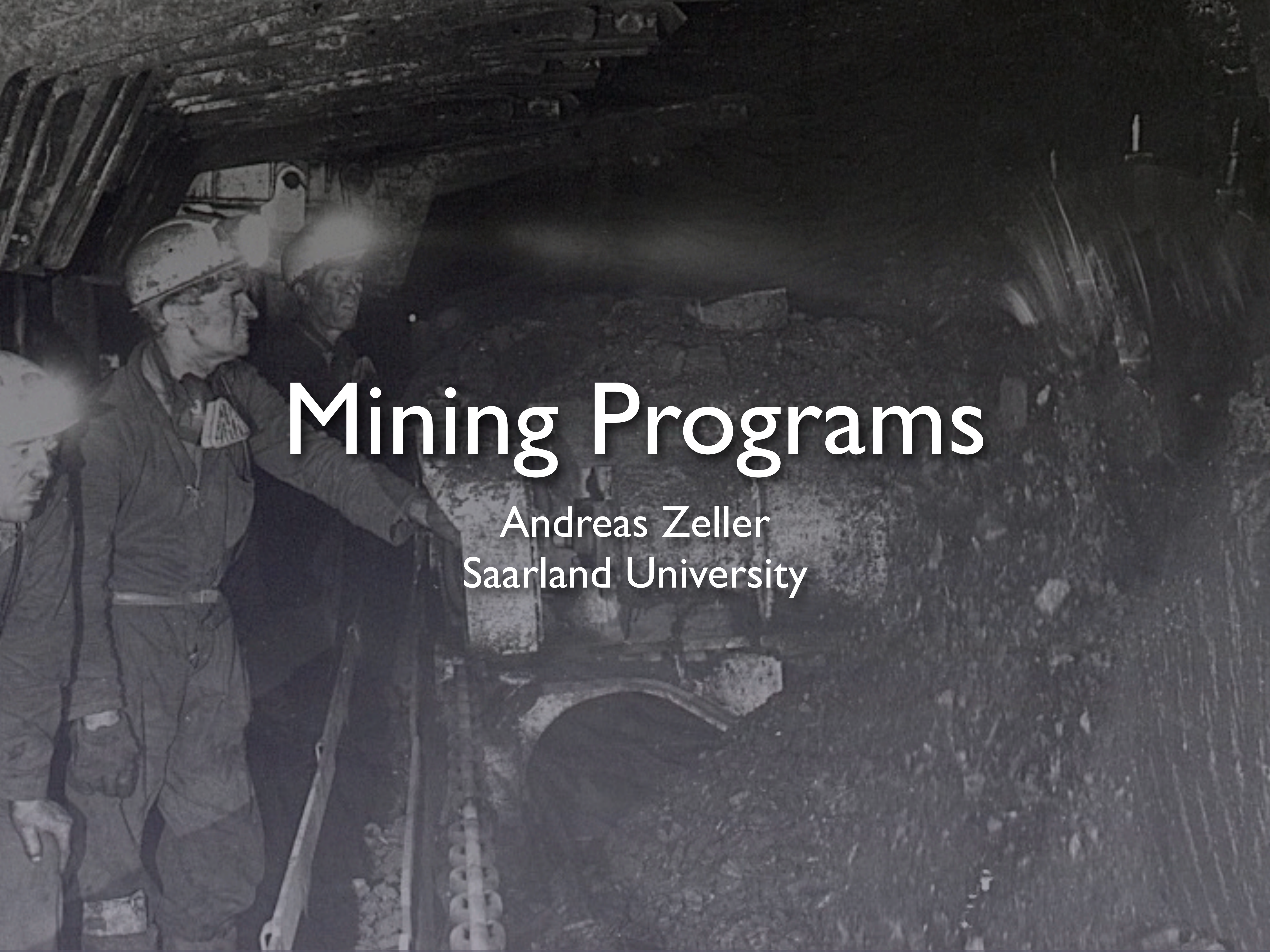


Mining Programs

Andreas Zeller
Saarland University


Saarbrücken

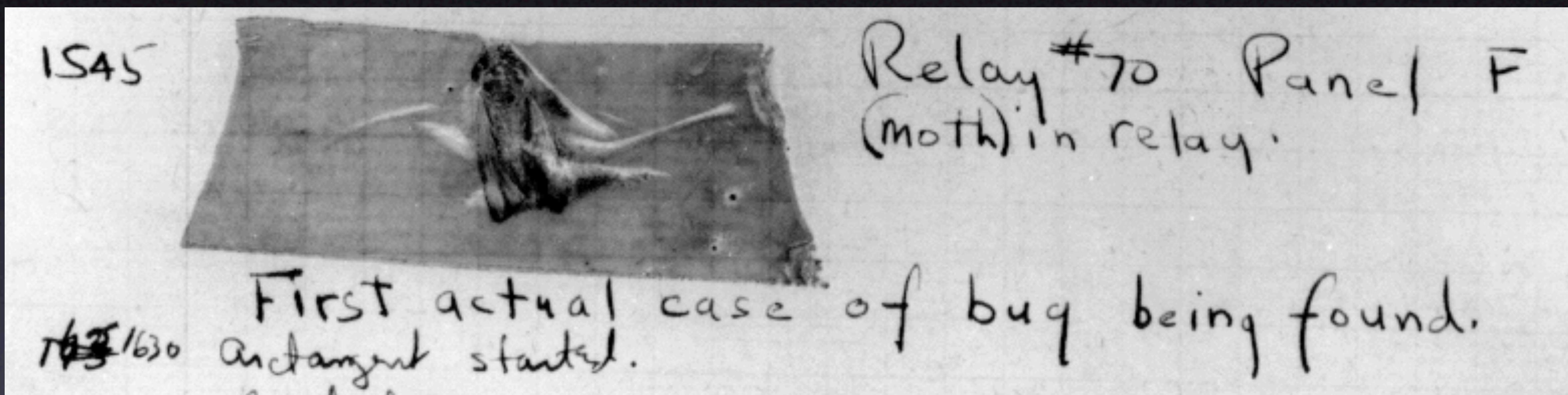
Saarbrücken

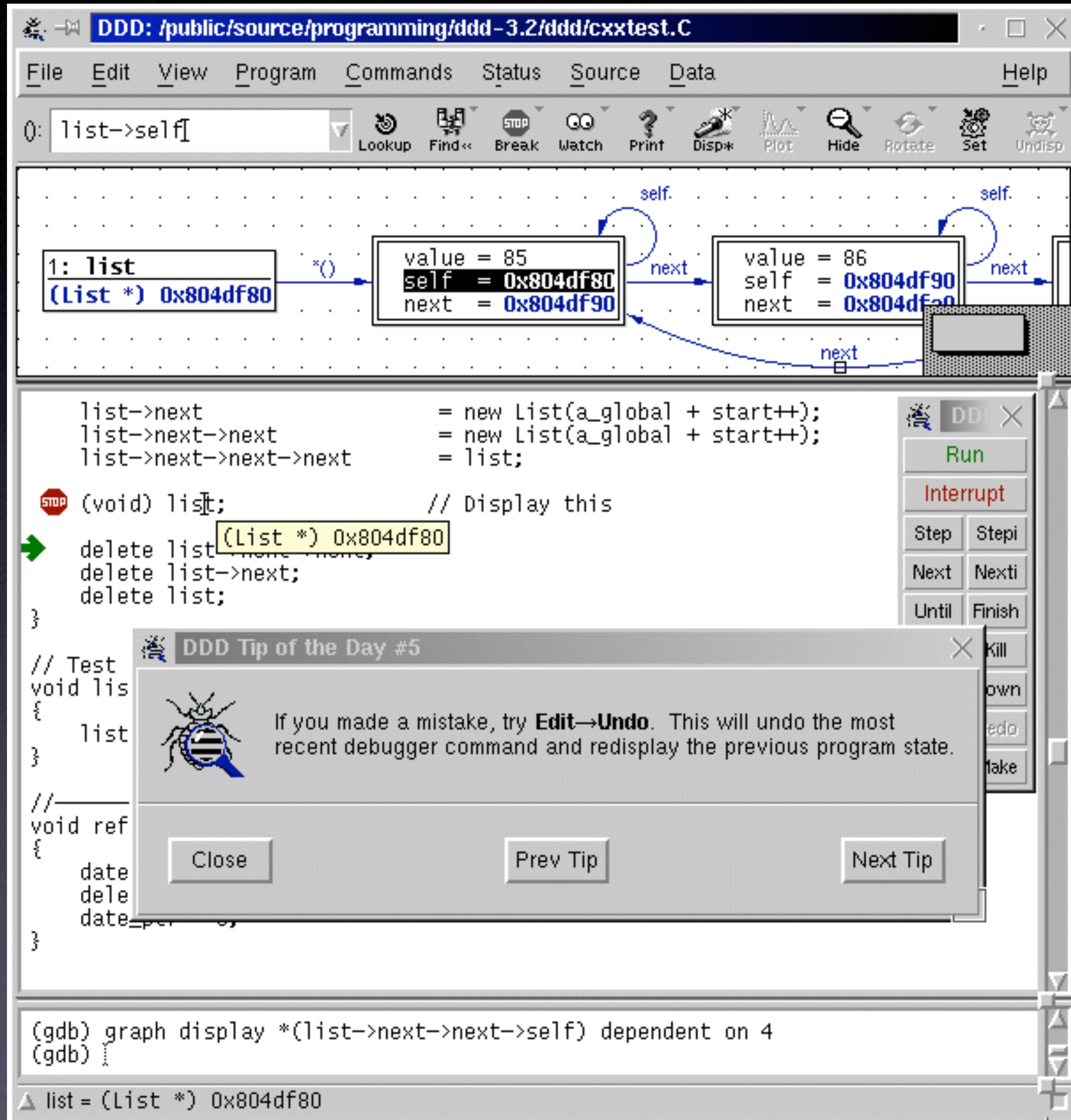
Saarbrücken

Mining

The First Bug

(September 9, 1947)

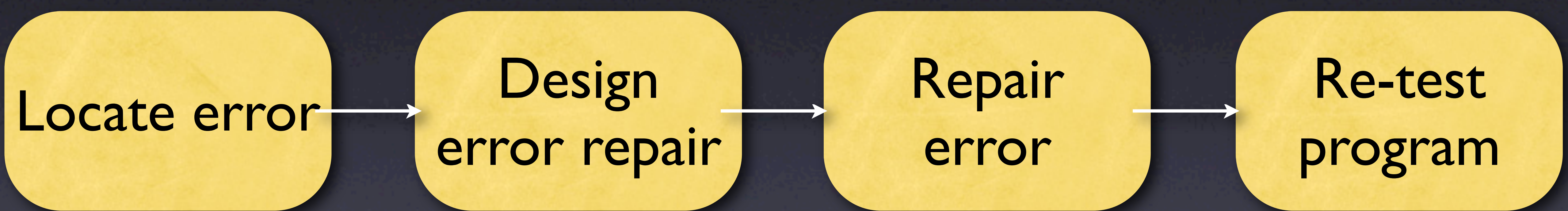


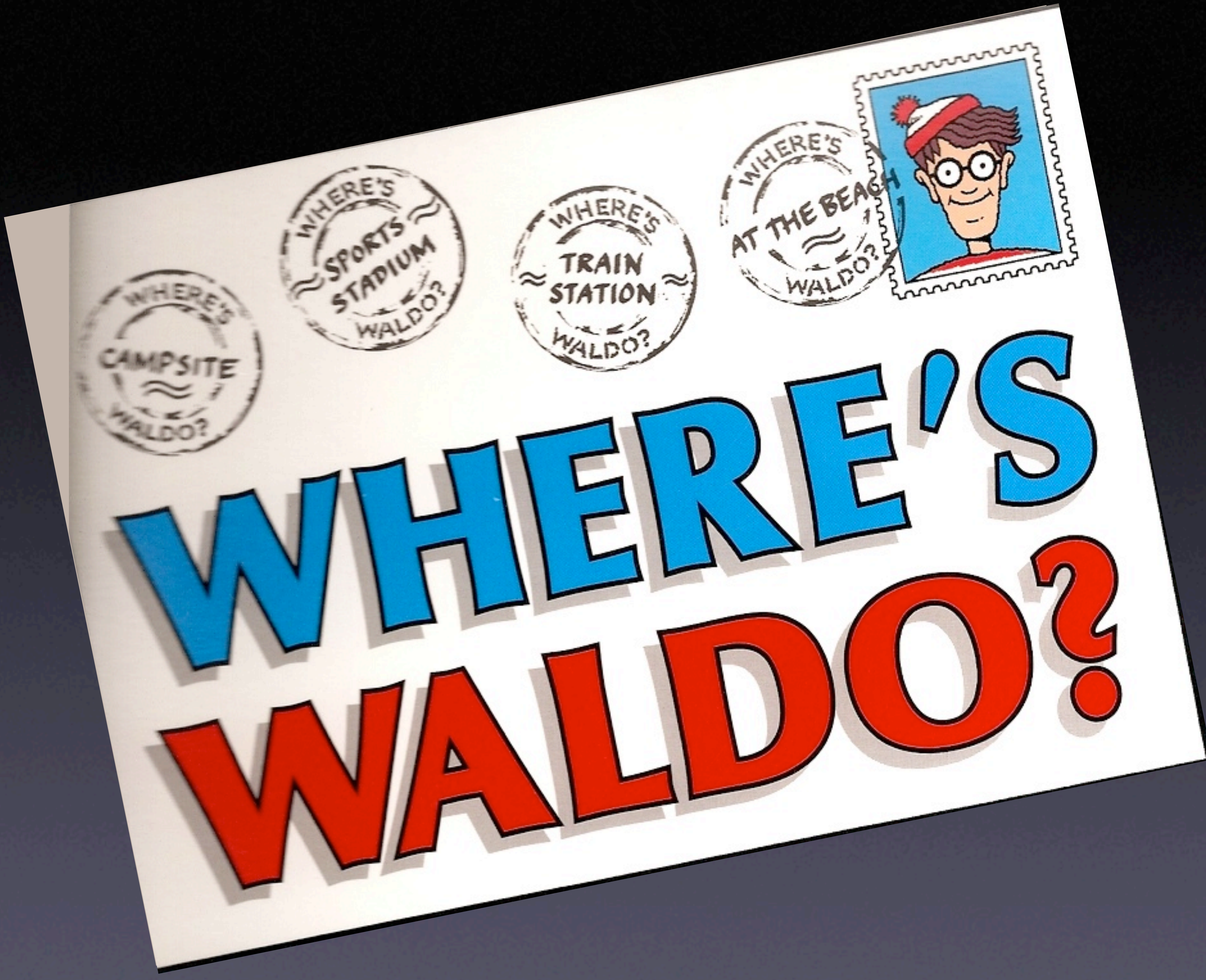


DDD: /public/source/programming/ddd-3.2/ddd/cxxtest.C
File Edit View Program Commands Status Source Data Help
(): list->self
Lookup Find« Break Watch Print Disp* Plot Hide Rotate Set Undisp
1: list
(List *) 0x804df80
*()
value = 85
self = 0x804df80
next = 0x804df90
self.
next
value = 86
self = 0x804df90
next
list->next = new List(a_global + start++);
list->next->next = new List(a_global + start++);
list->next->next->next = list;
(void) list; // Display this
(List *) 0x804df80
delete list
delete list->next;
delete list;
}
// Test
void lis
{
list
}
void ref
{
date
dele
}
Run
Interrupt
Step Stepi
Next Nexti
Until Finish
DDD Tip of the Day #5
If you made a mistake, try Edit→Undo. This will undo the most recent debugger command and redisplay the previous program state.
Close
Prev Tip
Next Tip
(gdb) graph display *(list->next->next->self) dependent on 4
(gdb)
list = (List *) 0x804df80

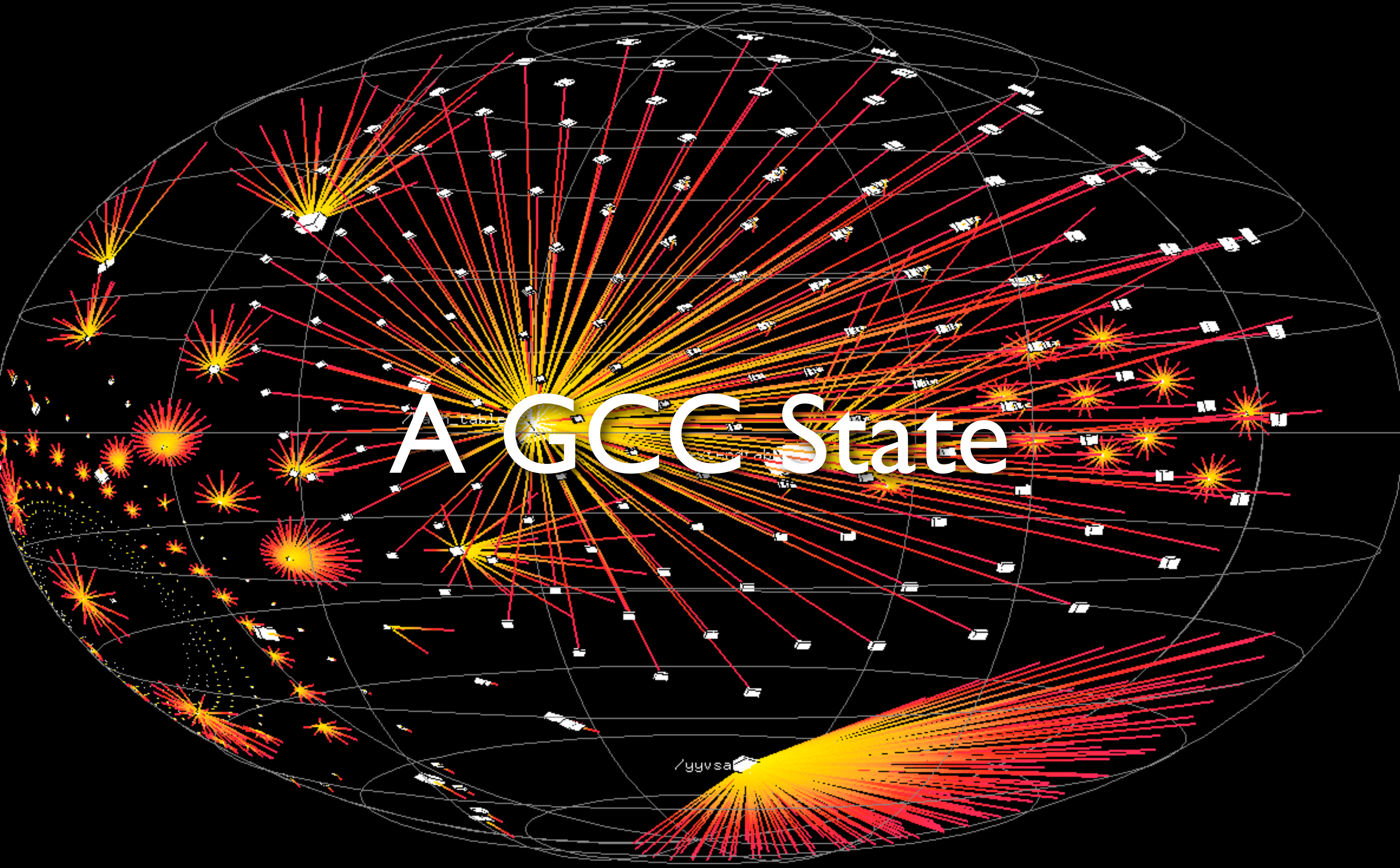
How to Debug

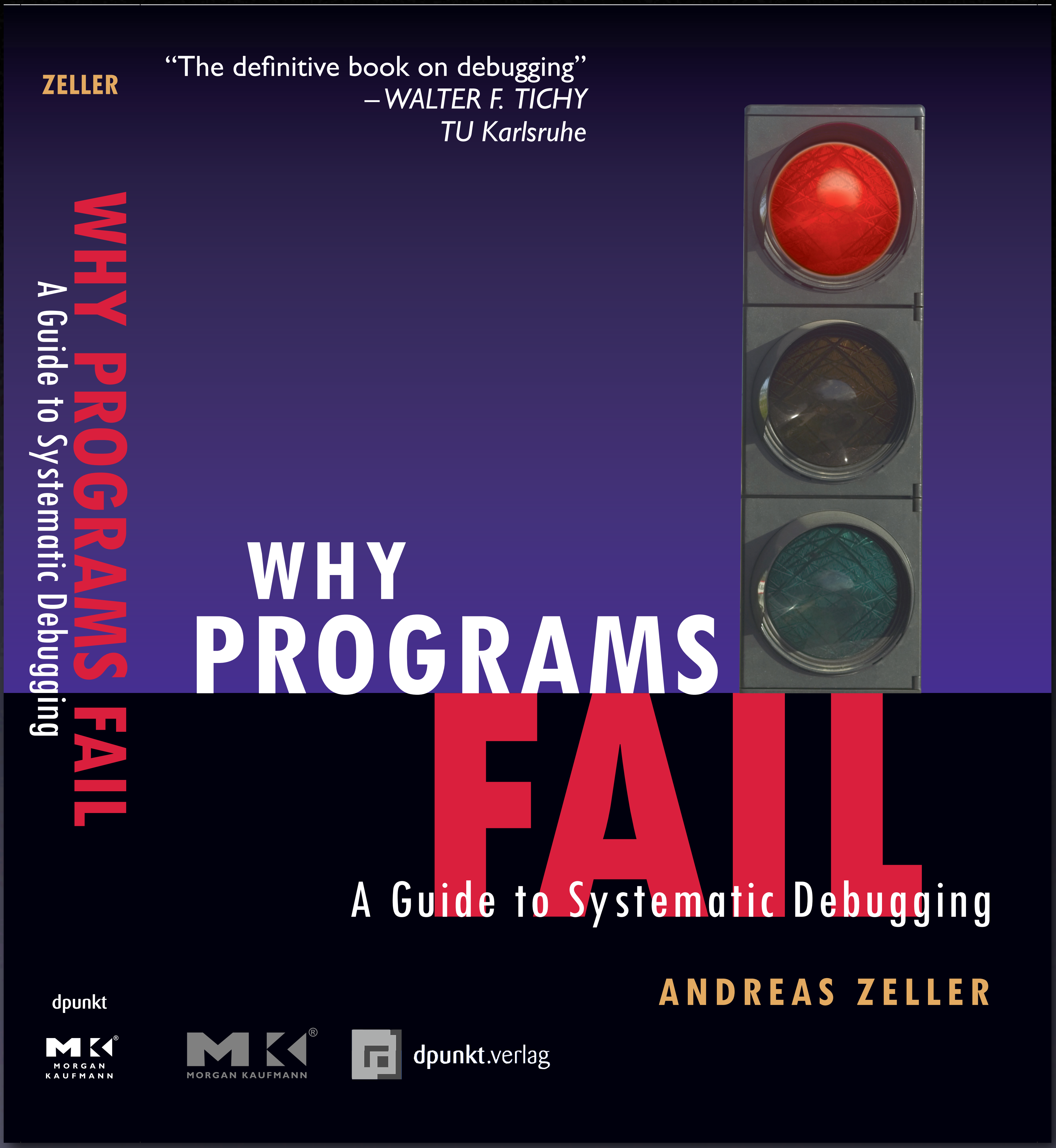
(Sommerville 2004)



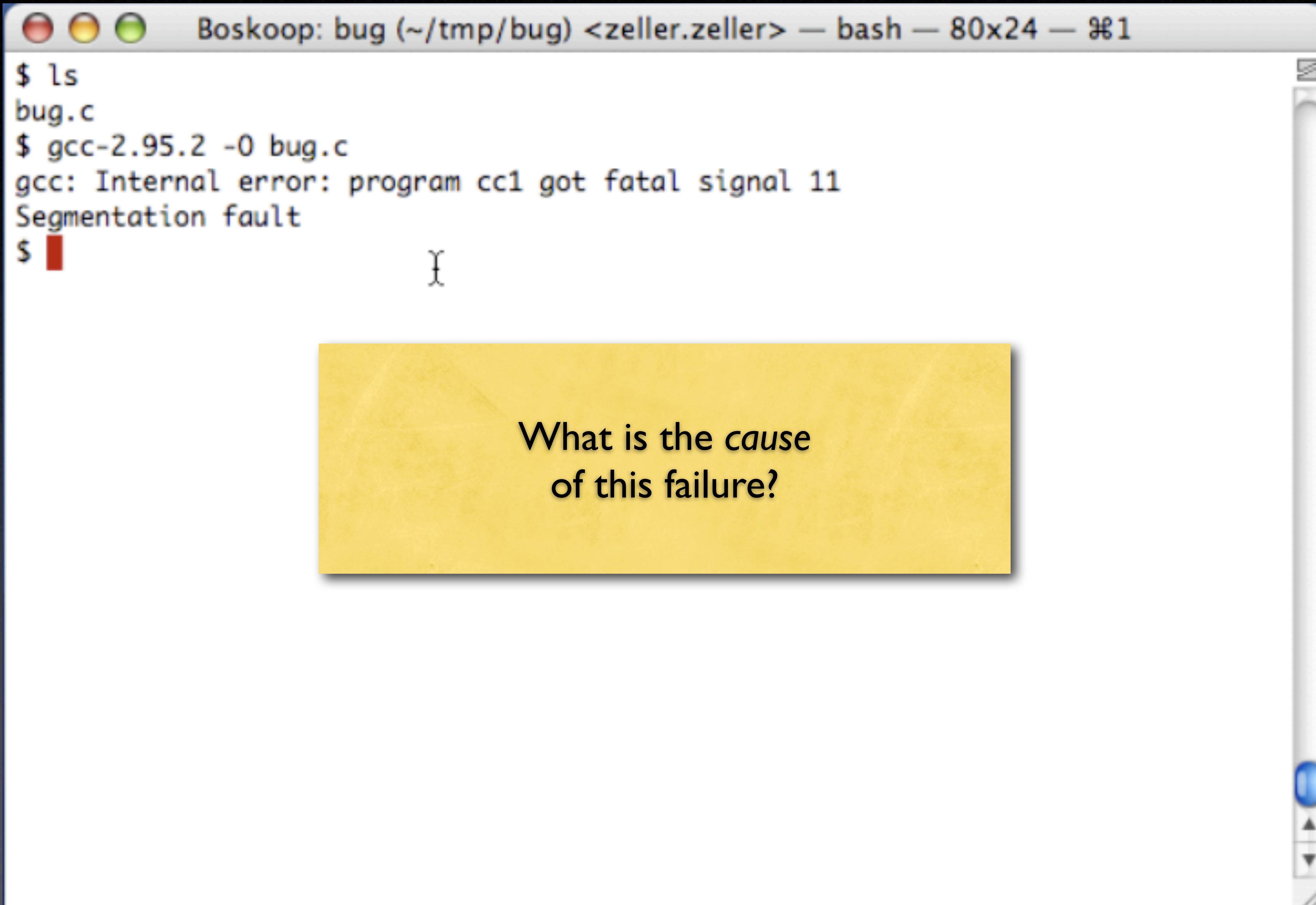

WHERE'S CAMPSITE WALDO?
WHERE'S SPORTS STADIUM WALDO?
WHERE'S TRAIN STATION WALDO?
WHERE'S AT THE BEACH WALDO?
WHERE'S
WALDO?

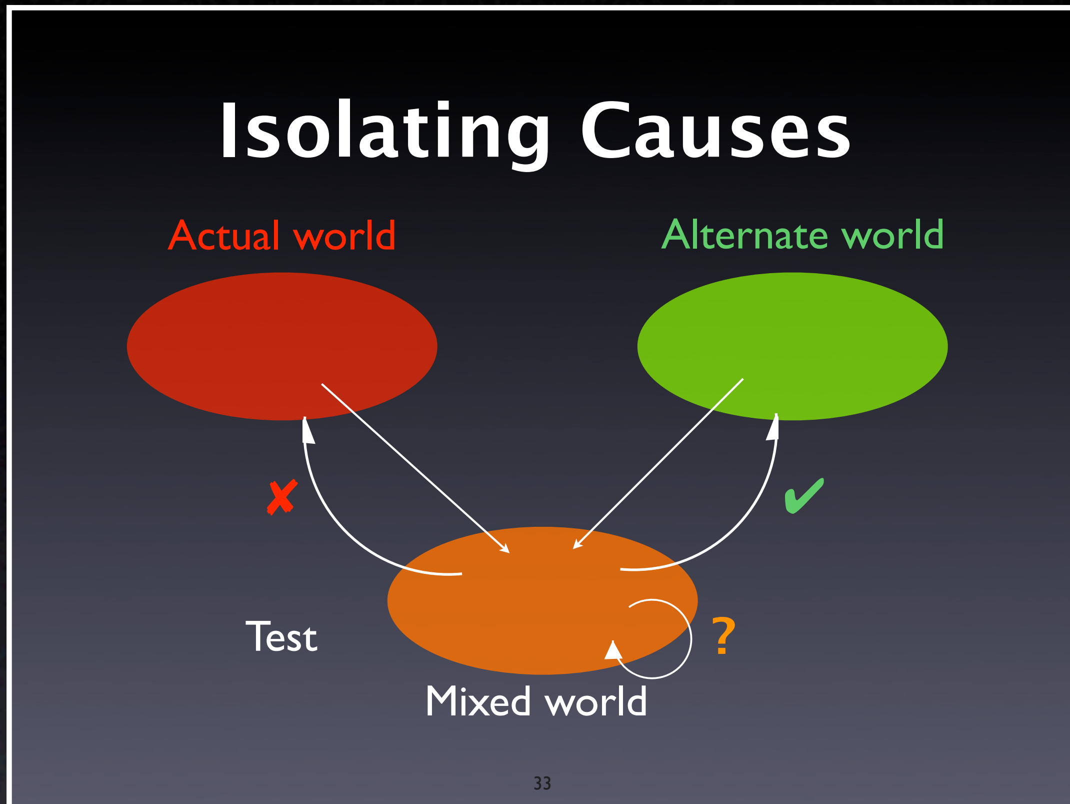
A GCC State

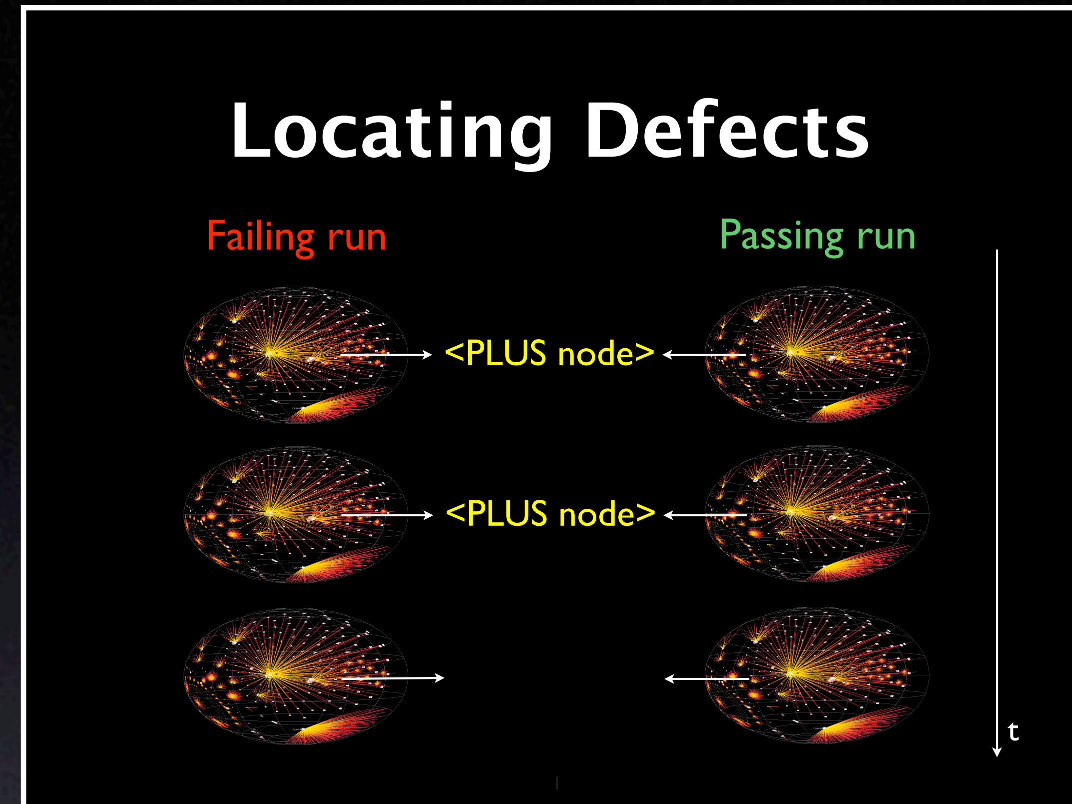


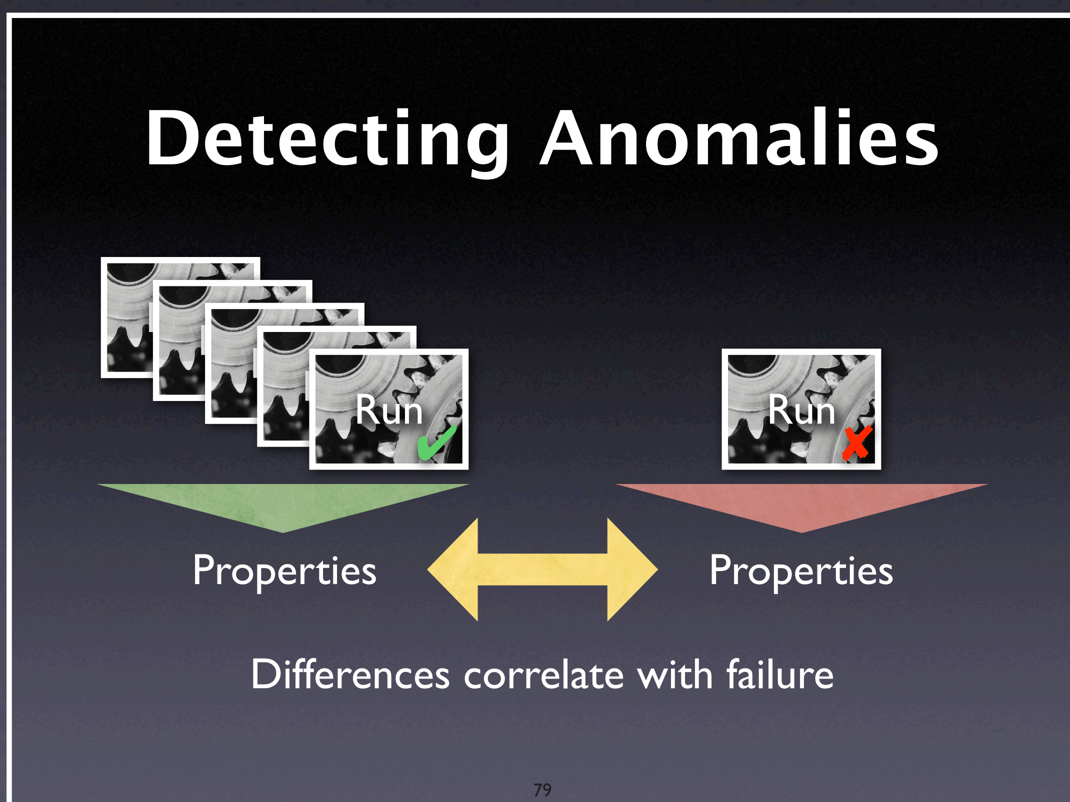


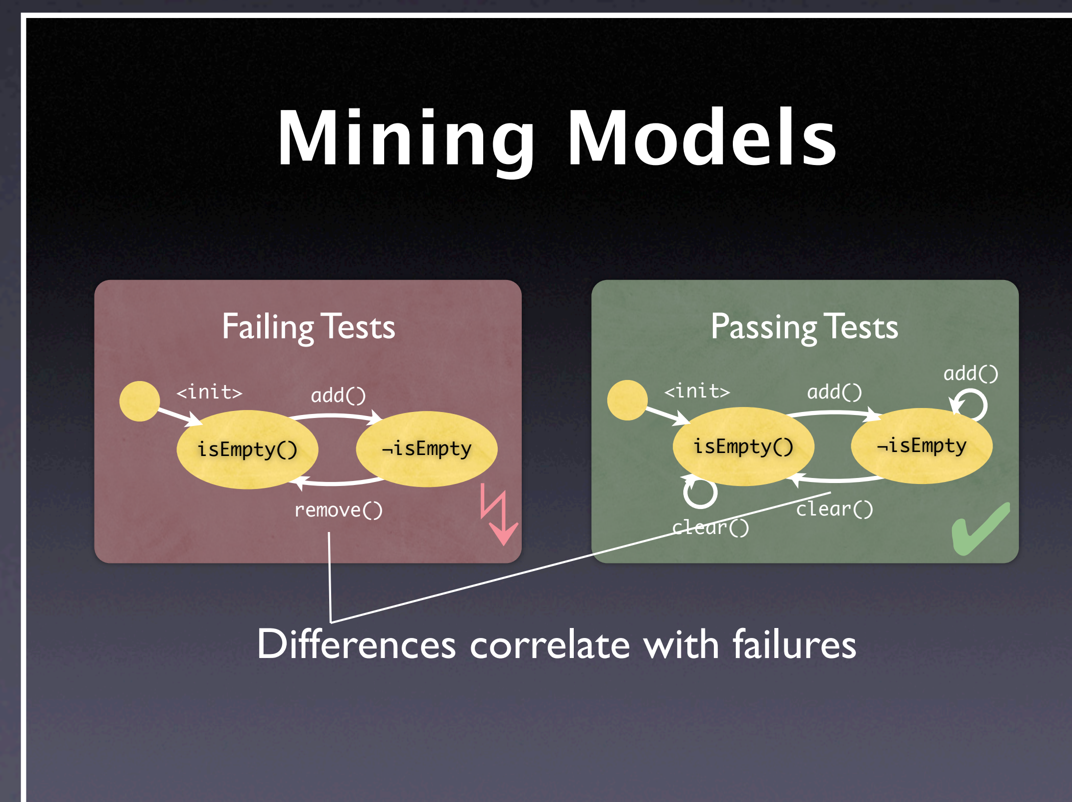
ZELLER
WHY PROGRAMS FAIL
A Guide to Systematic Debugging
dpunkt
MORGAN KAUFMANN
"The definitive book on debugging"
– WALTER F. TICHY
TU Karlsruhe
WHY
PROGRAMS
FAIL
A Guide to Systematic Debugging
ANDREAS ZELLER
MORGAN KAUFMANN
dpunkt.verlag

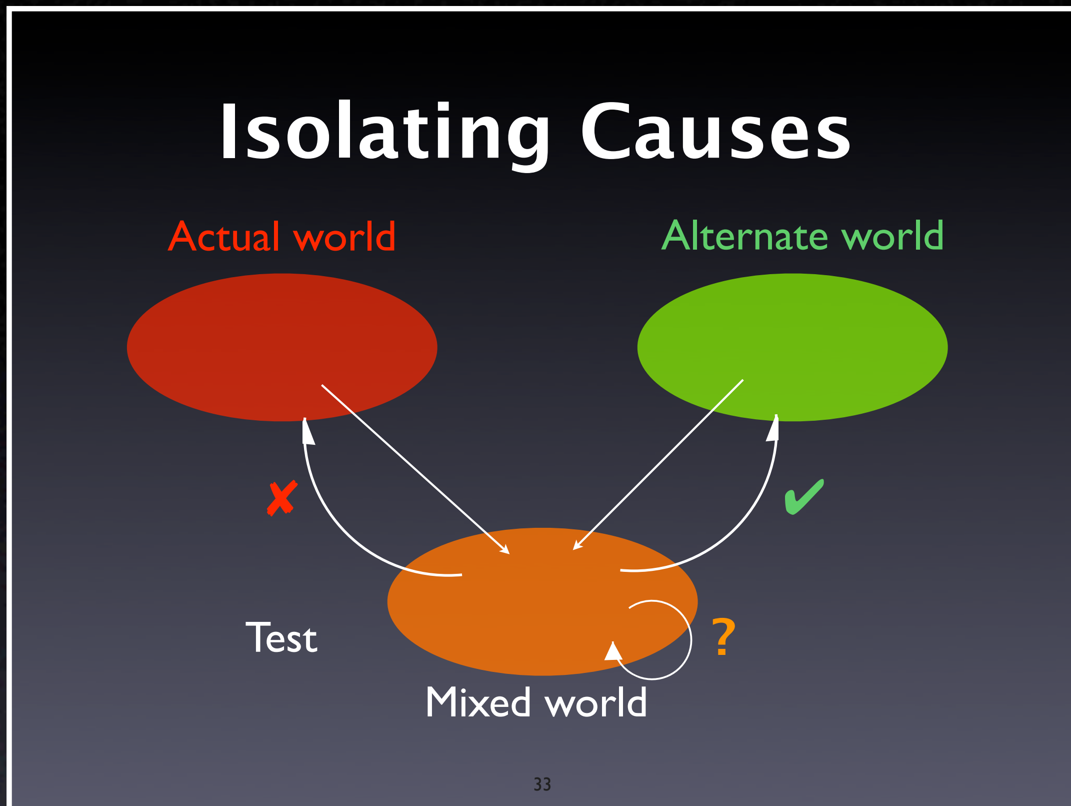

Boskoop: bug (~/tmp/bug) <zeller.zeller> — bash — 80x24 — ⌘1
$ ls
bug.c
$ gcc-2.95.2 -O bug.c
gcc: Internal error: program cc1 got fatal signal 11
Segmentation fault
$
What is the *cause* of this failure?


Isolating Causes
Actual world
Alternate world
✘
✔
Test
?
Mixed world
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t

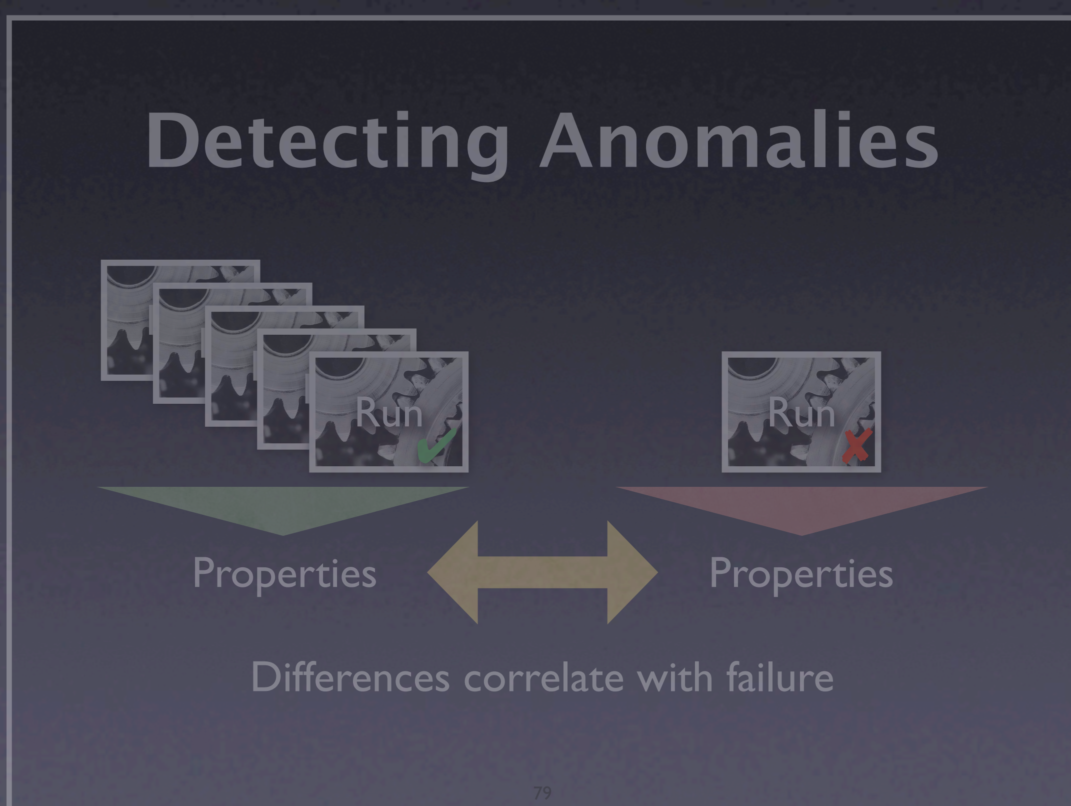

Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79


Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures

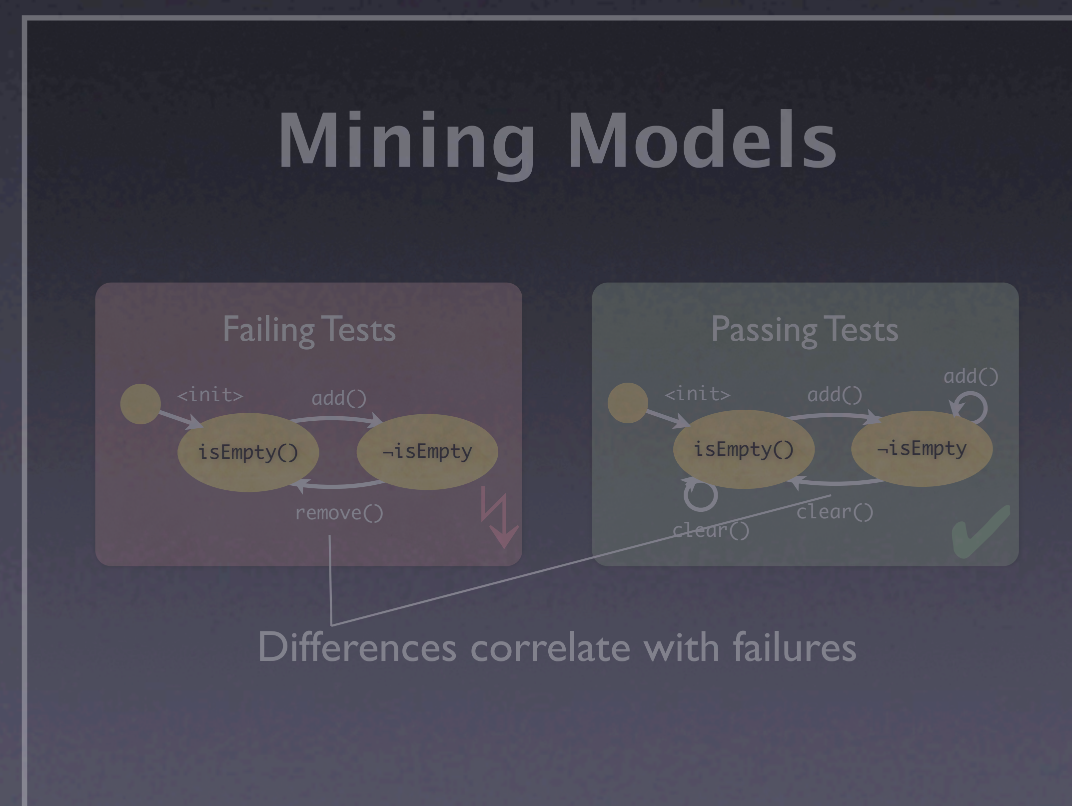


Isolating Causes
Actual world
Alternate world
✘
✔
Test
?
Mixed world
33

Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t



Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure



Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures

bug.c

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

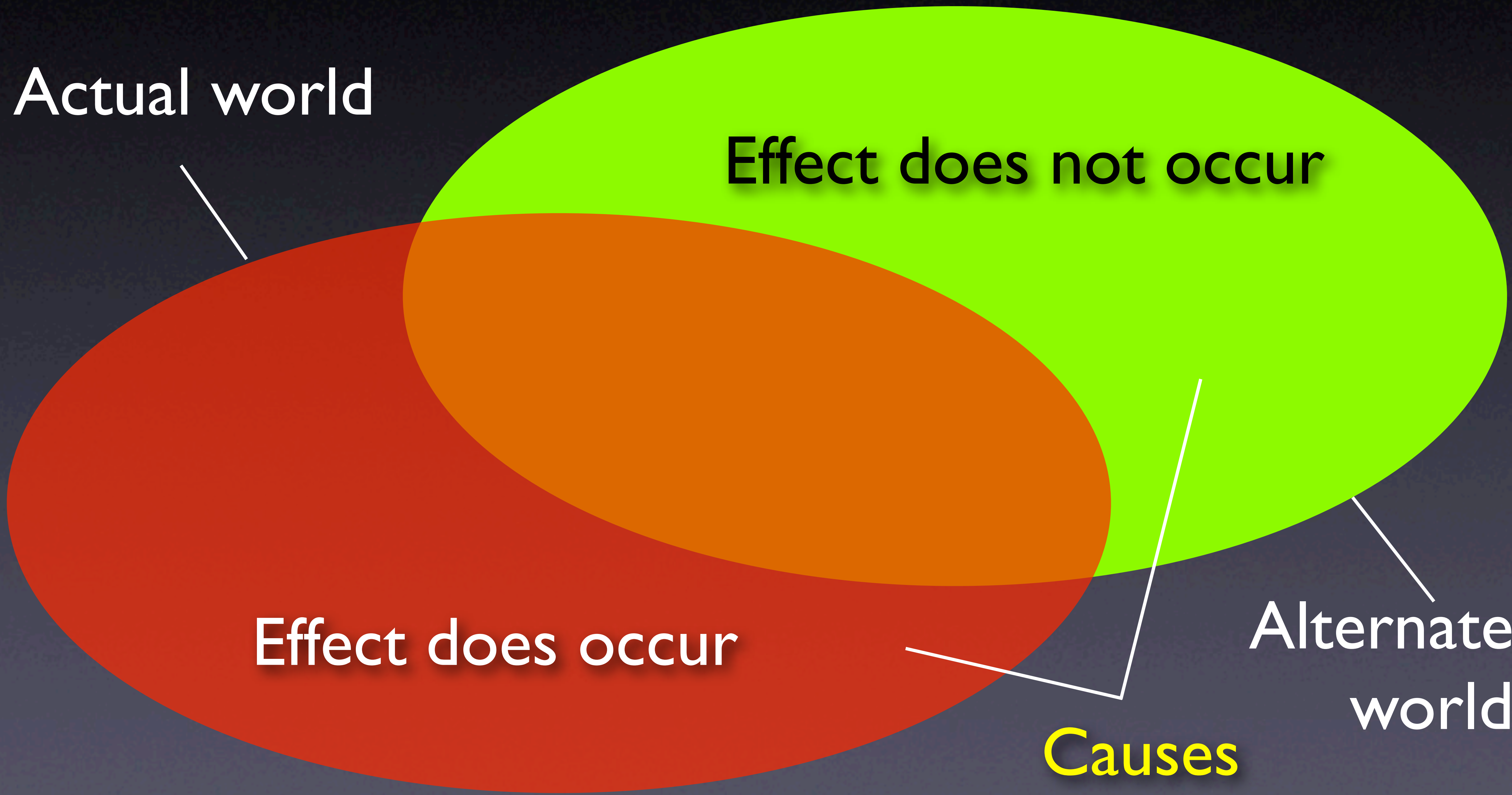
What is the cause of this failure?

Causality

What does it mean that "A causes B"?

