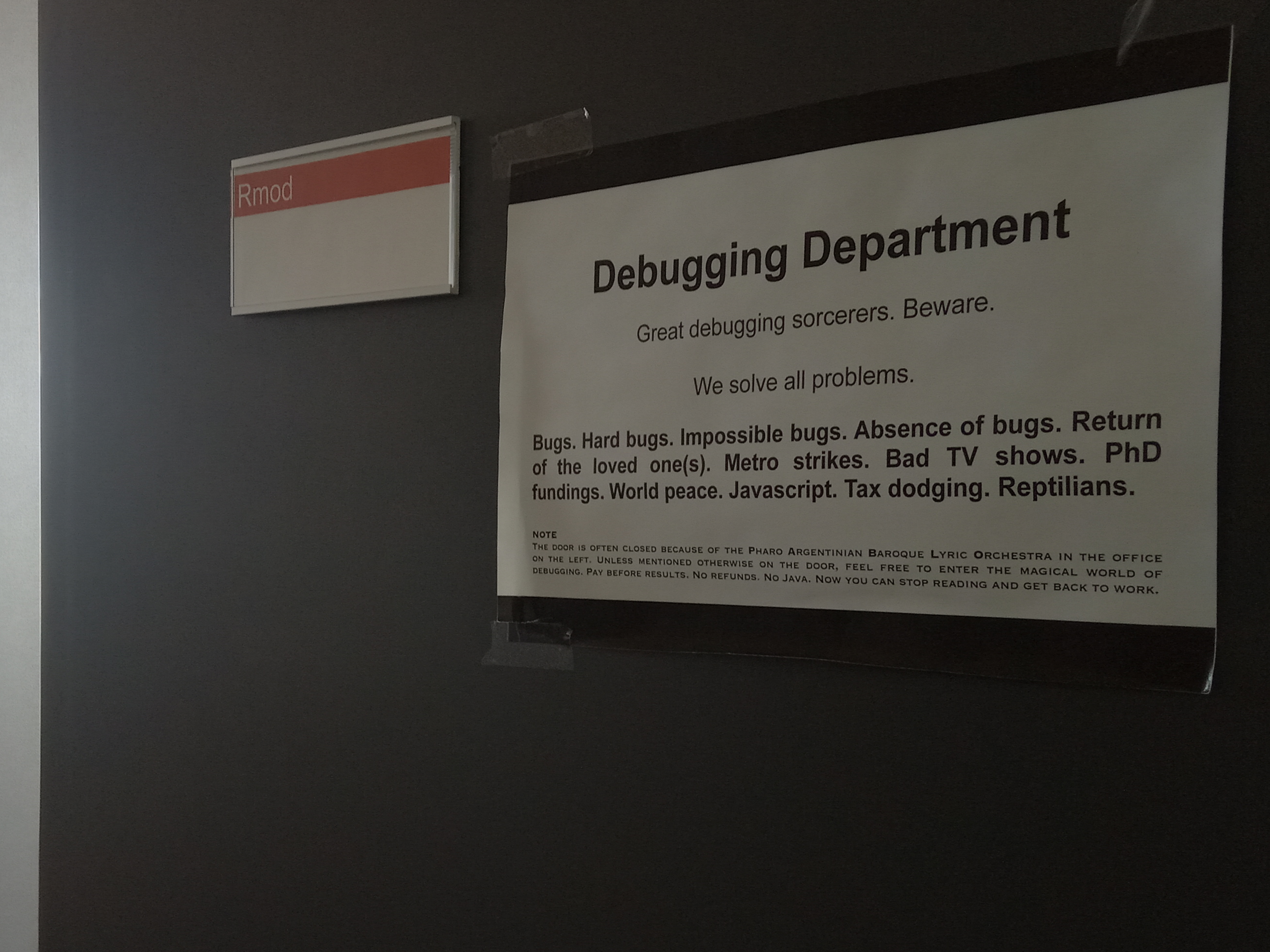
Object-Centric Debugging: a preview

Steven Costiou
RMoD
Inria Lille - Nord Europe

steven.costiou@inria.fr

2019



Rmod

Debugging Department

Great debugging sorcerers. Beware.

We solve all problems.

Bugs. Hard bugs. Impossible bugs. Absence of bugs. Return of the loved one(s). Metro strikes. Bad TV shows. PhD fundings. World peace. Javascript. Tax dodging. Reptilians.

NOTE
The door is often closed because of the Pharo Argentinian Baroque Lyric Orchestra in the office on the left. Unless mentioned otherwise on the door, feel free to enter the magical world of debugging. Pay before results. No refunds. No Java. Now you can stop reading and get back to work.

Part I
Object-Centric Debugging

Demo

What is object-centric debugging?

Object-centric debugging

- Debugging operations at the level of objects
  - Only target objects are affected
- Examples:
  - A breakpoint active for one object only
  - A method available for one object only

Why object-centric debugging?

Why Object-centric debugging?

- Debugging one object among many:
  - Collections (Hinkle, Jones, Johnson, 1993)
  - Events
  - Graphical objects

Object-Centric Features
(preview)

Object-centric breakpoints

- Break when a message is received
  - **haltOnCall** => on every method call
  - **haltOnCall: #selector** => for given selector only
  - **haltOnNextCall** => on next method call
  - **haltOnceOnCall: #selector** => only once for given selector
  - **haltOnCallWhen: condition** => if condition is met

Object-centric breakpoints

- Break on state access
  - **haltOnWriteTo: #instVarName** => when instVarName is written
  - **haltOnRead: #instVarName** => when instVarName is read
  - **haltOnWrite** => when any instance variable is written
  - **haltOnRead** => when any instance variable is read

Object-centric behavior

- Object-centric methods
  - **compile: sourceCode** => compiles and add new methods
  - **uses: aTrait** => acquires behavior from Trait
  - **acquire: aCompiledMethod** => acquire the method

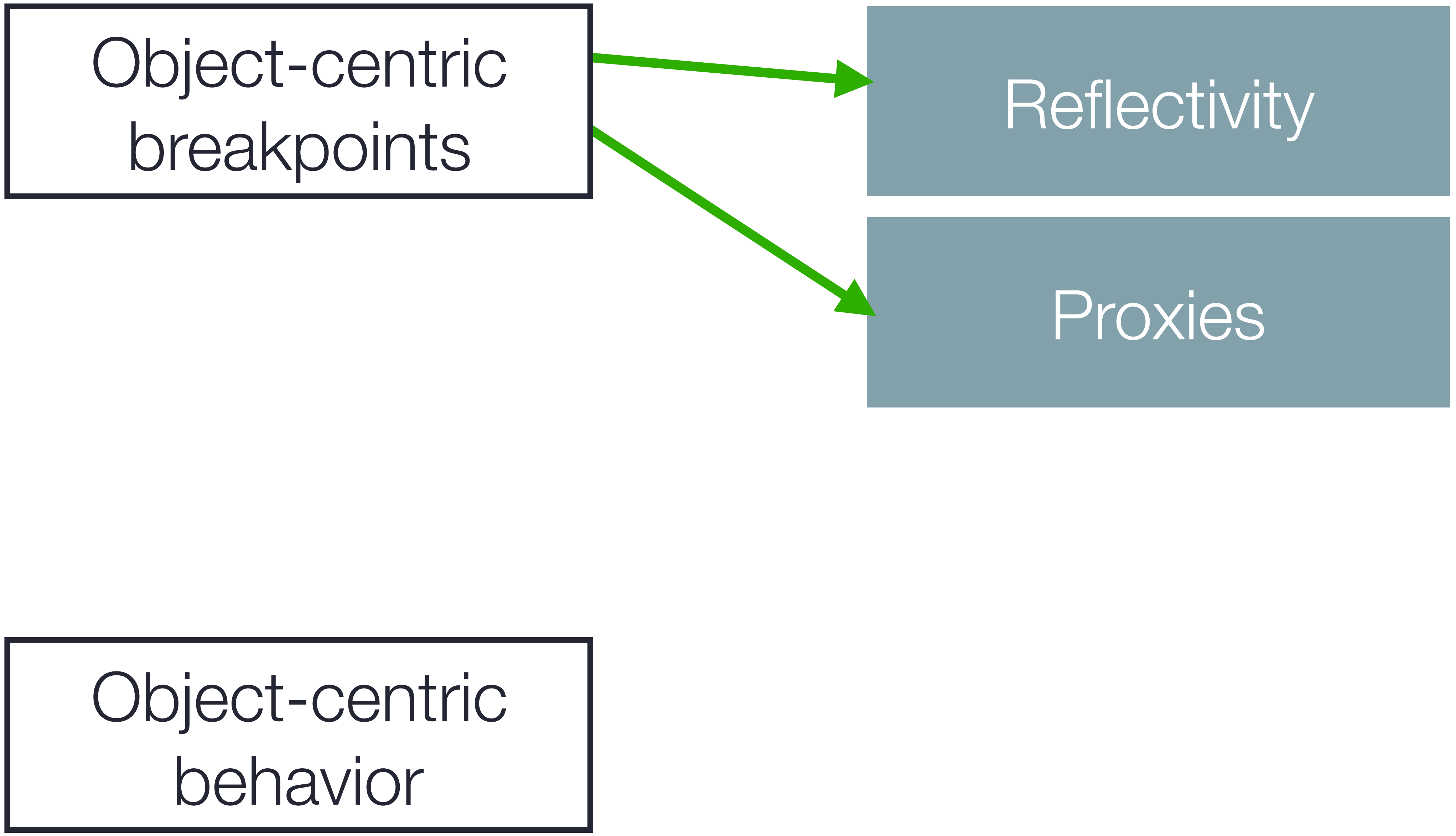
Object-centric debugging, how is it implemented?

