A Solution to part 1A of the POPLmark Challenge

Arthur Charguéraud

June 2006

Abstract

Contents

Introduction

The first part of this document gives an overview of the solution, going through all the key ideas used and giving the exact statements of the results proved. Part two presents the definitions used and a few straight-forward lemmas associated to those definitions. Part three contains the statements and the proofs of all the non straight-forward lemmas. In part four we prove that the definitions that we used are equivalent to the ones from the description of the challenge. In the fifth and last part, we discuss the implementation in COQ of our formalization.

This document comes with two COQ source files. The file `solution.v` implements all the results of part 1A. The translation from the paper version (section 2 and 3 of this document) is nearly word to word - apart from the fact that the proofs are much shorter in COQ than on paper. Part five of the document contains a quick tutorial to help beginners in COQ follow through the proofs. Finally, the file `solution_future.v` is a speculation on how the same solution would be implemented if a couple of simple features were added to COQ - see the end of section 5 for more details.

1 Overview of the Solution

We address here the key issues of the challenge: representation of types, representation of environments, and definition of subytping. We also present the results of part 1A as they are stated in this solution.

1.1 Locally Nameless Representation of Types

This representation is half way between the representation with names and the representation with De Bruijn indices. Bound variables, which are bound inside a type, are represented by their De Bruijn index. Free variables, which occur both in the environment and in the types, are represented by names.

The motivation for this encoding is to take the best of names and indices, or rather get rid of the inconveniences coming with each of the two techniques. With locally nameless, there is no need for any shifting because free variables are not indices, and there is no need for alpha-equivalence because two types are equal if and only if they have the same syntactical expression.

In this introduction, we will not detail the inner working of locally nameless, but only describe how to manipulate values encoded with this technique. A well-formed type is one of:

Top	*maximum type*
$\lfloor X \rfloor$	*free variable with name* X
$T_1 \to T_2$	*type of functions*
$\forall\, T_1 . T_2$	*universal type*

A De Bruijn index on its own is never well-formed: indices must always be hidden behind a binder.

The type $\forall\, T_1 . T_2$ corresponds to what we write as $(\forall\, X <: T_1 .\, T_2)$ on paper. The reason why the name X does not appear in the notation $\forall\, T_1 . T_2$ is because the X is encoded using De Bruijn indices and so no name needs to be introduced.

The key operation is to turn the variable bound in T_2 by the binder node $\forall\, T_1 . T_2$ into a free variable, which one needs to do in order to explore T_2. If Y is a variable name, then we write ${T_2}^Y$ the result of

this operation. The equivalent transformation in a name representation takes from $(\forall\ X <: T_1.\ T_2)$ to $[X \to Y]T_2$. The use of that operation appears in the second premise of the SA-ALL rule, which is stated as: $(E,\ X <: T_1) \vdash {S_2}^X <: {T_2}^X$.

1.2 Environments as Sets

A binding $(X <: U)$ is by definition the pair made of a variable name X and a type U. An environment is simply a set of bindings. Note that we still define well-formed environment as lists of bindings built incrementally upon well-formed types. The motivation for having environments defined as sets is to release the constraint of ordering in environments, thus avoiding the need for any permutation lemma.

To describe that a binding $(X <: T)$ belongs to an environment E, we use the belonging relation:

$$(X <: T) \in E$$

An environment F is a weakening of an environment E if and only if the set F contains the set E:

$$E \subset F$$

In the narrowing lemma, we want to describe an environment F equal to a copy of another environment E in which the binding $(N <: Q)$ has been replaced by the binding $(N <: P)$. We can encode this as:

$$\begin{array}{rlll} & (N <: Q) \in E & \wedge & (F \setminus (N <: P)) \subset E \\ \wedge & (N <: P) \in F & \wedge & (E \setminus (N <: Q)) \subset F \end{array}$$

1.3 Subtyping Rules: Changing the `SA-All`

We take exactly the rules given in the description of the challenge, except for SA-ALL. The version of the SA-ALL rule in the paper uses an existential quantification:

$$\frac{E \vdash T_1 <: S_1 \qquad \exists X \# E,\ \ (E,\ X <: T_1) \vdash S_2 <: T_2}{E \vdash (\forall\ X <: S_1.\, S_2) <: (\forall\ X <: T_1.\, T_2)}\ \textsc{sa-all-exists-fresh}$$

From the equivariance property of the subytping relation, the universal quantification is equivalent:

$$\frac{E \vdash T_1 <: S_1 \qquad \forall X \# E,\ \ (E,\ X <: T_1) \vdash S_2 <: T_2}{E \vdash (\forall\ X <: S_1.\, S_2) <: (\forall\ X <: T_1.\, T_2)}\ \textsc{sa-all-forall-fresh}$$

In our solution, we release the constraint of freshness and use the following version of the rule:

$$\frac{E \vdash T_1 <: S_1 \qquad \forall X,\quad (E,\ X <: T_1) \vdash S_2 <: T_2}{E \vdash (\forall\ X <: S_1.\, S_2) <: (\forall\ X <: T_1.\, T_2)}\ \textsc{sa-all-forall}$$

The motivation for releasing freshness is to make the proofs a lot simpler. The consequence of this generalization is that environments may contain several bindings over the same name. However the relation defined with SA-ALL-FORALL is exactly the same as the relation defined with SA-ALL-EXISTS-FRESH or SA-ALL-FORALL. We discuss the change about environments and prove the equivalence result between the two subtyping relations in the adequacy section of this document.

We prove all the results of part 1A using only the SA-ALL-FORALL version of the rule. However, it does not seem possible to prove that substitution preserves subtyping without using an equivariance argument. Proving equivariance in our system is simply proving that the rule SA-ALL-EXISTS-FRESH is admissible. This is a little tedious to do, but not difficult.

1.4 Well-formed Subtyping

When we informally write $E \vdash S <: T$, there is an implicit assumption that E is a well-formed environment and that types S and T are well-formed in E. Instead of adding some well-formation hypotheses in the derivation rules, we will only add well-formation hypotheses at the root of the derivation.

More precisely, we first define "unsafe" subtyping $E \;\backslash\!\!-\; S <: T$ using the derivation rules that we write on paper, i.e. without any well-formation hypothesis. Then we define the "safe" (or "well-formed") subtyping relation $E \vdash S <: T$ as:

$$E \;\backslash\!\!-\; S <: T \quad \wedge \quad E \text{ is well-formed} \quad \wedge \quad S \text{ and } T \text{ are well-formed in } E$$

There are two motivations for this: first the statement of the derivation rules look nicer since they don't contain any well-formation premises, and second it makes the proofs a little easier (especially for the proof-search tactics). The proof of correctness of this encoding requires a short argumentation, which can be found in the adequacy section of this document.

