Overview

- Introduction
- Single & efficient: CHA, RTA
- Analyzing assignments: VTA, DTA
- Call graphs and points-to analysis: Spark
Variable Type Analysis (VTA)

- Reason about assignments
- Infer what types the objects involved in a call may have
- Prune calls that are infeasible based on the inferred types
Example

```java
public class A {
    public void f(C c) {
        c.m();
    }
}

public class B {
    public void f(C c) {
        c.m();
    }
}
```

```java
a = new X();
...
b = a;
...
o.f(b);
```
Type Propagation

Four steps:

- Form initial conservative call graph
  - E.g., using CHA or RTA
- Build type-propagation graph
- Collapse strongly connected components
- Propagate types in one iteration
Building Type Propagation Graph

- Assume statement $a = b$; is in method $C.m$

  ![Diagram](image)

- Assume another statement $a.f = b$; where field $f$ is declared in $A$

  ![Diagram](image)
A a1, a2, a3;
B b1, b2, b3;
C c;

a1 = new A();
a2 = new A();
b1 = new B();
b2 = new B();
c = new C();

a1 = a2;
a3 = a1;
a3 = b3;
b3 = (B) a3;
b1 = b2;
b1 = c;
Note: Slide fixed and lecture video
Side Note: Field Representations

How does the analysis represent \( a.f \)?

- **Field-sensitive**: Represented as \( a.f \)
- **Field-insensitive**: Represented as \( a.* \) or \( a \)
- **Field-based**: Represented as \( A.f \), where \( A \) is class of \( a \)
How does the analysis represent $a.f$?

- **Field-sensitive**: Represented as $a.f$
- **Field-insensitive**: Represented as $a.*$ or $a$
- **Field-based**: Represented as $A.f$, where $A$ is the class of $a$

VTA is field-based
Variable Type Analysis (VTA)

**Pros**
- More precise than RTA: Considers only those types that may actually reach the call site
- Still relatively fast

**Cons**
- Requires initial call graph (i.e., actually a refinement algorithm)
- Some imprecision remains, e.g., because of field-based analysis
Declared-Type Analysis (DTA)

- “Small brother of VTA”
- Also reasons about assignments and how they propagate types
- But: Not per variable, but per type
Declared-Type Analysis (DTA)

- **Pros**
  - Faster than VTA: Graph is smaller, propagation is faster
  - More precise than RTA

- **Cons**
  - Less precise than VTA: Does not distinguish variables of same type