Implementation

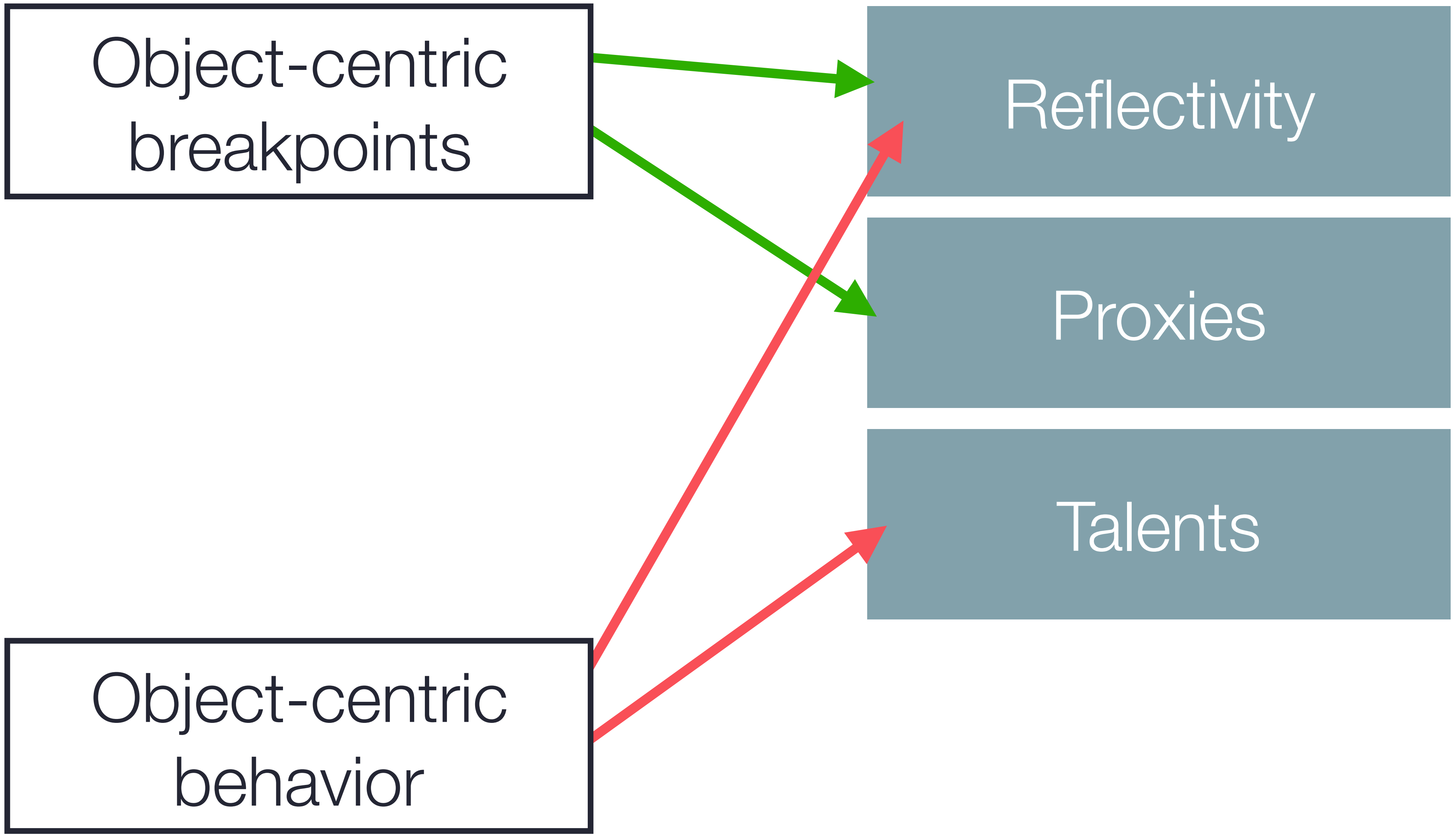
Object-centric breakpoints

Object-centric behavior

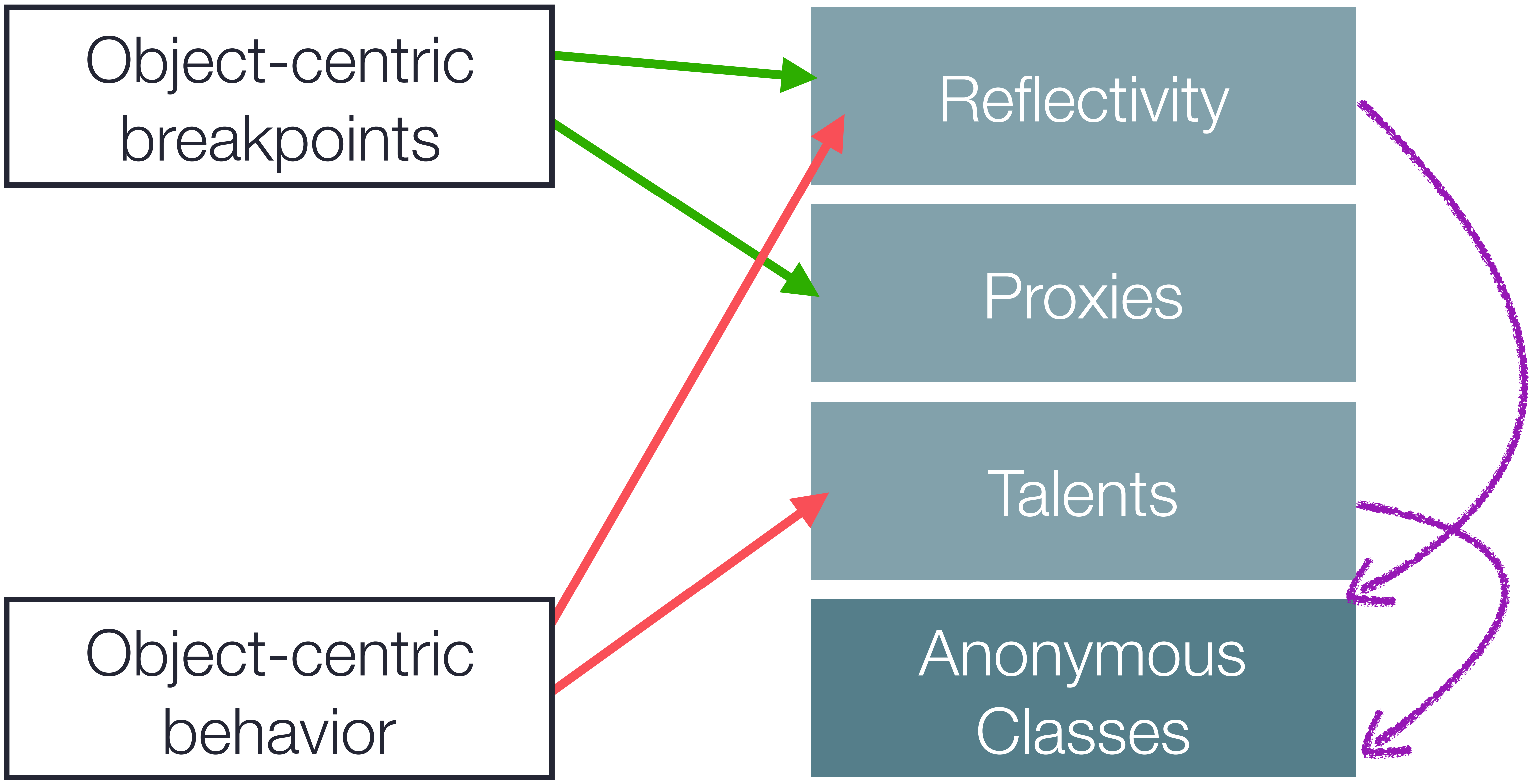
Implementation



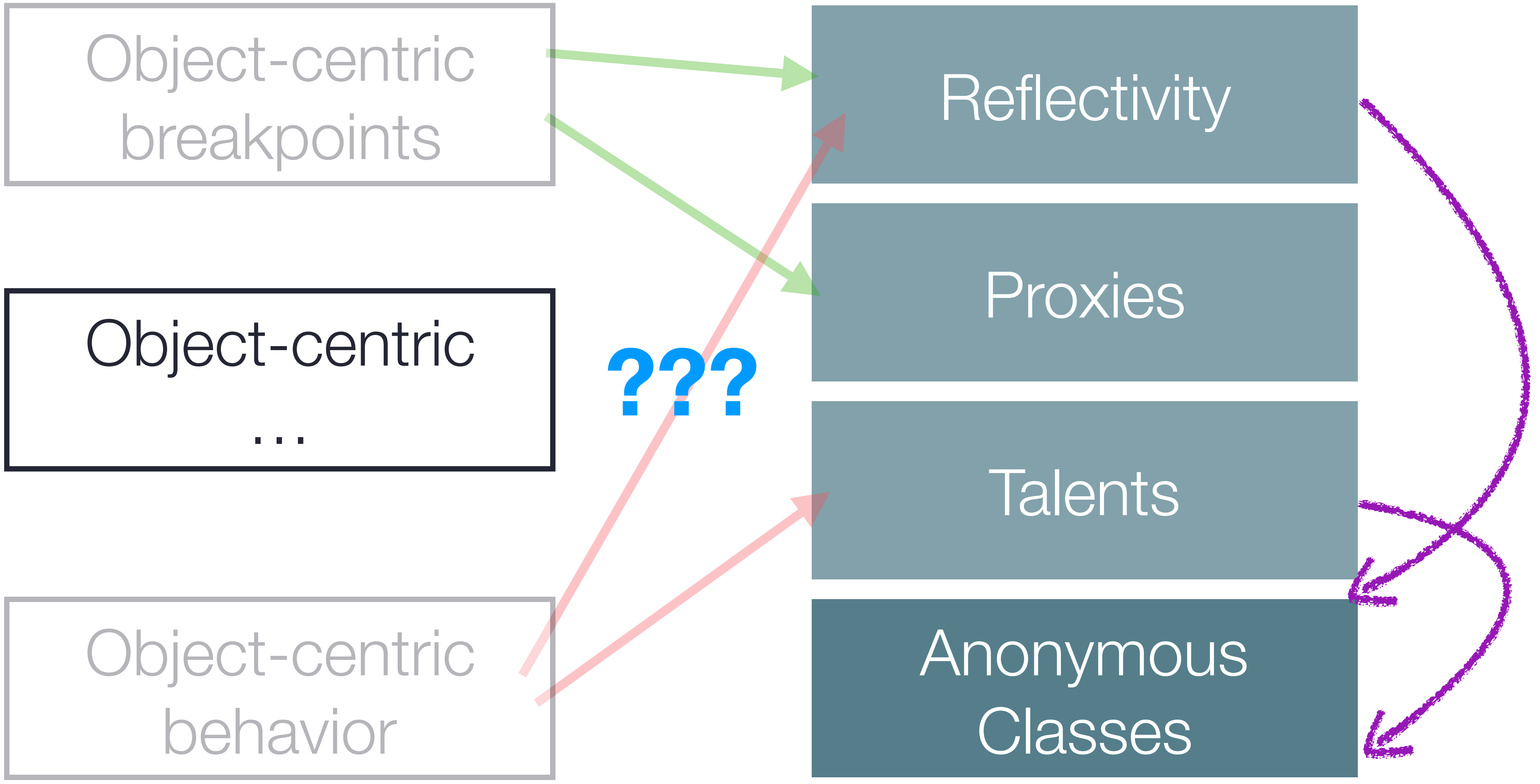
Implementation



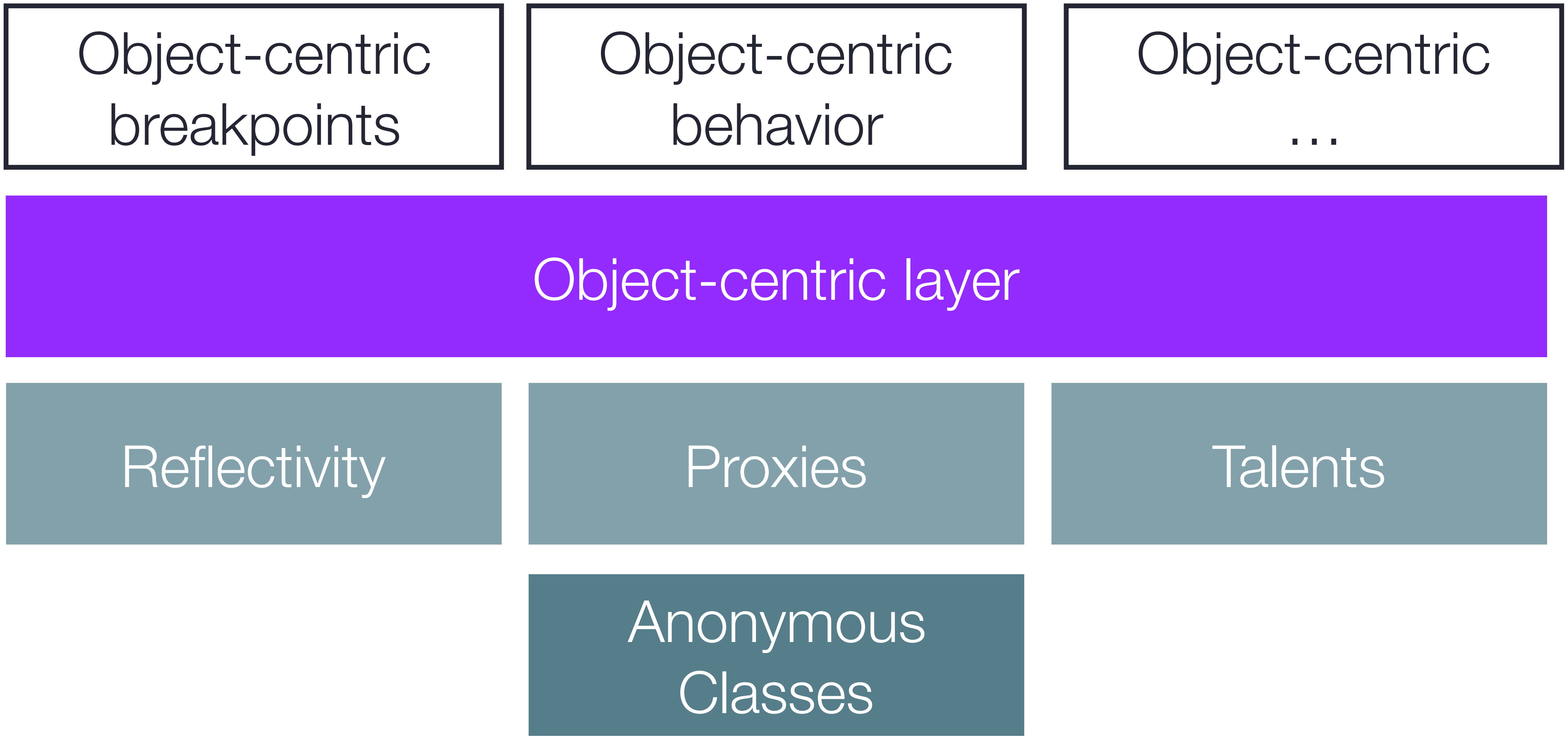
Implementation



Implementation



Implementation goal



