LoRe: A Programming Model for Verifiably Safe Local-First Software (Artifact)

Julian Haas
Technische Universität Darmstadt, Germany

Ragnar Mogk
Technische Universität Darmstadt, Germany

Elena Yanakieva
University of Kaiserslautern-Landau, Germany

Annette Bieniusa
University of Kaiserslautern-Landau, Germany

Mira Mezini
Technische Universität Darmstadt, Germany

Abstract

Local-first software manages and processes private data locally while still enabling collaboration between multiple parties connected via partially unreliable networks. Such software typically involves interactions with users and the execution environment (the outside world). The unpredictability of such interactions paired with their decentralized nature make reasoning about the correctness of local-first software a challenging endeavor. Yet, existing solutions to develop local-first software do not provide support for automated safety guarantees and instead expect developers to reason about concurrent interactions in an environment with unreliable network conditions.

We propose *LoRe*, a programming model and compiler that automatically verifies developer-supplied safety properties for local-first applications. *LoRe* combines the declarative data flow of reactive programming with static analysis and verification techniques to precisely determine concurrent interactions that violate safety invariants and to selectively employ strong consistency through coordination where required. We propose a formalized proof principle and demonstrate how to automate the process in a prototype implementation that outputs verified executable code. Our evaluation shows that *LoRe* simplifies the development of safe local-first software when compared to state-of-the-art approaches and that verification times are acceptable.

2012 ACM Subject Classification Software and its engineering → Formal software verification; Software and its engineering → Distributed programming languages; Software and its engineering → Data flow languages; Software and its engineering → Consistency; Theory of computation → Pre- and post-conditions; Theory of computation → Program specifications; Computer systems organization → Peer-to-peer architectures

Keywords and phrases Local-First Software, Reactive Programming, Invariants, Consistency, Automated Verification

Digital Object Identifier 10.4230/DARTS.9.2.11

Funding This work was funded by the German Federal Ministry of Education and Research together with the Hessen State Ministry for Higher Education (ATHENE), the German Research Foundation (SFB 1053), and the German Federal Ministry for Economic Affairs and Climate Action project SafeFBDC (01MK21002K).

Related Article Julian Haas, Ragnar Mogk, Elena Yanakieva, Annette Bieniusa, and Mira Mezini, "LoRe: A Programming Model for Verifiably Safe Local-First Software", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 12:1–12:15, 2023.
`https://doi.org/10.4230/LIPIcs.ECOOP.2023.12`

Related Conference 37th European Conference on Object-Oriented Programming (ECOOP 2023), July 17–21, 2023, Seattle, Washington, United States

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2023 Call for Artifacts and the ACM Artifact Review and Badging Policy.

Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Scope

The goal of this artifact is to evaluate the performance of the verification process used by the LoRe compiler. In particular, it allows the reproduction of the performance benchmarks presented in Section 4 of the related paper.

2 Content

The artifact package includes:

- **readme.md:** A readme describing how to use the artifact (written in markdown).
- **lore.docker.tar:** An executable docker image of LoRe's verification backend that allows compiling LoRe programs to Viper intermediate language.
- **viper.docker.tar:** An executable docker image that contains the Viper[1] verifier and utilities for benchmarking.
- **examples:** The LoRe source code of two example applications: The calendar application from the paper and the TPC-C benchmark.
- **sources:** The source code of LoRe's verification backend (written in Scala).

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). In addition, the source code of the LoRe compiler is available at `https://github.com/stg-tud/LoRe`.

4 Tested platforms

Software: The artifact includes two executables packaged as x86 Docker images. Therefore, it requires a container runtime such as Docker[2] or Podman[3] that is capable of running docker images. When run on a different platform than x86 (such as an Apple Silicon Mac), one likely has to specify the platform as in `docker run --platform linux/amd64`. Depending on the local Docker installation, one may need to prefix the described docker commands with `sudo`.
Hardware: The artifact is supposed to run on normal consumer hardware such as laptops and desktop computers. The verification process relies on the Z3 SMT solver which requires a certain level of computing power and memory. We therefore recommend a machine with at least 16GB of memory.

5 License

6 MD5 sum of the artifact

e27898622113ac3de4936bafda3151fa

7 Size of the artifact

1.43 GiB

[1] `https://www.pm.inf.ethz.ch/research/viper.html`
[2] `https://www.docker.com/`
[3] `https://podman.io/`