Disjoint Polymorphism with Intersection and Union Types (Artifact)

Baber Rehman[*]
Huawei Technologies
Hong Kong SAR, China
brehman@connect.hku.hk

Bruno C. d. S. Oliveira
The University of Hong Kong
Hong Kong SAR, China
bruno@cs.hku.hk

Abstract

This artifact contains Coq formalization associated with the paper titled *Disjoint Polymorphism with Intersection and Union Types*. The paper studies a novel disjointness algorithm for deterministic type-based union elimination by exploiting union ordinary and union splittable types. Proposed disjointness algorithm naturally extends for a variant of parametric polymorphism called disjoint polymorphism. This artifact contains Coq formalization of all the metatheory presented in the paper, including proofs of all the theorems and lemmas.

CCS Concepts

• **Theory of computation → Type theory**.

Keywords

Intersection types, Union types, Disjointness, Polymorphism

ACM Reference Format:
Baber Rehman and Bruno C. d. S. Oliveira. 2024. Disjoint Polymorphism with Intersection and Union Types (Artifact). In *Proceedings of the 26th ACM International Workshop on Formal Techniques for Java-like Programs (FTfJP '24), September 20, 2024, Vienna, Austria.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3580438

1 Scope

All of the the metatheory and theorems stated in the paper are available in the artifact. Please follow the steps under Section 3 to download the artifact and look into the directories to verify the claims. Next, we explain the code structure in artifact and its correlation with the paper.

1.1 Code Structure

There are two sub-folders in the artifact in *coq* folder:

(1) section2
(2) section3

Correlation between folders in artifact and the paper is shown in Table 1.

[*]Also with The University of Hong Kong.

FTfJP '24, September 20, 2024, Vienna, Austria
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1111-4/24/09
https://doi.org/10.1145/3580438

Table 1: Correlation of folders in artifact and paper.

Folder	System	Reference in paper
section2	λ_u	Discussed in section 2
section3	Polymorphic λ_u	Discussed in section 3

Table 2: Correlation of section 2 in paper and folder section2 in artifact.

Lemma in Paper	Coq file	Lemma(s) in Coq File
Definition 1.1	syntax.v	DisjSpec
Lemma 2.1	syntax.v	UO_or_US
Lemma 2.2	syntax.v	Disj_sound
Lemma 2.3	syntax.v	Disj_Complete_size
Lemma 2.4	syntax.v	sub_refl
Lemma 2.5	syntax.v	sub_transitivity
Theorem 2.6	typing.v	preservation
Theorem 2.7	typing.v	progress
Theorem 2.8	typing.v	determinism

1.1.1 Folder section2.

- *syntax.v* contains syntax and disjointness properties of λ_u with intersection types.
- *typing.v* contains semantics and properties related to type-safety and determinism.

Correlation of important lemmas in artifact and in the paper is shown in Table 2.

1.1.2 Folder section3.

- *syntax.v* contains syntax of the λ_u with intersection types, nominal types and disjoint polymorphism.
- *typing.v* contains semantics and properties related to disjointness, type-safety and determinism.

Correlation of important lemmas in artifact and in the paper is shown in Table 3.

2 Contents

The artifact package includes:

Table 3: Correlation of section 3 in paper and folder section3 in artifact.

Lemma in Paper	Coq file	Lemma(s) in Coq File
Lemma 3.1	typing.v	sub_disjoint3
Lemma 3.2	typing.v	sub_refl
Lemma 3.3	typing.v	sub_transitivity
Theorem 3.4	typing.v	preservation
Theorem 3.5	typing.v	progress
Theorem 3.6	typing.v	determinism
Lemma 3.7	typing.v	typing_through_subst_te
Lemma 3.8	typing.v	typing_narrowing
Lemma 3.9	typing.v	typing_weakening
Lemma 3.10	typing.v	UO_sub_union
Lemma 3.11	typing.v	disj_sym
Lemma 3.12	typing.v	val_check_disjoint_types

- Coq formalization of the calculi discussed in paper
- Docker file
- README file
- A copy of the publication

3 Getting the Artifact

The artifact is available at:
https://github.com/baberrehman/FTfJP2024-artifact.

We provide two alternatives to download and run the artifact:

(1) Docker Image
(2) Build from Scratch

3.1 Docker Image

This is the easiest way to run the artifact. We provide a dockerfile in this artifact with all the dependencies installed. You may simply build and run the docker image using the provided dockerfile. Please make sure that the docker is already installed on your system to follow this track.

(1) Clone the repo:
https://github.com/baberrehman/FTfJP2024-artifact
(2) Change directory to the root of the artifact:
`$ cd FTfJP2024-artifact`
(3) Build docker image:
`$ sudo docker build -t ftfjp24disj .` (**Note that . is a part of command**)
(4) Run the docker container with an interactive shell:
`$ sudo docker run -it ftfjp24disj`
(5) This will open the home directory of the artifact's docker container
(6) Change directory to coq formalization inside the docker container:
`$ cd FTfJP2024-artifact/coq/`
(7) Build the artifact by running make command: `make` in *section2* and *section3* folders
(8) *vim* is pre-installed in the docker container to look into the files

Running `make` command in *section2* and *section3* folders:

section2.

- `$ cd section2`
- `$ make`

section3.

- `$ cd section3`
- `$ make`

Please feel free to go through the code in each section. *vim* and *cat* commands are available to look into the files. Recommended way to look into the files is by downloading the code archive from the given git repo. This completes the evaluation of artifact following docker image.

3.2 Build from Scratch

This section explains how to build the artifact from scratch.

3.2.1 Prerequisites. We tested all the Coq files using Coq version 8.13.2. Please use same version for the sake of consistency. We recommend installing Coq using the opam package installer.

- `$ opam install coq.8.13.2`

Refer to this link for more information and installation steps: https://coq.inria.fr/opam-using.html

3.2.2 Required Libraries. Coq TLC library is required to compile the code. TLC library can also be installed using the opam package installer. Run the following commands one by one to install TLC by opam package installer:

- `$ opam repo add coq-released http://coq.inria.fr/opam/released`
- `$ opam install coq-tlc.20210316`

Please refer to this link for detailed compilation and installation of Coq TLC: https://github.com/charguer/tlc/tree/20210316.

3.2.3 Getting the files. Use the following command to clone the git repo:

- `$ git clone https://github.com/baberrehman/FTfJP2024-artifact.git`

You should be able to see all the Coq files inside folder *coq* after cloning the repo. There are 2 sub-folders in the *coq* folder, with make file in each:

(1) section2 ⟶ Discussed in section 2 in paper
(2) section3 ⟶ Discussed in section 3 in paper

Note: Please make sure to run **eval $(opam env)** if Coq is installed using opam. This command can be skipped otherwise.

Open the terminal in each folder and run make:

3.2.4 Folder section2.

- `$ eval $(opam env) (optional)`
- `$ make`

Similarly, *section3* can be compiled by opening the terminal in *section3* folder and running the `make` command. This completes the evaluation of artifact following build from scratch.