The Essence of Nested Composition (Artifact)

Xuan Bi[1]
The University of Hong Kong, Hong Kong, China
xbi@cs.hku.hk

Bruno C. d. S. Oliveira[1]
The University of Hong Kong, Hong Kong, China
bruno@cs.hku.hk

Tom Schrijvers[2]
KU Leuven, Belgium
tom.schrijvers@cs.kuleuven.be

Abstract

The artifact contains the Coq formalization of NeColus, as described in the paper "The Essence of Nested Composition". We document in detail how to build and compile the Coq proofs, as well as the proof structure and organization.

2012 ACM Subject Classification Software and its engineering → Object oriented languages
Keywords and phrases nested composition; family polymorphism; intersection types; coherence
Digital Object Identifier 10.4230/DARTS.4.3.1
Related Conference 32nd European Conference on Object-Oriented Programming (ECOOP 2018), July 19–21, 2018, Amsterdam, Netherlands

1 How to Use

The artifact contains the Coq formalization of NeColus, as described in the paper "The Essence of Nested Composition". We document in detail how to build and compile the Coq proofs, as well as the proof structure and organization.

1.1 Building Instructions

Our Coq proofs are written in *Coq 8.7.2*. The only dependency is the Coq library *metalib*, which is used extensively because we use the *locally nameless representation* for terms and *cofinite quantification* in various judgements.

1.1.1 Prerequisites

1. Install *Coq 8.7.2*. The recommended way to install Coq is via OPAM. Refer to `https://coq.inria.fr/opam/www/using.html` for detailed steps. Or one could download the pre-built packages for Windows and MacOS via `https://github.com/coq/coq/releases/tag/V8.7.2`. Choose a suitable installer according to your platform.
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), then install *metalib*:
   a. Open terminal
   b. Type `git clone https://github.com/plclub/metalib`
   c. Type `cd metalib/Metalib`

[1] Funded by Hong Kong Research Grant Council projects number 17210617 and 17258816
[2] Funded by The Research Foundation - Flanders

d. Type `make install`

1.1.2 Build and Compile the Proofs

1. Unzip the archive file
2. Go to *coq* directory
3. Type `make` in the terminal to build and compile the proofs.
4. You should see something like the following (suppose `>` is the prompt):

```
> make
{ echo "-R . Top" ; ls *.v ; } > _CoqProject && coq_makefile -arg '...
COQDEP target_inf.v
COQDEP syntax_ott.v
COQDEP Target_Safety.v
COQDEP Target_Adequacy.v
COQDEP Subtype_Property.v
COQDEP Source_Property.v
COQDEP Normalize.v
COQDEP Logical_Relation_Infrastructure.v
COQDEP Logical_Relation_Def.v
COQDEP LibTactics.v
COQDEP Infrastructure.v
COQDEP ExtraLemmas.v
COQDEP Compatibility_Lemma.v
COQDEP Coherence.v
COQDEP Coercion_Compatibility.v
COQC LibTactics.v
COQC syntax_ott.v
COQC target_inf.v
COQC Infrastructure.v
COQC Target_Safety.v
COQC Logical_Relation_Def.v
COQC Source_Property.v
COQC Target_Adequacy.v
COQC Normalize.v
COQC Logical_Relation_Infrastructure.v
COQC Compatibility_Lemma.v
COQC Coercion_Compatibility.v
COQC Coherence.v
COQC ExtraLemmas.v
COQC Subtype_Property.v
```

1.1.3 No Axioms

This work checks with Coq's native theory – it includes no axioms or other extensions. You can verify by running `grep "Axiom\|Admitted" *.v`.

