Program analysis

Software Engineering
Andrzej Wasylkowski • Saarland University

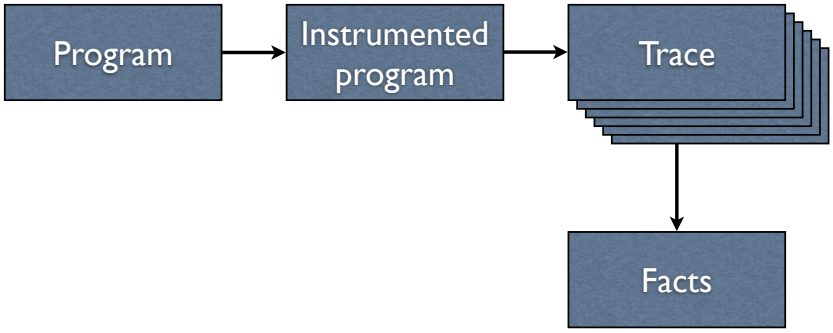
Program analysis: Overview

- Means of discovering facts about programs
  - Type correctness
  - Performance bottlenecks
  - Optimization possibilities
  - Potential defects
  - …

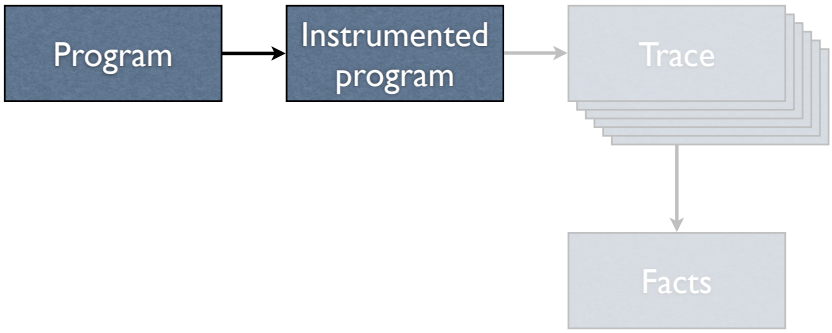
Program analysis: Overview

- Two types of program analyses
  - Static analysis
  - Dynamic analysis

Dynamic analysis: How does it work?



Dynamic analysis: Instrumentation



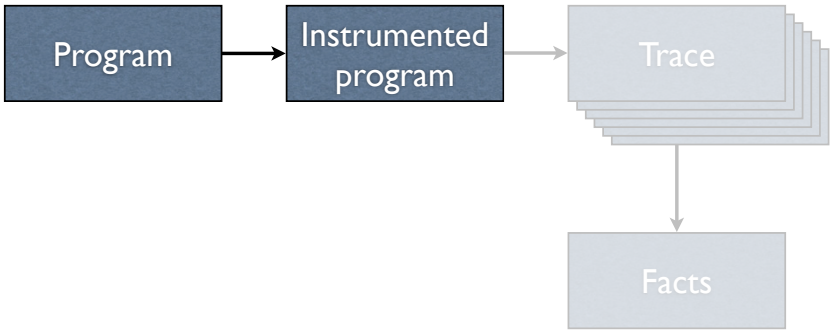
Dynamic analysis: Instrumentation

```
…
public String getSummary () {
  StringBuffer summary = new StringBuffer ();
  summary.append (this.name);
  summary.append (" (");
  summary.append (getTotalPrice ());
  summary.append (")");
  return summary.toString ();
}
…
```

