Direct Foundations for Compositional Programming (Artifact)

Andong Fan[1]
Zhejiang University, Hangzhou, China

Xuejing Huang[1]
The University of Hong Kong, China

Han Xu
Peking University, Beijing, China

Yaozhu Sun
The University of Hong Kong, China

Bruno C. d. S. Oliveira
The University of Hong Kong, China

Abstract

Our companion paper proposes a new formulation of the F_i^+ calculus with disjoint polymorphism and a merge operator based on Type-Directed Operational Semantics. The artifact contains Coq formalization of the F_i^+ calculus and our new implementation of the CP language, which demonstrates the new F_i^+ can serve as the direct foundation for Compositional Programming.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases Intersection types, disjoint polymorphism, operational semantics

Digital Object Identifier 10.4230/DARTS.8.2.4

Funding This research was funded by the University of Hong Kong and Hong Kong Research Grants Council projects number 17209519, 17209520 and 17209821.

Acknowledgements We thank the anonymous reviewers for their helpful comments.

Related Article Andong Fan, Xuejing Huang, Han Xu, Yaozhu Sun, and Bruno C. d. S. Oliveira, "Direct Foundations for Compositional Programming", in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 18:1–18:28, 2022.
https://doi.org/10.4230/LIPIcs.ECOOP.2022.18

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

We claim in the paper that we formalize the TDOS variant of the F_i^+ calculus, together with its type-soundness and determinism proof in the Coq proof assistant [5]. This artifact contains the corresponding Coq proof scripts. All definitions presented in the paper and lemmas, theorems, and corollaries with their proofs can be found.

In addition, this Coq proof artifact is also reusable: any extension or changes to the system can be easily done with our provided Ott [8] specification, and after possible adjustments, all the proofs can be reused for the formalization of the extended or altered system.

[1] The first two authors contributed equally to this work.

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

Furthermore, our new implementation of the CP language is a self-contained interpreter. It demonstrates that F_i^+ can serve as the direct foundation of Compositional Programming. It can be regarded as a reference implementation of our algorithmic rules. Not only can readers try CP programs, but they can also reuse our code as part of their own research or extend our implementation with new features.

The formalization of F_i^+ is defined based on the locally nameless representation with cofinite quantification [3]. We rely on the Penn's metatheory library (metalib) [1] and use the `LibTactics.v` from the TLC Coq library [4]. The Ott tool generates the Coq definitions (`syntax_ott.v`) and the LNgen [2] tool generates some infrastructure code (`rules_inf.v`). The proof structure and strategy are inspired by the formalization of the λ_i^+ calculus [6], which is also based on TDOS.

2 Content

The artifact package includes:

- a Docker [7] image which contains the following code with environment set up
- `README.md`: the instructions of compliling and testing our artifact, together with the correspondence between the paper lemmas, theorems, corollaries and Coq proofs
- `paper.pdf`: the companion paper (extended version) with appendices
- `fiplus/coq` directory: the Coq formalization and proofs of F_i^+
- `fiplus/spec` directory: the Ott specification of F_i^+
- `CP-next/` directory: a PureScript implementation of the CP language built on top of F_i^+

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 Coq formalization within the artifact is also available at `https://github.com/andongfan/CP-Foundations`. An online demo of our CP implementation is available at `https://plground.org`.

4 Tested platforms

Any system with Docker available can access, compile, and test our artifact. We prepared a configured Docker image containing all code and packages to compile and run our artifact.

To build the proofs from scratch, one needs to install Coq, Ott, metalib, and LNgen. LNgen additionally relies on GHC [9]. To run the CP implementation outside the container, one needs to install Node.JS. The detailed instructions can be found inside the `README`.

5 License

The artifact is available under the GNU General Public License v3.0.

6 MD5 sum of the artifact

4afe79ce3b2bbb80e8f9d0e419d22922

7 Size of the artifact

1.46 GiB

References

1 Brian Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. Engineering formal metatheory. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pages 3–15, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1328438.1328443.

2 Brian Aydemir and Stephanie Weirich. Lngen: Tool support for locally nameless representations. Technical Report MS-CIS-10-24, Department of Computer and Information Science, University of Pennsylvania, June 2010. URL: https://repository.upenn.edu/cis_reports/933/.

3 Arthur Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, 49(3):363–408, 2012.

4 Arthur Charguéraud and François Pottier. Tlc: a non-constructive library for coq. https://www.chargueraud.org/softs/tlc/.

5 The Coq Development Team. *The Coq Reference Manual, version 8.11.1*, April 2020. Available electronically at https://coq.inria.fr/distrib/current/refman/.

6 XUEJING HUANG, JINXU ZHAO, and BRUNO C. D. S. OLIVEIRA. Taming the merge operator. *Journal of Functional Programming*, 31:e28, 2021. doi:10.1017/S0956796821000186.

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

8 Peter Sewell, Francesco Zappa Nardelli, Scott Owens, Gilles Peskine, Thomas Ridge, Susmit Sarkar, et al. Ott: Effective tool support for the working semanticist. *Journal of functional programming*, 20(1):71–122, 2010.

9 GHC Team. Ghc use's guide documentation. https://downloads.haskell.org/~ghc/latest/docs/users_guide.pdf, 2020.