Dead Paths Analysis

Applied Static Analysis 2016

Dr. Michael Eichberg (Organizer)

Johannes Lerch, Ben Hermann, Sebastian Proksch, Karim Ali Ph.D.

Software Technology Group

TECHNISCHE UNIVERSITÄT DARMSTADT

When is a path dead?

```
if (maxBits > 4 || maxBits < 8) {
    maxBits = 8;
}
if (maxBits > 8) {
    maxBits = 16;
}
```

Hidden inside a 278 LOC method

OpenJDK 8 update 25, com.sun.imageio.plugins.png.PNGMetadata, line 1842ff

When is a path dead?

```
if (maxBits > 4 || maxBits < 8) {
    maxBits = 8;
}
if (maxBits > 8) {
    maxBits = 16;
}
```

Hidden inside a 278 LOC method

OpenJDK 8 update 25, com.sun.imageio.plugins.png.PNGMetadata, line 1842ff

When is a path dead?

```
if (maxBits > 4 || maxBits < 8) {
    maxBits = 8;
}
if (maxBits > 8) {
    maxBits = 16;
}
```

Hidden inside a 278 LOC method

OpenJDK 8 update 25, com.sun.imageio.plugins.png.PNGMetadata, line 1842ff

When is a path dead?

```
if (maxBits > 4 || maxBits < 8) {
    maxBits = 8;
}
if (maxBits > 8) {
    maxBits = 16;
}
```

Hidden inside a 278 LOC method

OpenJDK 8 update 25, com.sun.imageio.plugins.png.PNGMetadata, line 1842ff

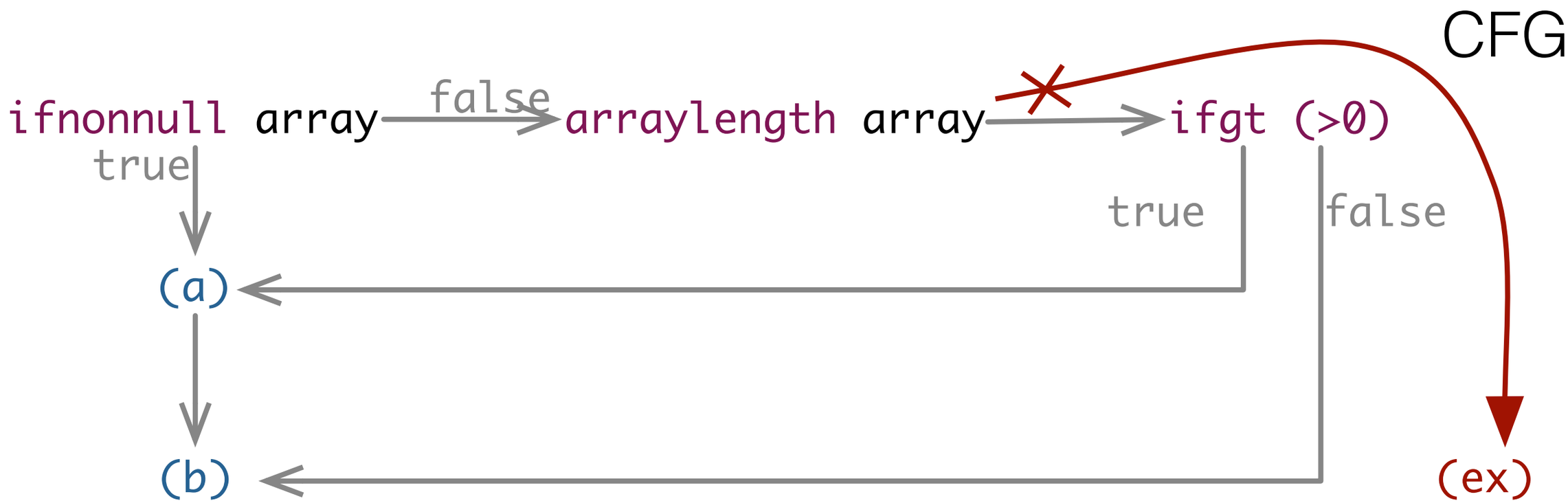
Hypothesis

In well-written code every path between an instruction and all its successors is eventually taken.

A path that will never be taken indicates an issue.

