Type Theory and Coq 2024-2025 (resit)

04-03-2025
12:45-14:45

1. Apply the principal typing algorithm PT to establish whether the following lambda term is typable in simple type theory:

$$\lambda xyz.xyz(zy)$$

Give all intermediate steps of the algorithm. Also, if this term is typable then explicitly give a principal type.

We annotate all variables and applicative subterms with type variables:

$$\lambda \overset{a}{x}\overset{b}{y}\overset{c}{z}.\,\overbrace{\overbrace{\underbrace{\overset{a}{x}\overset{b}{y}}_{f}\,\overset{c}{z}}^{e}(\underbrace{\overset{c}{z}\overset{b}{y}}_{g})}^{d} \;:\; a \to b \to c \to d$$

This gives rise to the following equations, which we simplify according to the PT algorithm:

$$\left\{\begin{array}{lcl} \underline{e} & = & g \to d \\ f & = & c \to e \\ a & = & b \to f \\ c & = & b \to g \end{array}\right. \overset{\text{(I)}}{\Longleftrightarrow} \left\{\begin{array}{lcl} e & = & g \to d \\ \underline{f} & = & c \to g \to d \\ a & = & b \to f \\ c & = & b \to g \end{array}\right. \overset{\text{(I)}}{\Longleftrightarrow}$$

$$\left\{\begin{array}{lcl} e & = & g \to d \\ f & = & c \to g \to d \\ a & = & b \to c \to g \to d \\ \underline{c} & = & b \to g \end{array}\right. \overset{\text{(I)}}{\Longleftrightarrow} \left\{\begin{array}{lcl} e & = & g \to d \\ f & = & (b \to g) \to g \to d \\ a & = & b \to (b \to g) \to g \to d \\ c & = & b \to g \end{array}\right.$$

With these substitutions, the type $a \to b \to c \to d$ becomes one of the principal types of this term:

$$(b \to (b \to g) \to g \to d) \to b \to (b \to g) \to d$$

2. Consider the following proposition of minimal propositional logic:

$$(a \to b) \to (b \to a) \to (a \to a)$$

Someone gives the following proof term from simple type theory for this:

$$\lambda f : a \to b.\, \lambda g : b \to a.\, \lambda x : a.\, g\,(f x)$$

Now answer the following four sub-questions:

(a) Give the natural deduction proof for this proposition that corresponds to this proof term.

$$\dfrac{\dfrac{\dfrac{\dfrac{[b \to a^g] \qquad \dfrac{[a \to b^f] \qquad [a^x]}{b}\,E{\to}}{a}\,E{\to}}{a \to a}\,I[x]{\to}}{(b \to a) \to a \to a}\,I[g]{\to}}{(a \to b) \to (b \to a) \to a \to a}\,I[f]{\to}$$

(b) Does this proof term have a detour? Explain your answer.

No. There is no detour, which is an introduction rule that is directly followed by an elimination rule for the same connective.

(c) Give the full type derivation of this proof term in simple type theory.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash g : b \to a} \qquad \dfrac{\dfrac{}{\Gamma \vdash f : a \to b} \qquad \dfrac{}{\Gamma \vdash x : a}}{\Gamma \vdash fx : b}}{\Gamma \vdash g\,(fx) : a}}{f : a \to b,\ g : b \to a \vdash \lambda x : a.\, g\,(fx) : a \to a}}{f : a \to b \vdash \lambda g : b \to a.\, \lambda x : a.\, g\,(fx) : (b \to a) \to a \to a}}{\vdash \lambda f : a \to b.\, \lambda g : b \to a.\, \lambda x : a.\, g\,(fx) : (a \to b) \to (b \to a) \to a \to a}$$

where

$$\Gamma := f : a \to b,\ g : b \to a,\ x : a$$

(d) Give a Coq proof script for this proposition that corresponds to the proof term. You may only use the tactics `intro`, `intros`, `apply`, `exact` and `assumption`. This proof script should be in the place of the dots in:

```
Lemma exercise_two (a b : Prop) :
  (a -> b) -> (b -> a) -> (a -> a).
Proof.
...
Qed.
```

You do not need to copy the lines of the lemma already given here, just giving the tactics part is enough.

```
intros f g x. apply g. apply f. apply x.
```

3. In the Church-Rosser proof that was presented in the lectures, the relation of *parallel reduction* is inductively defined by the four rules:

$$\frac{}{x \Longrightarrow x}\,(\mathsf{var}) \qquad \frac{M \Longrightarrow M'}{\lambda x.M \Longrightarrow \lambda x.M'}\,(\lambda) \qquad \frac{P \Longrightarrow P' \quad N \Longrightarrow N'}{(\lambda x.P)N \Longrightarrow P'[x := N']}\,(\beta) \qquad \frac{M \Longrightarrow M' \quad P \Longrightarrow P'}{MP \Longrightarrow M'P'}\,(\mathsf{app})$$