1.5 Statements of the Results

These are the final results from part 1A of the challenge, as stated at the end of the source file `solution.v`:

★ SUBTYPING-REFLEXIVITY:

$$\frac{\vdash E \; ok \qquad E \vdash T \; wf}{E \vdash T <: T}$$

★ SUBTYPING-WEAKENING:

$$\frac{E \vdash S <: T \qquad E \subset F \qquad \vdash F \; ok}{F \vdash S <: T}$$

★ SUBTYPING-NARROWING:

$$\frac{E \vdash P <: Q \qquad E,\; X <: Q \vdash S <: T}{E,\; X <: P \vdash S <: T}$$

★ SUBTYPING-TRANSITIVITY:

$$\frac{E \vdash S <: Q \qquad E \vdash Q <: T}{E \vdash S <: T}$$

2 Formal Definitions

2.1 Types and Environments

Syntax of types in the system $F_{<:}$

T	:=	Top	*maximum type*
		$\lceil i \rceil$	*bound variable with De Bruijn index i*
		$\lfloor X \rfloor$	*free variable with name X*
		$T_1 \to T_2$	*type of functions*
		$\forall\, T_1 . T_2$	*universal type*

Note: to avoid any confusion and to stay as formal as possible, we make a clear distinction between the the index i and the bound variable $\lceil i \rceil$, and between the name X and the free variable $\lfloor X \rfloor$.

A binding is a pair of a variable name and a type, written $(X <: U)$. An environment is a set of bindings. We will use the following notions on sets: operations to insert or remove one element, and relations of belonging of an element to a set and inclusion of a set into another one.

2.2 Substitution from Index to Name

Substitution of a bound variable of index k for a free variable of name Y in a type T, written $T^{[Y/k]}$, is defined recursively over types:

$$\begin{array}{lcll} \mathrm{Top}^{[X/k]} & = & \mathrm{Top} \\ \lceil i \rceil^{[X/k]} \quad (k \neq i) & = & \lceil i \rceil \\ \lceil i \rceil^{[X/k]} \quad (k = i) & = & \lfloor X \rfloor \\ \lfloor Y \rfloor^{[X/k]} & = & \lfloor Y \rfloor \\ (T_1 \to T_2)^{[X/k]} & = & ({T_1}^{[X/k]}) \to ({T_2}^{[X/k]}) \\ (\forall\, T_1 . T_2)^{[X/k]} & = & \forall\, ({T_1}^{[X/k]}) . ({T_2}^{[X/k+1]}) \end{array}$$

The notation ${T_2}^X$ is short for ${T_2}^{[X/0]}$. It is used to pass through bindings, as explained in the first section of this document. This notation occurs in the rules WF-ALL and SA-ALL defined further.

2.3 Well-formed Types and Environments

A type is well-formed in an environment if when we explore it, first we never reach any De Bruijn index, and second any free variable X that we reach there exists a binding of the form $(X <: U)$ in the environment. By "explore" we mean that we deconstruct the type, and each type we go through a binder, we turn the variable bound by this binder from a bound variable into a free variable, and we insert the free variable into the environment. We shall formalize this now.

$E \vdash T\ wf$ is the notation used to describe that the type T is well-formed in the environment E. This relation is formally defined inductively by the following rules:

$$\frac{}{E \vdash \mathrm{Top}\ wf}\ \text{WF-TOP} \qquad \frac{(X <: U) \in E}{E \vdash \lfloor X \rfloor\ wf}\ \text{WF-FVAR}$$

$$\frac{E \vdash T_1\ wf \qquad E \vdash T_2\ wf}{E \vdash T_1 \to T_2\ wf}\ \text{WF-ARROW} \qquad \frac{E \vdash T_1\ wf \qquad \forall X, \forall U,\ (E,\ X <: U) \vdash {T_2}^X\ wf}{E \vdash \forall\, T_1 . T_2\ wf}\ \text{WF-ALL}$$

The universal quantification of the binding $(X <: U)$ in the rule WF-ALL is necessary for well-formation to go through the SA-ALL subtyping rule.

An environment E is well-formed (notation is "$\vdash E\ ok$") if it is built incrementally upon well-formed types, starting initially from the empty environment. The corresponding well-formation rules are:

$$\frac{}{\vdash \varnothing\ ok}\ \text{WF-ENV-EMPTY} \qquad \frac{\vdash E\ ok \qquad E \vdash T\ wf}{\vdash (E,\ X <: T)\ ok}\ \text{WF-ENV-PUSH}$$

2.4 Unsafe and Safe Subtyping

Definition of the unsafe subtyping relation $E \vdash S <: T$ (without well-formation hypotheses):

$$\frac{}{E \vdash S <: \mathrm{Top}}\ \textsc{sa-top} \qquad \frac{}{E \vdash \lfloor X \rfloor <: \lfloor X \rfloor}\ \textsc{sa-refl-tvar} \qquad \frac{(X <: U) \in E \qquad E \vdash U <: T}{E \vdash \lfloor X \rfloor <: T}\ \textsc{sa-trans-tvar}$$

$$\frac{E \vdash T_1 <: S_1 \qquad E \vdash S_2 <: T_2}{E \vdash S_1 \to S_2 <: T_1 \to T_2}\ \textsc{sa-arrow} \qquad \frac{E \vdash T_1 <: S_1 \qquad \forall X,\ (E,\ X <: T_1) \vdash {S_2}^X <: {T_2}^X}{E \vdash \forall\, S_1.\, S_2 <: \forall\, T_1.\, T_2}\ \textsc{sa-all}$$

Note the universal quantification of variable X in the SA-ALL case as discussed in the introduction.

Definition of the subtyping relation $E \vdash S <: T$ as the unsafe relation applied to well-formed arguments:

$$E \vdash S <: T \qquad \Longleftrightarrow \qquad (E \vdash S <: T) \ \wedge\ (\vdash E\ ok) \ \wedge\ (E \vdash S\ wf) \ \wedge\ (E \vdash T\ wf)$$

2.5 Weakening Environments

The traditional statement of lemma WEAKENING PRESERVES SUBTYPING, followed by the one we prove:

$$\frac{E \vdash S <: T}{E,\ E' \vdash S <: T}\ (\textsc{using lists}) \qquad \frac{E \vdash S <: T \qquad E \subset F}{F \vdash S <: T}\ (\textsc{using sets})$$

Definition: F is a weakening of an environment E if F contains all the binding that E has, i.e. $E \subset F$.