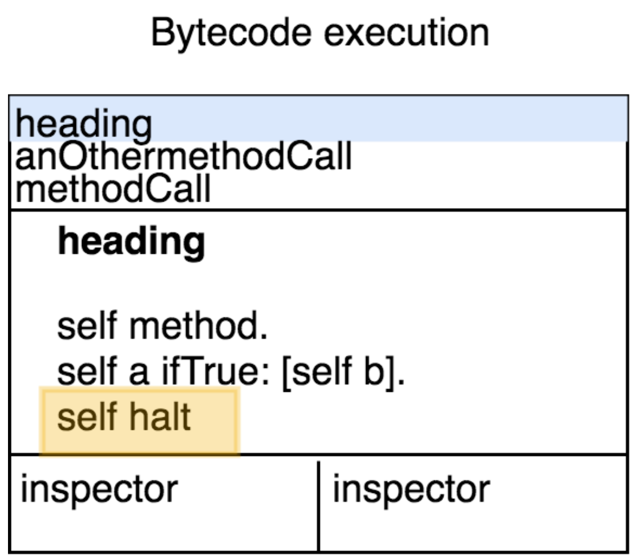
Current problems

- Implementation is mixing up different techniques without any clear interaction model
- Requires to migrate the object to an anonymous subclass
- Installation of object-centric instrumentation is not thread-safe
- Sometimes make tools unstable
- Obtaining objects to debug (but work has been done on that…)

