Towards Strong Normalization for Dependent Object Types (DOT) (Artifact)*

Fei Wang[1] **and Tiark Rompf**[2]

1 **Purdue University, USA**
`wang603@purdue.edu`
2 **Purdue University, USA**
`tiark@purdue.edu`

Abstract

This artifact provides the fully mechanized proof of strong normalization for $D_{<:}$, a variant of (Dependent Object Types) DOT [3] that excludes recursive functions and recursive types. The intersection type and recursive self type are further integrated, moving towards DOT. The key proof idea follows the method of Girard and Tait [1, 4].

1998 ACM Subject Classification D.3.3 Language Constructs and Features
Keywords and phrases Scala, DOT, strong normalization, logical relations, recursive types
Digital Object Identifier 10.4230/DARTS.3.2.5
Related Article Fei Wang and Tiark Rompf, "Towards Strong Normalization for Dependent Object Types (DOT)", in Proceedings of the 31st European Conference on Object-Oriented Programming (ECOOP 2017), LIPIcs, Vol. 74, pp. 27:1–27:25, 2017.
`http://dx.doi.org/10.4230/LIPIcs.ECOOP.2017.27`
Related Conference European Conference on Object-Oriented Programming (ECOOP 2017), June 18-23, 2017, Barcelona, Spain

1 Scope

The purpose of providing this artifact is to demonstrate the mechanized Coq [2] proof in full detail. The readers can check and make sure that there are no hidden "admits" in the proof, and verify that the lemmas and theorems in the paper are well supported by the lemmas and theorems in the proof.

2 Content

The artifact package includes:

- `SfLib.v`, the software foundation library used in this artifact.
- `dsubsup_total.v`, the strong normalization proof for $D_{<:}$, closely matching the presentation in Section 3 of the companion paper.
- `dsubsup_total_rec.v`, the strong normalization proof for $D_{<:}$ with recursive self type and intersection type, closely matching the presentation in Section 4 of the companion paper.
- `Makefile`
- `README.md`

Appendix A of the companion paper describes the correspondence between the formalism on paper and the development in Coq. Lemma 3 and Theorem 4 of the companion paper are not

* This work was supported in part by NSF through awards 1553471 and 1564207.

```
  COQDEP dsubsup_total_rec.v
  COQDEP dsubsup_total.v
  COQDEP SfLib.v
  COQC SfLib.v
  COQC dsubsup_total.v
  COQC dsubsup_total_rec.v
```

Figure 1 Expected result of successfully running the artifact

included in the artifact because they are about System $F_{<:}$, not System $D_{<:}$. The corresponding lemma and theorem in System $D_{<:}$ are more powerful.

To run and verify our artifact, install "coqc –version 8.6" with

```
brew install coq
```

(for Mac OS), or, follow the instructions on the webpage `https://coq.inria.fr/download`. No extra environment set-up is needed. After installation, change directory to the artifact, and run

```
make
```

in the console. The expected result is that it takes about 10 minutes to finish, and there are no error messages. The literal expected console output is in Figure 1.

If the readers are interested in reviewing the proof in CoqIde or other interpreters, and stepping through the proof, the readers should run

```
coqc SfLib.v
```

in the artifact directory, before stepping through the proof. The command will generate a compiled library at the local directory. Otherwise, an error of

```
The file /path/to/artifact/SfLib.vo contains library Top.SfLib
and not library SfLib
```

may occur. For some background about the potential error, it is because the makefile (which we auto-generated using default flags with the standard `coq_makefile` command) introduces a designated "Top" scope, while "coqc" assumes no such scope by default. It is possible to use this "Top" scope manually by passing the "-R" flag, as in `coqc -R / Top dsubsup_total.v`.

In `dsubsup_total_rec.v`, the `val_type_unfold` lemma is "admitted" for performance reasons. The commented proof for that lemma takes Coq an hour or more to complete (for reasons that are not clear). However, the right-hand side of `val_type_unfold` has been copied and pasted literally from `val_type` (which the readers may verify), so there is no question about the validity of the lemma.

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 latest version of our code is available on the website: `https://github.com/TiarkRompf/minidot/tree/master/ecoop17`

4 Tested platforms

The artifact is known to work on any platform running Coq version 8.6 (`https://coq.inria.fr/download/`).

5 License

6 MD5 sum of the artifact

3bc74da5f2828e59e9df5f8f7993baf6

7 Size of the artifact

40.5 KB

Acknowledgements. The authors wish to thank Nada Amin, Martin Odersky, Sandro Stucki, and the anonymous ECOOP reviewers. This research was supported by NSF through awards 1553471 and 1564207.

References

1 Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

2 The Coq development team. *The Coq proof assistant reference manual*, 2012. Version 8.4. URL: `http://coq.inria.fr`.

3 Tiark Rompf and Nada Amin. Type soundness for dependent object types (DOT). In *OOPSLA*, pages 624–641. ACM, 2016.

4 William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, 32(2):198–212, 1967.