super-Charging Object-Oriented Programming Through Precise Typing of Open Recursion (Artifact)

Andong Fan
The Hong Kong University of Science and Technology (HKUST), Hong Kong, China

Lionel Parreaux
The Hong Kong University of Science and Technology (HKUST), Hong Kong, China

Abstract

This artifact consists of an SBT project with a Scala implementation of the MLscript programming language extended with "super-charged" object-oriented programming features (SuperOOP), introduced in the corresponding paper. We provide a test suite that includes SuperOOP examples and a web demo that gives live typing and running results of the user input source.

2012 ACM Subject Classification Software and its engineering → Object oriented languages

Keywords and phrases Object-Oriented Programming, the Expression Problem, Open Recursion

Digital Object Identifier 10.4230/DARTS.9.2.22

Acknowledgements We thank the anonymous reviewers for their helpful comments as well as Cunyuan Gao for his help with the implementation.

Related Article Andong Fan and Lionel Parreaux, "super-Charging Object-Oriented Programming Through Precise Typing of Open Recursion", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 11:1–11:28, 2023.
`https://doi.org/10.4230/LIPIcs.ECOOP.2023.11`

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.

1 Scope

In the corresponding paper, we refer to our implementation of SuperOOP as part of the MLscript programming language. This artifact contains that implementation. MLscript code examples in the paper should be *functional* as they are shown. Our implementation shows the desired typing and running results of those examples. Specifically, under `shared/src/test/diff/ecoop23`:

- `Intro.mls` contains the example in paper Section 1
- `ExpressionProblem.mls` contains the motivating paper example in paper Section 2
- `PolymorphicVariants.mls` contains a modular evaluator of extended lambda calculus case study [1] in the appendix of the paper
- `SimpleRegionDSL.mls` contains a simple "regions" DSL case study [3] in the appendix of the paper

Moreover, we explain how the MLscript compiler codebase is organized and introduce the compiler implementation in the codebase documentation. Of particular interest, we discuss how class and mixin typing is implemented and how corresponding JavaScript code is generated in that document. We hope this could inspire the *reuse* of the artifact to extend the MLscript compiler with new features.


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

2 Content

The artifact package consists of a Docker [2] image (`linux/amd64`) and a `README.md` which includes the instructions to load the image and test our artifact. Sources inside the Docker image with all necessary dependencies installed are as follows:

- `mls-codebase-doc.md`: the codebase documentation
- `shared/src/main/scala/mlscript` directory: the sources of the MLscript compiler
- `shared/src/test/scala/mlscript` directory: testing infrastructure
- `shared/src/test/diff` directory: MLscript tests

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 implementation is open-source and available at `https://github.com/hkust-taco/superoop`. The Docker image is pushed to Docker Hub at `https://hub.docker.com/r/superoop/superoop-docker/`. The web demo is also online and available at `https://hkust-taco.github.io/superoop/`.

4 Tested platforms

Any system with Docker available can access, compile, and test our artifact. The Docker image archived in this artifact is for the `amd64` platform, which contains all dependencies to compile and run our artifact. We tested that all 428 MLscript tests pass in 15 seconds on Intel® Core™ i7-13700KF and AMD Ryzen™ 9 5900X processors. We have also prepared an image for the `arm64` platform and pushed that to our Docker Hub repository. On a MacBook Pro (2021) with Apple M1 Max processor, all the tests pass in 15 seconds.

To test our artifact from scratch, one needs to install a recent Java Virtual Machine (JVM), SBT, and NodeJS. We explicitly support NodeJS versions `16.14` to `16.17`, `17`, `18`, and `19`.

5 License

The artifact is available under the MIT License.

6 MD5 sum of the artifact

41a2dcbd8e090b15582145f2c9a42fad

7 Size of the artifact

686.2 MB

References

1 Jacques Garrigue. Code reuse through polymorphic variants. In *In Workshop on Foundations of Software Engineering*, 2000. URL: `https://www.math.nagoya-u.ac.jp/~garrigue/papers/variant-reuse.pdf`.

2 Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

3 Yaozhu Sun, Utkarsh Dhandhania, and Bruno C. d. S. Oliveira. Compositional embeddings of domain-specific languages. *Proc. ACM Program. Lang.*, 6(OOPSLA2), October 2022. `doi:10.1145/3563294`.