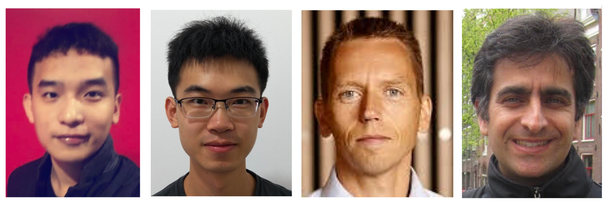
Precision-Guided Context Sensitivity for Pointer Analysis

Yue Li, Tian Tan, Anders Møller, Yannis Smaragdakis

{*yueli, tiantan, amoeller*}*@cs.au.dk* *smaragd@di.uoa.gr*



Problem

Context sensitivity (C.S.) produces high precision but comes with heavy efficiency costs

- Conventional pointer analyses apply C.S. to all methods, including the ones that do not benefit from C.S.
- Applying C.S. to only precision-critical methods and C.I. to other methods, can improve the efficiency while preserving the precision of pointer analysis.
- How to identify the precision-critical methods?

Challenge

It is still unclear where and how imprecision is introduced in a context-insensitive pointer analysis:

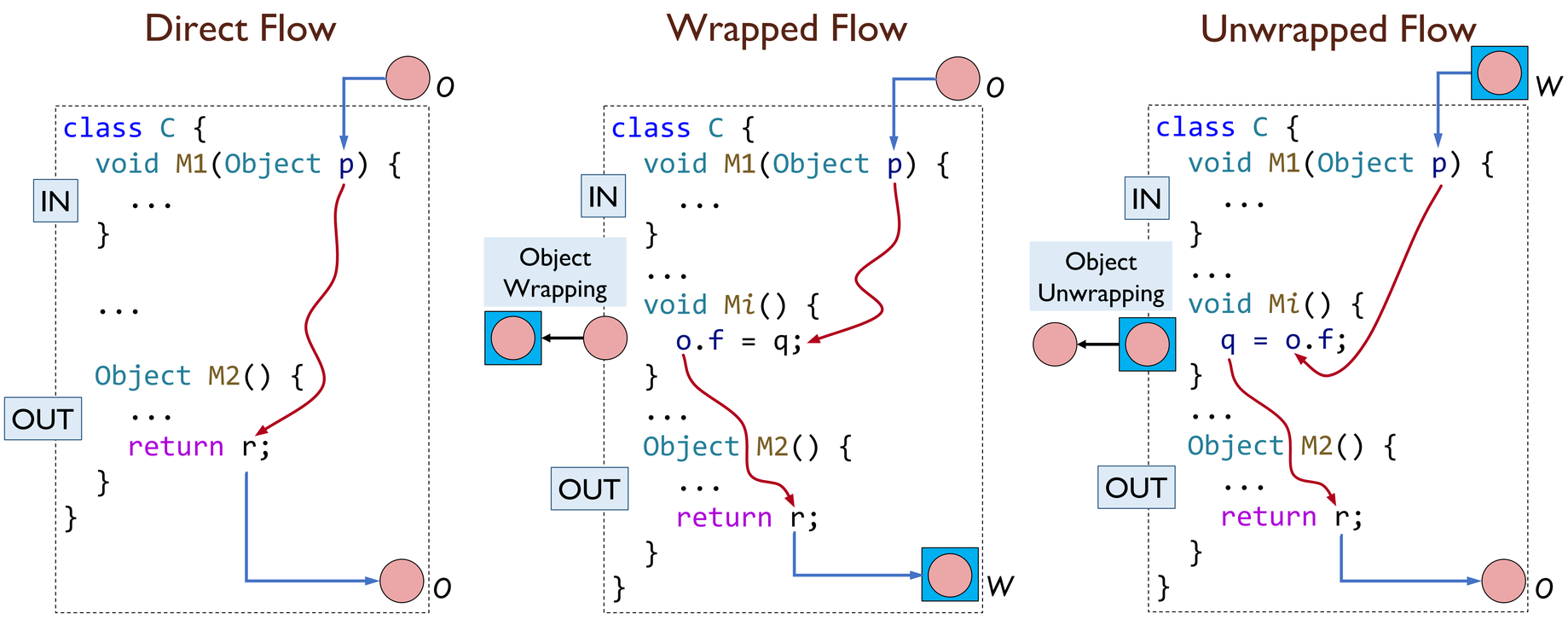
- When a context-sensitive analysis will yield precision benefits, or when omitting context sensitivity for a method would introduce imprecision?