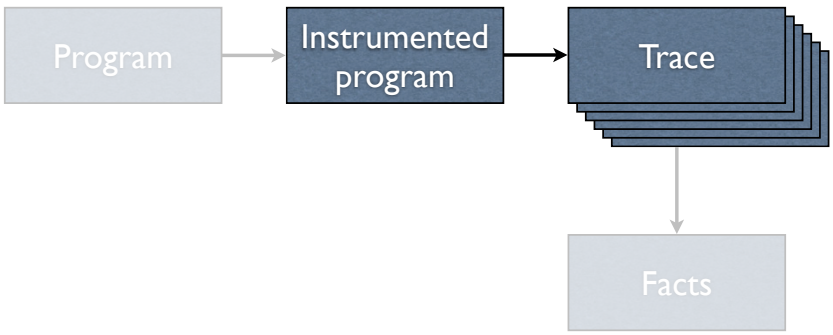
Dynamic analysis: Instrumentation

```
…
public String getSummary () {
  try {
    Tracer.entry ("getSummary entry on " + this.name);
    StringBuffer summary = new StringBuffer ();
    summary.append (this.name);
    summary.append (" (");
    summary.append (getTotalPrice ());
    summary.append (")");
    return summary.toString ();
  }
  finally {
    Tracer.exit ("getSummary exit on " + this.name);
  }
}
…
```

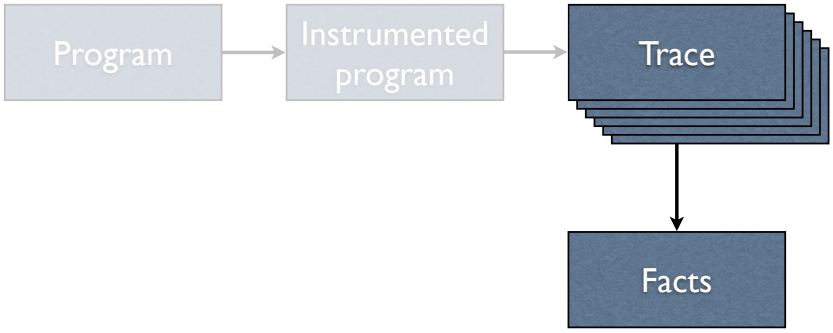
Dynamic analysis: Instrumentation



Dynamic analysis: Running



Dynamic analysis: Analyzing the traces



Dynamic analysis: Analyzing the traces

```
…
[1247571625107] process entry
…
[1247571625110] getSummary entry on camera
[1247571625115] getTotalPrice entry on camera
…
[1247571625205] getTotalPrice exit on camera
[1247571625210] getSummary exit on camera
[1247571625250] getSummary entry on iron
…
[1247571666440] process exit
…
```

Dynamic analysis: Analyzing the traces

```
…
[1247571625107] process entry
…
[1247571625110] getSummary entry on camera
[1247571625115] getTotalPrice entry on camera
…
[1247571625205] getTotalPrice exit on camera
[1247571625210] getSummary exit on camera
[1247571625250] getSummary entry on iron
…
[1247571666440] process exit
…
```

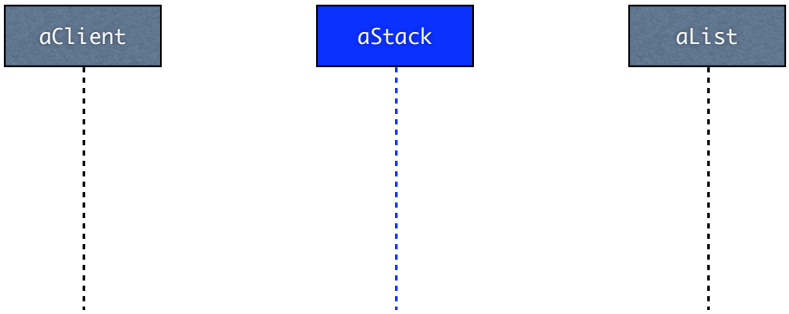
getTotalPrice is the bottleneck here

AMPLE

- Defect localization tool
- Uses dynamic analysis
  - Learns common program behavior
  - Uncommon behavior = defective component