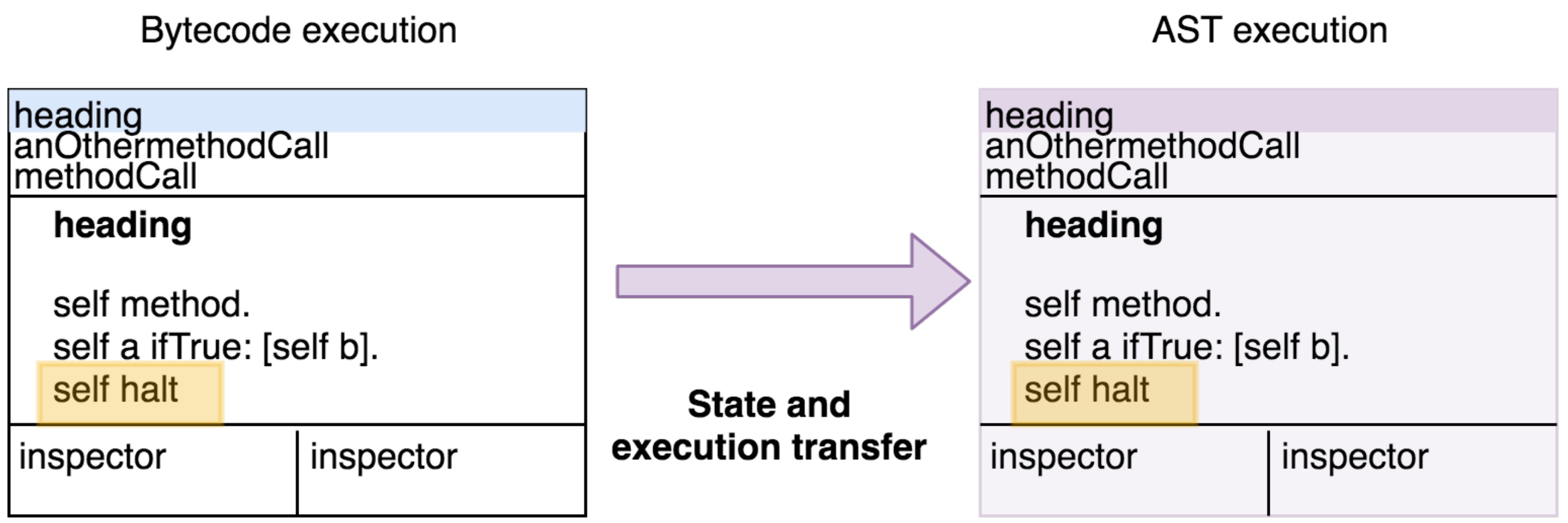
Part II
Object-Centric Reverse Debugging

