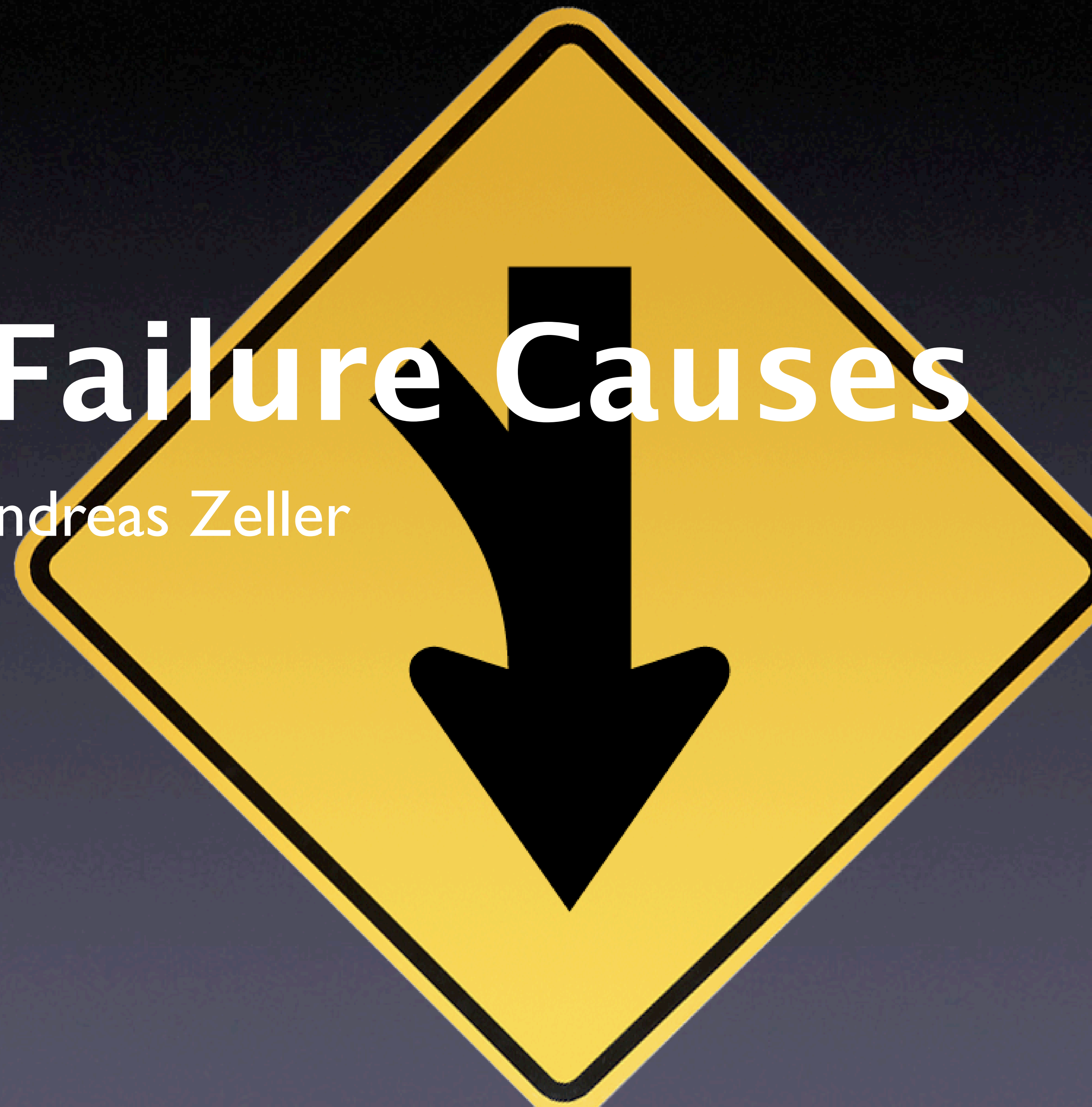
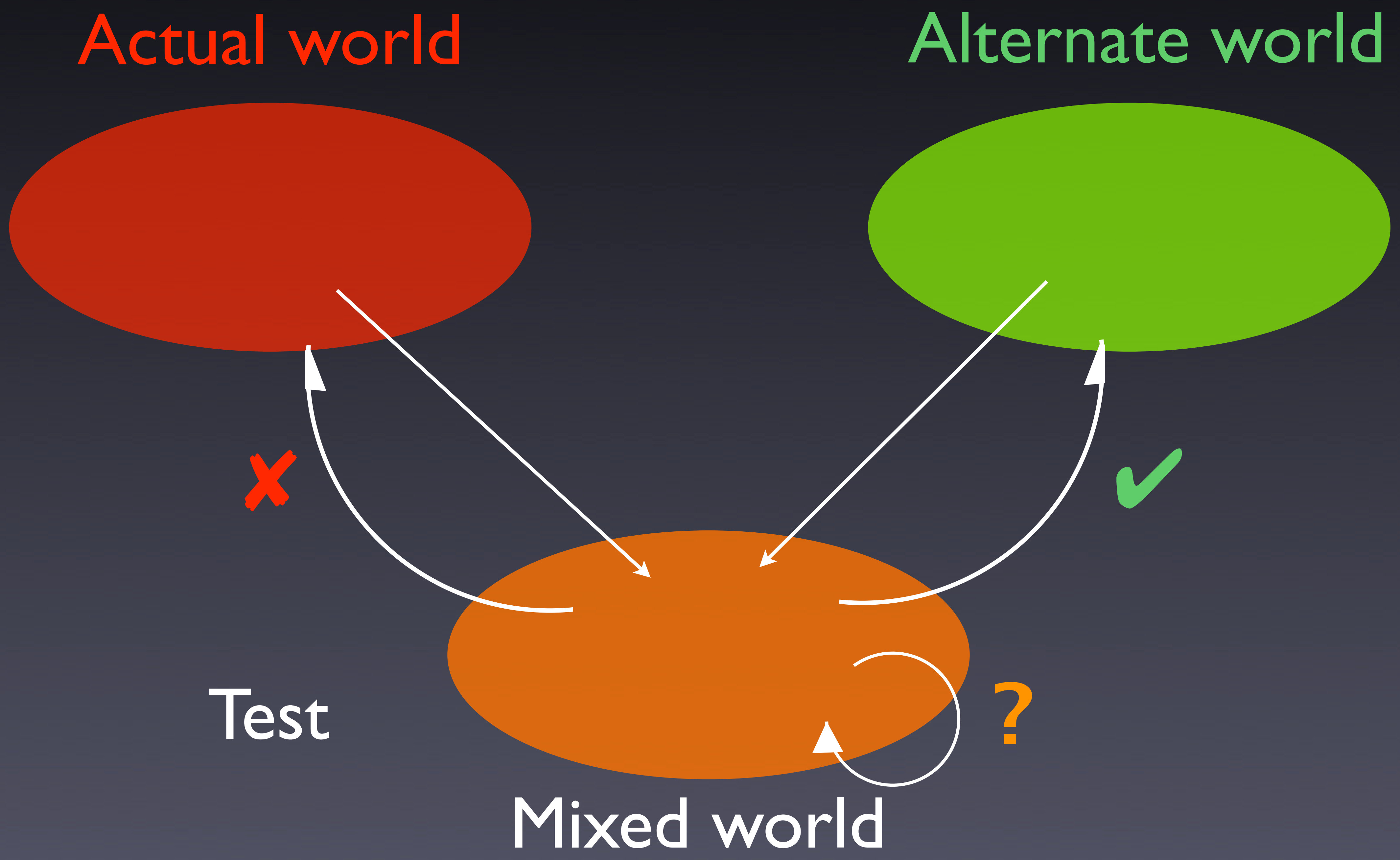
Isolating Failure Causes