- Counterfactual approach: If A had not occurred, then B would not have occured
- Statistical approach: A results in an increase in the probability of B

Counterfactual Causality



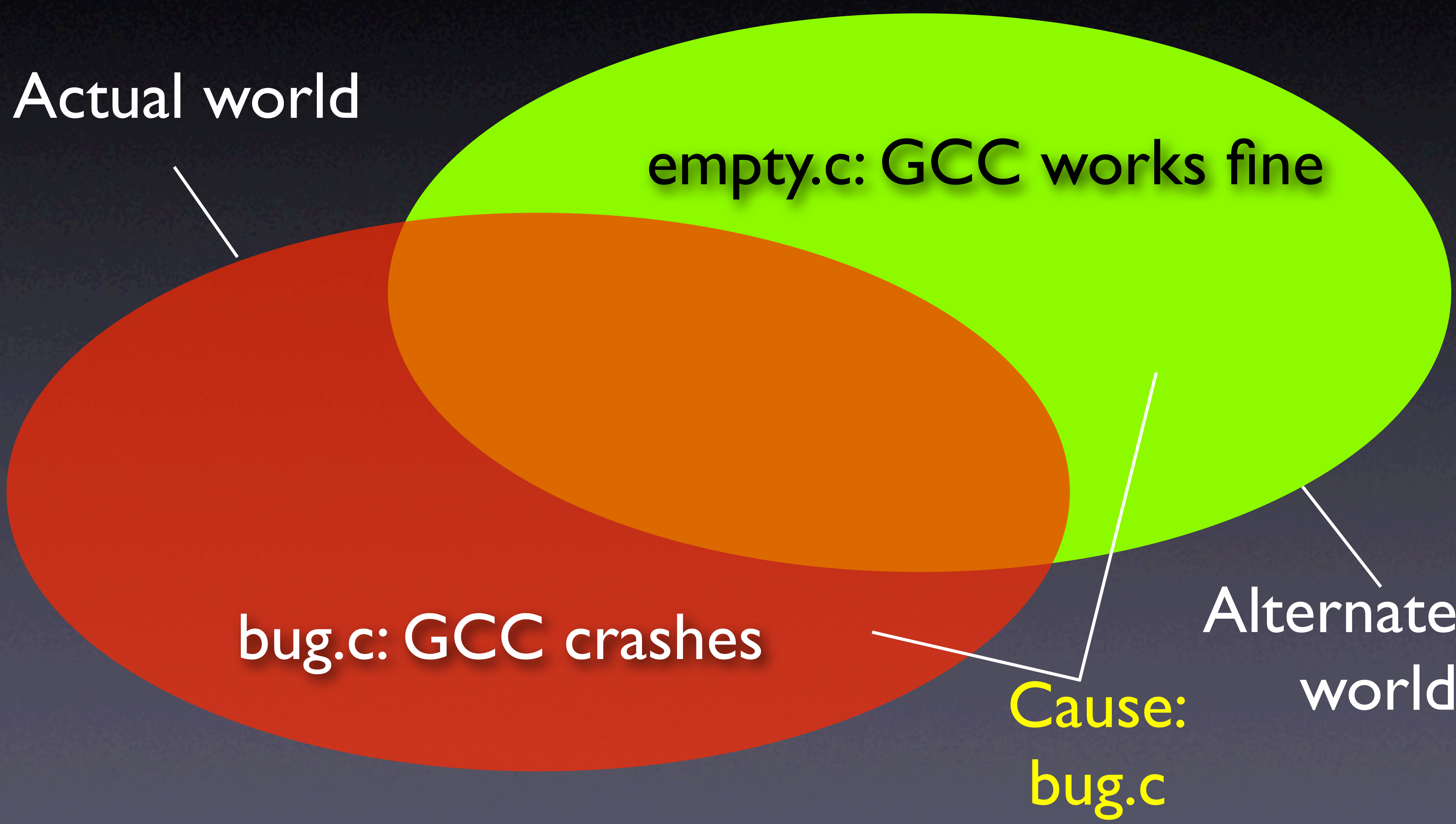
bug.c

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

empty.c

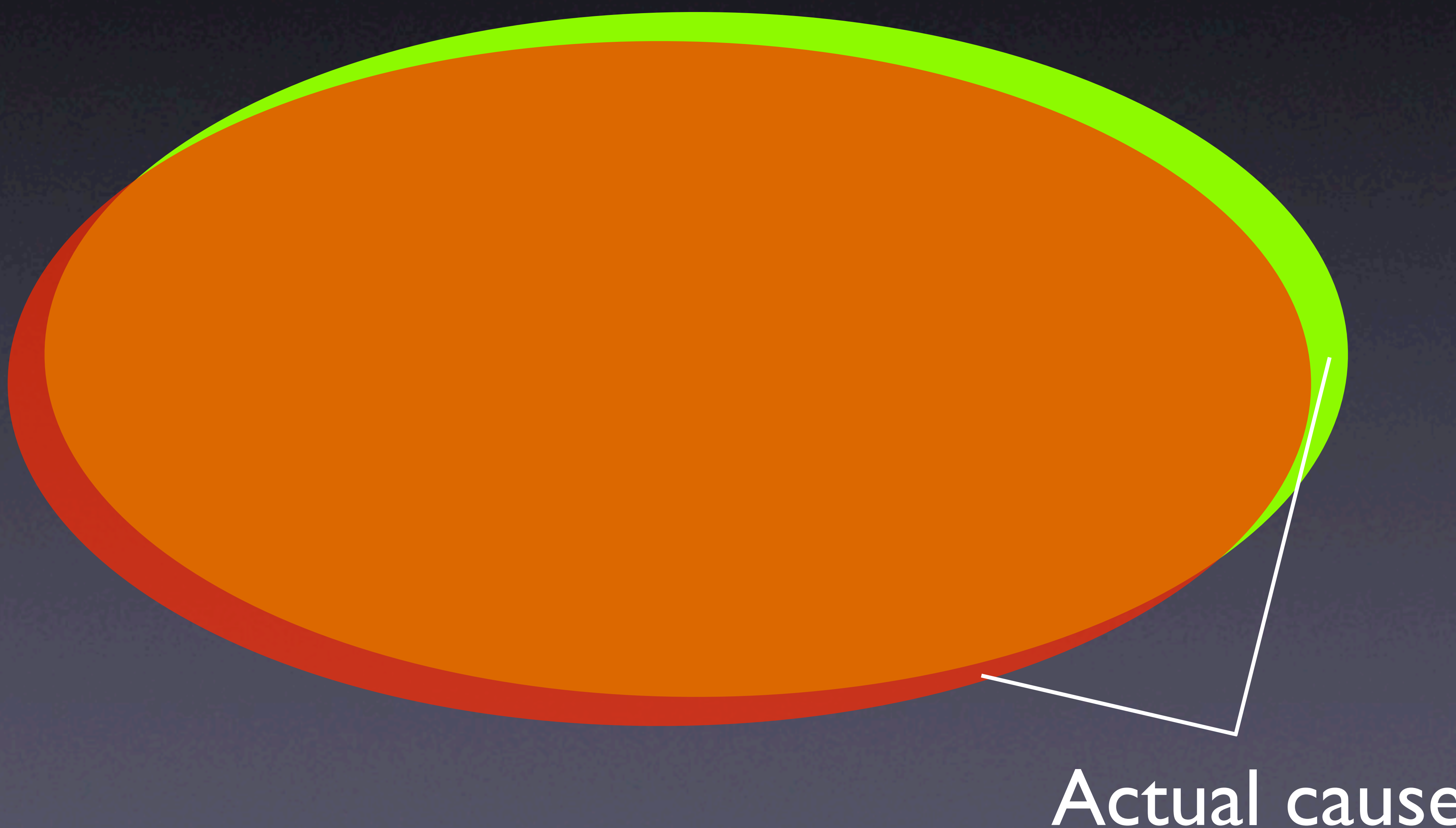
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Causes as Differences



Actual Causes

"The" cause (*actual cause*) is a *minimal difference*



Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Actual cause narrowed down

Isolating Causes

```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

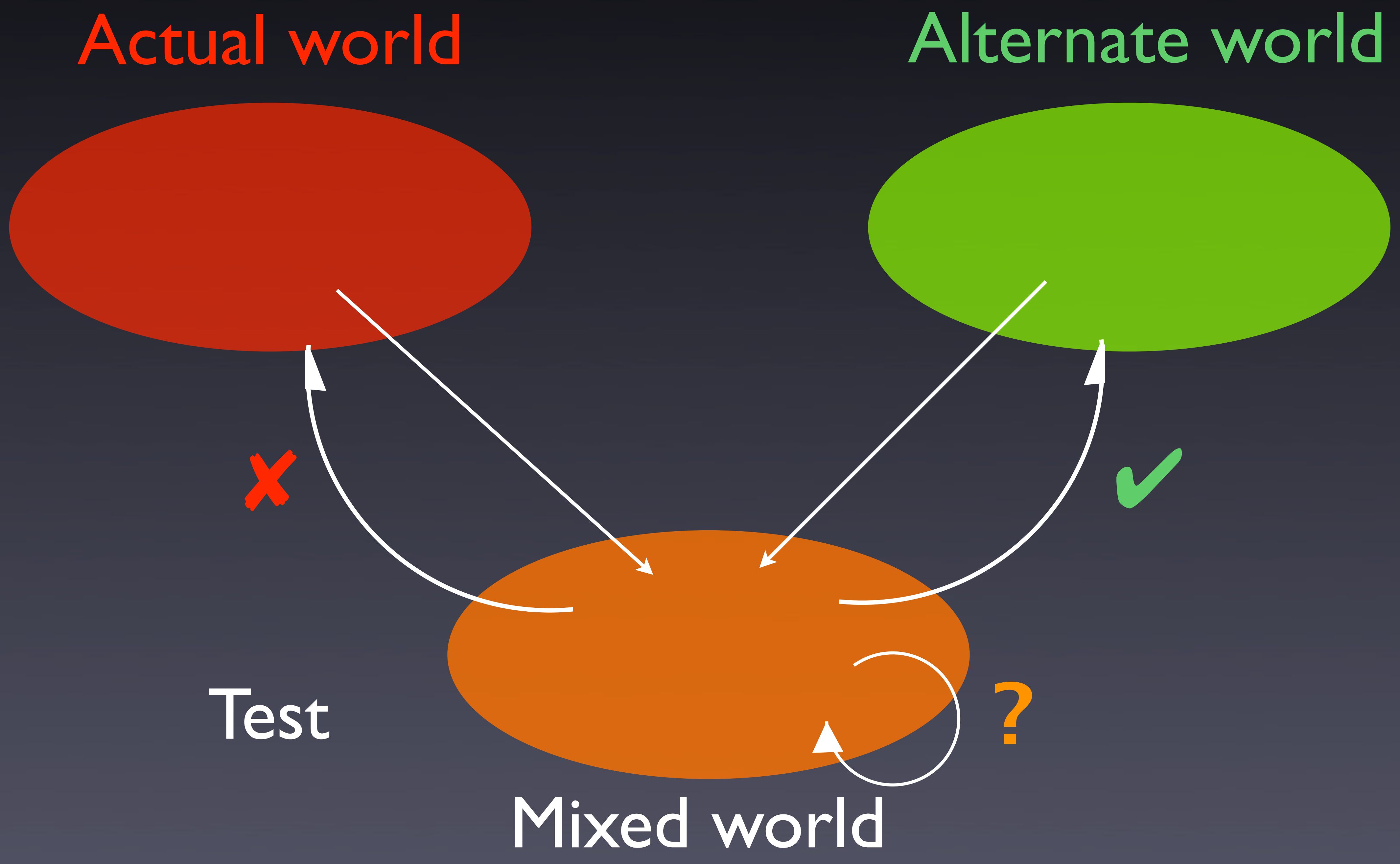
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Isolating Causes

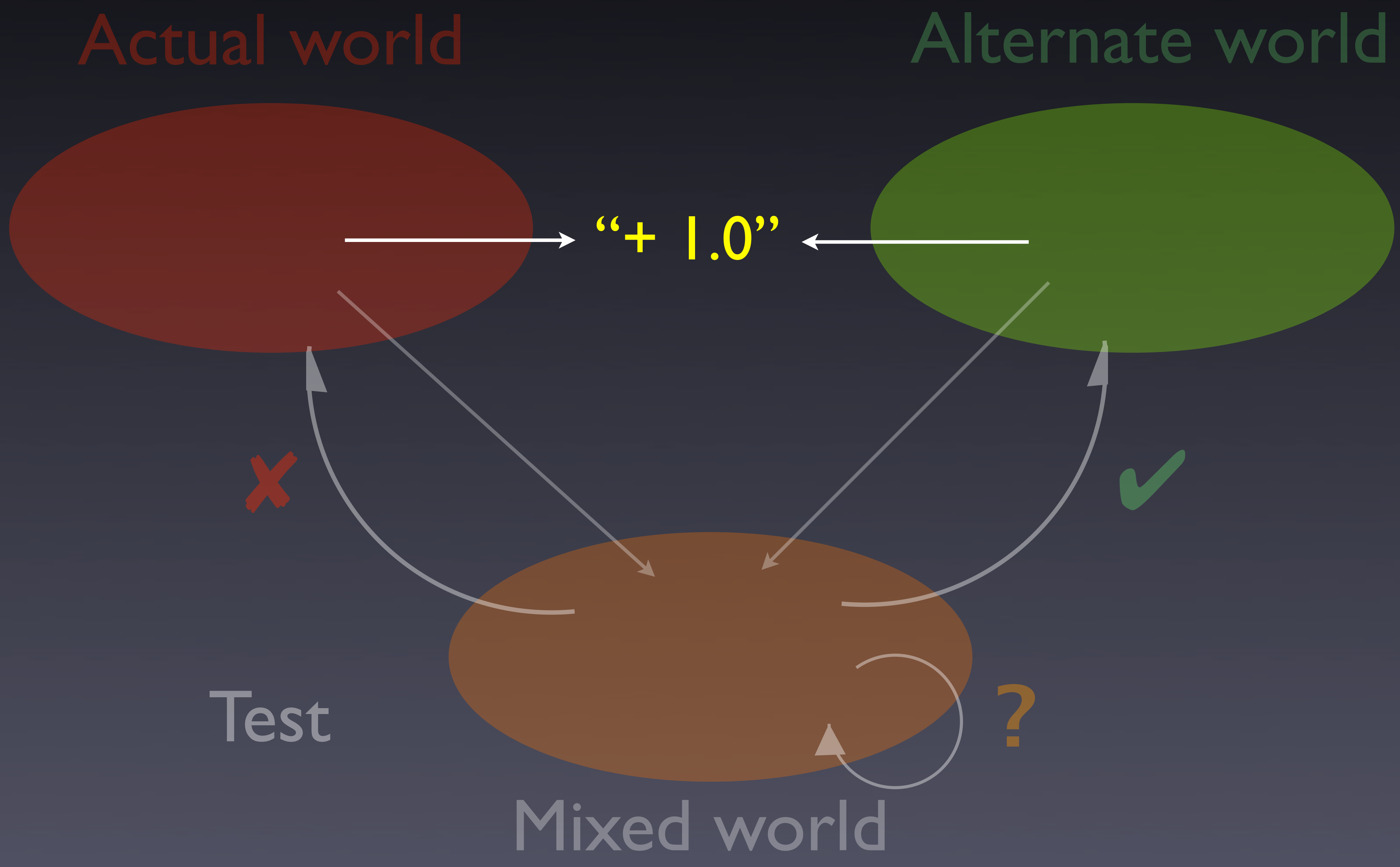
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Actual cause of the GCC crash

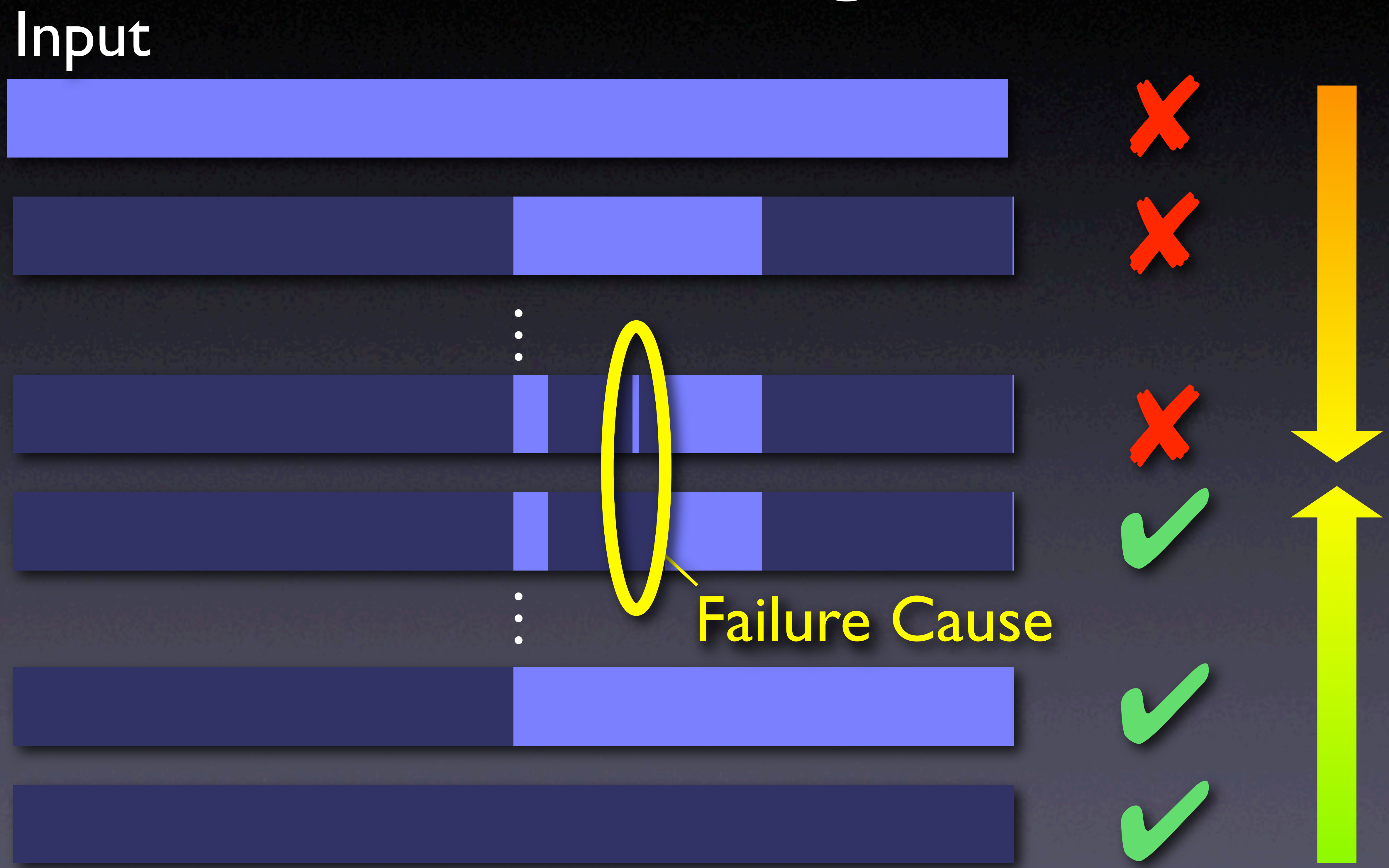
Isolating Causes



Isolating Causes



Isolating



Configuration

Circumstance

$$\delta$$

All circumstances

$$C = \{\delta_1, \delta_2, \ldots\}$$

Configuration $c \subseteq C$

$$c = \{\delta_1, \delta_2, \ldots \delta_n\}$$

Tests

Testing function

$$test(c) \in \{✔, ✘, ?\}$$

Initial configurations

$$test(c_{✔}) = ✔$$
$$test(c_{✘}) = ✘$$

Minimal Difference

Goal: Subsets $c'_{\text{✘}}$ and $c'_{\text{✔}}$

$$\emptyset = c_{\text{✔}} \subseteq c'_{\text{✔}} \subset c'_{\text{✘}} \subseteq c_{\text{✘}}$$

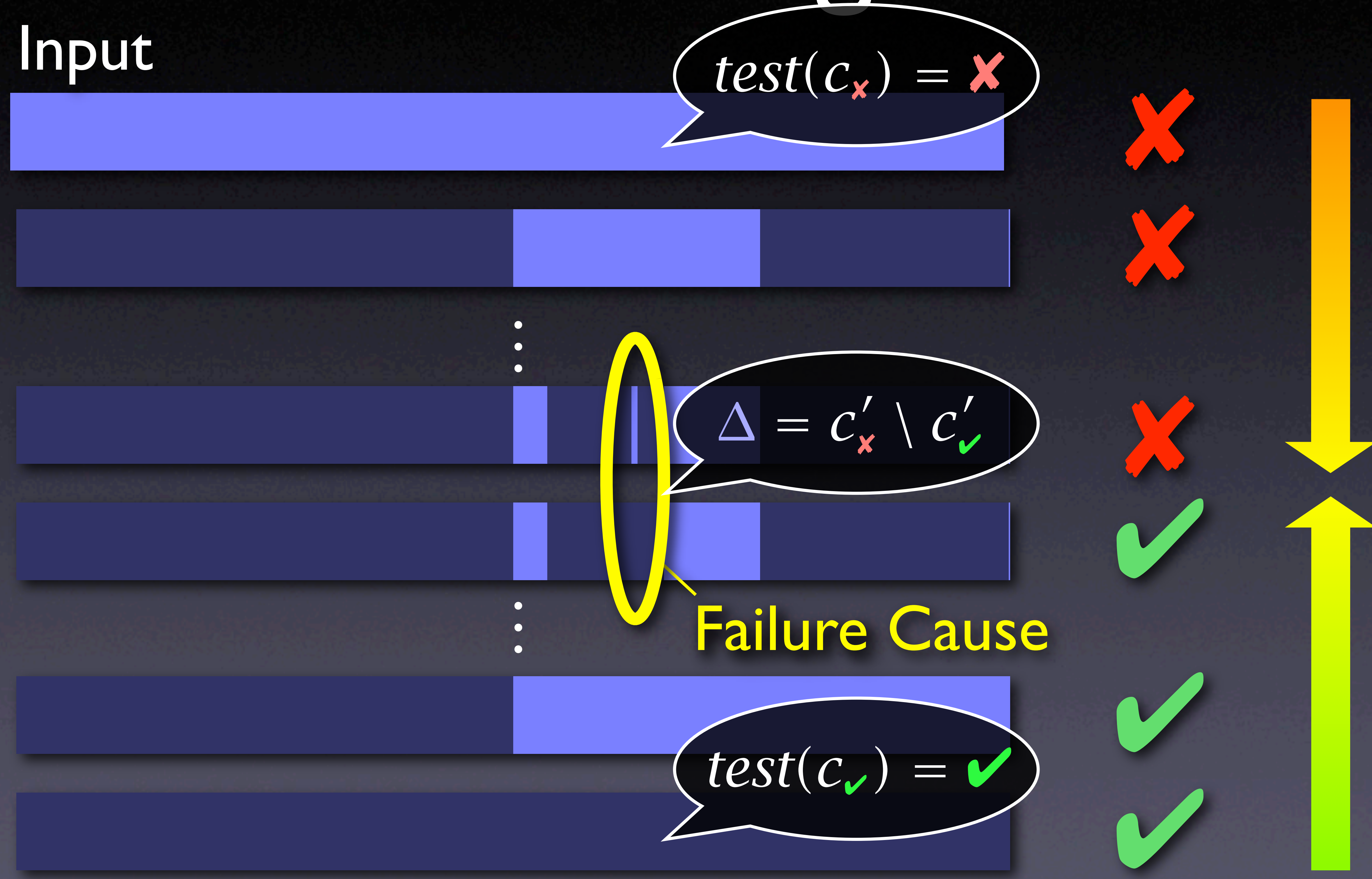
Difference

$$\Delta = c'_{\text{✘}} \setminus c'_{\text{✔}}$$

Difference is 1-minimal

$$\forall \delta_i \in \Delta \cdot test(c'_{\text{✔}} \cup \{\delta_i\}) \neq \text{✔} \wedge test(c'_{\text{✘}} \setminus \{\delta_i\}) \neq \text{✘}$$

Isolating



Algorithm Sketch

- Minimize the *difference* between two sets – with working sets $c'_{\text{✘}}$ and $c'_{\checkmark}$
- Compute subsets

$$\Delta_1 \cup \Delta_2 \cup \cdots \cup \Delta_n = \Delta = c'_{\text{✘}} \setminus c'_{\checkmark}$$

- For each subset, test
  - the *addition* $c'_{\checkmark} \cup \Delta_i$
  - the *removal* $c'_{\text{✘}} \setminus \Delta_i$

Test Outcomes

	✘	✔
$test(c'_{\text{✘}} \setminus \Delta_i)$	$c'_{\text{✘}} := c'_{\text{✘}} \setminus \Delta_i$	$c'_{\text{✔}} := c'_{\text{✘}} \setminus \Delta_i$
$test(c'_{\text{✔}} \cup \Delta_i)$	$c'_{\text{✘}} := c'_{\text{✔}} \cup \Delta_i$	$c'_{\text{✔}} := c'_{\text{✔}} \cup \Delta_i$
otherwise	increase granularity	

most valuable outcomes

dd in a Nutshell

$$dd(c_{\checkmark}, c_{\times}) = (c'_{\checkmark}, c'_{\times}) \quad \Delta = c'_{\times} \setminus c'_{\checkmark} \text{ is 1-minimal}$$

$$dd(c_{\checkmark}, c_{\times}) = dd'(c_{\checkmark}, c_{\times}, 2)$$

$$dd'(c'_{\checkmark}, c'_{\times}, n) =$$

$$\begin{cases}
(c'_{\checkmark}, c'_{\times}) & \text{if } |\Delta| = 1 \\
dd'(c'_{\times} \setminus \Delta_i, c'_{\times}, 2) & \text{if } \exists i \in \{1..n\} \cdot \mathit{test}(c'_{\times} \setminus \Delta_i) = \checkmark \\
dd'(c'_{\checkmark}, c'_{\checkmark} \cup \Delta_i, 2) & \text{if } \exists i \in \{1..n\} \cdot \mathit{test}(c'_{\checkmark} \cup \Delta_i) = \times \\
dd'(c'_{\checkmark} \cup \Delta_i, c'_{\times}, \max(n-1, 2)) & \text{else if } \exists i \in \{1..n\} \cdot \mathit{test}(c'_{\checkmark} \cup \Delta_i) = \checkmark \\
dd'(c'_{\checkmark}, c'_{\times} \setminus \Delta_i, \max(n-1, 2)) & \text{else if } \exists i \in \{1..n\} \cdot \mathit{test}(c'_{\times} \setminus \Delta_i) = \times \\
dd'(c'_{\checkmark}, c'_{\times}, \min(2n, |\Delta|)) & \text{else if } n < |\Delta| \text{ (“increase granularity”)} \\
(c'_{\checkmark}, c'_{\times}) & \text{otherwise}
\end{cases}$$

```python
def dd(c_pass, c_fail):
    n = 2
    while 1:
        delta = listminus(c_fail, c_pass)
        deltas = split(delta, n); offset = 0; j = 0
        while j < n:
            i = (j + offset) % n
            next_c_pass = listunion(c_pass, deltas[i])
            next_c_fail = listminus(c_fail, deltas[i])
            if test(next_c_fail) == FAIL and n == 2:
                c_fail = next_c_fail; n = 2; offset = 0; break
            elif test(next_c_fail) == PASS:
                c_pass = next_c_fail; n = 2; offset = 0; break
            elif test(next_c_pass) == FAIL:
                c_fail = next_c_pass; n = 2; offset = 0; break
            elif test(next_c_fail) == FAIL:
                c_fail = next_c_fail; n = max(n - 1, 2); offset = i; break
            elif test(next_c_pass) == PASS:
                c_pass = next_c_pass; n = max(n - 1, 2); offset = i; break
            else:
                j = j + 1
        if j >= n:
            if n >= len(delta):
                return (delta, c_pass, c_fail)
            else:
                n = min(len(delta), n * 2)
```

Properties

number of tests t – worst case:

$$t = |\Delta|^2 + 7|\Delta| \quad \text{where} \quad \Delta = c_{\times} \setminus c_{\checkmark}$$

number of tests t – best case
(no unresolved outcomes):

$$t \le \log_2(\Delta)$$

size of difference – no unresolved outcomes

$$|c'_{\times} \setminus c'_{\checkmark}| = 1$$

Applications

Input	Code Changes	Schedules

Mozilla Bug #24735

Ok the following operations cause mozilla to crash consistently on my machine

-> Start mozilla
-> Go to bugzilla.mozilla.org
-> Select search for bug
-> Print to file setting the bottom and right margins to .50
   (I use the file /var/tmp/netscape.ps)
-> Once it's done printing do the exact same thing again on
   the same file (/var/tmp/netscape.ps)
-> This causes the browser to crash with a segfault

Isolating Input

<SELECT NAME="priority" MULTIPLE SIZE=7> ✘

Difference narrowed down

<SELECT NAME="priority" MULTIPLE SIZE=7> ✔

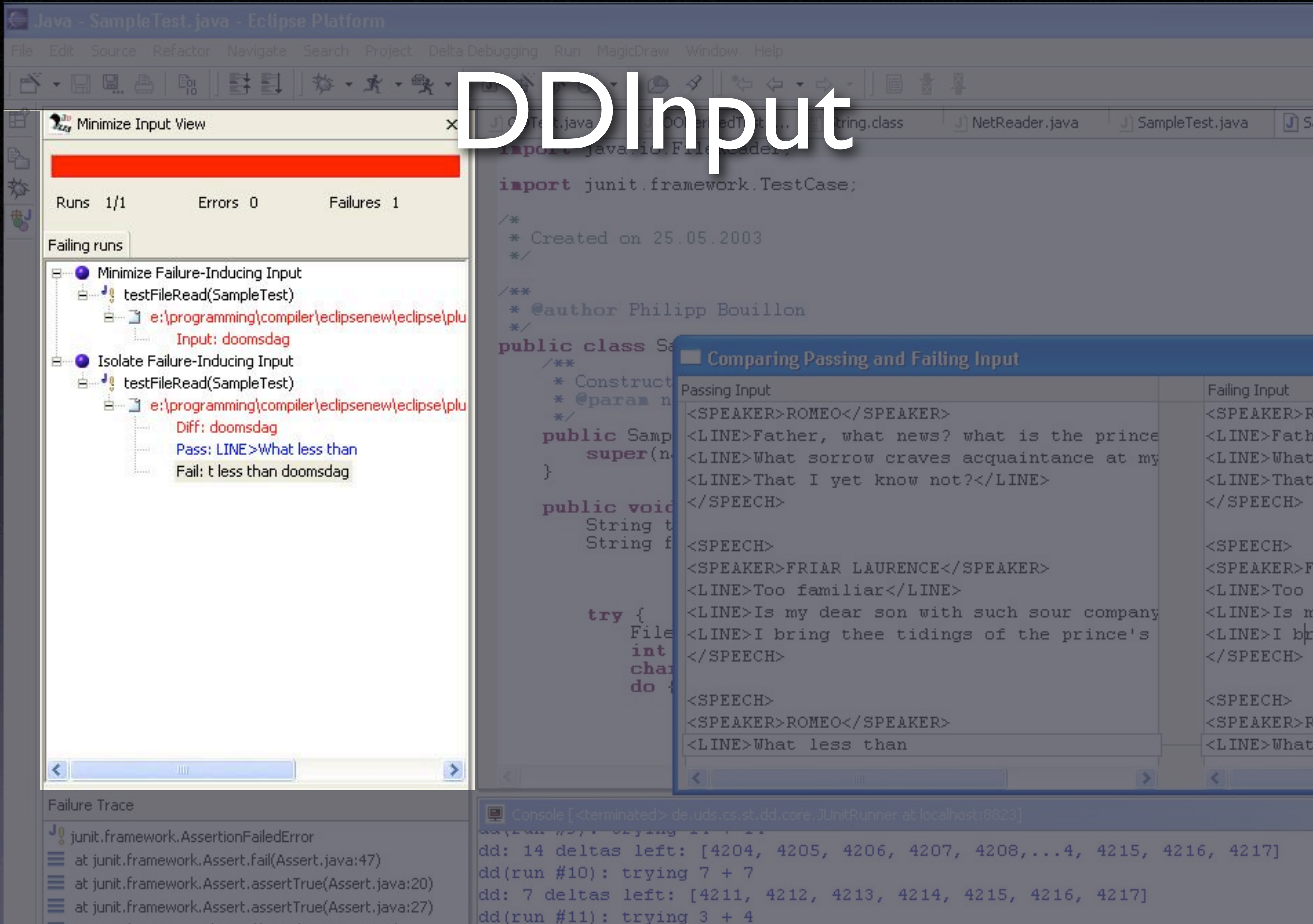
<SELECT NAME="priority" MULTIPLE SIZE=7> ✔

Isolating Input

```
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
```

Failure Cause

DDInput

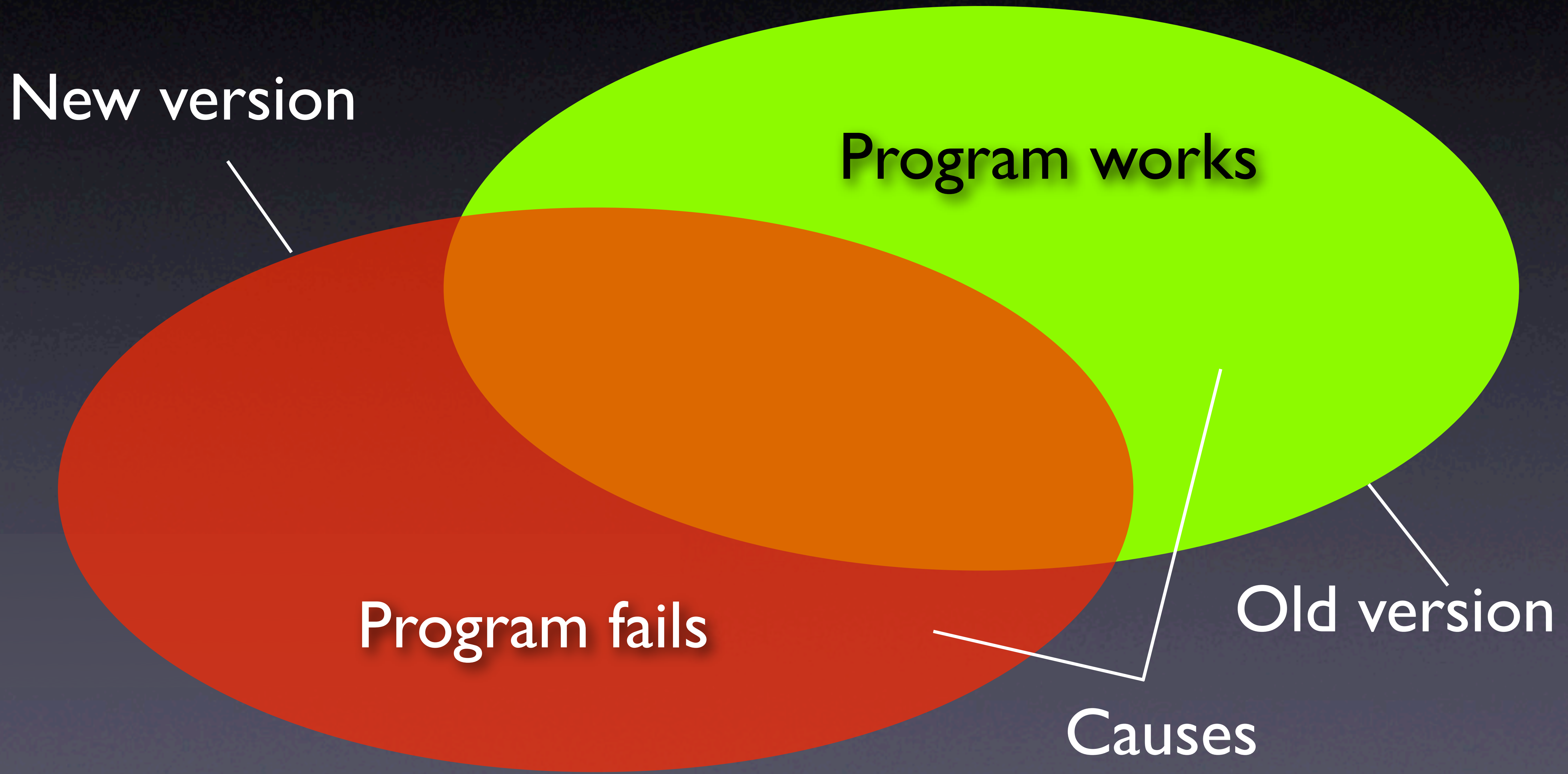

Java - SampleTest.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Delta Debugging Run MagicDraw Window Help
Minimize Input View
Runs 1/1
Errors 0
Failures 1
Failing runs
Minimize Failure-Inducing Input
testFileRead(SampleTest)
e:\programming\compiler\eclipsenew\eclipse\plu
Input: doomsdag
Isolate Failure-Inducing Input
testFileRead(SampleTest)
e:\programming\compiler\eclipsenew\eclipse\plu
Diff: doomsdag
Pass: LINE>What less than
Fail: t less than doomsdag
Failure Trace
junit.framework.AssertionFailedError
at junit.framework.Assert.fail(Assert.java:47)
at junit.framework.Assert.assertTrue(Assert.java:20)
at junit.framework.Assert.assertTrue(Assert.java:27)
NetReader.java
SampleTest.java
import junit.framework.TestCase;
* Created on 25.05.2003
* @author Philipp Bouillon
public class
try {
Comparing Passing and Failing Input
Passing Input
Failing Input
<SPEAKER>ROMEO</SPEAKER>
<LINE>Father, what news? what is the prince
<LINE>What sorrow craves acquaintance at my
<LINE>That I yet know not?</LINE>
</SPEECH>
<SPEECH>
<SPEAKER>FRIAR LAURENCE</SPEAKER>
<LINE>Too familiar</LINE>
<LINE>Is my dear son with such sour company
<LINE>I bring thee tidings of the prince's
</SPEECH>
<SPEECH>
<SPEAKER>ROMEO</SPEAKER>
<LINE>What less than
dd: 14 deltas left: [4204, 4205, 4206, 4207, 4208,...4, 4215, 4216, 4217]
dd(run #10): trying 7 + 7
dd: 7 deltas left: [4211, 4212, 4213, 4214, 4215, 4216, 4217]
dd(run #11): trying 3 + 4

Code Changes

From: Brian Kahne <bkahne@ibmoto.com>
To: DDD Bug Report Address <bug-ddd@gnu.org>
Subject: Problem with DDD and GDB 4.17

When using DDD with GDB 4.16, the run command correctly uses any prior command-line arguments, or the value of "set args". However, when I switched to GDB 4.17, this no longer worked: If I entered a run command in the console window, the prior command-line options would be lost. [...]

Version Differences



What was Changed

```
$ diff -r gdb-4.16 gdb-4.17
diff -r gdb-4.16/COPYING gdb-4.17/COPYING
5c5
< 675 Mass Ave, Cambridge, MA 02139, USA
---
> 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
282c282
< Appendix: How to Apply These Terms to Your New Programs
---
> How to Apply These Terms to Your New Programs
```

…and so on for 178,200 lines (8,721 locations)

Challenges

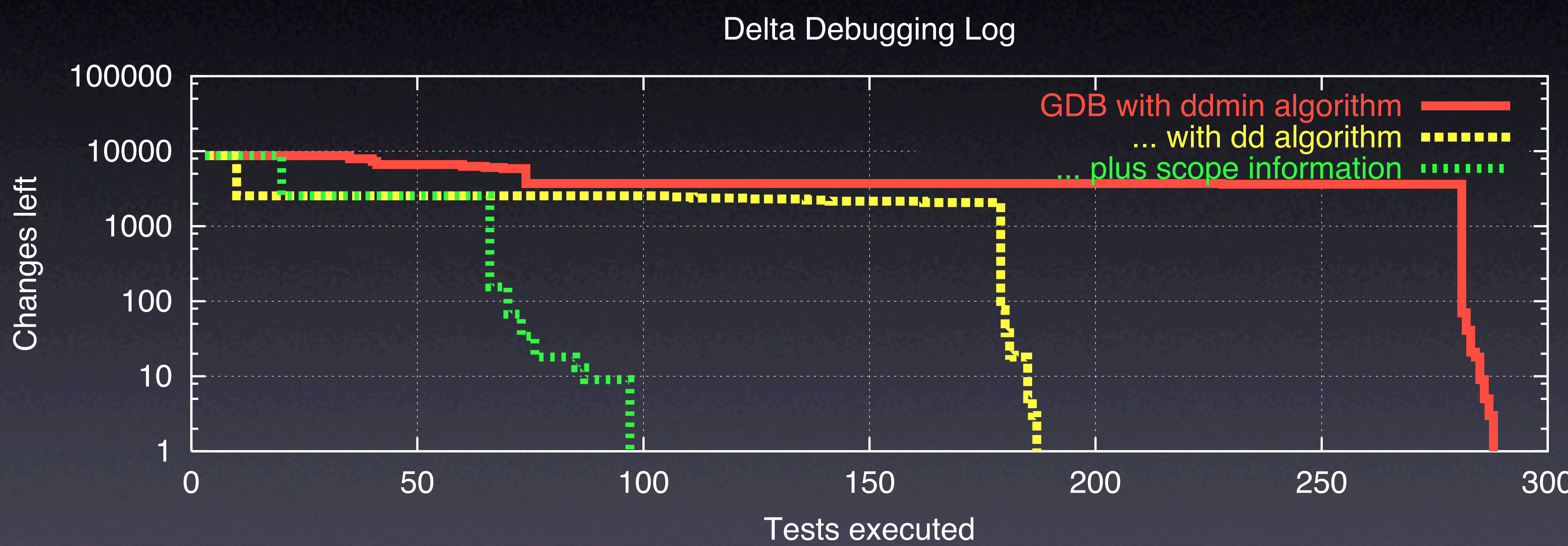
- Granularity – within some large change, only a few lines may be relevant
- Interference – some (later) changes rely on other (earlier) changes
- Inconsistency – some changes may have to be combined to produce testable code

Delta debugging handles all this

General Plan

- Decompose diff into changes per location (= 8,721 individual changes)
- Apply subset of changes, using PATCH
- Reconstruct GDB; build errors mean unresolved test outcome
- Test GDB and return outcome

Isolating Changes



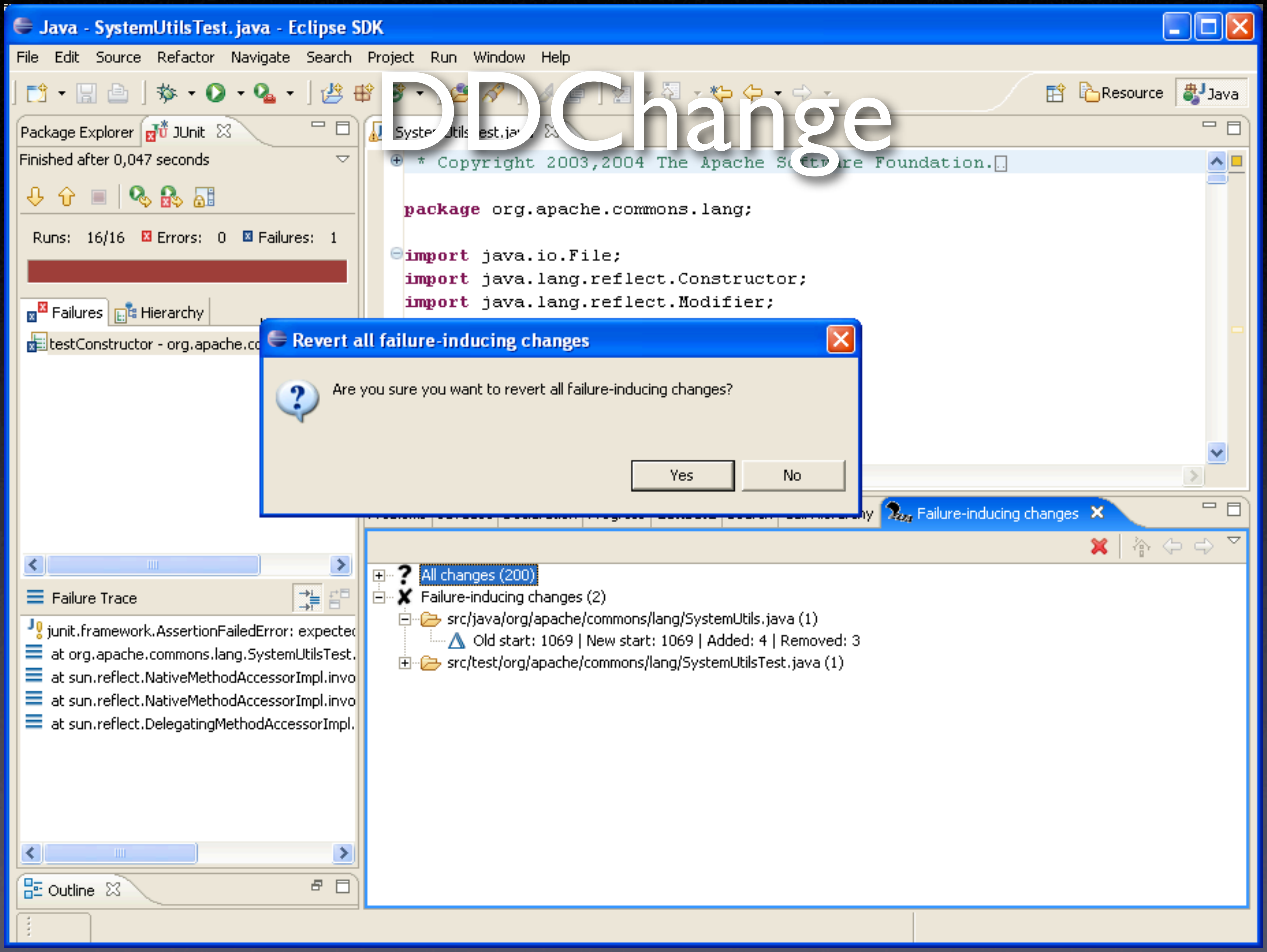
- Result after 98 tests (= 1 hour)

The Failure Cause