Furthermore, an operation M^* on preterms is defined by:

$$\begin{aligned} x^* &:= x \\ (\lambda x.M)^* &:= \lambda x.M^* \\ ((\lambda.P)N)^* &:= P^*[x := N^*] \\ (MN)^* &:= M^*N^* \qquad \text{if } M \text{ is not a } \lambda\text{-abstraction} \end{aligned}$$

The key theorem then is:

$$\forall M,\, P\ (\text{if } M \Longrightarrow P \text{ then } P \Longrightarrow M^*)$$

This theorem is proved by induction. Give the details of the case of that inductive proof for the (λ) rule. Be explicit about what is inducted over, what is the induction hypothesis of this case, and what is to be proved from that.

Quotes from the slides of the third lecture:

> PROOF by induction on the derivation of $M \Longrightarrow P$.

and:

> case (2)
>
> $$\frac{M \Longrightarrow M'}{\lambda x.M \Longrightarrow \lambda x.M'}\,(\lambda)$$
>
> IH: $M' \Longrightarrow M^*$.
> We need to prove: $\lambda x.M' \Longrightarrow (\lambda x.M)^*$
> We have $(\lambda x.M)^* = \lambda x.M^*$.
> $\lambda x.M' \Longrightarrow \lambda x.M^*$ follows immediately from IH and the definition of $\Longrightarrow$.

4. Consider the following proposition of predicate logic:

$$\forall x.\big(\neg p(x) \to \neg\forall y.\, p(y)\big)$$

Now answer the following two sub-questions:

(a) Give a natural deduction proof of this proposition. For all relevant inference rules, show that the variable condition is satisfied.

$$\dfrac{\dfrac{\dfrac{\dfrac{[\neg p(x)^H] \qquad \dfrac{[\forall y.\, p(y)^{H_1}]}{p(x)}\,E\forall}{\bot}\,E{\to}}{\neg\forall y.\, p(y)}\,I[H_1]{\to}}{\neg p(x) \to \neg\forall y.\, p(y)}\,I[H]{\to}}{\forall x.(\neg p(x) \to \neg\forall y.\, p(y))}\,I\forall$$

(b) Give the λP proof term that corresponds to this proof. For this term the context will need to contain $\bot$, where we then define $\neg A := A \to \bot$. Therefore, this proof term needs to be well-typed in the context:

$$D : *,\ p : D \to *,\ \bot : *$$

Note that we just ask for the proof term and *not* for a type derivation for this term.

You are allowed to either use mathematical notation for this term, or to use Coq syntax.

$$\lambda x : D.\, \lambda H : (p\, x \to \bot).\, \lambda H_1 : (\Pi x : D.\, p\, y).\, H\, (H_1\, x)$$

```
fun (x : D) (H : p x -> False) (H1 : forall y : D, p y) =>
  H (H1 x)
```

5. Consider the following four preterms:

$$\begin{gathered}\lambda n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat}\\ \lambda n : \mathsf{nat}.\, \Pi m : \mathsf{nat}.\, \mathsf{nat}\\ \Pi n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat}\\ \Pi n : \mathsf{nat}.\, \Pi m : \mathsf{nat}.\, \mathsf{nat}\end{gathered}$$

Or in Coq syntax:

```
fun (n m : nat) => nat
fun (n : nat) => forall (m : nat), nat
forall (n : nat), fun (m : nat) => nat
forall (n m : nat), nat
```

Now answer the following three sub-questions:

(a) Which of these four preterms is well-typed in the calculus of constructions, with context:

$$\mathsf{nat} : *,\ z : \mathsf{nat},\ s : \mathsf{nat} \to \mathsf{nat}$$

The following three preterms are well-typed:

$$\lambda n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat}$$
$$\lambda n : \mathsf{nat}.\, \Pi m : \mathsf{nat}.\, \mathsf{nat}$$
$$\Pi n : \mathsf{nat}.\, \Pi m : \mathsf{nat}.\, \mathsf{nat}$$

Generally one would write the Π-types as function types, which we also will do in the rest of this answer:

$$\lambda n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat}$$
$$\lambda n : \mathsf{nat}.\, (\mathsf{nat} \to \mathsf{nat})$$
$$\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$$

The preterm

$$\Pi n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat}$$

or

$$\mathsf{nat} \to (\lambda m : \mathsf{nat}.\, \mathsf{nat})$$

is not well-typed, as the right-hand side B of a term $\Pi x : A.\, B$ has to be typed with a sort, but the type of $\lambda m : \mathsf{nat}.\, \mathsf{nat}$ is $\mathsf{nat} \to *$, which is not a sort.