★ LEMMAS ABOUT WEAKEN:

$$\frac{}{E \subset E}\ \textsc{weaken-init} \qquad \frac{E \subset F}{(E,\ X <: T) \subset (F,\ X <: T)}\ \textsc{weaken-push} \qquad \frac{E \subset F \qquad E \vdash T\ wf}{F \vdash T\ wf}\ \textsc{weaken-wf}$$

The two first lemma are immediate from the notion of set. The intuition behind WEAKEN-WF is that since all free variables of T are defined in E and that E is contained by F, it must be true that all free variables of T are defined in F. The proof goes by induction on the derivation of $E \vdash T\ wf$.

2.6 Narrowing Environments

The traditional statement of lemma NARROWING PRESERVES SUBTYPING:

$$\frac{E \vdash P <: Q \qquad E,\ N <: Q,\ E' \vdash S <: T}{E,\ N <: P,\ E' \vdash S <: T}\ (\textsc{using lists})$$

And the one that we use:

$$\frac{G\ \textit{updates}\ F\ \textit{at}\ N\ \textit{from}\ Q\ \textit{to}\ P\ \textit{and contains}\ E \qquad E \vdash P <: Q \qquad F \vdash S <: T}{G \vdash S <: T}\ (\textsc{using sets})$$

Where the relation "G *updates* F *at* N *from* Q *to* P *and contains* E" is defined as:

$$E \subset G \quad \wedge \quad (N <: P) \in G \quad \wedge \quad (F \setminus (N <: Q)) \subset G$$

The next four paragraphs discuss this definition.

F is the source environment $(E,\ N <: Q,\ E')$ and G is the target environment $(E,\ N <: P,\ E')$. The environment G is exactly a copy of F in which the binding $(N <: Q)$ has been replaced by the binding $(N <: Q)$. Note that both F and G contain E and E'.

Because the environment E' does not appear outside F and G, we don't need to explicitly say anything about it, and so we can encode the relations between the three other environments E, F, and G as:

$$\begin{array}{lccccc} & E \subset F & \wedge & (N <: Q) \in F & \wedge & (G \setminus (N <: P)) \subset F \\ \wedge & E \subset G & \wedge & (N <: P) \in G & \wedge & (F \setminus (N <: Q)) \subset G \end{array}$$

Important technical remark: for the removal of a binding from an environment (which is a set of bindings), the equality of two bindings must be decidable. This implies that both the equality over variable names and the equality over types must be decidable.

Now let's do a little simplification. In the proof of the narrowing lemma, we are trying to build a derivation of $G \vdash S <: T$. In this proof, we never use the hypotheses describing the environment F, and so it is fine to forget about all the hypotheses about F. This explains why the definition of the "update" relation contains only half of the hypotheses given above.

Using this relation as a premise of the narrowing lemma has for consequence that we prove slightly more general version of this lemma than the usual one. The version we prove states that if $E \vdash P <: Q$ and $F \vdash S <: T$ and if G is *a weakening* of a copy of F in which the $(N <: Q)$ has been replaced by the binding $(N <: Q)$, then $G \vdash S <: T$ holds. This change is quite minor, and it simplifies the proof a little.

Finally, we have three lemmas about the udpate relation, similar to those we had for the weaken relation.

★ UPDATE-INIT:

$$\frac{}{(E,\ N <: P)\ \mathit{updates}\ (E,\ N <: Q)\ \mathit{at}\ N\ \mathit{from}\ Q\ \mathit{to}\ P\ \mathit{and\ contains}\ E}$$

★ UPDATE-PUSH:

$$\frac{G\ \mathit{updates}\ F\ \mathit{at}\ N\ \mathit{from}\ Q\ \mathit{to}\ P\ \mathit{and\ contains}\ E}{(G,\ X <: U)\ \mathit{updates}\ (F,\ X <: U)\ \mathit{at}\ N\ \mathit{from}\ Q\ \mathit{to}\ P\ \mathit{and\ contains}\ E}$$

★ UPDATE-WF:

$$\frac{G\ \mathit{updates}\ F\ \mathit{at}\ N\ \mathit{from}\ Q\ \mathit{to}\ P\ \mathit{and\ contains}\ E \qquad F \vdash T\ \mathit{wf}}{G \vdash T\ \mathit{wf}}$$

The two first lemmas are immediate from the definition. The third one is to be proved by induction on $F \vdash T\ \mathit{wf}$, the intuition being that since the domain of G contains the domain of F, any type T well-formed in F is also well-formed in G. Note that UPDATE-WF is only used in an non-interesting lemma which is not implemented, and so UPDATE-WF itself is not implemented.

3 Formal Proofs

Notation: the abbreviation "IH" stands for "induction hypothesis".

3.1 Reflexivity of Subtyping

★ SUBTYPING-REFLEXIVITY:

$$\frac{\vdash E \ ok \qquad E \vdash T \ wf}{E \vdash T <: T}$$

Proof: since we have all the required well-formation hypotheses, we only need to show the reflexivity of unsafe subypting, i.e. is $E \ \backslash\text{-} \ T <: T$. We prove this by induction on the derivation of $E \vdash T \ wf$:

- WF-TOP: use SA-TOP.
- WF-FVAR: use SA-REFL.
- WF-ARROW: use SA-ARROW, IH1, IH2.
- WF-ALL: use SA-ALL, IH1, and IH2.

3.2 Weakening Preserves Subtyping

We start with an equivalent result about the unsafe subtyping relation:

★ SUB-WEAKENING:

$$\frac{E \ \backslash\text{-} \ S <: T \qquad E \subset F}{F \ \backslash\text{-} \ S <: T}$$

Proof: we show by induction on the derivation of $E \ \backslash\text{-} \ S <: T$ that $\forall F, \ E \subset F \Rightarrow F \vdash S <: T$

- SA-TOP: use SA-TOP.
- SA-REFL-TVAR: use SA-REFL-TVAR.
- SA-TRANS-TVAR: use SA-TRANS-TVAR, with $[\ (X <: U) \in E \ \Rightarrow \ (X <: U) \in F \ ]$ and IH1.
- SA-ARROW: use SA-ARROW, IH1, IH2.
- SA-ALL: use SA-ALL, IH1, and IH2 with WEAKEN-PUSH to show that $(E, \ X <: T) \subset (F, \ X <: T)$.

Now we can prove that weakening preserves the safe subtyping relation:

★ SUBTYPING-WEAKENING:

$$\frac{E \vdash S <: T \qquad E \subset F \qquad \vdash F \ ok}{F \vdash S <: T}$$

Proof: use SUB-WEAKENING, and WEAKEN-WF to show that S and T are well-formed in F.

