Restrictable Variants: A Simple and Practical Alternative to Extensible Variants (Artifact)

Magnus Madsen
Department of Computer Science, Aarhus University, Denmark

Jonathan Lindegaard Starup
Department of Computer Science, Aarhus University, Denmark

Matthew Lutze
Department of Computer Science, Aarhus University, Denmark

Abstract

In this artifact, we provide an implementation of the $\lambda_{\text{var}}^{\text{res}}$ calculus, as described in the related article. The implementation is an extension of the Flix programming language compiler, supporting restrictable variants and the two partial pattern-matching constructs: choose and choose-⋆.

The artifact consists of the extended Flix compiler, a Visual Studio Code extension supporting common IDE features such as syntax highlighting and type hovering, and a collection of Flix files demonstrating the use of restrictable variants. The Flix files correspond to examples presented in the paper.

Users are invited to modify the Flix files in order to observe the influence of their changes on the inferred types in the provided programs.

2012 ACM Subject Classification Theory of computation → Program semantics

Keywords and phrases restrictable variants, extensible variants, refinement types, Boolean unification

Digital Object Identifier 10.4230/DARTS.9.2.12

Related Article Magnus Madsen, Jonathan Lindegaard Starup, and Matthew Lutze, "Restrictable Variants: A Simple and Practical Alternative to Extensible Variants", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 17:1–17:27, 2023. https://doi.org/10.4230/LIPIcs.ECOOP.2023.17

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

The artifact supports claims from Section 4 (Implementation) and Section 6 (Case Studies). It also contains demonstrations of the Color example introduced in Section 2 (Motivation).

Specifically, the artifact contains a Flix compiler in the form of a JAR file, modified to support restrictable variants and the associated features. The modified Flix compiler is compatible with Flix's Visual Studio Code extension, which provides IDE features such as syntax highlighting and type-hovering. The compiler along with the extension allow the user to browse and execute the provided Flix code, observing the types inferred by the implementation of $\lambda_{\text{var}}^{\text{res}}$.

The artifact contains three files of Flix code, each of which covers an example from the paper. The file `src/colors.flix` demonstrates the basic Color variant, introduced in Section 2, Example 1. It includes the isWarm function example from the paper, as well as several other functions on the Color type.

The file `src/boolean-formulas.flix` demonstrates the Exp variant, introduced in Section 2 and further described in Section 6.1. Because multiple functions in the paper share the same name, they are renamed in the Flix code to avoid illegal definition overlap. Examples in Section 2 correspond to function definitions in `src/booleans.flix` as follows:

- Example 2: eval corresponds to the evalNaive definition.
- Example 3: simplify corresponds to the simplifyNaive definition.
- Examples 4–7: map, simplify, run, subst, fasteval, and fastrun are each named as in the paper.

It includes additional utility functions – equality and string conversion – to facilitate user manipulation of the artifact.

The file `src/sequences.flix` demonstrates the Seq variant, a generalization of the Option type, the List type, and the Nel (non-empty list) type, as discussed in Section 6.2. The file contains 89 functions operating on the Seq type. The signatures differ slightly from the description in the paper: Option, List, and Nel are renamed to SOption, SList, and SNel to avoid name collisions with the existing standard library. Furthermore, the sequence is generalized: an extra type parameter a allows the sequence to be polymorphic over its element type, rather than limiting it to Int32. Finally, the file contains a main function, demonstrating the use of the provided Seq functions and the types inferred for expressions using them.

The source code and compiler are available in the artifact under the `demo.zip` archive; the compiler can be run after installation of dependencies as described in the provided `README.md` file. Alternatively, the same compiler and source code, complete with dependencies, is provided in the VirtualBox virtual machine image `variants.ova`.

2 Content

The artifact package includes:

- `LICENSE.md` – A file with the Apache License 2.0 information.
- `README.md` – A file describing the use of the artifact and the differences between the implementation and the formalism.
- `demo.zip` – A zipped Flix project that can be run locally in VSCode after installation of the requirements as described in `README.md`.
  - `README.md` – A copy of the top-level `README.md` file.
  - `flix.jar` – The Flix compiler extended to support restrictable variants.
  - `src/colors.flix` – Flix code of the Color examples of the paper.
  - `src/boolean-formulas.flix` – The case study of Section 6.1 of the paper.
  - `src/sequences.flix` – The case study of Section 6.2 of the paper.
- `variants.ova` – A VirtualBox image with the preinstalled requirements and the project in `demo.zip` ready for use.

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).

4 Tested platforms

There are no special requirements for running the virtual machine, except around 7 GB to store it. To run the artifact locally, VSCode must be installed along with the Flix extension and Java 11 or above. The only hardware requirement to run the artifact locally is at least two CPU-cores.

We have tested the artifact on Windows 10 with 4 cores and 16GB of RAM, and Ubuntu 22.04 with 8 cores and 16GB of RAM. With these two setups we have run the artifact both as a VM with VirtualBox and locally with the three requirements installed.

5 License

The artifact is available under Apache License, Version 2.0.

6 MD5 sum of the artifact

84d987ad32e43484b9ef15cfea8124ff

7 Size of the artifact

6.56 GiB