Type-Directed Operational Semantics for Gradual Typing (Artifact)

Wenjia Ye
The University of Hong Kong, Hong Kong

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

Xuejing Huang
The University of Hong Kong, Hong Kong

Abstract

This artifact includes the Coq formalization associated with the paper **Type-Directed Operational Semantics for Gradual Typing** submitted in ECOOP 2021. The paper illustrates how to employ TDOS on gradually typed languages using two calculi. The first calculus, called λB, is inspired by the semantics of the blame calculus(λB^g) and is sound with λB^g. The second calculus, called λB^r, explores a different design space in the semantics of gradually typed languages. This document explains how to run the Coq formalization. Artifact can either be compiled in the pre-built docker image with all the dependencies installed or it could be built from the scratch. Sections 1-7 explain the basic information about the artifact. Section 7 explains how to get the docker image for the artifact. Section 8 explains the prerequisites and the steps to run coq files from scratch. Section 9 explains coq files briefly. Section 10 shows the correspondence of important lemmas, definitions and pictures discussed in the paper with their respective Coq formalization.

2012 ACM Subject Classification Software and its engineering → Object oriented languages; Theory of computation → Type theory; Software and its engineering → Polymorphism

Keywords and phrases Gradual Typing, Operational Semantics, Type Systems

Digital Object Identifier 10.4230/DARTS.7.2.9

Funding This work has been sponsored by Hong Kong Research Grant Council projects number 17209519 and 17209520.

Related Article Wenjia Ye, Bruno C. d. S. Oliveira, and Xuejing Huang, "Type-Directed Operational Semantics for Gradual Typing", in 35th European Conference on Object-Oriented Programming (ECOOP 2021), LIPIcs, Vol. 194, pp. 12:1–12:30, 2021.
https://doi.org/10.4230/LIPIcs.ECOOP.2021.12

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

1 Scope

This artifact contains the Coq [5] formalization associated with the paper **Type-Directed Operational Semantics for Gradual Typing** submitted in ECOOP 2021. We provide the Coq formalization of λB, λB^g and λB^r systems. These calculus are defined via the locally nameless representation with cofinite quantification [3]. We relies on the Penn's metatheory library [1]. Ott [6] tool and LNgen [2] are used to generate some of the Coq definitions and infrastructure codes. *LibTatics.v* which is from the TLC Coq library [4] is also be used.

2 Content

The artifact package includes:

- Calculus Coq Formalization

Dagstuhl Artifacts Series, Vol. 7, Issue 2, Artifact No. 9, pp. 9:1–9:6
Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Variant Coq Formalization
- Copy of related paper (Type-Directed Operational Semantics for Gradual Typing)
- README file with compilation instructions

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). The artifact is also available at: `https://github.com/YeWenjia/TypedDirectedGradualTyping`.

4 Tested platforms

The artifact has been tested using Coq 8.10.2.

5 MD5 sum of the artifact

d9a38e1c93c960a4901f6aa9cb006062

6 Size of the artifact

2.9 GB

Acknowledgements. We thank the anonymous reviewers for their helpful comments.

7 Docker Image

This section explains how to pull the docker image of artifact from docker hub and use it. Run the following commands one by one in terminal:

1. $ docker pull wenjiaye/ecoop2021
2. $ docker run -it wenjiaye/ecoop2021
3. $ eval $(opam env)

The artifact is located in **/home/coq/coq/** directory.

There are two folders in the artifact, with **make** file in each:

1. **Calculus** → contains λB and λB^g formulation
2. **Variant** → contains λB^r formulation

Go to each folder and run make:

1. $ cd /home/coq/coq/Calculus
2. $ eval $(opam env)
3. $ make

1. $ cd /home/coq/coq/Variant
2. $ eval $(opam env)
3. $ make

8 Build from Scratch

This section explains how to build the artifact from scratch.

8.1 Prerequisites

We tested all the Coq files using Coq version 8.10.2. Please use same version for the sake of consistency. We recommend installing Coq using the opam package installer.