3.3 Transitivity Proposition

In the description of the challenge, the statement of the transitivity lemma used is:

$$\forall Q, \quad \forall E, \quad \forall S, \quad \forall T, \quad E \vdash S <: Q \quad \Rightarrow \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T$$

The proof goes by induction on Q, with a nested induction on $E \vdash S <: Q$, and in each case a case analysis on $E \vdash S <: T$. It is very important that the variable E be quantified after the variable Q so as to have E universally quantified in the induction hypothesis. Indeed, in the narrowing lemma, which is proved together with transitivity, we need to invoke the transitivity on the type Q in some environment extending E (and not in the environment E itself).

In our solution, we do not want to do an induction on the type Q but on the well-formation judgment $E \vdash Q\ wf$. But this would require to introduce E before doing the induction, and have something like:

$$\forall E, \quad \forall Q, \quad E \vdash Q\ wf \quad \Rightarrow \quad \forall S, \quad \forall T, \quad \Rightarrow \quad E \vdash S <: Q \quad \Rightarrow \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T$$

As we explained earlier, the statement above is not strong enough to go through narrowing. So, we generalize it to support the weakening of the environment in which Q is defined:

$$\begin{array}{l} \forall E, \quad \forall Q, \quad E \vdash Q\ wf \quad \Rightarrow \\ \quad \big[\forall F, \quad E \subset F \quad \Rightarrow \quad \forall S, \quad \forall T,\ E \vdash S <: Q \quad \Rightarrow \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T\big] \end{array}$$

We will say that transitivity of unsafe subtyping holds for a type Q well-formed in an environment E if:

$$\forall F, \quad E \subset F \quad \Rightarrow \quad \forall S, \quad \forall T,\ E \vdash S <: Q \quad \Rightarrow \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T$$

And we will write this proposition as:

$$sub\_trans\_prop\ (E \vdash Q\ wf)$$

3.4 Narrowing Preserves Subtyping

We start with a result about the unsafe subtyping relation:

★ SUB-NARROWING:

$$\frac{\begin{array}{c} G\ updates\ F\ at\ N\ from\ Q\ to\ P\ and\ contains\ E \\ sub\_trans\_prop\ (E \vdash Q\ wf) \qquad E\ \backslash\text{-}\ P <: Q \qquad F\ \backslash\text{-}\ S <: T \end{array}}{G\ \backslash\text{-}\ S <: T}$$

Reminder: the first premise "$G\ updates\ F\ at\ N\ from\ Q\ to\ P\ and\ contains\ E$" is defined as:

$$E \subset G \quad \wedge \quad (N <: P) \in G \quad \wedge \quad (F \setminus (N <: Q)) \subset G$$

Proof: we show by induction on the derivation of $F\ \backslash\text{-}\ S <: T$ the following statement:

$$\forall G, \quad G\ updates\ F\ at\ N\ from\ Q\ to\ P\ and\ contains\ E \quad \Rightarrow \quad G\ \backslash\text{-}\ S <: T$$

- SA-TOP: use SA-TOP.
- SA-REFL-TVAR: use SA-REFL-TVAR.
- SA-TRANS-TVAR:
  - first case $X \neq N$ or $U \neq Q$: use SA-TRANS-TVAR. For the first premise, we use the relation $(F \setminus (N <: Q)) \subset G$ which tells us that $(X <: U) \in F \ \Rightarrow\ (X <: U) \in G$. For the second premise we invoke IH1.

– second case $X = N$ and $U = Q$: the proof of $F \ \backslash\text{-}\ S <: T$ is an instance of SA-TRANS-TVAR:

$$\frac{(N <: Q) \in F \qquad F \ \backslash\text{-}\ Q <: T}{F \ \backslash\text{-}\ \lfloor N \rfloor <: T}$$

and we build another instance of this rule to prove $G \ \backslash\text{-}\ S <: T$:

$$\frac{(N <: P) \in G \qquad \dfrac{\text{SUB-TRANS-PROP}\ \dfrac{\text{SUB-WEAKENING}\ \dfrac{E \ \backslash\text{-}\ P <: Q \qquad E \subset G}{G \ \backslash\text{-}\ P <: Q} \qquad \dfrac{F \ \backslash\text{-}\ Q <: T}{G \ \backslash\text{-}\ Q <: T}\ \text{IH1}}{G \ \backslash\text{-}\ P <: T}}{G \ \backslash\text{-}\ \lfloor N \rfloor <: T}$$

We can invoke the transitivity of subtyping on Q in the environment G because G extends E.

– SA-ARROW: use SA-ARROW, IH1, IH2.

– SA-ALL: use SA-ALL, IH1, and IH2 with UPDATE-PUSH.

Once transitivity is proved, we show the narrowing lemma in its nice-looking presentation:

★ SUBTYPING-NARROWING:

$$\frac{E \vdash P <: Q \qquad E,\ X <: Q \vdash S <: T}{E,\ X <: P \vdash S <: T}$$

Proof: use SUB-NARROWING, SUB-TRANSITIVITY which is proved in the next part, NARROW-INIT to show that $(E,\ X <: P)$ updates $(E,\ X <: Q)$ at X from Q to P, and NARROW-WF to show that S and T are well-formed in $(E,\ X <: P)$.

3.5 Transitivity of Subtyping

We start with an equivalent result about the unsafe subtyping relation:

★ SUB-TRANSITIVITY:

$$\forall E, \quad \forall Q, \quad E \vdash Q\ wf \quad \Rightarrow \quad sub\_trans\_prop\ (E \vdash Q\ wf\ )$$

Reminder: for a given E and a given Q, this expands to:

$$E \vdash Q\ wf \quad \Rightarrow \quad \forall F, \forall S, \forall T, \quad E \subset F \quad \wedge \quad F \ \backslash\text{-}\ S <: Q \quad \wedge \quad F \ \backslash\text{-}\ Q <: T \quad \Rightarrow \quad F \ \backslash\text{-}\ S <: T$$

Proof: by induction on the derivation of $E \vdash Q\ wf$, with a nested induction on the derivation of $F \ \backslash\text{-}\ S <: Q$. In each case, we will do a case analysis on the last rule used in the derivation of $F \ \backslash\text{-}\ Q <: T$. Let's study the cases of $F \ \backslash\text{-}\ S <: Q$ one by one:

– SA-TOP: $Q = \text{Top}$, and so T must also be Top, and therefore we can conclude with SA-TOP.