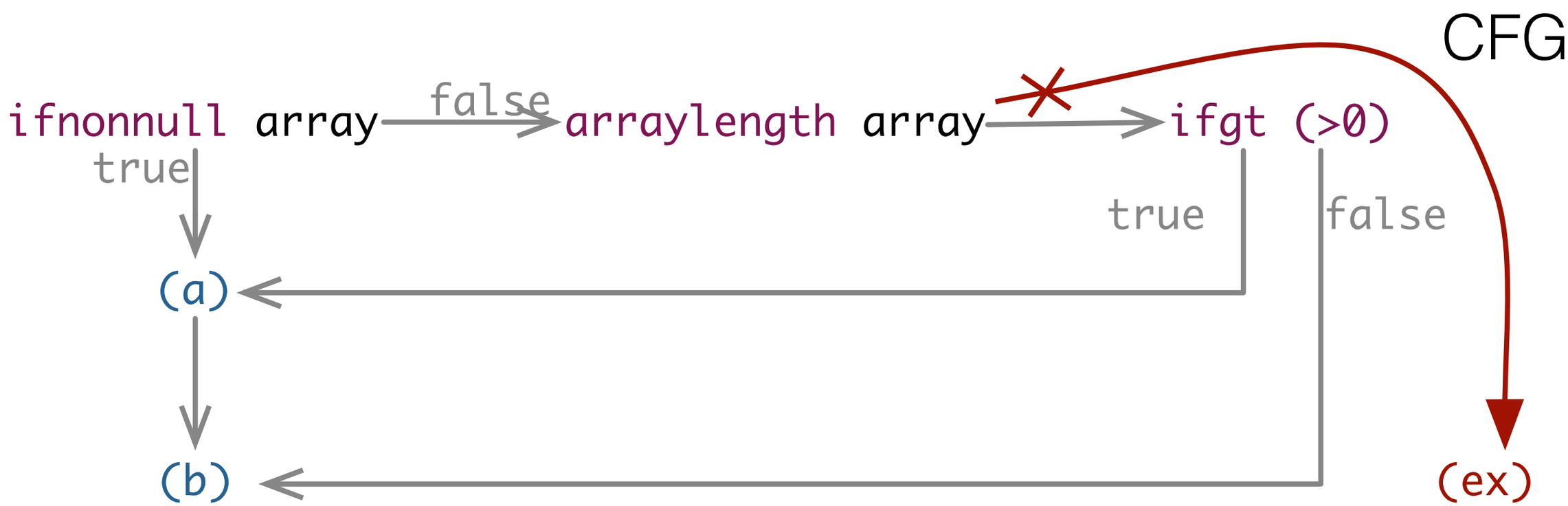
Identifying Infeasible Paths

```
public static X doX(SomeType[] array) {
   if (array != null || array.length > 0)
   { (a) }
   // … (b)
}// (ex)
```