Demo

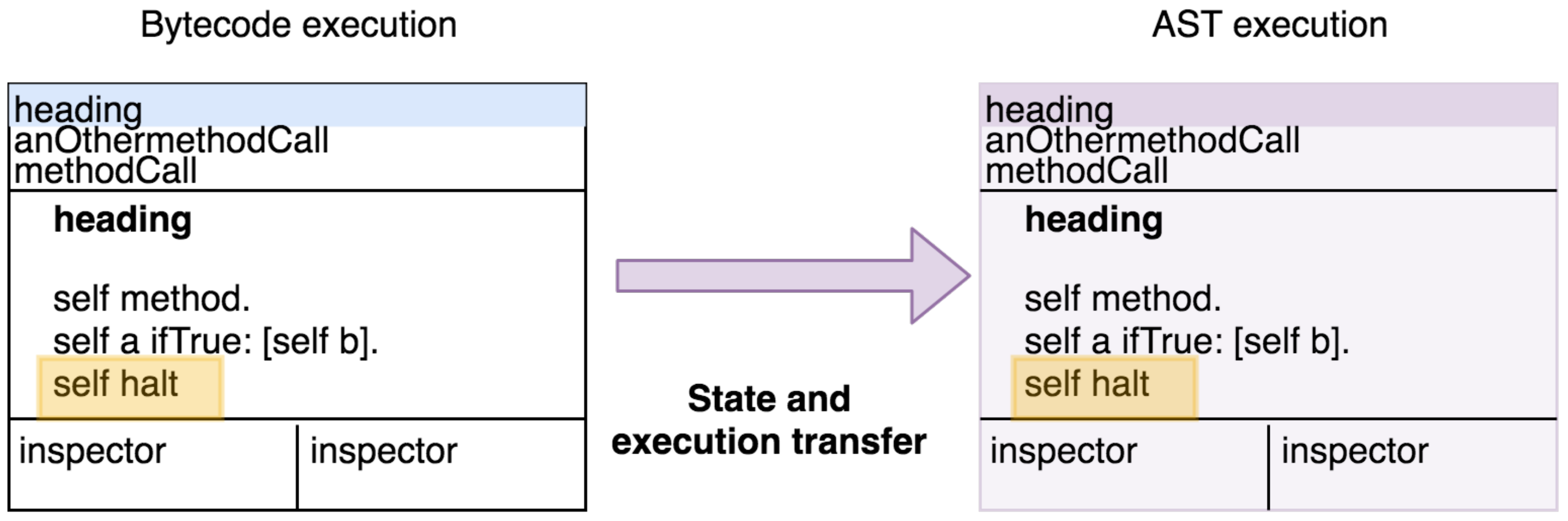
Reverse object-centric debugger



Reverse object-centric debugger

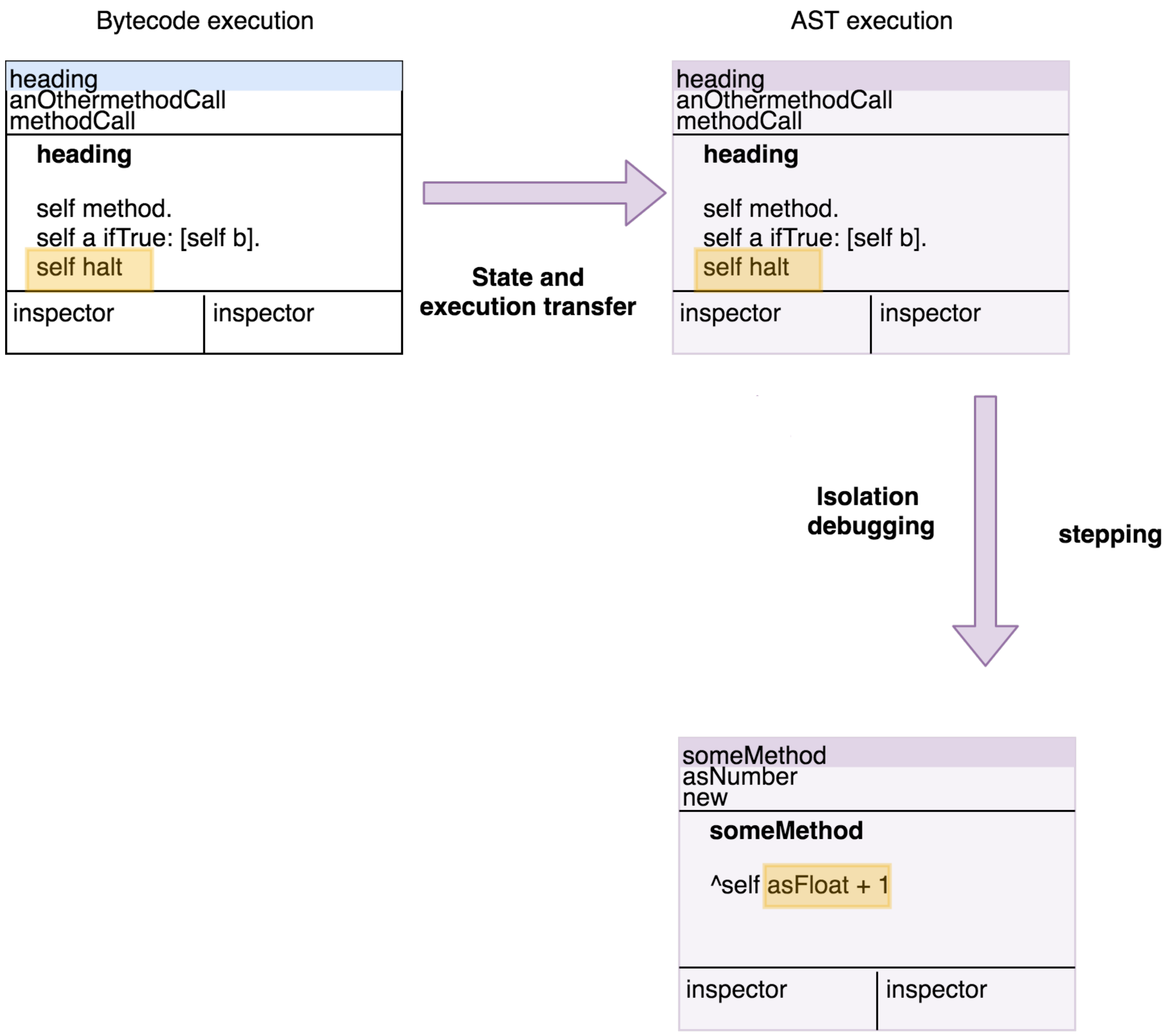


Reverse object-centric debugger

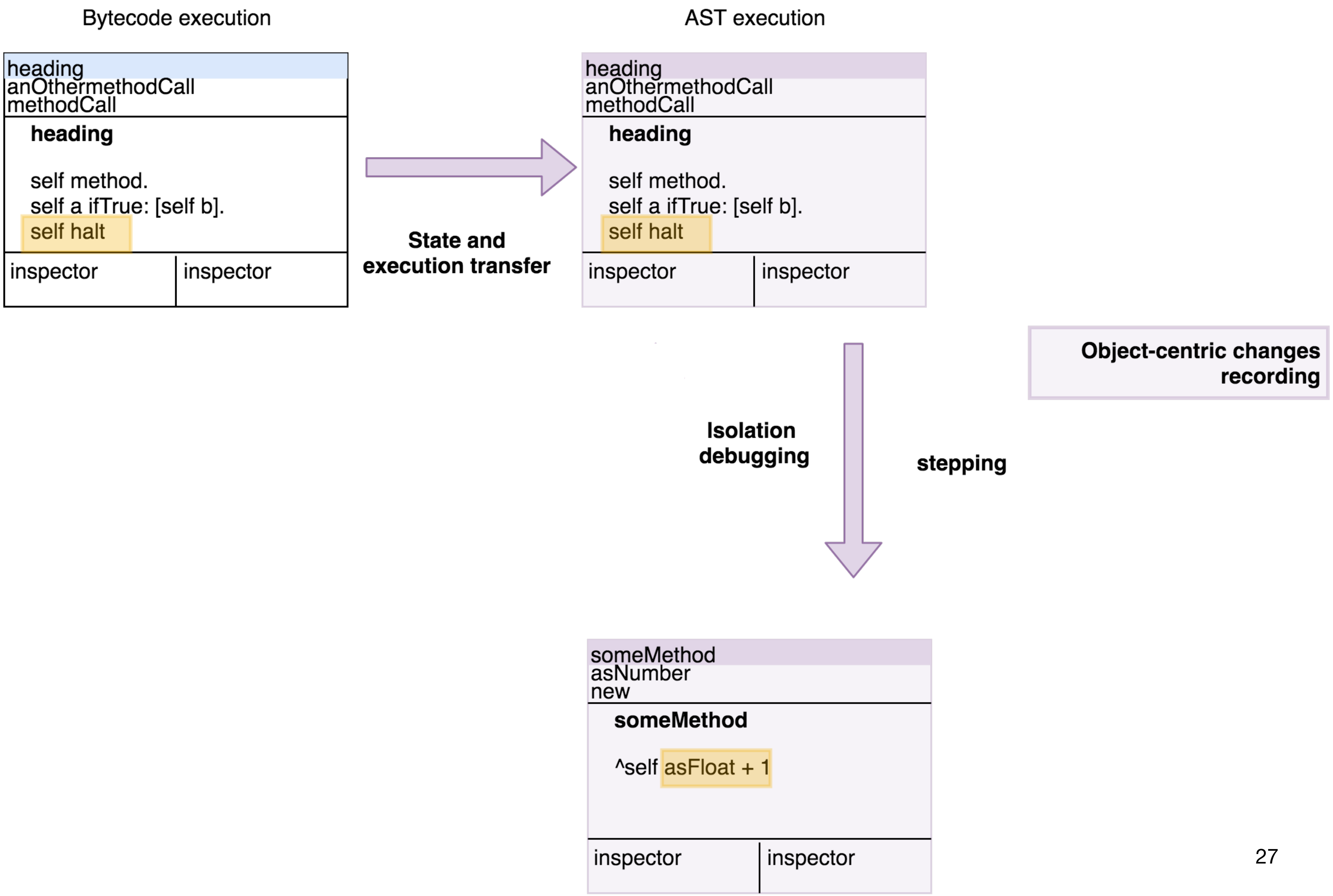


Isolation debugging

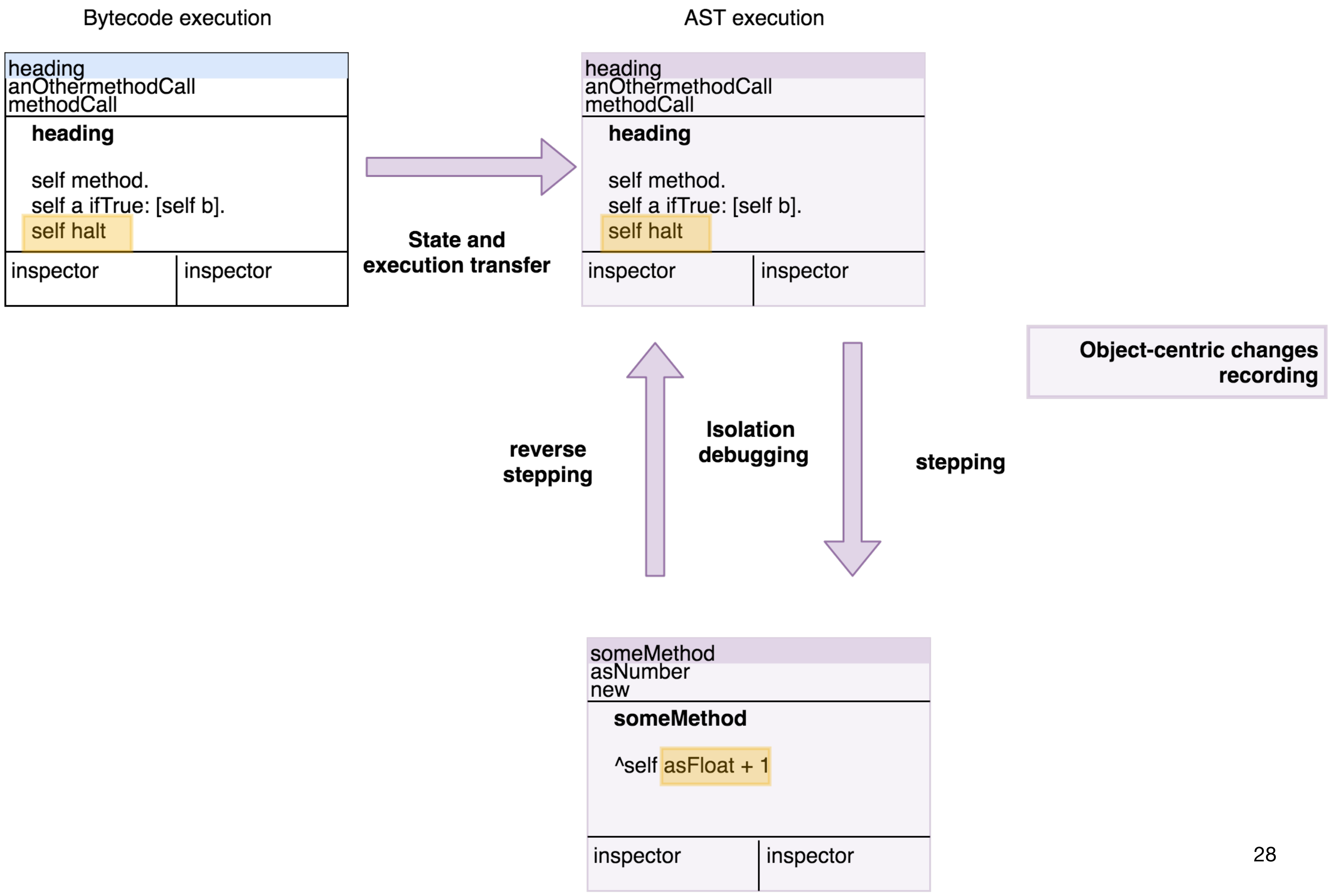
Reverse object-centric debugger



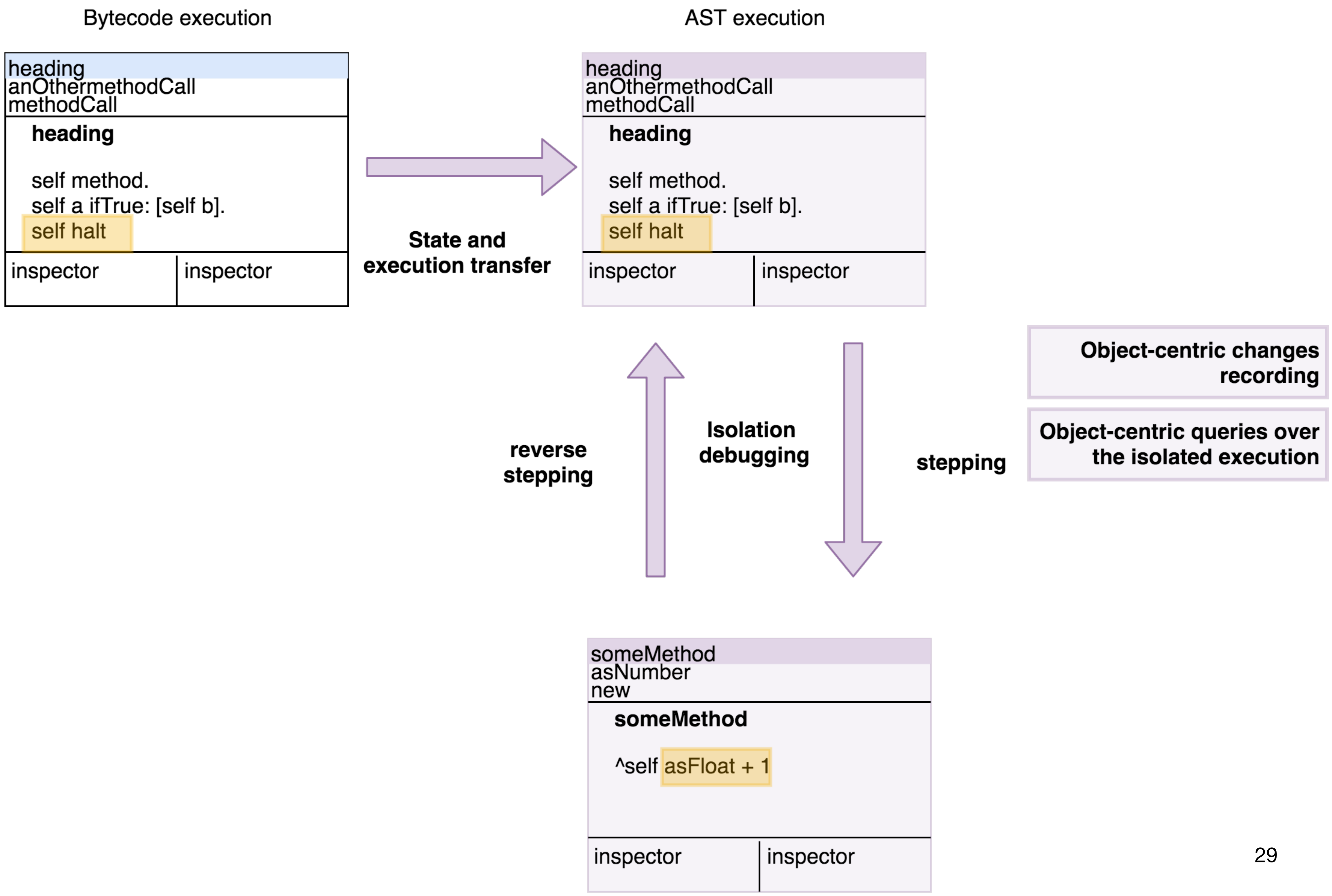
Reverse object-centric debugger



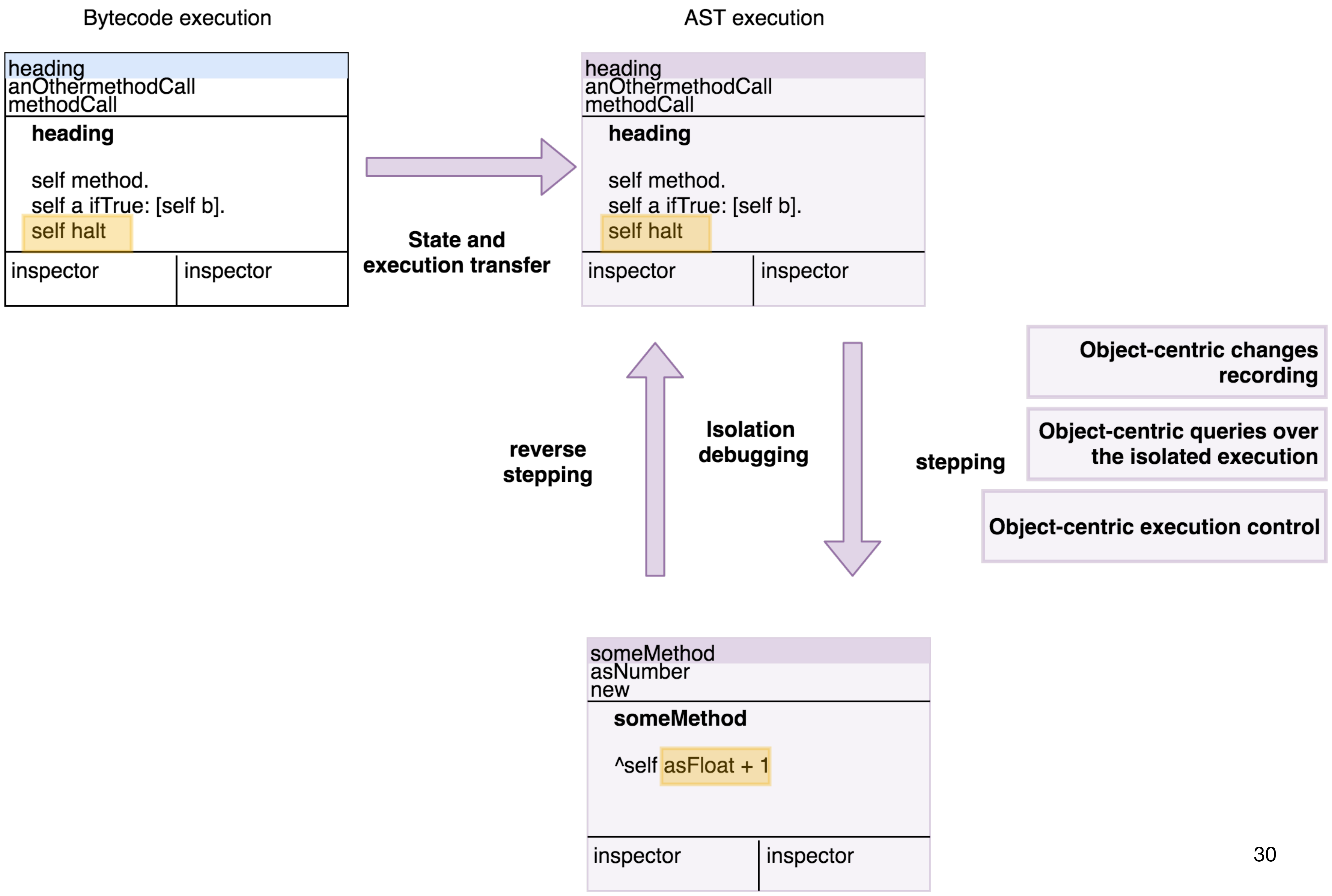
Reverse object-centric debugger



