Type Theory and Coq 2024-2025
(resit)

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

Write your name and student number on each paper that you hand in.

This exam consists of 9 exercises. Each exercise is worth 10 points. The first 10 points are free. The final mark is the number of points divided by 10.

Write all natural deduction proofs and type derivations using the notation from Femke's course notes.

Good luck!

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.

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.

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

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

   (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.

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

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

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.

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.

   (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.

5. Consider the following four preterms:

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

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:

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

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

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

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 5.

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

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:

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

Now answer the following two sub-questions:

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

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

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`.

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

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

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.

(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.

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

Typing rules of λP

axiom

$$\frac{}{\vdash * : \Box}$$

variable

$$\frac{\Gamma \vdash A : s}{\Gamma,\, x : A \vdash x : A}$$

weakening

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma,\, x : C \vdash A : B}$$

application

$$\frac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

abstraction

$$\frac{\Gamma,\, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$$

product

$$\frac{\Gamma \vdash A : * \qquad \Gamma,\, x : A \vdash B : s}{\Gamma \vdash \Pi x : A.\, B : s}$$

conversion

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \quad \text{when } B =_\beta B'$$