```
diff -r gdb-4.16/gdb/infcmd.c gdb-4.17/gdb/infcmd.c
1239c1278
< "Set arguments to give program being debugged when it is
started.\n
---
> "Set argument list to give program being debugged when
it is started.\n
```

- Documentation becomes GDB output
- DDD expects Arguments,
  but GDB outputs Argument list

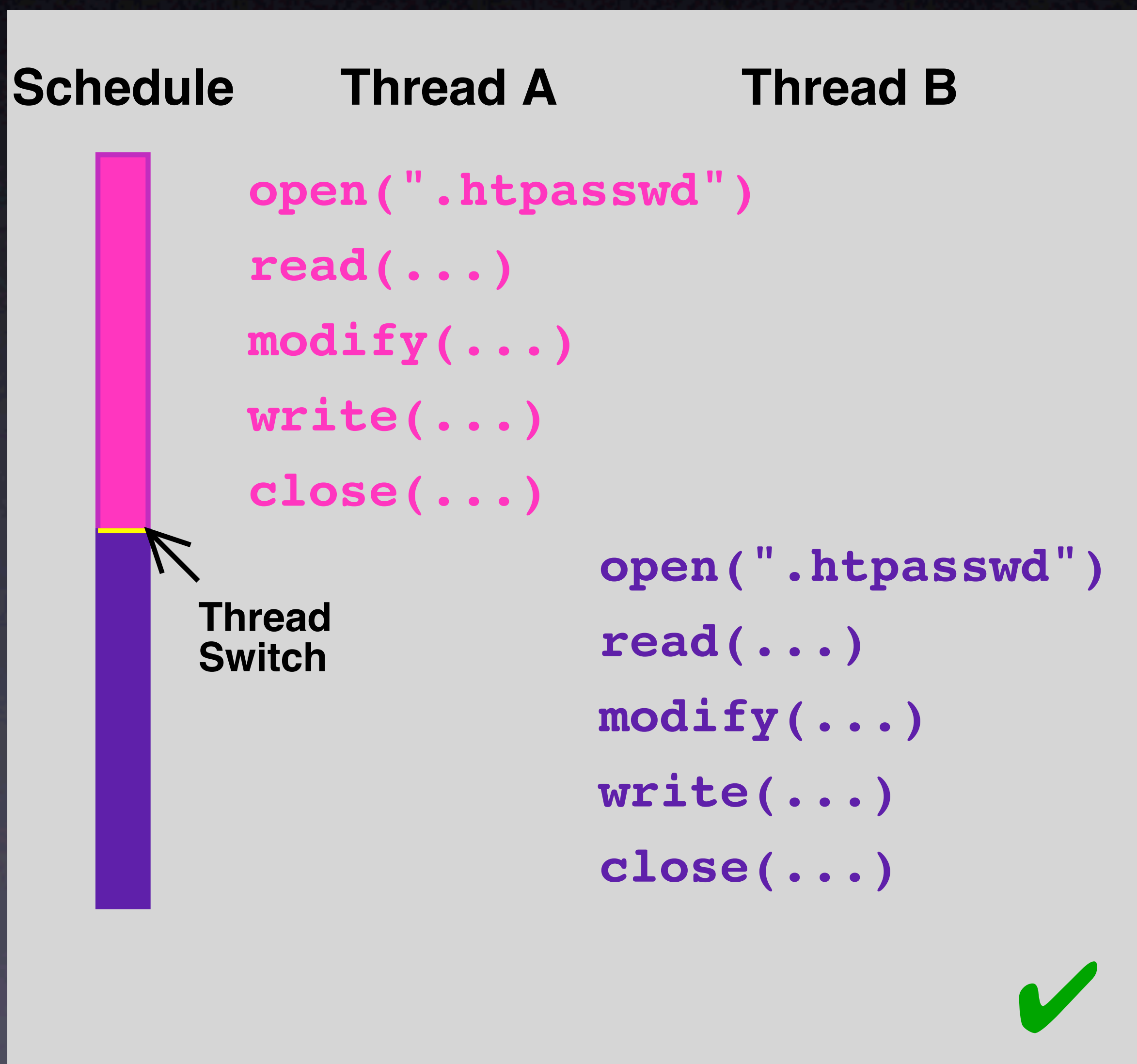
DDChange

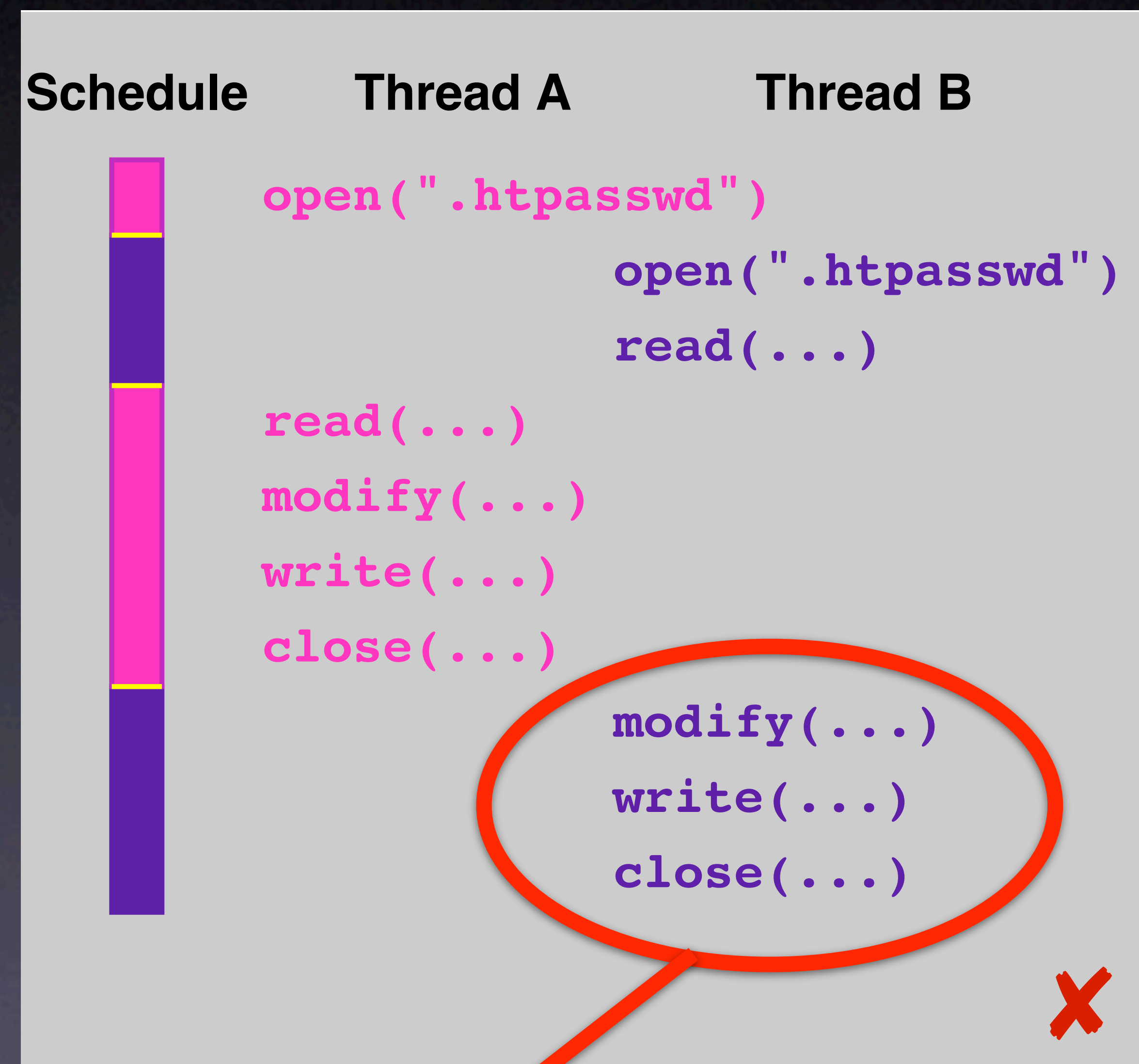

Java - SystemUtilsTest.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Resource
Java
Package Explorer
JUnit
Finished after 0,047 seconds
Runs: 16/16
Errors: 0
Failures: 1
Failures
Hierarchy
testConstructor - org.apache.co
Failure Trace
junit.framework.AssertionFailedError: expected
at org.apache.commons.lang.SystemUtilsTest.
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.DelegatingMethodAccessorImpl.
Outline
* Copyright 2003,2004 The Apache Software Foundation.
package org.apache.commons.lang;
import java.io.File;
import java.lang.reflect.Constructor;
import java.lang.reflect.Modifier;
Revert all failure-inducing changes
Are you sure you want to revert all failure-inducing changes?
Yes
No
Failure-inducing changes
All changes (200)
Failure-inducing changes (2)
src/java/org/apache/commons/lang/SystemUtils.java (1)
Old start: 1069 | New start: 1069 | Added: 4 | Removed: 3
src/test/org/apache/commons/lang/SystemUtilsTest.java (1)

Optimizations

- History – group changes by creation time
- Reconstruction – cache several builds
- Grouping – according to scope
- Failure Resolution – scan error messages for possibly missing changes

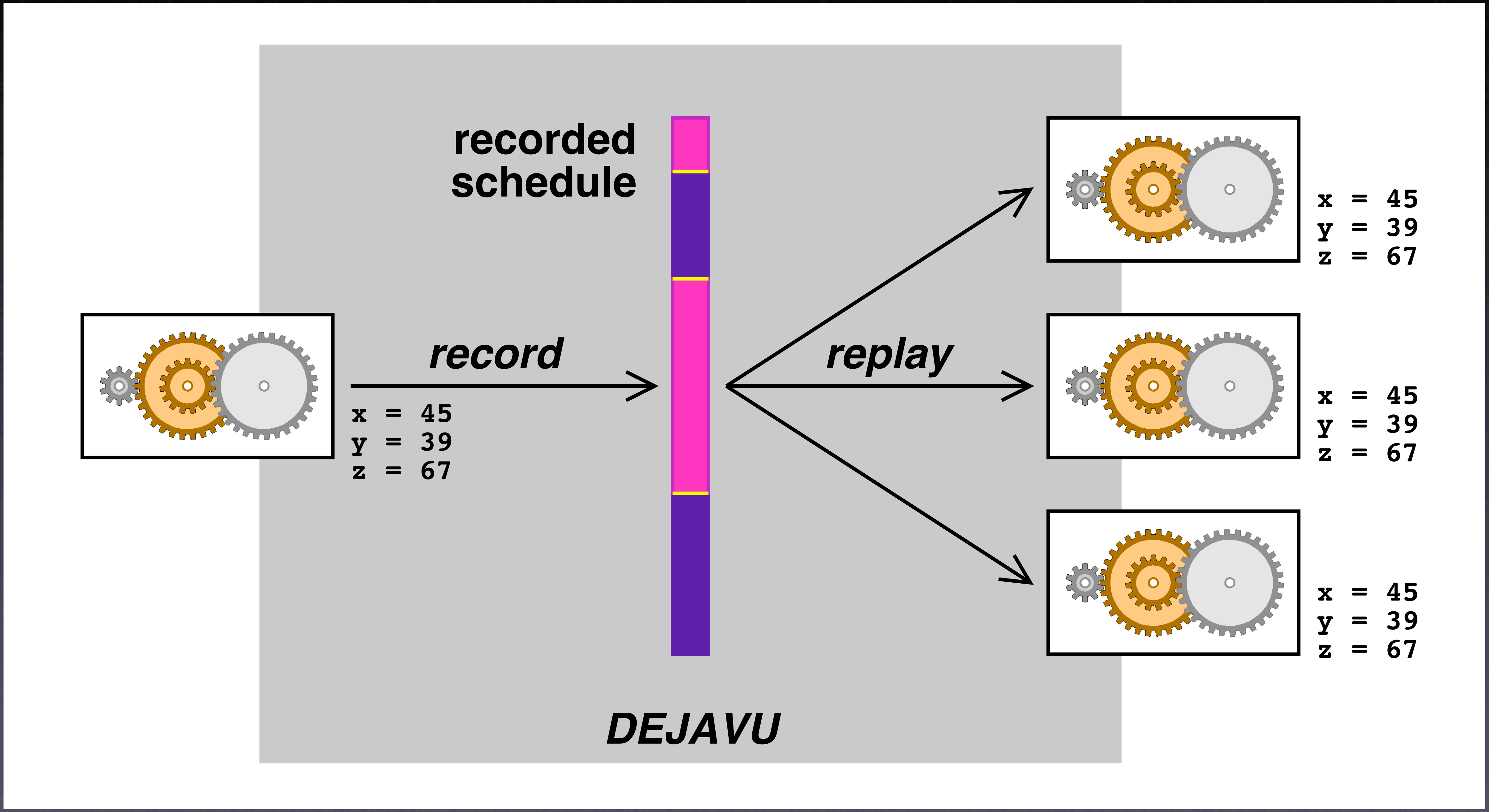
Thread Schedules



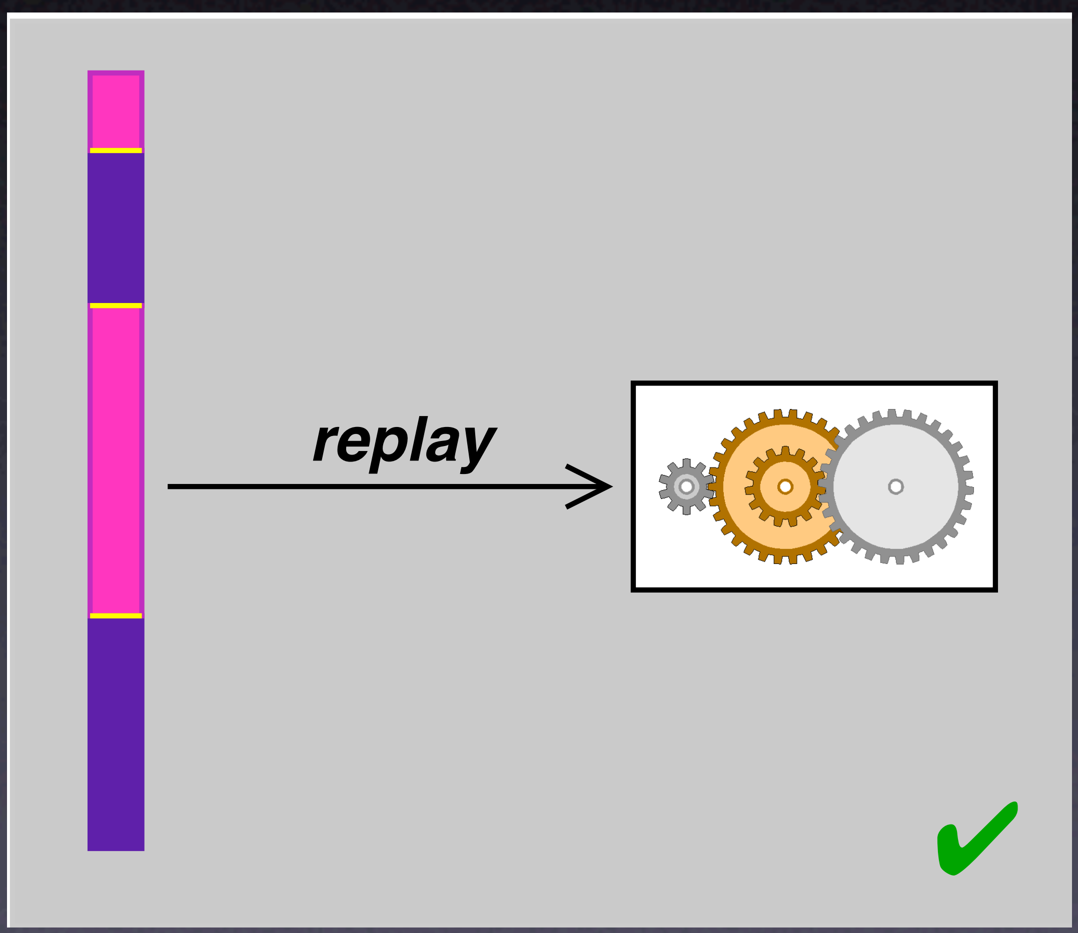


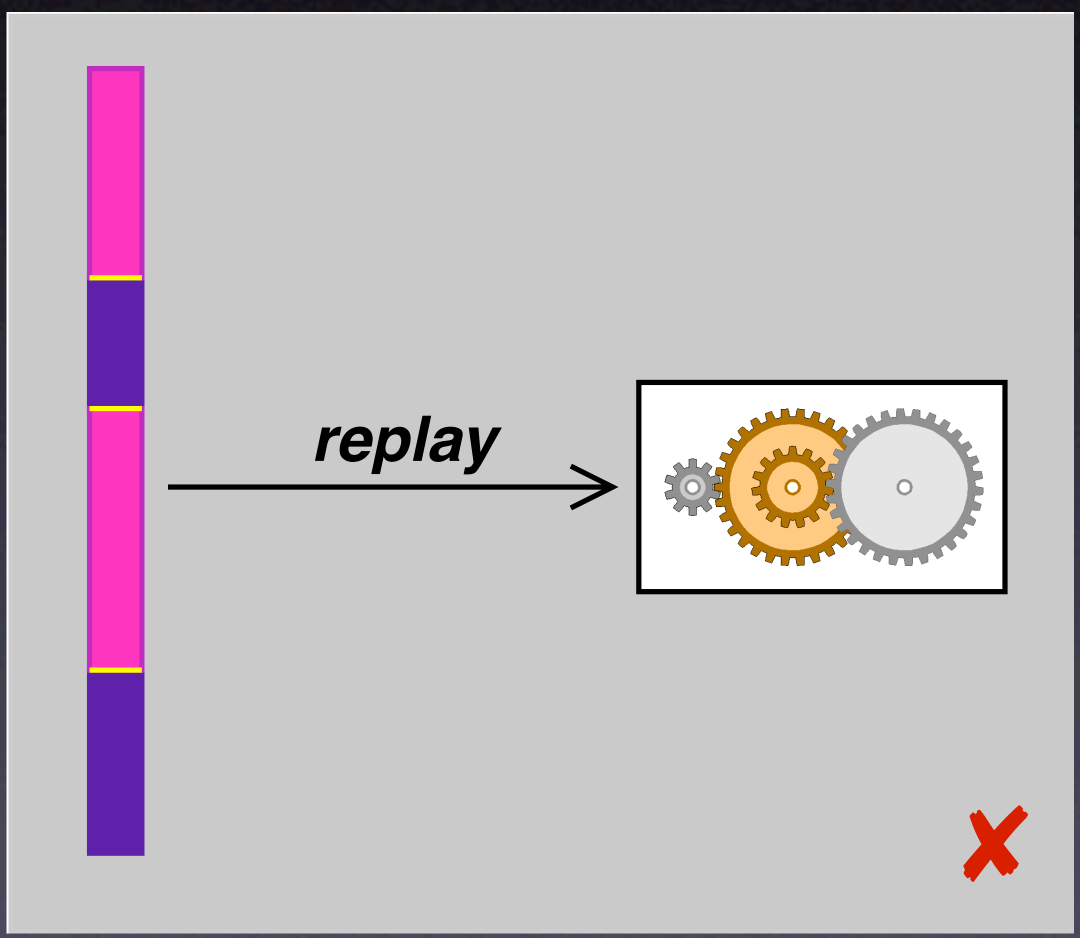
A’s updates get lost!

Record + Replay



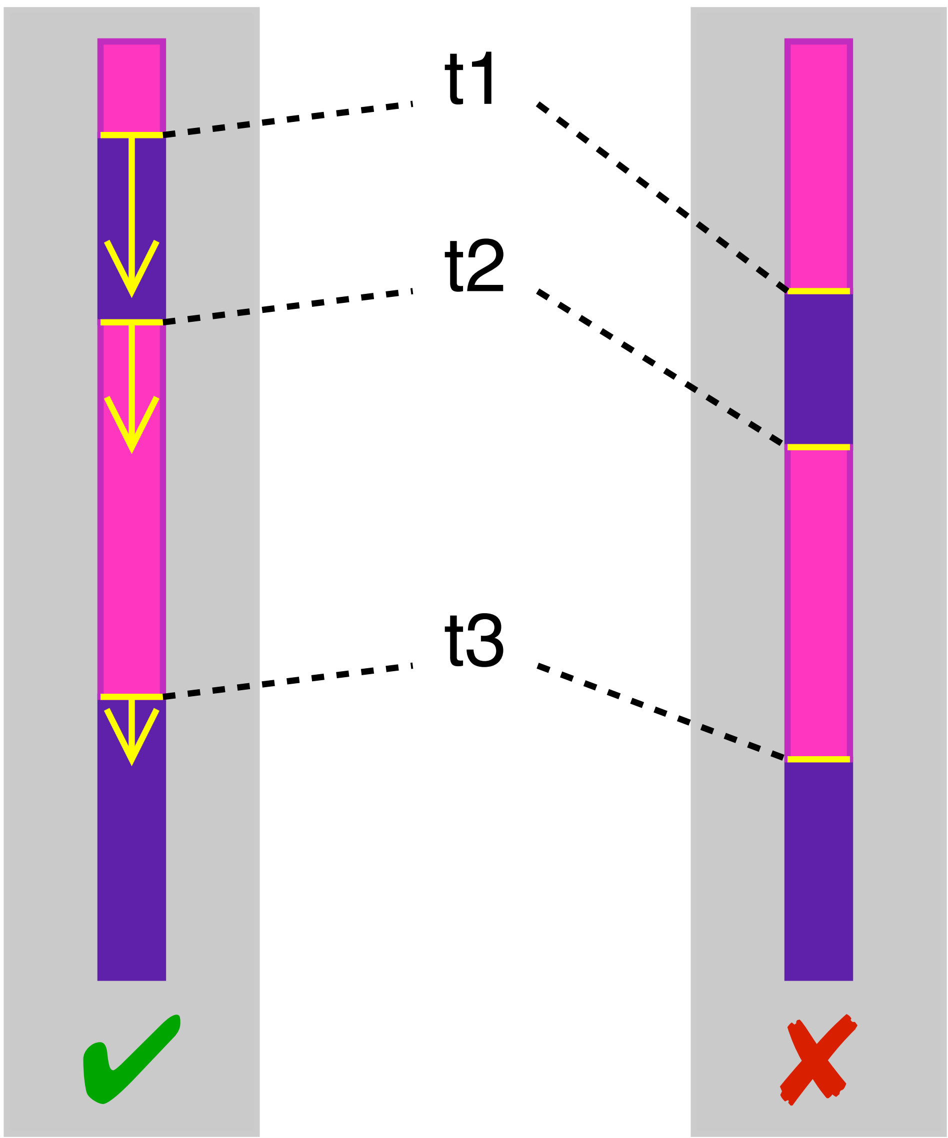
Schedules as Input





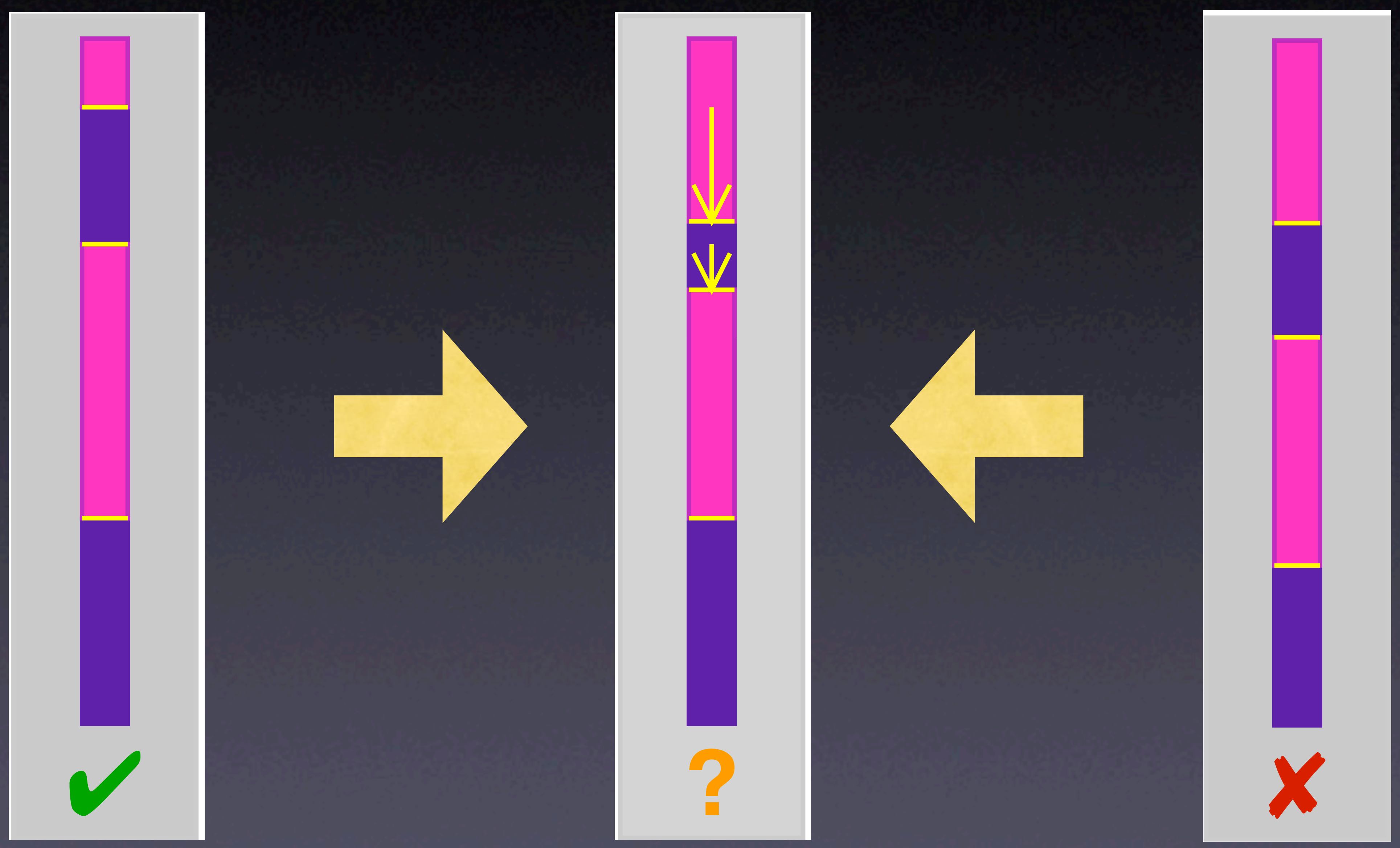
The schedule difference causes the failure!

Finding Differences

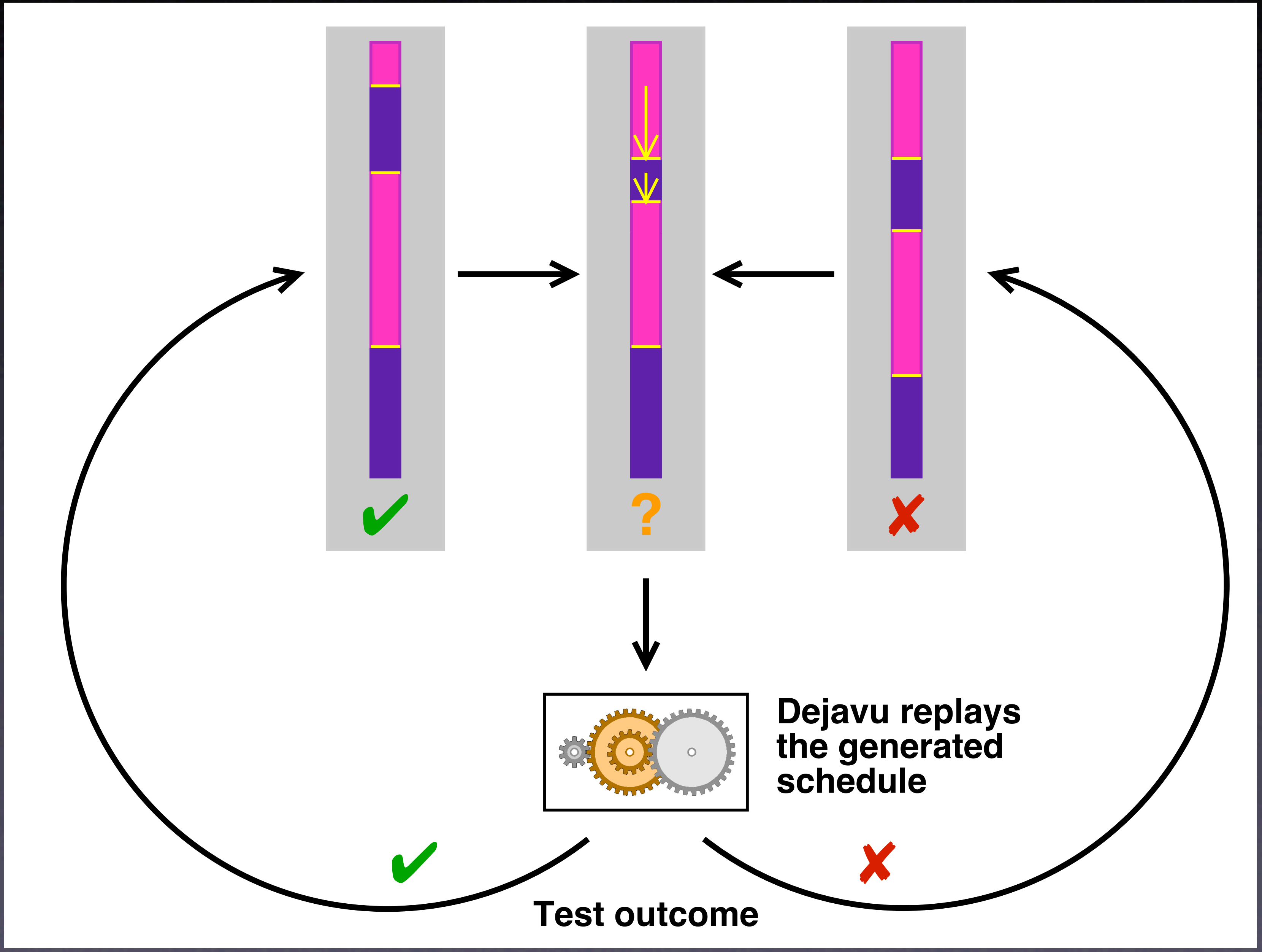


- We start with runs ✔ and ✘
- We determine the differences Δ_i between thread switches t_i:
  - t_1 occurs in ✔ at “time” 254
  - t_1 occurs in ✘ at “time” 278
  - The difference $\Delta_1 = |278 - 254|$ induces a *statement interval:* the code executed between “time” 254 and 278
  - Same applies to t_2, t_3, etc.

Isolating Differences



Isolating Differences



Example: Raytracer

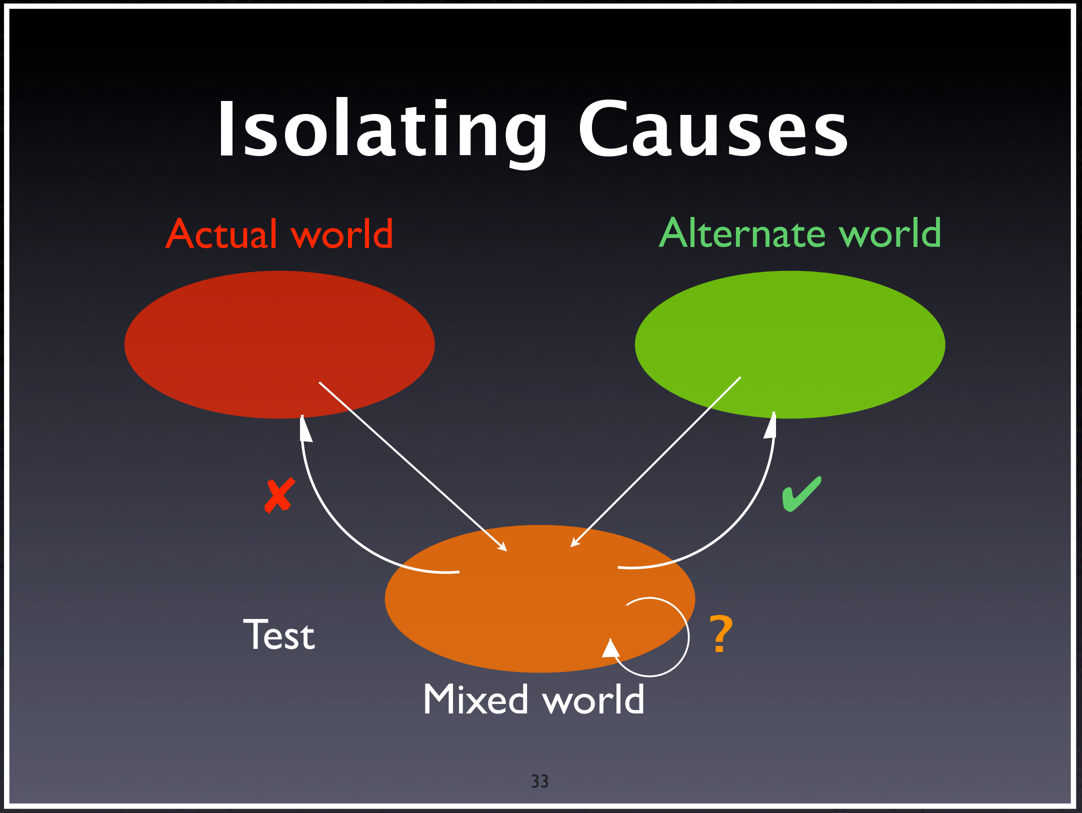
- Raytracer program from Spec JVM98 suite
- Injected a simple *race condition*
- Set up *automated test* + *random schedules*
- Obtained *passing* and *failing* schedule
- 3,842,577,240 differences, each moving a thread switch by ±1 *yield point* (time unit)

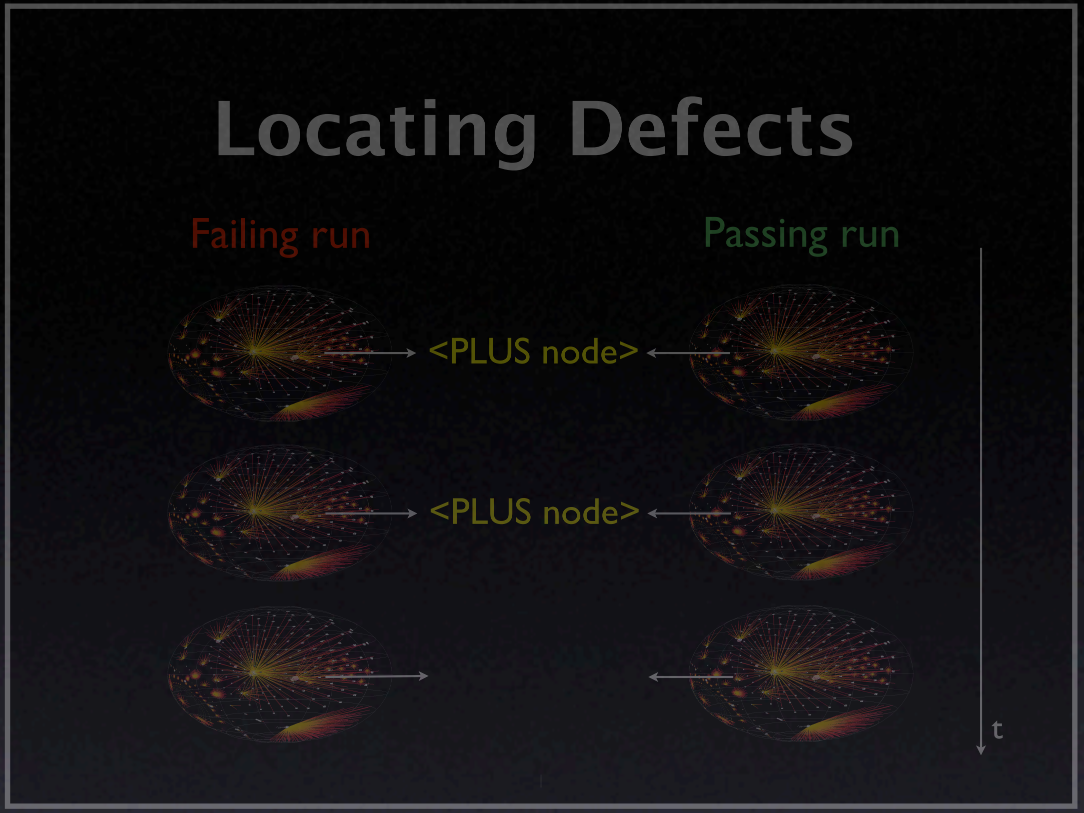
Isolating Schedules

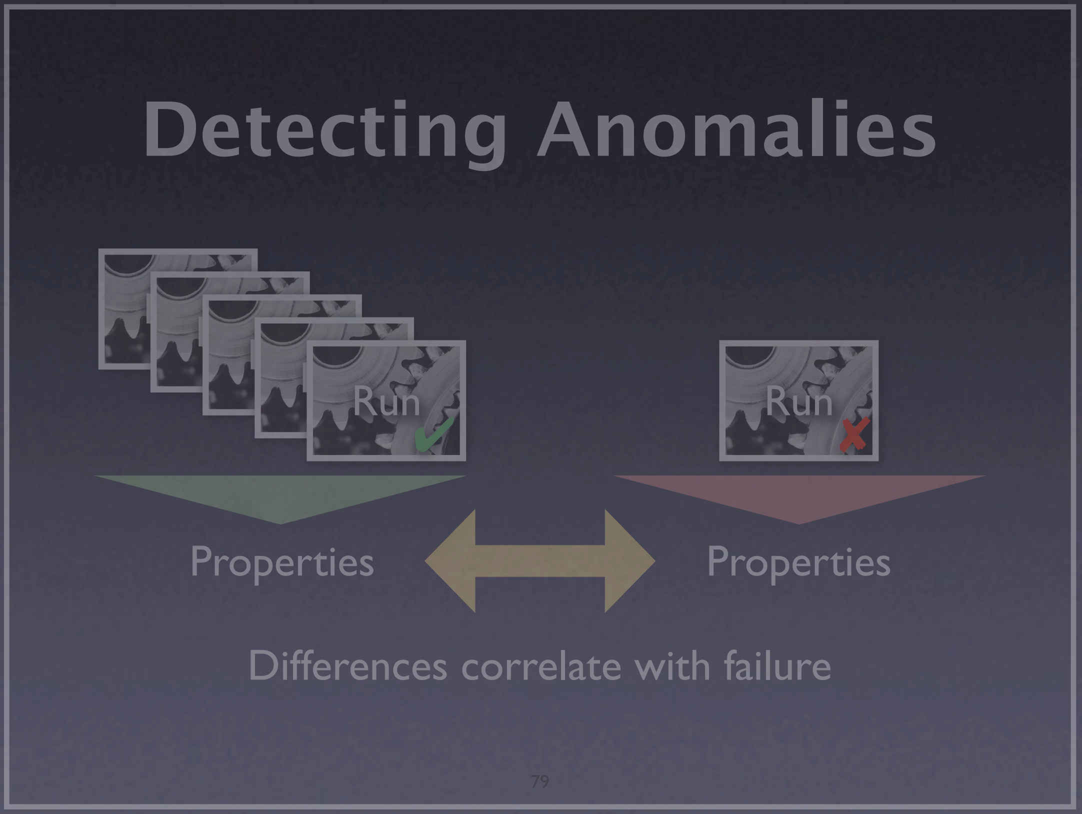
no unresolved outcomes: complexity is $O(\log_2 n)$

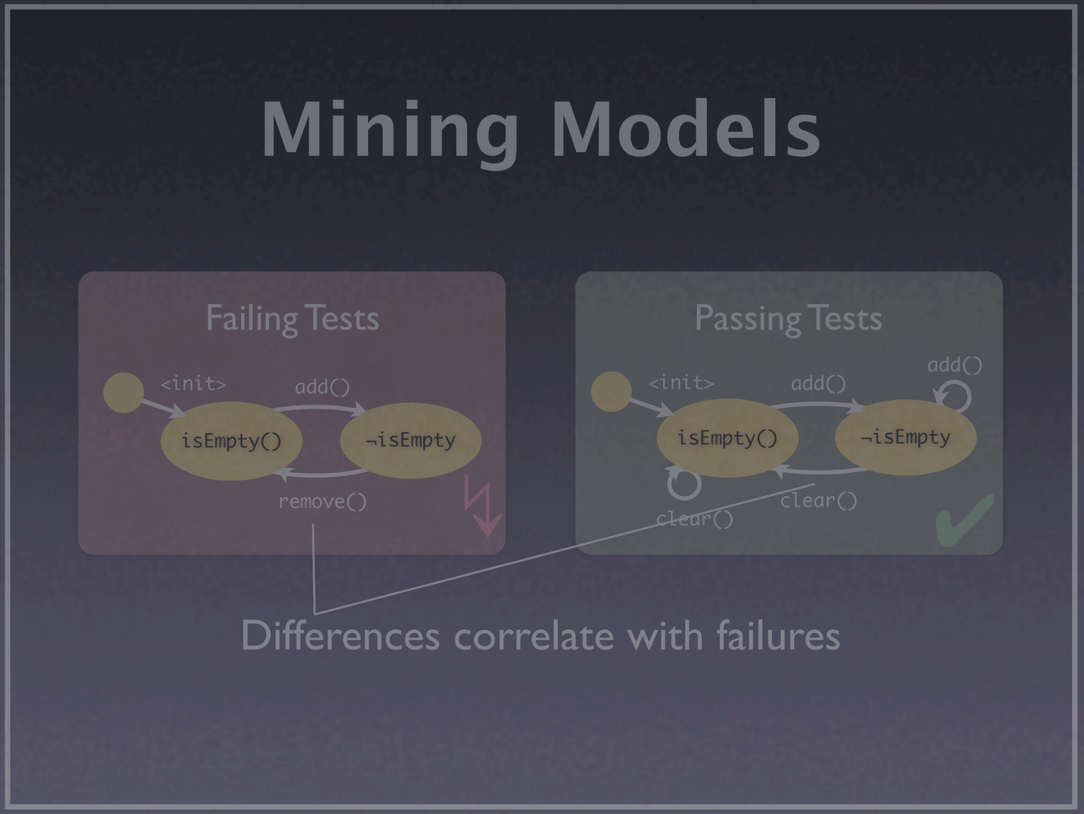
The Failure Cause

```
public class Scene { …
    private static int ScenesLoaded = 0;
    (more methods…)
    private
    int LoadScene(String filename) {
        int OldScenesLoaded = ScenesLoaded;
        (more initializations…)
        infile = new DataInputStream(…);
        (more code…)
        ScenesLoaded = OldScenesLoaded + 1;
        System.out.println("" +
                ScenesLoaded + " scenes loaded.");
        …
    }
    …
}
```

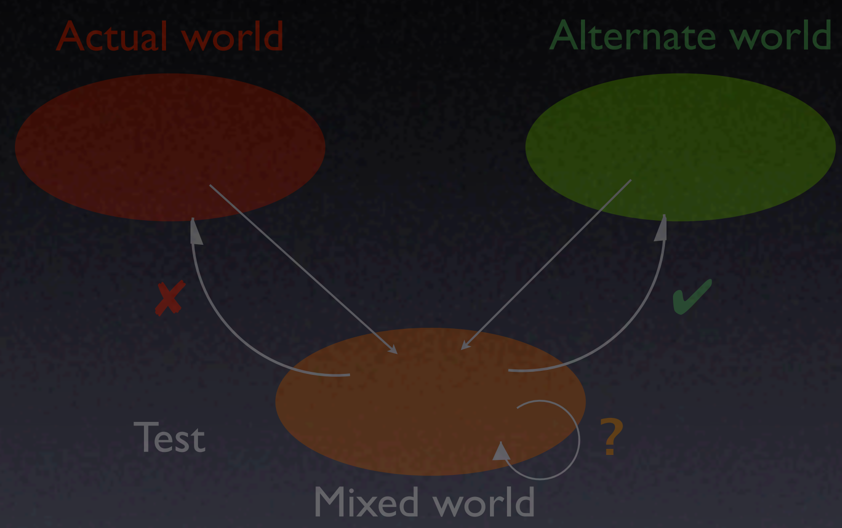

Isolating Causes
Actual world
Alternate world
✘
✔
Test
?
Mixed world
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t


Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79

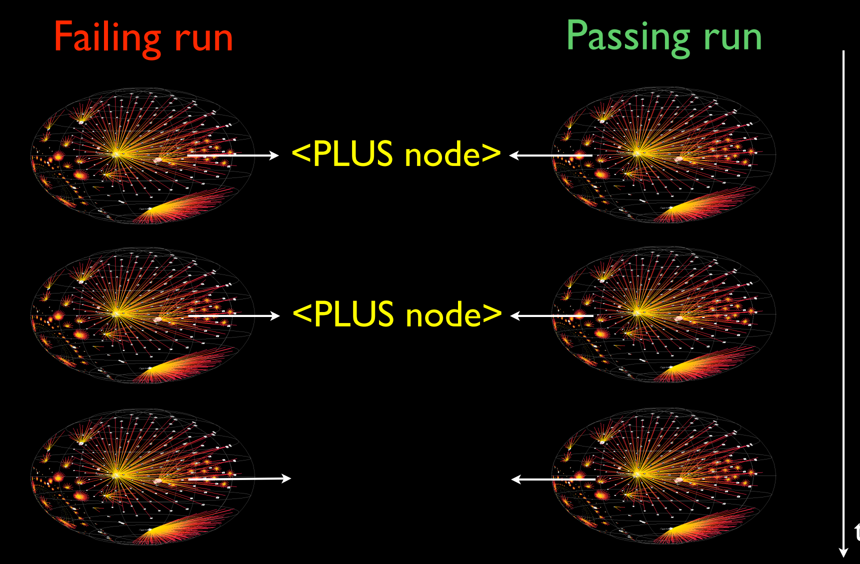

Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures

Isolating Causes

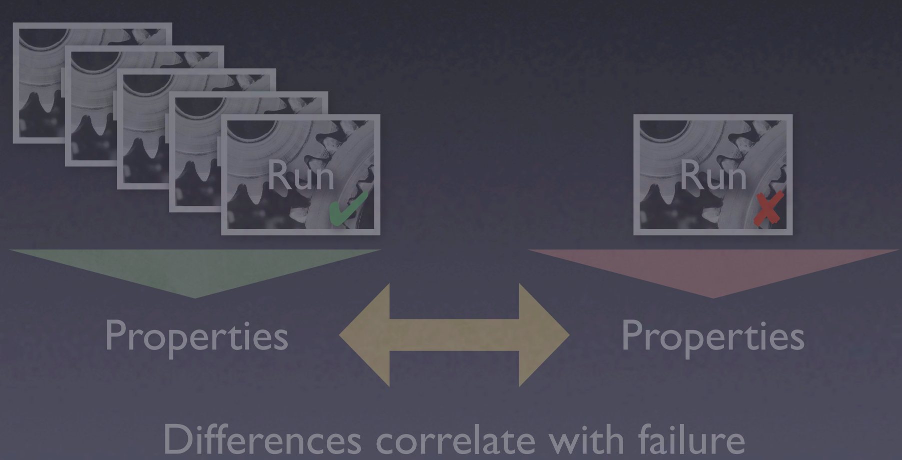


33

Locating Defects

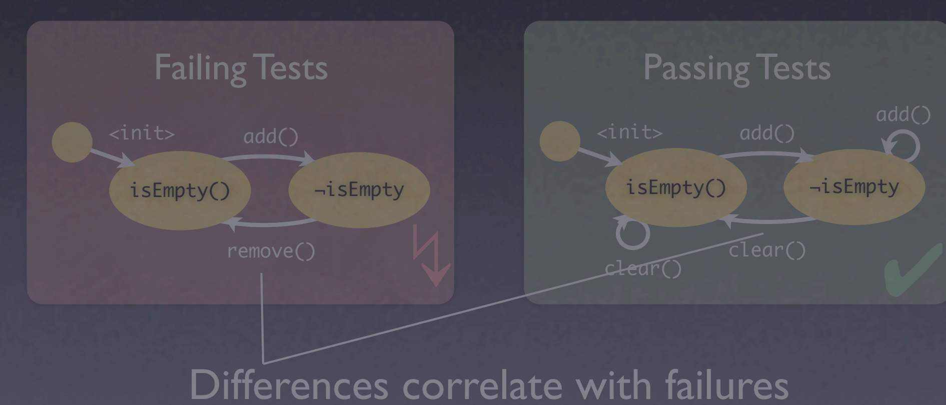


Detecting Anomalies



79

Mining Models



Boskoop: bug (~/tmp/bug) <zeller.zeller> — bash — 80x24 — ⌘1