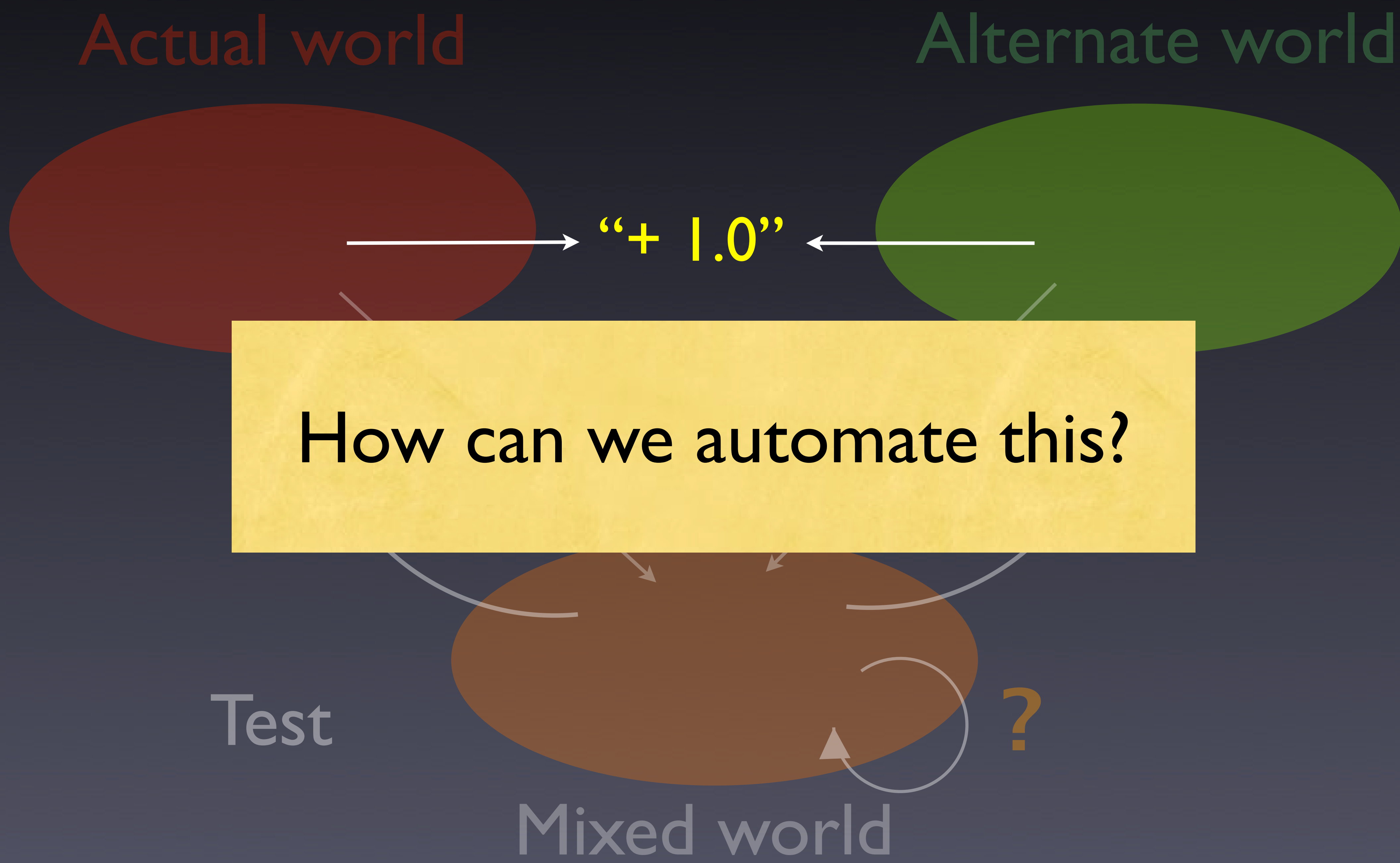
Andreas Zeller



Isolating Causes



Isolating Causes



How can we automate this?

Simplifying 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> ✔

Simplifying

Isolating Input

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

Difference narrowed down

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

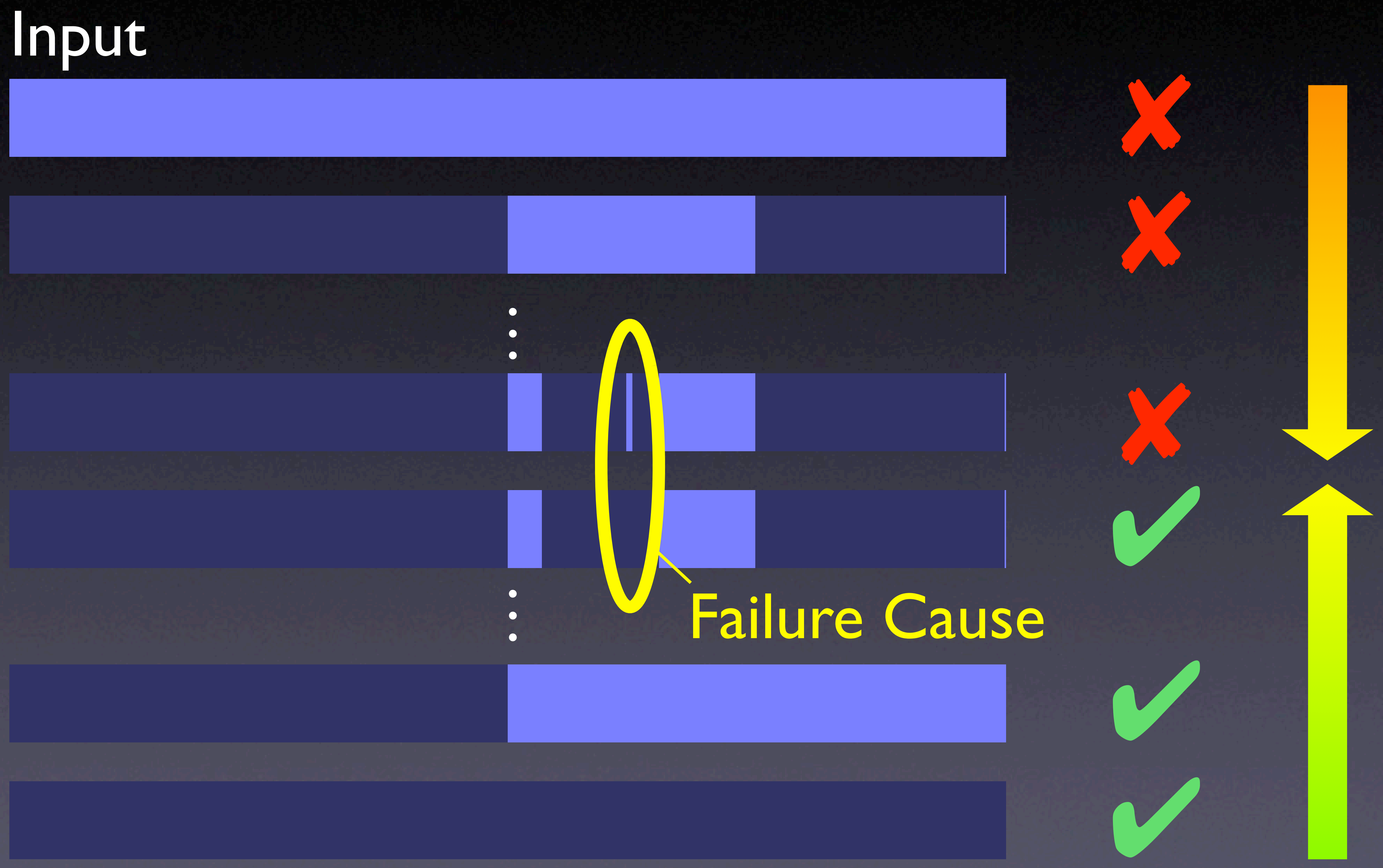
`<SELECT NAME="priority" MULTIPLE SIZE=7>` ✔

Isolating Input

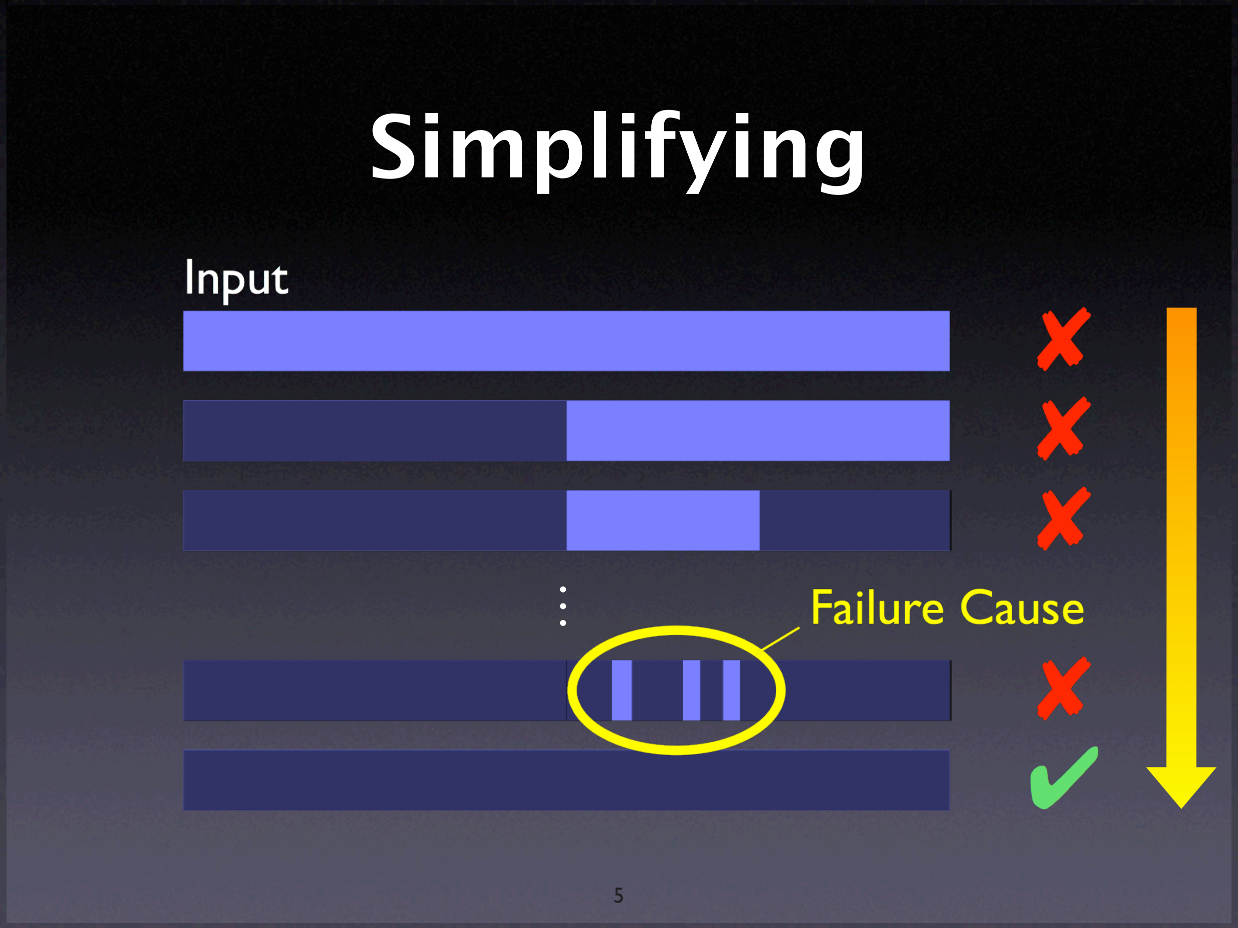
```
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<SELECT NAME="priority" MULTIPLE SIZE=7>
```

