# Artifact Guide

$\iota$DOT: A DOT Calculus with Object Initialization

IFAZ KABIR, University of Alberta, Canada
YUFENG LI, University of Waterloo, Canada
ONDŘEJ LHOTÁK, University of Waterloo, Canada

The Dependent Object Types (DOT) calculus serves as a foundation of the Scala programming language, with a machine-verified soundness proof. However, Scala's type system has been shown to be unsound due to null references, which are used as default values of fields of objects before they have been initialized. This paper proposes $\iota$DOT, an extension of DOT for ensuring safe initialization of objects. DOT was previously extended to $\kappa$DOT with the addition of mutable fields and constructors. To $\kappa$DOT, $\iota$DOT adds an initialization effect system that statically prevents the possibility of reading a null reference from an uninitialized object. To design $\iota$DOT, we have reformulated the Freedom Before Commitment object initialization scheme in terms of disjoint subheaps to make it easier to formalize in an effect system and prove sound. Soundness of $\iota$DOT depends on the interplay of three systems of rules: a type system close to that of DOT, an effect system to ensure definite assignment of fields in each constructor, and an initialization system that tracks the initialization status of objects in a stack of subheaps. We have proven the overall system sound and verified the soundness proof using the Coq proof assistant.

## 1 GETTING STARTED GUIDE

This artifact presents the Coq formalization of the type-safety proof as presented in Section 5 of our paper for:

- the base $\iota$DOT calculus described in our paper[1],
- the extension of $\iota$DOT calculus described Section 6.1 of our paper which can allocate literals on free subheaps, and
- the extension of $\iota$DOT calculus described Section 6.2 of our paper which can reason about objects being locally initialized.

Our Coq proof can be either found at the following link.
https://drive.google.com/file/d/1P2-txE06s5nC08gcy2XujDALdQXD-Ama/view?usp=sharing

### 1.1 Compiling the Proof

System Requirements:

- make
- the dot program from the Graphviz collection
- an installation of Coq 8.10.2, preferably using opam
- the TLC library (version 20181116) which can be be installed through

[1] The version we proved type safe in Coq is a bit more general than the paper version. Since the extensions we were interested in proving type safe needed subtyping between initialization types, we formalized a version of the base calculus which uses subtyping and extended that proof for the various extensions.

Authors' addresses: Ifaz Kabir, University of Alberta, Canada, ikabir@ualberta.ca; Yufeng Li, University of Waterloo, Canada, yufeng.li@uwaterloo.ca; Ondřej Lhoták, University of Waterloo, Canada, olhotak@uwaterloo.ca.

2018. 2475-1421/2018/1-ART1 $15.00
https://doi.org/

```
opam repo add coq-released https://coq.inria.fr/opam/released
opam pin add coq-tlc 20181116
opam install coq-tlc
```

To compile the proof, unzip the artifact, open up a terminal, navigate to the unzipped directory on the command line, and run

```
make
```

This will compile the proof and regenerate the documentation in all the subdirectories.

## 2 STEP-BY-STEP INSTRUCTIONS

### 2.1 Overview

The Coq development presented in this artifact formalizes the type-safety proof of the $\iota$DOT calculus and its extensions as presented in our paper. Specifically, it defines the calculi themselves (abstract syntax, type system, and operational semantics) and their type safety proofs.

We do not prove the type and initialization safety theorem (Theorem 5.1) directly as that requires formal reasoning about divergence, but we prove the progress and preservation lemmas (Lemmas 5.2 and 5.3), and that initial configurations are well-typed (Lemma 5.4). Simple informal reasoning then gives us the type and initialization safety theorem.

### 2.2 How to Review this Artifact

*2.2.1 Inspecting Source Files.* The documentation can be accessed through the Readme.html file in the artifact directory or directly through the various idot-*/src/html directories. The idot-base, idot-free-literals, and idot-local directories contain the Coq code for the base $\iota$DOT calculus, the free literals extension of the $\iota$DOT calculus (Section 6.1), and the local initialization extension (Section 6.2) respectively.

*2.2.2 Verifying Correctness.* Successful compilation using make indicates a correct proof.

You can grep for strings like admit and Admitted in the proof files to verify that we proved all the theorems. You can also browse the code in Emacs using the Proof General mode or coqide and see what assumptions or hypotheses have used by adding the following Coq command:

```
Print Assumptions <lemma name>.
```

For example, to see what assumptions the Preservation Theorem uses, add the command Print Assumptions preservation. in Safety.v on Line 40 (after the proof of the preservation theorem).

### 2.3 Used Libraries and Axioms

The $\iota$DOT calculus extends the $\kappa$DOT calculus of Kabir and Lhoták [2018], which in turn is an extension of the WadlerFest DOT calculus of Amin et al. [2016]. The $\iota$DOT Coq formalization extends the $\kappa$DOT Coq formalization of Kabir and Lhoták [2018], which in turn extended the simplified safety proof of Rapoport et al. [2017].

The $\iota$DOT calculus is formalized using the locally nameless representation with cofinite quantification [Aydemir et al. 2008] in which free variables are represented as named variables, and bound variables are represented as de Bruijn indices. We use the TLC library Arthur Charguéraud that provides useful infrastructure for metatheory proofs. We configure Coq with the following axioms:

- functional extensionality
- propositional extensionality
- indefinite description

These axioms are inherited from the TLC library.

### 2.4 Paper Correspondence

The correspondence between the paper and Coq formalization is documented in the various idot-*/src/README.html files in the artifact directory.

Since most of the paper is about the base ιDOT calculus, the idot-base/src/README.html describes most of the paper correspondence. The idot-base/src/README.html also describes the ways in which the paper version of the ιDOT base calculus differs from the version in the Coq proof. The idot-free-literals/src/README.html file describes how the initialization rules differ from the base calculus in the free literals extension. The idot-free-literals/src/README.html file describes how the initialization rules differ from the base calculus in the free literals extension, and the the idot-local/src/README.html does the same for the local initialization extension.

## ACKNOWLEDGMENTS

Much of this artifact guide is taken from the artifact guide of Rapoport and Lhoták [2019] and adapted for ιDOT.

## REFERENCES

Nada Amin, Samuel Grütter, Martin Odersky, Tiark Rompf, and Sandro Stucki. 2016. The Essence of Dependent Object Types. In *A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday (Lecture Notes in Computer Science)*, Sam Lindley, Conor McBride, Philip W. Trinder, and Donald Sannella (Eds.), Vol. 9600. Springer, 249–272. https://doi.org/10.1007/978-3-319-30936-1_14

Brian Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. 2008. Engineering Formal Metatheory. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Francisco, California, USA) *(POPL '08)*. ACM, New York, NY, USA, 3–15. https://doi.org/10.1145/1328438.1328443

Ifaz Kabir and Ondřej Lhoták. 2018. κDOT: Scaling DOT with Mutation and Constructors. In *Proceedings of the 9th ACM SIGPLAN International Symposium on Scala* (St. Louis, MO, USA) *(Scala 2018)*. ACM, New York, NY, USA, 40–50. https://doi.org/10.1145/3241653.3241659

Marianna Rapoport, Ifaz Kabir, Paul He, and Ondřej Lhoták. 2017. A Simple Soundness Proof for Dependent Object Types. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 46 (Oct. 2017), 27 pages. https://doi.org/10.1145/3133870

Marianna Rapoport and Ondřej Lhoták. 2019. *A Path to DOT: Formalizing Fully Path-Dependent Types (Artifact)*. https://doi.org/10.5281/zenodo.3366234