```
$ ls
bug.c
$ gcc-2.95.2 -O bug.c
gcc: Internal error: program cc1 got fatal signal 11
Segmentation fault
$
```

What is the cause of this failure?

Isolating Causes

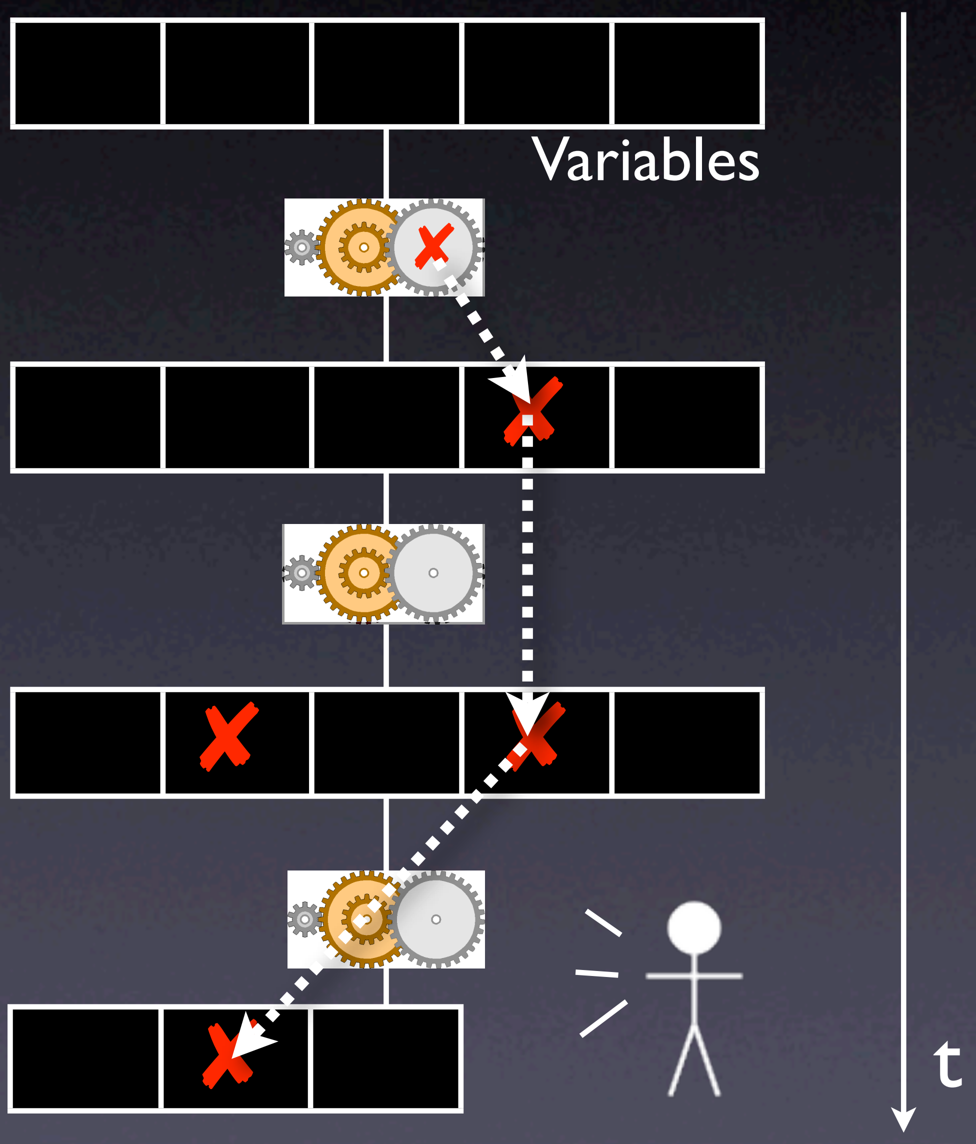
```
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
```

Actual cause of the GCC crash

From Defect to Failure

1. The programmer creates a *defect* – an error in the code.
2. When executed, the defect creates an *infection* – an error in the state.
3. The infection *propagates*.
4. The infection causes a *failure*.

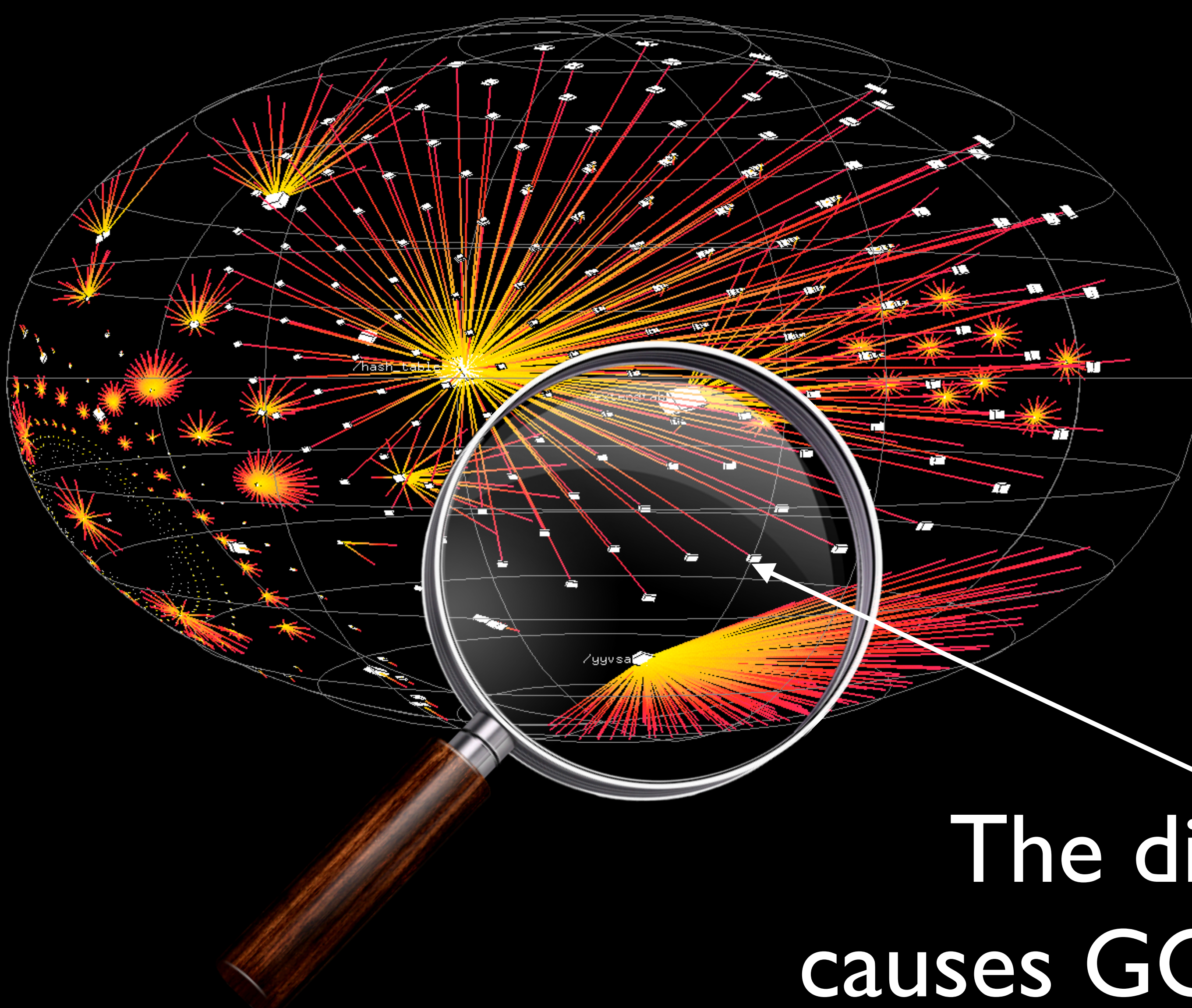
This infection chain must be traced back – and broken.

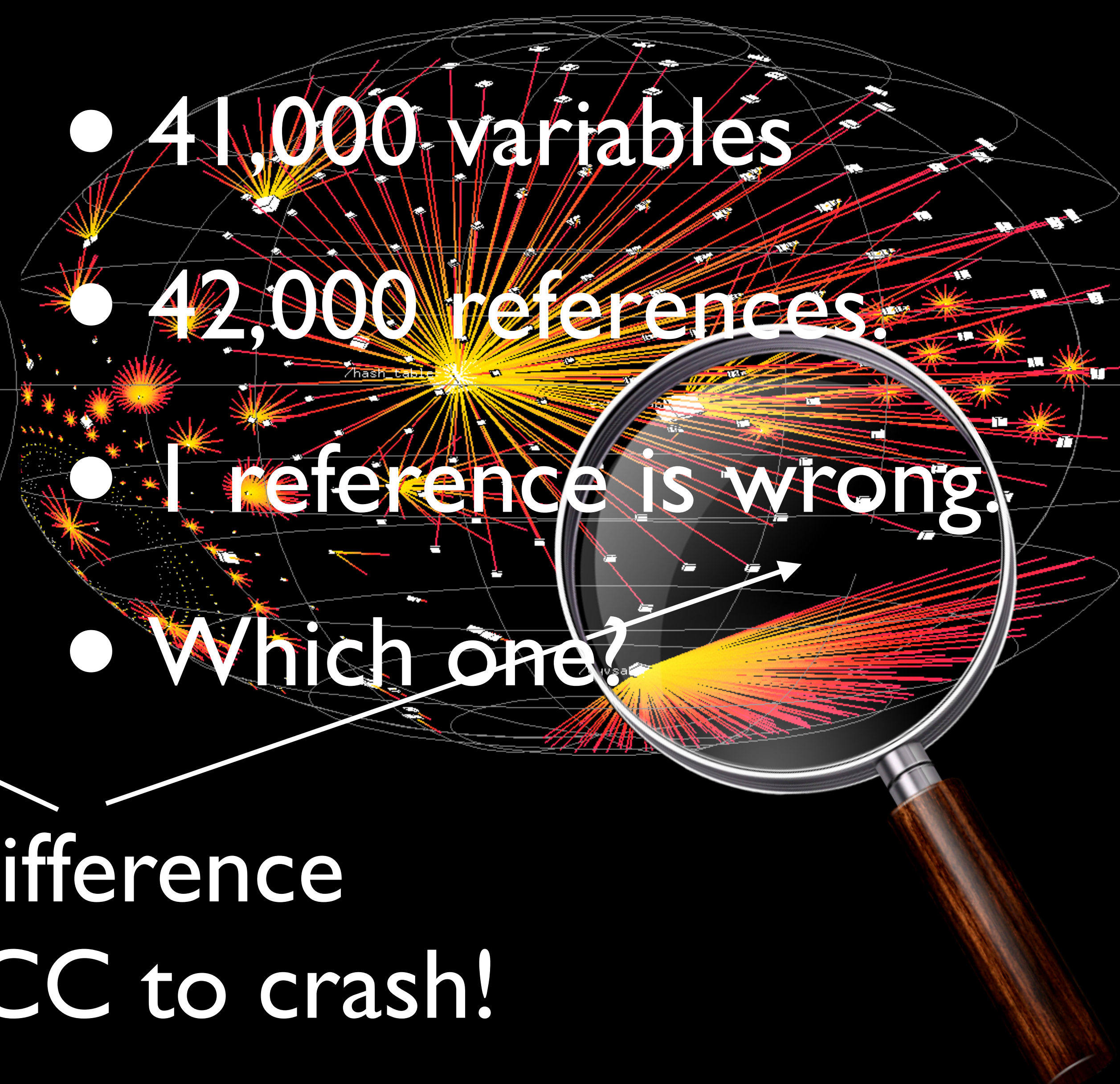


Causes in State

Infected state

Sane state

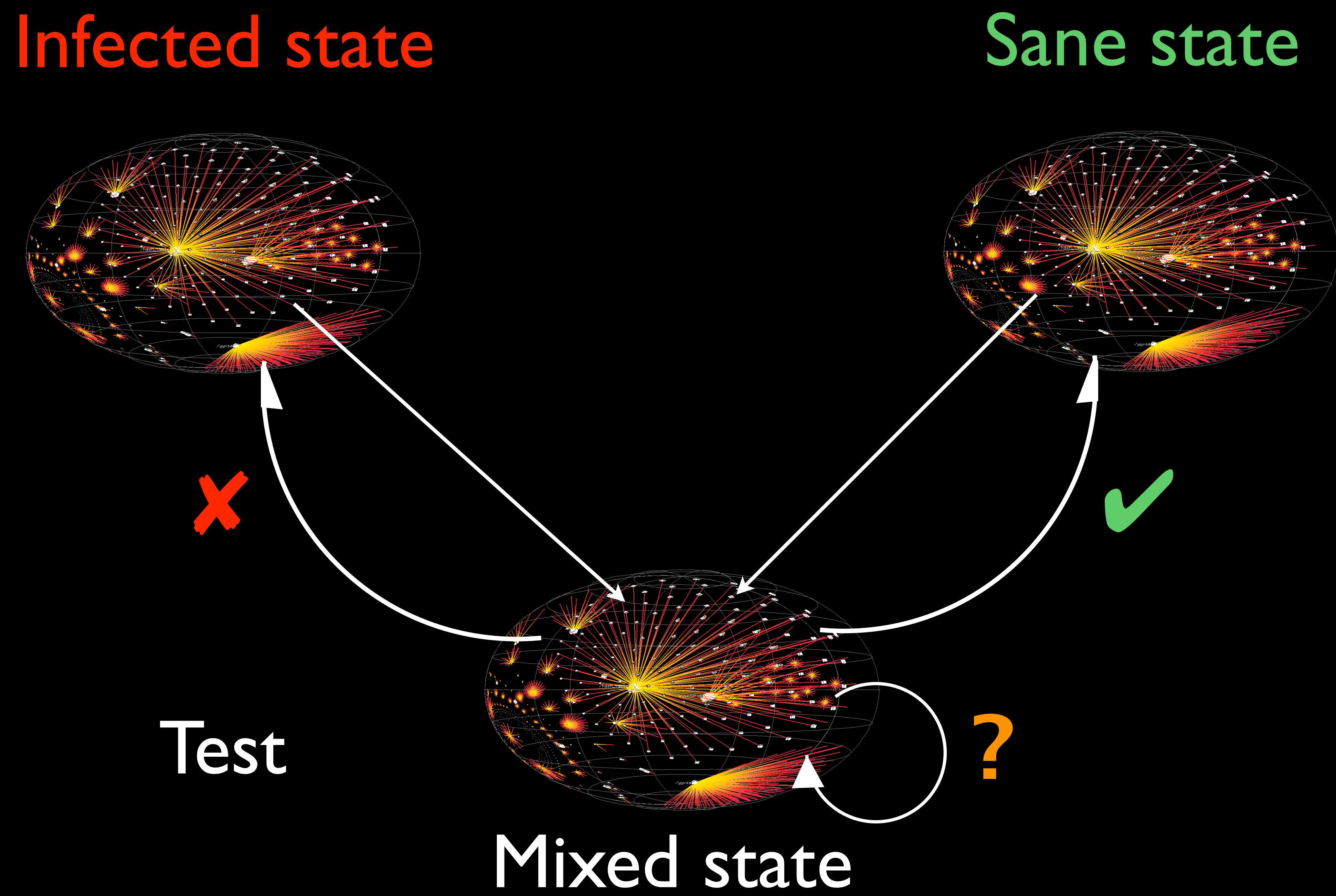




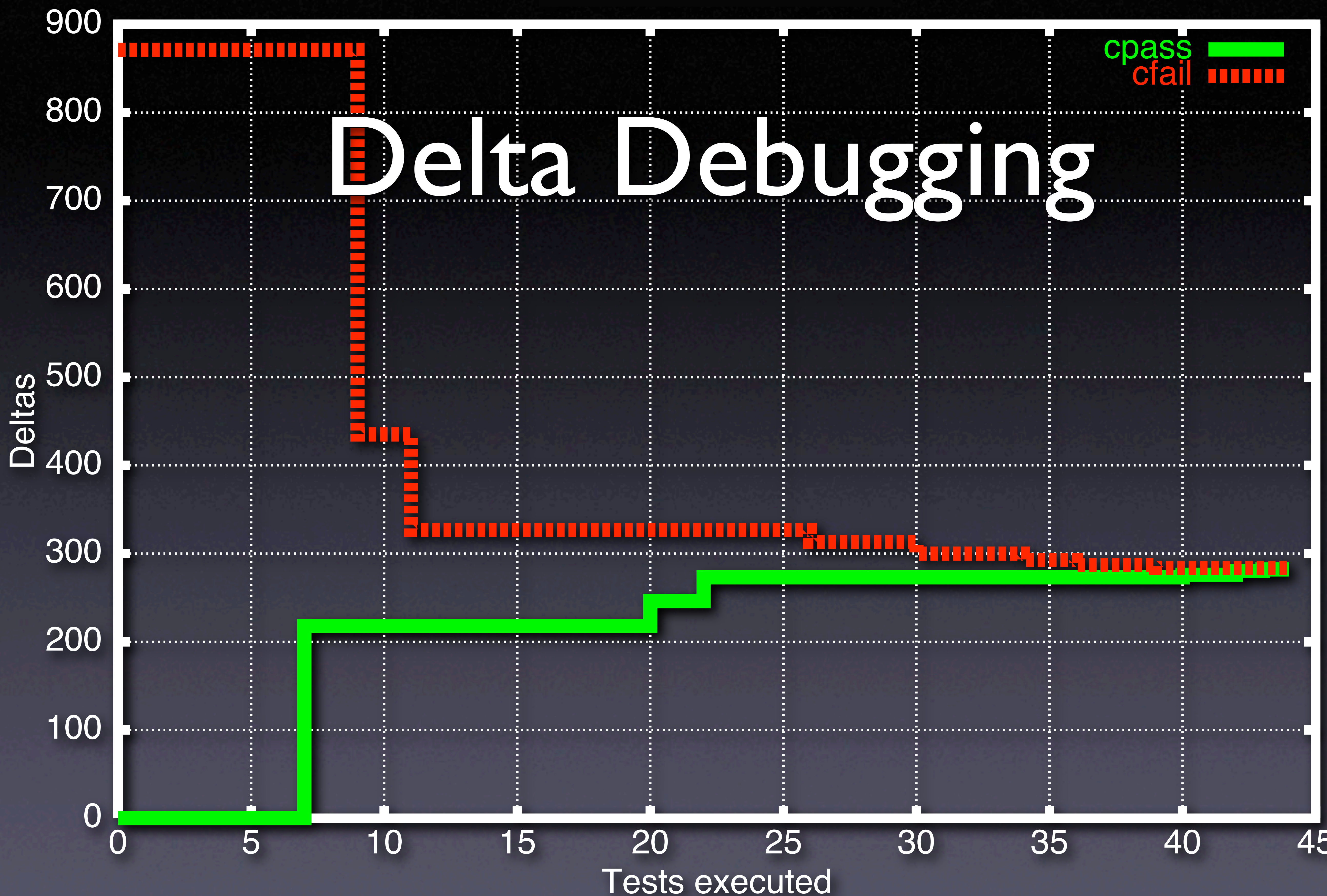
- 41,000 variables
- 42,000 references.
- 1 reference is wrong.
- Which one?

The difference
causes GCC to crash!

Search in Space



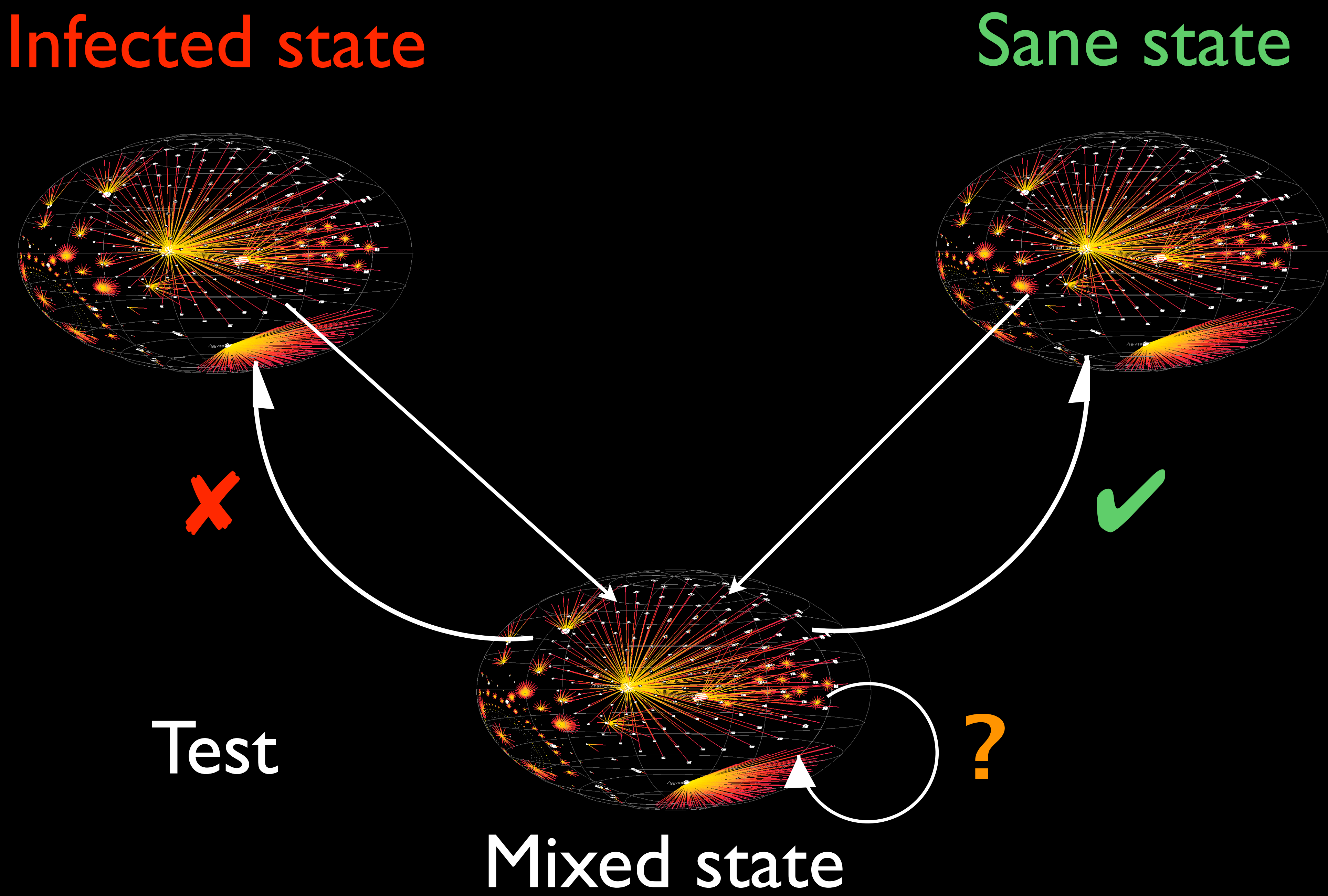
Delta Debugging



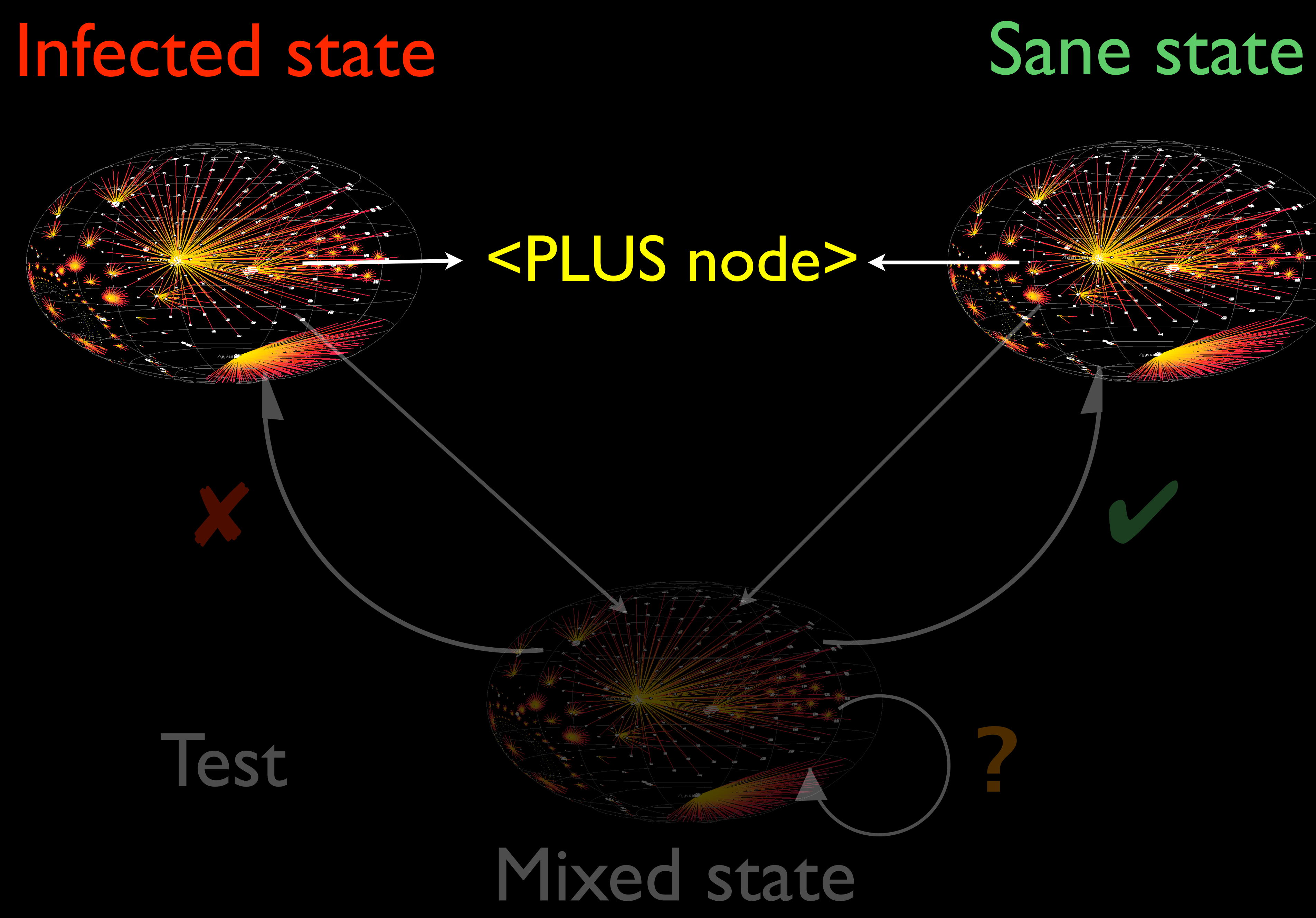
Delta Debugging

first_loop_store_insn→fld[1].rtx→fld[1].rtx→
fld[3].rtx→fld[1].rtx→code == PLUS

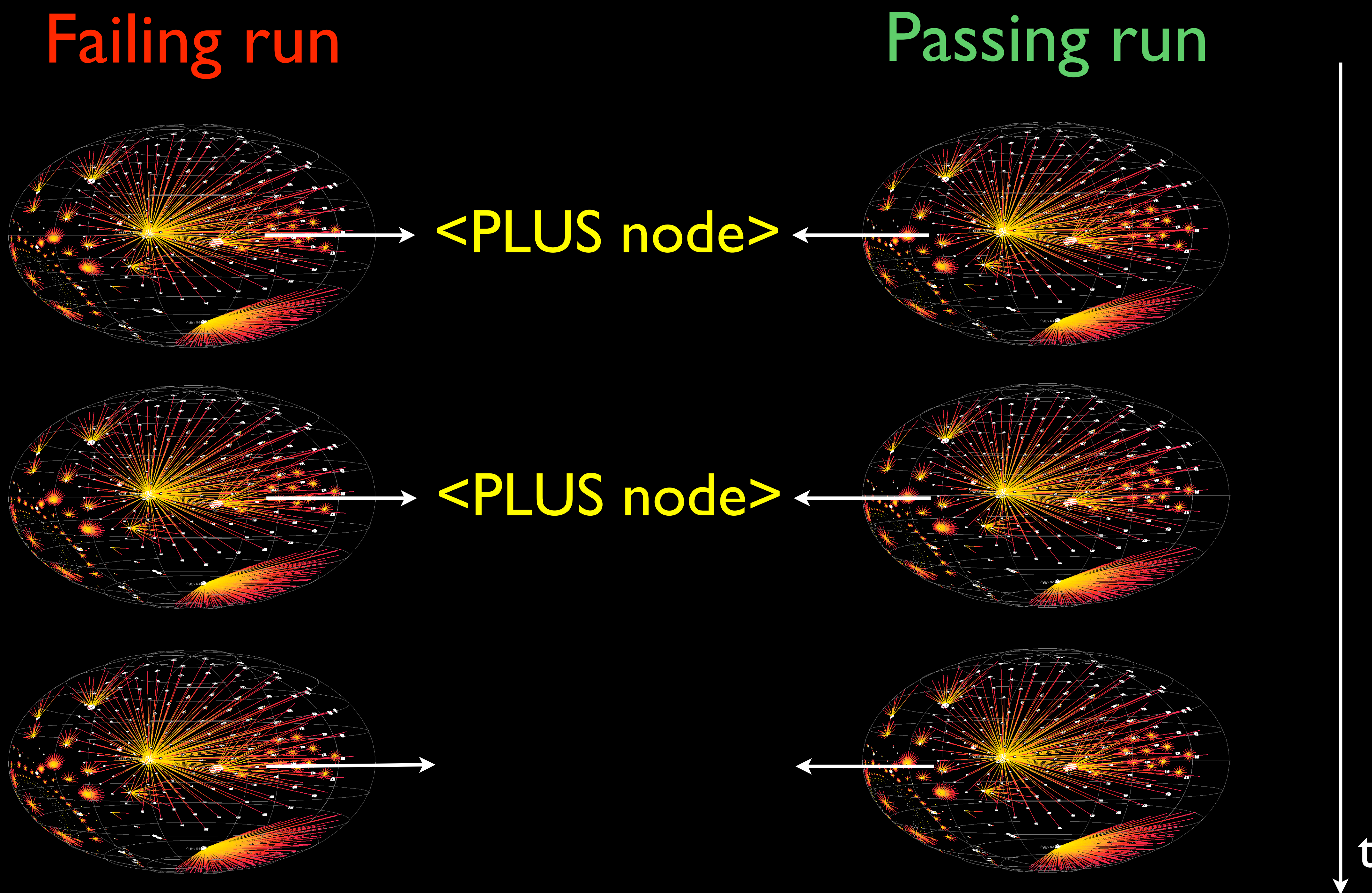
Search in Space



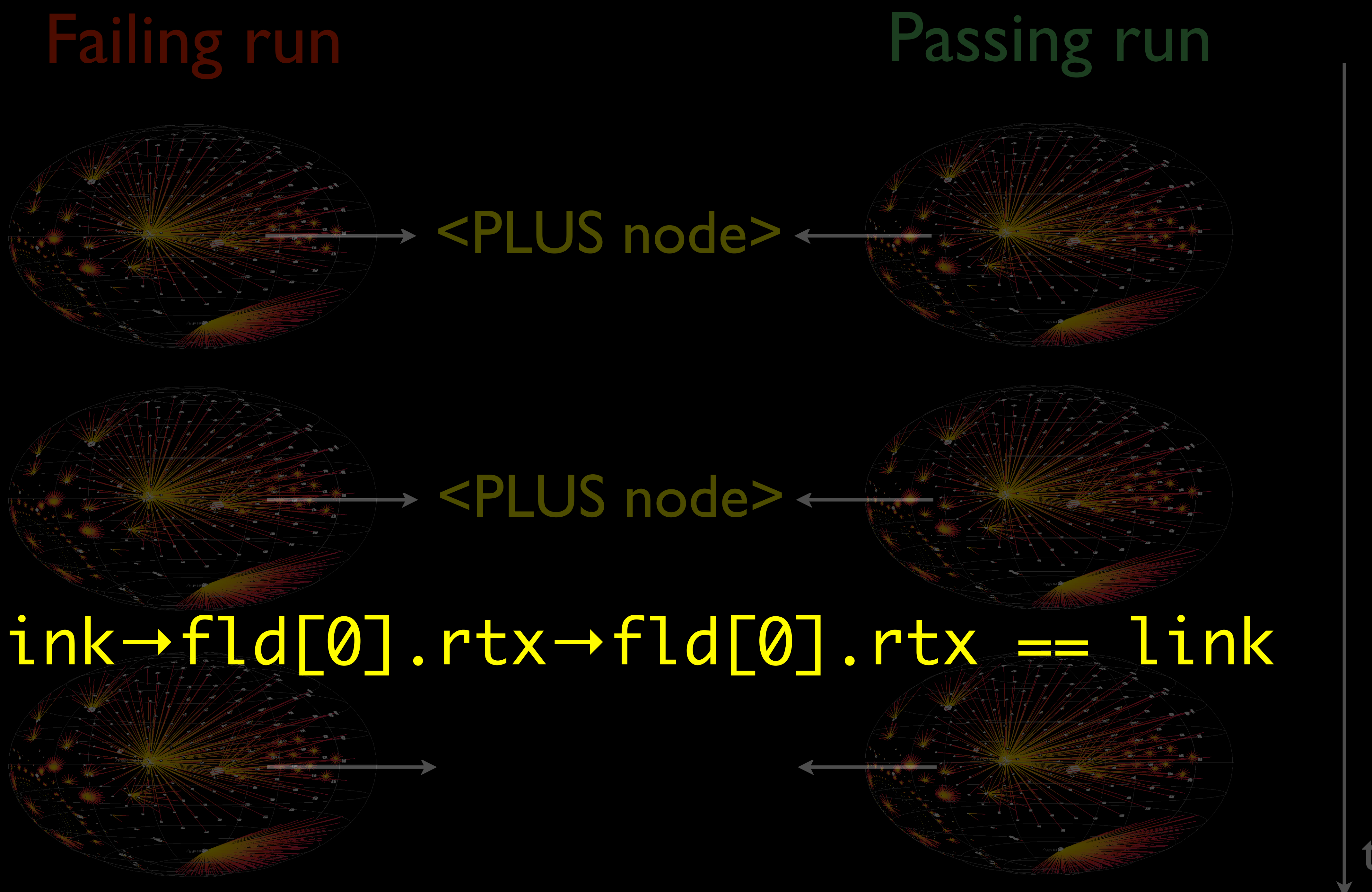
Search in Space



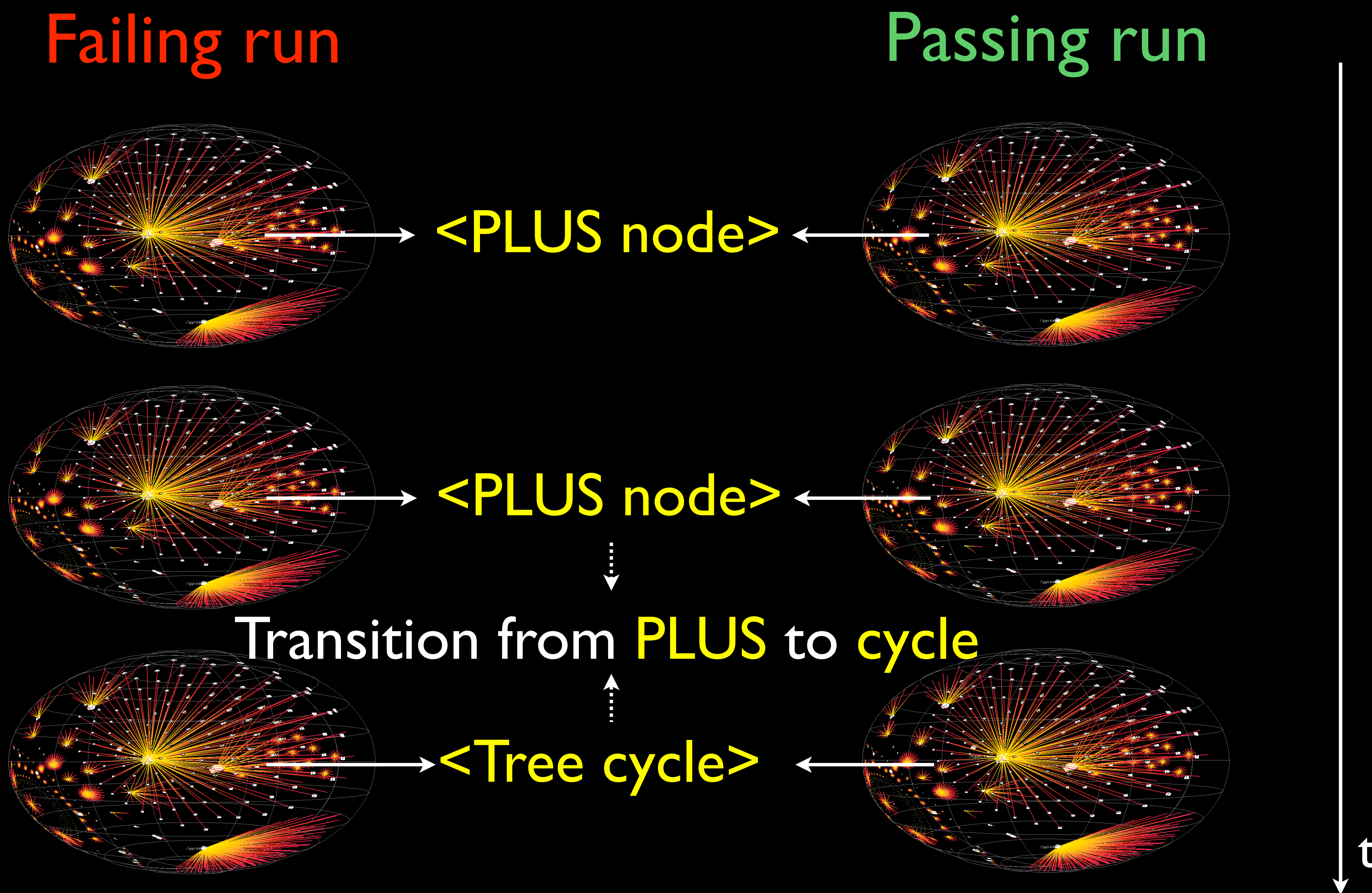
Search in Time



Search in Time



Search in Time



Why Transitions?

- Each failure cause in the program state is caused by some statement
- These statements are executed at cause transitions
- Cause transitions thus are statements that cause the failure

All GCC Transitions

Location	New cause at transition
<Start>	argv[3]
toplev.c:4755	name
toplev.c:2909	dump_base_name
c-lex.c:187	finput→_IO_buf_base
c-lex.c:1213	nextchar
c-lex.c:1213	yyssa[41]
c-typeck.c:3615	yyssa[42]
c-lex.c:1213	last_insn→fld[1].rtx→…→fld[1].rtx.code
c-decl.c:1213	sequence_result[2]→…→fld[1].rtx.code
combine.c:4271	x→fld[0].rtx→fld[0].rtx

combine.c

```
if (GET_CODE (XEXP (x, 0)) == PLUS {
    x = apply_distributive_law
      (gen_binary (PLUS, mode,
                                    (x, 0), 0),
                                   ),

                                XEXP (XEXP (x, 0), 1),
                                XEXP (x, 1))));

    if (GET_CODE (x) != MULT)
        return x;
}
```

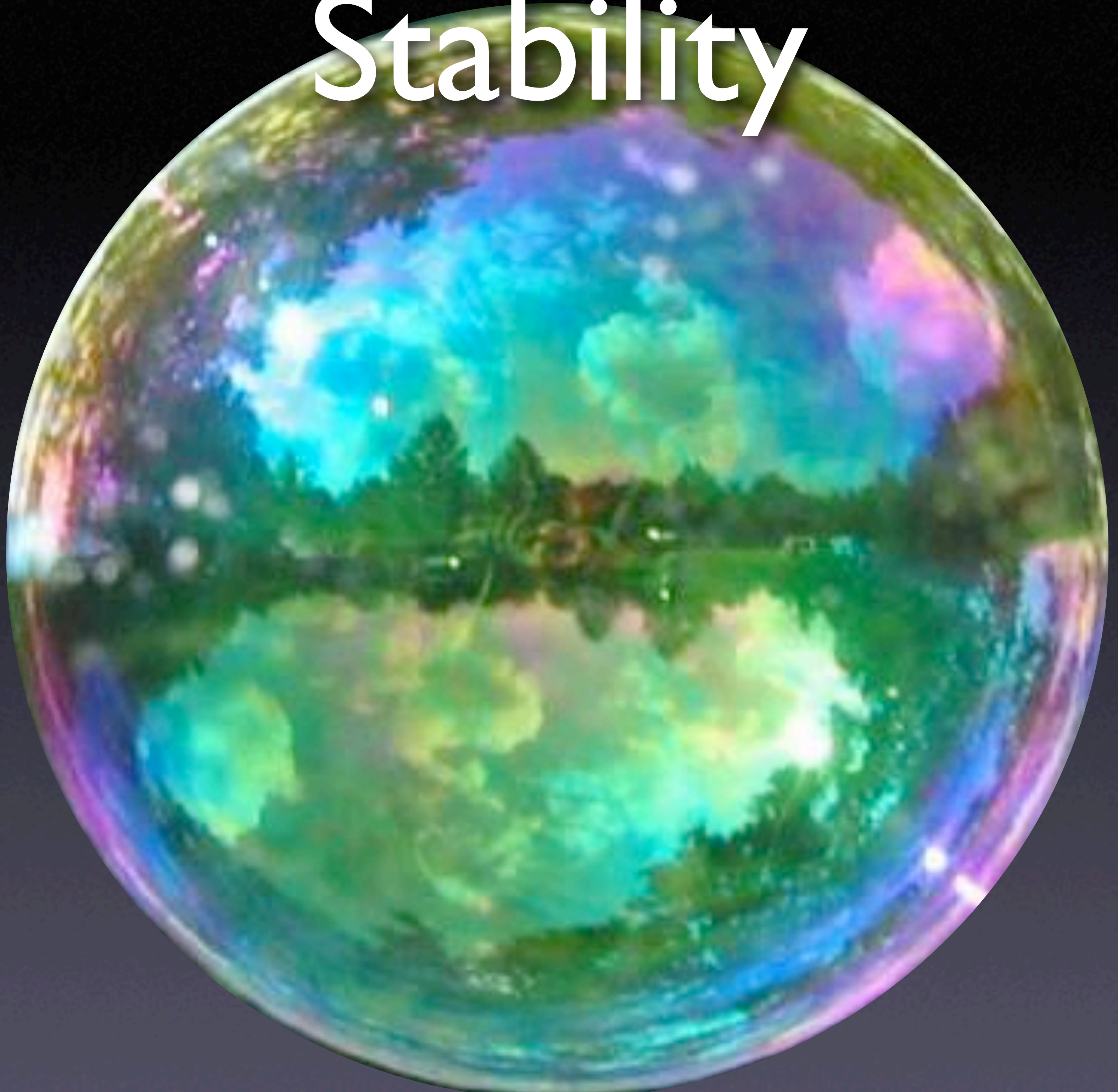
2 lines out of 338,000

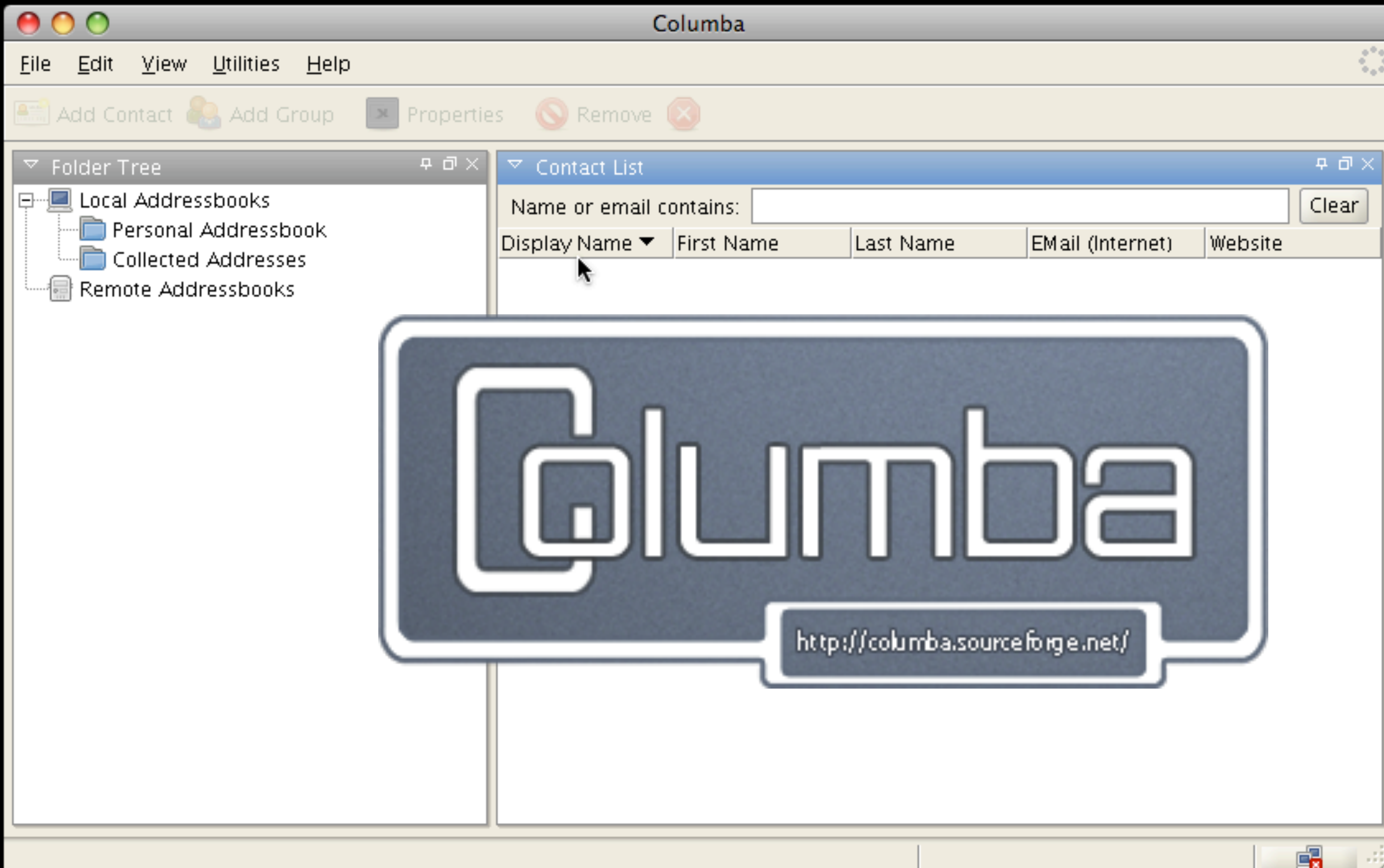
Should be copy_rtx()

Implementations

C	Java	Python
Web service + command line	Eclipse plug-in	Module
24 months	12 months	2 days

Stability



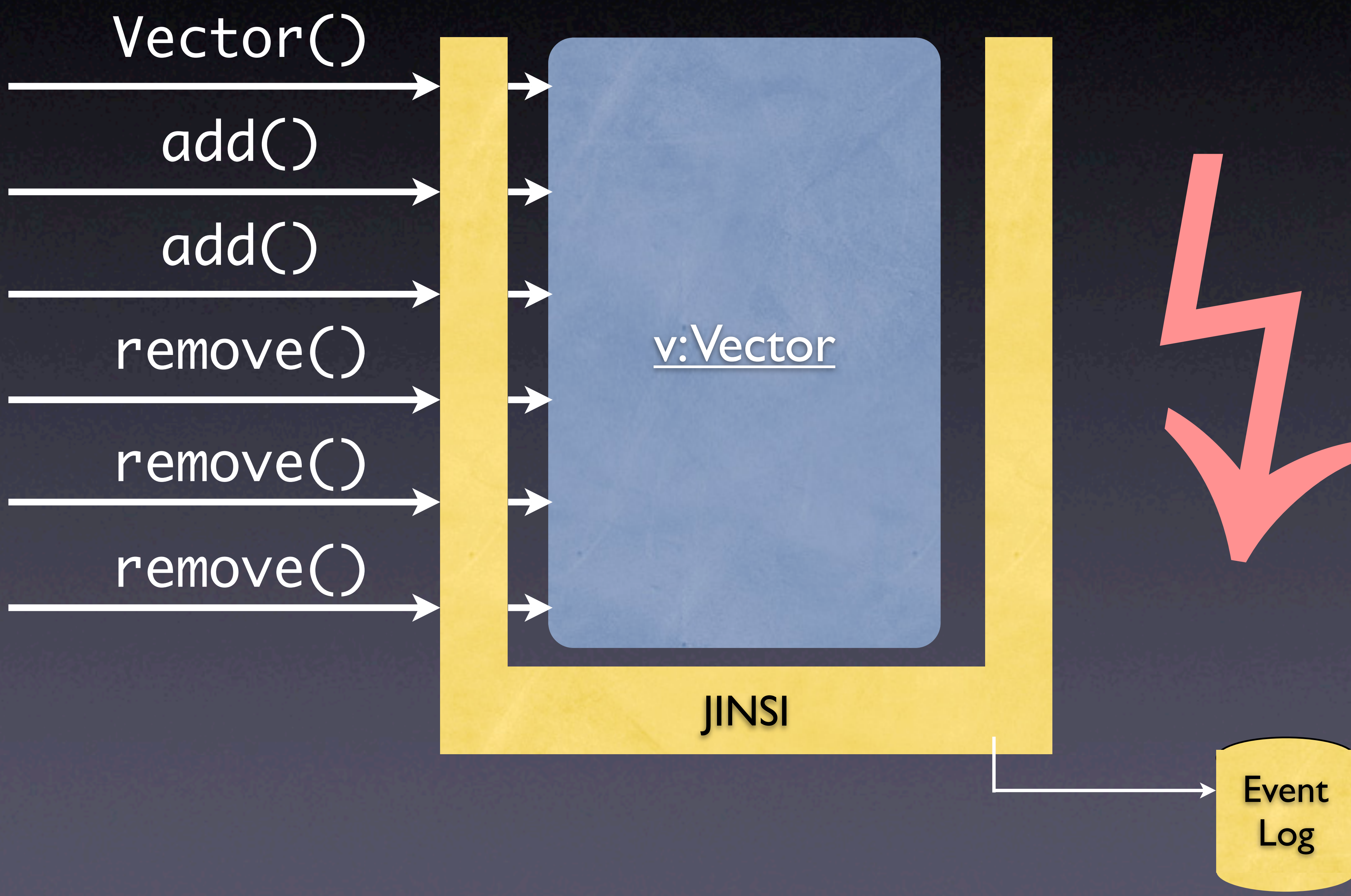

Columba
File Edit View Utilities Help
Add Contact
Add Group
Properties
Remove
Folder Tree
Local Addressbooks
Personal Addressbook
Collected Addresses
Remote Addressbooks
Contact List
Name or email contains:
Clear
Display Name
First Name
Last Name
EMail (Internet)
Website
Columba
http://columba.sourceforge.net/

mozilla.csv

```
Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9358406, +49 681 302 5590, lindig@cs.uni-sb.de
Mehlmann, Martin,,, mehlmann@st.cs.uni-sb.de
Meyer zu Tittingdorf, Friederike, +49 6894 800035, +49 681 302-58099,
meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-5070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6594, t.offergeld@univw.uni-sb.de
PC, CC 2006,,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171825, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796953511, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-58091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813025069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 51165, +49 68130264011,
Security, AG,,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4513, rseidel@cs.uni-sb.de
Sekretariat, Sekretariat,, +49 681 302-64011, office@st.cs.uni-sb.de
Sliwerski, Jacek, +491741333208,, sliwers@st.cs.uni-sb.de
Slusallek, Philipp, +49 6826 1 88 71 32, +49 681 302-3830, slusallek@cs.uni-sb.de
Slusallek USA, Philipp, +1 650 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 582770, +49 681 302-5311, smolka@ps.uni-sb.de
Software-Evolution, AG,,, softevo@st.cs.uni-sb.de
Thiel, Frank,,, hausmeister@cs.uni-sb.de
Weiß, Cathrin,,, weiss@st.cs.uni-sb.de
Wilhelm, Reinhard,, +49 681 302-4399, wilhelm@cs.uni-sb.de
Zeller, Andreas,,, zeller@cs.uni-sb.de
Zeller, Andreas, +49 681 3710465, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 851 51542 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de
```

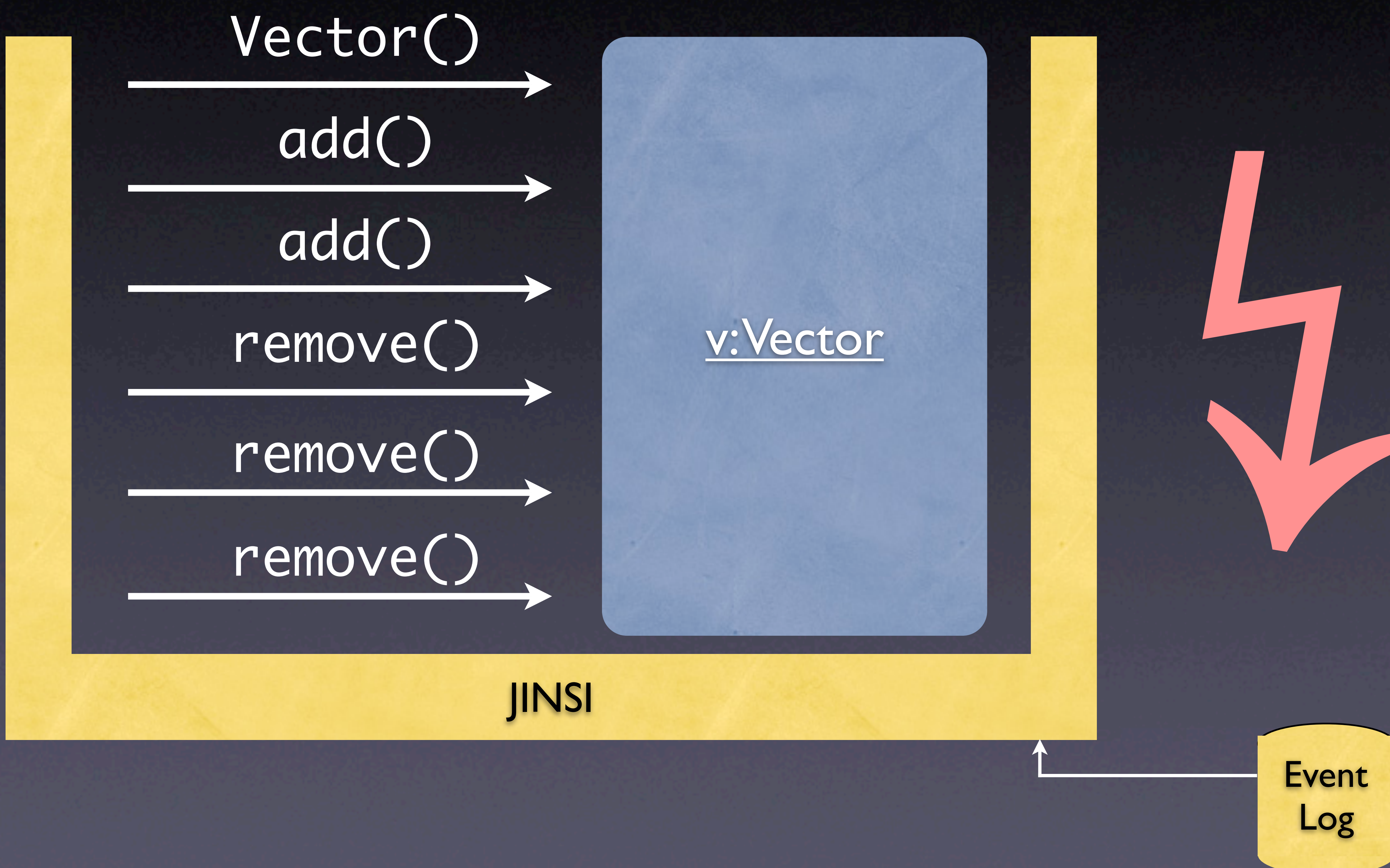
Isolating Relevant Calls

Step 1: Record



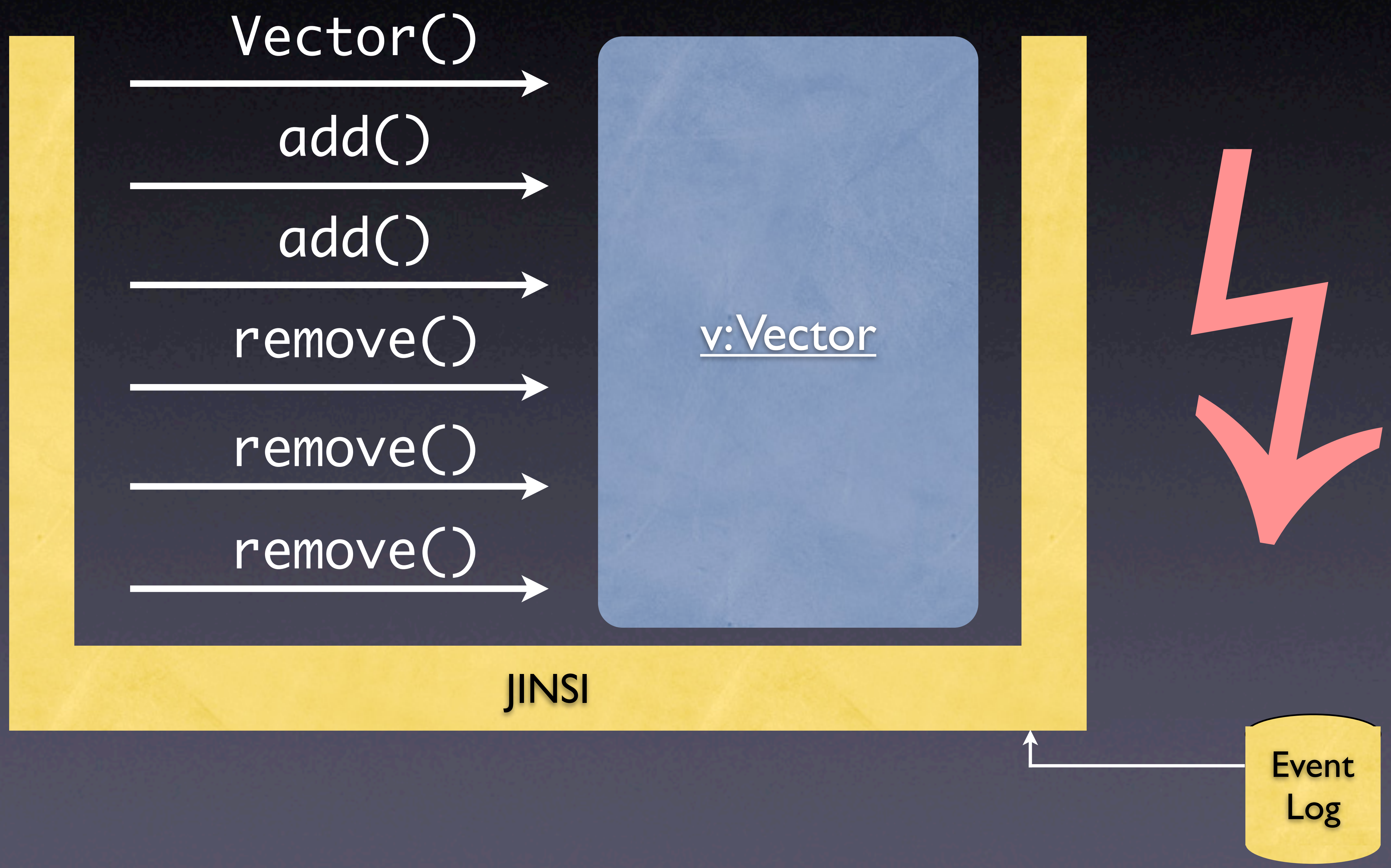
Isolating Relevant Calls

Step 2: Replay



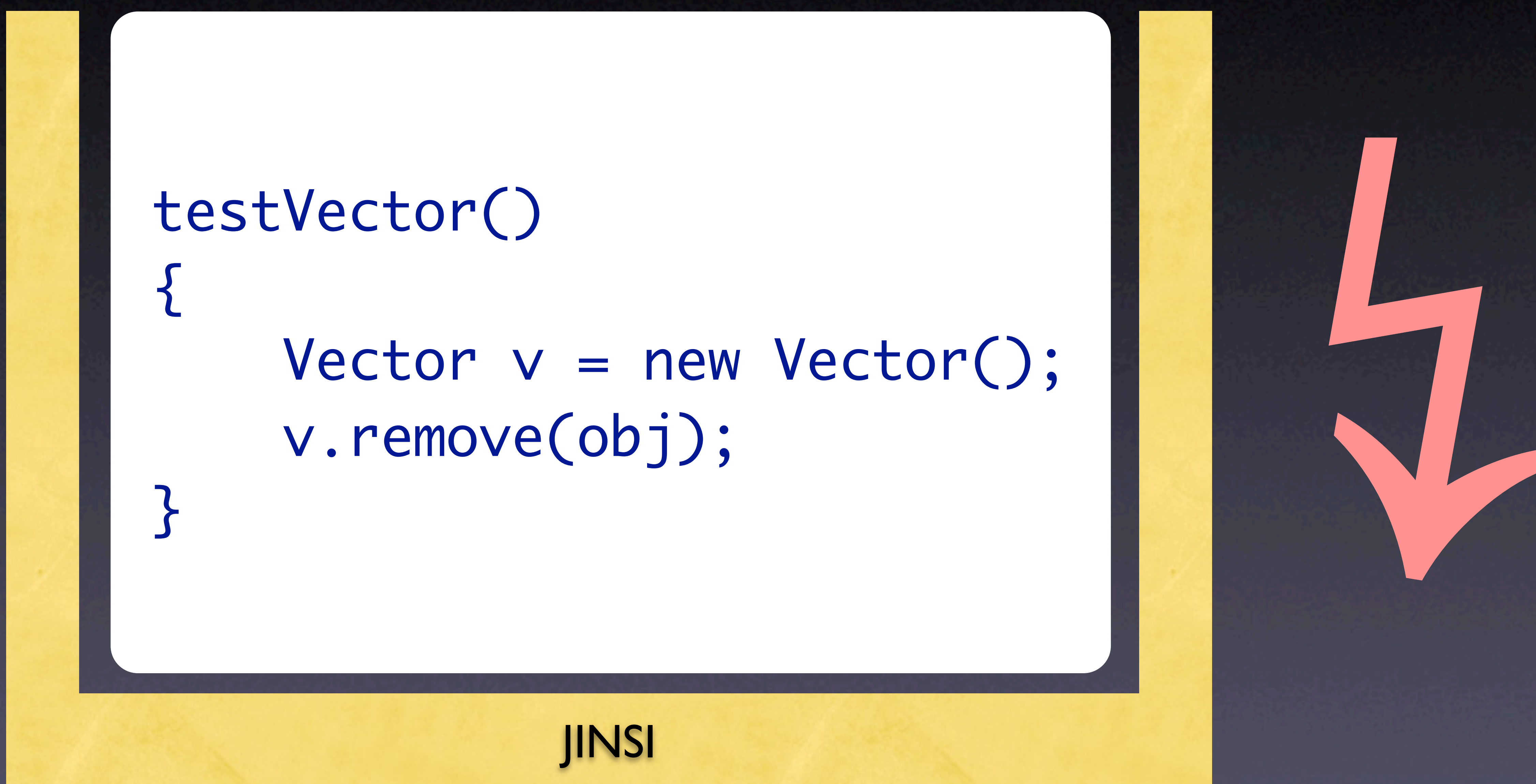
Isolating Relevant Calls

Step 3: Simplify



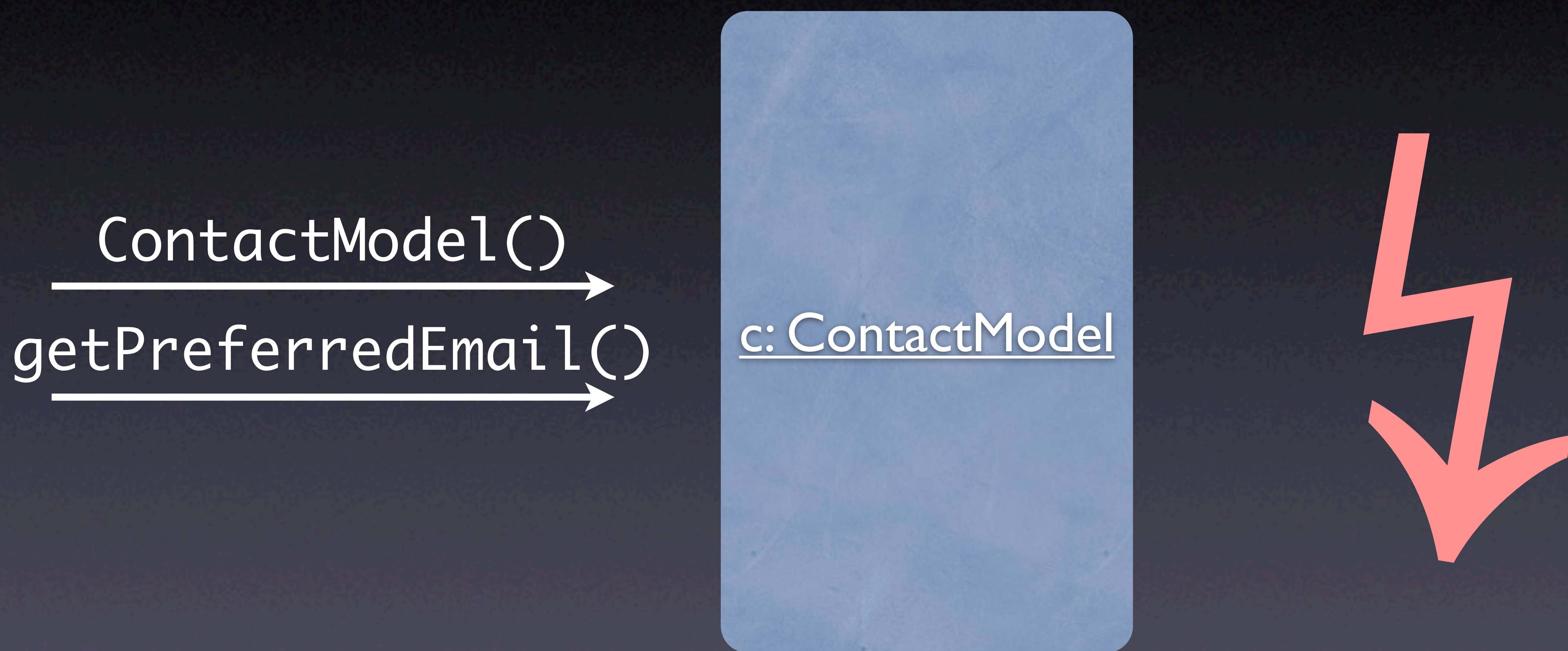
Isolating Relevant Calls

Step 4: Create Unit Test



Columba ContactModel

Columba ContactModel



Unit Test

```
testContactModel()
{
    ContactModel c = new ContactModel();
    String s = c.getPreferredEmail();
}
```

getPreferredEmail

```
public String getPreferredEmail() {
   Iterator it = getEmailIterator();

   // get first item
   IEmailModel model = (IEmailModel) it.next();

