Merging Equivalent Contexts for Scalable Heap-Cloning-Based Context-Sensitive Points-to Analysis

Work of Guoqing Xu and Atanas Rountev

Hakjoo Oh

2009.01.17
Background

- BDD
  - provides an effective representation for context-sensitive points-to relationships
  - automatically merges equivalent contexts
Motivation

• BDD
  • is slower than non-BDD analysis (2x)
  • is not effective as much with heap-cloning
Goal

• develop a technique for
  • merging equivalent contexts explicitly
• application to points-to analysis with heap-cloning
Equivalent Contexts

- a coarse-grained (procedure-level) def.
- for a method $m$

$c_0$ and $c_1$ are equivalent iff

$$\forall p \in \text{Ptr}_m. \ pt(p, c_0) = pt(p, c_1)$$
Equivalent Contexts
(for a method m)

set of all contexts

\[(c_0, c_1) \in R_{eq}^m \text{ iff } \forall p \in Ptr_m. pt(p, c_0) = pt(p, c_1)\]
Equivalent Contexts
(for a pointer $p$ in $m$)

set of all contexts (for a pointer $p$ in $m$)

$(c_0, c_1) \in R_{eq}^{p,m}$ iff $pt(p, c_0) = pt(p, c_1)$
Points-to Relationships with Heap-Cloning

• (p,cp,o,co), a fully context-sensitive rep.
  • cp: from main to the p-defining method
  • co: from main to the o-creating method
Inlining-based Analysis

- By bottom-up, inline each method with
  - variable renaming
  - parameter passing
- At main, apply intraprocedural analysis

\[
p^{(a,d,l)} := \cdots := \text{malloc}_{o}^{(a,e)}(p, adl, o, ae)
\]
Key Idea of Merging

\[(p, \text{adl}, o, ae)\] is equivalent to \[(p, bfdl, o, bfe)\]?

\[(p, \text{adl}, o, ae)\] \((p, d l, o, e)\)

\[(p, bfdl, o, bfe)\] \((p, fdl, o, fe)\)

creation of \((p, d l, o, e)\) is not related to callers of addName

\(\Rightarrow\) we can merge call-paths a, bf for \((p, d l, o, e) : \text{URC}\)
Points-to Analysis

• clone pointers and targets only for URCs
Performance
Conclusion

• Non-BDD-based technique that merges equivalent contexts with heap-cloning

• Faster than BDD-implementation when cloning heaps