What If We Don't Pop the Stack? The Return of 2nd-Class Values (Artifact)

Anxhelo Xhebraj
Purdue University, West Lafayette, IN, USA

Oliver Bračevac
Purdue University, West Lafayette, IN, USA

Guannan Wei
Purdue University, West Lafayette, IN, USA

Tiark Rompf
Purdue University, West Lafayette, IN, USA

Abstract

The main paper presents $\lambda^{1/2}_{\hookleftarrow}$, a type system and operational semantics with 2nd-class values and delayed stack reclamation. This artifact contains a compiler implementation of the calculus in Scala Native, the code for the case studies shown in the paper, and code for reproducing the evaluation.

2012 ACM Subject Classification Software and its engineering → General programming languages

Keywords and phrases Call stack, closures, stack allocation, memory management, 2nd-class values, capabilities, effects

Digital Object Identifier 10.4230/DARTS.8.2.26

Related Article Anxhelo Xhebraj, Oliver Bračevac, Guannan Wei, and Tiark Rompf, "What If We Don't Pop the Stack? The Return of 2nd-Class Values", in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 15:1–15:29, 2022.
https://doi.org/10.4230/LIPIcs.ECOOP.2022.15

Related Conference 36th European Conference on Object-Oriented Programming (ECOOP 2022), June 6–10, 2022, Berlin, Germany

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2022 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

This artifact contains the implementation of the system presented in the accompanying paper. Through this artifact it is possible to reproduce the tables and figures of the performance evaluation in Section 7 of the paper. The artifact is made available as a docker image that contains all the necessary dependencies to build and run the implementation artifacts.

2 Content

The artifact package includes:

- An implementation of the type system and execution model presented in the paper. The implementation is an extension of the Scala Native compiler [1, 2].
- Benchmark programs demonstrating uses of the type system and performance improvements from Section 7.


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

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 artifact is also available at: `https://github.com/angelogeb/scala-native`.

4 Tested platforms

The artifact was tested on Ubuntu 20.04.4 LTS with processor Intel® Core™ i5-6300HQ CPU @ 2.30GHz, Java 8 and LLVM 10. Compilation of the artifacts requires at least 8GB of RAM.

5 License

The artifact is available under the Apache 2.0 license.

6 MD5 sum of the artifact

40a9fa2598abd8969a8c074ec0350a39

7 Size of the artifact

1.1 GiB

8 Docker Image

To load the downloaded image run the following command from the terminal:

```
docker load -i local-scala-native.tar.gz
```

To run the image interactively, run:

```
docker run --name ecoop_container --rm -it local-scala-native bash
```

The docker image contains documentation on how to navigate the files and directories of the artifact. To explore the repository inside the container we suggest to use Visual Studio Code[1] with the Remote Containers extension. [2] [3]

References

1 Scala Native Contributors. Scala Native. URL: `https://github.com/scala-native/scala-native`.

2 Denys Shabalin. *Just-in-time performance without warm-up*. PhD thesis, EPFL, Lausanne, 2020. `doi:10.5075/epfl-thesis-9768`.

[1] `https://code.visualstudio.com/`
[2] `https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers`
[3] `https://code.visualstudio.com/docs/remote/containers`