   // backwards compatiblity
   // -> its not possible anymore to create a
   // contact model without email address
   if (model == null)
      return null;

   return model.getAddress();
}
```

Failure Cause

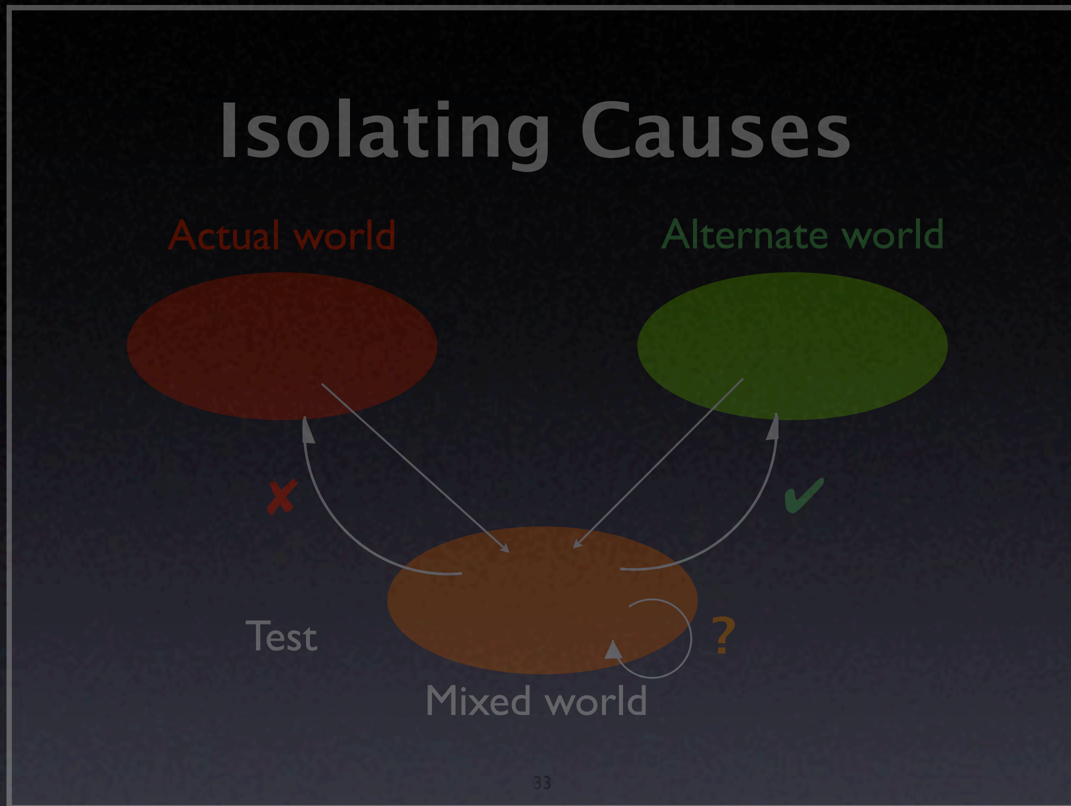
Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9378406, +49 681 302 7790, lindig@cs.uni-sb.de
Mehlmann, Martin,,, mehlmann@st.cs.uni-sb.de
Meyer zu Tittingdorf, Friederike, +49 6894 890037, +49 681 302-78099, meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-7070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6794, t.offergeld@univw.uni-sb.de
PC, CC 2006,,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171827, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796973711, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-78091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813027069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 71167, +49 68130264011,
Security, AG,,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4713, rseidel@cs.uni-sb.de
Sekretariat, Sekretariat,, +49 681 302-64011, office@st.cs.uni-sb.de
Sliwerski, Jacek, +491741333208,, sliwers@st.cs.uni-sb.de
Slusallek, Philipp, +49 6826 1 88 71 32, +49 681 302-3830, slusallek@cs.uni-sb.de
Slusallek USA, Philipp, +1 670 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 782770, +49 681 302-7311, smolka@ps.uni-sb.de
Software-Evolution, AG,,, softevo@st.cs.uni-sb.de
Thiel, Frank,,, hausmeister@cs.uni-sb.de
Weiß, Cathrin,,, weiss@st.cs.uni-sb.de
Wilhelm, Reinhard,, +49 681 302-4399, wilhelm@cs.uni-sb.de
Zeller, Andreas,,, zeller@cs.uni-sb.de
Zeller, Andreas, +49 681 3710467, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 871 71742 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de

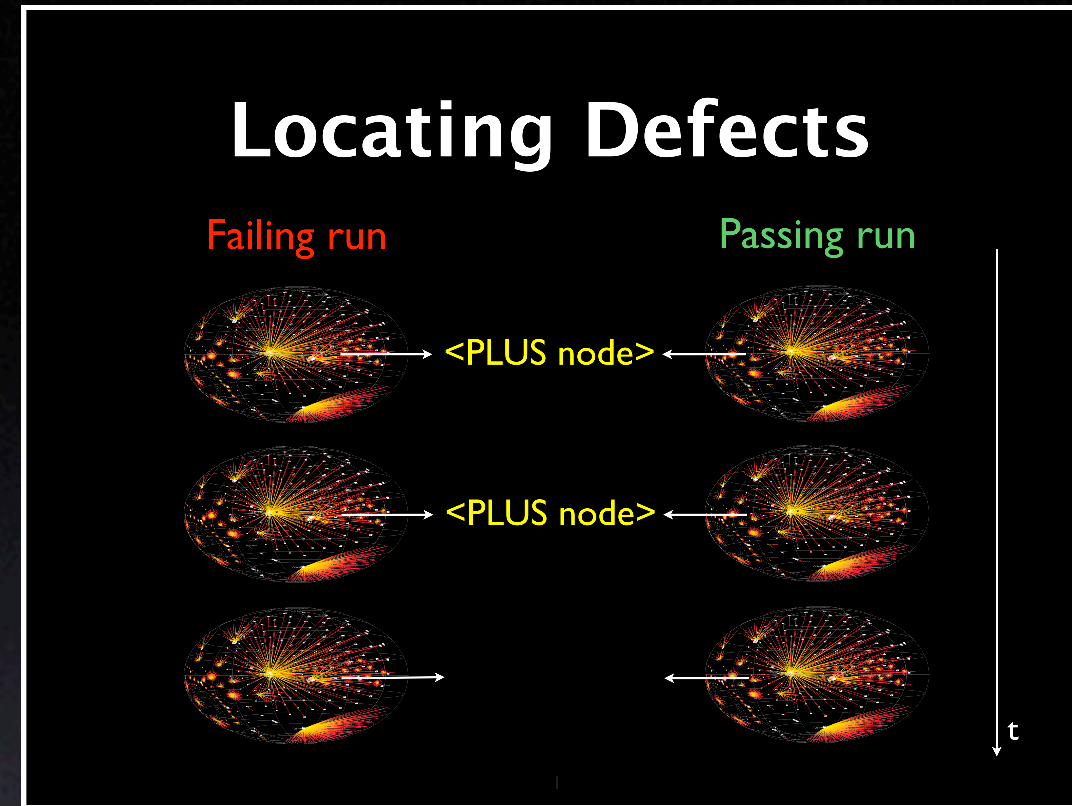
Failure Cause

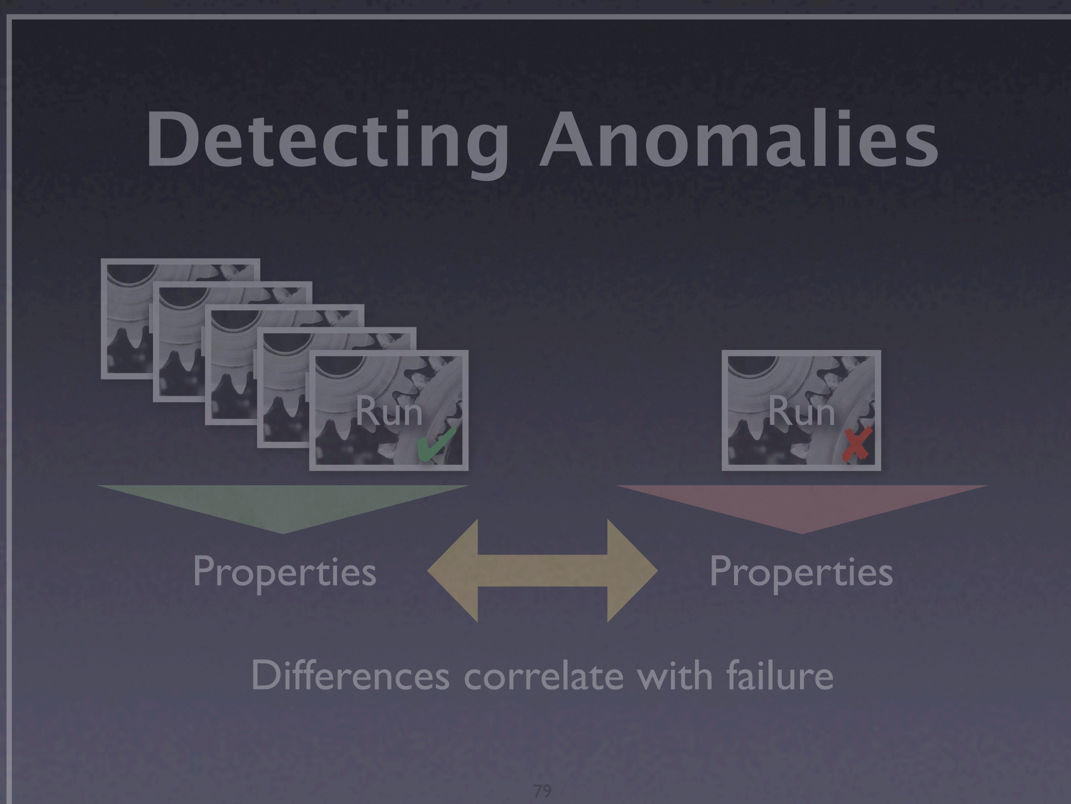
Lenhof, Hans-Peter,, +49 681 302-64701, lenhof@cs.uni-sb.de
Lindig, Christian, +49 681 9378406, +49 681 302 7790, lindig@cs.uni-sb.de
Mehlmann, Martin,,, me[illegible]mann@st.cs.uni-sb.de
Meyer zu Tittingdorf, [illegible]94 89005[illegible]302-78099,
meyer@cs.uni-sb.de
Mileva, Yana,, +49 681 302-64020, mileva@cs.uni-sb.de
Müller-Perich, Elisabeth,, +49 681 302-7070, sekr.techfak@rz.uni-sb.de
Nir-Bleimling, Naomi,, +49 68130264011, naomi@wjpserver.cs.uni-sb.de
Offergeld, Thilo,, +49 681 302-6794, t.offergeld@univw.uni-sb.de
PC, CC 2006,,, cc2006pc@st.cs.uni-sb.de
Paul, Wolfgang, +4968171827, +49 6813022436, wjp@cs.uni-sb.de
Premraj, Rahul, +44 7796973711, +49 681 302-64013, premraj@cs.uni-sb.de
Reindel, Erich, +49 6371 912842, +49 681 302-78091, reindel@cs.uni-sb.de
Schuler, David,, +49 681 302-64026, schuler@st.cs.uni-sb.de
Schuler, Erika,, +49 6813027069, schuler@tf.uni-sb.de
Schäfer, Christa, +49 6897 71167, +49 68130264011,
Security, AG,,, security@st.cs.uni-sb.de
Seidel, Raimund, +49 6894 383698, +49 681 302-4713, rseidel@cs.uni-sb.de
Sekretariat, Sekretariat,, +49 681 302-64011, office@st.cs.uni-sb.de
Sliwerski, Jacek, +491741333208,, sliwers@st.cs.uni-sb.de
Slusallek, Philipp, +49 6826 1 88 71 32, +49 681 302-3830, slusallek@cs.uni-sb.de
Slusallek USA, Philipp, +1 670 391 9186, +1 408 486 2788, slusallek@cs.uni-sb.de
Smolka, Gert, +49 681 782770, +49 681 302-7311, smolka@ps.uni-sb.de
Software-Evolution, AG,,, softevo@st.cs.uni-sb.de
Thiel, Frank,,, hausmeister@cs.uni-sb.de
Weiß, Cathrin,,, weiss@st.cs.uni-sb.de
Wilhelm, Reinhard,, +49 681 302-4399, wilhelm@cs.uni-sb.de
Zeller, Andreas,,, zeller@cs.uni-sb.de
Zeller, Andreas, +49 681 3710467, +49 681 302-64011, zeller@cs.uni-sb.de
Zimmermann, Tom, +49 871 71742 (Eltern), +1 403 210 9470, zimmerth@cs.uni-sb.de

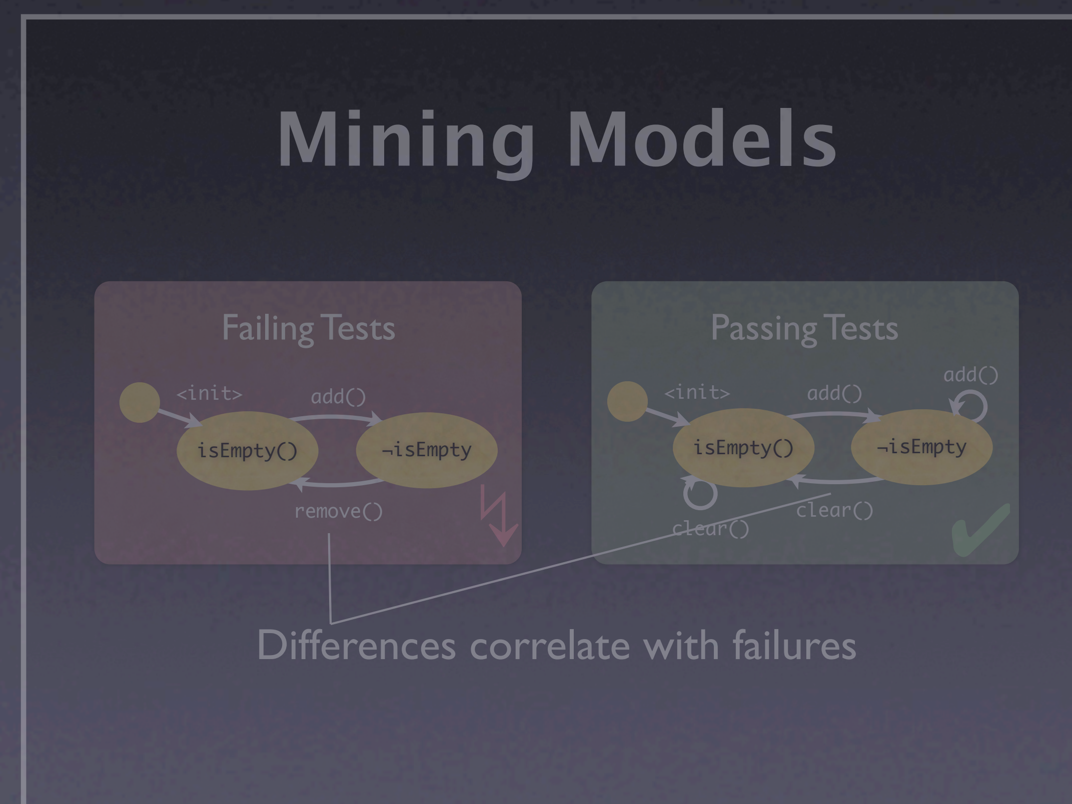
Conclusion

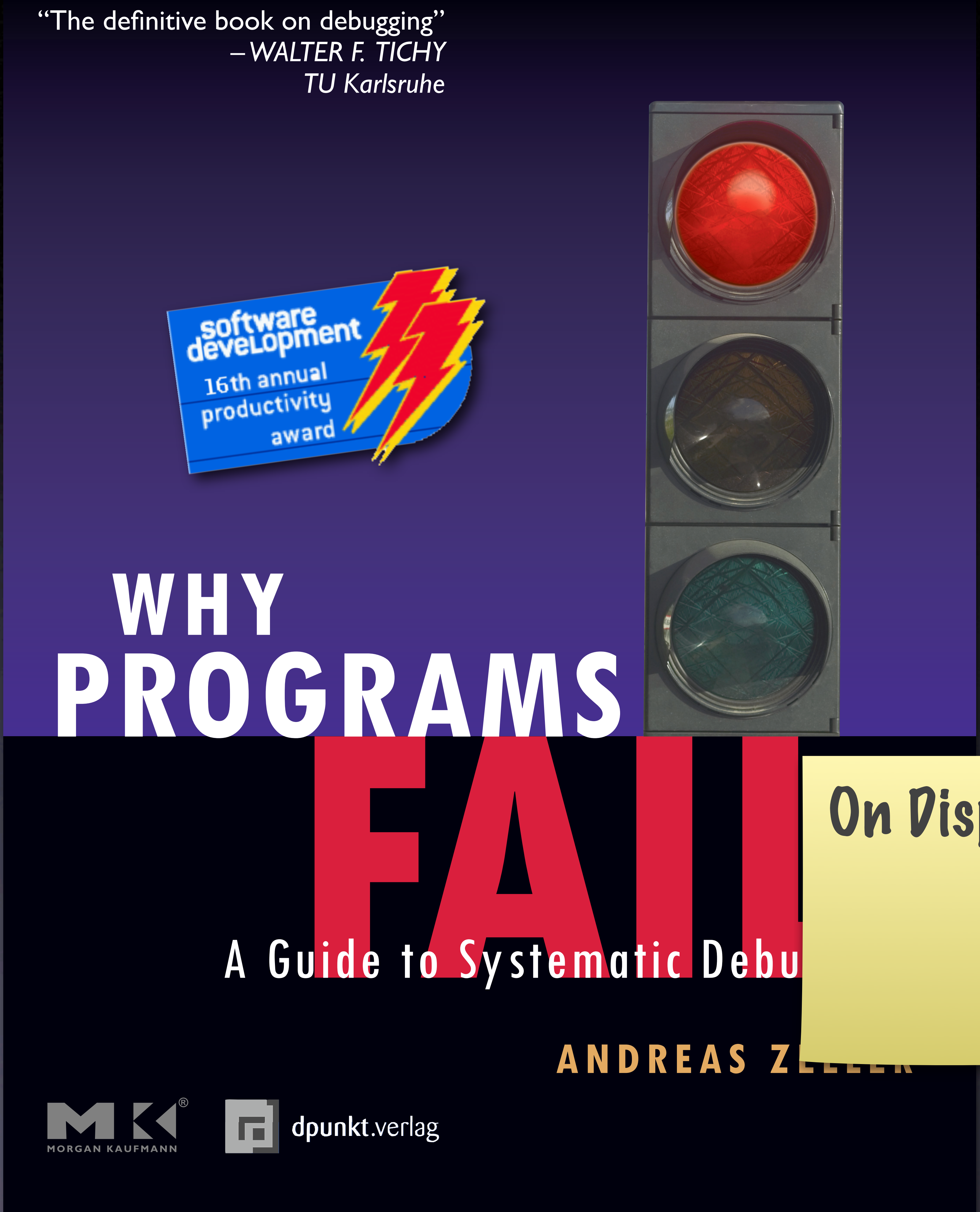
- Delta debugging on *states* is nice, but cumbersome
- Alternative: delta debugging on *object interaction*
- First promising results with JINSI prototype
- Combined solution for reproducing + simplifying


Isolating Causes
Actual world
Alternate world
Test
Mixed world
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t

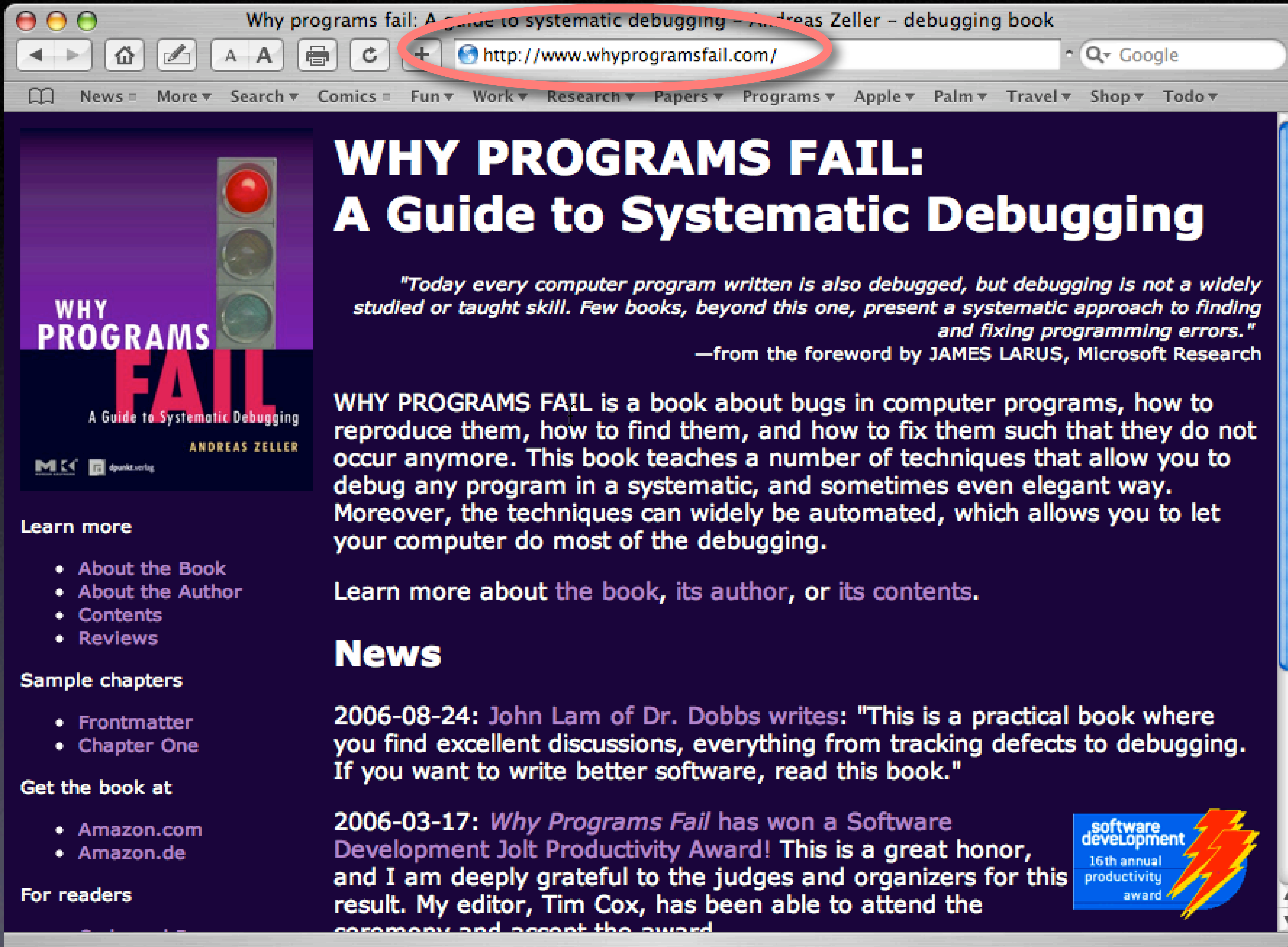

Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79


Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures


"The definitive book on debugging"
–WALTER F. TICHY
TU Karlsruhe
software development
16th annual productivity award
WHY
PROGRAMS
FAI
A Guide to Systematic Debu
ANDREAS Z
MORGAN KAUFMANN
dpunkt.verlag

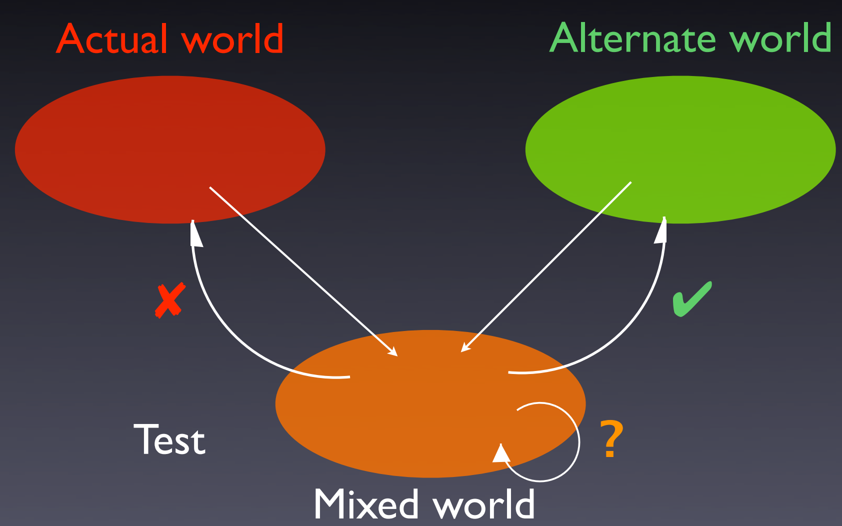
On Display!

#1 in
Software Engineering Books
Algorithms Books
Software Design Books

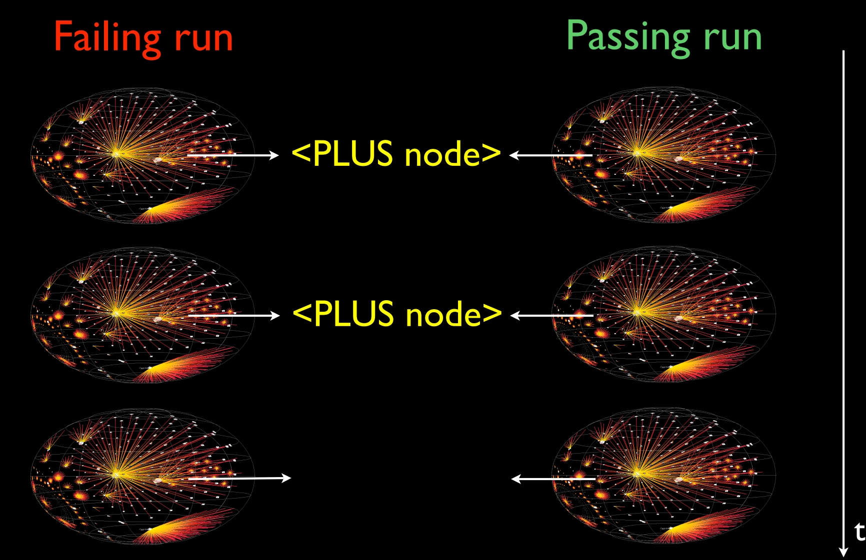


Why programs fail: A guide to systematic debugging – Andreas Zeller – debugging book
http://www.whyprogramsfail.com/
Google
News More Search Comics Fun Work Research Papers Programs Apple Palm Travel Shop Todo
WHY PROGRAMS FAIL
A Guide to Systematic Debugging
ANDREAS ZELLER
WHY PROGRAMS FAIL:
A Guide to Systematic Debugging
"Today every computer program written is also debugged, but debugging is not a widely studied or taught skill. Few books, beyond this one, present a systematic approach to finding and fixing programming errors."
—from the foreword by JAMES LARUS, Microsoft Research
WHY PROGRAMS FAIL is a book about bugs in computer programs, how to reproduce them, how to find them, and how to fix them such that they do not occur anymore. This book teaches a number of techniques that allow you to debug any program in a systematic, and sometimes even elegant way. Moreover, the techniques can widely be automated, which allows you to let your computer do most of the debugging.
Learn more about the book, its author, or its contents.
News
2006-08-24: John Lam of Dr. Dobbs writes: "This is a practical book where you find excellent discussions, everything from tracking defects to debugging. If you want to write better software, read this book."
2006-03-17: Why Programs Fail has won a Software Development Jolt Productivity Award! This is a great honor, and I am deeply grateful to the judges and organizers for this result. My editor, Tim Cox, has been able to attend the
software development
16th annual productivity award
Learn more
About the Book
About the Author
Contents
Reviews
Sample chapters
Frontmatter
Chapter One
Get the book at
Amazon.com
Amazon.de
For readers

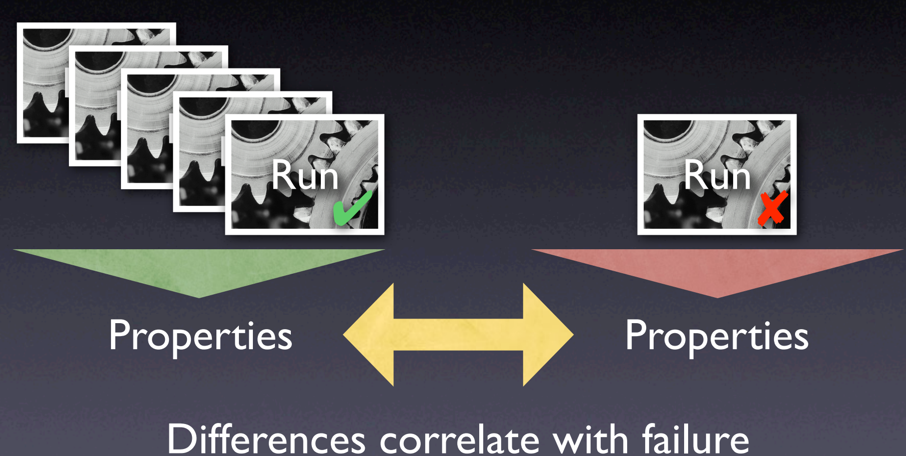
Isolating Causes



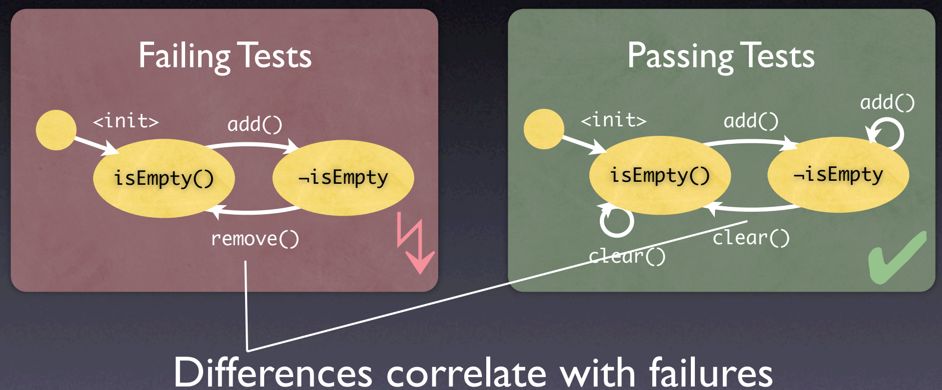
Locating Defects

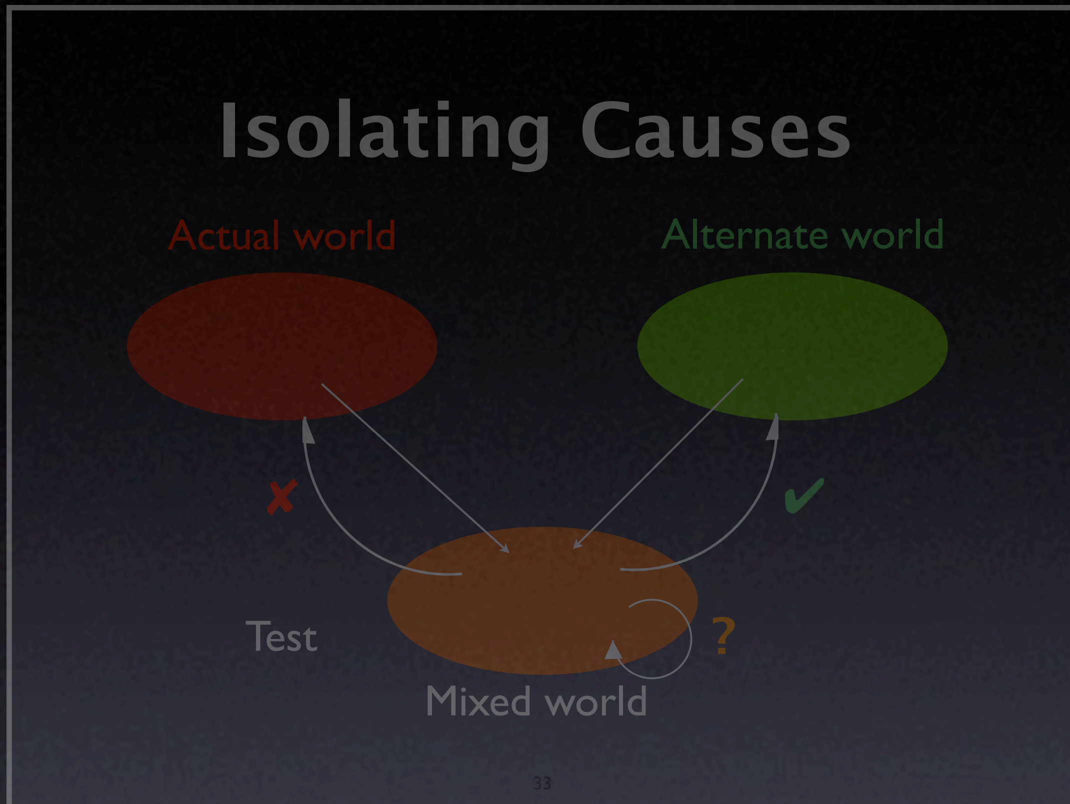


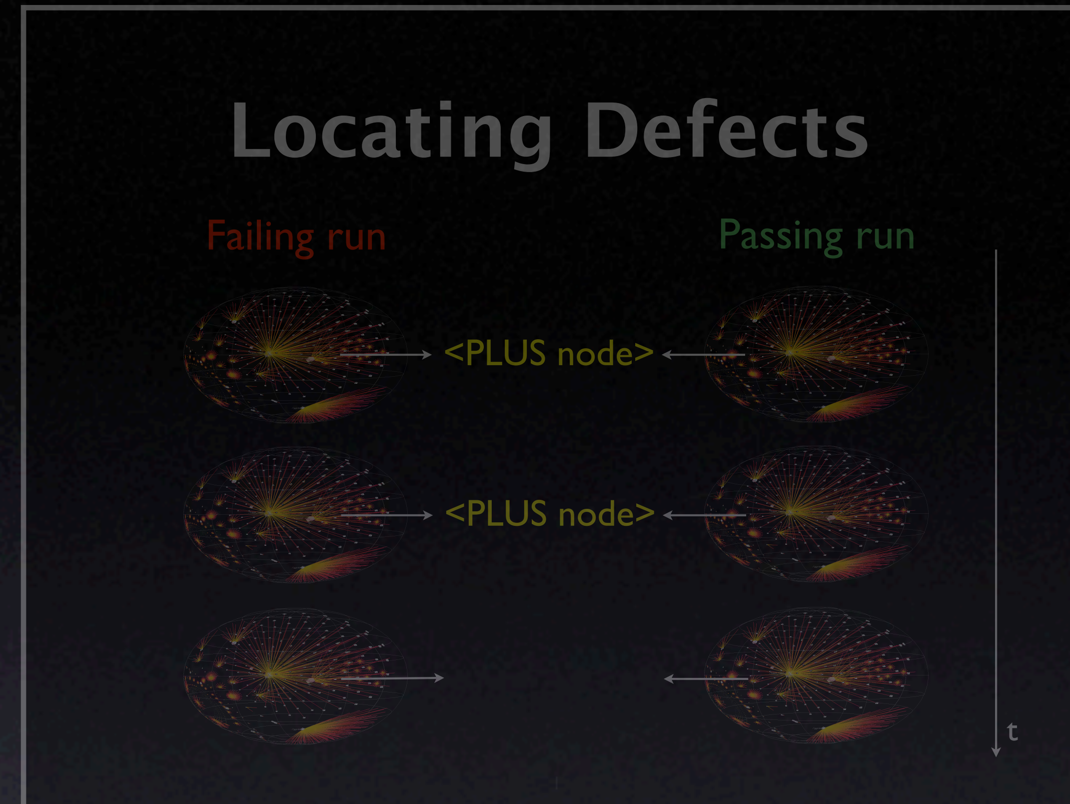
Detecting Anomalies

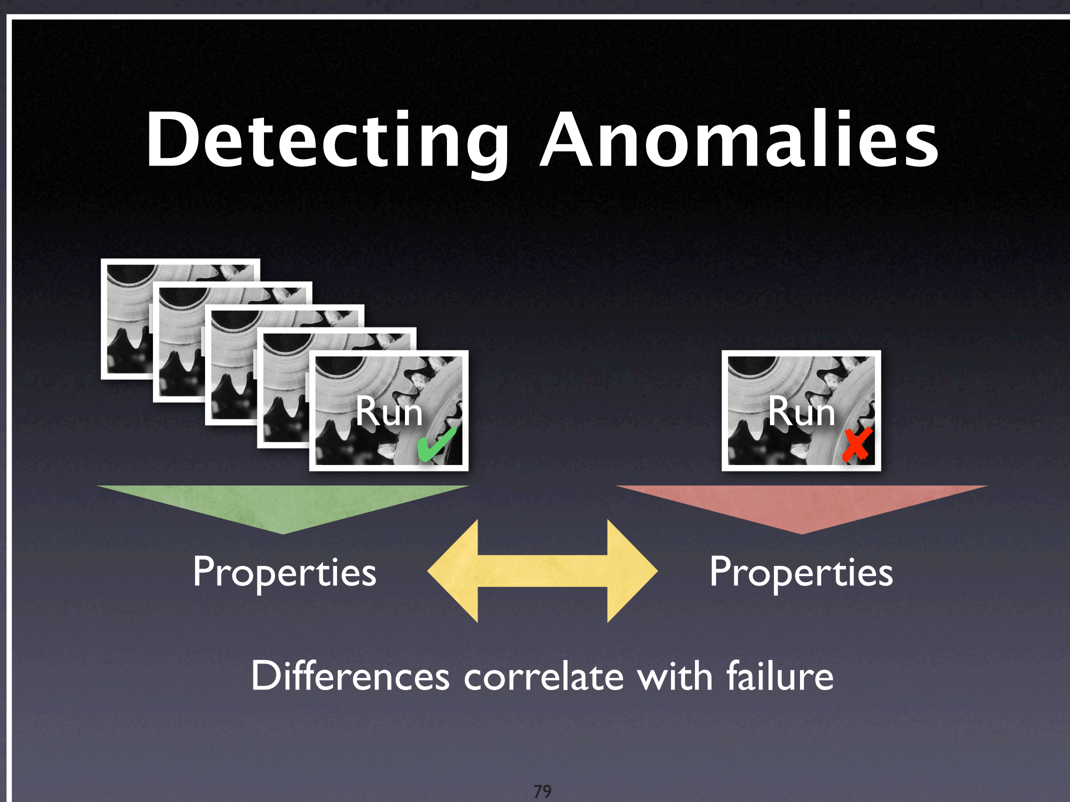


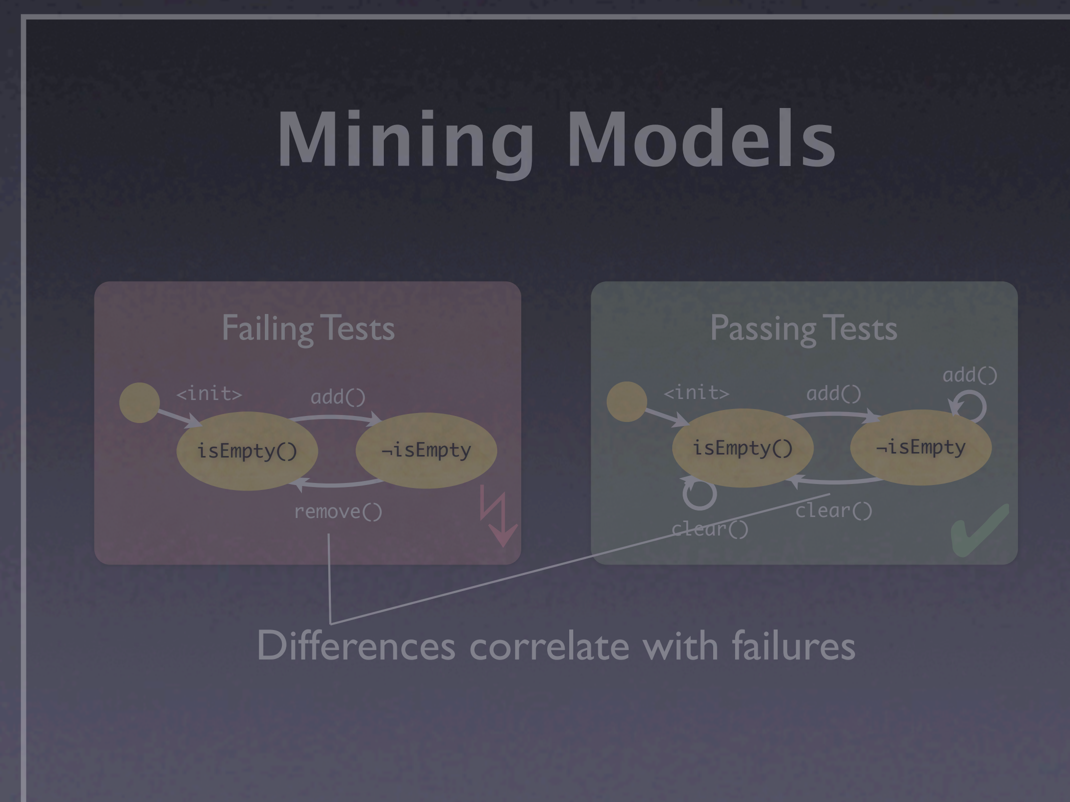
Mining Models



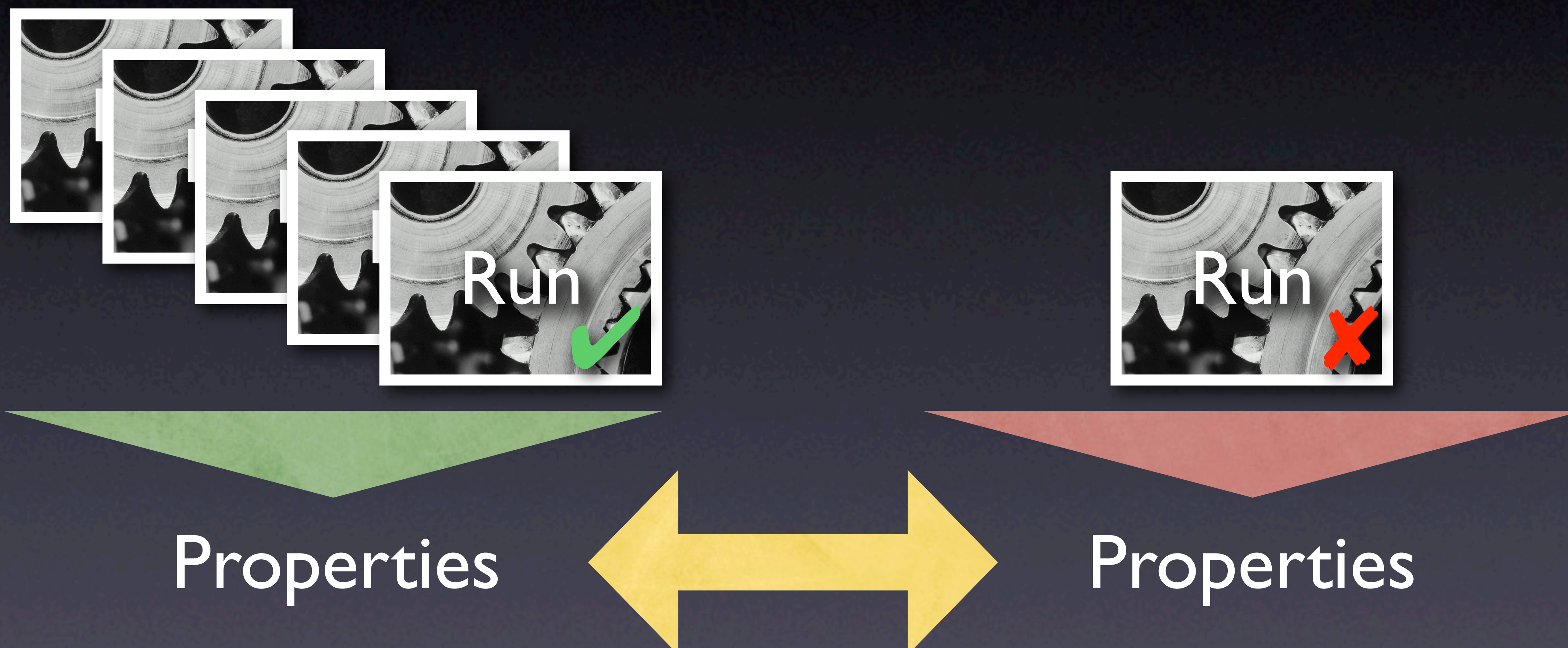

Isolating Causes
Actual world
Alternate world
Test
Mixed world
?
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t


Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79


Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures

Detecting Anomalies



Differences correlate with failure

Properties

Coverage	Sequences	Returns

Properties

Coverage	Sequences	Returns

Comparing Coverage

1. Every failure is caused by an infection, which in turn is caused by a defect
2. The defect must be *executed* to start the infection
3. Code that is executed *in failing runs only* is thus likely to cause the defect

The middle program

```
$ middle 3 3 5
middle: 3

$ middle 2 1 3
middle: 1
```

```
int main(int arc, char *argv[])
{
    int x = atoi(argv[1]);
    int y = atoi(argv[2]);
    int z = atoi(argv[3]);
    int m = middle(x, y, z);

    printf("middle: %d\n", m);

    return 0;
}
```

```
int middle(int x, int y, int z) {
    int m = z;
    if (y < z) {
        if (x < y)
            m = y;
        else if (x < z)
            m = y;
    } else {
        if (x > y)
            m = y;
        else if (x > z)
            m = x;
    }
    return m;
}
```

Obtaining Coverage

for C programs

Obtaining Coverage

for Python programs

```
if __name__ == "__main__":
    sys.settrace(tracer)
    x = sys.argv[1]
    y = sys.argv[2]
    z = sys.argv[3]
    m = middle(x, y, z)

    print "middle:", m
```

Obtaining Coverage

for Python programs

```
def tracer(frame, event, arg):
    code = frame.f_code
    function = code.co_name
    filename = code.co_filename
    line = frame.f_lineno
    print filename + ":" + `line` + \
        ":" + function + "():", \
        event, arg
    return tracer
```

Obtaining Coverage

for Python programs

```
$ ./middle.py 3 3 5
./middle.py:13:middle(): call None
./middle.py:14:middle(): line None
./middle.py:15:middle(): line None
./middle.py:16:middle(): line None
./middle.py:18:middle(): line None
./middle.py:19:middle(): line None
./middle.py:26:middle(): line None
./middle.py:26:middle(): return 3
middle: 3
```

x	3	1	3	5	5	2
y	3	2	2	5	3	1
z	5	3	1	5	4	3
`int middle(int x, int y, int z) {`	●	●	●	●	●	●
`    int m = z;`	●	●	●	●	●	●
`    if (y < z) {`	●	●	●	●	●	●
`        if (x < y)`		●				
`            m = y;`		●				
`        else if (x < z)`	●				●	●
`            m = y;`	●					●
`    } else {`	●		●	●		
`        if (x > y)`			●			
`            m = y;`			●			
`        else if (x > z)`						
`            m = x;`						
`    }`						
`    return m;`	●	●	●	●	●	●
`}`	✔	✔	✔	✔	✔	✘

Discrete Coloring

executed only in failing runs
highly suspect

executed in passing and failing runs
ambiguous

executed only in passing runs
likely correct

x	3	1	3	5	5	2
y	3	2	2	5	3	1
z	5	3	1	5	4	3
`int middle(int x, int y, int z) {`	●	●	●	●	●	●
`int m = z;`	●	●	●	●	●	●
`if (y < z) {`	●	●	●	●	●	●
`if (x < y)`		●				
`m = y;`		●				
`else if (x < z)`	●				●	●
`m = y;`	●					●
`} else {`	●		●	●		
`if (x > y)`			●			
`m = y;`			●			
`else if (x > z)`						
`m = x;`						
`}`						
`return m;`	●	●	●	●	●	●
`}`	✔	✔	✔	✔	✔	✘

x	3	1	3	5	5	2
y	3	2	2	5	3	1
z	5	3	1	5	4	3
`int middle(int x, int y, int z) {`	●	●	●	●	●	●
`int m = z;`	●	●	●	●	●	●
`if (y < z) {`	●	●	●	●	●	●
`if (x < y)`		●				
`m = y;`		●				
`else if (x < z)`	●				●	●
`m = y;`	●					●
`} else {`	●		●	●		
`if (x > y)`			●			
`m = y;`			●			
`else if (x > z)`						
`m = x;`						
`}`						
`return m;`	●	●	●	●	●	●
`}`	✔	✔	✔	✔	✔	✘

Continuous Coloring

executed only in failing runs

passing and failing runs

executed only in passing runs

Hue

$$hue(s) = red\ hue + \frac{\%passed(s)}{\%passed(s) + \%failed(s)} \times hue\ range$$

0% passed  100% passed

Brightness

frequently executed

$$bright(s) = \max(\%passed(s), \%failed(s))$$

rarely executed

x	3	1	3	5	5	2
y	3	2	2	5	3	1
z	5	3	1	5	4	3
`int middle(int x, int y, int z) {`	●	●	●	●	●	●
`    int m = z;`	●	●	●	●	●	●
`    if (y < z) {`	●	●	●	●	●	●
`        if (x < y)`		●				
`            m = y;`		●				
`        else if (x < z)`	●				●	●
`            m = y;`	●					●
`    } else {`	●		●	●		
`        if (x > y)`			●			
`            m = y;`			●			
`        else if (x > z)`						
`            m = x;`						
`    }`						
`    return m;`	●	●	●	●	●	●
`}`	✔	✔	✔	✔	✔	✘

Source: Jones et al., ICSE 2002

Source: Jones et al., ICSE 2002

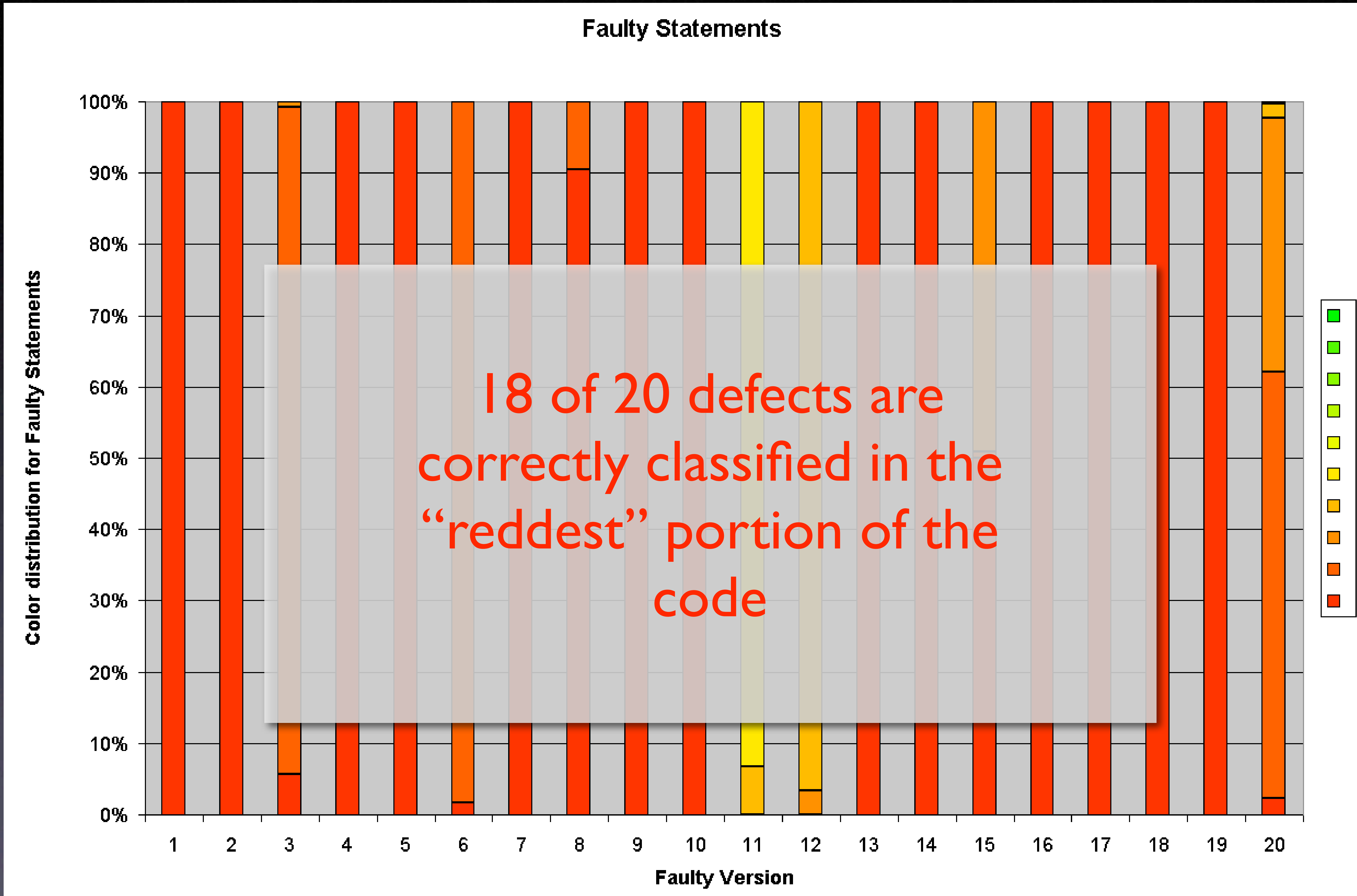
Evaluation

How well does comparing coverage detect anomalies?

- How green are the defects? *(false negatives)*
- How red are non-defects? *(false positives)*

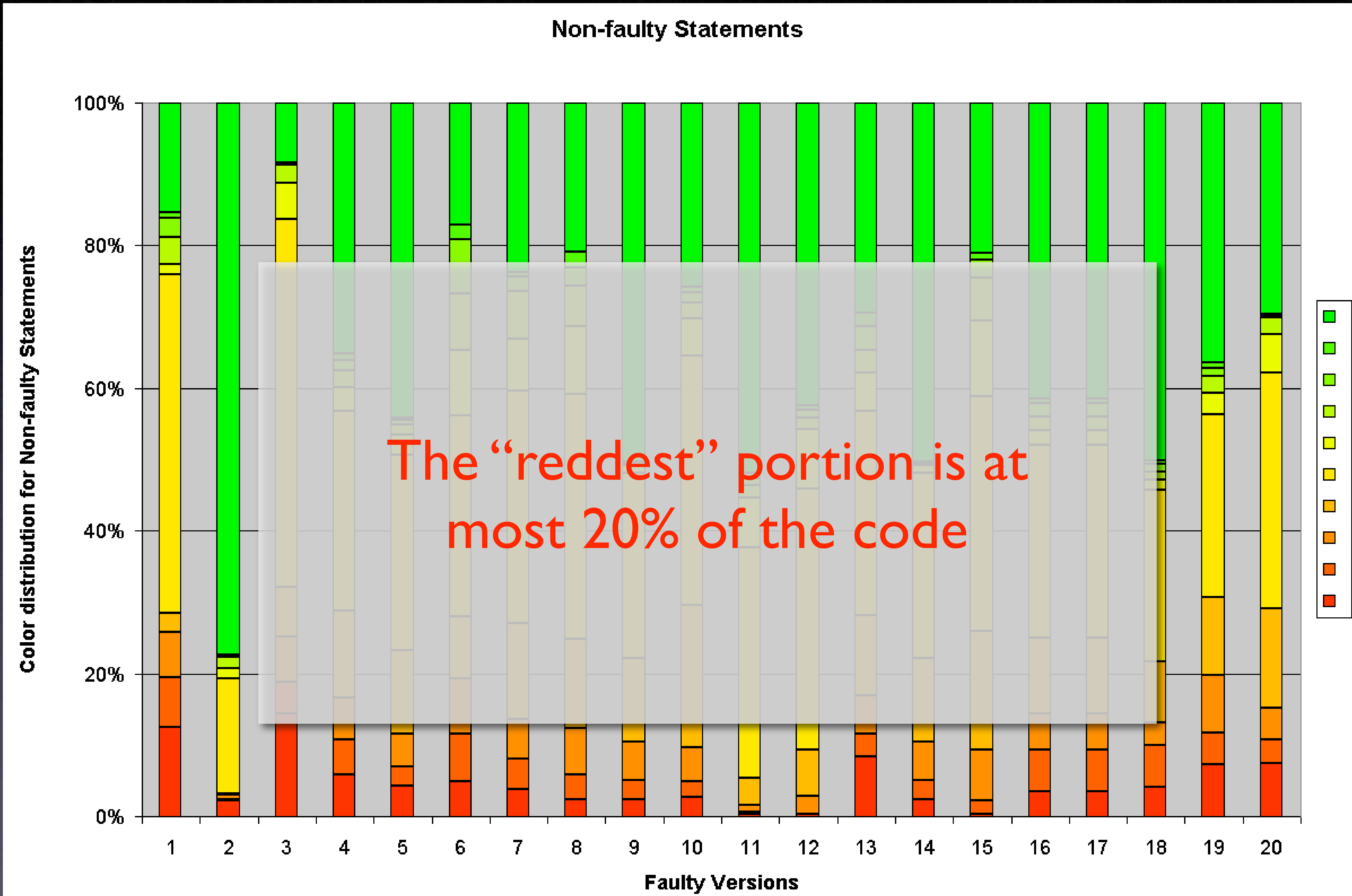
Space

- 8000 lines of executable code
- 1000 test suites with156 to 4700 test cases
- 20 defective versions with one defect each (corrected in subsequent version)



18 of 20 defects are correctly classified in the “reddest” portion of the code

Source: Jones et al., ICSE 2002



Source: Jones et al., ICSE 2002

Siemens Suite

- 7 C programs, 170–560 lines
- 132 variations with one defect each
- 108 all yellow (i.e., useless)
- 1 with one red statement (at the defect)

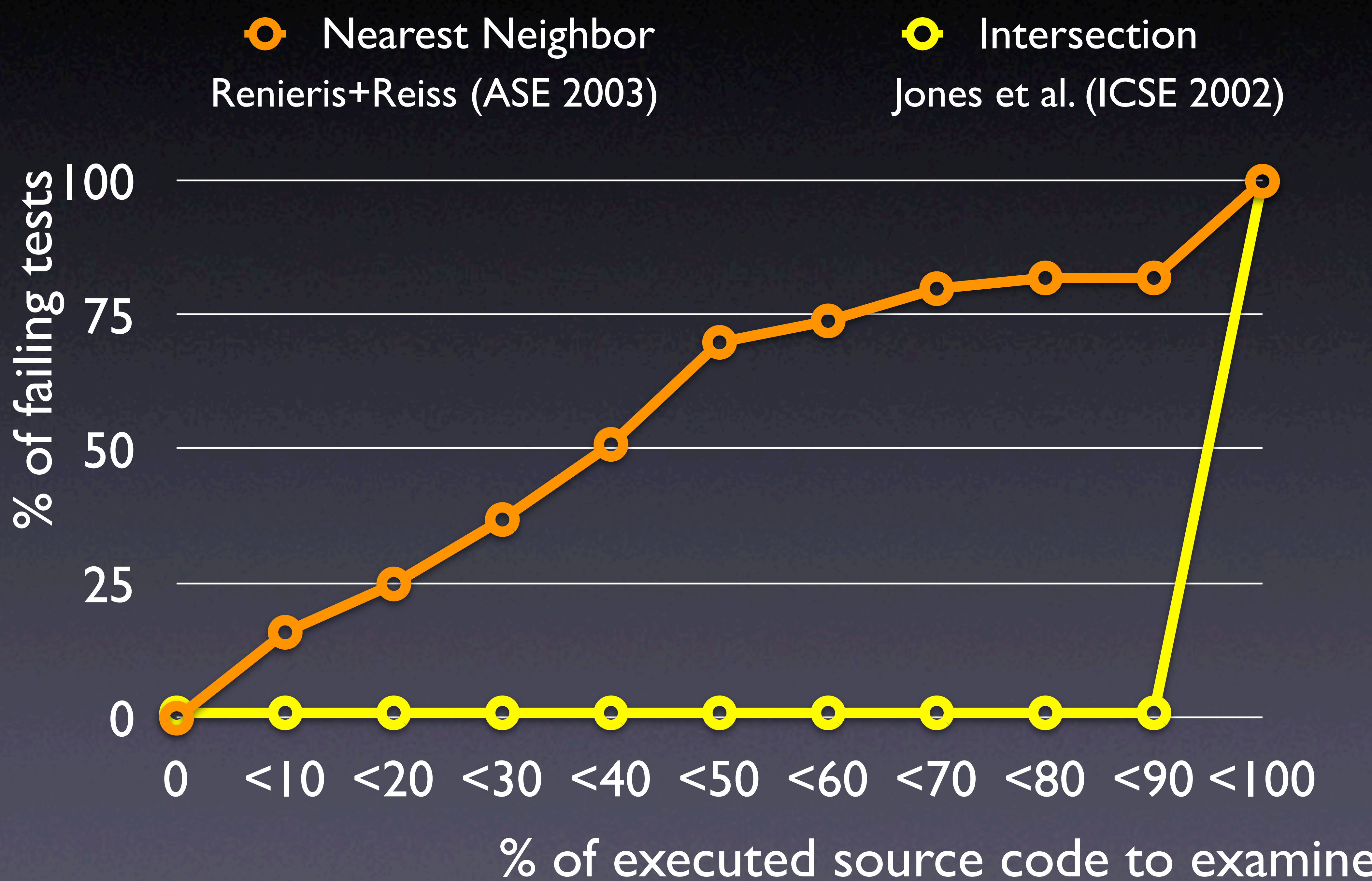
Source: Renieris and Reiss, ASE 2003

Nearest Neighbor

Nearest Neighbor

Compare with the single run
that has the most similar coverage

Locating Defects



Results obtained from Siemens test suite; can not be generalized

Properties

Coverage	Sequences	Returns

Sequences

Sequences of locations can correlate with failures:

open() read() close()	✔
open() close() read()	✘
close() open() read()	✘

…but all locations are executed in both runs!

The AspectJ Compiler