– SA-REFL-TVAR: $S = Q = \lfloor X \rfloor$, thereafter the goal $F \ \backslash\text{-}\ S <: T$ is identical to the hypothesis $F \ \backslash\text{-}\ Q <: T$

– SA-TRANS-TVAR: we have an instance of SA-TRANS-TVAR and we build another instance of this rule:

$$\frac{(X <: U) \in F \qquad F \;\backslash\!\text{-}\; U <: Q}{F \;\backslash\!\text{-}\; \lfloor X \rfloor <: Q} \quad \Rightarrow \quad \frac{(X <: U) \in F \qquad \dfrac{F \;\backslash\!\text{-}\; U <: Q \qquad F \;\backslash\!\text{-}\; Q <: T}{F \;\backslash\!\text{-}\; U <: T}\ \text{IH1}}{F \;\backslash\!\text{-}\; \lfloor X \rfloor <: T}$$

Note: the induction hypothesis IH1 used is the one coming from the induction on $F \;\backslash\!\text{-}\; S <: Q$.

– SA-ARROW: $S = S_1 \to S_2$ and $Q = Q_1 \to Q_2$, and so T is either Top in which case we use SA-TOP, or it is $T_1 \to T_2$ in which case we use SA-ARROW with the induction hypotheses IH1 and IH2 coming from the induction on $E \vdash Q \ \mathit{wf}$.

– SA-ALL: $S = \forall\, S_1.\, S_2$ and $Q = \forall\, Q_1.\, Q_2$, and so T is either Top in which case we use SA-TOP, or it is $\forall\, T_1.\, T_2$ in which case we use SA-ALL. For both premises of SA-ALL we will use the induction hypotheses IH1 and IH2 coming from the induction on $E \vdash Q \ \mathit{wf}$. The application of IH2 is not immediate, since we need to invoke narrowing. For a given name X, the derivation to use is:

$$\frac{\dfrac{[\text{3 hypotheses}] \qquad (F,\ X <: Q_1) \;\backslash\!\text{-}\; S_2^X <: Q_2^X}{(F,\ X <: T_1) \;\backslash\!\text{-}\; S_2^X <: Q_2^X}\ \text{SUB-NARROWING} \qquad (F,\ X <: T_1) \;\backslash\!\text{-}\; Q_2^X <: T_2^X}{(F,\ X <: T_1) \;\backslash\!\text{-}\; S_2^X <: T_2^X}\ \text{IH2}$$

Where [3 hypotheses] stands for the three other arguments of the narrowing lemma, namely:

  – $(F,\ X <: Q_1) \vdash Q_1 \ \mathit{wf}$, proved by WEAKEN-WF applied to $F \subset (F,\ X <: Q_1)$ and $F \vdash Q_1 \ \mathit{wf}$.
  – $F \;\backslash\!\text{-}\; T_1 <: Q_1$, which is a consequence of $F \;\backslash\!\text{-}\; Q <: T$ being an instance of SA-ALL.
  – $(F,\ X <: T_1)$ *updates* $(F,\ X <: Q_1)$ *at* X *from* Q_1 *to* T_1 *and contains* F, from NARROW-INIT.

Also, to apply IH2 we need to show $(E,\ X <: T_1) \subset (F,\ X <: T_1)$ using WEAKEN-PUSH.

Then we can show the transivity of the safe subtyping relation under its nice-looking presentation:

★ SUBTYPING-TRANSITIVITY:

$$\frac{E \vdash S <: Q \qquad E \vdash Q <: T}{E \vdash S <: T}$$

Proof: apply SUB-TRANSITIVITY on $E \vdash Q \ \mathit{wf}$, and WEAKEN-INIT to show that $E \subset E$.

4 Adequacy of the Solution

4.1 Encoding Well-formation

When we informally write $E \vdash S <: T$, we implicitly assume that E is a well-formed environment and that types S and T are well-formed in E. The way we understand this is that one should add well-formation premises to each of the rules defining the subtyping relation. Therefore, the aim of this part is to prove that having well-formation hypotheses only at the root of the derivation is equivalent to having well-formation hypotheses in each node of the derivation.

The first direction is trivial: if we have well-formation in each node, then in particular we have well-formation at the root. To prove the other way, we need to show that if we have:

$$E \;\backslash\!\text{-}\; S <: T \quad \wedge \quad E \text{ well-formed} \quad \wedge \quad S \text{ and } T \text{ well-formed in } E$$

then we can deduce well-formation hypotheses for all the types and environments which may appear in the derivation of E \- $S <: T$. We prove this by induction on the derivation. The only non-trivial step is to go through a SA-TRANS-TVAR node, for which we need to use the following argument: a well-formed environment only contains well-formed types. Formally this is:

$$\frac{(X <: U) \in E \qquad \vdash E \ ok}{E \vdash U \ wf}$$

To prove this, we do an induction on the derivation of the judgment $\vdash E \ ok$, and because $(X <: U) \in E$ we will get at some point the hypothesis $F \vdash U \ wf$ (from WF-ENV-PUSH) where F is a subset of E. From there we invoke WEAKEN-WF and conclude that $E \vdash U \ wf$.

4.2 Non-Uniqueness of Names Defined in the Environemnt

When using the SA-ALL-EXISTS-FRESH or the SA-ALL-FORALL-FRESH version of the SA-ALL rule, all the bindings contained in an environment bind different names. As a consequence, it is very natural to see environment as a function mapping names to types. Moreover, it is convenient in some proofs to have this hypothesis of functionality:

$$\frac{\vdash E \ ok \qquad (X <: U) \in E \qquad (X <: V) \in E}{U = V} \text{ ENV-FUNCTIONAL}$$

We accept multiple bindings over the same name, and release the freshness constraint:

$$\frac{\vdash E \ ok \qquad X \# E \qquad E \vdash T \ wf}{\vdash (E,\ X <: T) \ ok} \text{ (WITH FRESHNESS)} \qquad \frac{\vdash E \ ok \qquad E \vdash T \ wf}{\vdash (E,\ X <: T) \ ok} \text{ (WITHOUT FRESHNESS)}$$

Note: this is not like shadowing (our environments are sets and not lists). For instance, the two following judgments are both valid:

$$(X <: U) \in (\varnothing,\ X <: U,\ X <: V)$$
$$(X <: V) \in (\varnothing,\ X <: U,\ X <: V)$$

The removal of the freshness hypothesis is clearly a change from the description of the challenge. Is this a problem? We believe that environments are just a tool used to define the subtyping relation. If we find it more convenient to use another tool to define the exact same subtyping relation, then we do not see any reason to not use this other tool.