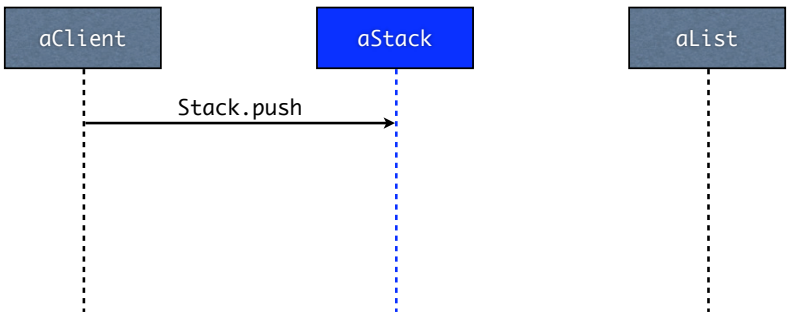
Dallmeier, Valentin, Christian Lindig, and Andreas Zeller. 2005. Lightweight defect localization for Java. In *ECOOP 2005: Proceedings of the 19th European conference on object-oriented programming*, 528–550. Lecture Notes in Computer Science 3586. Berlin: Springer-Verlag.

AMPLE: Tracing Objects



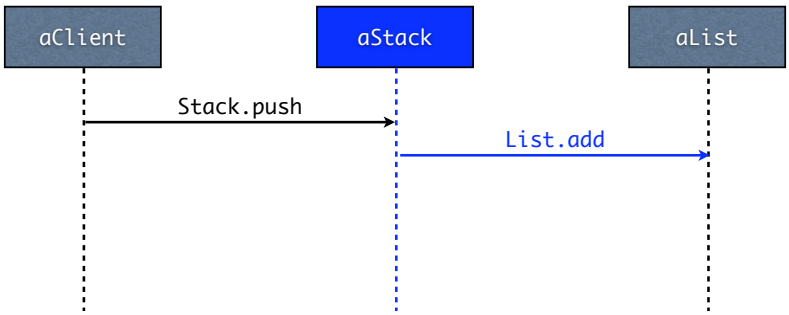
aStack:

AMPLE: Tracing Objects



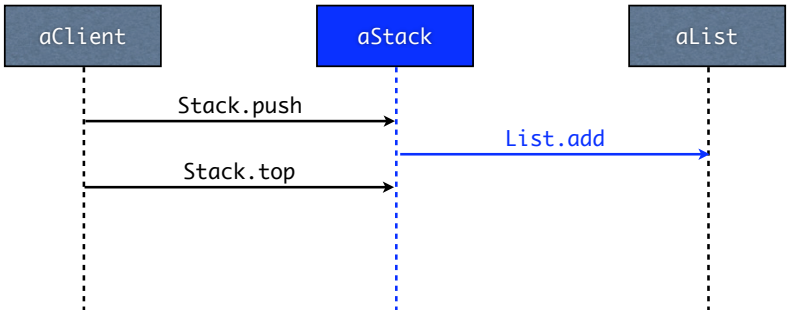
aStack:

