Program Analysis

Call Graphs (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

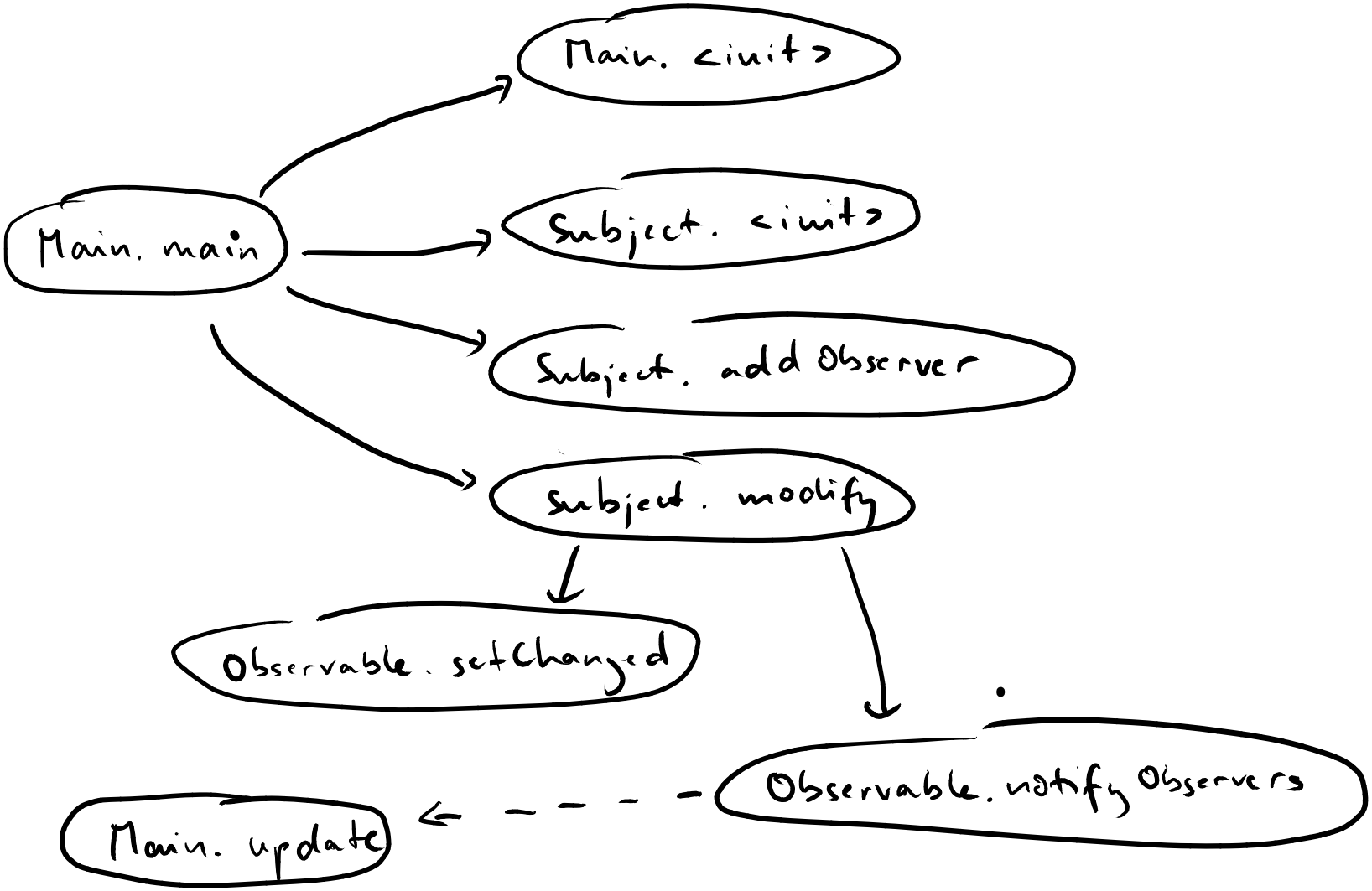
Slides adapted from Eric Bodden

Call Graph Analysis

- **Call graph: Abstraction of all method calls in a program**
  - Nodes: Methods
  - Edges: Calls
  - Flow-insensitive: No execution order
- **Here: Static call graph**
  - Abstract of all calls that may execute