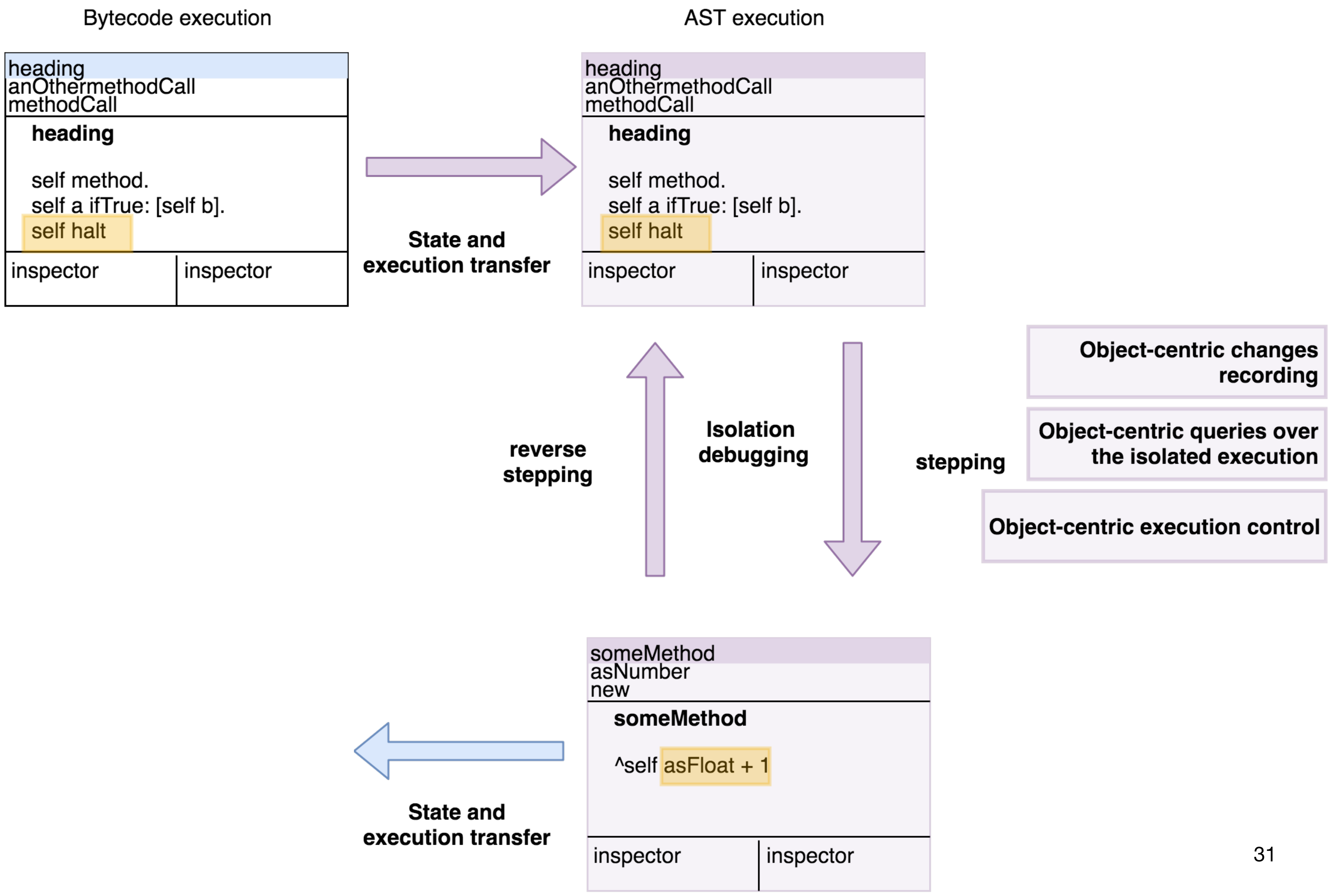
Reverse object-centric debugger



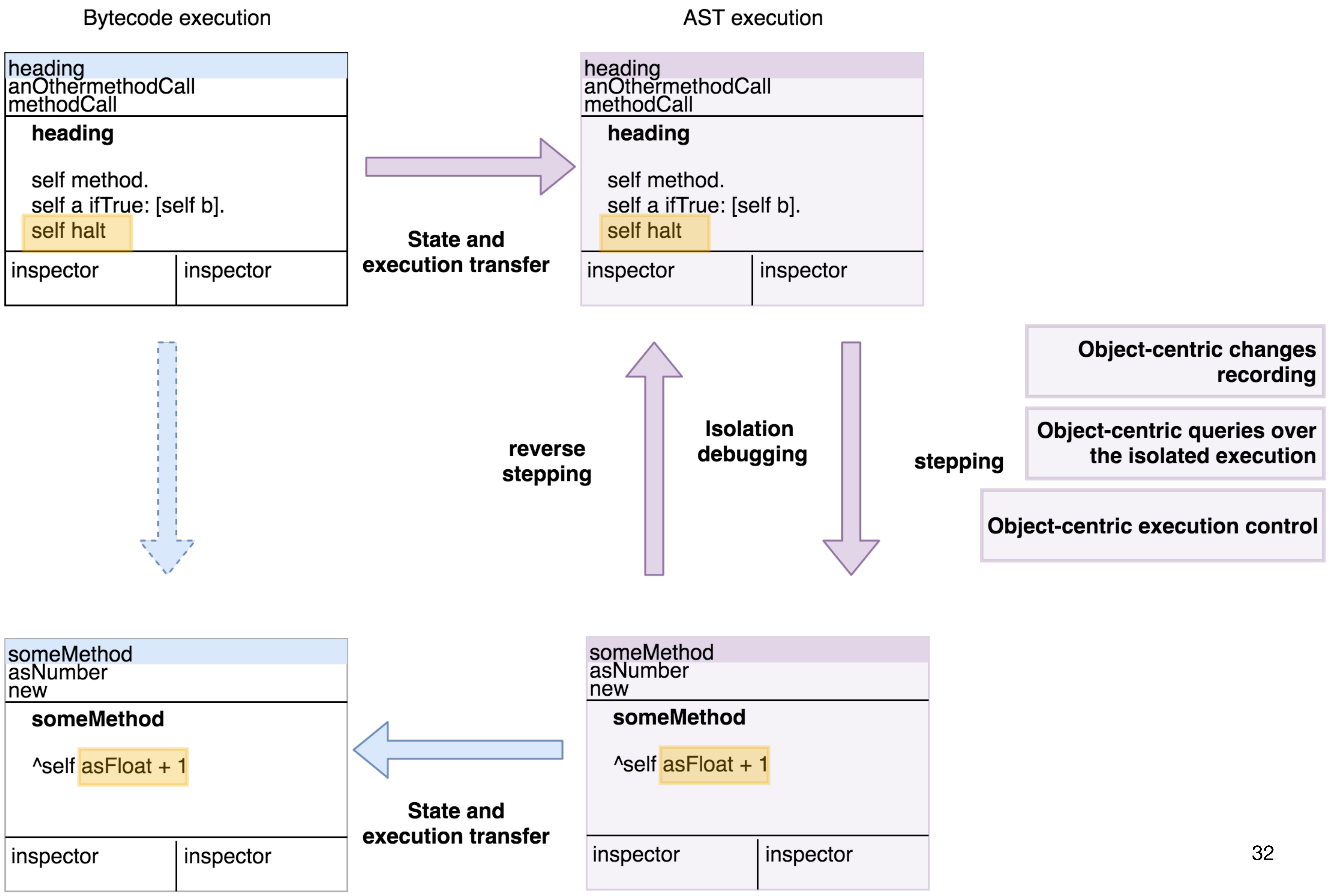
Reverse object-centric debugger



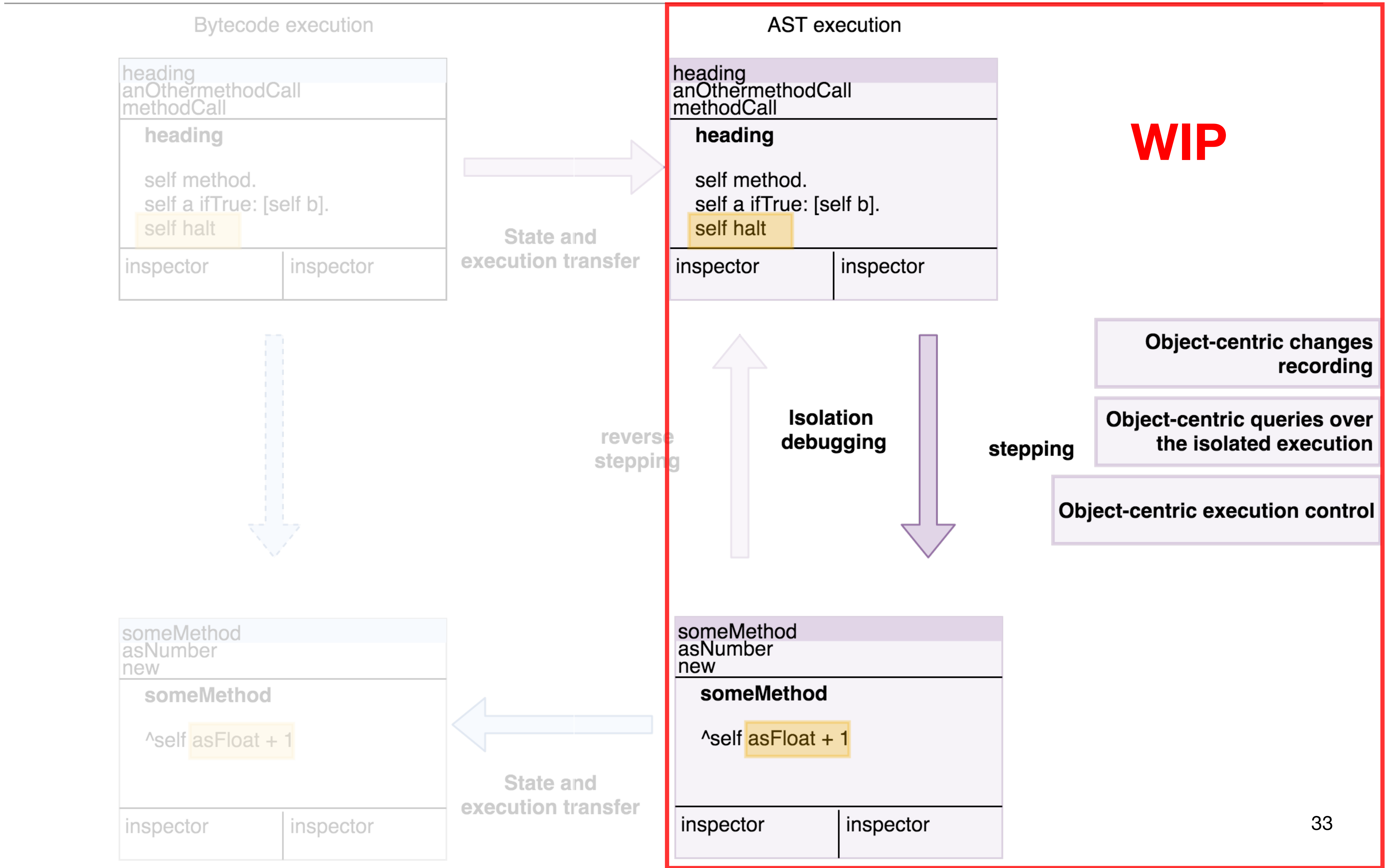
Reverse object-centric debugger



Reverse object-centric debugger



Reverse object-centric debugger



Reverse object-centric debugger

Who’s working on it?

- **AST interpreter** - Carolina Hernandez Phillips
- **Reverse-execution** - Vincent Aranega, Steven Costiou