AMPLE: Tracing Objects



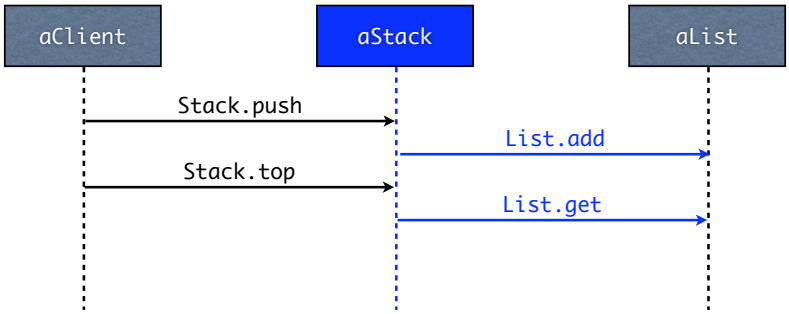
aStack: List.add

AMPLE: Tracing Objects



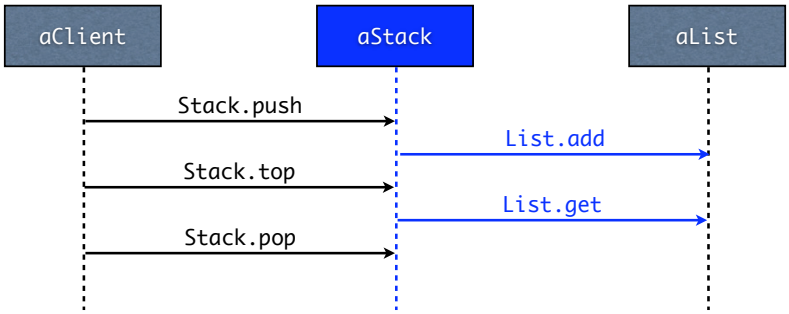
aStack: List.add

AMPLE: Tracing Objects



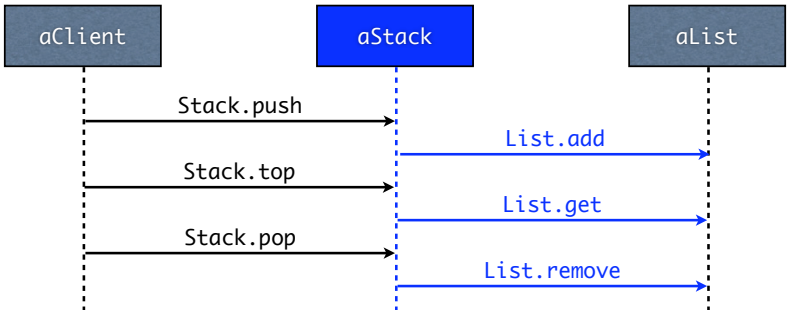
aStack: List.add, List.get

AMPLE: Tracing Objects



aStack: List.add, List.get

AMPLE: Tracing Objects



aStack: List.add, List.get, List.remove

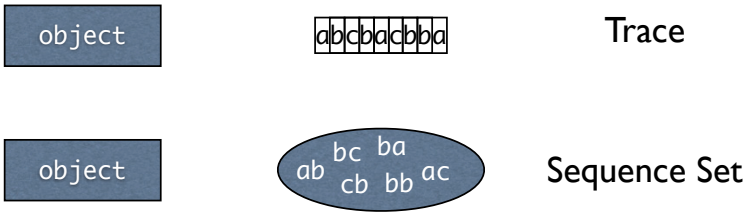
AMPLE: Aggregating Traces

object

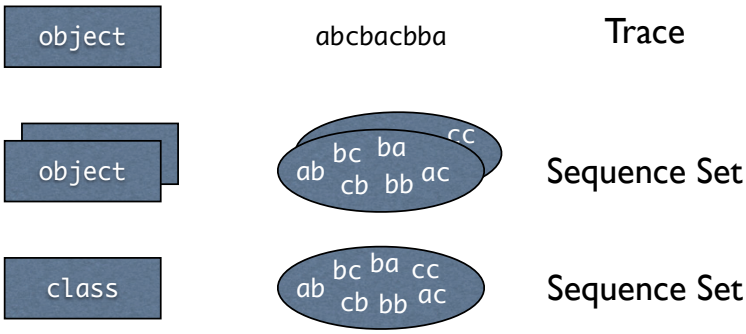
abcbacbba

Trace

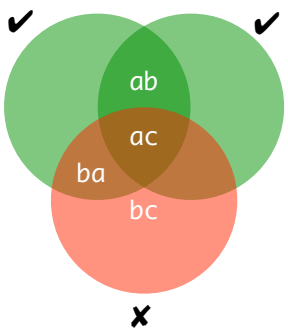
AMPLE: Aggregating Traces



AMPLE: Aggregating Traces



AMPLE: Ranking Classes



- Calculate weight of each sequence
- Calculate weight of each class
- High weight = responsible for the failure