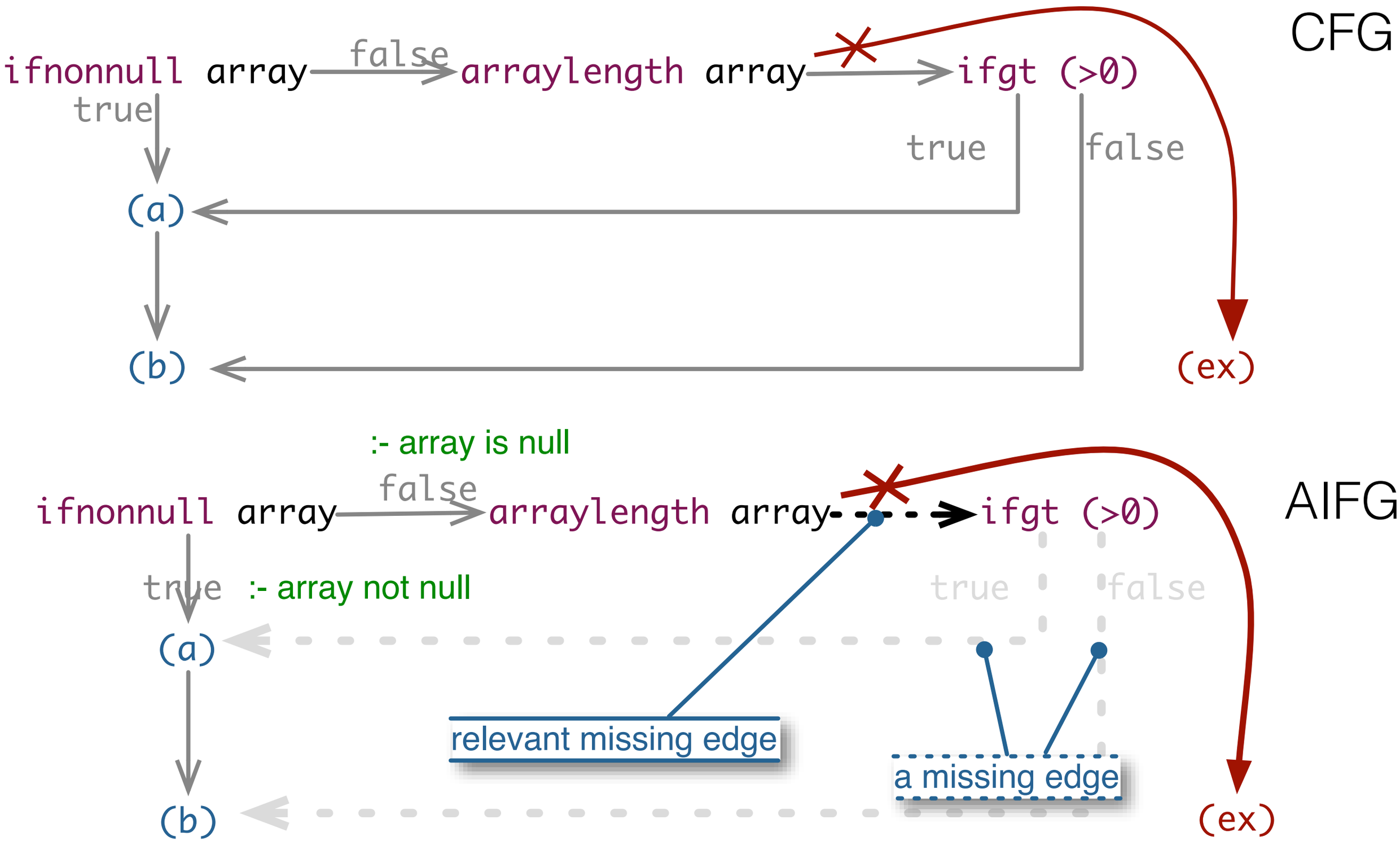


Identifying Infeasible Paths

```
public static X doX(SomeType[] array) {
   if (array != null || array.length > 0)
   { (a) }
   // … (b)
}// (ex)
```



Identifying Infeasible Paths



Abstract Interpretation

- Not targeted at a specific goal
- Not a whole program analysis, but instead everything may be an entry point
- Inter-procedural, path-, flow-, object- and context-sensitive with configurable call chain length (typically low)

Goals

Abstract Interpretation

- Not targeted at a specific goal
- Not a whole program analysis, but instead everything may be an entry point
- Inter-procedural, path-, flow-, object- and context-sensitive with configurable call chain length (typically low)

Goals

Usable for libraries

Abstract Interpretation

- Not targeted at a specific goal
- Not a whole program analysis, but instead everything may be an entry point
- Inter-procedural, path-, flow-, object- and context-sensitive with configurable call chain length (typically low)

Goals

Usable for libraries

Scales to industry-size applications

Abstract Interpretation

- **Integers**
  Support all arithmetic operations (of the JVM); maximum size for intervals before we consider them as AnyInt.

- **Reference value**
  Objects distinguished by their allocation site; alias- and path-sensitive.

Post-Processing

Compiler Generated Dead Code

The Intricacies of Java

Established Idioms

Assertions

Reflection and Reflection-like Mechanisms

Post-Processing

Compiler Generated Dead Code

```
void conditionInFinally(java.io.File f) {
  boolean completed = false;
  try {
    f.canExecute();
    completed = true;
  } finally {
    if (completed) doSomething(); }
}
```

Finally blocks are included twice by Java compilers which - in combination with standard idioms - often leads to dead code in the byte code!

Post-Processing

The Intricacies of Java

```
Throwable doFail() { throw new Exception(); }

Object compute(Object o) {
  if (o == null) { return doFail(); }
  else return o;
}
```

Using a method that always throws an exception leaves a dead return path.

