Dataflow Analysis: Class Analysis (conclusion)
Announcements

- Quiz 2

- HW2
  - Post question on Submitty
    - Setup, please do set this up as soon as possible!
    - Starter code, class analysis framework and worklist algorithm
    - Soot
  - There are already some useful posts
Announcements

- Office hours change
  - Mondays 4-5pm on Webex
  - Fridays 4-5pm on Webex
  - Mondays and Thursdays 2pm - in SAGE 3713
  - or Wednesdays 3:30pm - 4:30pm
Outline of Today’s Class

- Rapid Type Analysis (RTA), last time
- HW2, Class analysis framework questions?
- The XTA analysis family
- 0-CFA
- Points-to analysis (PTA)
Your Homework

- A bunch of flow-insensitive, context-insensitive analyses for Java
  - RTA, XTA, and optionally other
  - Simple property space
  - Simple transfer functions
    - E.g., in fact, RTA gets rid of most CFG nodes, processes just 2 kinds of nodes

- Millions of lines of code in seconds
“Classical” Points-to Analysis
(Imperative, i.e., “operational”)

- Flow-insensitive, context-insensitive analysis
  - Makes sense for points-to analysis

\[ Pt = 0 \] /* initialize solution to empty points-to graph */

\[ F = \{ f_1, f_2, \ldots, f_n \} \] /* all transfer functions, without “kills“, including ones for implicit assignments */

\[ W = \{ f_1, f_2, \ldots, f_n \} \]

while \( W \neq \emptyset \) do {
    remove \( f_j \) from \( W \)
    \[ Pt = f_j(Pt) \]
    if \( Pt \) changed then
        \[ W = W \cup F \] /* Safe to add all transfer functions! */
“Classical” Points-to Analysis (vs. Declarative)

Known as Andersen’s Points-to Analysis

\(\text{pts}(p)\) denotes the points-to set of \(p\)

1. \(p = \&a\) \(\{ a \} \subseteq \text{pts}(p)\)
2. \(p = q\) \(\text{pts}(q) \subseteq \text{pts}(p)\)
3. \(p = *q\) for each \(x\) in \(\text{pts}(q)\). \(\text{pts}(x) \subseteq \text{pts}(p)\)
4. \(*p = q\) for each \(x\) in \(\text{pts}(p)\). \(\text{pts}(q) \subseteq \text{pts}(x)\)

Use worklist-like algorithm to compute least solution of these constraints
Class Analysis

- Problem statement: What are the **classes** of objects that a (Java) **reference** variable may refer to?

- Applications
  - Call graph construction
    - Nodes are method
    - Edges represent calling relationships
    - Notion of methods reachable from **main**
  - Virtual call resolution
R is the set of reachable methods
I is the set of instantiated types

1. \{ \text{main} \} \subseteq R \quad // \text{Algo: initialize } R \text{ with } \text{main}

2. for each method \( m \in R \) and each new site \( \text{new } C \) in \( m \)
   \{ C \} \subseteq I \quad // \text{Algo: add } C \text{ to } I; \text{ schedule}
   // “successor” constraints
3. for each method \( m \in R \), each **virtual call** \( y.n(z) \) in \( m \), each class \( C \) in \( \text{SubTypes}(\text{StaticType}(y)) \cap I \), and \( n' \), where \( n' = \text{resolve}(C,n) \)

\[ \{ n' \} \subseteq R \] // Algo: add target \( n' \) to \( R \), if not already there. Schedule “successors”
Let’s take a moment (or two) to go over HW2 class analysis framework.
**XTA Analysis Family**

- Due to Tip and Palsberg
  - Frank Tip and Jens Palsberg, “Scalable Propagation-Based Call Graph Construction Algorithms”, OOPSLA ’00

- Generalizes RTA
- Improves on RTA by keeping more info
  - What if we kept sets per method and per field rather than a “blob”?
\( R \) is the set of reachable methods

\( S_m \) is the set of types that flow to method \( m \)

\( S_f \) is the set of types that flow to field \( f \)

1. \( \{ \text{main} \} \subseteq R \)

2. for each method \( m \in R \) and each new site new \( C \) in \( m \)
   \( \{ C \} \subseteq S_m \)
3. for each method $m \in R$, each virtual call $y.n(z)$ in $m$, each class $C$ in $\text{SubTypes(StaticType}(y)) \cap S_m$ and $n'$, where $n' = \text{resolve}(C,n)$

$$\{ n' \} \subseteq R \quad \text{// add n' to R if not already there}$$

$$\{ C \} \subseteq S_{n'} \quad \text{// add C to S_{n'}, if not already there}$$

$$S_m \cap \text{SubTypes(StaticType}(p)) \subseteq S_{n'}$$

$$S_{n'} \cap \text{SubTypes(StaticType}(ret)) \subseteq S_m$$

($p$ denotes the parameter of $n'$, and $ret$ denotes the return of $n'$)
4. for each method $m \in R$, each field read $x = y.f$ in $m$
\[ S_f \subseteq S_m \]

5. for each method $m \in R$, each field write $x.f = y$ in $m$
\[ S_m \cap \text{SubTypes}(\text{StaticType}(f)) \subseteq S_f \]
Practical Concerns

- Multiple parameters
- Direct calls
  - either static invoke calls or
  - special invoke calls
- Array reads and writes!
- Static fields

See Tip and Palsberg for more
Example: RTA vs. XTA

```java
public class A {
    public static void main() {
        n1();
        n2();
    }

    static void n1() {
        A a1 = new B();
        a1.m();
    }

    static void n2() {
        A a2 = new C();
        a2.m();
    }
}
```

RTA:
- `R = \{ \text{main}, n1, n2 \}
  \cup \{ \text{B.m()}, \text{C.m()} \}
- `\( I = \{ B, C \} \)`
- `\{ B \}`
- `\{ C \}`

XTA:
- `\{ B \}`
- `\{ C \}`
public class `AndExp` extends `BoolExp` {
    private `BoolExp` left;
    private `BoolExp` right;

    public `AndExp`(`BoolExp` left, `BoolExp` right) {
        this.left = left;
        this.right = right;
    }

    public boolean evaluate(Context c) {
        private `BoolExp` l = this.left;
        private `BoolExp` r = this.right;
        return l.evaluate(c) && r.evaluate(c);
    }
}
public class OrExp extends BoolExp {
    private BoolExp left;
    private BoolExp right;

    public OrExp(BoolExp left, BoolExp right) {
        this.left = left;
        this.right = right;
    }

    public boolean evaluate(Context c) {
        private BoolExp l = this.left;
        private BoolExp r = this.right;
        return l.evaluate(c) || r.evaluate(c);
    }
}

Boolean Expression Hierarchy:
RTA vs. XTA vs. “Ground Truth”
main() {
    Context theContext = new Context();
    BoolExp x = new VarExp("X");
    BoolExp y = new VarExp("Y");
    BoolExp exp = new AndExp(  
        new Constant(true), new OrExp(x, y) );
    theContext.assign(x, true);
    theContext.assign(y, false);
    boolean result = exp.evaluate(theContext);
}

Boolean Expression Hierarchy:  
RTA vs. XTA vs. "Ground Truth"

\[ \mathcal{M} = \{ \text{VarExp, Constant, AndExp, OrExp} \} \]

\[ \text{Sim}_{\text{RTA}} \leq \text{Sim}_{\text{XTA}} \leq \text{Sim}_{\text{ Ground Truth }} \]
Outline of Today’s Class

- Rapid Type Analysis (RTA), last time
- HW2, Class analysis framework questions?
- The XTA analysis family
  - 0-CFA
  - Points-to analysis (PTA)