4.3 Equivalence of `SA-all-forall` and `SA-all-exists`

Let's call $E \vdash_\forall S <: T$ the subtyping relation defined using the SA-ALL-FORALL rule, and let $E \vdash_\exists S <: T$ be the subtyping relation obtained using the SA-ALL-EXISTS version. All the subtyping rules other than SA-ALL are shared by the two relations. Our goal is to prove the equivalence of these two relations:

$$E \vdash_\forall S <: T \qquad \Longleftrightarrow \qquad E \vdash_\exists S <: T$$

4.4 From `SA-all-forall` to `SA-all-exists`

★ FROM FORALL TO EXISTS: $E \vdash_\forall S <: T \quad \Rightarrow \quad E \vdash_\exists S <: T$

Intuitively, if a derivation holds for any variable names X, then in particular it holds for one fresh variable name X. We use this idea to transform the quantification from $(\forall X)$ to $(\exists X \# E)$ in the SA-ALL rule.

Proof: by induction on the derivation of $E \vdash_\forall S <: T$. The only non-trivial case is SA-ALL. We have:

$$\frac{E \vdash_\forall T_1 <: S_1 \qquad \forall Y,\ (E,\ Y <: T_1) \vdash_\forall {S_2}^Y <: {T_2}^Y}{E \vdash_\forall \forall\, S_1.\, S_2 <: \forall\, T_1.\, T_2}\ \text{SA-ALL-FORALL}$$

And we want to show that:

$$\frac{E \vdash_\exists T_1 <: S_1 \qquad \exists X \# E,\ (E,\ X <: T_1) \vdash_\exists {S_2}^X <: {T_2}^X}{E \vdash_\exists \forall\, S_1.\, S_2 <: \forall\, T_1.\, T_2}\ \text{SA-ALL-EXISTS}$$

First we pick a variable name X fresh from E. We can do so because the set of names is assumed to be non-finite and that the environment E is finite. Then we apply the hypothesis $[\forall Y,\ (E,\ Y <: T_1) \vdash_\forall {S_2}^Y <: {T_2}^Y]$ to this particular X and get $[(E,\ X <: T_1) \vdash_\forall {S_2}^X <: {T_2}^X]$. We conclude using the two induction hypotheses.

4.5 From `SA-all-exists` to `SA-all-forall`

Now we want to prove the more interesting direction, and show that:

$$E \vdash_\exists S <: T \quad \Rightarrow \quad E \vdash_\forall S <: T$$

First we define a multi-substitution functions as a total functions mapping names to names. Then we then extend the application of a renaming function m onto types:

$$\begin{array}{lcl} m\ \text{Top} & = & \text{Top} \\ m\ \lceil i \rceil & = & \lceil i \rceil \\ m\ \lfloor X \rfloor & = & \lfloor m\ X \rfloor \\ m\ (T_1 \to T_2) & = & (m\ T_1) \to (m\ T_2) \\ m\ (\forall\, T_1.\, T_2) & = & \forall\, (m\ T_1).\, (m\ T_2) \end{array}$$

And also to the application of a renaming function on environments:

$$\begin{array}{lcl} m\ \varnothing & = & \varnothing \\ m\ (E,\ X <: T) & = & m\ E,\ m\ X <: m\ T \end{array}$$

And let $[X \to Y]m$ stand for the function m' defined by: $(m'\ Z)$ is Y if $X = Z$ and $(m\ Z)$ otherwise.

We are now going to show that:

$$E \vdash_\exists S <: T \quad \Rightarrow \quad \forall m, \quad m\ E \vdash_\forall m\ S <: m\ T$$

We will later conclude taking m to be the identity function, but we need to have this generalized version in order to have an induction principle strong enough to go through the binders in the subtyping derivation.

Proof: by induction on the derivation of $E \vdash_\exists S <: T$.

– SA-TOP: use SA-TOP.

– SA-REFL-TVAR: use SA-REFL-TVAR.

– SA-TRANS-TVAR: use SA-TRANS-TVAR, and prove its first premise using the implication:

$$(X <: U) \in E \quad \Rightarrow \quad (m\ X <: m\ U) \in m\ E$$

which is a strait-forward induction on the derivation of $(X <: U) \in E$.

– SA-ARROW: use SA-ARROW, IH1, IH2.

– SA-ALL: this is the interesting case; we have an instance of SA-ALL-EXISTS:

$$\frac{E \vdash_\exists T_1 <: S_1 \qquad \exists X \# E,\ \ (E,\ X <: T_1) \vdash_\exists S_2^X <: T_2^X}{E \vdash_\exists \forall\ S_1.\, S_2 <: \forall\ T_1.\, T_2}$$

and we want to build for any substitution m an instance of SA-ALL-FORALL:

$$\frac{m\ E \vdash_\forall m\ T_1 <: m\ S_1 \qquad \forall Y,\ \ \big(m\ E,\ m\ Y <: m\ T_1\big) \vdash_\forall (m\ S_2)^Y <: (m\ T_2)^Y}{m\ E \vdash_\forall \forall\ m\ S_1.\, m\ S_2 <: \forall\ m\ T_1.\, m\ T_2}$$

To fix the well-formation hypotheses hidden in the instance of SA-ALL-FORALL, we need to show:

$$E \vdash T\ \ wf \quad \Rightarrow \quad (m\ E) \vdash (m\ T)\ \ wf$$

which goes by induction on $E \vdash T\ \ wf$, using a composition in the case WF-ALL.

Before we may conclude using IH1 and IH2, we need to transform the second premise. Let X be the witness given by the instance of SA-ALL-EXISTS, and let m_Y be equal to $[X \to Y]m$. We show that the second premise in the SA-ALL-FORALL instance above is equivalent to:

$$\forall Y,\ \ \big(m_Y\ (E,\ X <: T_1)\big) \vdash_\forall m_Y\ ({S_2}^X) <: m_Y\ ({T_2}^X)$$

To prove this, we need to show that:

$$\begin{array}{ll} (1) & (Z <: U) \in E \quad \wedge \quad X \# E \quad \Rightarrow \quad ([X \to Y]m)\ Z = m\ Z \\ (2) & E \vdash T\ \ wf \quad \wedge \quad X \# E \quad \Rightarrow \quad ([X \to Y]m)\ T = m\ T \\ (3) & \vdash E\ \ ok \quad \wedge \quad X \# E \quad \Rightarrow \quad ([X \to Y]m)\ E = m\ E \end{array}$$

Lemma (1) is a consequence of $Z \neq X$. Lemma (2) is to be proved by induction on $E \vdash T\ \ wf$ and uses lemma (1) in case WF-FVAR. Lemma (3) is to be proved by induction on $\vdash E\ \ ok$ and uses lemma (4) in case WF-ENV-PUSH.