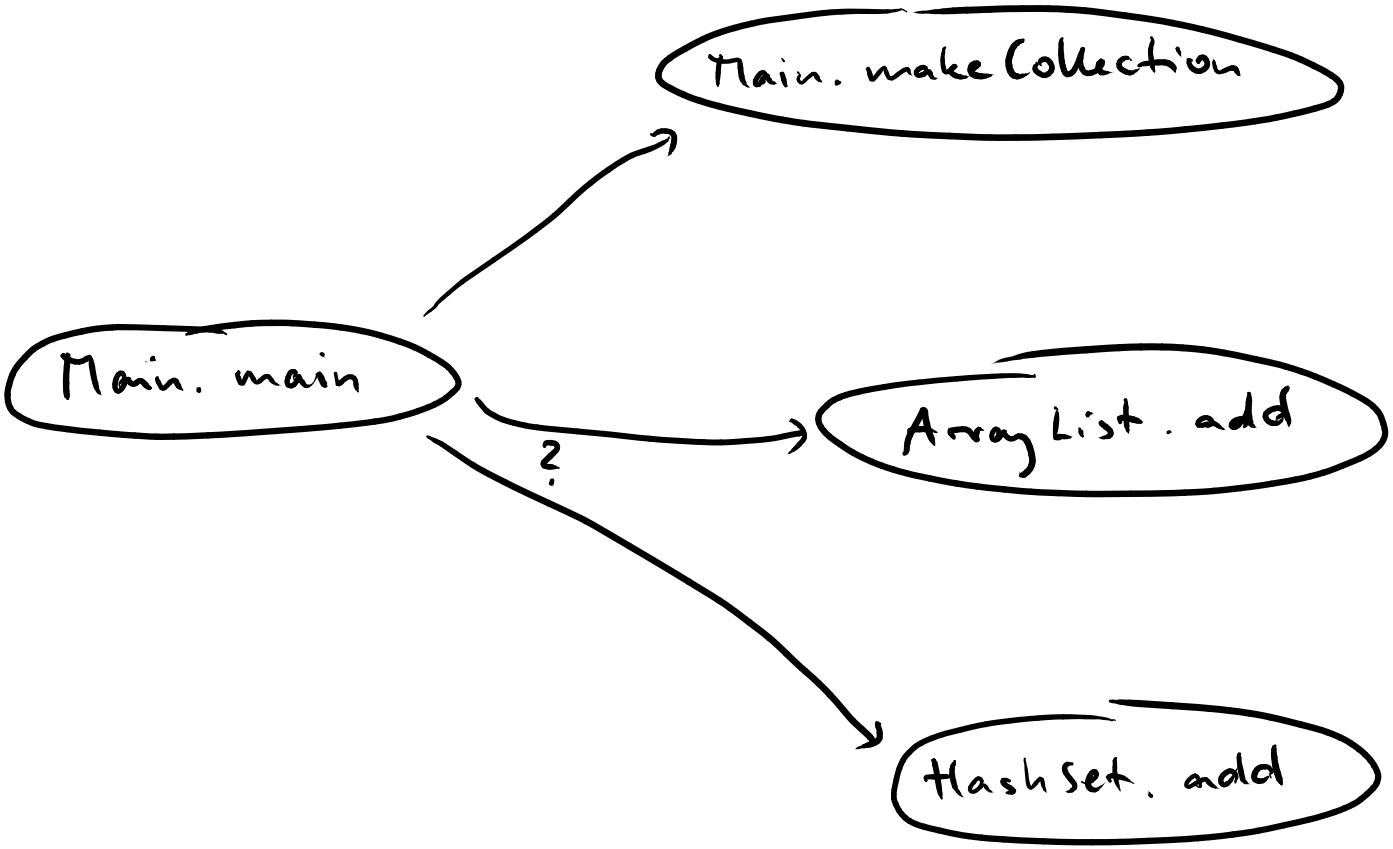
Example

```
public class Main implements Observer {
  public static void main(String[] args) {
    Main m = new Main();
    Subject s = new Subject();
    s.addObserver(m);
    s.modify();
  }

  public void update(Observable o, Object arg) {
    System.out.println(o+" notified me!");
  }

  static class Subject extends Observable
    public void modify() {
      setChanged();
      notifyObservers();
    }
}
```


Main.<init>
Main.main
Subject.<init>
Subject.addObserver
Subject.modify
Observable.setChanged
Observable.notifyObservers
Main.update

