Scalable Propagation-Based Algorithms for Call Graph Construction

Frank Tip (IBM Research)
Jens Palsberg (Purdue University)

OOPSLA 2000 (Object-Oriented Programming, Systems, Languages, and Applications)
PROLANGS Group

- Programming languages and compilers
- Software engineering
  - Understanding, restructuring, maintenance, testing, verification
- Static analysis
  - a.k.a. program analysis
Outline

- Static analysis
- Class analysis for OO programs
  - Two simple analyses: CHA and RTA
- Tip & Palsberg
  - Four new analyses
  - Empirical results
- Summary
Static Analysis

- The program is not actually executed
- Output: info about run-time properties
  - “There exists a program execution for which property X is true”
  - “Property X is true for all possible executions”
Examples

- Every time \( b = a + 1 \) is executed, \( a \) has value 2
  - Compiler optimization: replace with \( b = 3 \)
- There exists an execution of \( *p = 1 \) in which \( p \) points to variable \( x \)
  - Disambiguate the indirect expression \( *p \)
Applications

- Compiler optimizations
  - Traditional application area
- Software maintenance
  - Complex dependencies in large programs
  - If I make this change, what will be affected?
    - “What code uses the value computed at this statement?”
Applications (cont)

- Software checking
  - Find bugs without executing the program
    - “Variable x is used without being initialized”
    - “Possible race condition for variable x”

- Software testing
  - Statically: find all possible alternatives
  - Dynamically: run test suites and track which alternatives have been tested
    - Evaluate and improve test suites
Object-Oriented Programs

- Declare a set of **classes**
  - A class may be a **subclass** of another class
- Create **objects** of certain classes
  - Java, C++: `new A()`
- Some **variables** may point to objects
  - Java: `A x;` C++: `A* x;`
  - Example: statement `x=new A();`
Class Analysis for OO Programs

- At different moments, a variable may point to objects from different classes

- Example: class A, subclasses B and C
  - Java: variable x of type A may point to objects of classes A, B, or C

- Given a variable x, what are the classes of the objects that x may point to?
Class Hierarchy Analysis (CHA)

- CHA: the simplest class analysis
  - Start with the type T of the variable
  - Find all subclasses of T (transitively)
- Problem: spurious classes
  - \texttt{A x; ... x = new B(); ...}
- Solution: more sophisticated analyses
  - Fewer spurious classes
  - Increased analysis cost
Applications of Class Analysis

- Resolution of virtual calls
  - \texttt{x.m();} may invoke different methods

- Example: class A, subclasses B and C
  - Method \texttt{m} declared in A and \texttt{overridden} in C
  - Variable \texttt{x} of type A, at virtual call \texttt{x.m();}
    - Possible targets: \texttt{A.m} and \texttt{C.m}
  - What if \texttt{x} only points to objects of A or B?
Applications of Class Analysis

- Call graph construction
  - Required for many other static analyses
  - Removal of virtual calls + inlining

- Compiler optimizations
  - Removal of redundant synchronization
  - Allocation of objects on the stack

- Tools for maintenance and testing
Rapid Type Analysis (RTA)

- “Type analysis” = “Class analysis”
- Ignores unused parts of the program
- Incrementally builds 2 sets
  - Methods reachable from `main`
  - Classes that are instantiated in reachable methods: `new A()`
- Solution: \( \text{CHA} \cap \text{InstantiatedClasses} \)
High-level Structure

- Call graph construction during the analysis
  - Same is true for most class analyses
- Queue of methods waiting to be processed
  - If m is found to be reachable => Q.add(m)
- while (Q.notEmpty())
  processMethodBody(Q.getFirst());
Processing Method Bodies

- Call statements
  - Add new reachable methods to \( Q \)
  - Virtual calls: find possible targets \textbf{only} w.r.t. \textit{InstantiatedClasses}

- Object creation statements
  - \textbf{new A()}: add A to \textit{InstantiatedClasses}
  - May have to revisit virtual calls
More Precise Analyses

- RTA keeps one set of classes
- Idea: use many separate sets
  - Granularity: class, method, field, statement
  - Call graph construction during the analysis
- Tip & Palsberg: 4 different analyses
  - Relatively efficient (scalable)
  - Relatively imprecise
CTA: Class-level Granularity

- For each class C, consider the set of
  - all variables occurring in C’s methods
  - all fields declared in C
- Maintain a set of possible classes $S_C$ for this set of fields/variables
- Same overall structure as RTA
  - Different actions inside method bodies
CTA: Propagation

- new A() inside class C: add $A$ to $S_C$
- Inside class C: read of field $f$ (decl. in B)
- Some of C’s variables/fields may be assigned the value of $f$
  - Add to $S_C$ everything that $f$ points to
  - Propagate $S_B$ to $S_C$
  - More precisely: add $S_B \cap \text{CHA}(f)$ to $S_C$
CTA: Propagation

- Inside class C: write of field f
  - f may be assigned the value of some of C’s variables/fields
  - Propagate $S_C$ to $S_B$
  - More precisely: add $S_C \cap CHA(f)$ to $S_B$

- Similarly for calls:
  - Actual-to-formal parameters
  - Return values
Finer Granularity

- MTA: add separate set of possible classes $S_f$ for each field $f$
  - Methods are treated as in CTA: $S_C$
- FTA: add separate set of possible classes $S_m$ for each method $m$
  - Fields are treated as in CTA: $S_C$
- $XTA = FTA + MTA$
Empirical Results (RTA vs XTA)

- 12 benchmark programs
  - Between 44 and 2332 classes
- Running times
  - RTA typically runs in a few seconds
  - XTA is 5 times slower on average
  - The analyses are fast and scalable
Precision Improvements

- Number of reachable methods
  - XTA: 1.6% less on average

- Number of call graph edges
  - XTA: 7.2% less on average

- Single-target virtual calls
  - All analyses do a good job
  - XTA is 12.5% better on average
Summary

- Class analysis: fundamental problem
  - CHA and RTA: simple and fast
- Tip & Palsberg: more precise analyses
  - Finer granularity: class, method, field
  - More expensive than RTA, but still OK
  - Is the extra precision worth the effort?
- Many more class analyses