OOPSLA 18, Paper #131 Artifact
Step-By-Step Guide

Fabian Muehlboeck, Cornell University
Ross Tate, Cornell University
({fabianm,ross}@cs.cornell.edu)

September 8, 2018

1 About Our Artifact

Our Coq formalization was developed to match the paper as closely as possible. There are a few points where our formalization is stronger than what the paper describes. In particular, instead of just proofs with assumptions, we allow proofs with assumptions and monotonicity (as discussed in Section 7.1) in several requirements, which is a relaxation of said requirements. There are only two other points in which our formalization is different from the paper:

1. For engineering reasons, the requirements of each section are front-loaded before the proofs of the Lemmas and Theorems of that section. This does not make it obvious that only the "hitherto" requirements are used in the proofs of the corresponding Lemmas.
2. Due to a special internal representation we use for types in the proofs of Section 4, the intermediate Lemmas used to prove transitivity have a slightly different form, and in particular Integrated Monotonicity and Integrated Assumptions are proved together in a single lemma. The original Lemmas are still proven, and they still represent what is fundamentally going on in the proof, but are not used in the rest of the proofs.

All these differences are pointed out at the respective points in the Coq code in coqdoc.

1.1 Supported Claims

The presented Coq framework supports all claims made in Sections 3 through 5 of the paper. The table below (the same as in `index.html`, where it may be more readable) gives an overview of which parts of those sections correspond to which parts in the formalization, and how:

Concept from Paper	**Corresponding Definitions/Lemmas/Theorems**	**Comment**
Literals	`Lit` in `Section3_Requirements.v`	Also an implicit parameter in many global definitions
Types	`Notation T := UIType Lit` defined almost everywhere	`UIType` is defined in `Section3_Requirements.v`
Declarative Literal Subtyping Rules	`DRule` and `DPremise` in `Section3_Requirements.v`	
Declarative Subtyping	`dsub` in `Section3_Requirements.v`	Based on `dsuba` and `dsubf` in `Section3_Requirements.v`

Reductive Literal Subtyping Rules	`RRule` and `RPremise` in `Section3_Requirements.v`	
Reductive Subtyping	`rsub` in `Section3_Requirements.v`	Based on `uisub`, `lsub`, and `rsubf` in `Section3_Requirements.v`
Requirement 1: Syntax-Directedness	`SyntaxDirectedness_Rules` and `SyntaxDirectedness_Premises` in `Section3_Requirements.v`	
Requirement 2: Well-Foundedness	`m`, `M`, `mlt`, `mltwf`, `m_ui_l`, `m_ui_r`, and `m_lit` in `Section3_Requirements.v`	
Requirement 3: Literal Reflexivity	`LiteralReflexivity` in `Section3_Requirements.v`	
Declarative Subtyping with Assumptions	`dsuba` in `Section3_Requirements.v`	
Reductive Subtyping with Assumptions	`rsubam` in `Section3_Requirements.v`	Also includes Monotonicity, which makes the proof stronger than in the paper (see explanation at definition)
Requirement 4: R-to-D Literal Conversion	`RRuleToDProof` in `Section3_Requirements.v`	
Requirement 5: D-to-R Literal Conversion	`DRuleToRProof` in `Section3_Requirements.v`	
Requirement 6: Literal Transitivity	`LiteralTransitivity` in `Section3_Requirements.v`	
Decidability of Declarative Subtyping Theorem	`DecidabilityOfDeclarativeSubtyping` in `Section3_Proofs.v`	
Extension Axioms	`extension` in `Section4_Requirements.v`	
Extended Subtyping	`esub` in `Section4_Requirements.v`	Based on `dsubda` and `extension` in `Section4_Requirements.v`
Integrator (DNF_c)	`Integrate` in `Section4_Requirements.v`	
Intersector ($\cap$)	`intersect` in `Section4_Requirements.v`	
Requirement 7: Intersector Completeness	`IntersectorCompleteness` in `Section4_Requirements.v`	
Requirement 8: Intersector Soundness	`IntersectorSoundness` in `Section4_Requirements.v`	
Lemma 1: Integrated Soundness	`IntegratedSoundness` in `Section4_Proofs.v`	
Requirement 9: Measure Preservation	`MeasurePreservation` in `Section4_Requirements.v`	
Lemma 2: Integrated Decidability	`IntegratedDecidability` in `Section4_Proofs.v`	
Requirement 10: Literal Dereliction	`LiteralDereliction` in `Section4_Requirements.v`	

Lemma 3: Dereliction	`Dereliction` in `Section4_Proofs.v`	
Intersected Predicate (ϕ)	`intersected` in `Section4_Requirements.v`	
Integrated Predicate ($\mathsf{dnf}_\phi/\mathsf{dnf}_\phi^\cap$)	`Integrated` and `Integrated_int` in `Section4_Requirements.v`	
Requirement 11: Intersector Integrated	`IntersectorIntegrated` in `Section4_Requirements.v`	
Lemma 4: Integrator Integrated	`IntegratorIntegrated` in `Section4_Proofs.v`	
Requirement 12: Literal Promotion	`LiteralPromotion` in `Section4_Requirements.v`	
Lemma 5: Promotion	`Promotion` in `Section4_Proofs.v`	
Lemma 6: Integrated Monotonicity	`IntegratedMonotonicity` in `Section4_Proofs.v`	Actually mostly proved together in `integrated_assumptions'` in `Section4_Proofs.v`
Lemma 7: Integrated Assumptions	`IntegratedAssumptions` in `Section4_Proofs.v`	
Lemma 8: Integrated Promotion	`IntegratedPromotion` in `Section4_Proofs.v`	
Lemma 9: Integrated Reflexivity	`IntegratedReflexivity` in `Section4_Proofs.v`	
Lemma 10: D-to-I Literal Conversion	`DeclarativeToIntegratedLiteralConversion` in `Section4_Proofs.v`	
Lemma 11: Integrated Transitivity	`IntegratedTransitivity` in `Section4_Proofs.v`	
Lemma 12: Integrated Completeness	`IntegratedCompleteness` in `Section4_Proofs.v`	
Decidability of Extended Subtyping Theorem	`DecidabilityOfExtendedSubtyping` and `OptimizedDecidabilityOfExtendedSubtyping` in `Section4_Proofs.v`	
Requirement 13: Intersected Preservation	`IntersectedPreservation` in `Section5.v`	
Integrator Composability Theorem	Module `Composition` in `Section5.v`	Defines its `intersect` and `extension` as described in the paper and satisfies the `Intersector` module type and hence all the Lemmas and Theorems above.

1.2 Claims Not Supported By The Artifact

Anything after Section 5, such as the specific Ceylon extensions.

2 How To Evaluate The Artifact

We documented this Coq formalization in `coqdoc` in many places to both give an overview of how it works and how it corresponds to the claims in the paper. The file `index.html` contains the same table of correspondences between the paper and the formalization as above, as well as the signatures of all the files contained in the

portion of the artifact that is meant to support anything, and explanatory paragraphs for the important parts of those files. We believe that the largest part of evaluating this artifact will be to go through `index.html` along with the paper and examine the correspondences that we list.

3 How To Use This Framework

In general, the Coq files provide a *module type* for each section. To obtain the proofs provided by each section, one needs to instantiate the module type for that section (and the prior sections). One can use the instantiations of those module types to instantiate various modules that the framework provides, akin to the Lemmas in the paper, including at the end a subtyping decider for integrated subtyping with proofs of the expected properties.

We provide a documented example of how to formalize a type system with simple nominal class types and a simple extension for it (i.e. example instantiations of the module types for Sections 3 and 4) in `Example.html`.