Type Theory and Coq 2023-2024

16-01-2024
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. Consider the lambda term:

$$\lambda xy.\, x(y(xy))$$

   Apply the PT algorithm to this term, and determine either a principal type for it, or the fact that this term is not typable in simple type theory. Explicitly give all stages of the algorithm.

2. Consider the following type of simple type theory:

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

   Now answer the following two questions:

   (a) Give an inhabitant of this type.

   (b) Give a type derivation that shows that this inhabitant has the appropriate type.

3. Consider the following term of λP:

$$\lambda H : (\Pi x : D.\, \Pi y : D.\, Rxy).\, \lambda x : D.\, \lambda y : D.\, Hyx$$

   In this term there are free variables

$$\begin{aligned} &D : * \\ &R : D \to D \to * \end{aligned}$$

   which represent the domain of quantification, and a binary relation on this domain.

   Now answer the following three questions:

   (a) This term is a proof term of a proof in predicate logic. Give the statement that is proved as a formula of predicate logic.

(b) Give the natural deduction proof of this statement that corresponds to this term. If you need to check a variable condition, indicate where this is the case, and why it holds.

(c) Does this proof contain a detour? Explain your answer.

Note that you should not give a type derivation of this term.

4. Give type derivations in λP of the following judgment:

$$a : *,\ x : a,\ y : a \vdash y : a$$

For the rules of λP see page 5.

5. An impredicative definition of the Booleans in $\lambda 2$ is:

$$\mathsf{bool}_2 := \Pi a : *.\, a \to a \to a$$

Define three $\lambda 2$ terms (where ite stands for '*if then else*'):

$$\begin{aligned} \mathsf{true}_2 &: \mathsf{bool}_2 \\ \mathsf{false}_2 &: \mathsf{bool}_2 \\ \mathsf{ite}_2 &: \Pi a : *.\, \mathsf{bool}_2 \to a \to a \to a \end{aligned}$$

such that:

$$\begin{aligned} \mathsf{ite}_2\, A\, \mathsf{true}_2\, M N &\twoheadrightarrow_\beta M \\ \mathsf{ite}_2\, A\, \mathsf{false}_2\, M N &\twoheadrightarrow_\beta N \end{aligned}$$

It is sufficient that these reductions hold, you do not need to show this explicitly.

6. We define an inductive type of lists of Booleans in Coq:

```
Inductive boollist : Set :=
| nil : boollist
| cons: bool -> boollist -> boollist.
```

The recursor of this type has type:

```
boollist_rec
     : forall P : boollist -> Set,
       P nil ->
       (forall (b : bool) (l : boollist), P l -> P (cons b l)) ->
       forall l : boollist, P l
```

Now answer the following two questions:

(a) Define a function `count_trues` which counts the number of elements in the list that are equal to `true`, using `Fixpoint` and `match`. This function should have type:

```
count_trues
     : boollist -> nat
```

If you like, you can abbreviate

```
match M with
| true => N_1
| false => N_2
end
```

as

```
if M then N_1 else N_2
```

(b) Define the function `count_trues_rec` that computes the same count, by applying `boollist_rec` to appropriate arguments.

7. We define an inductive type in Coq of :

```
Inductive vec (A : Set) : nat -> Set :=
| vnil : vec A 0
| vcons : forall (n : nat) (x : A) (l : vec A n), vec A (S n).
```

Give the type of the (dependent) induction principle `vec_ind` for this type.

8. Consider the untyped lambda term:

$$M_8 := III(II)$$

In this we have as always that $I := (\lambda x.\, x)$.

Now answer the following two questions:

(a) Give the reduction graph of this term. If there are multiple ways to reduce a term to some other term, indicate this with multiple arrows.

(b) Compute M_8^* and $(M_8^*)^*$ and $((M_8^*)^*)^*$. Just giving the answers is enough, you do not need to explain how you obtained them.

The definition of M^* is:

$$\begin{aligned} x^* &= x \\ (\lambda x.\, M)^* &= \lambda x.\, M^* \\ (MN)^* &= \begin{cases} P^*[x := N^*] & \text{if } M = \lambda x.\, P \\ M^* N^* & \text{otherwise} \end{cases} \end{aligned}$$

9. Consider the lambda term:

$$\lambda x.\, (\lambda f.\, f(f(fx)))\, (\lambda y.\, (\lambda z.\, z)\, y)$$

This term is typable in simple type theory. A typed version is:

$$\lambda x : a.\, (\lambda f : a \to a.\, f(f(fx)))\, (\lambda y : a.\, (\lambda z : a.\, z)\, y)$$

or in Coq notation:

```
fun x : a => (fun f : a -> a => f (f (f x)))
  (fun y : a => (fun z : a => z) y)
```

Now answer the following three questions:

(a) Indicate what the redexes in this term are. You can do this either in the untyped or in the typed version of the term.

(b) For each of these redexes give its height.

(c) Indicate which redex or redexes may be contracted according to the reduction strategy from Turing's proof of weak normalization for simple type theory.

The height of a redex $(\lambda x : A.\, M)\, N$ is the height of the type of $(\lambda x : A.\, M)$. The height function h is defined on types by:

$$h(a) = 0 \qquad \text{for atomic types } a$$
$$h(A \to B) = \max(h(A) + 1, h(B))$$

which implies that:

$$h(A_1 \to \cdots \to A_n \to a) = \max(h(A_1), \ldots, h(A_n)) + 1$$

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

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