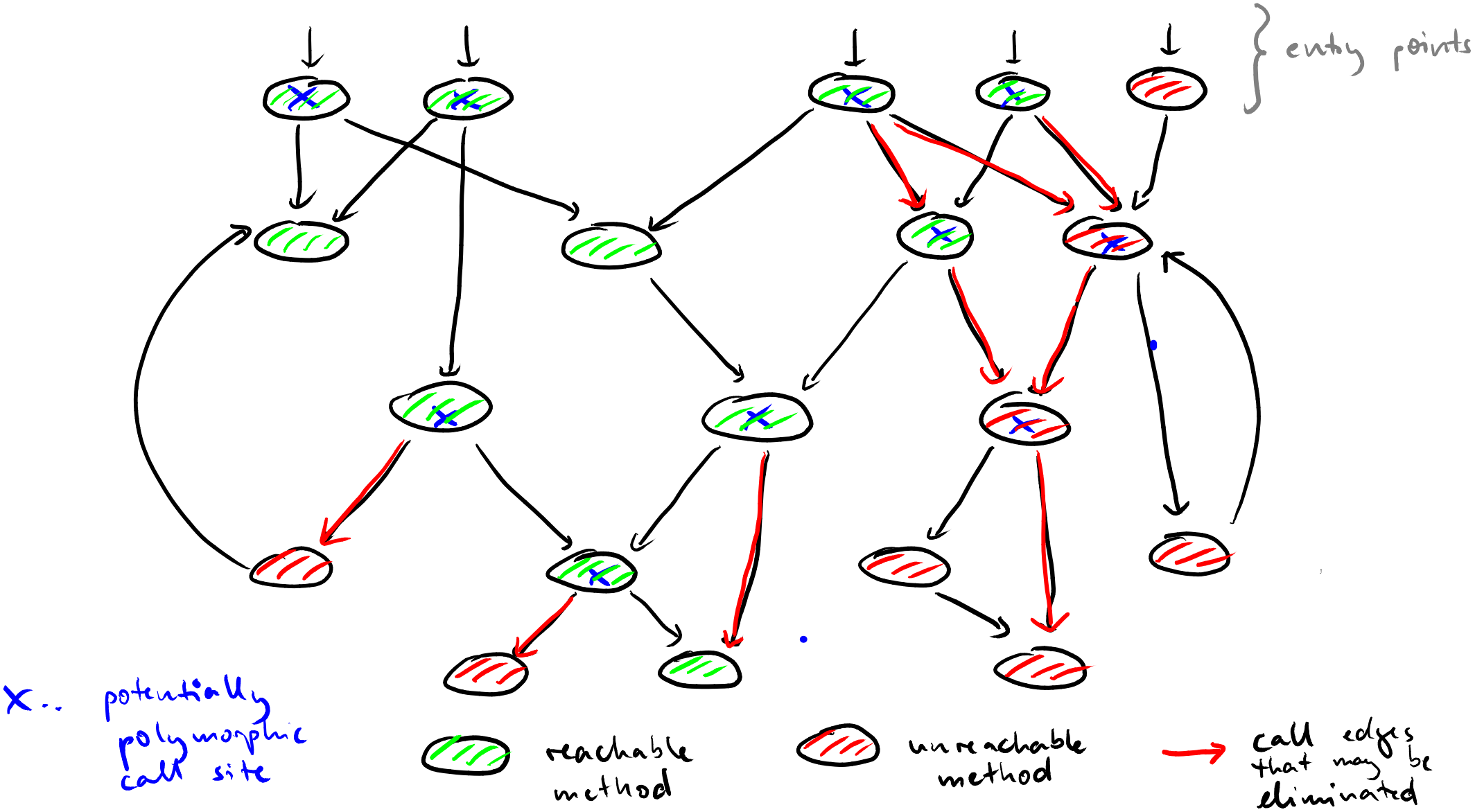
Problem: Polymorphic Calls

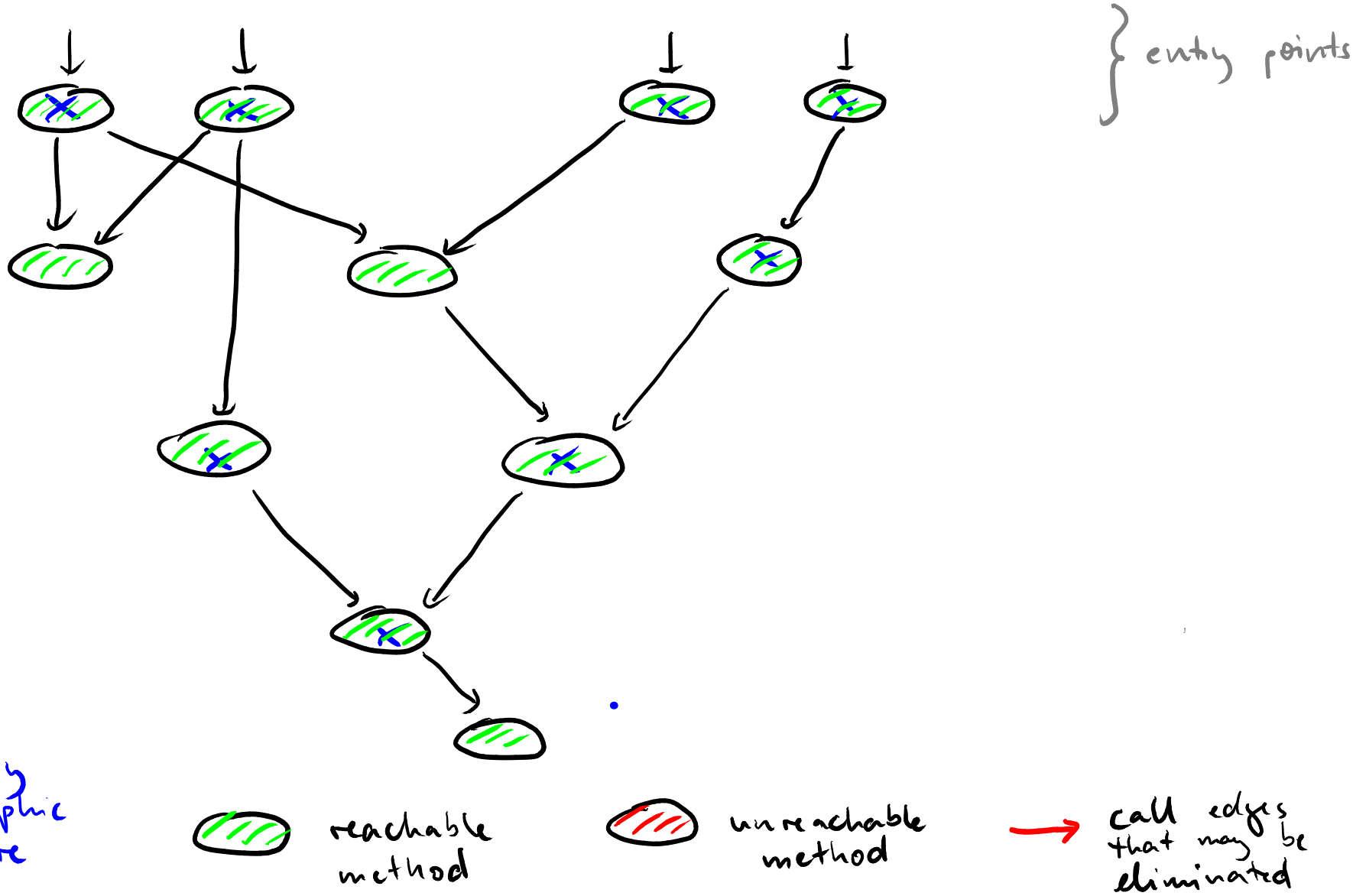
```
import java.util.*;

public class Main {
  public static void main(String[] args) {
    Collection c = makeCollection(args[0]);
    c.add("hello");
  }

  static Collection makeCollection(String s) {
    if(s.equals("list")) {
      return new ArrayList();
    } else {
      return new HashSet();
    }
  }
}
```


Main.makeCollection
Main.main
?
ArrayList.add
HashSet.add

Improving the Call Graph

- **Prune graph:**
  Focus on feasible behavior
- **Want to minimize**
  - Reachable methods
  - Call edges
  - Potentially polymorphic call sites


entry points
X.. potentially polymorphic call site
reachable method
unreachable method
call edges that may be eliminated


entry points
X.. potentially polymorphic call site
reachable method
unreachable method
call edges that may be eliminated

Overview

- Introduction ✔
- Single & efficient: CHA, RTA
- Analyzing assignments: VTA, DTA
- Call graphs and points-to analysis: Spark