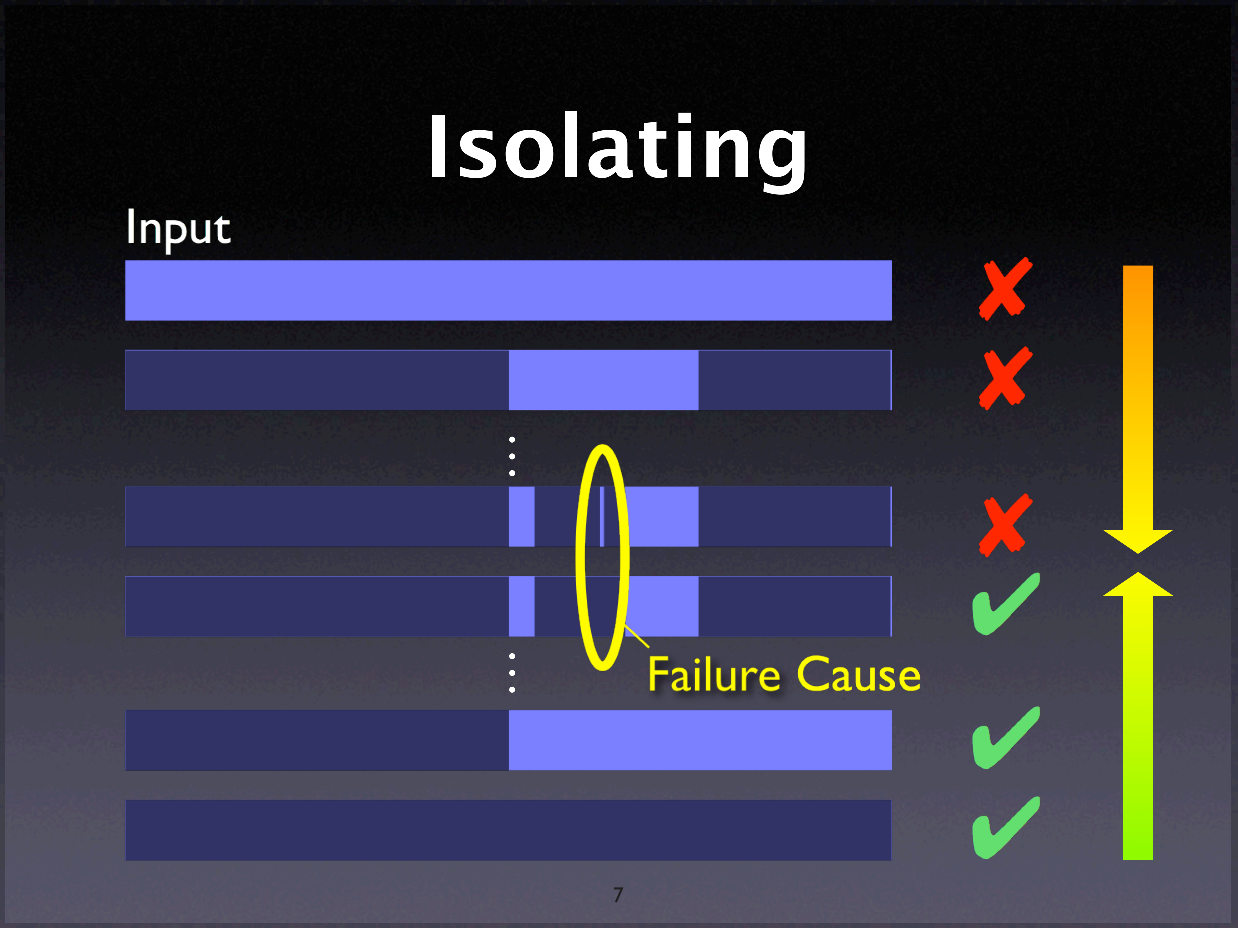
Failure Cause

Isolating



Finding Causes





- minimal input
- minimal context

- minimal difference
- common context

Configuration

Circumstance

$$\delta$$

All circumstances

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

Configuration $c \subseteq C$

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

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

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

Algorithm Sketch

- Extend *ddmin* such that it works on *two sets at a time* – $c'_{\times}$ and $c'_{\checkmark}$
- Compute subsets

