Automatic Testing of Interactive JavaScript Debuggers

Daniel Lehmann, mail@dlehmann.eu
TU Darmstadt, Germany

Advisor: Michael Pradel

TECHNISCHE UNIVERSITÄT DARMSTADT

SOLA SoftwareLab

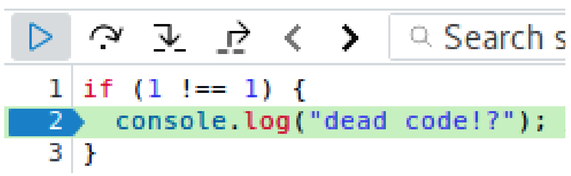
Abstract

- Debuggers are **crucial** for finding bugs, yet sometimes **buggy themselves**.
- Our approach **generates debugger actions** (breakpoints, steps, …) as input and **compares the behavior** of **different debuggers** against each other. Our testing is **interactive** as are the debuggers.
- We found **17 bugs** in debuggers of Firefox and Chromium; 6 were already fixed by the developers.

We also found differences between debuggers due to **underspecified behavior**.

1 Motivation

Debuggers are an **essential tool** for development.
Debugger bugs are **confusing** and can be even **harmful**:



Firefox Debugger seems to pause at dead code.

- See **bug caused by debugger** and not actual program.
- Developer could **introduce "wrong fixes"**.
- **Hard time to find real bugs**, e.g., when breakpoint is never hit.

Automatic testing has found hundreds of bugs in compilers [McKeeman 1998, Yang et al. PLDI 2011, Le et al. PLDI 2014].
Can we apply its ideas to debuggers?

- Generate test inputs + oracle whether test passed.
- **Differential testing**: use another implementation as oracle.

3 Example

Program-to-Debug-A.js

```
1 for (let i = 0; i < 3; i++) {
2     console.log(i);
3 }
```

Program-to-Debug-B.js

```
1 var x = 3;
2 for (var j = 7; j < x; j++) {
3     // do something with j
4     // ...
5 }
```

Differential Testing Phase

Trace A

```
Set breakpoint at line 1
  Chrome:  set at line 1
  Firefox: set at line 1
Start execution
  Both paused at line 1
  Call stack: ...
  Variables: ...
Step over
  Chrome:  paused at line 2
  Firefox: paused at line 3
Pause location difference
```

≈

Trace B

```
… (actions and outputs)
Both paused at line 2
Step over
  Chrome:  paused at line 3
  Firefox: paused at line 5
Pause location difference
```

Similar difference in Trace A and B:

- Pause location different.
- Last action is step over.
- Beforehand, both paused at a `ForStatement`.

→ Traces A and B are grouped together.

4 Results

41 programs-to-debug (26 from SunSpider) × 30 seeds = **1230 test runs**. **Differences in 79.7%** runs, most from breakpoints.

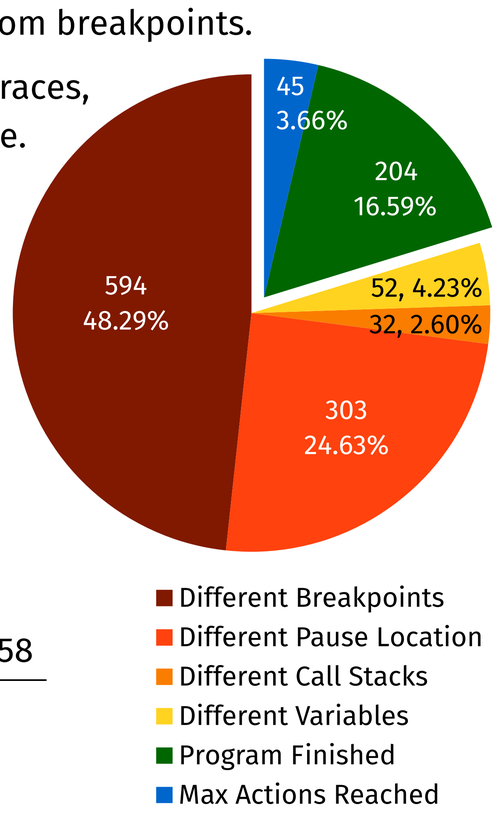
79 groups. Largest group contains 325 traces, all setting a breakpoint at an empty line.