We also need to show:

$$([X \to Y]m)\ ({S_2}^X) = (m\ S_2)^Y$$
$$([X \to Y]m)\ ({T_2}^X) = (m\ T_2)^Y$$

This is true because X is fresh from S_2 and T_2. Indeed, $\forall\ S_1.\, S_2$ and $\forall\ T_1.\, T_2$ are well-formed in E, and X is fresh from E.

★ FROM EXISTS TO FORALL: $E \vdash_\exists S <: T \quad \Rightarrow \quad E \vdash_\forall S <: T$

Proof: apply the previous lemma with m being the identity function.

5 Implementation in COQ

5.1 Reading Through the File

The file `solution.v` (version `8.0pl3`) is divided in the following sections:

- Using and extending COQ's library
- Types and substitution
- Environments and well-formation
- Definition of subtyping
- Weakening of environments
- Updating of environments
- Results about unsafe subtyping
- Results about safe subtyping

The first part contains some top-level directives for COQ and some user-defined tactics. The aim of those tactics will be explained further. It is not at all necessary to understand the definitions of these tactics. The other parts contain some definitions, statement of lemmas, proofs of lemmas. They also contain a few top-level directives, which are lines starting with either "Notation" (to introduce new notations) or "Hint" (used to expand the search database of the proof-search tactics).

5.2 Implementation of Environments

Instead of using the library for sets defined in COQ to represent environments, we define environments by hand. It would maybe be nicer to work with real sets, but this complicates the proofs a little bit because one has to use the lemmas from the library. We found that this complicates the proofs just by too much for the proof-search tactics to handle it. Our conclusion is that it is very easy to define environments by hand, and it let us work with the specialized notion of set of bindings, which makes life easier.

In the source file, environments are defined as:

$$\begin{array}{lcll} E & := & \varnothing & \textit{empty environment} \\ & & E,\ X <: U & \textit{environment } E \textit{ extended with a binding from variable } X \textit{ to type } U \end{array}$$

Belonging of a binding to an environment is captured by the relation $(X <: T) \in E$ defined as:

$$\frac{}{(X <: U) \in (E,\ X <: U)}\ \textsc{has-here} \qquad \frac{(X <: U) \in E}{(X <: U) \in (E,\ Y <: T)}\ \textsc{has-next}$$

Weakening is a set inclusion, which we encode as an implication:

$$E \subset F \qquad \Longleftrightarrow \qquad \forall X, \forall U, \quad (X <: U) \in E \ \Rightarrow\ (X <: U) \in F$$

Note: lemma WEAKEN-INIT does not appear in the source file because it is a trivial result.

The update relation (*G updates F at N from Q to P and contains E*) which is defined as:

$$E \subset G \quad \wedge \quad (N <: P) \in G \quad \wedge \quad (F \setminus (N <: Q)) \subset G$$

is encoded in our implementation as:

$$E \subset G \quad \wedge \quad (N <: P) \in G \quad \wedge \quad [\, \forall X \, \forall T, \quad X \neq N \ \wedge \ U \neq Q \ \wedge \ (X <: T) \in F \ \Rightarrow \ (X <: T) \in G \,]$$

We did mention the fact that equality over bindings had to be decidable in our solution. The COQ source contains a lemma to prove this result. This lemma named UPDATE-DECIDE states that if $(X <: U)$ and $(N <: Q)$ are two bindings, then:

$$(X = N \ \wedge \ U = Q) \quad \vee \quad (X \neq N \ \vee \ U \neq Q)$$

where the $\vee$ used is the constructive "or", and not the classical "or" (with excluded middle).

5.3 General Working of COQ

COQ let's you formally prove a conclusion P in a given context, i.e. a finite set of hypotheses $(H_i)_{i \in [1..n]}$. The all judgment $(H_i) \vdash P$ is the goal.

There are two interesting ways of proving a goal. First way: solves the goal using a proof search algorithm. Second way: apply a tactic that transforms a goal into one or several subgoals, and then recursively prove all the subgoals.

As a consequence, a COQ proof is a tree, with proof search tactics at leaves and transformation tactics at nodes. One can write a COQ proof either as a tree in the COQ source file, or as a linear juxtaposition of tactics (corresponding to the DFS trasversal of that tree). Example:

```
1st technique:  split_tactic; [ tactic1 | tactic2 ]
2nd technique:  split_tactic. tactic1. tactic2.
```

We use the second technique, and so the solution source file contains proofs as a sequence of tactics, with dots as separators.

For sake of factorization, it is possible to apply a transformation tactic t_1 and then apply a tactic t_2 to all the subgoals generated by t_1. This appears in a COQ source as in "`t1 ; t2`". For instance, if `split_tactic` generates two subgoal, and that both subgoals can be solved using `tactic1`, then one may replace:

```
            split_tactic. tactic1. tactic1.
with:       split_tactic; tactic1.
```

If a tactic t_1 generates a large number of subgoals and we want to apply a tactic t_2 to many of them, and if the tactic t_2 fails on one or more subgoals, then we can't use the construction "`t1 ; t2`". However we can tell COQ to try and use t_2 and do nothing if t_2 fails, using the "try" keyword in the following way: "`t1 ; try t2`".

5.4 Tactics Used in the Implementation

In this presentation, we mix tactics provided by COQ with some tactics that we define ourselves at the top of the source file. Note that we do not use any tactic specific to this project. *Convention:* all the tactics with a name ending with a ° symbol are enhanced versions of the equivalent builtin COQ tactics.

Proof search tactics:

- `auto°` performs a Prolog-style proof search on a limited depth. It solves the goal if possible, otherwise leaves it unchanged.
- `use L` applies the `auto°` tactic telling it that it should use lemma L.

Goal transformation tactics:

- `intros` transforms a goal $(H_i) \vdash [\forall A_1 A_2 ... A_N, P]$ into the goal $[(H_i), A_1, A_2, ..., A_N] \vdash P$.
- `assert° Q` applies on any goal $(H_i) \vdash P$, and produces two subgoals $(H_i) \vdash Q$ and $(H_i), Q \vdash P$.
- `apply° Q`: when Q is a proposition $A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_N \rightarrow R$ and the goal is $(H_i) \vdash R$, it leaves N subgoals: $(H_i) \vdash A_k$ for $k \in [1..N]$.
- `splits°`: when the goal is $(H_i) \vdash P_1 \wedge P_2 \wedge ... \wedge P_N$, it leaves N subgoals $(H_i) \vdash P_k$ for $k \in [1..N]$. For convenience, this tactic calls `intros` first.
- `inversion° X`: performs a case analysis on X when X has an inductive type.
- `induction° X`: performs an induction on X when X has an inductive type.
- `contradiction°`: prove any goal when False is deducible from the hypotheses.