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

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

Test Outcomes

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

most valuable outcomes

dd in a Nutshell

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

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

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

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

```
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:

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

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

$$t \leq \log_2(\Delta)$$

size of difference – no unresolved outcomes

$$|c'_{\text{✘}} \setminus c'_{\text{✔}}| = 1$$

Applications

Input	Code Changes	Schedules

Isolating Input

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

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

<SELECT NA

<SELECT NA

Failure

<SELECT NA

Isolation: 5 tests

Simplification: 48 tests

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

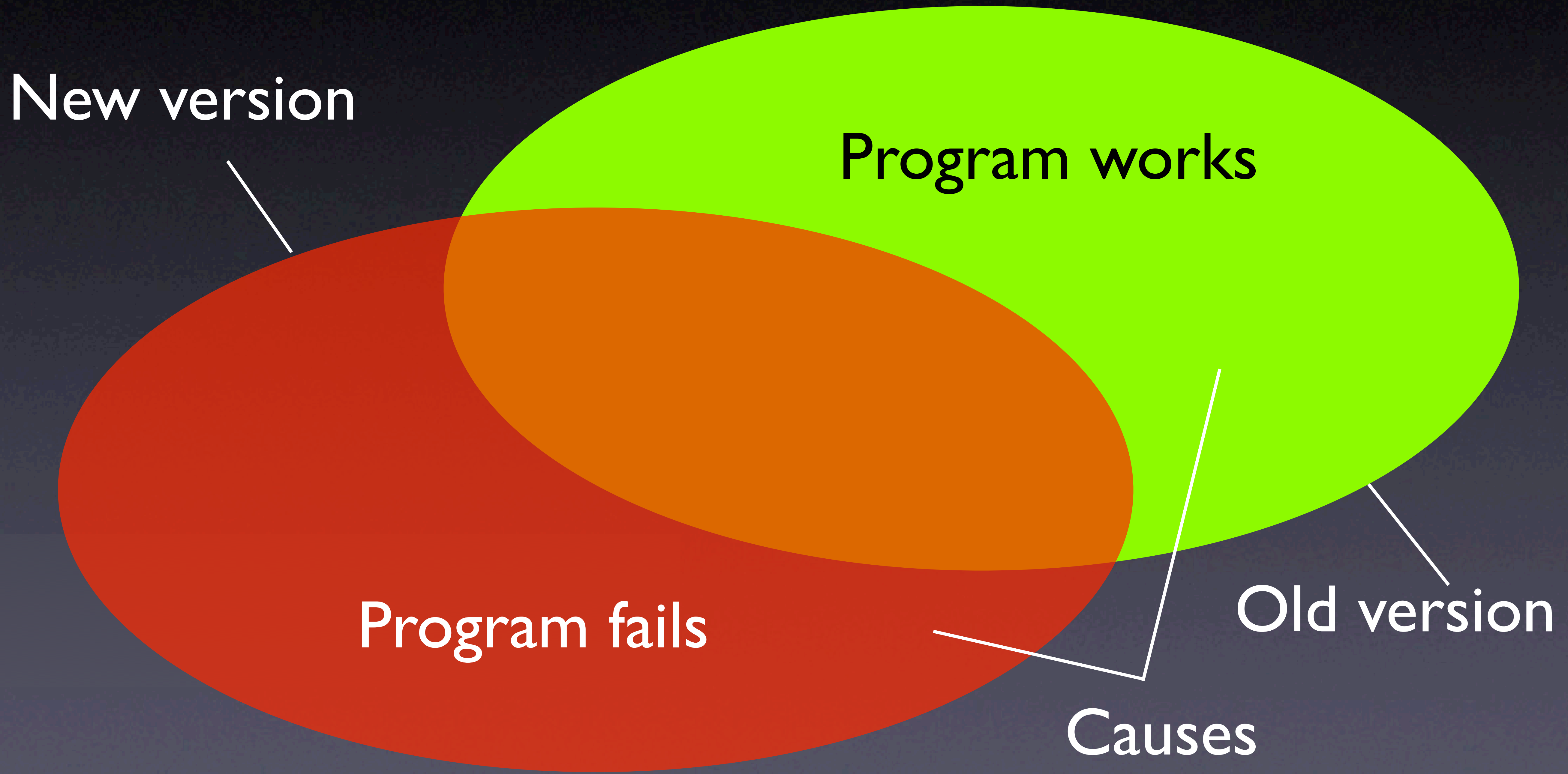
<SELECT NAME="priority" MULTIPLE SIZE=7>

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