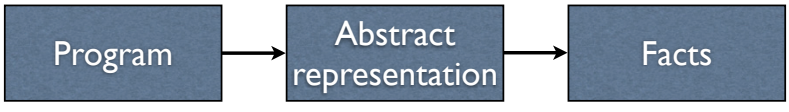
Dynamic analysis: Strengths

- Very precise
- Traces can be stored and reanalyzed
- Takes environment into account

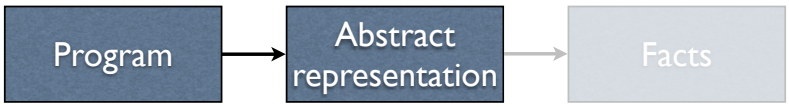
Dynamic analysis: Weaknesses

- Inherently unsound
- Depends on quality of test cases
- Slows down the execution
- Takes environment into account

Static analysis: How does it work?

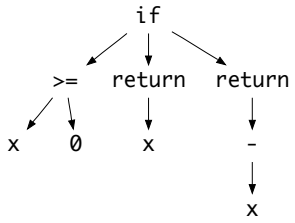


Static analysis: Abstraction



Static analysis: Creating an abstraction

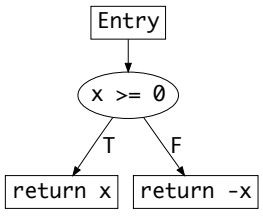
```
…
public int abs (int x) {
  if (x >= 0)
    return x;
  else
    return -x;
}
…
```



Method Abstract Syntax Tree

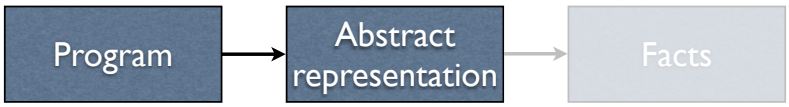
Static analysis: Creating an abstraction

```
…
public int abs (int x) {
  if (x >= 0)
    return x;
  else
    return -x;
}
…
```

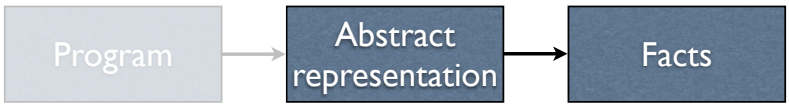


Method Control Flow Graph

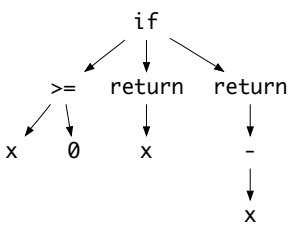
Static analysis: Abstraction



Static analysis: Facts extraction

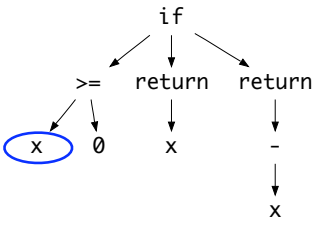


Static analysis: Facts extraction




```
x       : α
0       : int
retval  : β
```

Static analysis: Facts extraction



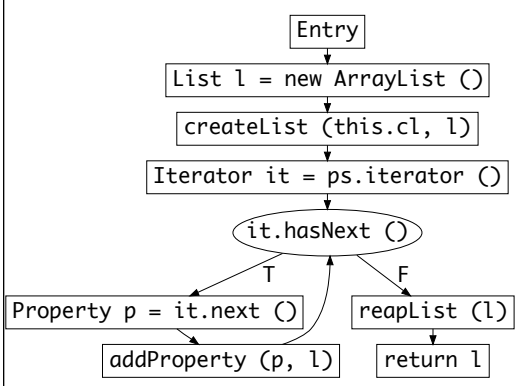

```
x      : int, float
0      : int
retval : int
```