Run the following command to install Coq via opam:

$ opam install coq.8.10.2

Refer to this link for more information and installation steps: `https://coq.inria.fr/opam-using.html`
Or one could download the pre-built packages for Windows and MacOS via `https://github.com/coq/coq/releases/tag/V8.10.2`
Make sure **Coq** is installed (type **coqc** in the terminal, if you see "command not found" this means you have not properly installed Coq)

8.2 Required Libraries

We rely on two Coq libraries: **metalib** (`https://github.com/plclub/metalib`) for the locally nameless representation in our proofs; and **LibTactics.v** (`http://gallium.inria.fr/~fpottier/ssphs/LibTactics.html`), which is included in the directory.

Open the terminal and run the following commands one by one to install **metalib**:

1. $ git clone https://github.com/plclub/metalib
2. $ cd metalib/Metalib
3. $ make install

8.3 Getting the artifact

Use the following commands to clone our git repo. Please note that $ symbol is not a part of command:

$ git clone https://github.com/YeWenjia/TypedDirectedGradualTyping.git
Alternatively you can download the zip file from repo and you should be able to see all the Coq files after unzipping it.

8.4 Proof Structure

There are two folders in the artifact, **docs** and **coq**. Folder **coq** contains all the Coq files. Coq files are further divided into categories with separate folders:

1. **Calculus** → contains λB and λB^g formulation
2. **Variant** → contains λB^r formulation

8.5 Compilation

Please make sure to run the following command before running make if you installed the Coq via opam:

$ eval $(opam env)

Makefiles are available in both **Calculus** and **Variant** folder. Run **make** command individually in each folder to compile.

8.6 Paper

You can also find a copy of our ECOOP2021 paper(Type-Directed Operational Semantics for Gradual Typing) in **docs** folder.

9 Overview of Coq Files

This section explains all the Coq files of λB, λB^g and λB^r systems that we formalized.

9.1 In the `coq/Calculus` directory:

`Calculus` directory contains the definition and proofs of the λB and λB^g calculus.

- `syntax_ott.v`: contains the locally nameless definitions of λB^g.
- `syntaxb_ott.v`: contains the locally nameless definitions of λB.
- `rules_inf.v` and `rulesb_inf.v`: generated from the **lngen** and modified by us.
- `Infrastructure.v`: contains the type systems of the λB^g and some lemmas.
- `Infrastructure_b.v`: contains the type systems of the λB and some lemmas.
- `Deterministic.v`: contains the proofs of the determinism property of λB^g.
- `Typing.v`: contains the proofs of some typing lemmas of λB^g.
- `Typing_b.v`: contains the proofs of some typing lemmas of λB.
- `ttyping.v`: contains the proofs of some elaboration typing lemmas.
- `Typ_Safety.v`: contains the proofs of the type preservation and progress properties of λB^g.
- `soundness.v`: contains the proofs of the soundness theorem with respect to λB.
- `soundness_blame.v`: contains the proofs of the soundness theorem with respect to λB.

9.2 In the `coq/Variant` directory:

`Variant` directory contains the definition and proofs of the variant calculus(λB^r).

- `syntax_ott.v`: contains the locally nameless definitions of λB^r.
- `rules_inf.v`: generated from the **lngen** and modified by us.
- `Infrastructure.v`: contains the type systems of the λB^r and some lemmas..
- `Deterministic.v`: the proofs of the determinism property of λB^r.
- `Type_Safety.v`: the proofs of the type preservation and progress properties of λB^r.
- `criteria.v`: contains the proofs of gradual guarantee theorem of λB^r.
- `Variant_Calculus.v`: contains the proofs of the blame semantics conformance of λB^r to λB^g.

Table 1 Overview of pictures.

Picture/Definition	Page Number	Coq File	Name in Coq
Figure 1 (λB calculus)	Page 5	Calculus/syntaxb_ott.v	
Figure 2 (syntax and well-formed values for λB^g)	Page 11	Calculus/syntax_ott.v	
Definition 1 (Dynamic Types of λB^g)	Page 11	Calculus/syntax_ott.v	principle_type
Figure 3 (Type system of λB^g calculus)	Page 12	Calculus/syntax_ott.v	ttyping
Figure 4 (Typed Reducction for the λB^g calculus.)	Page 13	Calculus/syntax_ott.v	TypedReduce
Figure 5 (Semantics of λB^g)	Page 15	Calculus/syntax_ott.v	step
Figure 6 (Syntax of the λB^r calculus.)	Page 16	Variant/syntax_ott.v	
Figure 7 (Type system of λB^r calculus)	Page 17	Variant/syntax_ott.v	Typing
Definition 17 (Dynamic Types of λB^r)	Page 17	Variant/syntax_ott.v	principal_type
Figure 8 (Typed Reduction for λB^r Calculus)	Page 18	Variant/syntax_ott.v	TypedReduce
Figure 9 (Semantics of λB^r Calculus)	Page 20	Variant/syntax_ott.v	step
Figure 10 (Translation)	Page 21	Variant/syntax_ott.v	trans
Figure 11 (Precision relations)	Page 22	Variant/syntax_ott.v	epre and tpre