Specialized tactics to deal with equalities:

- `injection H`: when H is an hypothesis of the form $Ca_1...a_N = Cb_1...b_N$ where C is an inductive constructor, then it adds N equalities to the context: $a_k = b_k$ for each $k \in [1..N]$.
- `discriminate`: when an hypothesis in the context has the form $Ca_1...a_N = C'b_1...b'_N$ where C and C' are two distinct inductive constructors, then it proves the goal by contradiction (since that hypothesis is absurd).
- `compare X Y`: applies to any goal $(H_i) \vdash P$, and generates three subgoals: first $[(H_i), X = Y] \vdash P$, second $[(H_i), X \neq Y] \vdash P$, and the third subgoal is to prove that equality between X and Y is decidable and is written as $(H_i) \vdash \{X = Y\} + \{X \neq Y\}$.
- `decide equality°`: proves that equality is decidable between terms of a certain category of types. It solves a goal $(H_i) \vdash \{X = Y\} + \{X \neq Y\}$ when X and Y are values of type T where T is an inductive type such that its constructors do not take proofs or functions as arguments, nor objects in dependent types.

Rewriting tactics:

- `unfold N`: replaces the occurence of the name N by the definition of N.
- `subst`: for any equation $X = Y$ found in the context, replaces all occurences of X by Y everywhere in the goal.

Note: in a few critical places, we will use `eauto` directly instead of `auto°` for efficiency reasons.

We also have a tactic named `generalize_equality` which fixes a problem with the `induction` tactic which sometimes looses information on the way. This tactic is used in the transitivity lemma to transform the conclusion from:

```
        forall (F : env) (S T : typ),
         weaken E F -> F \- S <: Q -> F \- Q <: T -> F \- S <: T
into:
        forall Q' : typ, Q' = Q -> forall (F : env) (S T : typ),
         weaken E F -> F \- S <: Q' -> F \- Q' <: T -> F \- S <: T
```

5.5 Improving COQ

The solution could look a little bit nicer if COQ was improved. The projected results after the modifications proposed below is given in file `solution_future.v`.

1) [not so hard to do, to compile faster] when we do an induction, we would like to be able to tell COQ to try and solve all the subgoals with `auto` but a few of them that we explicitly mention. This would save the useless effort of running a proof search when we know that it is not going to succeed. In our solution, this would cut the compilation time by 2. Indeed a proof-search which fails takes usually much more time than a successful proof-search.

2) [not so hard to do, convenient to have] provide built-in support for all the tactics defined in the first section of the solution, providing a top-level directive to turn into an aggressive auto search mode. This mode would get the proof assistant to try and solve as much as goal as it can on its own.

3) [easy to do, convenient to have] provide a top-level directive to automatically call `Hint Constructors` on each inductive definition and automatically call `Hint Unfold` on each definition.

4) [trivial to do, useful to have] fix a bug which is the loss of the name of the hypotheses which are not used further as part of a dependant type. This would let us invoke `induction H` directly and not have to do some `intros` first.

5) [easy to do, nice-looking] have a nice way to avoid the quantification of all the variables used in each case of an inductive definition. For instance one might want to simply write:

```
SA_trans_tvar : E has X <: U -> E \- U <: T -> E \- (fvar X) <: T
```

And tell somehow that the convention in inductive definitions is that E stands for an environment, X for a name, and T and U for types.

6) [probably difficult to do, important but only in a few particular cases] fix the induction tactic so that it does not lose information as it does when we try to prove transitivity of subtyping with the intuitive sequence of tactics (that is without using our `generalize_equality` tactic):

```
intros. unfold sub_trans_prop.
induction wfQ; intros Q' EQ F S T HW SsubQ QsubT;
induction SsubQ; try discriminate; intros;
inversion QsubT; eauto.
```

7) [difficult to do, quite important in general] provide a better support for finite sets, and make sure that the tactic `auto with sets` can solve all the trivial lemmas such as $E \subset F \Rightarrow (E,\ X <: U) \subset (F,\ X <: U)$. This might be already possible with either the current version or with the subversion of COQ; in such case it would be great if someone could show us how.

8) [difficult to do, not so important (at least in this context)] one might enhance a little bit the unification algorithm associated with the eauto and eapply tactics so that it can figure out the instanciation of all the arguments in the invokation of the lemmas SUB-NARROWING and SUB-TRANSITIVITY.

Conclusions

Key ideas used in this formalization:

- Representation: locally nameless types, environments as sets.
- Well-formed types: well-formation based on freshen, induction on well-formation judgments.
- Subtyping: well-formation hypotheses at the root, release freshness hypotheses in SA-ALL.
- Proofs: design definitions so as to ease the work of proof-search algorithm.

Formalizing metatheory in COQ:

- COQ is a very nice tool, and is way good enough to formalize metatheory (at least as we did). However a proof asssistant is the kind of tool one can always expect more from, and so we pointed out in the last section a list of things which could be improved in COQ.
- The solution makes an extensive use of the proof-search tactics, and it is not sure that other proof assistants (such as Isabelle for instance) can provide a tactic as powerful as the `eauto` tactic.

Acknowledgments

I wish to thanks my advisors Benjamin Pierce and Stephanie Weirich, and more generally the Programming Languages group at UPenn for hosting me this semestre. I am greatful to the French Ministry of Education and Research and to the University of Pennsylvania for funding this work.

References

[1] B. Aydemir, A. Bohannon, M. Fairbairn, J. Foster, B. Pierce, P. Sewell, D. Vytiniotis, G. Washburn, S. Weirich, and S. Zdancewic, *Mechanized Metatheory for the Masses: The PoplMark Challenge*, May 2005, http://fling-l.seas.upenn.edu/~plclub/cgi-bin/poplmark/index.php

[2] The Coq Development Team, *The Coq Proof Assistant Reference Manual Version 8.0*, 2004, http://pauillac.inria.fr/coq/doc/main.html

[3] Xavier Leroy, *POPLmark, locally nameless, in Coq*, October 2005, http://cristal.inria.fr/~xleroy/POPLmark/locally-nameless/

[4] Randy Pollack, *Reasoning About Languages with Binding*, February 2006, http://homepages.inf.ed.ac.uk/rap/export/bindingChallenge_slides.pdf

[This list of references is yet to be completed...]