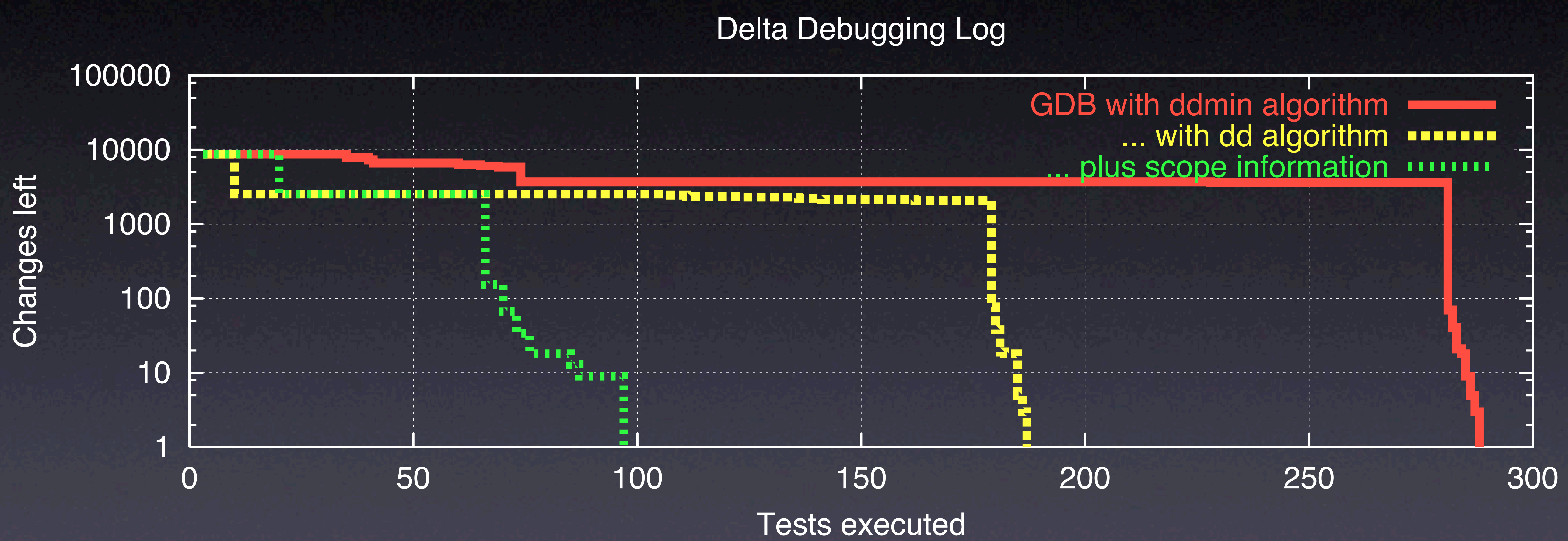
- Granularity – within some large change, only a few lines may be relevant
- Interference – some (later) changes rely on other (earlier) changes
- Inconsistency – some changes may have to be combined to produce testable code

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



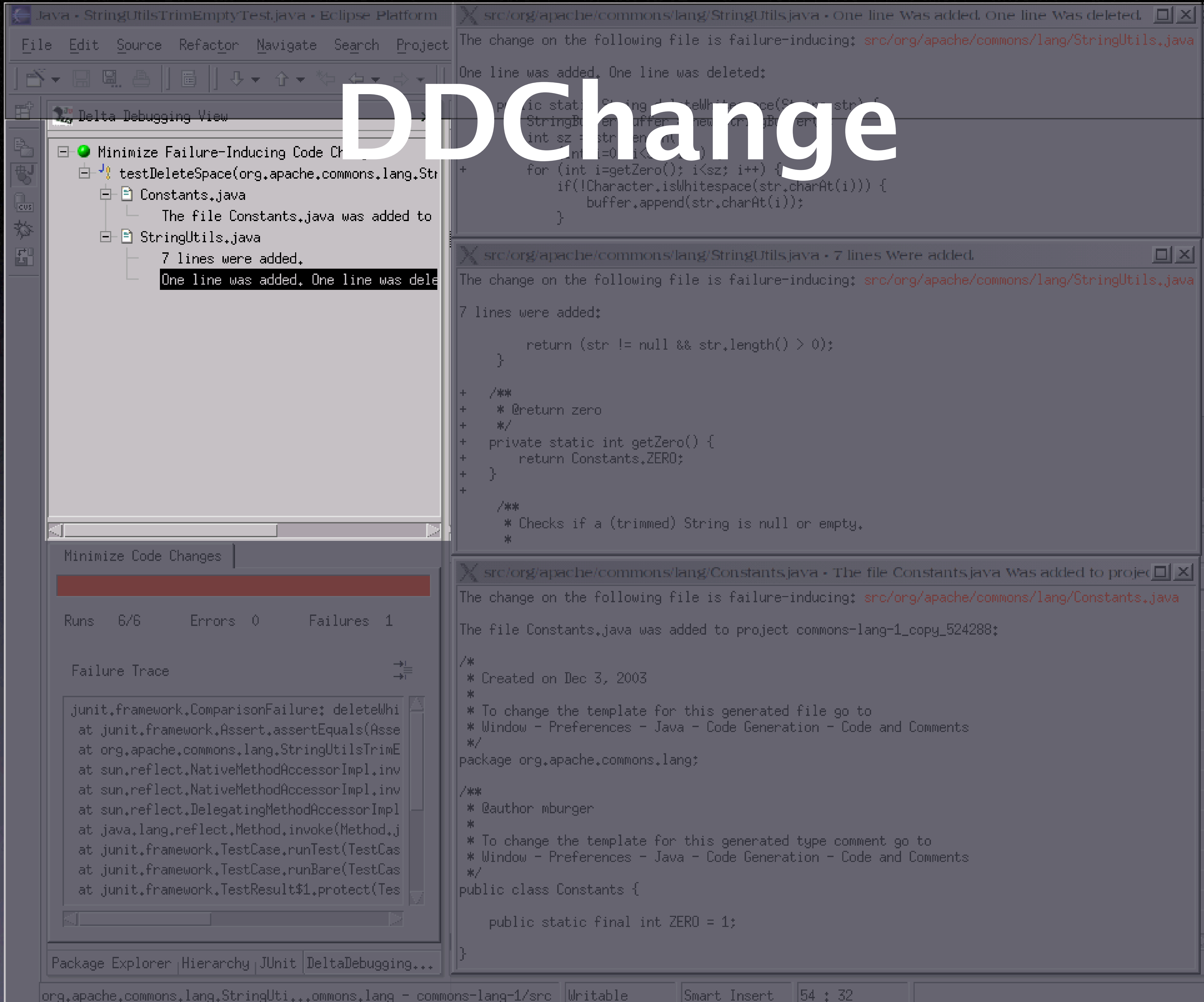
- Result after 98 tests (= 1 hour)

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

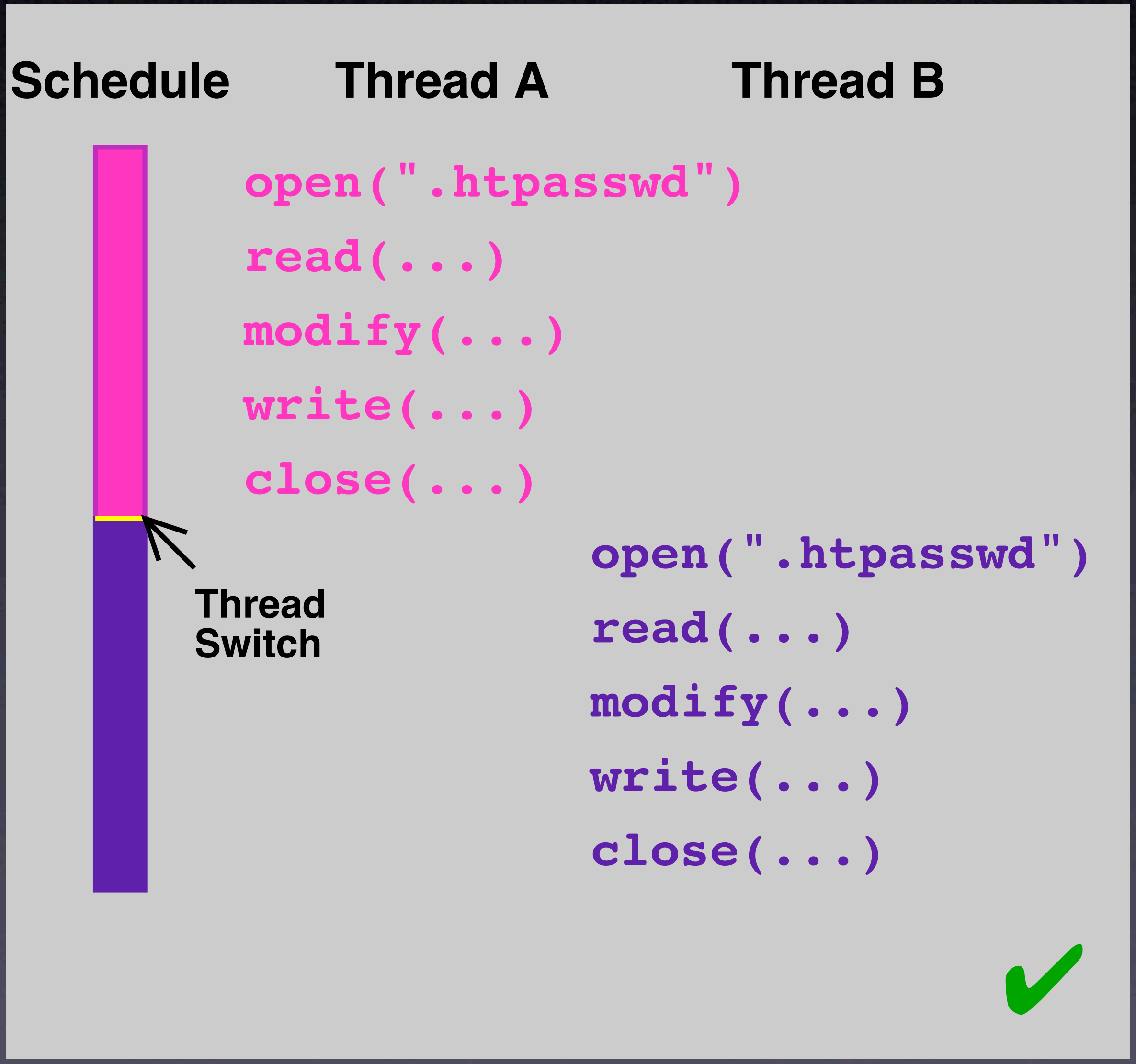


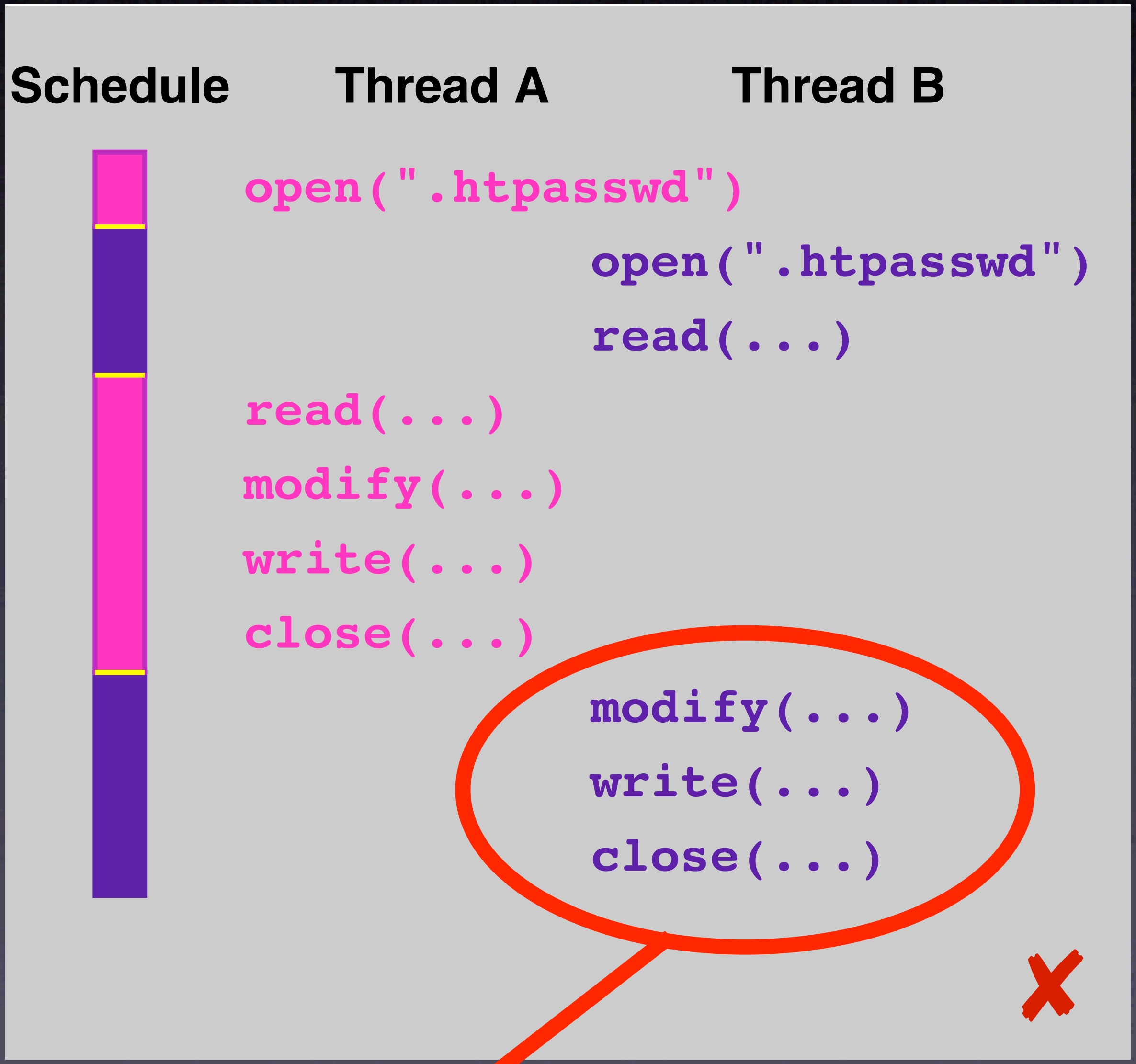
Java - StringUtilsTrimEmptyTest.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project
Delta Debugging View
Minimize Failure-Inducing Code Ch
testDeleteSpace(org.apache.commons.lang.Str
Constants.java
The file Constants.java was added to
StringUtils.java
7 lines were added.
One line was added. One line was dele
Minimize Code Changes
Runs 6/6 Errors 0 Failures 1
Failure Trace
junit.framework.ComparisonFailure: deleteWhi
at junit.framework.Assert.assertEquals(Asse
at org.apache.commons.lang.StringUtilsTrimE
at sun.reflect.NativeMethodAccessorImpl.inv
at sun.reflect.NativeMethodAccessorImpl.inv
at sun.reflect.DelegatingMethodAccessorImpl
