Multiverse Debugging: Non-Deterministic Debugging for Non-Deterministic Programs (Artifact)

Robbert Gurdeep Singh
Universiteit Gent, Belgium
Robbert.GurdeepSingh@ugent.be

Carmen Torres Lopez
Vrije Universiteit Brussel, Belgium
ctorresl@vub.be

Stefan Marr
School of Computing University of Kent, United Kingdom
s.marr@kent.ac.uk

Elisa Gonzalez Boix
Vrije Universiteit Brussel, Belgium
egonzale@vub.be

Christophe Scholliers
Universiteit Gent, Belgium
Christophe.Scholliers@ugent.be

Abstract

Many of today's software systems are parallel or concurrent. With the rise of Node.js and more generally event-loop architectures, many systems need to handle concurrency. However, their non-deterministic behavior makes it hard to debug. Today's interactive debuggers unfortunately do not support developers in debugging non-deterministic issues. They only allow exploring a single execution path. Therefore, some bugs may never be reproduced in the debugging session, because the conditions to trigger are not reached. As a solution, we propose multiverse debugging, a new approach for debugging non-deterministic programs that allow developers to observe all possible execution paths of a parallel program and debug it interactively. We introduce the concepts of multiverse breakpoints and stepping, which can halt a program in different execution paths, i.e. universes. We apply multiverse debugging to AmbientTalk, an actor-based language, resulting in Voyager, a proof of concept multiverse debugger that takes as input Featherweight AmbientTalk programs written in PLT-Redex, and allows programmers to interactively browse all possible execution states by means of multiverse breakpoints and stepping commands. We provide a proof of non-interference, i.e we prove that observing the behavior of a program by the debugger does not affect the behavior of that program and vice versa. Multiverse debugging establishes the foundation for debugging non-deterministic programs interactively, which we believe can aid the development of parallel and concurrent systems.

2012 ACM Subject Classification Software and its engineering → Concurrent programming languages; Software and its engineering → Software testing and debugging

Keywords and phrases Debugging, Concurrency, Actors, Formal Semantics

Digital Object Identifier 10.4230/DARTS.5.2.4

Funding *Robbert Gurdeep Singh*: Doctoral fellowship from the Special Research Fund (BOF) of Ghent University (reference number: BOF18/DOC/327)

Carmen Torres Lopez: FWO Research Foundation Flanders (FWO) grant, project number G004816N.

Acknowledgements We would like to thank Thomas Dupriez (ENS Paris-Saclay - RMoD, Inria, Lille-Nord Europe) for an initial implementation of the underlying visualization and reduction code.

Related Article Carmen Torres Lopez, Robbert Gurdeep Singh, Stefan Marr, Elisa Gonzalez Boix, and Christophe Scholliers, "Multiverse Debugging: Non-Deterministic Debugging for Non-Deterministic Programs", in 33rd European Conference on Object-Oriented Programming (ECOOP 2019), LIPIcs, Vol. 134, pp. 27:1–27:30, 2019. https://dx.doi.org/10.4230/LIPIcs.ECOOP.2019.27

Related Conference 33rd European Conference on Object-Oriented Programming (ECOOP 2019), July 15–19, 2019, London, United Kingdom

1 Scope

This artifact accompanies the article "Multiverse Debugging: Non-deterministic Debugging for Non-deterministic Programs". In the article we argue that parallelism has become an integral part of modern software ranging from large-scale server code to responsive web applications and networked embedded systems. While a wide range of high-level concurrency abstractions are available, understanding and debugging parallel programs remains challenging. The main reason why parallel programs are so difficult to debug is due to their non-determinism. The state of a parallel program at any given moment in time can alter to one of *many* possible successor states. As a consequence, it is very difficult to reason about their behavior and to reproduce bugs as they may only manifest in rare execution traces.

We propose *multiverse debugging*, a novel debugging technique for parallel programs which combines online breakpoint-based debugging with state exploration from static techniques. The key idea of multiverse debugging is that *non-deterministic programs require non-deterministic debugging*. Contrary to current state-of-the-art debuggers, which only execute the program in one execution path (i.e. one *universe*), a multiverse debugger can observe all possible universes. A multiverse debugger is itself a non-deterministic program which is able to explore all possible states of a parallel program while leveraging breakpoints and stepping commands of online debuggers to interactively search for the root cause of a bug. This means that regular breakpoints become *multiverse breakpoints* which are potentially triggered multiple times in different universes. As such, a multiverse debugger ensures that if a bug is in the program, it will be observed during the debugging session.

The main contributions of the article are:

1. The definition of multiverse debugging
2. A semantics for non-deterministic debugger and proof of non-interference
3. An implementation of applying multiverse debugging to an actor-based language called Voyager, a tool to interact with AmbientTalk programs written in PLT-Redex.

The first contribution is a conceptual contribution and thus it is not backed by the artifact. This artifact contains:

1. Voyager, a proof-of-concept tool which applies multiverse debugging over AmbientTalk programs written in PLT-Redex (contribution 3).
2. The semantics for a non-deterministic debugger in PLT-Redex, i.e. the Voyager semantics (part of contribution 2). The proof of non-interference has not been mechanized, and it is only shown in the companion article.

This artifact's main focus is on explaining Voyager, a proof-of-concept multiverse debugger for AmbientTalk programs. The goal of this tool is to give developers a first impression of what it would feel like to interactively debug a non-deterministic program with a multiverse debugger.

2 Content

In this artifact, we provide:

- A proof-of-concept multiverse debugging tool called *Voyager* packaged as an Open Virtualization Format archive for Virtual Box (`VoyagerECOOP2019.ova`). The tool's user interface is automatically started in a web-browser once the Virtual Machine has started.
- The debugging semantics for the AmbientTalk language: "`AmbientTalk-Debugger`" stored in the virtual machine at `/home/osboxes/demo/DebuggerPaperExample.zip`.
- Documentation on how to reproduce the examples of the research article: Opened in a second tab of the web-browser.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS).

4 Tested platforms

The `ova`-file has been tested in Virtual Box versions 5.2.22 and 6.0.4. For best performance, increase the assigned amount of CPU power, RAM and Video Memory.

5 License

GNU Lesser General Public License v3.0 or later

6 MD5 sum of the artifact

a11ad9a590566bf6f5af3393aecd80f8

7 Size of the artifact

3.61 GiB