Post-Processing

Established Idioms

```
switch (i) {
  case 1: break;
  // complete enumeration of all cases
  default: throw new UnknownError();
}
```

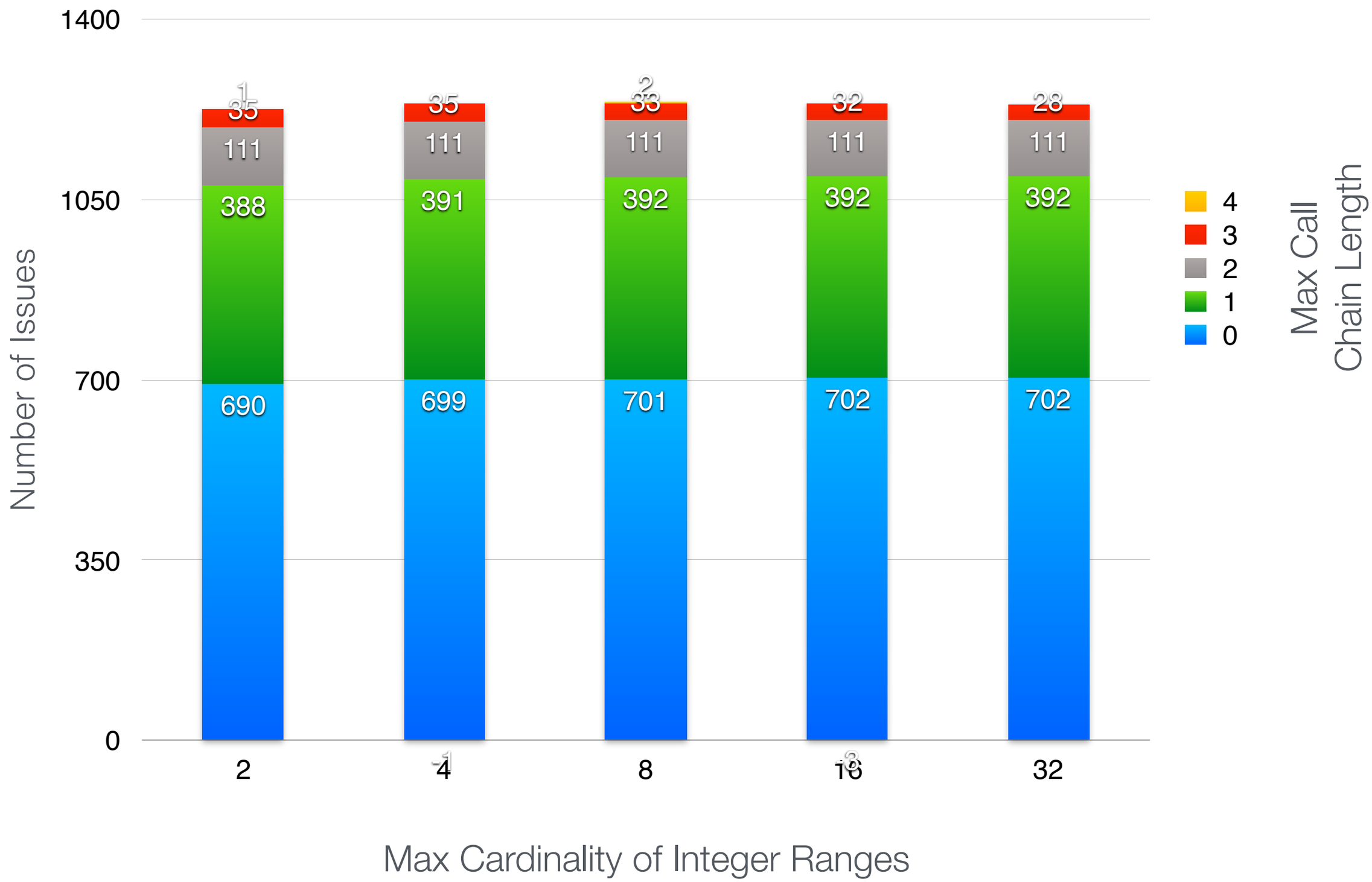
The analysis often proves that the default case cannot occur and, therefore, is dead code.

