Type Theory and Coq 2023-2024

Resit
2024-04-03
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:

   $$\lambda xyz.\, xy(z(zy))$$

   Use the PT algorithm to either find a principal type for this term, or to establish that it is not typable in simple type theory. Explicitly give all the intermediate steps of the algorithm.

2. Consider the following proof of minimal propositional logic:

   $$\dfrac{\dfrac{\dfrac{\dfrac{[a \to b \to c^{H_1}] \quad [a^{H_3}]}{b \to c} E{\to} \quad [b^{H_2}]}{c} E{\to}}{a \to c} I[H_3]{\to}}{b \to a \to c} I[H_2]{\to}}{(a \to b \to c) \to (b \to a \to c)} I[H_1]{\to}$$

   Give the corresponding proof term in Church-style simple type theory. Use the labels of the assumptions to name the variables.

3. Consider the following Coq script:

   ```
   Definition id (a : Set) : a -> a.
   intro x; apply x.
   Defined.
   Print id.
   ```

   Now answer the following two questions:

   (a) Give both the term and the type that the `Print` commando prints in Coq syntax, so using `fun` and `forall` and the sort `Set`.

   (b) Give both the term and the type that the `Print` commando prints in the mathematical notation of the lambda cube, so using λ and Π and the sort $*$.

4. Consider the following lambda term of λP:

$$\lambda H_1 : (\Pi x : D.\, px).\, \lambda H_2 : (\Pi x : D.\, px \to qx).\, (\lambda x : D.\, H_2 x (H_1 x))c$$

This term is well-typed in the context:

$$\begin{aligned} &D : *\,, \\ &p : D \to *\,, \\ &q : D \to *\,, \\ &c : D \end{aligned}$$

Now answer the following six questions:

(a) What is the beta redex in this term.

(b) Give the normal form of this term.

(c) Give the natural deduction proof of predicate logic that corresponds to the term from the exercise.

(d) Give the variable condition of this proof and show that it is satisfied.

(e) What is the detour in this proof.

(f) Give the normal form of the proof.

5. *(This exercise may take some time, consider postponing it until the end of the exam.)*

We want to give a derivation in λP of the judgement:

$$D : *,\, q : D \to *,\, c : D \vdash qc : *$$

See page 4 for the typing rules of λP. We do this in three stages, where in a later stage you may abbreviate a derivation that already has been given by vertical dots.

(a) Give a λP derivation of:

$$D : * \vdash D \to * : \Box$$

(b) In the remainder of this exercise we abbreviate:

$$\Gamma := D : *,\, q : D \to *$$

Give a λP derivation of:

$$\Gamma \vdash D : *$$

(c) Give a λP derivation of:

$$\Gamma,\, c : D \vdash qc : *$$

6. The impredicative definition of conjunction in minimal second order propositional logic is:

$$A \wedge_2 B := (\forall c.\, (a \to b \to c) \to c)$$

Give the corresponding impredicative definition of disjunction, so give a definition of $A \vee_2 B$.

7. This exercise is about unlabeled binary trees. These just consist of binary nodes and leaves, and contain no data. Now answer the following four questions:

   (a) Give the Coq definition of an inductuctive type `bintree` with constructors `leaf` and `node` that represents this kind of tree.

   (b) Give a Coq definition using `Fixpoint` and `match` of a function `count_leaves` that counts the number of leaves in a tree.

   (c) Give the type of the dependent recursion principle `bintree_rec` for your type.

   (d) Give a Coq definition by applying this recursion principle to define a function `count_leaves_rec` that counts the leaves in a tree.

8. This exercise is about the `even` predicate in Coq, which is defined as:

```
Inductive even : nat -> Prop :=
| even_O : even O
| even_SS : forall n : nat, even n -> even (S (S n))
```

   Now answer the following three questions:

   (a) Give the proof that four is even, i.e., a Coq term with type:

```
even (S (S (S (S O))))
```

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

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

9. The proof of strong normalisation of $\lambda\!\to$ that was presented in the lectures associates a saturated set of untyped lambda terms $[\![A]\!]$ with each simple type A. This is defined recursively on the structure of the type as:

$$\begin{aligned} [\![a]\!] &= \mathsf{SN} && \text{for atomic types } a \\ [\![A \to B]\!] &= \ldots && \text{for types } A \text{ and } B \end{aligned}$$

   Here SN it the set of strongly normalizing untyped lambda terms.

   Complete this by replacing the dots by a proper definition of $[\![A \to B]\!]$.

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

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