Decidable Subtyping for Path Dependent Types - Artifact Guide

In this document we describe the Coq formalism that accompanies the paper *Decidable Subtyping for Path Dependent Types*. The artifact consists of two separate proofs of subtype decidability for the type systems defined within the paper: Wyv_{core} and Wyv_{fix}. Both of these proofs are packaged within a virtual machine. In both instances, evaluation of the proofs can be done be compiling the coq source. Note: compilation takes a very long time and requires several hours at least for each formalism.

1 GETTING STARTED

To get started,

- Download the `.vdi` file: https://drive.google.com/open?id=1fLHnGtQ0dAikFR8FgTrbl-yBH5ekV_wJ
- Import the virtual-machine into virtual-box (Virtual-box v.6.0 was used to create the vm).
- Start the virtual-machine. Username: `wyvern`. Password: `wyvern`

Or, if you wish to compile the sources separate from the virtual machine, you may download the source from https://github.com/JulianMackay/Wyvern_Formalism. Coq version 8.9.1 is required to compile proofs.

If using the virtual machine, both formalisms are packaged in `~/popl2020/`.

2 WYV_{core}

The formalism for Wyv_{core} can be found at `~/popl2020/wself` To compile, open a terminal and run `make` in this folder.

```
make clean
make
```

The language definitions can be found in

```
wyv_commmon.v
wself.v
```

Algorithm definitions can be found in

```
rhs_mat_tree.v
```

Termination of the algorithm is guaranteed by coq's type system. The definition of the algorithm demonstrates that it terminates. Equivalence with the subtype rules defined in `wself.v` given in

```
wself_subtype_equiv.v
```

as the following theorems:

```
Theorem subtype_implies_sub_type_actual
Theorem sub_implies_subtype_type_actual
```

Author's address:

2020. 2475-1421/2020/1-ART1 $15.00
https://doi.org/

3 WYV_{fix}

The formalism for Wyv_{core} can be found at `~/popl2020/wfix` To compile, open a terminal and run `make` in this folder.

```
make clean
make
```

The language definitions can be found in

```
wyv_commmon.v
wfix.v
```

Algorithm definitions can be found in

```
rhs_mat_tree.v
```

Termination of the algorithm is guaranteed by coq's type system. The definition of the algorithm demonstrates that it terminates. Equivalence with the subtype rules defined in `wfix.v` given in

```
wfix_subtype_equiv.v
```

as the following theorems:

```
Theorem subtype_equivalence1
Lemma subtype_equivalence2
```

(Note: the designation of `subtype_equivalence2` as a `Lemma` is an error, and will be corrected to a `Theorem` at a later date).

4 STRUCTURE OF DECIDABILITY PROOF

The two proofs of decidability use three central definitions.

(1) Subtyping: the definition of subtyping as described in the paper (Figures 7 and 17 in the paper, and Coq Definition `sub_t` in `wself.v` for Wyv_{self}, and Figure 20 of the paper and Coq definition `sub_t` in `wfix.v` for Wyv_{fix}).
Wyv_{self}:

```
Inductive sub_t : env -> ty -> ty -> Prop
```

written as:

$$\Gamma \vdash \tau_1 <; \tau_2$$

Wyv_{fix}:

```
Inductive sub_t : env -> ty -> ty -> env -> Prop
```

written as:

$$\Gamma_1 \vdash \tau_1 <; \tau_2 \dashv \Gamma_2$$

(2) The subtype algorithm: the decision procedure for subtyping (Algorithm 1 in the paper, and the Coq definition `subtype` in `rhs_mat_tree.v`).

```
Program Fixpoint subtype (T1 T2 : tytree) {measure (shape_depth T1 + shape_depth T2)} : bool
```

(3) Conformity: an equivalence between types and the trees that represent them. This is discussed in Section 4.1 of the paper, and is defined by `conforms` in `wself_subtype_equiv.v` and `wfix_subtype_equiv.v`. Conformity enforces the material/shape separation as described in the paper.

```
Inductive conforms : env -> ty -> tytree -> Prop
```

Note that the formalism uses type trees (`tytree`), which are not exactly the type graphs discussed in the paper. Since one of the properties of type graphs is that every cycle contains a shape, every type graph can be represented by a type tree, where the leaves are either $\top$, $\bot$, or a shape. Thus, the type trees of the formalism are derived from type graphs.

Thus, the equivalence proofs (as identified in Sections 1 and 2) prove that for if types τ_1 and τ_2 conform to the type trees `T1` and `T2`, then `subtype T1 T2` will only return `true` if and only if $\Gamma \vdash \tau_1 <: \tau_2$ is derivable (or $\Gamma_1 \vdash \tau_2 <: \tau_2 \dashv \Gamma_2$ in the case of Wyv_{fix}).