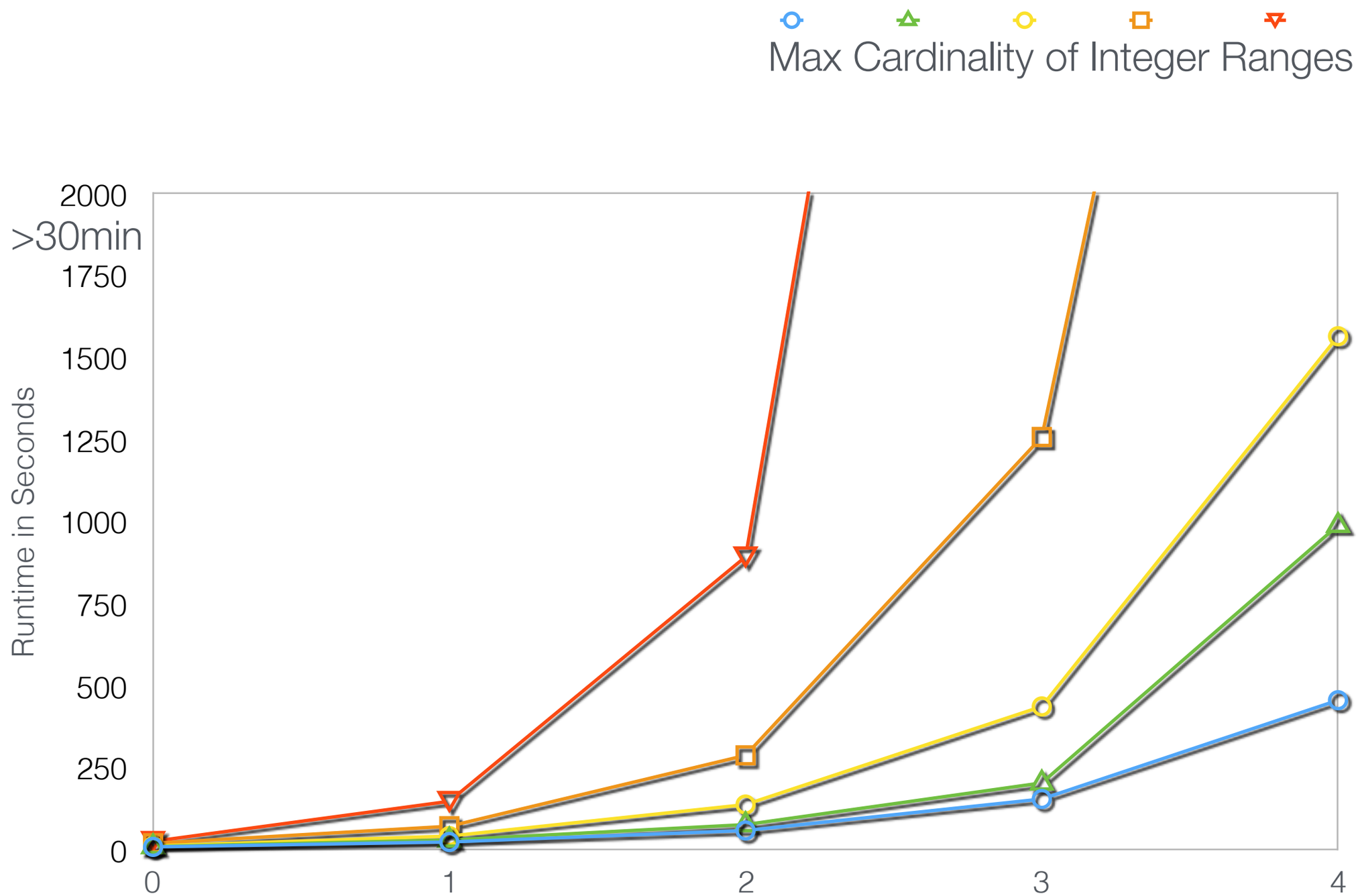
Study: JDK 8 Update 25

- Found 556 issues
- For 19 we found no source code
- 279 of 537 were considered irrelevant
- The remaining 258 issues were manually inspected

Category	Percentage
Null Confusion	54 %
Range Double Checks	11 %
Dead Extensibility	9 %
Unsupported Operation	7 %
Unexpected Return	5 %
Forgotten Constant	4 %
Confused Language	3 %
Type Confusion	3 %
Confused Conjunctions	2 %
Obviously Useless	1 %
False Positives	1 %

The effect of varying the maximum cardinality of integer ranges or the maximum call chain length on the number of identified issues..





The effect on the runtime if we vary the max cardinality/call chain length.