JADET

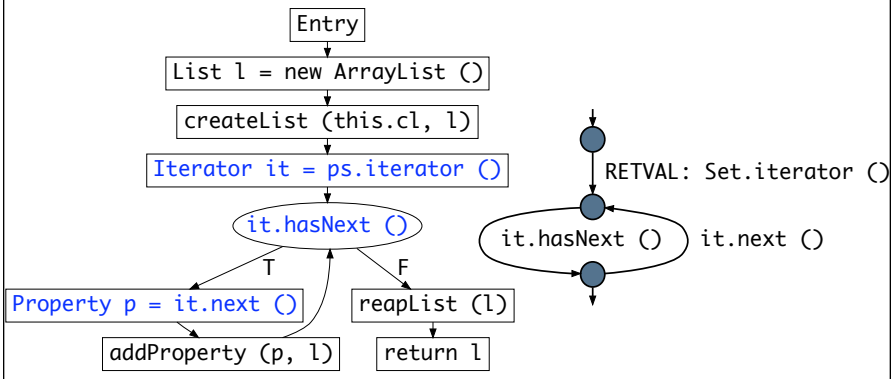
- Defect detection tool
- Uses static analysis
  - Learns common object usage
  - Uncommon usage = defective code

Wasylkowski, Andrzej, Andreas Zeller, and Christian Lindig. 2007. Detecting object usage anomalies. In *ESEC-FSE 2007: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 35–44. New York, NY: ACM.