10 Correspondence

This section briefly explains the important lemmas, definitions and pictures discussed in the paper and their correspondence with the coq formulation. Table 1 shows the correspondence of pictures or definitions and table 2 shows the correspondence of some important lemma. For example, one can find the **Lemma 2** (Dynamic Types) in file **Calculus/Typing.v** and the lemma name in file is **principle_inf**.

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, page 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*, 2020. Available electronically at `https://coq.inria.fr/distrib/current/refman/`.

6 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.

Table 2 Overview of lemmas.

Name of formalization	Coq File	Name in Coq File
Lemma 4 (Typed Reduction Preserves Values)	Calculus/Type_Safety.v	Tred_value
Lemma 5 (Preservation of TypedReduction)	Calculus/Type_Safety.v	TypedReduce_preservation
Lemma 6 (Progress of Typed Reduction)	Calculus/Type_Safety.v	tred_progress
Lemma 7 (Determinism of Typed Reduction)	Calculus/Deterministic.v	TypedReduce_unique
Lemma 8 (Typed Reduction Respects Consistency)	Calculus/Type_Safety	TypedReduce_sim
Theorem 9 (Determinism of λB^g calculus)	Calculus/Deterministic.v	step_unique
Theorem 10 (Type Preservation of λB^g Calculus)	Calculus/Type_Safety	preservation
Theorem 11 (Progress of λB^g Calculus)	Calculus/Type_Safety	progress
Theorem 12 (Type-Safety of Elaboration)	Calculus/ttyping.v	elaboration_soundness
Theorem 13 (Soundness of reduction)	Calculus/soundness.v	soundness_mul_two
Theorem 14 (Soundness of blame reduction)	Calculus/soundness_blame.v	Soundness_blame_two
Lemma 15 (Soundness of Typed Reduction)	Calculus/soundness.v	Tred_soundness
Lemma 16 (Soundness of Typed Reduction for blame)	Calculus/soundness_blame.v	Tred_blame_soundness
Lemma 18 (Dynamic Types of Values)	Variant/Type_Safety.v	principle_inf2
Lemma 19 (Dynamic Types of Saved Forms)	Variant/Type_Safety.v	principle_inf
Lemma 20 (Checked expressions can be inferred)	Variant/Typing	Typing_chk2
Lemma 21 (Transitivity of typed reduction)	Variant/Type_Safety.v	TypedReduce_trans
Lemma 22 (Preservation of TypedReduction)	Variant/Type_Safety.v	TypedReduce_preservation
Lemma 23 (Progress of Typed Reduction)	Variant/Type_Safety.v	TypedReduce_progress
Lemma 24 (Determinism of Typed Reduction)	Variant/Deterministic.v	TypedReduce_unique
Theorem 25 (Determinism of λB^r calculus)	Variant/Deterministic.v	step_unique
Theorem 26 (Type Preservation of λB^r Calculus)	Variant/Type_Safety	preservation
Theorem 27 (Progress of λB^r Calculus)	Variant/Type_Safety	progress
Theorem 28 (Conformance of blame semantics)	Variant/Variant_Calculus	variant_calculus
Theorem 29 (Static Gradual Guarantee)	Variant/criteria.v	precise_type
Lemma 30 (Dynamic Gradual Guarantee for Typed Reduction)	Variant/criteria.v	tdynamic_guarantee
Theorem 31 (Dynamic Gradual Guarantee)	Variant/criteria.v	dynamic_guarantee
Theorem 32 (Dynamic Gradual Guarantee)	Variant/criteria.v	dynamic_guarantees