```
$ ajc Test3.aj
$ java test.Test3
test.Test3@b8df17.x Unexpected Signal : 11
occurred at PC=0xFA415A00
Function name=(N/A) Library=(N/A) ...
Please report this error at http://
java.sun.com/...
$
```

Coverage Differences

- Compare the failing run with passing runs
- `BcelShadow.getThisJoinPointVar()` is invoked in the failing run only
- Unfortunately, this method is correct

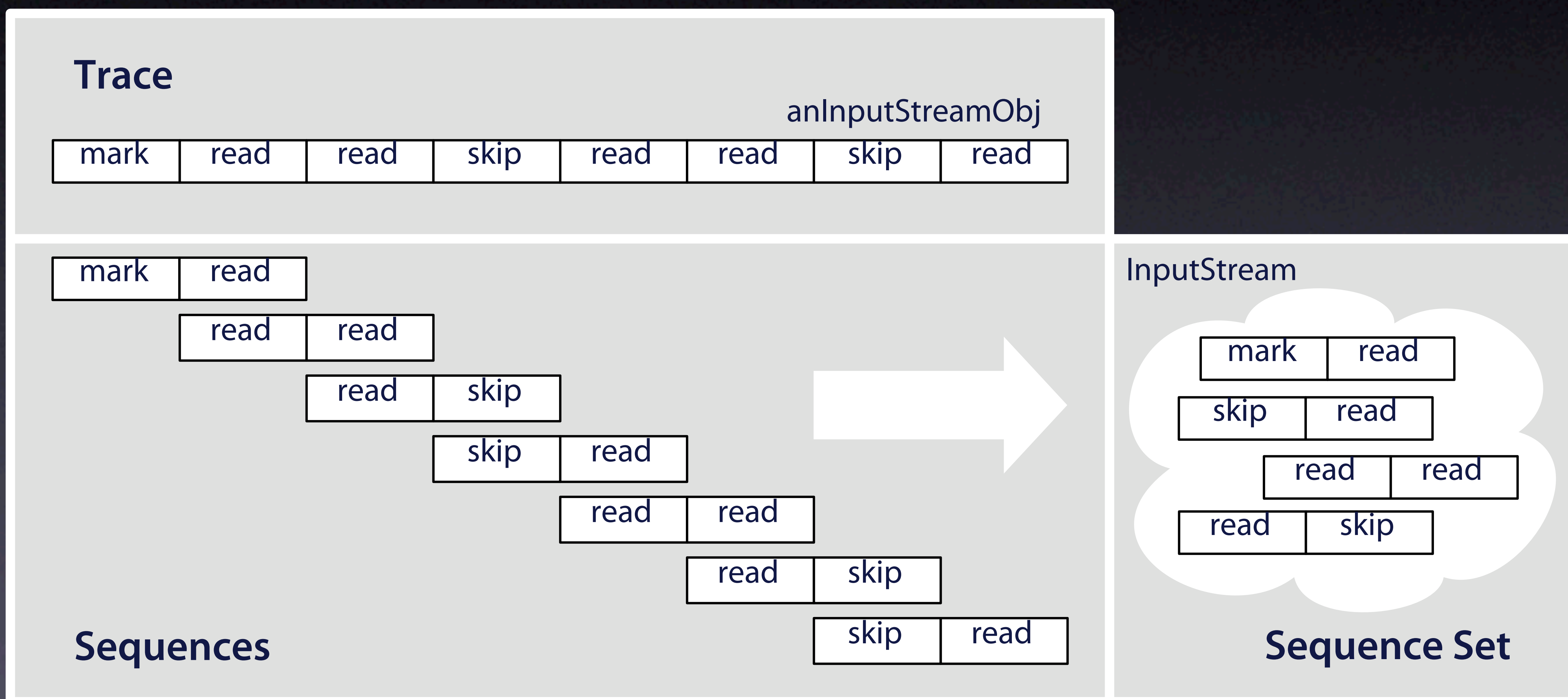
Sequence Differences

This *sequence* occurs only in the failing run:

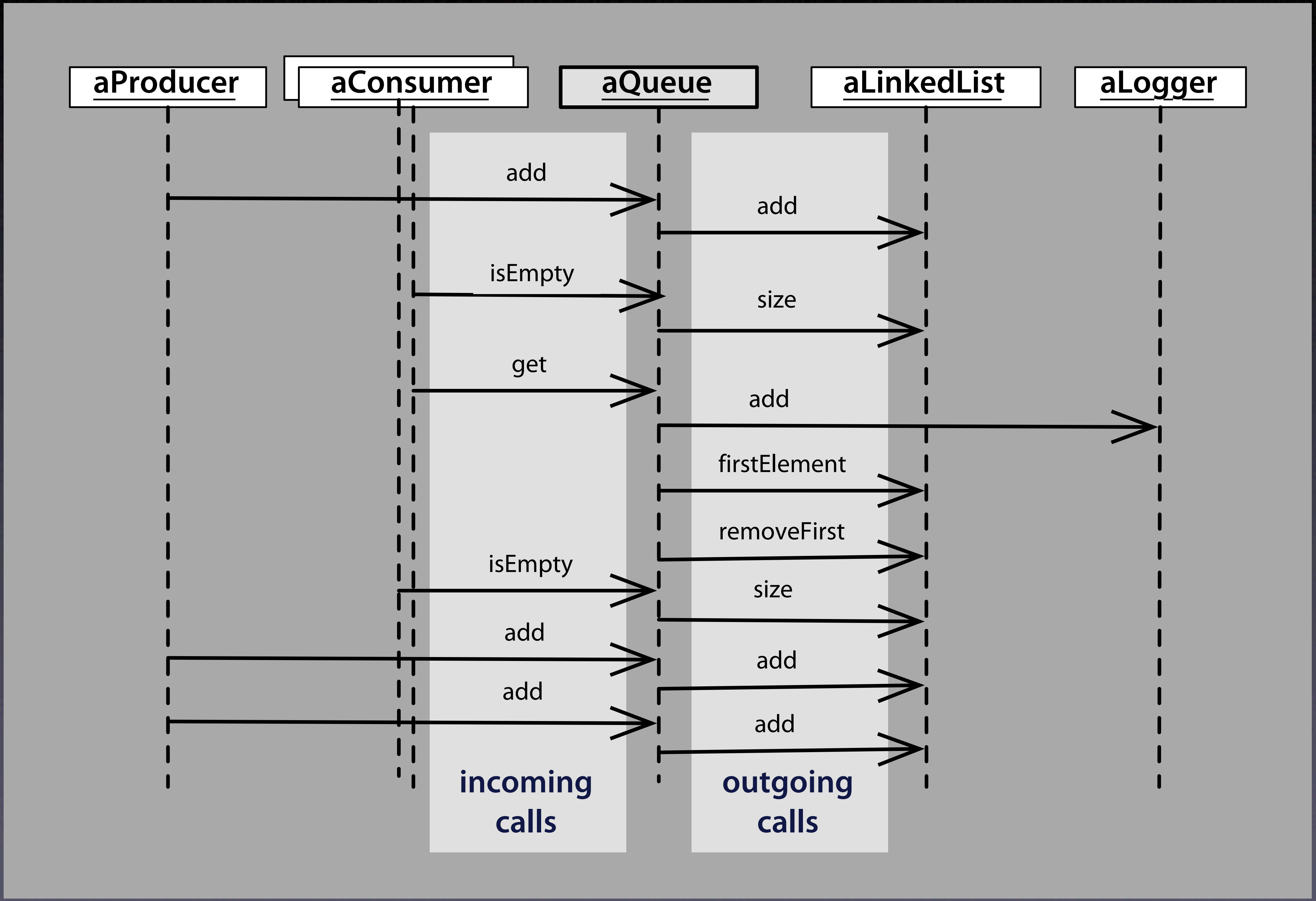
```
⟨ ThisJoinPointVisitor.isRef(),
  ThisJoinPointVisitor.canTreatAsStatic(),
  MethodDeclaration.traverse(),
  ThisJoinPointVisitor.isRef(),
  ThisJoinPointVisitor.isRef() ⟩
```

Defect location

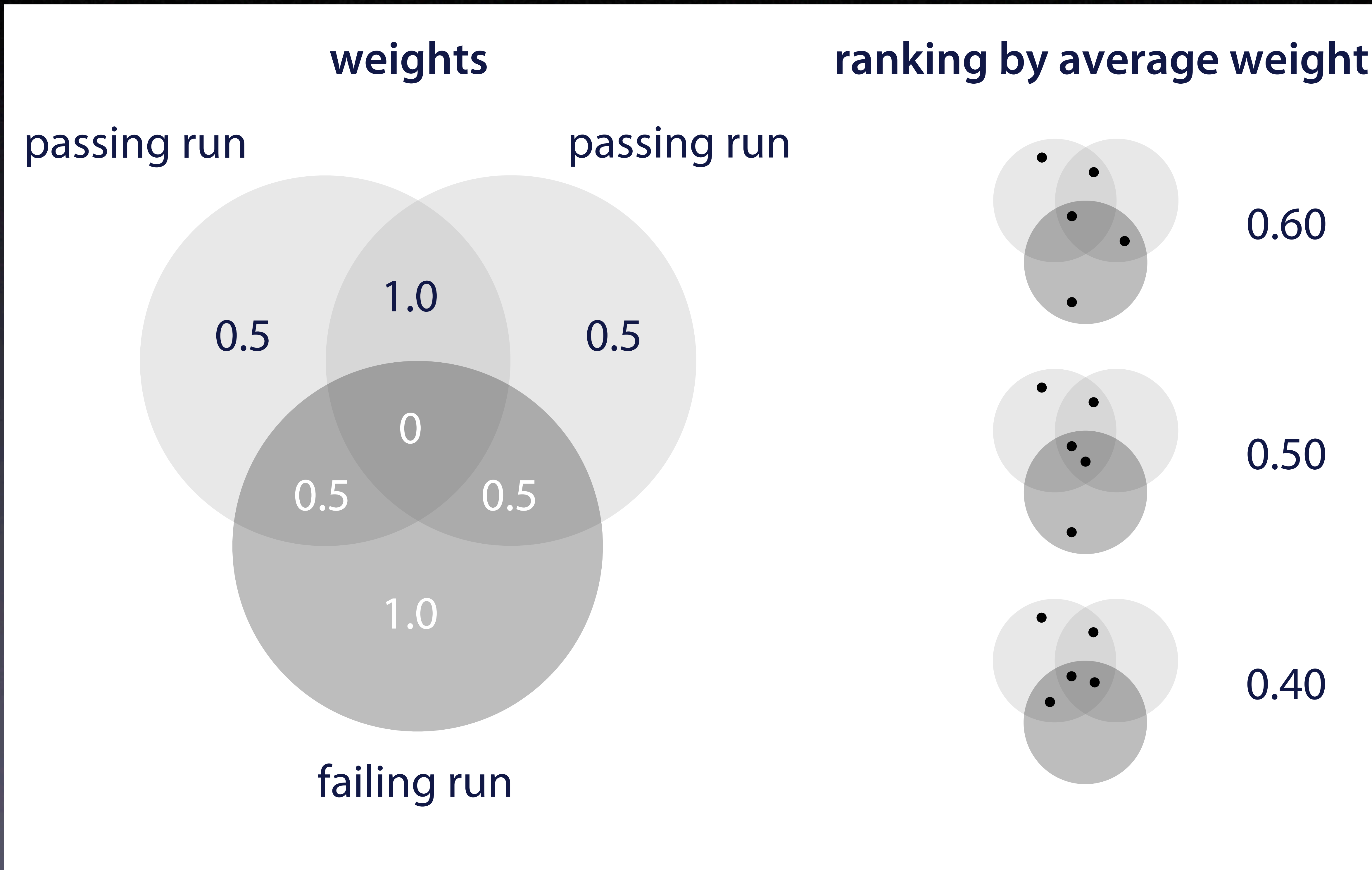
Collecting Sequences



Ingoing vs. Outgoing



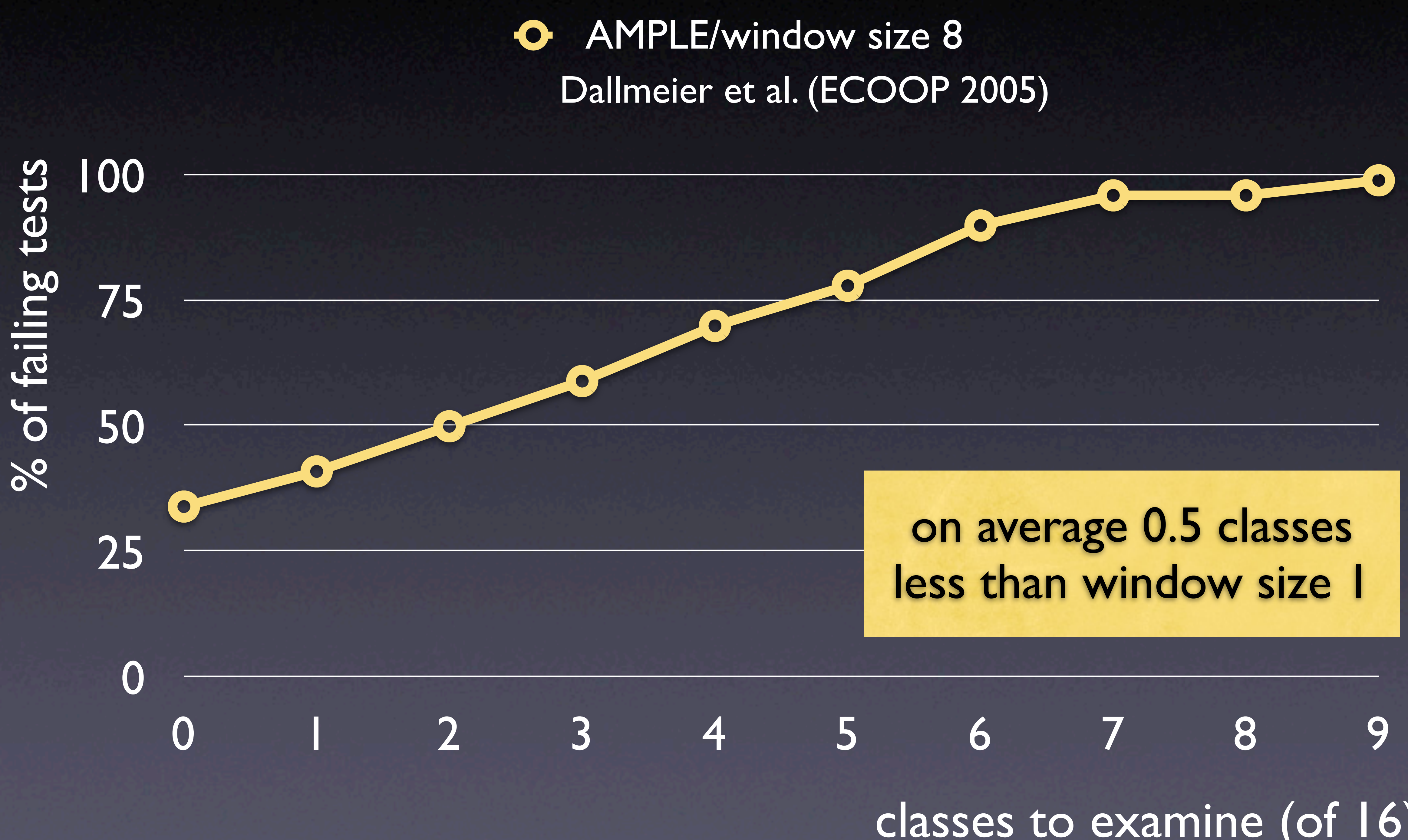
Anomalies


weights
ranking by average weight
passing run
passing run
0.5
1.0
0.5
0
0.5
0.5
1.0
failing run
0.60
0.50
0.40

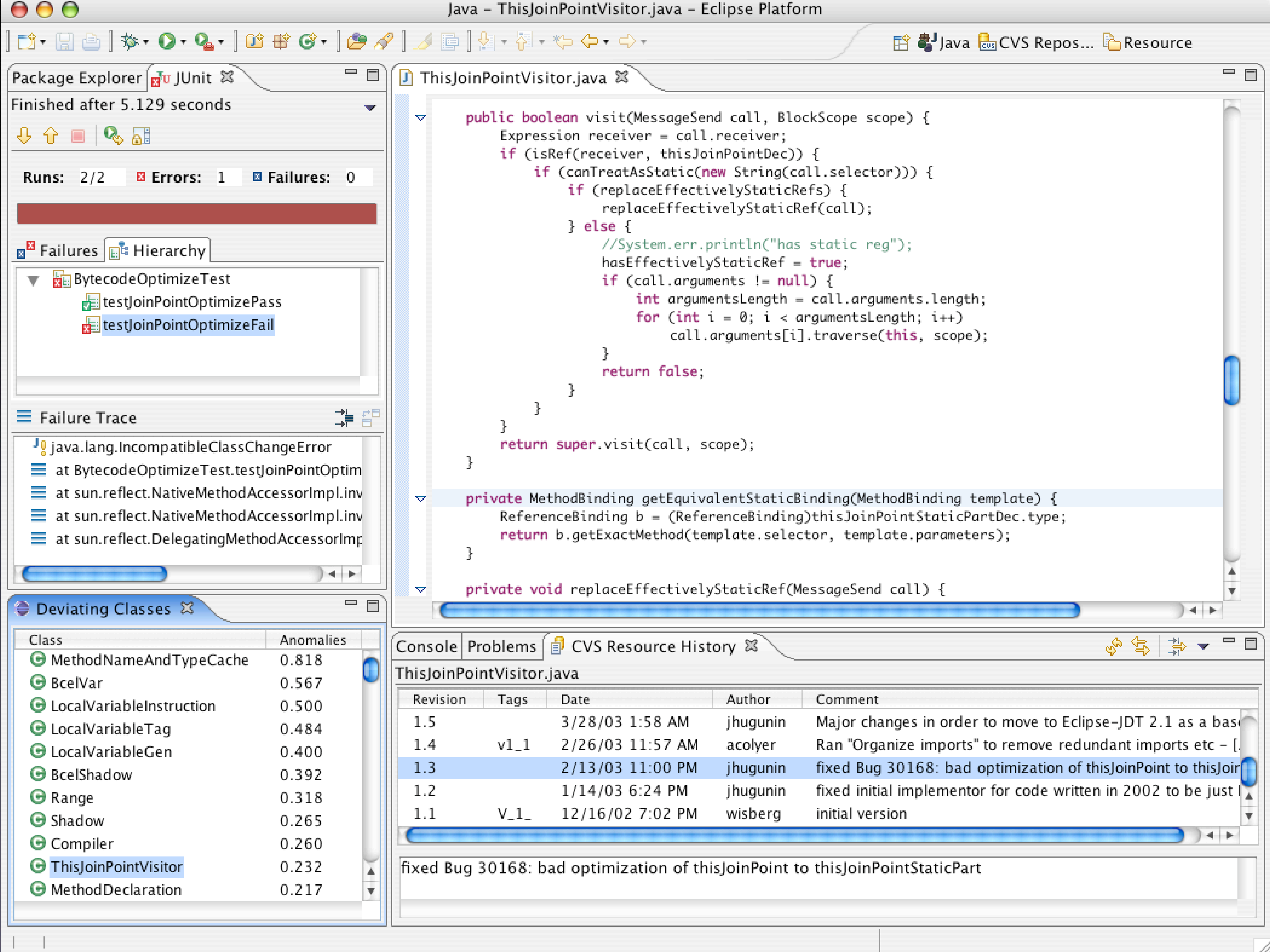
NanoXML

- Simple XML parser written in Java
- 5 revisions, each with 16–23 classes
- 33 errors discovered or seeded

Locating Defects



Results obtained from NanoXML; can not be generalized


Java – ThisJoinPointVisitor.java – Eclipse Platform
Java
CVS Repos...
Resource
Package Explorer
JUnit
Finished after 5.129 seconds
Runs: 2/2
Errors: 1
Failures: 0
Failures
Hierarchy
BytecodeOptimizeTest
testJoinPointOptimizePass
testJoinPointOptimizeFail
Failure Trace
java.lang.IncompatibleClassChangeError
at BytecodeOptimizeTest.testJoinPointOptim
at sun.reflect.NativeMethodAccessorImpl.inv
at sun.reflect.NativeMethodAccessorImpl.inv
at sun.reflect.DelegatingMethodAccessorImp
ThisJoinPointVisitor.java
public boolean visit(MessageSend call, BlockScope scope) {
Expression receiver = call.receiver;
if (isRef(receiver, thisJoinPointDec)) {
if (canTreatAsStatic(new String(call.selector))) {
if (replaceEffectivelyStaticRefs) {
replaceEffectivelyStaticRef(call);
} else {
//System.err.println("has static reg");
hasEffectivelyStaticRef = true;
if (call.arguments != null) {
int argumentsLength = call.arguments.length;
for (int i = 0; i < argumentsLength; i++)
call.arguments[i].traverse(this, scope);
}
return false;
}
}
}
return super.visit(call, scope);
}
private MethodBinding getEquivalentStaticBinding(MethodBinding template) {
ReferenceBinding b = (ReferenceBinding)thisJoinPointStaticPartDec.type;
return b.getExactMethod(template.selector, template.parameters);
}
private void replaceEffectivelyStaticRef(MessageSend call) {
Deviating Classes
Class
Anomalies
MethodNameAndTypeCache 0.818
BcelVar 0.567
LocalVariableInstruction 0.500
LocalVariableTag 0.484
LocalVariableGen 0.400
BcelShadow 0.392
Range 0.318
Shadow 0.265
Compiler 0.260
ThisJoinPointVisitor 0.232
MethodDeclaration 0.217
Console
Problems
CVS Resource History
ThisJoinPointVisitor.java
Revision
Tags
Date
Author
Comment
1.5 3/28/03 1:58 AM jhugunin Major changes in order to move to Eclipse-JDT 2.1 as a bas
1.4 v1_1 2/26/03 11:57 AM acolyer Ran "Organize imports" to remove redundant imports etc – [
1.3 2/13/03 11:00 PM jhugunin fixed Bug 30168: bad optimization of thisJoinPoint to thisJoir
1.2 1/14/03 6:24 PM jhugunin fixed initial implementor for code written in 2002 to be just
1.1 V_1_ 12/16/02 7:02 PM wisberg initial version
fixed Bug 30168: bad optimization of thisJoinPoint to thisJoinPointStaticPart

Properties

Coverage	Sequences	Returns

Detecting Anomalies

Differences correlate with failure

Liblit’s Sampling

- We want properties of runs in the field
- Collecting all this data is too expensive
- Would a *sample* suffice?
- Sampling experiment by Liblit et al. (2003)

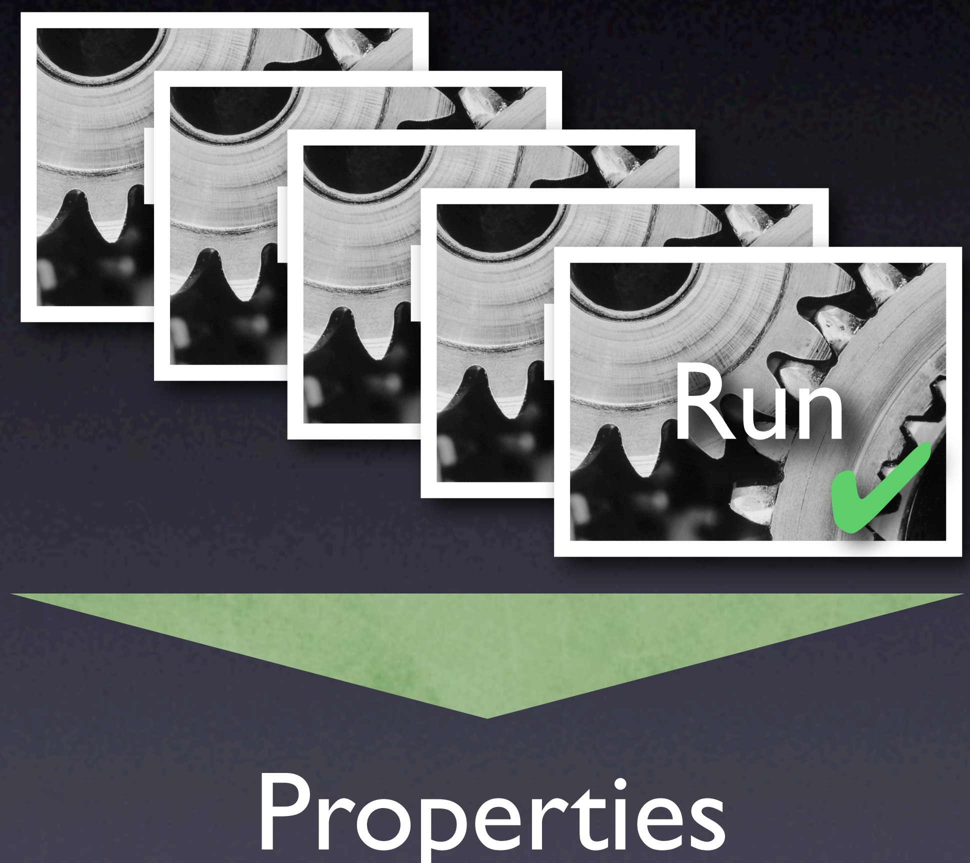
Return Values

- Hypothesis: *function return values* correlate with failure or success
- Classified into positive / zero / negative

CCRYPT fails

- CCRYPT is an interactive encryption tool
- When CCRYPT asks user for information before overwriting a file, and user responds with EOF, CCRYPT crashes
- 3,000 random runs
- Of 1,170 predicates, only file_exists() > 0 and xreadline() == 0 correlate with failure

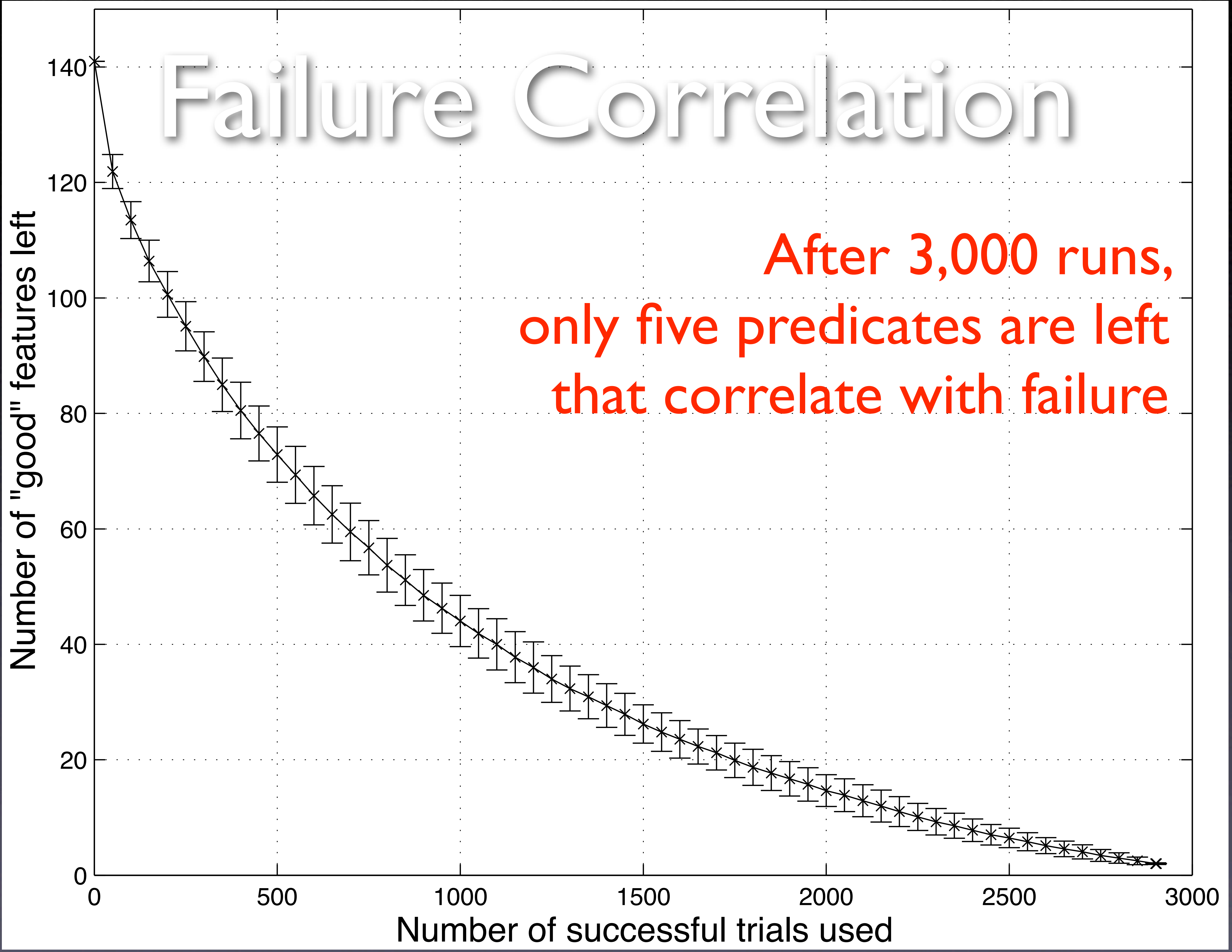
Liblit's Sampling



- Can we apply this technique to remote runs, too?
- 1 out of 1000 return values was sampled
- Performance loss <4%

Failure Correlation

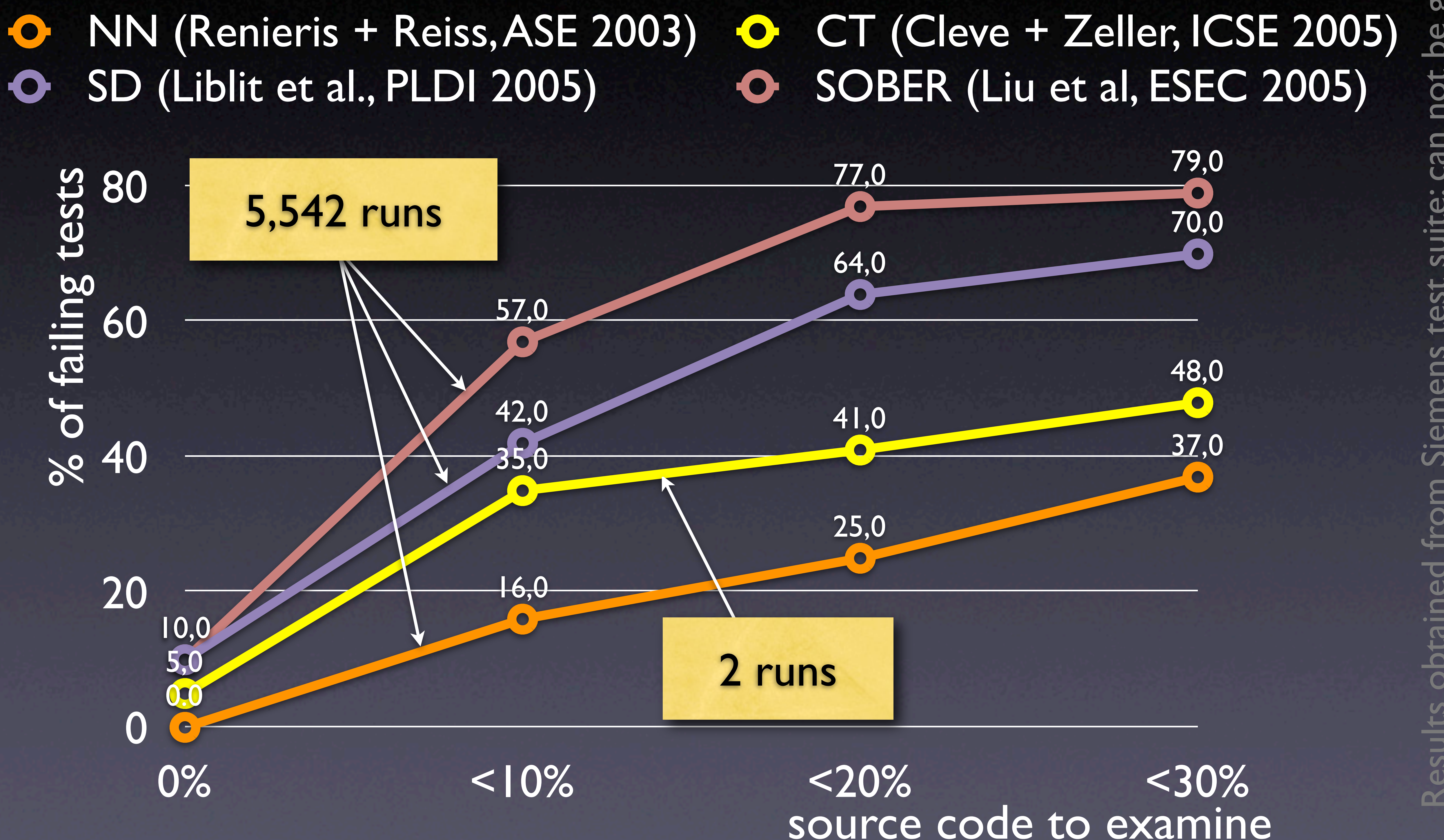
After 3,000 runs,
only five predicates are left
that correlate with failure



Web Services

- Sampling is first choice for web services
- Have 1 out of 100 users run an instrumented version of the web service
- Correlate instrumentation data with failure
- After sufficient number of runs, we can automatically identify the anomaly

Locating Defects

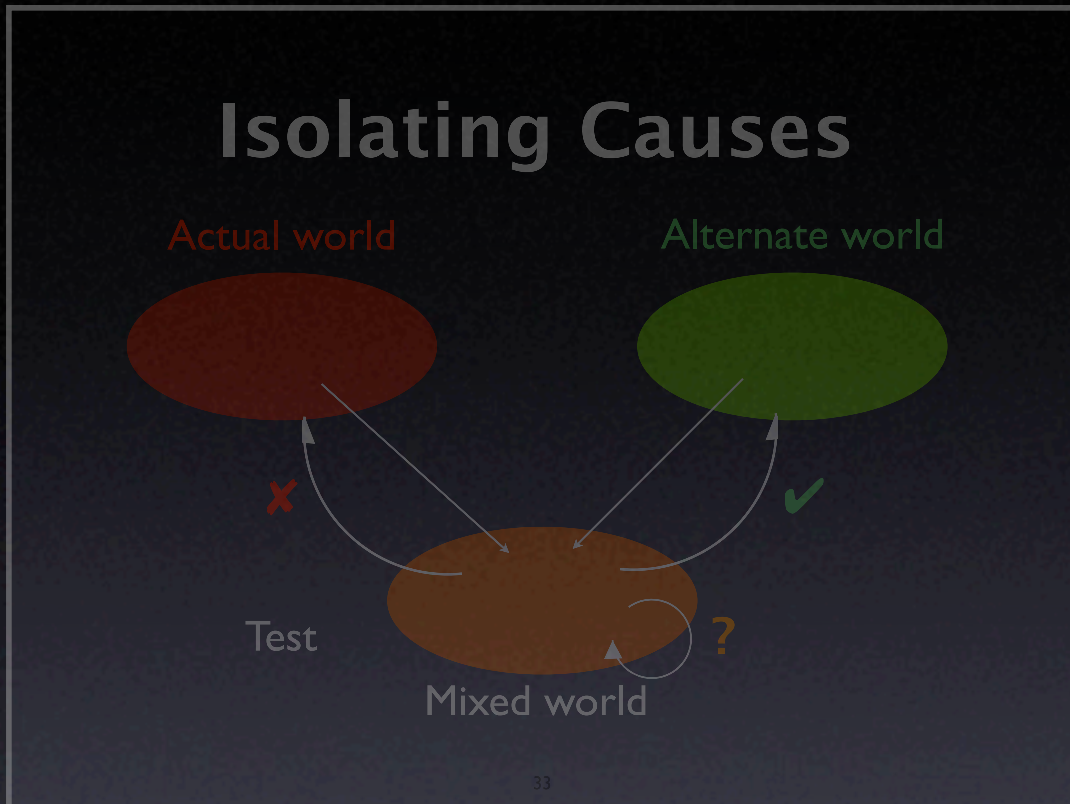


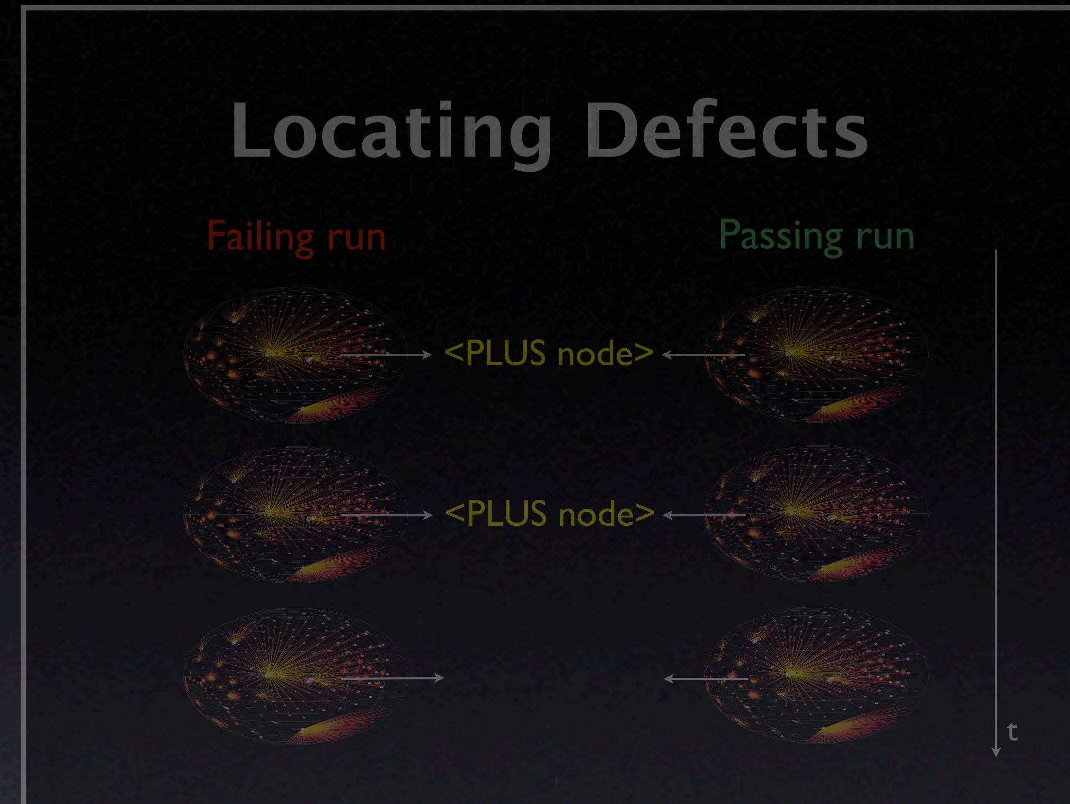
Anomalies and Causes

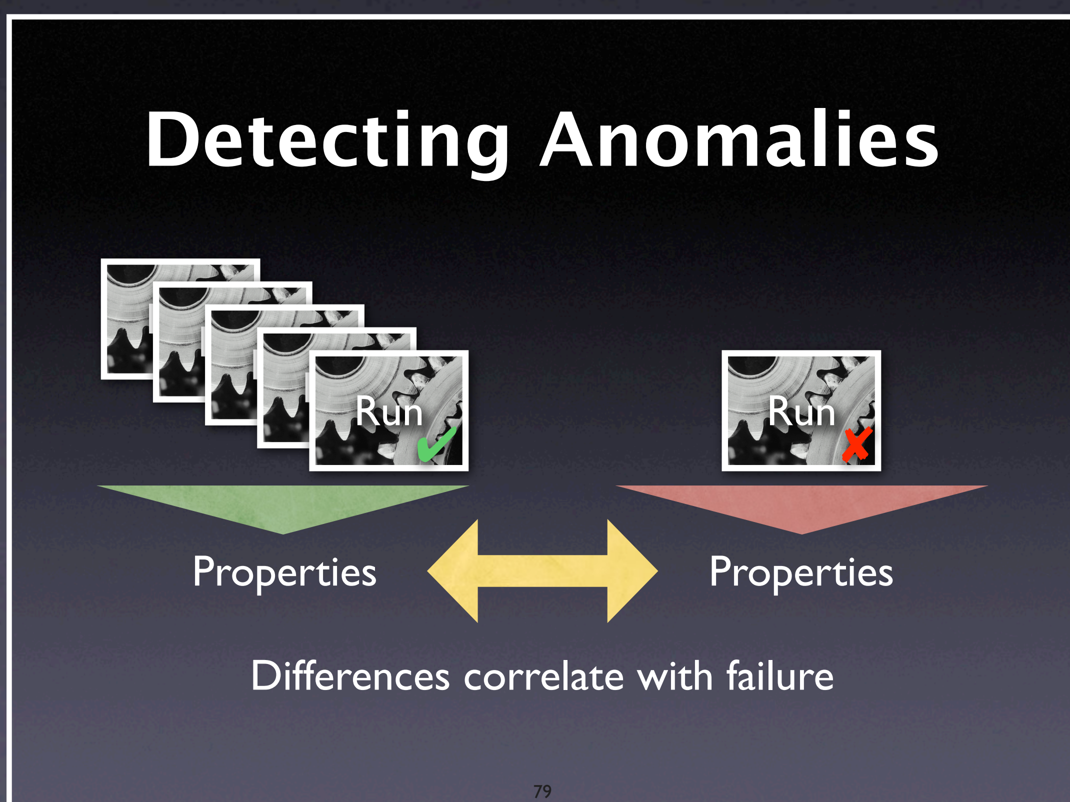
- An anomaly is not a cause, but a correlation
- Although correlation ≠ causation, anomalies can be excellent hints
- Future belongs to those who exploit
  - Correlations in *multiple runs*
  - Causation in *experiments*

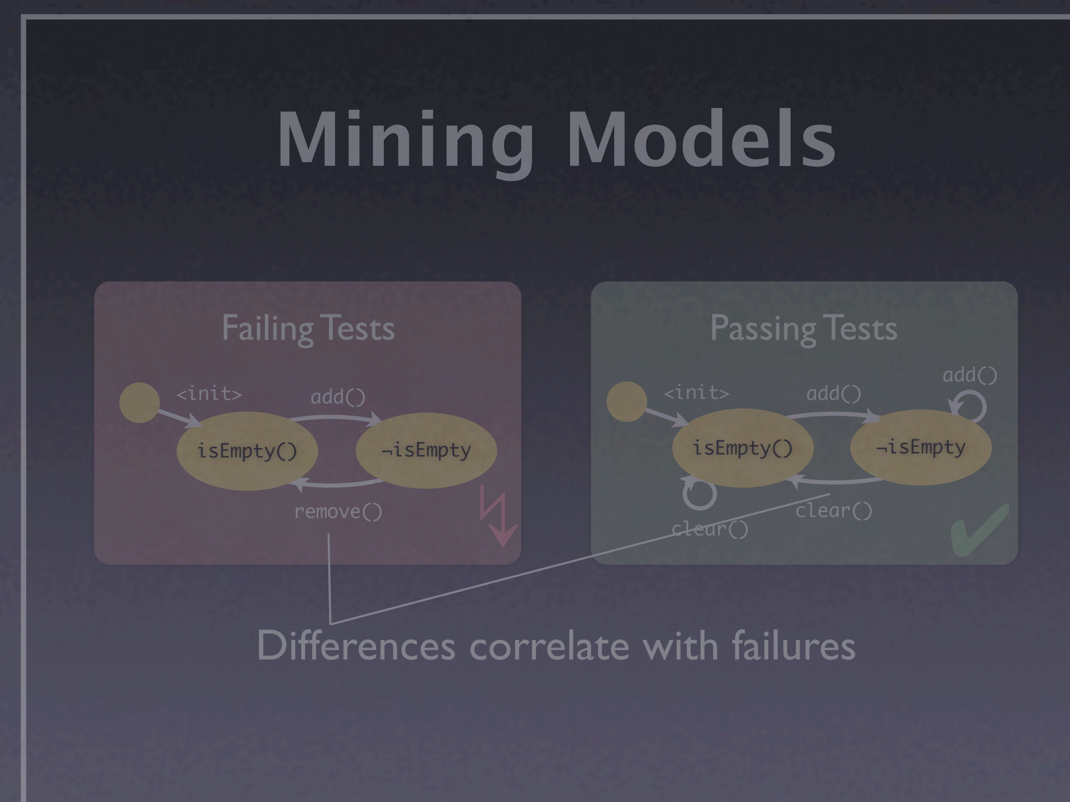
Concepts

- Comparing coverage (or other features) shows anomalies correlated with failure
- Nearest neighbor or sequences locate errors more precisely than just coverage
- Low overhead + simple to realize

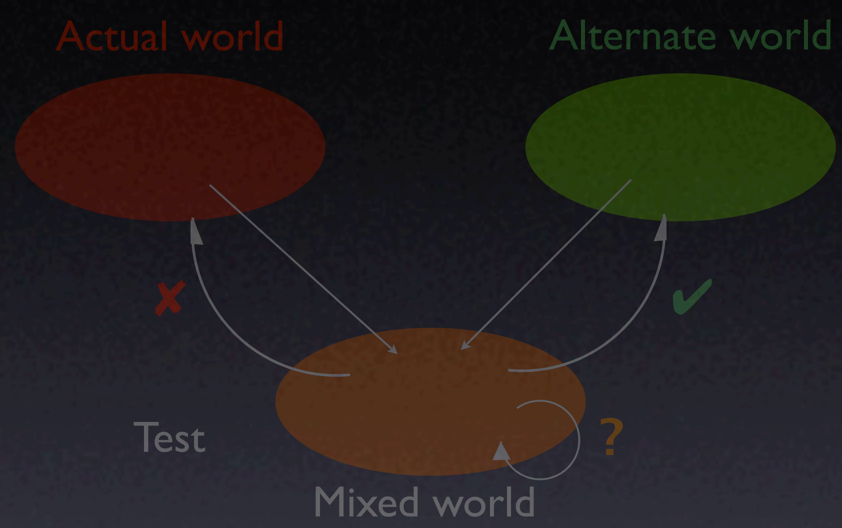

Isolating Causes
Actual world
Alternate world
Test
Mixed world
?
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t


Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79

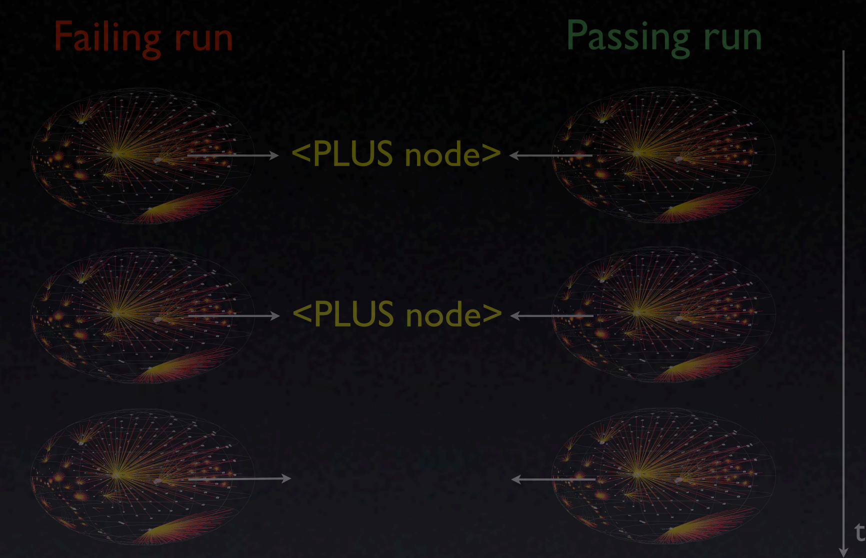

Mining Models
Failing Tests
Passing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures

Isolating Causes

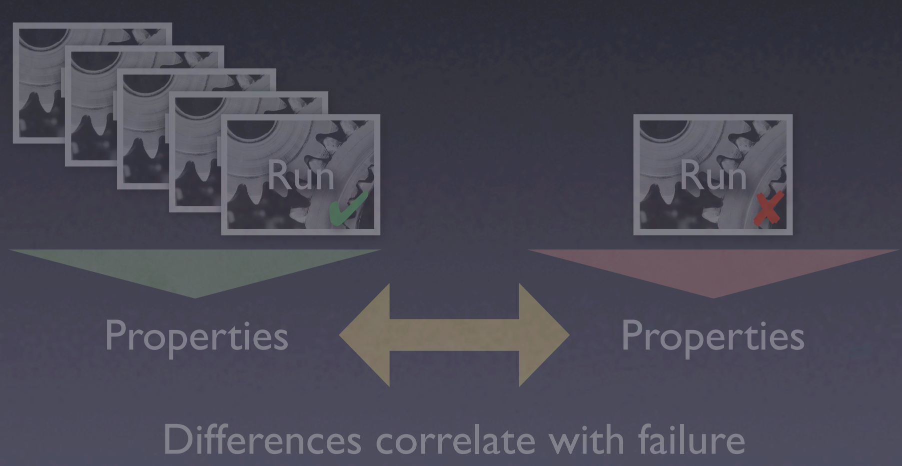


33

Locating Defects

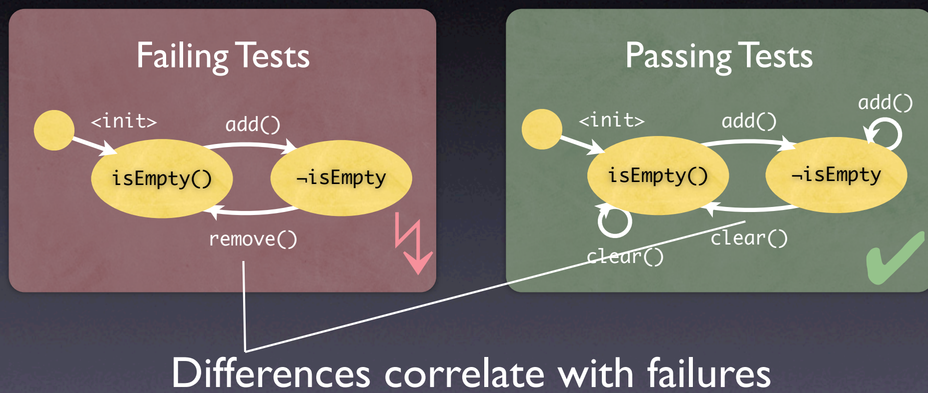


Detecting Anomalies

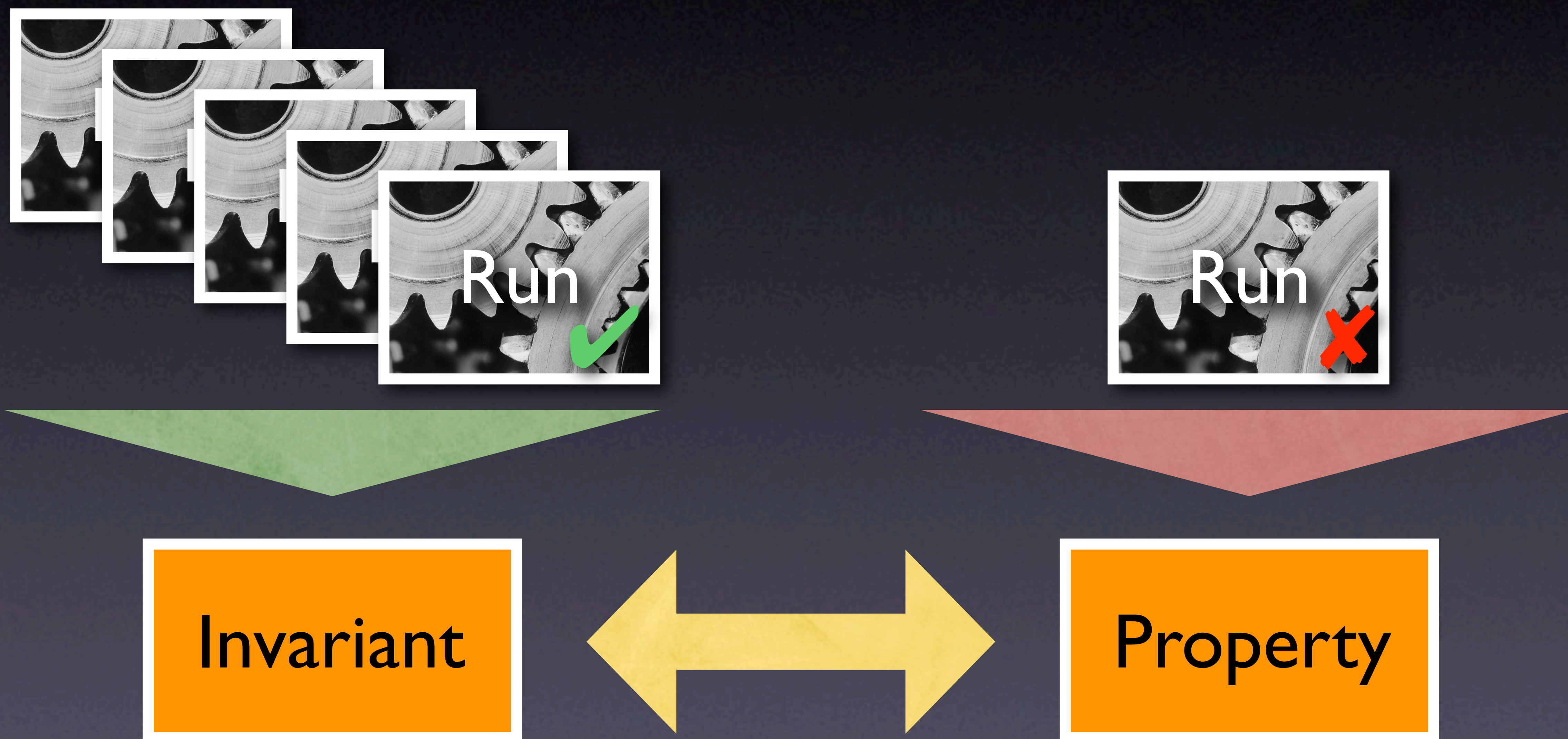


79

Mining Models



Dynamic Invariants



At f(), x is odd

At f(), x = 2

Daikon

- Determines *invariants* from program runs
- Written by Michael Ernst et al. (1998–)
- C++, Java, Lisp, and other languages
- analyzed up to 13,000 lines of code

Daikon