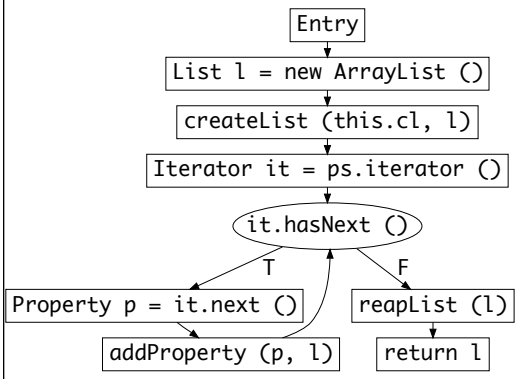
JADET: Creating models



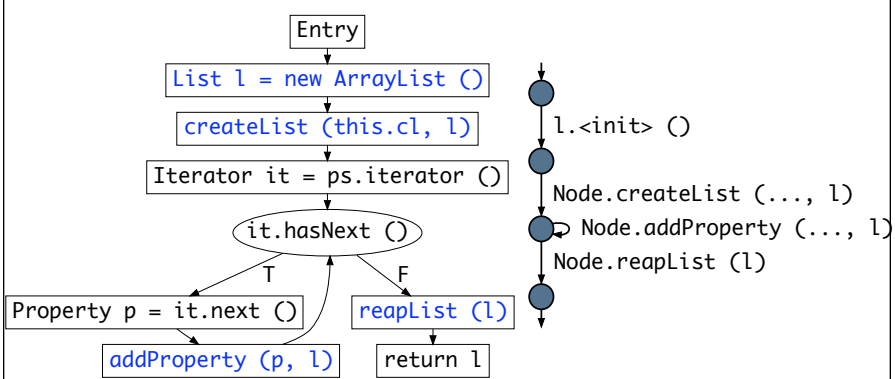
JADET: Creating models



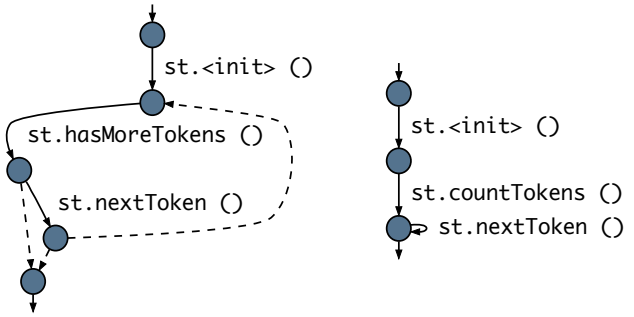
JADET: Creating models



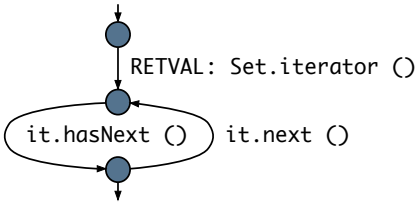
JADET: Creating models



JADET: Sample models

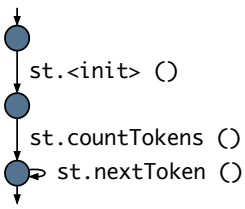


JADET: Temporal properties



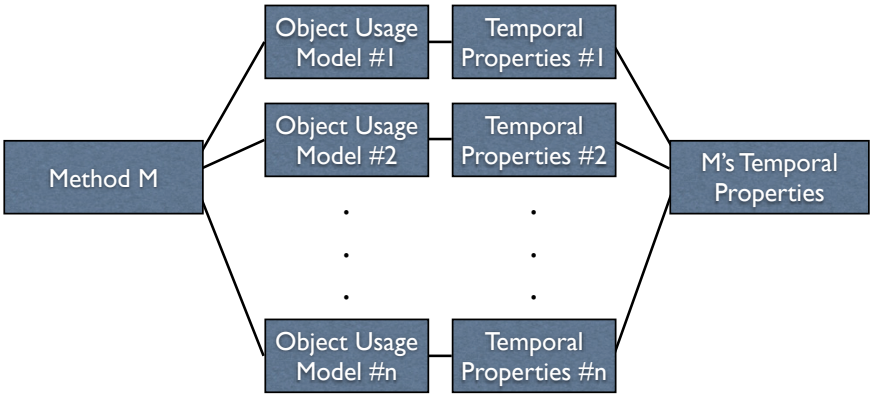

```
   RETVAL: Set.iterator() < Iterator.hasNext() @ this
   RETVAL: Set.iterator() < Iterator.next() @ this
Iterator.hasNext() @ this < Iterator.next() @ this
Iterator.hasNext() @ this < Iterator.hasNext() @ this
   Iterator.next() @ this < Iterator.hasNext() @ this
   Iterator.next() @ this < Iterator.next() @ this
```

JADET: Temporal properties

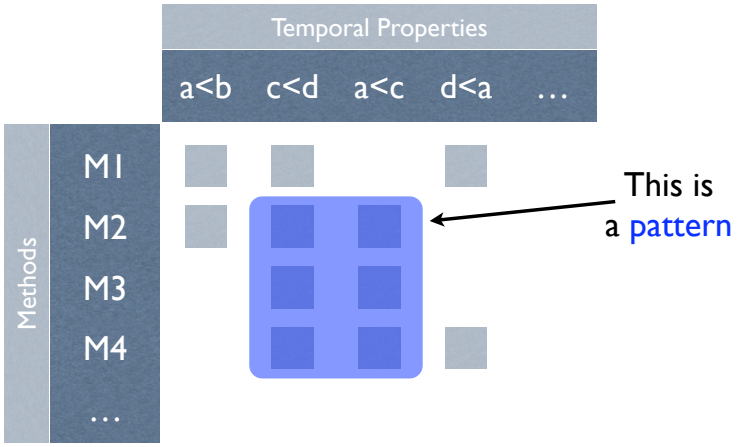



```
    StringTokenizer.<init>() @ this < StringTokenizer.countTokens() @ this
    StringTokenizer.<init>() @ this < StringTokenizer.nextToken() @ this
StringTokenizer.countTokens() @ this < StringTokenizer.nextToken() @ this
  StringTokenizer.nextToken() @ this < StringTokenizer.nextToken() @ this
```

