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

Direct Flow

Wrapped Flow

Unwrapped Flow

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

Overview
Zipper

Context-Insensitive
Pointer Analysis

OGF Construction

OFG: object flow graph, per program

OFG: object flow graph, per class

Graph Reachability on OFG

Context-Sensitive
Pointer Analysis

PFG Construction

PFG: precision flow graph, per class

Reachability on PFG

Flows in Program

Paths in PFG

Simple Graph Reachability

${\text{Methods Involved in the Flows}}$
i.e., precision-critical methods

${\text{from IN to OUT methods}}$

Zipper

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

Example: Setter & Getter

Example: Collection & Iterator

Example: JDK Synchronized Container

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

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

Results

Zipper vs. Conventional (2-object-sensitivity)

Methods Analyzed with Context Sensitivity

<table>
<thead>
<tr>
<th></th>
<th>Precision-Critical Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zipper</td>
<td>38%</td>
</tr>
<tr>
<td>Conventional</td>
<td>100%</td>
</tr>
</tbody>
</table>

Precision

<table>
<thead>
<tr>
<th></th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zipper</td>
<td>98.8%</td>
</tr>
<tr>
<td>Conventional</td>
<td>100%</td>
</tr>
</tbody>
</table>

Analysis Time

<table>
<thead>
<tr>
<th></th>
<th>3.4X of speedup (up to 9.2X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zipper</td>
<td></td>
</tr>
<tr>
<td>Conventional</td>
<td></td>
</tr>
</tbody>
</table>