```
public int ex1511(int[] b, int n)
{
    int s = 0;
    int i = 0;
    while (i != n) {
        s = s + b[i];
        i = i + 1;
    }
    return s;
}
```

Precondition

```
n == size(b[])
b != null
n <= 13
n >= 7
```

Postcondition

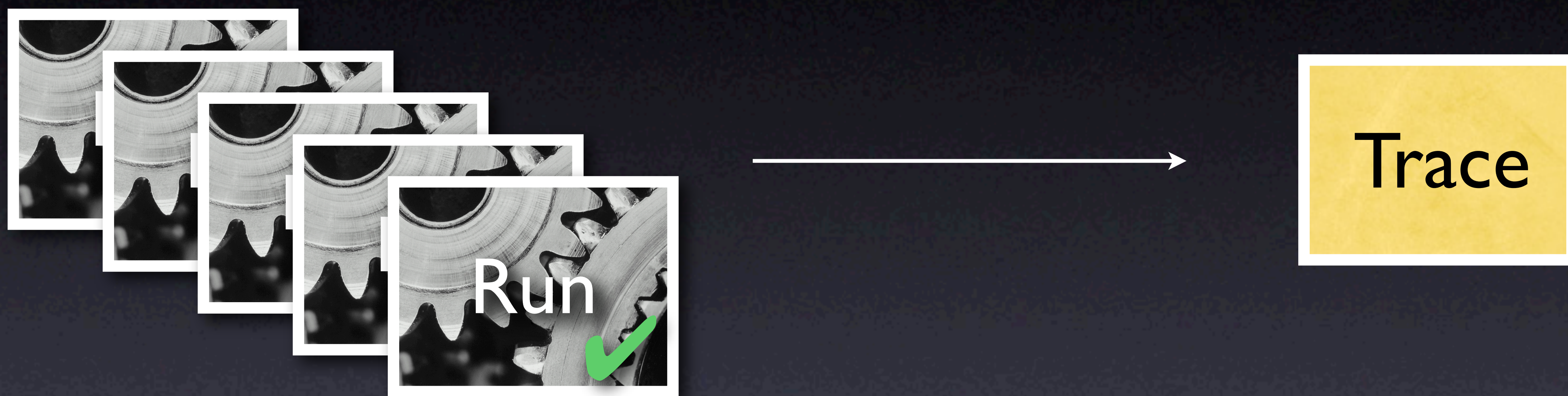
```
b[] = orig(b[])
return == sum(b)
```

- Run with 100 randomly generated arrays of length 7–13

Daikon

Run
get trace
Trace
filter invariants
Invariant
report results
Postcondition
b[] = orig(b[])
return == sum(b)

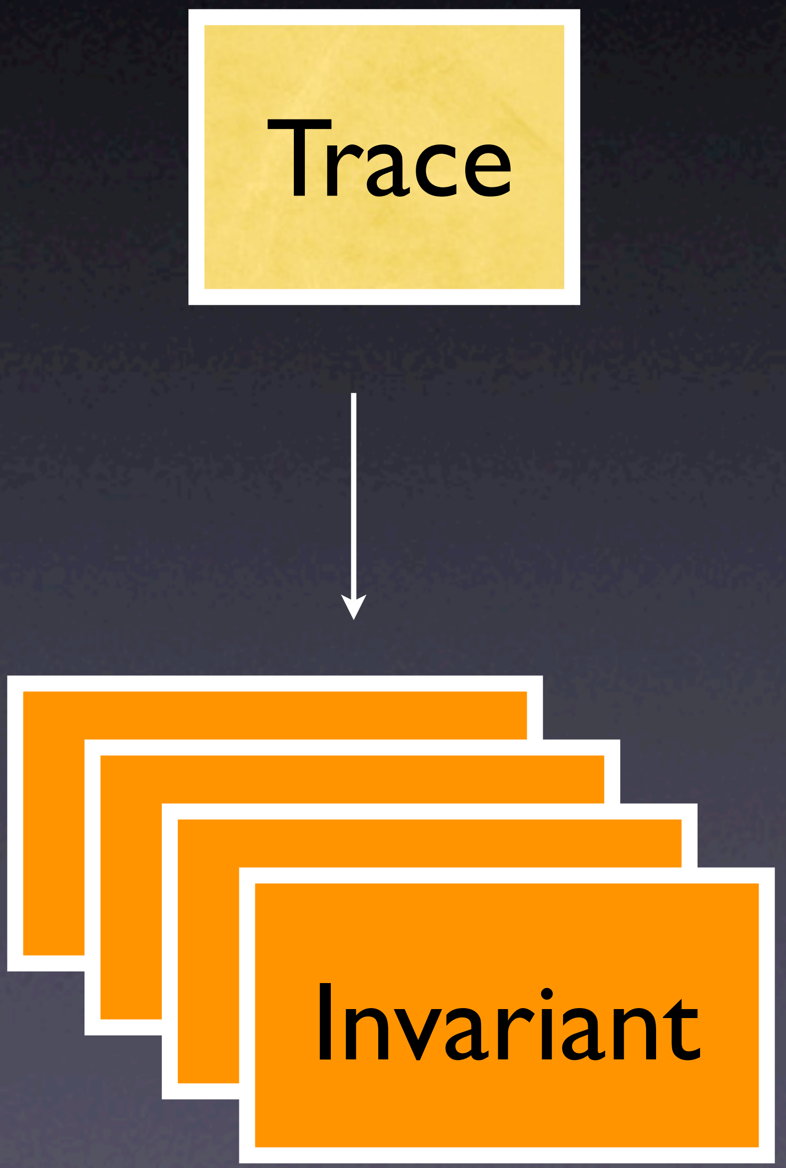
Getting the Trace



- Records all variable values at all function entries and exits
- Uses VALGRIND to create the trace

Filtering Invariants

- Daikon has a library of *invariant patterns* over variables and constants
- Only matching patterns are preserved



Method Specifications

using *primitive data*

$x = 6$	$x \in \{2, 5, -30\}$	$x < y$
$y = 5x + 10$	$z = 4x + 12y + 3$	$z = fn(x, y)$

using *composite data*

A subseq B	$x \in A$	sorted(A)

checked at method entry + exit

Object Invariants

string.content[string.length] = '\0'
node.left.value ≤ node.right.value
this.next.last = this

checked at entry + exit of public methods

Matching Invariants

```
public int ex1511(int[] b, int n)
{
    int s = 0;
    int i = 0;
    while (i != n) {
        s = s + b[i];
        i = i + 1;
    }
    return s;
}
```

A == B

Pattern

s size(b[])
sum(b[])
n
orig(n)
return ...

Variables

Matching Invariants

==	s	n	size(b[])	sum(b[])	orig(n)	ret
s		✗			✗	
n	✗			✗		✗
size(b[])						
sum(b[])		✗				
orig(n)	✗					✗
ret		✗			✗	

run 1

A == B

Pattern

s size(b[])
sum(b[])
n
orig(n)
return …

Variables

Matching Invariants

==	s	n	size(b[])	sum(b[])	orig(n)	ret
s		✗	✗		✗	
n	✗			✗	✗	✗
size(b[])	✗			✗		✗
sum(b[])		✗	✗		✗	
orig(n)	✗	✗		✗		✗
ret		✗	✗		✗	

run 2

A == B

Pattern

s size(b[])
sum(b[])
n
orig(n)
return …

Variables

Matching Invariants

==	s	n	size(b[])	sum(b[])	orig(n)	ret
s		✗	✗		✗	
n	✗			✗	✗	✗
size(b[])	✗			✗		✗
sum(b[])		✗	✗		✗	
orig(n)	✗	✗		✗		✗
ret		✗	✗		✗	

run 3

A == B

Pattern

s size(b[])
sum(b[])
n
orig(n)
return …

Variables

Matching Invariants

==	s	n	size(b[])	sum(b[])	orig(n)	ret
s		✗	✗		✗	
n	✗			✗	✗	✗
size(b[])	✗			✗		✗
sum(b[])		✗	✗		✗	
orig(n)	✗	✗		✗		✗
ret		✗	✗		✗	

s == sum(b[])

s == ret

n == size(b[])

ret == sum(b[])

Matching Invariants

```
public int ex1511(int[] b, int n)
{
    int s = 0;
    int i = 0;
    while (i != n) {
        s = s + b[i];
        i = i + 1;
    }
    return s;
}
```

s == sum(b[])

s == ret

n == size(b[])

ret == sum(b[])

Enhancing Relevance

- Handle polymorphic variables
- Check for derived values
- Eliminate redundant invariants
- Set statistical threshold for relevance
- Verify correctness with static analysis

Daikon Discussed

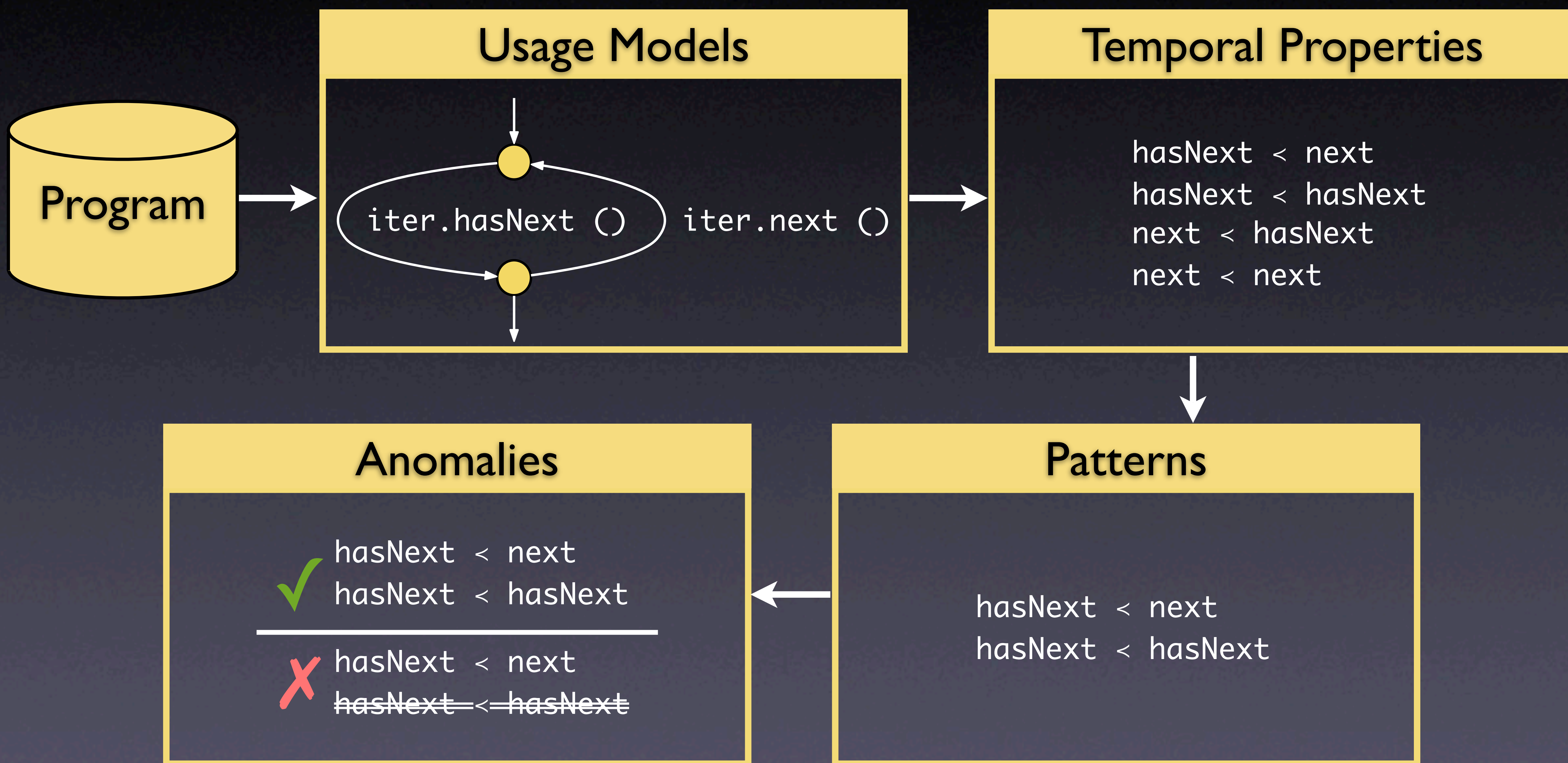
- As long as some property can be observed, it can be added as a pattern
- Pattern vocabulary determines the invariants that can be found (“sum()”, etc.)
- Checking all patterns (and combinations!) is expensive
- Trivial invariants must be eliminated

Why mine executions?

- We can also mine *other program code!*

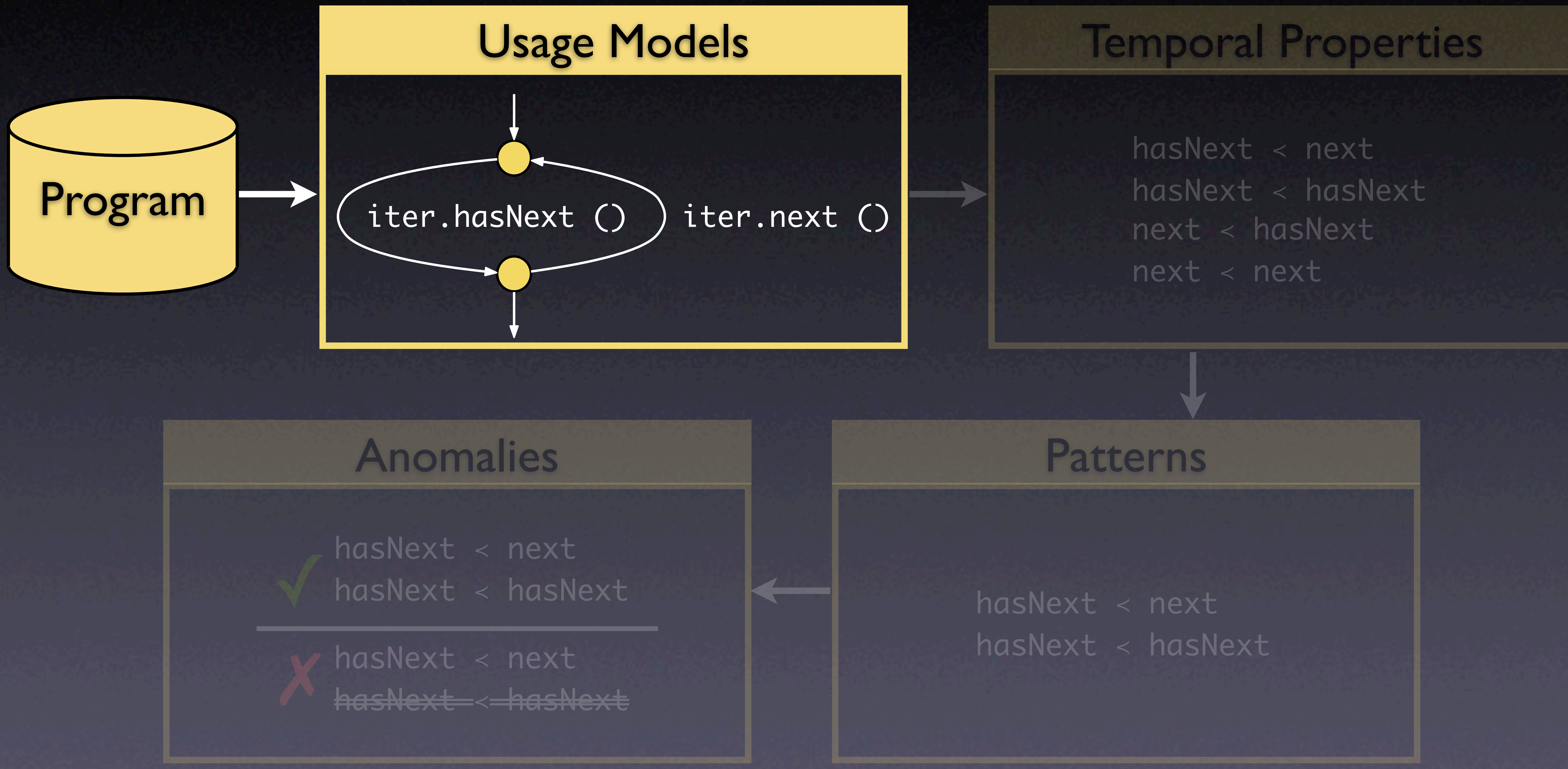
Andrzej Wasylkowski

OP-Miner



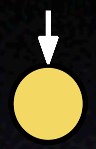
Program
Usage Models
iter.hasNext ()
iter.next ()
Temporal Properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
~~hasNext < hasNext~~

OP-Miner



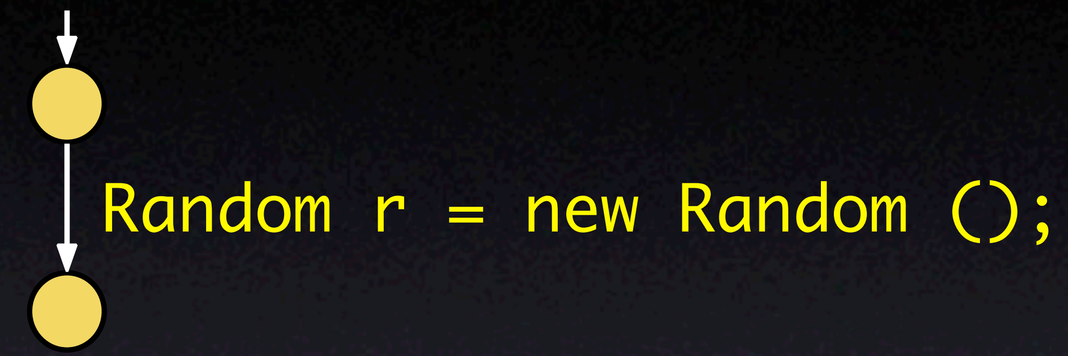
Program
Usage Models
iter.hasNext ()
iter.next ()
Temporal Properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
~~hasNext < hasNext~~

Method Models



```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

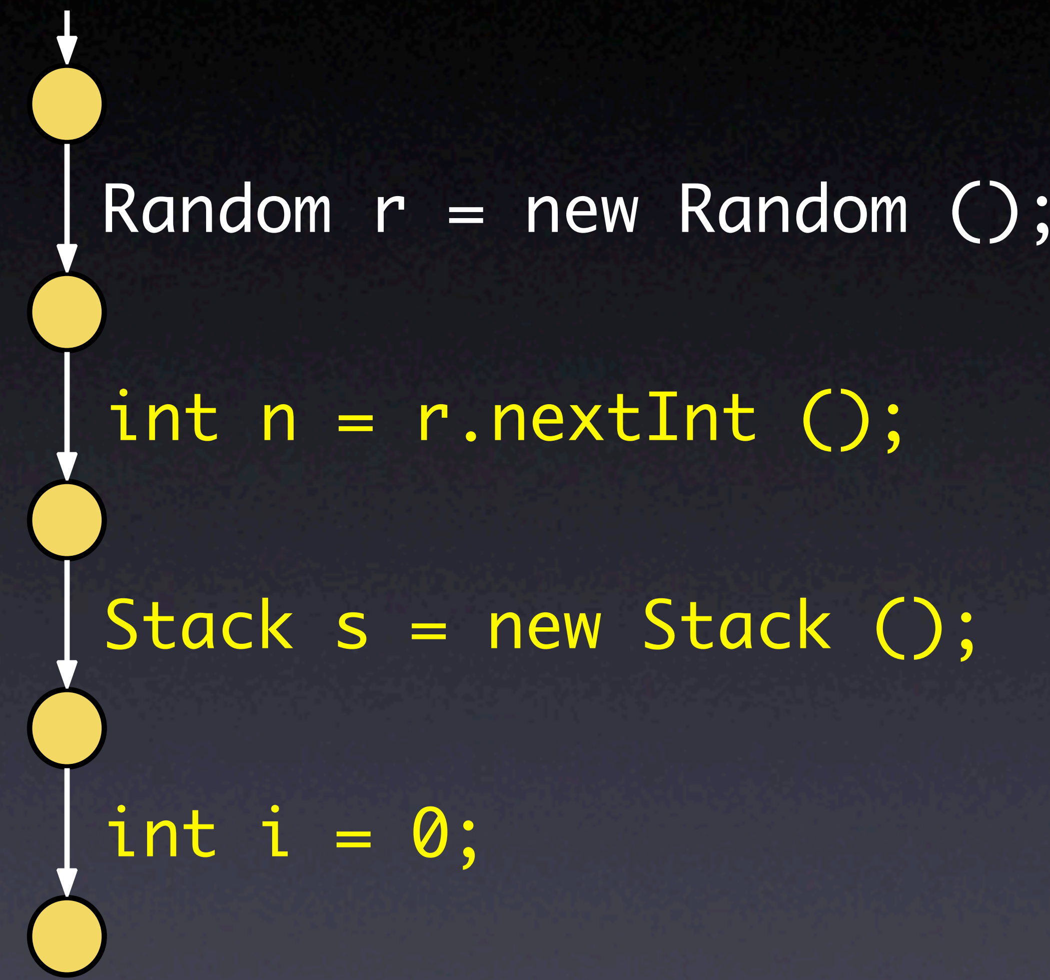
Method Models




```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

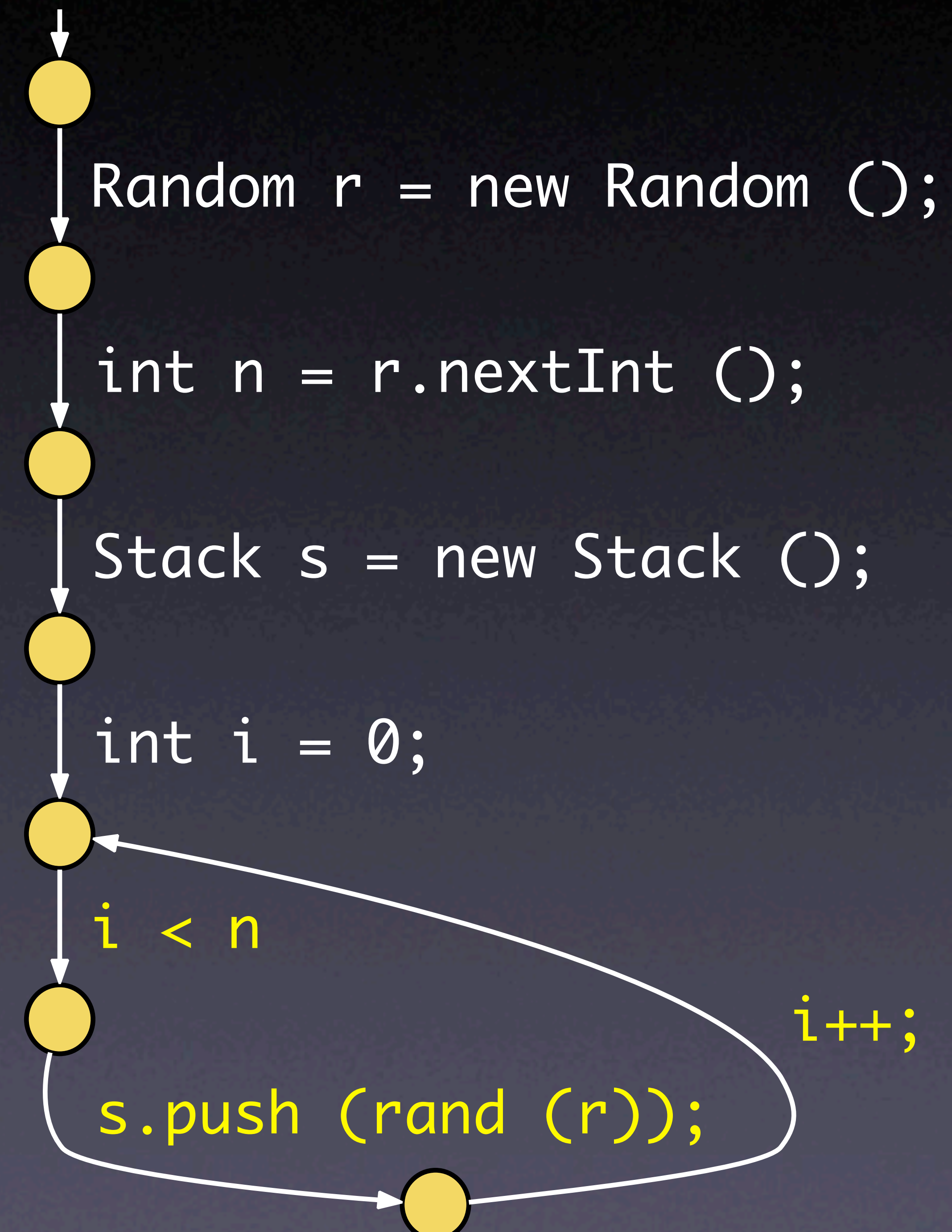
Method Models

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



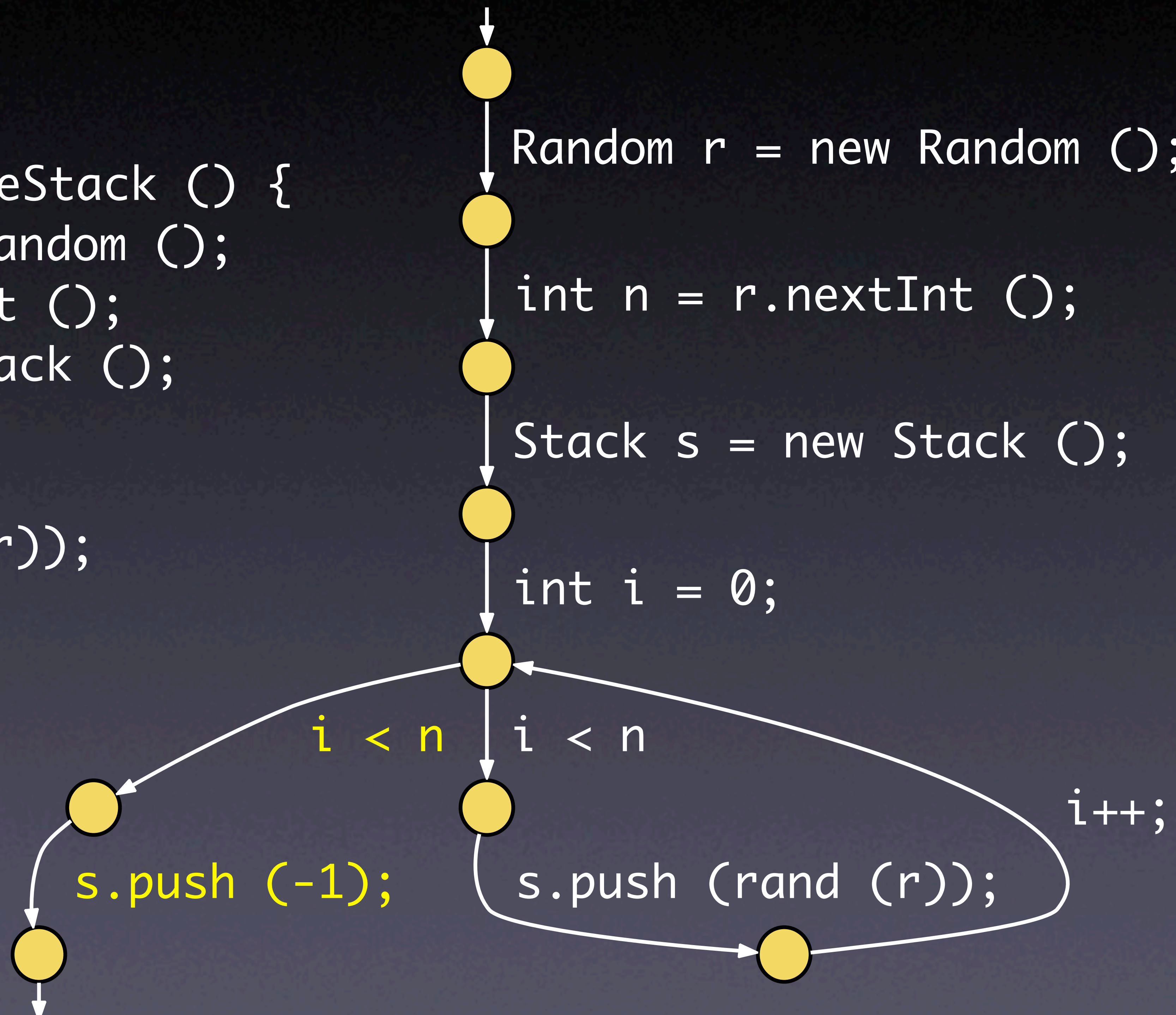
Method Models

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



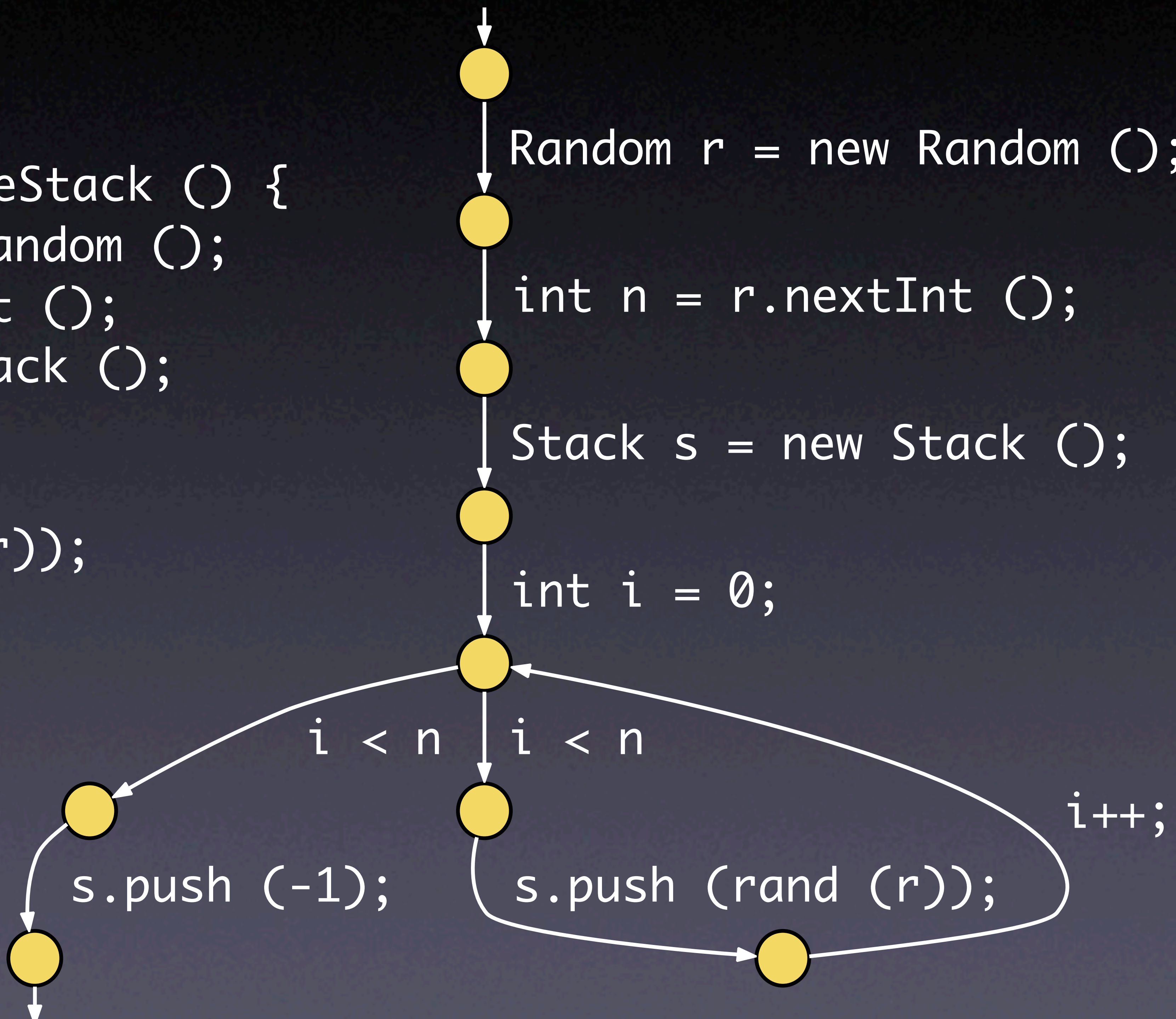
Method Models

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

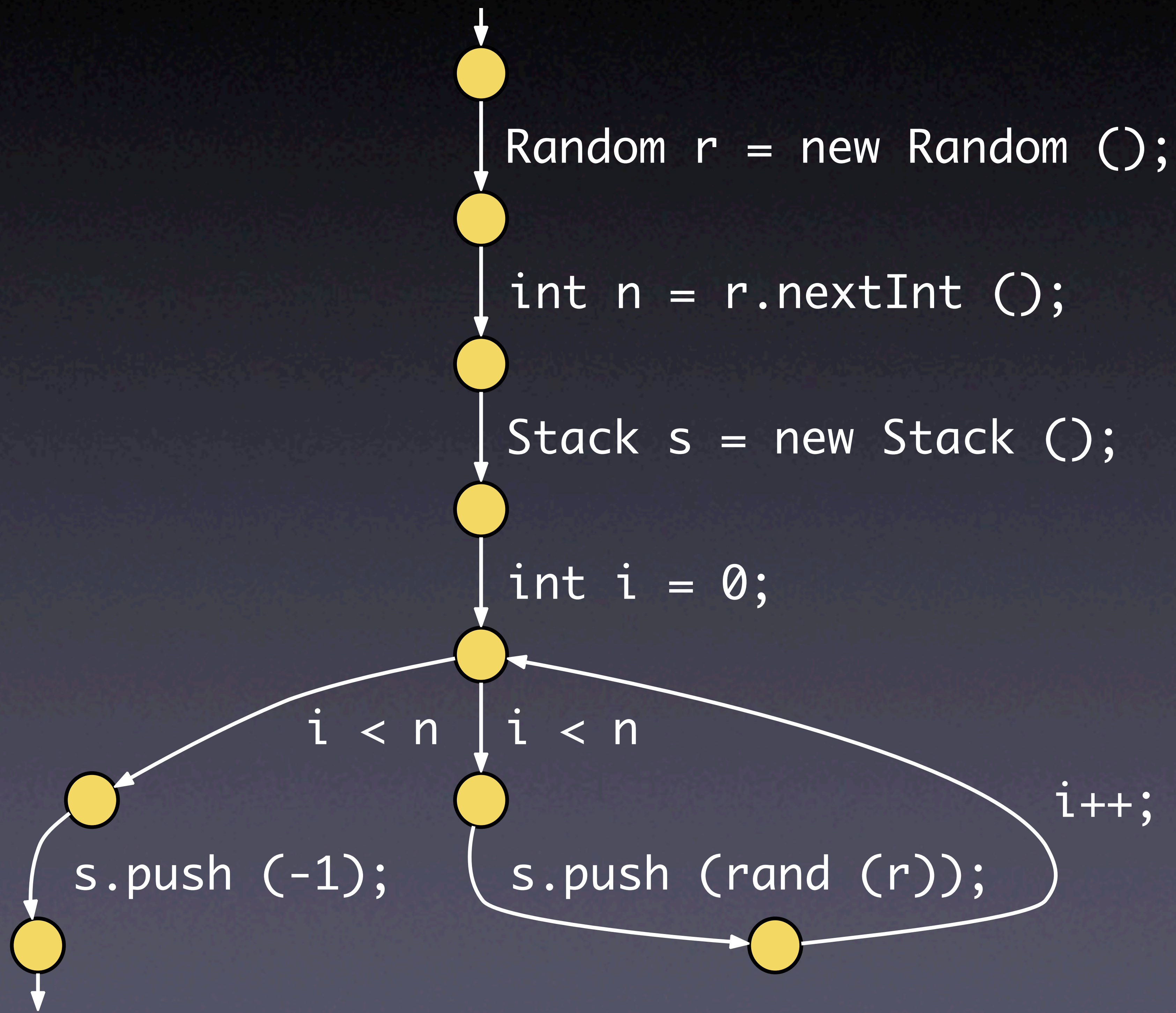


Method Models

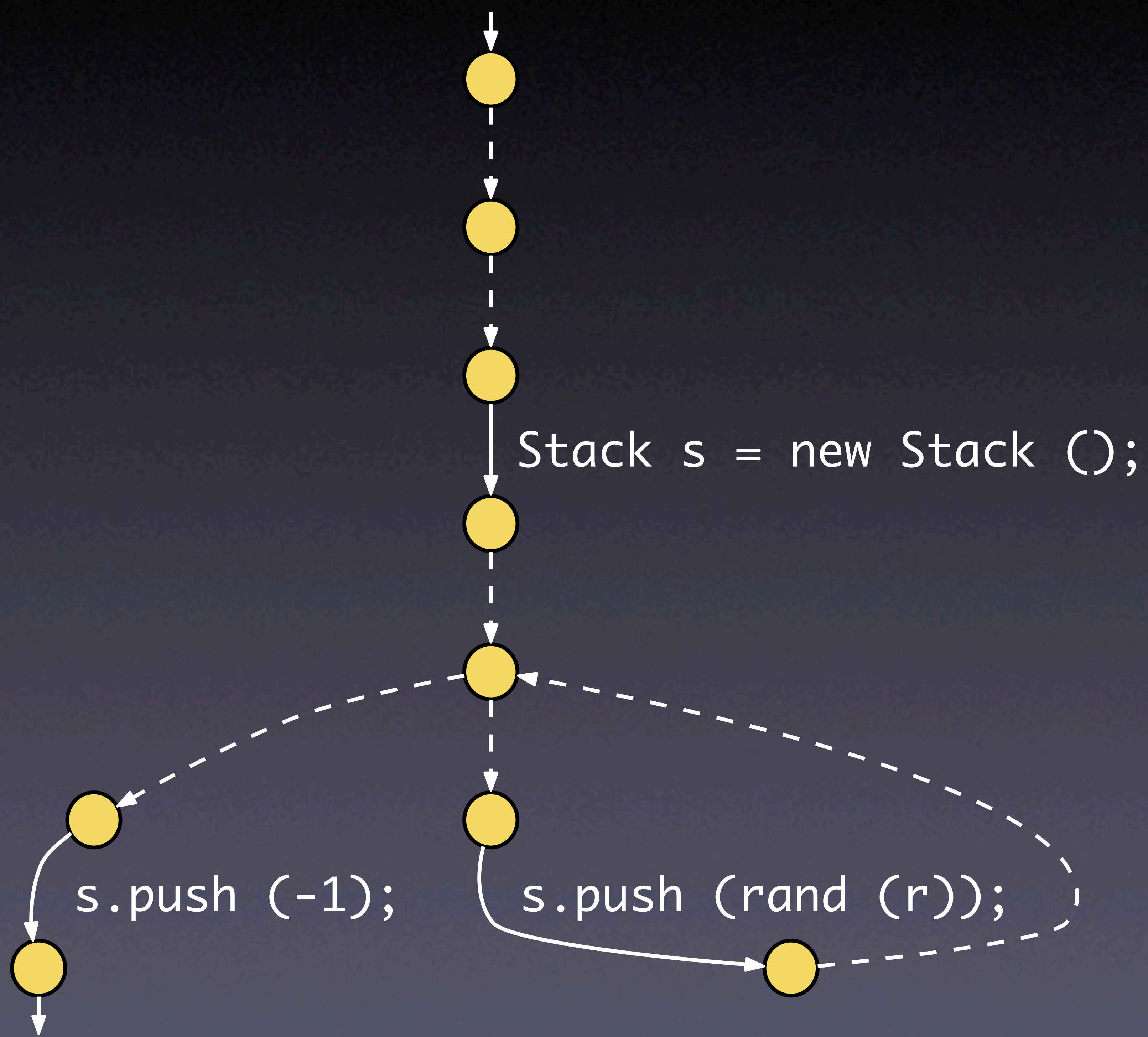
```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