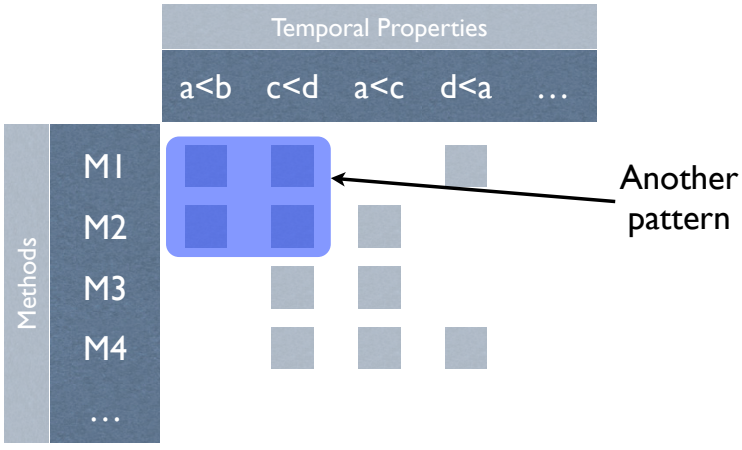
JADET: Overview



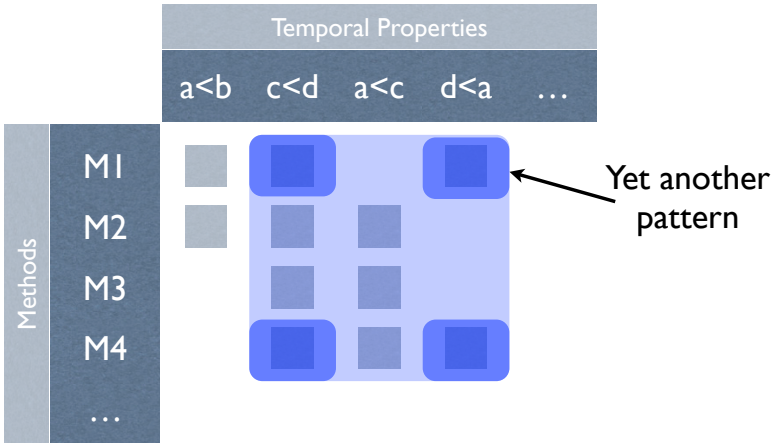
JADET: Methods vs. properties



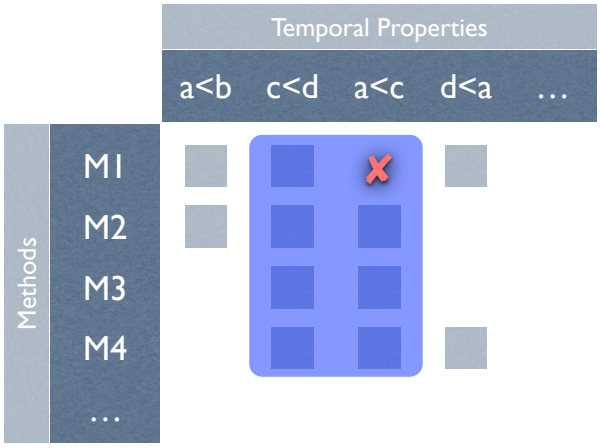
JADET: Methods vs. properties



JADET: Methods vs. properties



JADET: Detecting violations



Static analysis: Strengths

- Soundness (typically)
- Program need not be run
- Can be applied to parts only

Static analysis: Weaknesses

- Imprecise (in general undecidable)
- Time- and memory-consuming
- Can be difficult to get done right

Hybrid analysis?

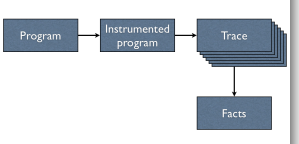
- Use static to drive dynamic
- Use dynamic to drive static
- True hybrid (multiple intertwined phases)

Program analysis: Overview

- Means of discovering facts about programs
  - Type correctness
  - Performance bottlenecks
  - Optimization possibilities
  - Potential defects
  - …

Summary

Dynamic analysis: How does it work?



Static analysis: How does it work?