1.2 Proof Structure

Table 1 shows how each of the definitions and theorems in the paper corresponds to the formalizations in the artifact. An outline of the main proof files is explained below:

- `spec/source.ott,spec/target.ott` – Ott specifications of NeColus and λ_c (run `make` will generate corresponding Coq files if they do not exist)
- `coq/syntax_ott.v` – Locally Nameless definitions of NeColus and λ_c, plus various judgements. They are generated by Ott (`https://github.com/ott-lang/ott`).
- `coq/target_inf.v` – Auxiliary proofs of Locally Nameless in Coq (most are substitution related lemmas). They are generated by lngen (`https://github.com/plclub/lngen`).
- `coq/Infrastructure.v` - Auxiliary lemmas of NeColus and λ_c
- `coq/Source_Property.v` - Main properties of NeColus
- `coq/Target_Safety.v` - Type safety of λ_c
- `coq/Normalize.v` - Normalization of λ_c
- `coq/Logical_Relation_Def.v` - Definitions of the logical relations
- `coq/Compatibility.v` - Compatibility lemmas of the logical relations
- `coq/Coercion_Compatibility.v` - Coercion compatibility lemmas
- `coq/Coherence.v` - Coherence property of NeColus
- `coq/Subtype_Property.v` - Soundness and completeness of the algorithmic subtyping
- `coq/ExtraLemmas.v` - Some corollaries about the calculus

Table 1 Paper-to-artifact correspondence

Definition / Theorem	Paper	File	Name of formalization
Syntax of NeColus	Page 10, Figure 3	`coq/syntax_ott.v`	`Inductive styp, sexp`
Declarative subtyping	Page 10, Figure 4	`coq/syntax_ott.v`	`Inductive sub`
Type system	Page 11, Figure 5	`coq/syntax_ott.v`	`Inductive has_type`
Disjointness	Page 12, Figure 6	`coq/syntax_ott.v`	`Inductive disjoint`
Syntax of λ_c	Page 12, Figure 7	`coq/syntax_ott.v`	`Inductive typ, exp`
Coercion typing	Page 13, Figure 8	`coq/syntax_ott.v`	`Inductive cotyp`
Preservation	Page 13, Theorem 1	`coq/Target_Safety.v`	`Theorem preservation`
Progress	Page 13, Theorem 2	`coq/Target_Safety.v`	`Theorem progress`
Coercion reduction	Page 14, Figure 9	`coq/syntax_ott.v`	`Inductive step`
Coercion preserve type	Page 13, Lemma 3	`coq/Source_Property.v`	`Lemma co_sound`
Elaboration soundness	Page 13, Lemma 4	`coq/Source_Property.v`	`Lemma elab_sound`
Kleene equality	Page 15, Definition 6	`coq/Logical_Relation_Def.v`	`Definition kleene_equiv`
NeColus contextual equivalence	Page 16, Definition 7	`coq/Coherence.v`	`Definition ctx_equiv`
Logical relation	Page 17, Figure 11	`coq/Logical_Relation_Def.v`	`Fixpoint rel_v`
Disjointness relation	Page 18, Lemma 11	`coq/Compatibility_Lemma.v`	`Lemma disjoint_rel_v`
Interp of contexts	Page 18, Definition 12	`coq/Logical_Relation_Def.v`	`Inductive rel_g`
Logical equivalence	Page 18, Definition 13	`coq/Logical_Relation_Def.v`	`Definition rel_e_open`
Coercion compatibility	Page 19, Lemma 14	`coq/Coercion_Compatibility.v`	`Lemma coercion_compat1/2`
Inference uniqueness	Page 19, Theorem 15	`coq/Source_Property.v`	`Lemma inference_unique`
Fundamental property	Page 19, Theorem 16	`coq/Coherence.v`	`Lemma coherence_log`
Congruence	Page 19, Lemma 17	`coq/Coherence.v`	`Lemma congruence`
Coherence	Page 19, Theorem 10	`coq/Coherence.v`	`Theorem coherence_thm`
Algorithmic subtyping	Page 21, Figure 12	`coq/syntax_ott.v`	`Inductive ASub`
Soundness	Page 22, Theorem 25	`coq/Subtype_Property.v`	`Theorem ASub2sub`
Completeness	Page 22, Theorem 31	`coq/Subtype_Property.v`	`Theorem sub2ASub`