Many differences are not clear bugs, but still confusing when switching debuggers. → Debugger specification?

Whole new area for exploration:

- Test more features: conditional breakpoints, mutating variables.
- Other Debuggers: GDB, LLDB.
- Minimize test cases automatically.

Bugs	Firefox 54	Chromium 58
Reported	13	4
Of those, fixed	3	3



2 Approach

Phase 1: Differential Testing

Unlike compilers, debuggers are interactive, so testing has to be as well:

- Inputs are not just programs, but also **debugger actions**:
  `Set breakpoint at line x,`
  `Start execution,`
  `Resume, Step in/out/over.`
- Next possible actions depend on previous **debugger outputs**:
  `Breakpoint at line x,`
  `Paused at line x,`
  `Variables, Call stack.`
- Stop action generation when execution finished or when debuggers diverge.
  (It is not useful to compare more behavior when the debuggers are, e.g., already paused at different locations.)

We can record **many traces** (= debugger actions and outputs) for one program-to-debug by choosing **different seeds** during action generation.

Phase 2: Group Differences

We obtain many traces with differences:

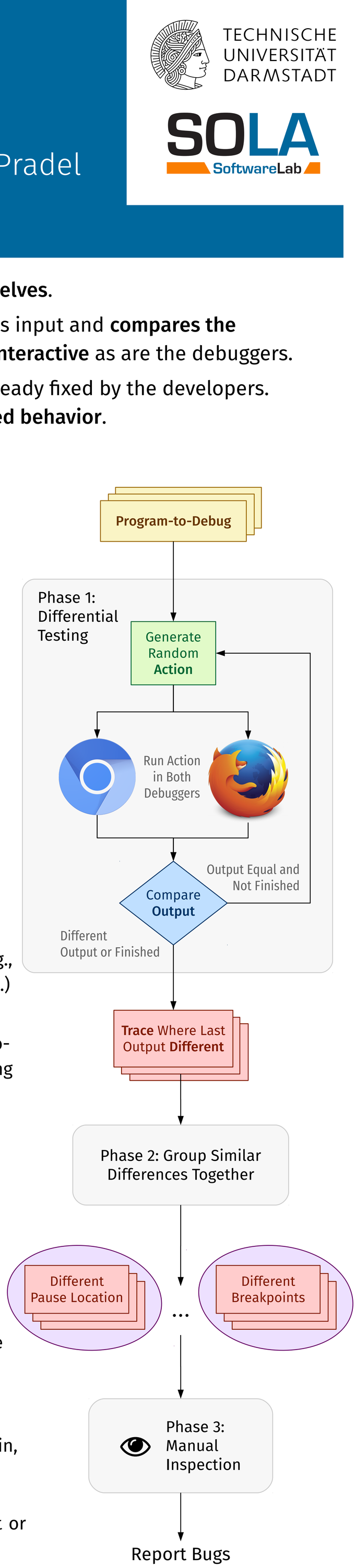
- **Too many** to inspect all manually.
- We observe many **similar, recurring differences** between debuggers.

Automatically group traces with similar differences together, based on:

- **Last debugger output**, e.g., difference in breakpoint location, or in pause location, or different variables.
- **Last debugger action**, e.g., did the debuggers perform a resume, a step in, or a step over?
- **AST node of last pause**, e.g., did the debuggers pause at a `ForStatement` or `VariableDecl` before the action?

Phase 3: Manual Inspection

Inspect ≥ 1 traces from each group and build minimal test case by hand.



Code & Results: `https://github.com/danleh/automatic-debugger-testing`