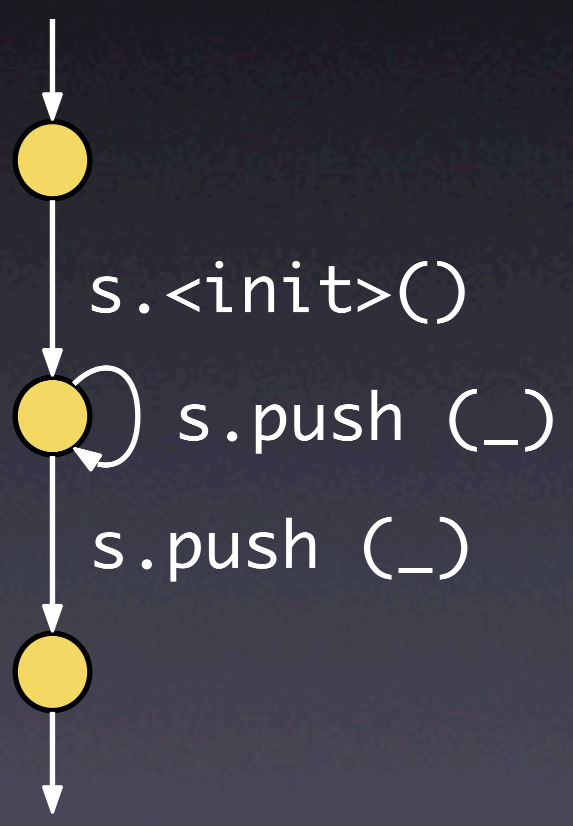
Usage Models



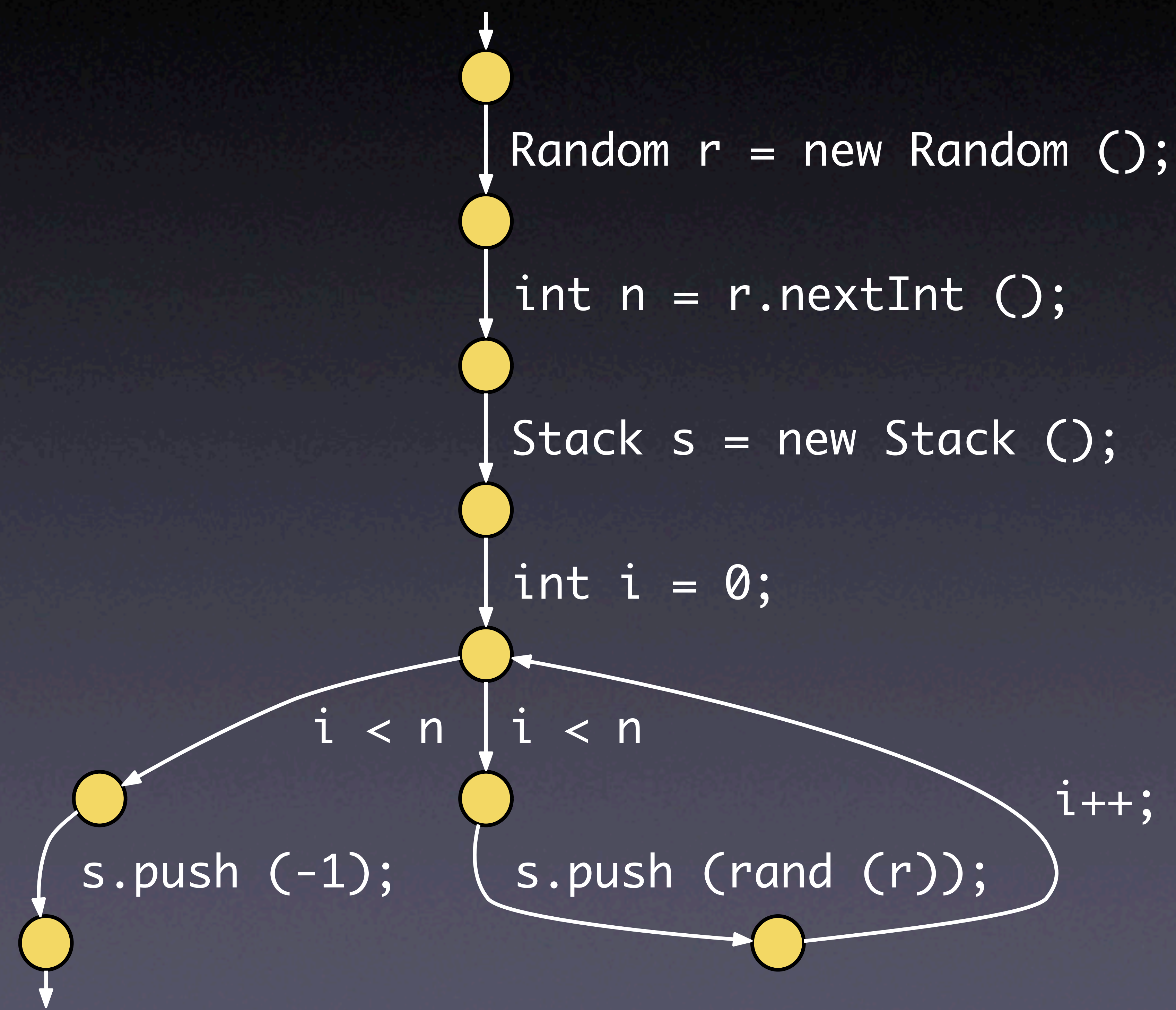
Usage Models



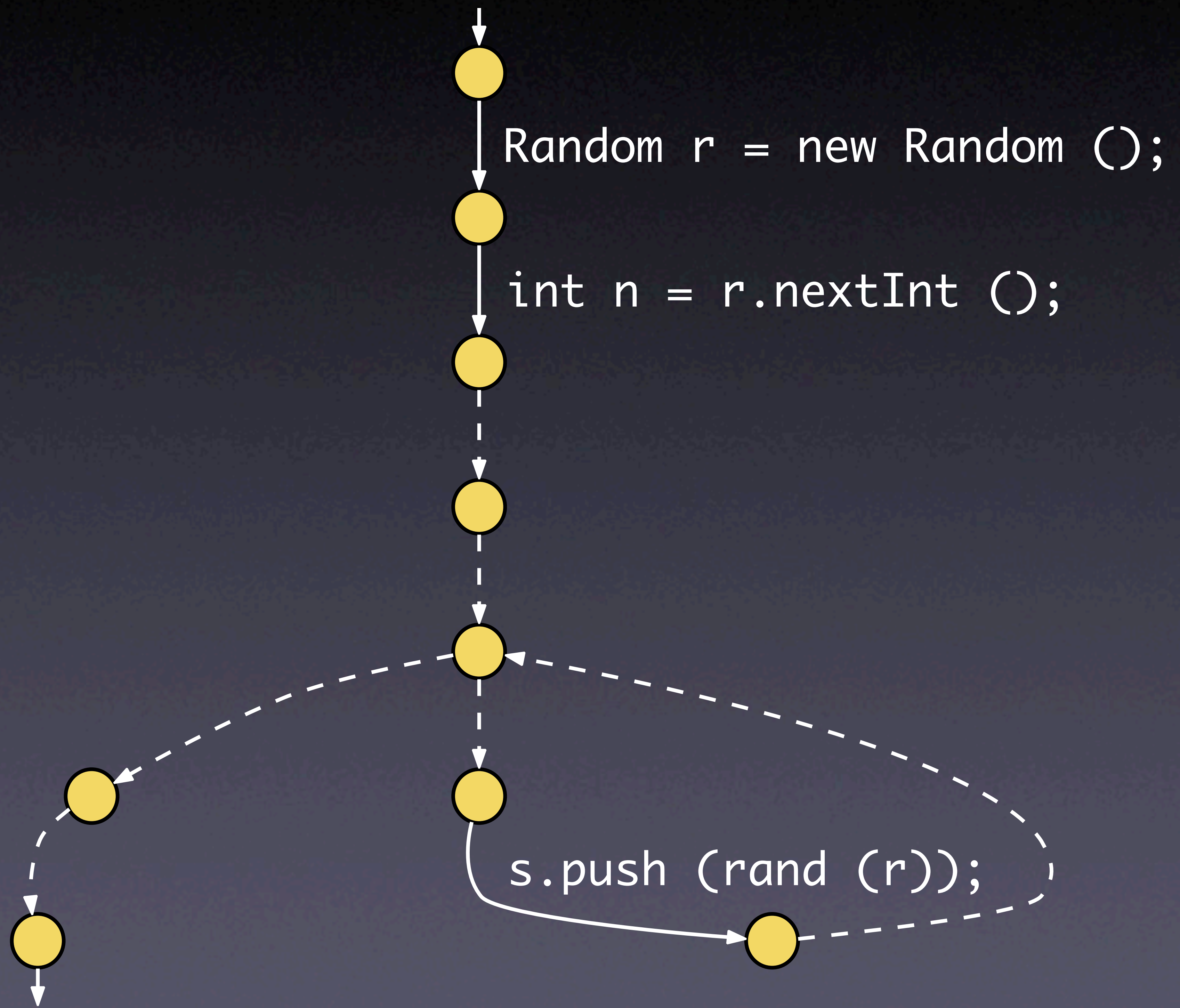
Usage Models



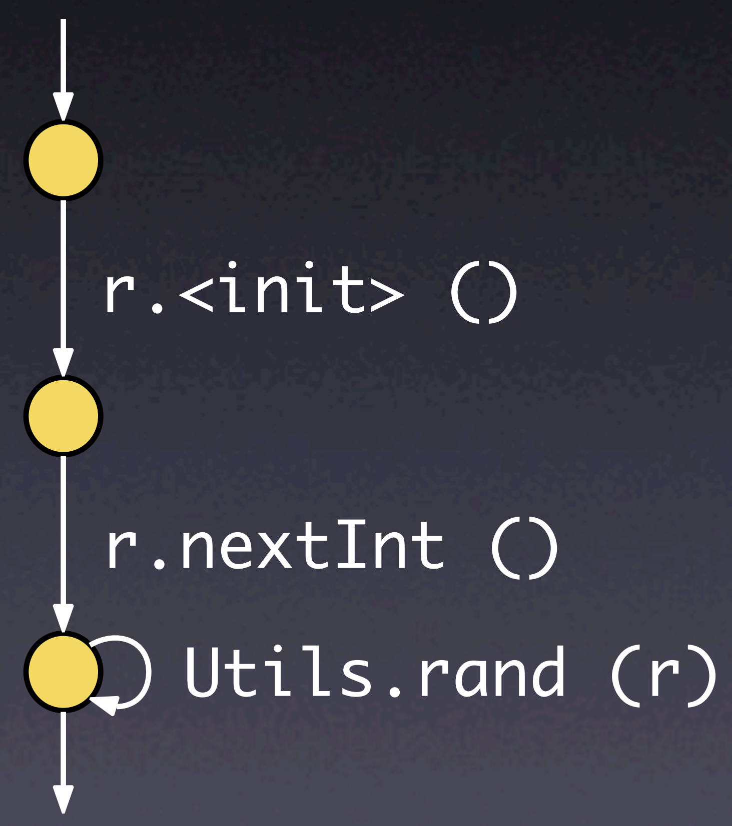
Usage Models



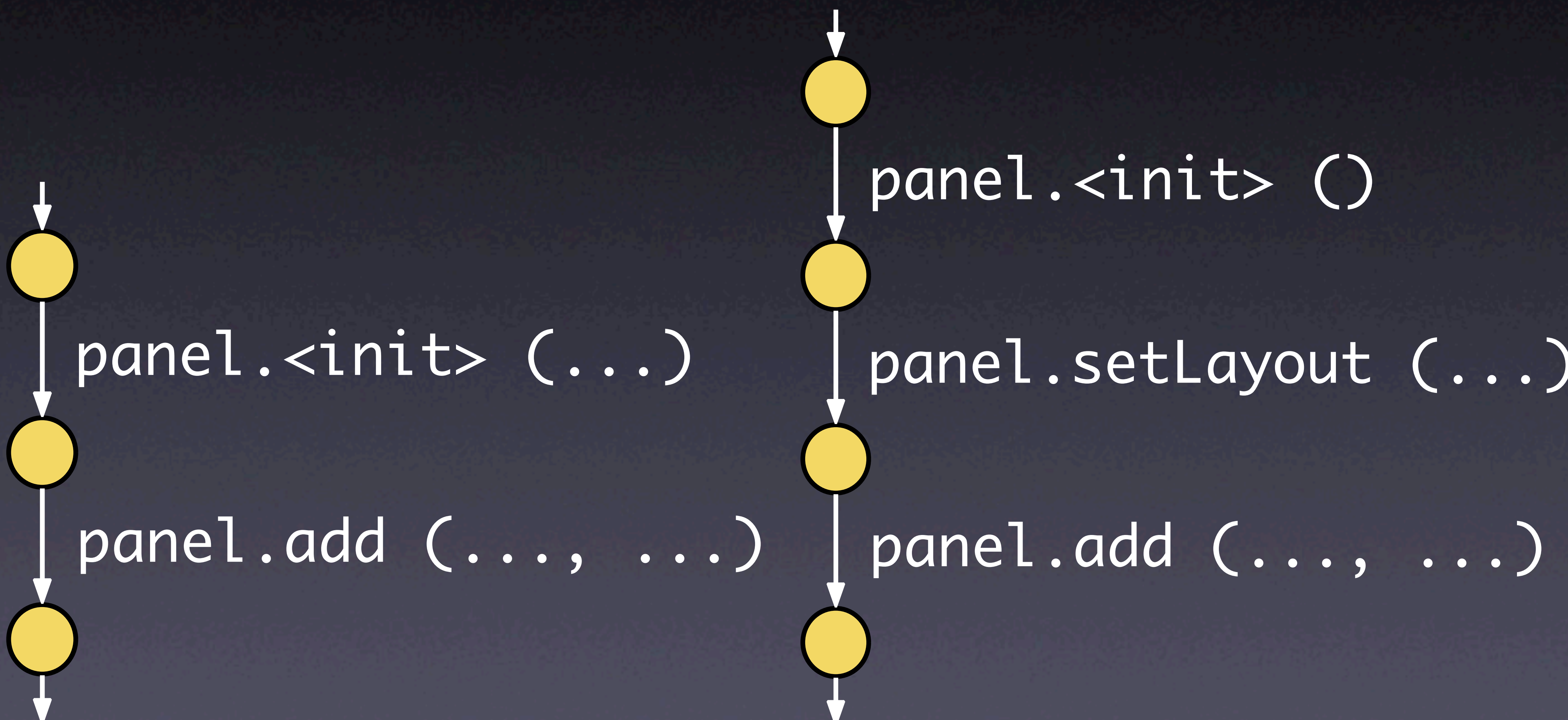
Usage Models



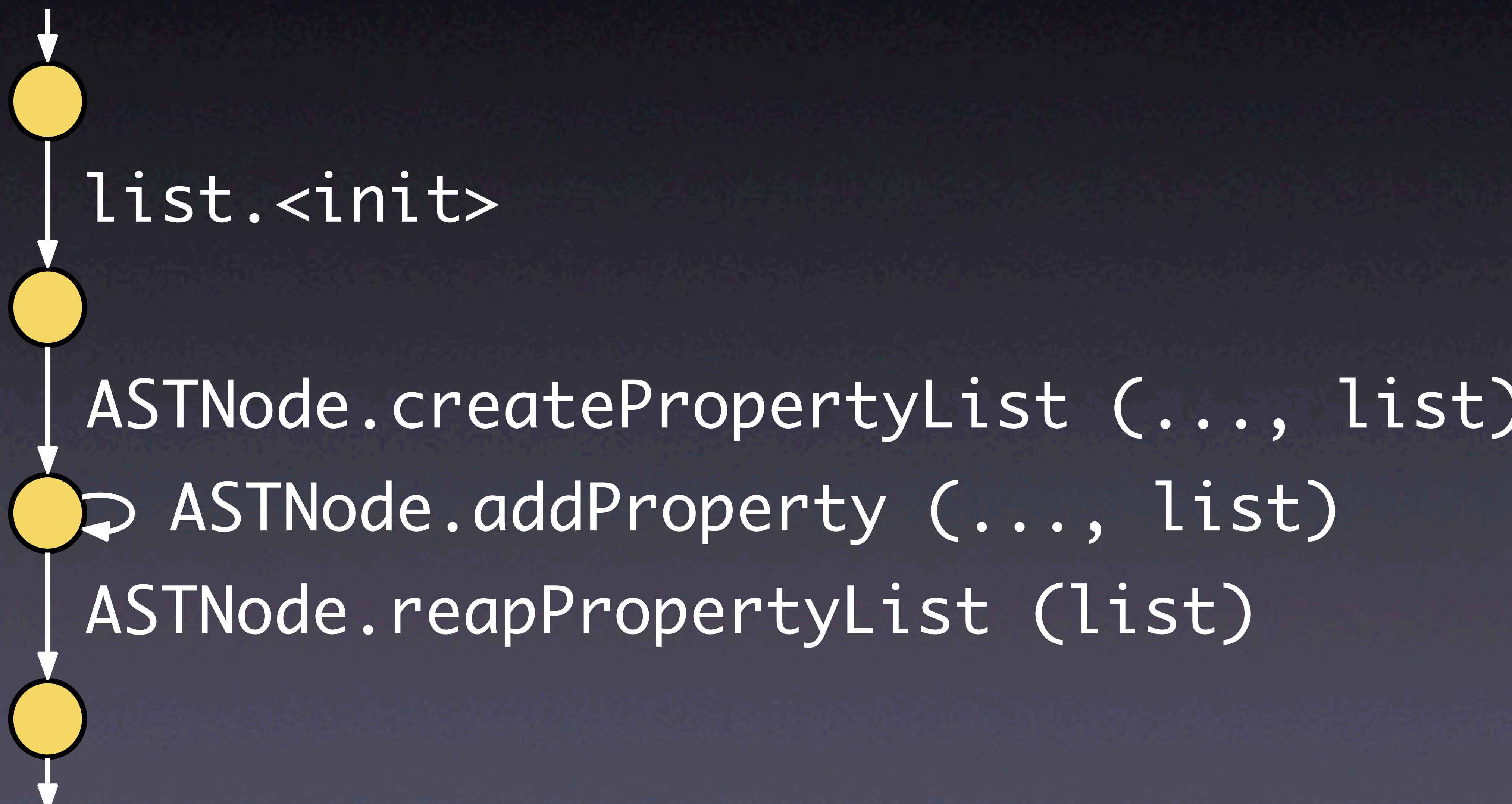
Usage Models



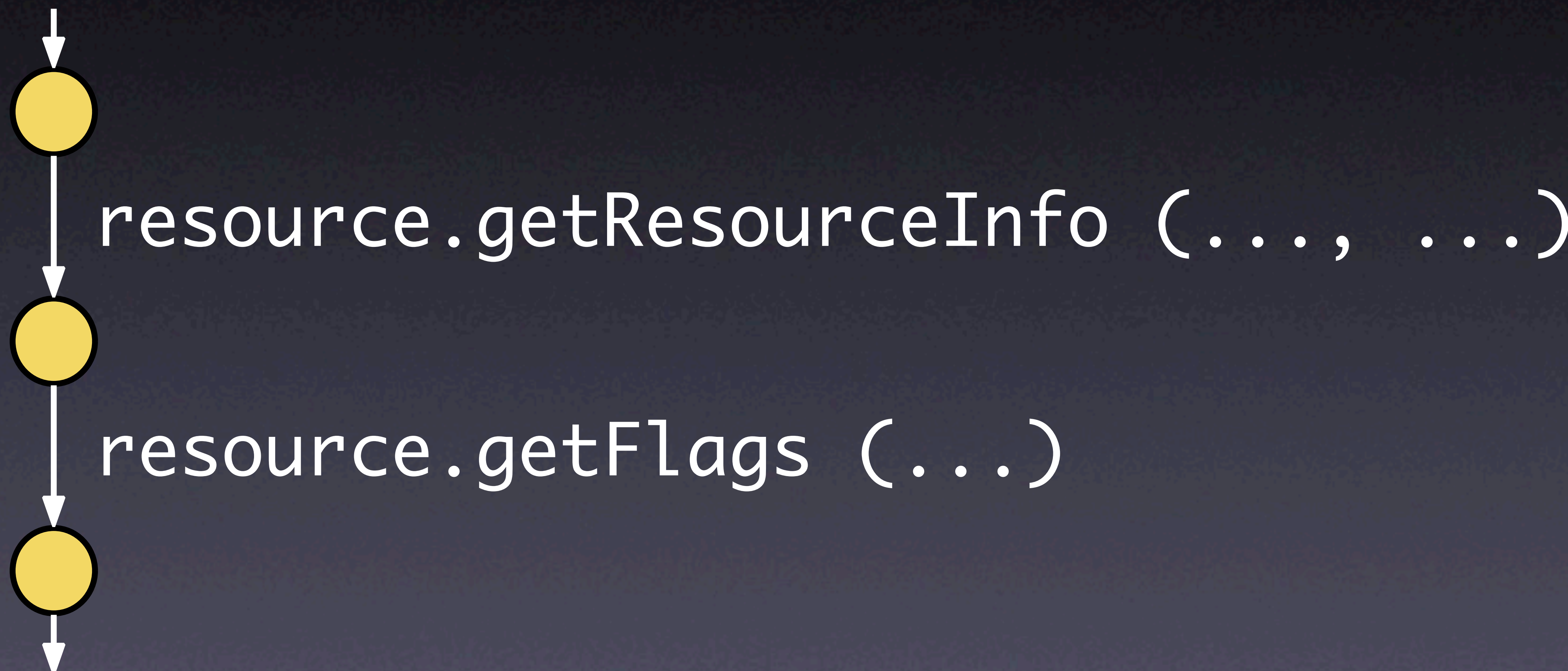
JPanel.add()



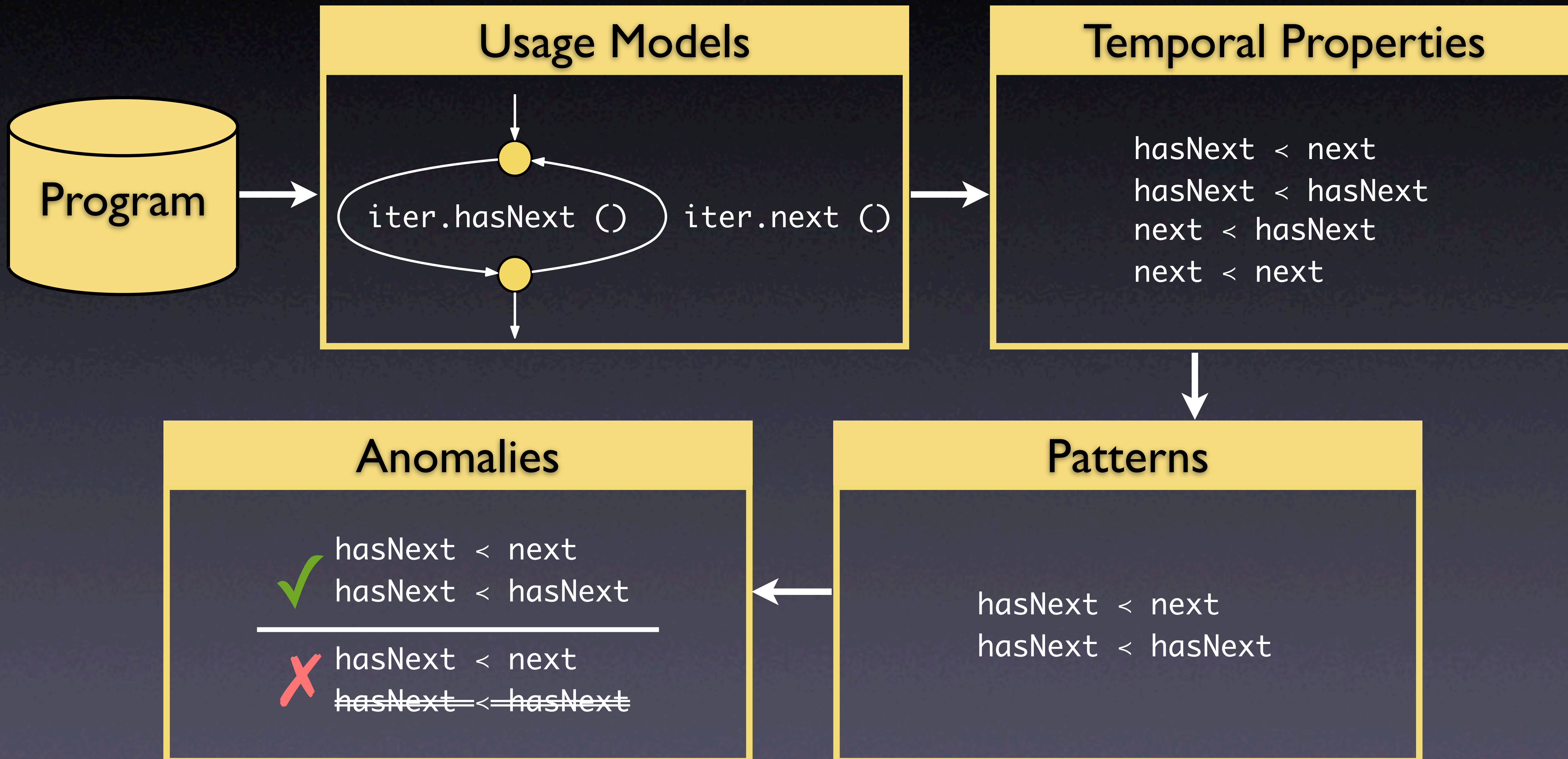
ASTNode.reapPropertyList()



Resource.getFlags()

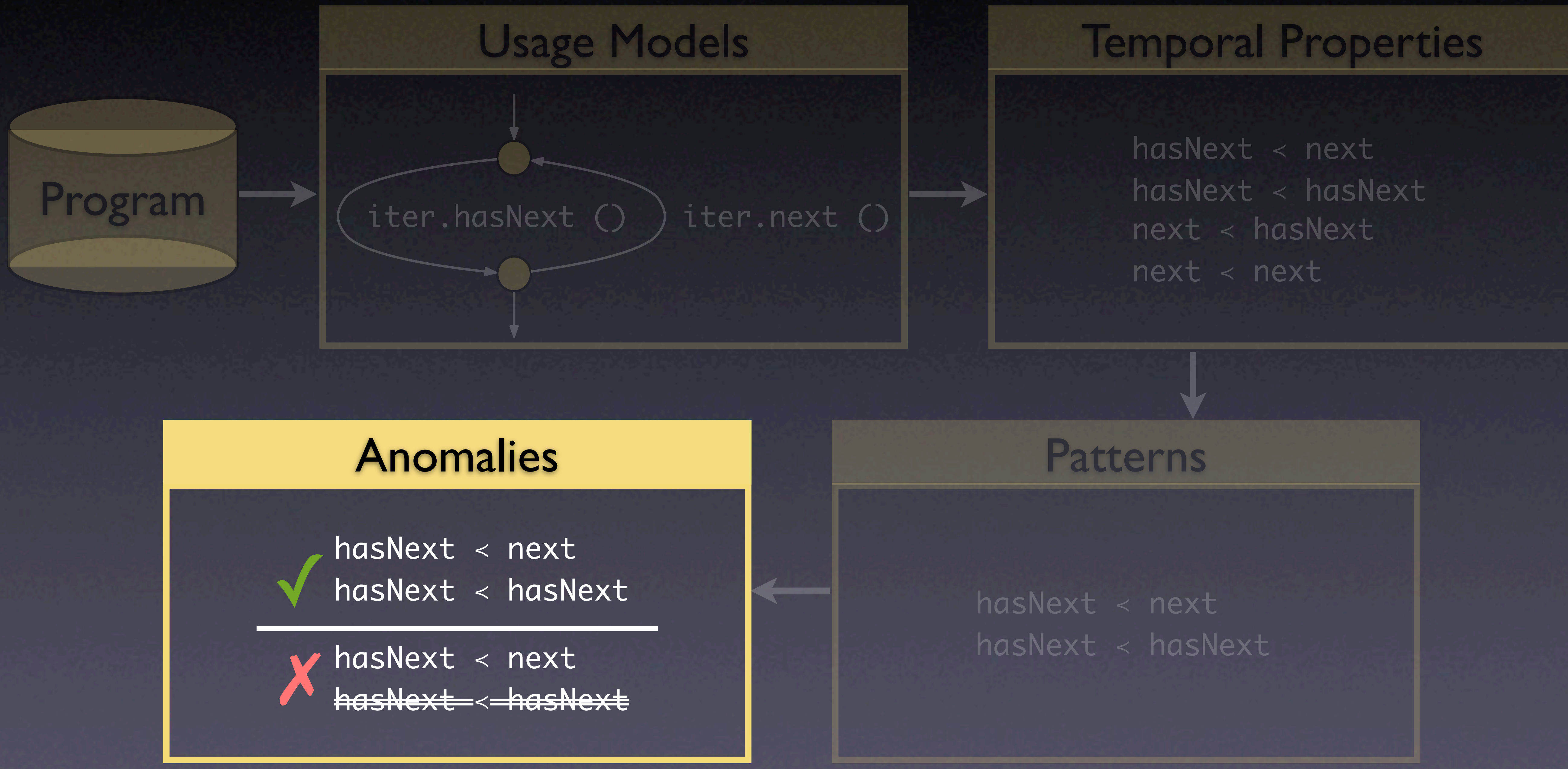


OP-Miner



Usage Models
iter.hasNext ()
iter.next ()
Program
Temporal Properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
~~hasNext < hasNext~~

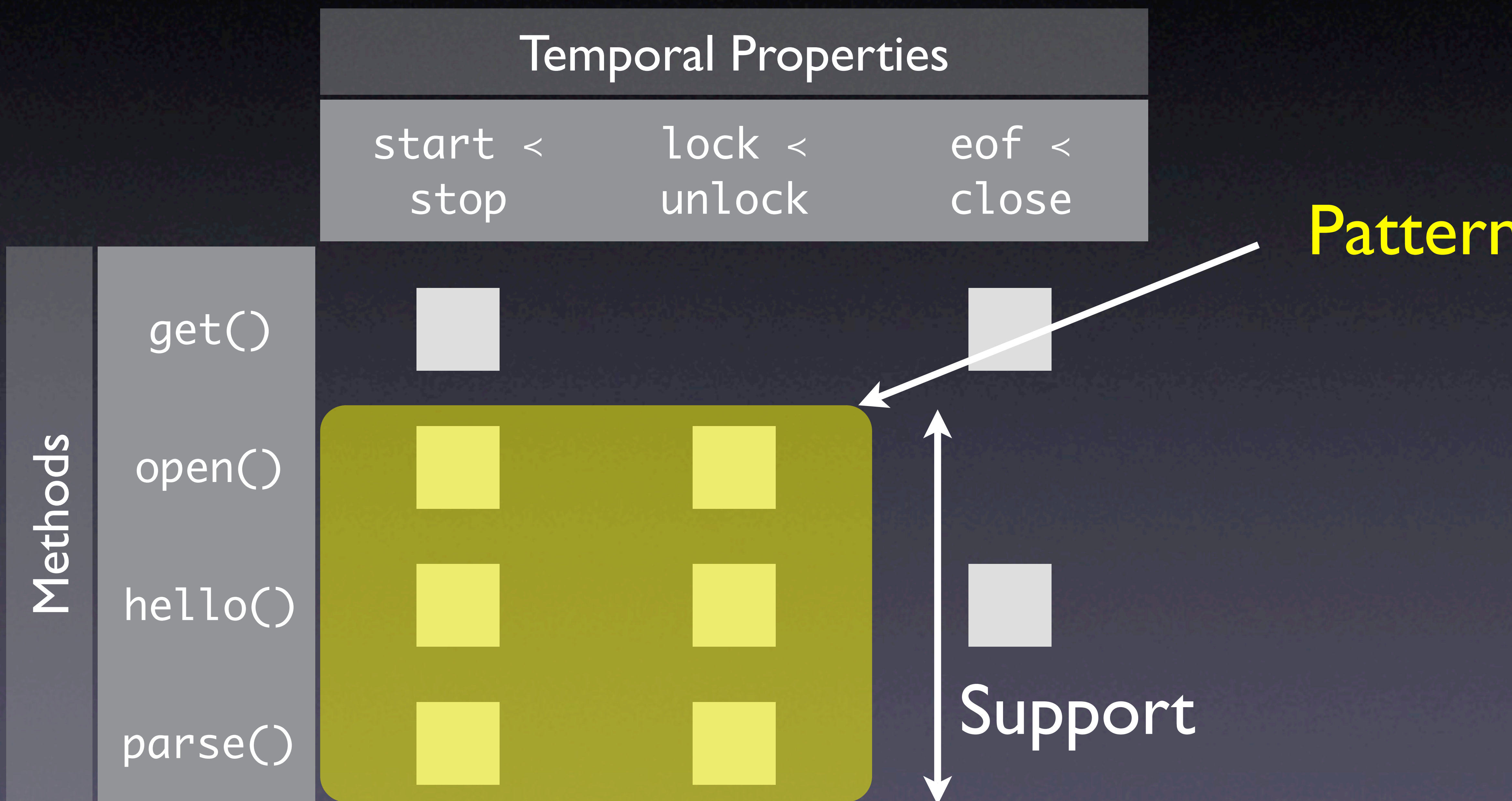
OP-Miner


Program
Usage Models
iter.hasNext ()
iter.next ()
Temporal Properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
~~hasNext < hasNext~~

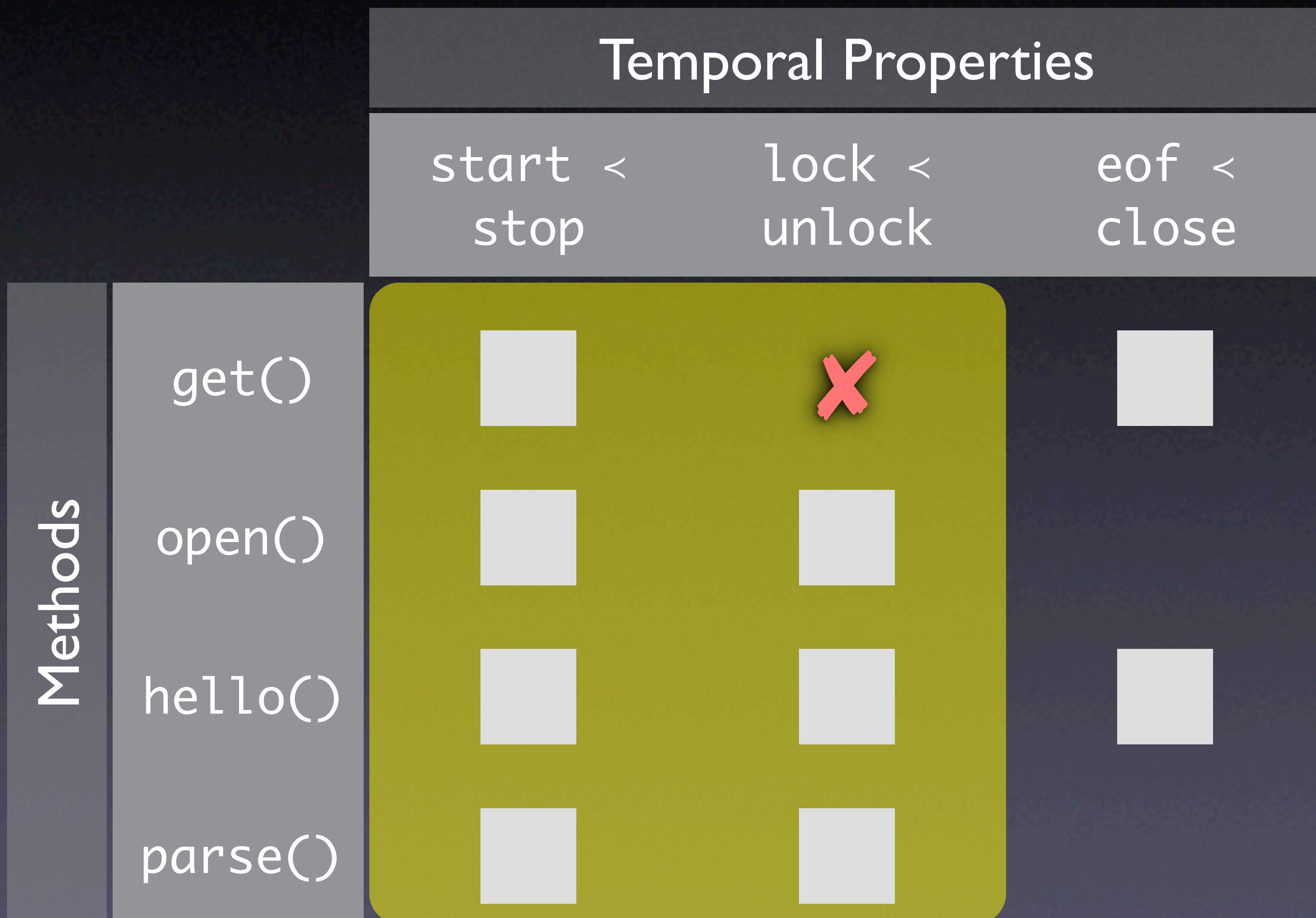
Methods vs. Properties

		Temporal Properties		
		start < stop	lock < unlock	eof < close
Methods	get()	■		■
	open()	■	■	
	hello()	■	■	■
	parse()	■	■	

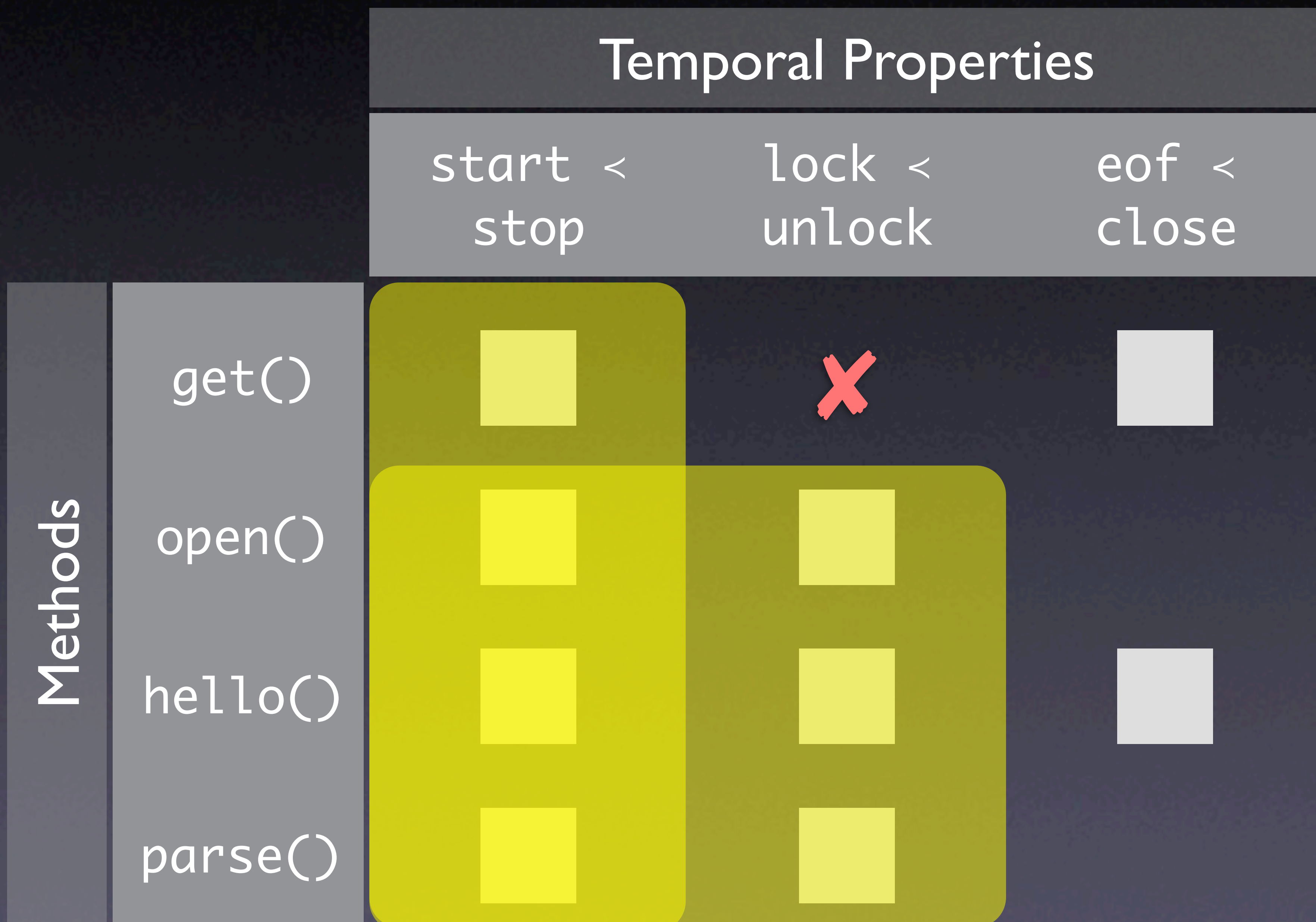
Methods vs. Properties



Discovering Anomalies



Discovering Anomalies



Case Study: AspectJ

- 2,954 classes
- 36,045 methods
- 1,154 methods with OP support ≥ 20
- 300 violations found in 8 minutes
- Examined every single one

A Defect

```
for (Iterator iter = itdFields.iterator();
     iter.hasNext();) {
    ...
    for (Iterator iter2 = worthRetrying.iterator();
         iter.hasNext();) {
         ...  
    }
}
```

should be iter2

bug.aj

```
@interface A {}

aspect Test {
   declare @field : @A int var* : @A;
   declare @field : int var* : @A;

   interface Subject {}

   public int Subject.vara;
   public int Subject.varb;
}

class X implements Test.Subject {}
```

pippin: fse2008 — bash — 80×24

```
[141] $
```

Another Defect

```
public void visitNEWARRAY (NEWARRAY o) {
   byte t = o.getTypecode ();
   if (!((t == Constants.T_BOOLEAN) ||
         (t == Constants.T_CHAR) ||
          ...
        (t == Constants.T_LONG))) {
      constraintViolated (o, "(...) '+t+' (...)");
   }
}
```

should be double quotes

A False Positive

```
Name internalNewName (String[] identifiers)
  ...
  for (int i = 1; i < count; i++) {
    SimpleName name = new SimpleName(this);
    name.internalSetIdentifier(identifiers[i]);
    ...
  }
  ...
}
```

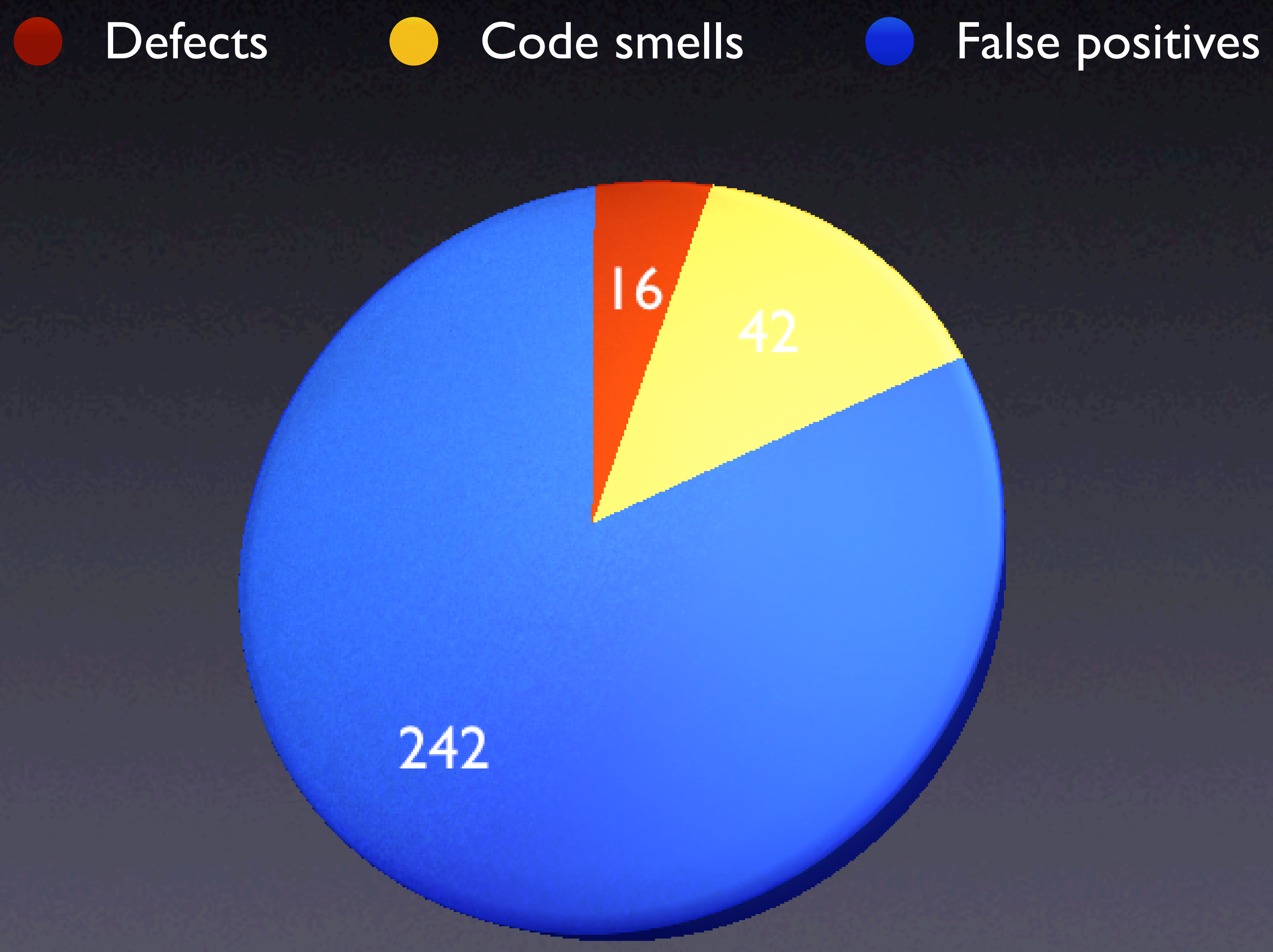
should stay as is

A Code Smell

```
public String getRetentionPolicy ()
{
   ...
   for (Iterator it = ...; it.hasNext();)
   {
         ... = it.next();
         ...
         return retentionPolicy;
   }
   ...
}
```

should be fixed

AspectJ

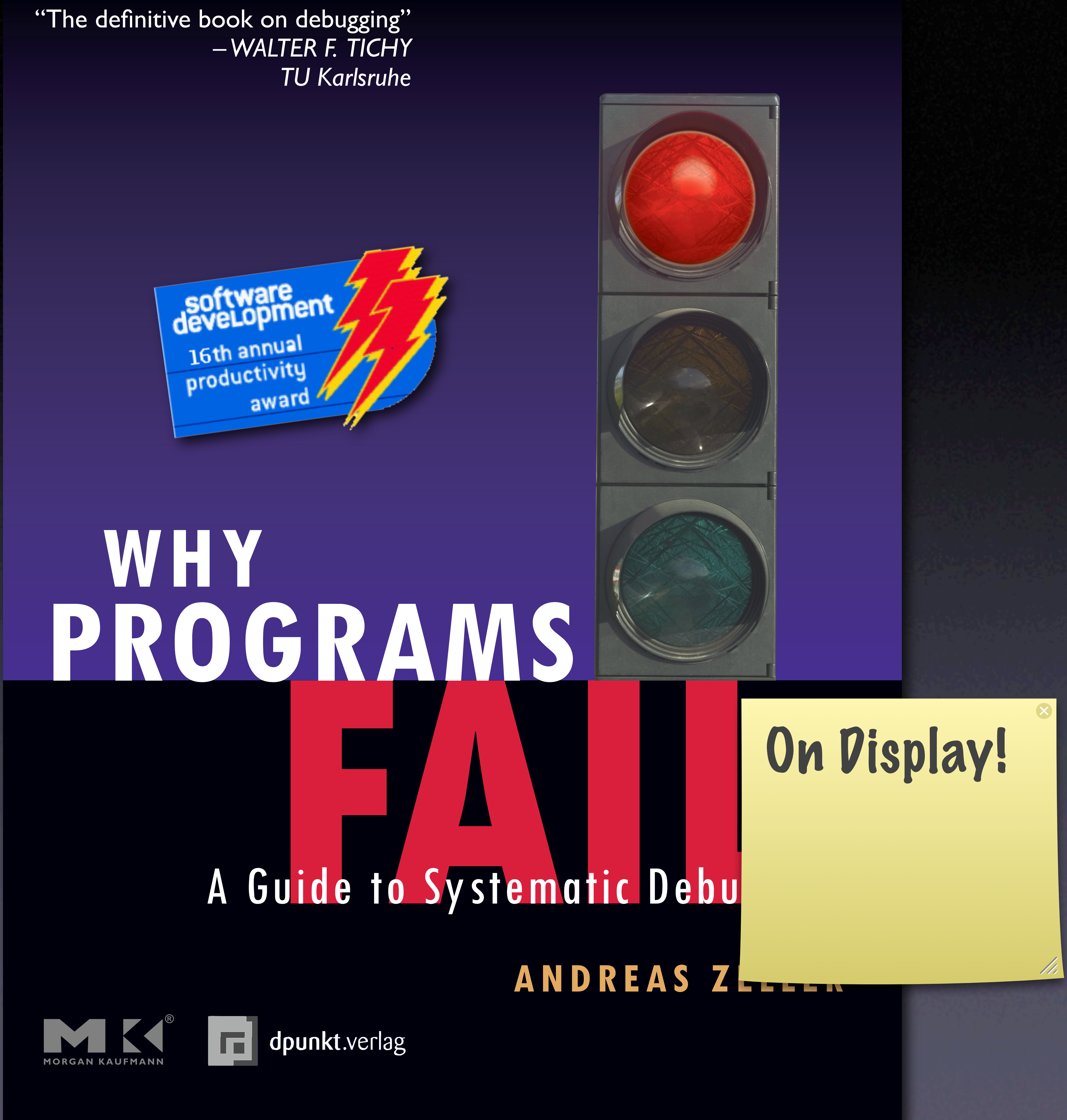


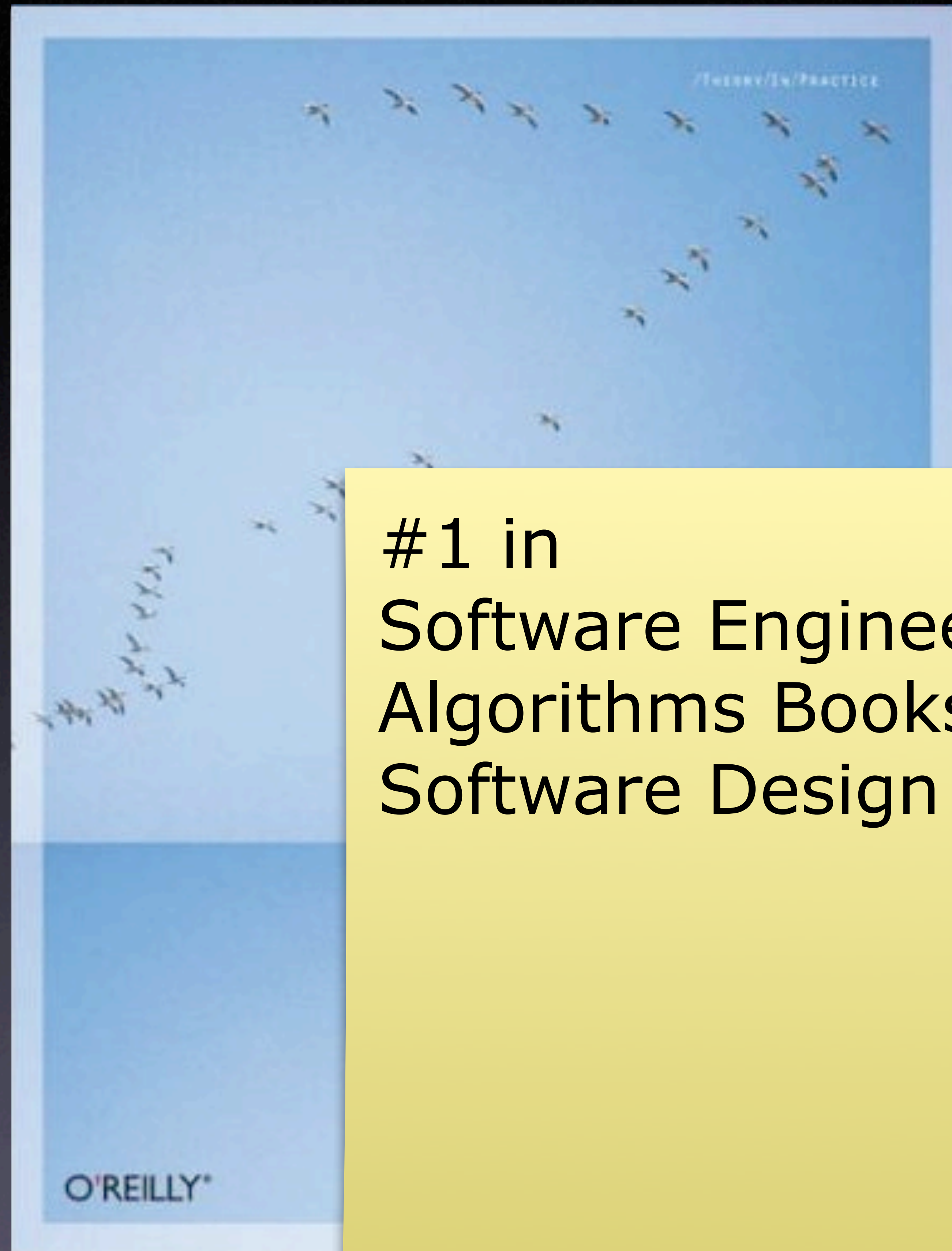
More Results

Program	# Violations		# Defects	# Code smells	# False positives	Efficiency
	Total	Investigated				
Act-Rbot 0.8.2	25	25	2	13	10	**60%**
Apache Tomcat 6.0.16	55	55	0	9	46	**16%**
ArgoUML 0.24	305	28	0	12	16	**43%**
AspectJ 1.5.3	300	300	16	42	242	**19%**
Azureus 2.5.0.0	315	85	1	26	58	**32%**
Columba 1.2	57	57	4	15	38	**33%**
jEdit 4.2	11	11	0	4	7	**36%**
	1,068	**562**	**23**	**121**	**417**	**26%**

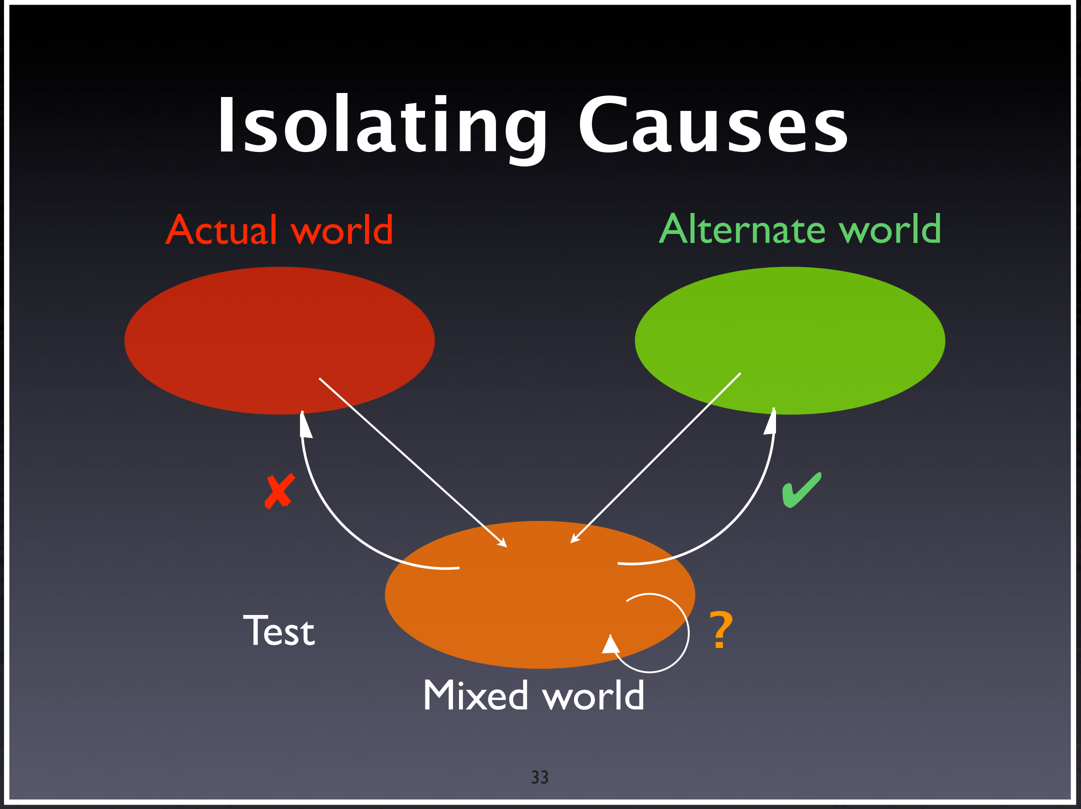
OP-Miner

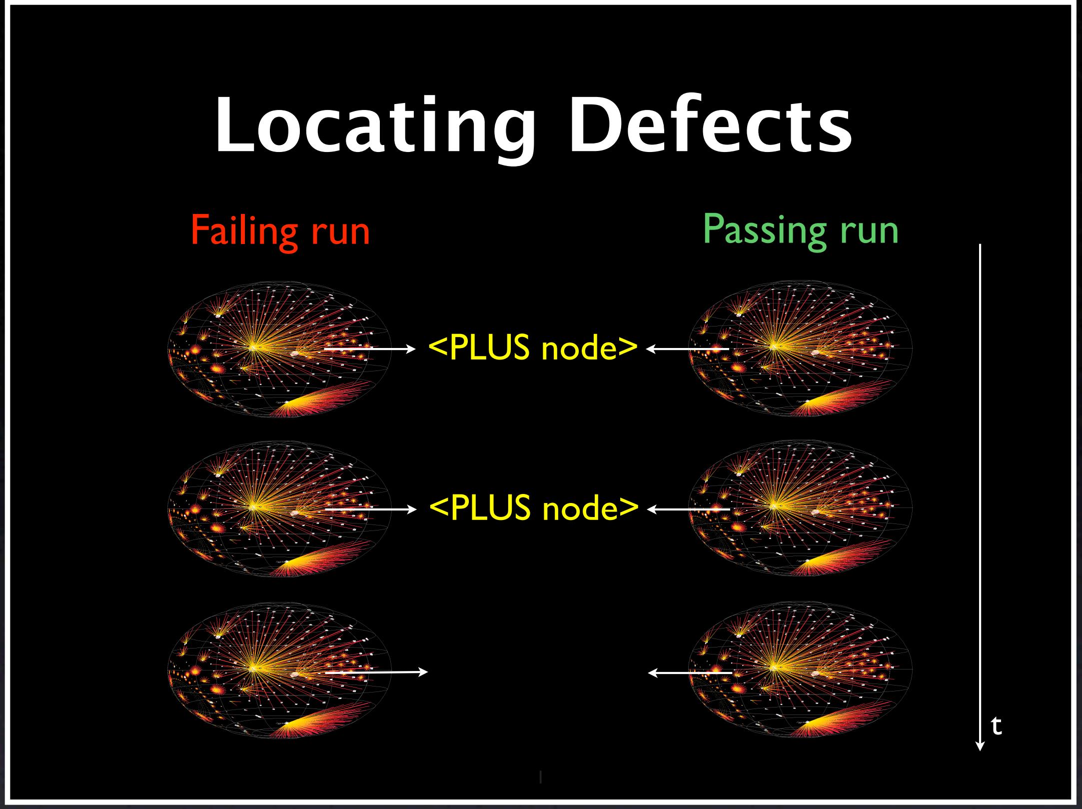
- OP-Miner learns *operational preconditions*
  i.e., how to typically construct arguments
- Learns from normal argument usage
  for specific projects or across projects
- Fully automatic
- Found dozens of verified defects

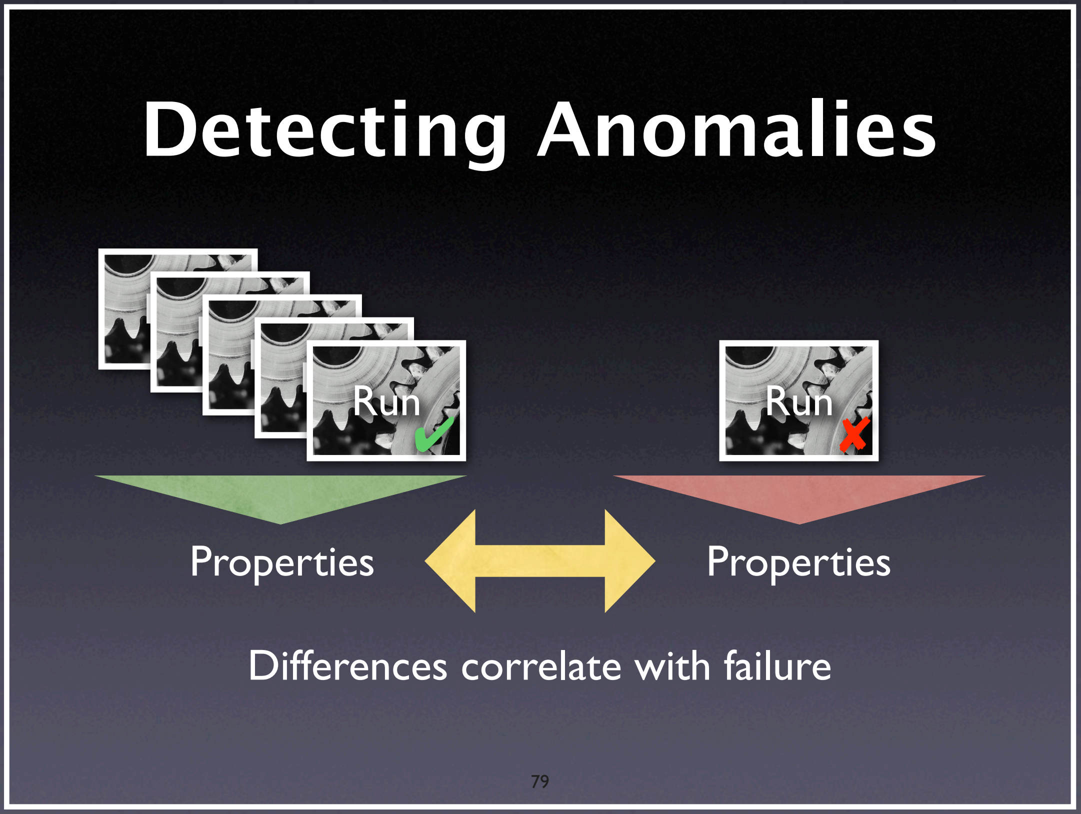

"The definitive book on debugging"
—WALTER F. TICHY
TU Karlsruhe
software development
16th annual productivity award
WHY
PROGRAMS
FAI
A Guide to Systematic Debu
ANDREAS Z
MORGAN KAUFMANN
dpunkt.verlag
On Display!

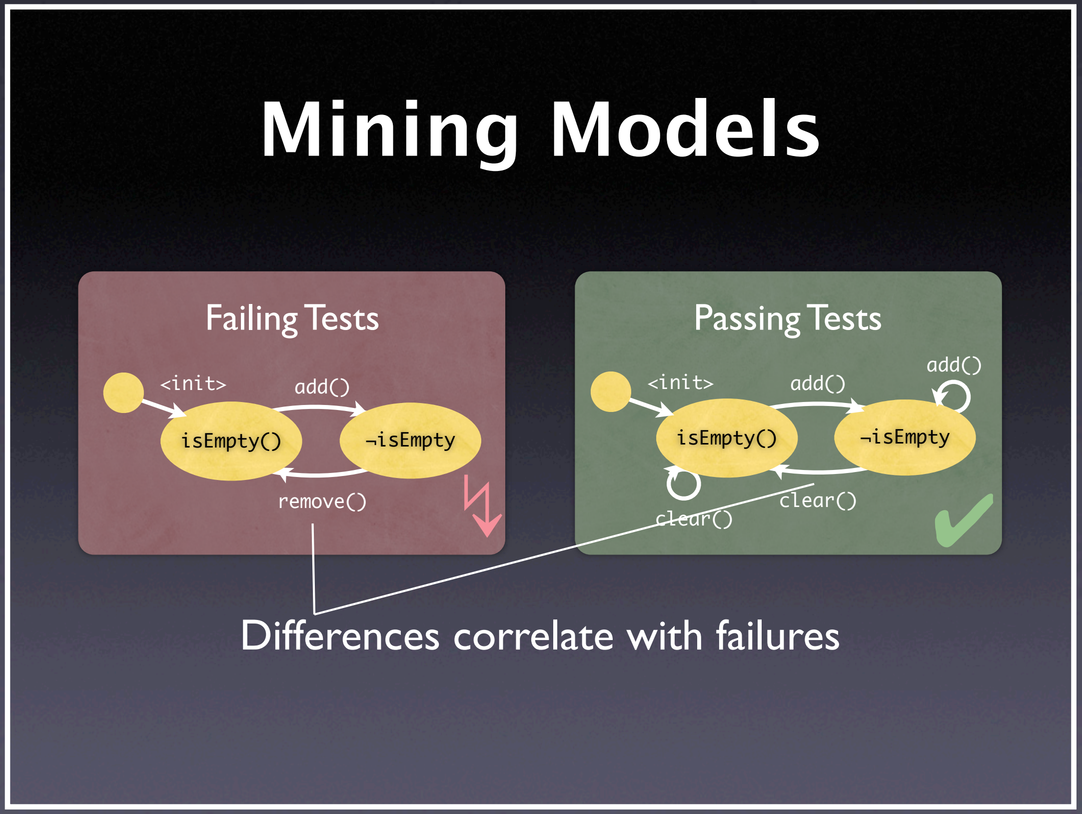


#1 in
Software Engineering Books
Algorithms Books
Software Design Books


Isolating Causes
Actual world
Alternate world
✘
✔
Test
?
Mixed world
33


Locating Defects
Failing run
Passing run
<PLUS node>
<PLUS node>
t


Detecting Anomalies
Run
Run
Properties
Properties
Differences correlate with failure
79


Mining Models
Failing Tests
<init>
add()
isEmpty()
¬isEmpty
remove()
Passing Tests
<init>
add()
add()
isEmpty()
¬isEmpty
clear()
clear()
Differences correlate with failures