Solution

Three fundamental precision-loss patterns + Zipper

Insight: Precision-Loss Patterns

Most cases of imprecision arise in a context-insensitive pointer analysis fit into:



- Red arrow represents the value flow via *assignments*, field *load*, field *store* operations, method *calls* or *returns*
- A wrapped/unwrapped flow can involve multiple object wrappings/unwrappings
- The three fundamental flows can be combined

Example: *Setter* & *Getter*

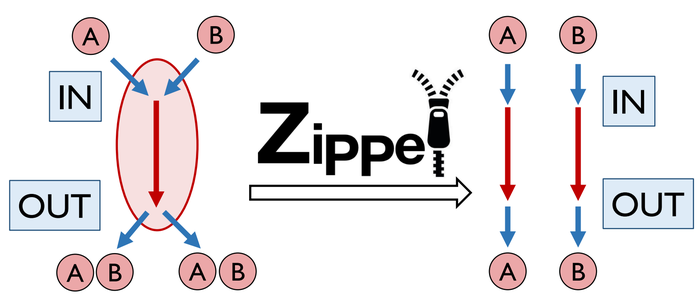
Example: *Collection* & *Iterator*

Example: *JDK Synchronized Container*

Zipper

Methodology

- Identify precision-critical methods
- Apply context sensitivity only to



Precision-critical methods are the methods that are involved in the flows

Technique

We propose precision flow graph (PFG)

- expresses *direct*, *wrapped* and *unwrapped* flows, and their *combinations*, in an uniform way

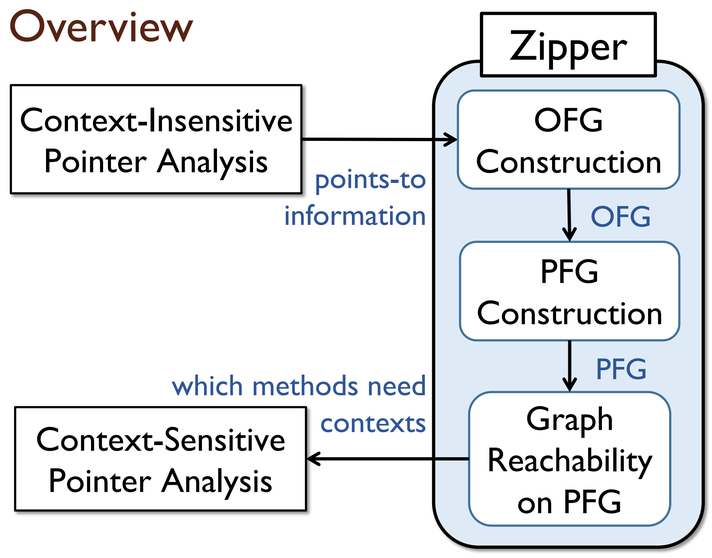
Flows in Program ⟺ Paths in PFG

Methods Involved in the Flows ⟺ Simple Graph Reachability on PFG

i.e., precision-critical methods

from IN to OUT methods

Overview



OFG: object flow graph, per program

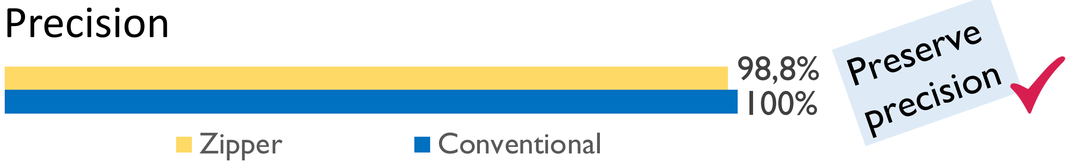
PFG: precision flow graph, per class

Results

Zipper vs. Conventional (2-object-sensitivity)

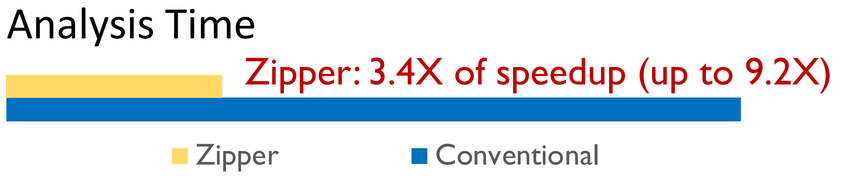
Methods Analyzed with Context Sensitivity

Precision



Preserve precision ✓

Analysis Time



Improve efficiency ✓

Implementation & Artifact

- Written in Java (core: 1500 LOC)
- Can be easily integrated with any pointer analysis tools
- Has been Integrated with DOOP
- Released as an open-source tool at: http://www.brics.dk/zipper
- **Artifact** successfully evaluated