(b) For each of these four preterms that are well-typed: give their type.

The types are:

$$\lambda n : \mathsf{nat}.\, \lambda m : \mathsf{nat}.\, \mathsf{nat} \;:\; \mathsf{nat} \to \mathsf{nat} \to *$$
$$\lambda n : \mathsf{nat}.\, (\mathsf{nat} \to \mathsf{nat}) \;:\; \mathsf{nat} \to *$$
$$\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \;:\; *$$

(c) Of each of these four preterms that are well-typed: which are types themselves?

A type is a term typed by a sort, so the only type in this list is:

$$\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$$

6. Give λP derivations of the following three typing judgments:

$$a : * \vdash a \to a : *$$
$$a : *,\, f : a \to a \vdash a : *$$
$$a : *,\, f : a \to a,\, x : a \vdash fx : a$$

In each derivation, you may replace a sub-derivation of a judgment earlier in the list by vertical dots. For the rules of λP see page 8.

Hint: it is often efficient to use weakening as early as possible.

$$\dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{a : * \vdash a : *} \qquad \dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{a : * \vdash a : *} \quad \dfrac{\dfrac{}{\vdash * : \Box}}{a : * \vdash a : *}}{a : *,\ x : a \vdash a : *}}{a : * \vdash a \to a : *}$$

$$\dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{a : * \vdash a : *} \quad \dfrac{\vdots}{a : * \vdash a \to a : *}}{a : *,\ f : a \to a \vdash a : *}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{a : * \vdash a \to a : *}}{a : *,\ f : a \to a \vdash f : a \to a} \quad \dfrac{\vdots}{a : *,\ f : a \to a, \vdash a : *}}{a : *,\ f : a \to a,\ x : a \vdash f : a \to a} \qquad \dfrac{\dfrac{\vdots}{a : *,\ f : a \to a \vdash a : *}}{a : *,\ f : a \to a,\ x : a \vdash x : a}}{a : *,\ f : a \to a,\ x : a \vdash fx : a}$$

7. Consider the following Coq definitions of the natural numbers and of option types.

```
Inductive nat : Set :=
| O : nat
| S : nat -> nat.

Inductive option (A : Set) : Set :=
| None : option A
| Some : A -> option A.
```

Note that in contrast with the Coq standard library, with these definitions the argument `A` of the `option` constructors is not implicit.

We want to define a predecessor function `pred` with type:

```
pred
     : nat -> option nat
```

for which the following equalities hold:

$$\begin{aligned} \texttt{pred O} &\twoheadrightarrow_{\beta\delta\iota\zeta\eta} \texttt{None nat} \\ \texttt{pred (S } n\texttt{)} &\twoheadrightarrow_{\beta\delta\iota\zeta\eta} \texttt{Some nat } n \end{aligned}$$

Now answer the following two sub-questions:

(a) Define this function `pred` using `match`.

```
Definition pred (n : nat) {struct n} : option nat :=
  match n with
  | O => None nat
  | S m => Some nat m
  end.
```

(b) Define this function `pred` using an application of `nat_rec`.

```
Definition pred : nat -> option nat :=
  nat_rec (fun _ => option nat) (None nat)
    (fun (m : nat) (_ : option nat) => Some nat m)
```

8. Consider the following Coq definition:

```
Inductive foo (A B : Prop) : Prop :=
| bar : A -> foo A B.
| baz : B -> foo A B.
```

Now answer the following three sub-questions:

(a) Which logical connective is defined here under the name of `foo`.
Disjunction.

(b) Give the type of the dependent induction principle `foo_ind_dep`.

```
foo_ind_dep
     : forall (A B : Prop) (P : foo A B -> Prop)
       (forall a : A, P (bar A B a) ->
       (forall b : B, P (baz A B b) ->
       forall H : foo A B, P H
```

(c) Give the type of the non-dependent induction principle `foo_ind`.

```
foo_ind
     : forall A B P : Prop,
       (A -> P) -> (B -> P) -> foo A B -> P
```

9. Answer the following two sub-questions:

(a) What property of a type system is abbreviated as SR. Give both the term that this is an abbreviation of, as well as the statement of this property.
Subject Reduction:
If $\Gamma \vdash M : A$ and $M \rightarrow_\beta M'$ then $\Gamma \vdash M' : A$.

(b) One proves SR with induction on a type derivation, together with a basic property called the *substitution property.* Give the statement of this property.

If Γ, $x : B$, $\Delta \vdash M : A$ and $\Gamma \vdash N : B$ then
Γ, $\Delta[x := N] \vdash M[x := N] : A[x := N]$.

(Do not give any proofs, neither of SR nor of the substitution property.)