- **Object-centric debugger** - Steven Costiou
- **Scriptable debugger** - Thomas Dupriez

Thanks! Questions?

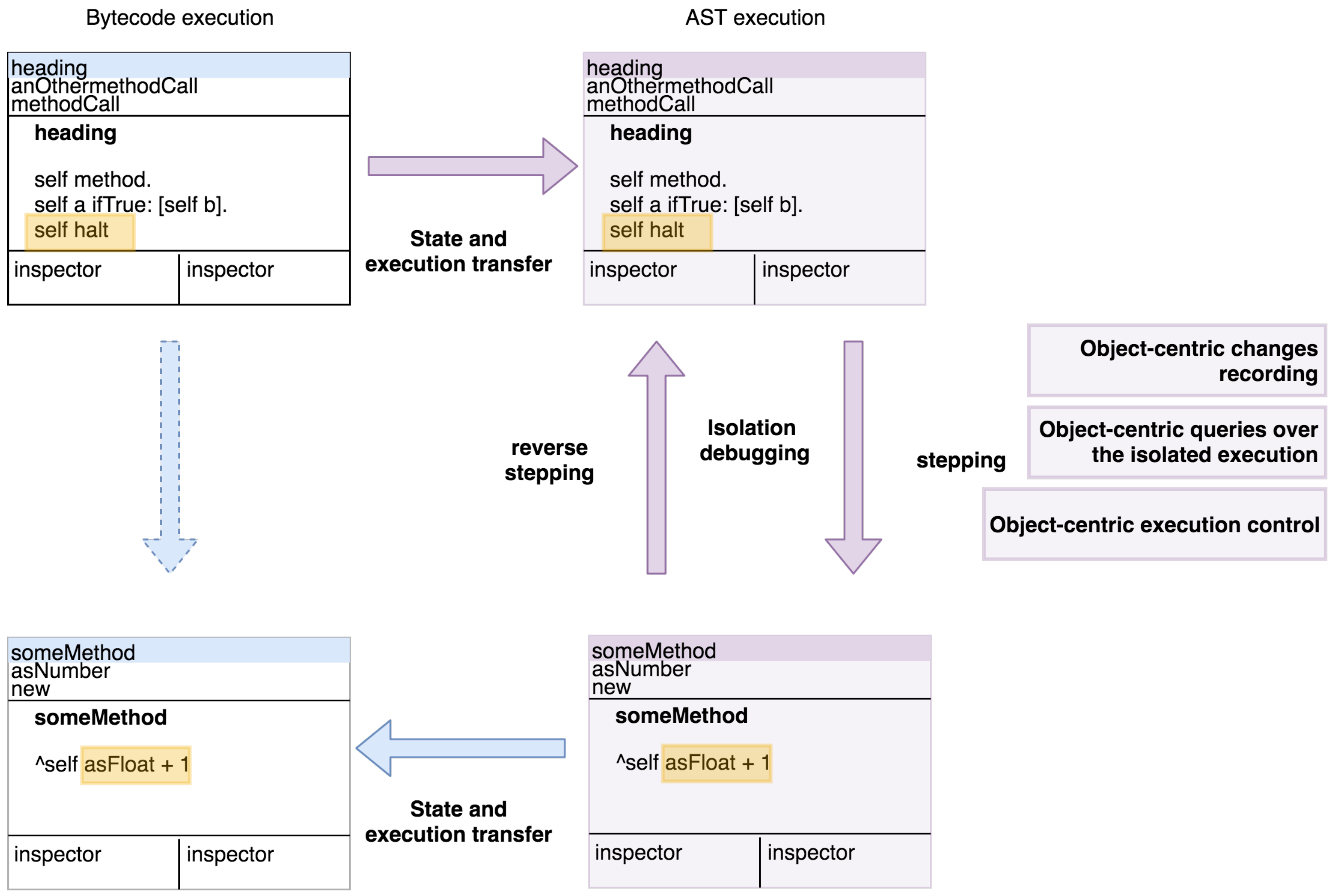
Object-Centric Debugging

- **haltOnCall**
- **haltOnCall:** #selector
- **haltOnNextCall**
- **haltOnceOnCall:** #selector
- **haltOnCallWhen:** condition

- **haltOnWriteTo:** #instVarName
- **haltOnRead:** #instVarName
- **haltOnWrite**
- **haltOnRead**

- **compile:** sourceCode
- **uses:** aTrait
- **acquire:** aCompiledMethod

Object-Centric Reverse Debugging


Bytecode execution
AST execution
heading
anOthermethodCall
methodCall
heading
self method.
self a ifTrue: [self b].
self halt
inspector
inspector
State and execution transfer
reverse stepping
Isolation debugging
stepping
Object-centric changes recording
Object-centric queries over the isolated execution
Object-centric execution control
someMethod
asNumber
new
someMethod
^self asFloat + 1
inspector
inspector
State and execution transfer