Compositional Programming (Artifact)

Weixin Zhang
University of Bristol, United Kingdom
The University of Hong Kong, China

Yaozhu Sun
The University of Hong Kong, China

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

Abstract

Our main paper presents CP, a Compositional Programming language in a statically typed modular programming style. This artifact includes its Haskell implementation, together with several examples and three case studies written in CP. All code snippets in our main paper can be type-checked and run using our CP interpreter.

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

Keywords and phrases Expression Problem, Compositionality, Traits

Digital Object Identifier 10.4230/DARTS.7.2.11

Funding This work has been sponsored by the Hong Kong Research Grant Council projects number 17210617 and 17209519. The first author conducted this research while at the University of Hong Kong.

Related Article Weixin Zhang, Yaozhu Sun, and Bruno C. d. S. Oliveira, "Compositional Programming", in ACM Transactions on Programming Languages and Systems (TOPLAS), Vol. 43, 2021.
`https://doi.org/10.1145/3460228`

Related Conference 35th European Conference on Object-Oriented Programming (ECOOP 2021), July 12–16, 2021, Aarhus, Denmark (Virtual Conference)

1 Scope

Compositional Programming is a new statically typed modular programming style proposed in our main paper. In Compositional Programming, there is no Expression Problem: it is easy to get extensibility in multiple dimensions. Moreover, it allows us to naturally model attribute-grammar-like programs and generally deal with modular programs with complex dependencies. A concrete language design, CP, is formalized and proved to be type-safe in our main paper (Section 5).

In order to demonstrate the expressiveness and applicability of CP, we conducted three case studies (see Section 6 in our main paper). In these studies, the need for dealing with extensibility and complex dependencies arises. They can help readers better understand the key concepts of Compositional Programming. The case studies we have implemented are listed below:

- *Scans*, a DSL for describing parallel prefix circuits;
- A *mini interpreter* for an expression language;
- *C0 compiler*, which compiles a subset of C to JVM instructions.

In addition to the three case studies, several examples are accompanied in the artifact. The code snippets in Section 3 & 4 of our main paper can also be found here.

2 Content

The artifact package includes:

- `src/` contains the implementation of CP written in Haskell;
- `app/` contains the interpreter driver;
- `test/` contains Hspec testing code;
- `examples/` contains several examples written in CP;
- `case-studies/` contains three case studies (Scans, the mini interpreter and C0 compiler);
- `stack.yaml` and `package.yaml` are configuration files for Haskell Stack;
- `README.md` documents detailed instructions for building, running and testing the artifact.

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 on Docker Hub and can be pulled and run using the following command:

```
docker run -it yzyzsun/cp
```

If you prefer building from scratch, the source code is available at `https://github.com/wxzh/CP`.

4 Tested platforms

The artifact has been tested using LTS Haskell 12.16 (GHC 8.4.4), which is specified in `stack.yaml`.

5 License

The artifact is available under the BSD 3-clause license.

6 MD5 sum of the artifact

b77837955aff63c19767f7bbcd53f075

7 Size of the artifact

573 MiB