at java.lang.reflect.Method.invoke(Method.j
at junit.framework.TestCase.runTest(TestCas
at junit.framework.TestCase.runBare(TestCas
at junit.framework.TestResult$1.protect(Tes
Package Explorer Hierarchy JUnit DeltaDebugging...
src/org/apache/commons/lang/StringUtils.java - One line was added. One line was deleted.
The change on the following file is failure-inducing: src/org/apache/commons/lang/StringUtils.java
One line was added. One line was deleted:
for (int i=getZero(); i<sz; i++) {
if(!Character.isWhitespace(str.charAt(i))) {
buffer.append(str.charAt(i));
src/org/apache/commons/lang/StringUtils.java - 7 lines were added.
The change on the following file is failure-inducing: src/org/apache/commons/lang/StringUtils.java
7 lines were added:
return (str != null && str.length() > 0);
/**
* @return zero
*/
private static int getZero() {
return Constants.ZERO;
}
/**
* Checks if a (trimmed) String is null or empty.
src/org/apache/commons/lang/Constants.java - The file Constants.java was added to projec
The change on the following file is failure-inducing: src/org/apache/commons/lang/Constants.java
The file Constants.java was added to project commons-lang-1_copy_524288:
/*
* Created on Dec 3, 2003
*
* To change the template for this generated file go to
* Window - Preferences - Java - Code Generation - Code and Comments
*/
package org.apache.commons.lang;
/**
* @author mburger
*
* To change the template for this generated type comment go to
* Window - Preferences - Java - Code Generation - Code and Comments
*/
public class Constants {
public static final int ZERO = 1;
}
org.apache.commons.lang.StringUti...ommons.lang - commons-lang-1/src
Writable
Smart Insert
54 : 32

Optimizations

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

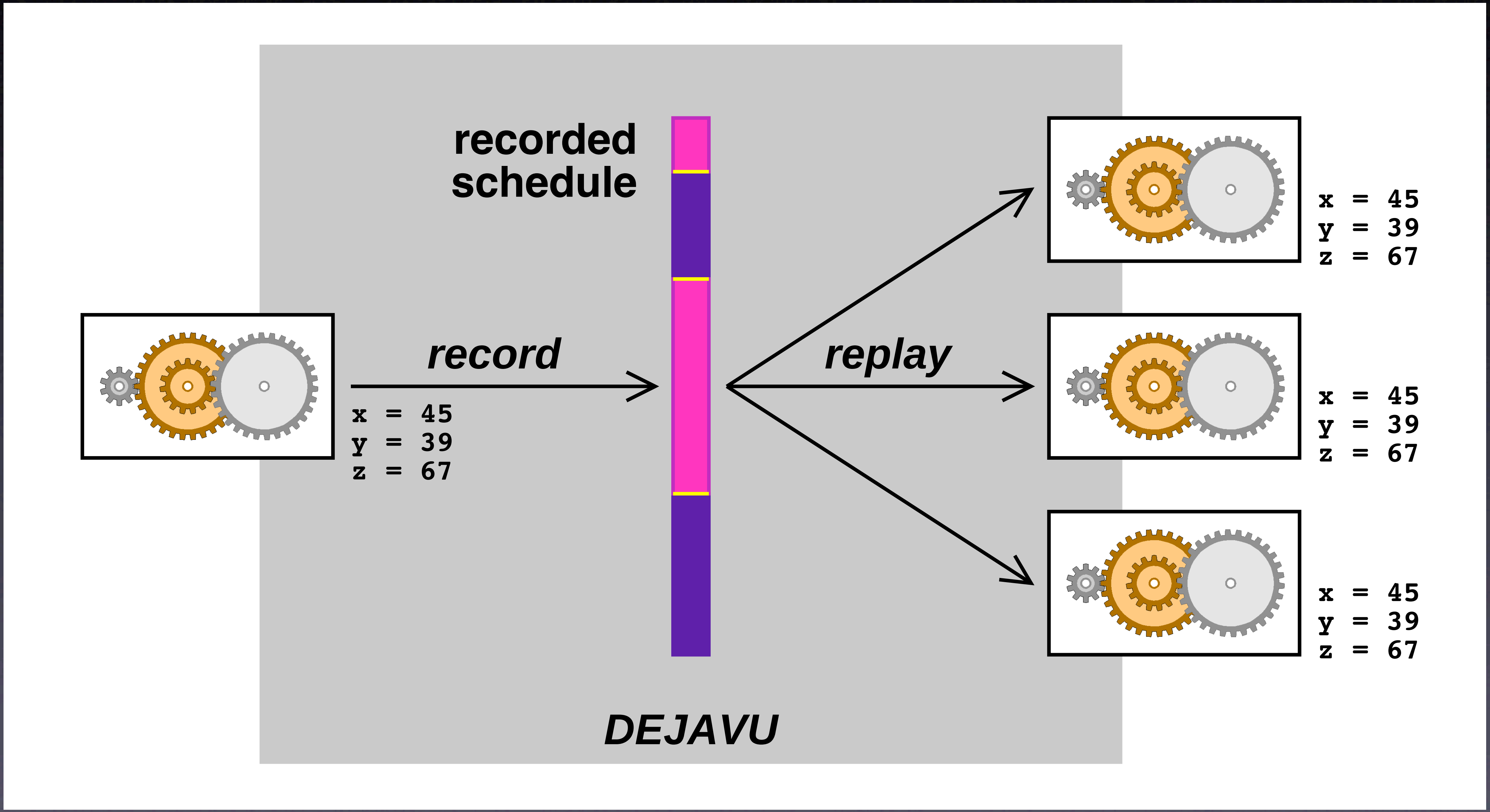
Thread Schedules



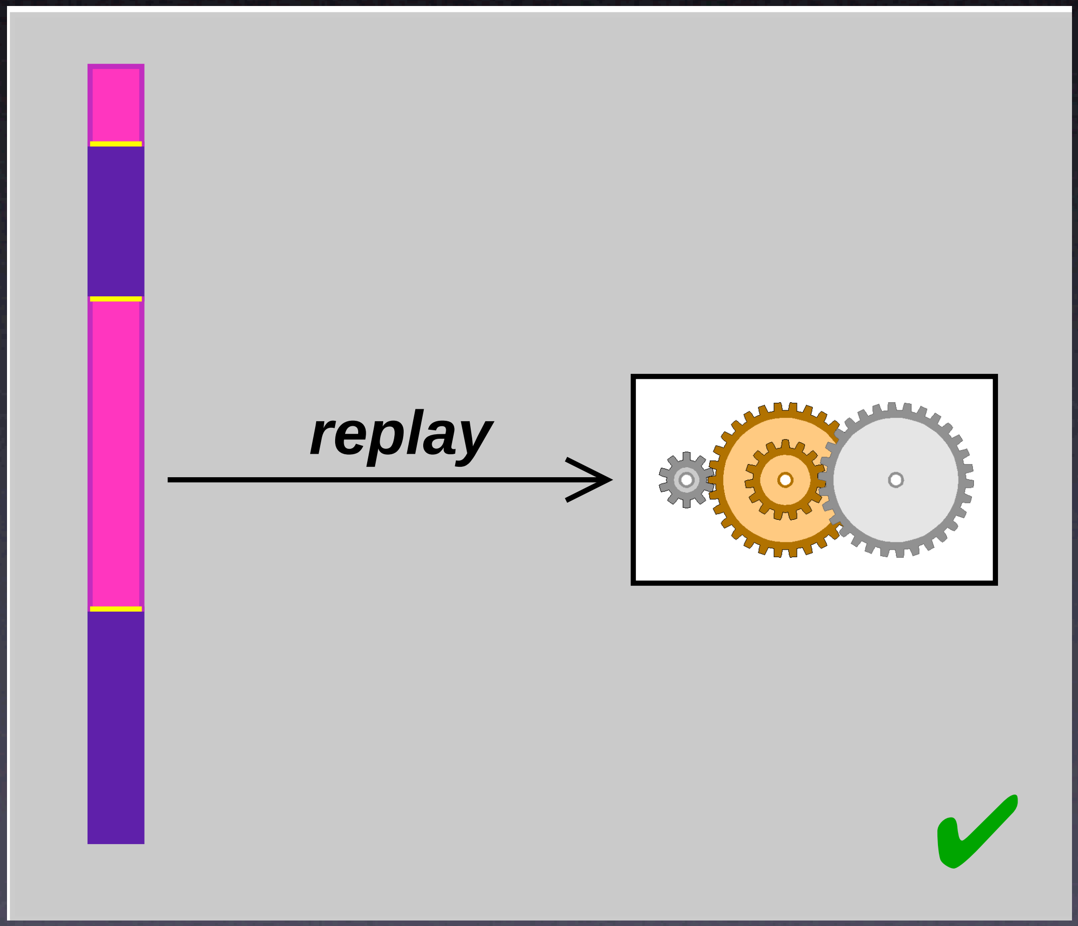


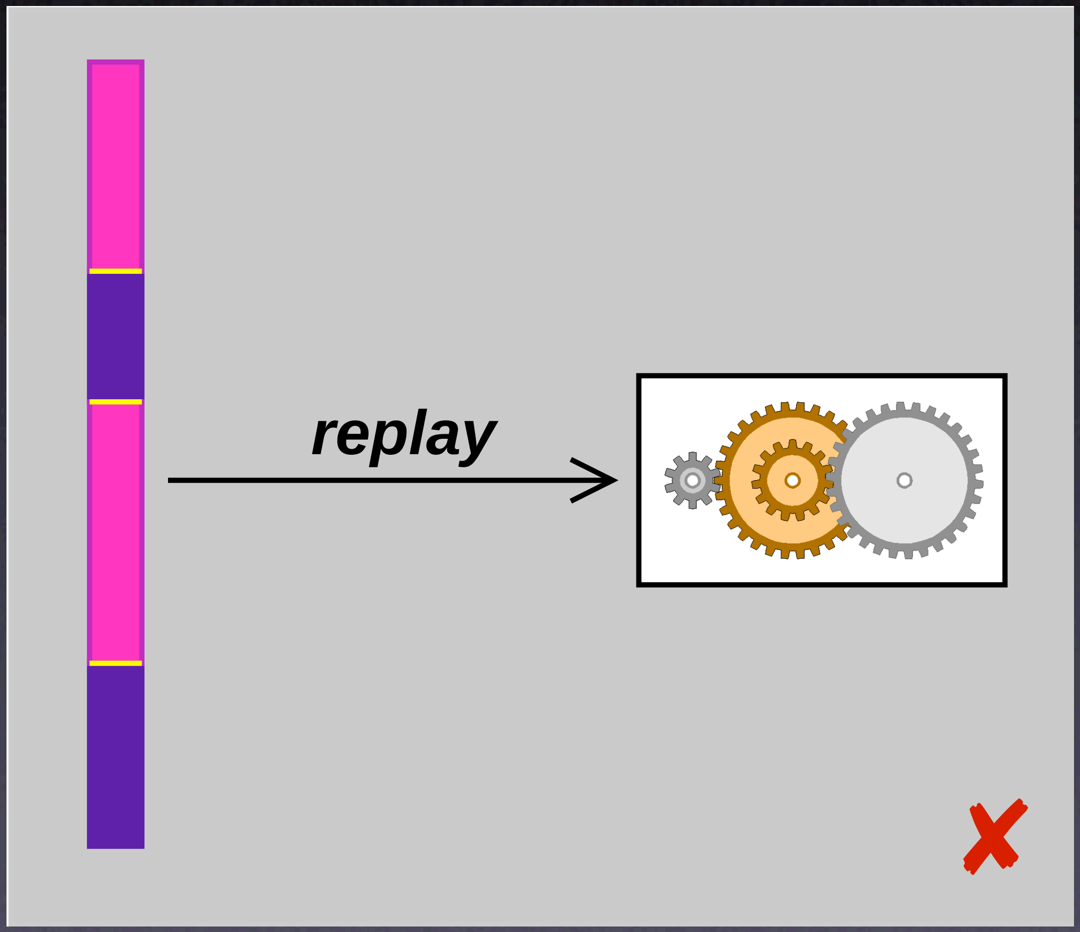
A’s updates get lost!

Record + Replay



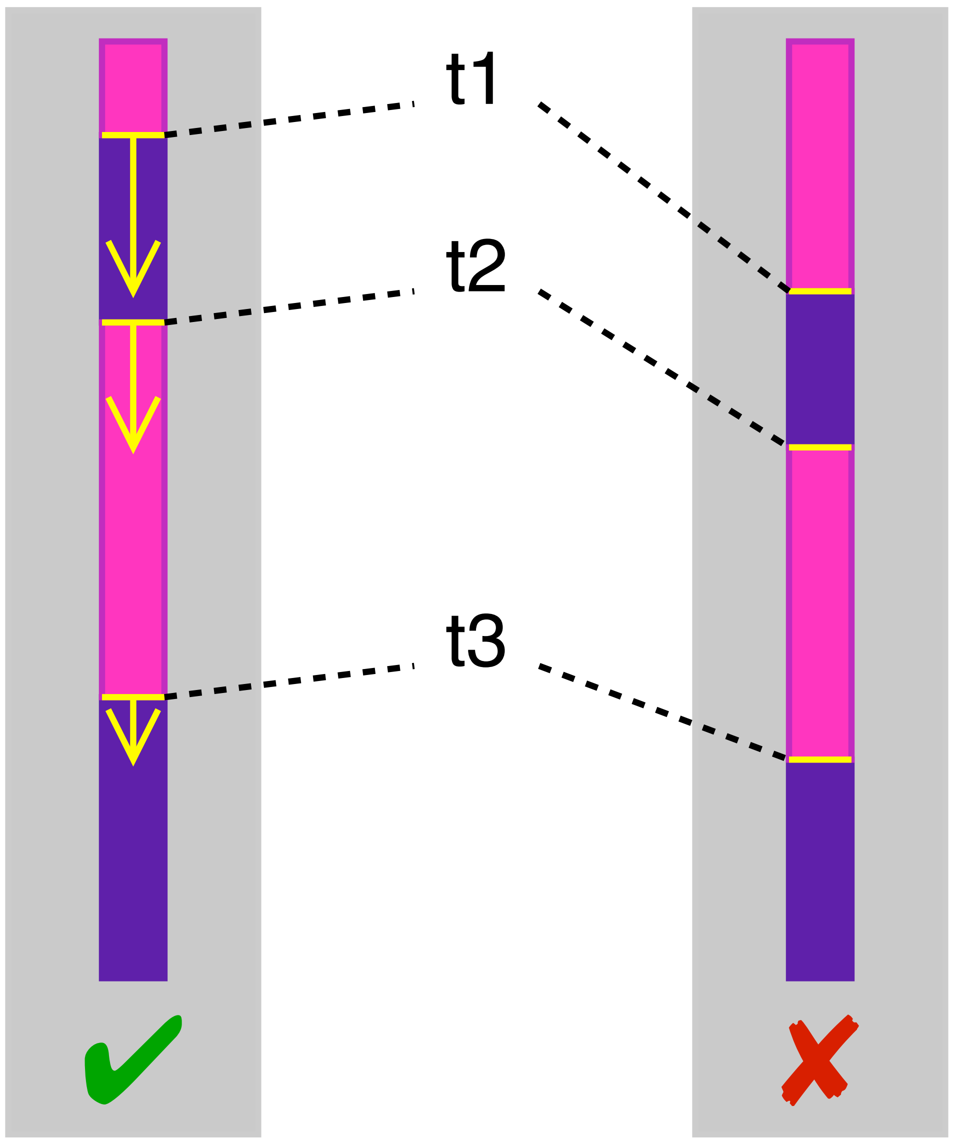
Schedules as Input





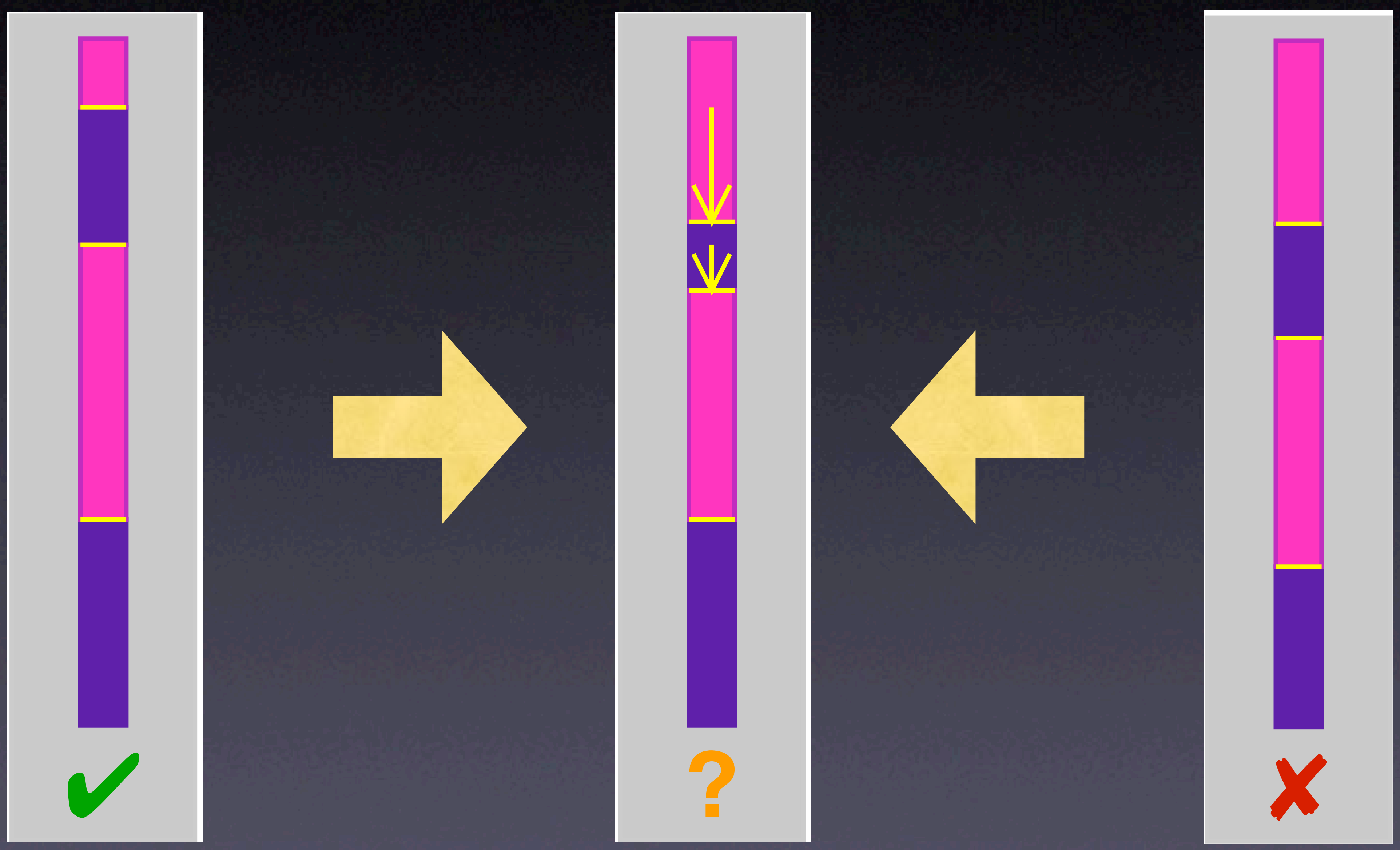
The schedule difference causes the failure!

Finding Differences

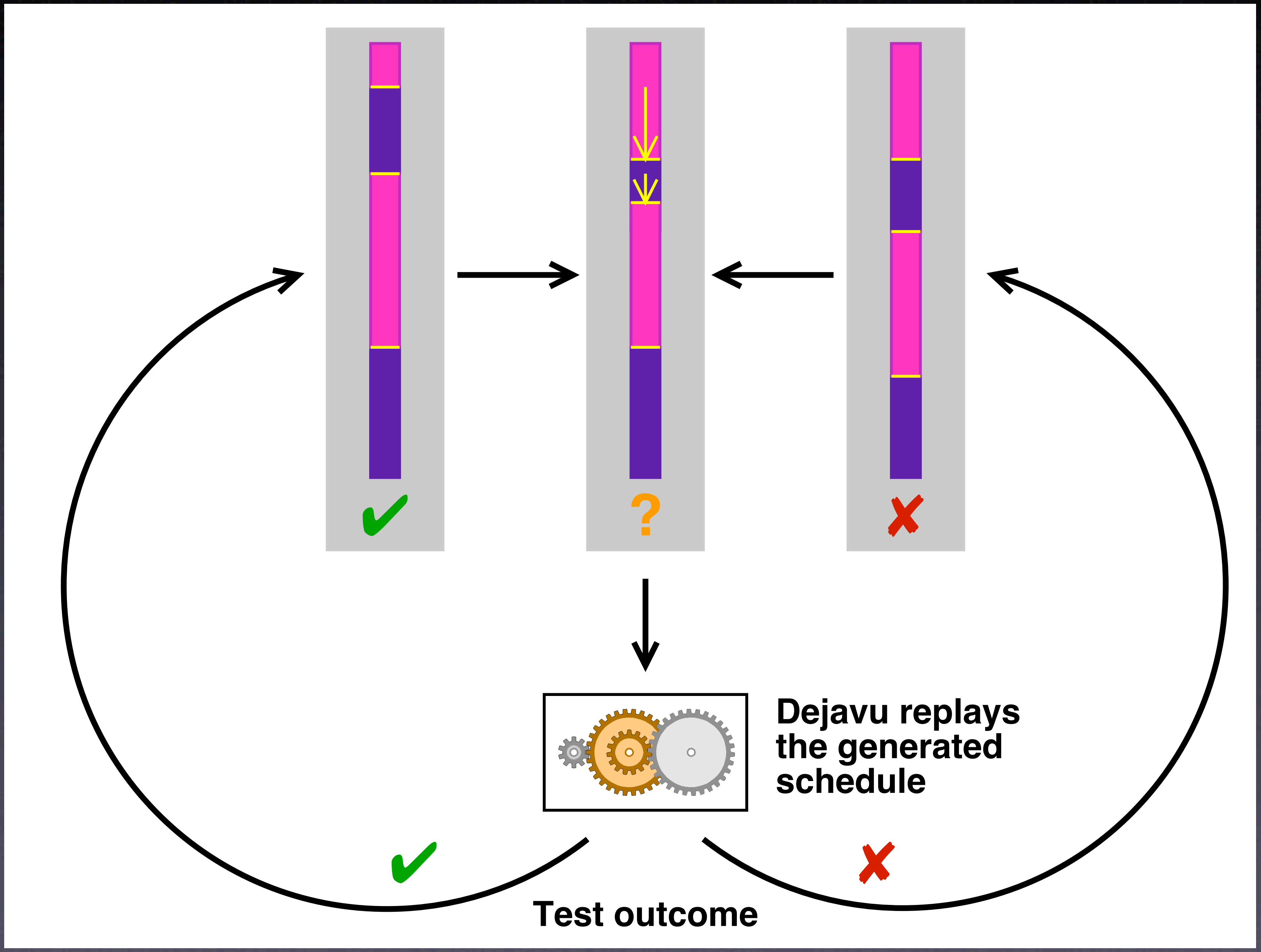


- We start with runs ✔ and ✘
- We determine the differences Δ_i between thread switches t_i:
  - t_1 occurs in ✔ at “time” 254
  - t_1 occurs in ✘ at “time” 278
  - The difference $\Delta_1 = |278 - 254|$ induces a *statement interval:* the code executed between “time” 254 and 278
  - Same applies to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

Isolating Differences



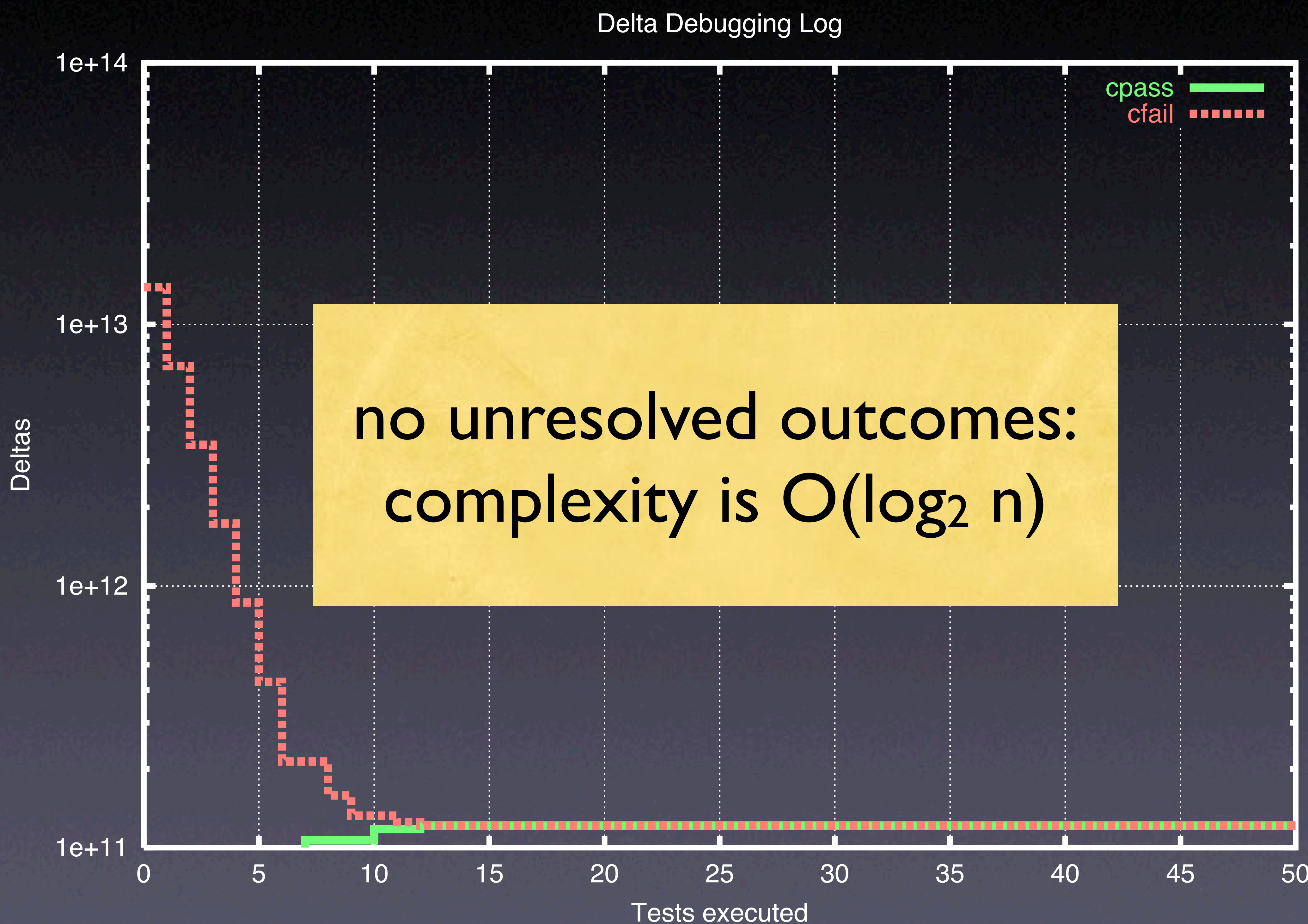
Isolating Differences



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.");
        …
    }
    …
}
```

General Issues

- How do we choose the *alternate world*?
- How do we *decompose* the configuration?
- How do we know *a* failure is *the* failure?
- How do we disambiguate *multiple causes?*
- How do I get to the defect?

Concepts

- To isolate failure causes automatically, use
  - an *automated test case*
  - a means to *narrow down the difference*
  - a *strategy* for proceding.
- One possible strategy is Delta Debugging.

Concepts (2)

- Delta Debugging can isolate failure causes
  - in the (general) *input*
  - in the *version history*
  - in *thread schedules*
- Every such cause implies a *fix* – but not necessarily a correction.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit

http://creativecommons.org/licenses/by/1.0

or send a letter to Creative Commons, 559 Abbott Way, Stanford, California 94305, USA.