Type Theory and Coq 2022-2023

16-11-2022
10:30-13:00

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

This exam consists of 12 exercises. Each exercise is worth 15 points. The first 20 points are free. The final mark is the number of points divided by 20.

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

Good luck!

The next four exercises are concerned with simple type theory and propositional logic.

1. Give a type derivation in Church-style simple type theory of the judgment:

$$x : (a \to b) \to a \to b,\ y : a \to b \ \vdash x\,(\lambda z : a.\, xyz) : a \to b$$

   You may abbreviate contexts in the style:

$$\Gamma_1 := x : (a \to b) \to a \to b,\ y : a \to b$$

2. Consider the following Coq script:

```
Parameter a b c : Prop.

Lemma S : (a -> b -> c) -> (a -> b) -> a -> c.
intros x y z. apply x.
- apply z.
- apply y. apply z.
Qed.
```

   Give the (Church-style) term that Coq builds with this script, i.e., the term that will be printed by the command:

```
Print S.
```

   You may write this term both in mathematical style or using Coq syntax, whatever you prefer.

3. Apply the PT algorithm to determine whether the following lambda term is typable in Curry-style simple type theory:

$$\lambda xy.x\,(\lambda z.zyx)$$

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

4. Consider the following formula of propositional logic:

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

   (a) Give a natural deduction proof of this formula.

   (b) Give the proof term that corresponds to this proof. In this term you may use the constants:

$$\begin{array}{lcl} a & : & * \\ b & : & * \\ c & : & * \\ \mathsf{and} & : & * \to * \to * \\ \mathsf{conj} & : & \Pi\, a : *.\ \Pi\, b : *.\ a \to b \to \mathsf{and}\ a\ b \\ \pi_1 & : & \Pi\, a : *.\ \Pi\, b : *.\ \mathsf{and}\ a\ b \to a \\ \pi_2 & : & \Pi\, a : *.\ \Pi\, b : *.\ \mathsf{and}\ a\ b \to b \end{array}$$

   Or, in Coq syntax:

```
a     : Prop
b     : Prop
c     : Prop
and   : Prop -> Prop -> Prop
conj  : forall a b : Prop, a -> b -> and a b
proj1 : forall a b : Prop, and a b -> a
proj2 : forall a b : Prop, and a b -> b
```

   You may write this term both in mathematical style or using Coq syntax, whatever you prefer.

The next two exercises are concerned with dependent types and predicate logic.

5. Consider the following natural deduction proof in minimal predicate logic:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\forall y.\, p(y) \to q(y)^{H_2}]}{p(y) \to q(y)}\, E\forall \quad \dfrac{[\forall y.\, p(y)^{H_1}]}{p(y)}\, E\forall}{q(y)}\, E\to}{\forall y.\, q(y)}\, I\forall}{q(f(x, g(x)))}\, E\forall}{\big(\forall y.\, p(y) \to q(y)\big) \to q(f(x, g(x)))}\, I[H_2]\to}{\big(\forall y.\, p(y)\big) \to \big(\forall y.\, p(y) \to q(y)\big) \to q(f(x, g(x)))}\, I[H_1]\to}{\forall x.\, \big(\big(\forall y.\, p(y)\big) \to \big(\forall y.\, p(y) \to q(y)\big) \to q(f(x, g(x)))\big)}\, I\forall$$

   This proof contains a detour.

   (a) Explain what the detour in this proof is.

(b) Give the λP proof term that corresponds to this proof, in which you may use the constants:

$$\begin{array}{lcl} D & : & * \\ p & : & D \to * \\ q & : & D \to * \\ f & : & D \to D \to D \\ g & : & D \to D \end{array}$$

Or, in Coq syntax:

```
D : Set
p : D -> Prop
q : D -> Prop
f : D -> D -> D
g : D -> D
```

You may write this term both in mathematical style or using Coq syntax, whatever you prefer.

(c) Indicate the subterm that is the redex that corresponds to the detour.

(d) Give the normal form of the proof.

6. Give derivations of the following two typing judgments. See page 6 for the typing rules of λP.

(a)
$$a : *,\ x : a \vdash a \to * : \Box$$

(b)
$$a : *,\ x : a,\ b : a \to * \vdash b\,x : *$$

In this second derivation you may replace instances of the first derivation by dots.

The next two exercises are concerned with polymorphic types and second order propositional logic.

7. Give a proof in second order propositional logic of the formula:

$$a \to \mathsf{or}_2\ a\ b$$

where we defined the impredicative version or_2 of disjunction by:

$$\mathsf{or}_2\ A\ B := \forall c.\,(A \to c) \to (B \to c) \to c$$

8. In the context $b : *,\ t : b$ consider the Church-style $\lambda 2$ type:

$$\forall a.\,a \to b$$

(a) Write this type using PTS syntax, i.e., according to the grammar:

$$M ::= x \mid MM \mid \lambda x : M.\, M \mid \Pi x : M.\, M \mid s$$
$$s ::= * \mid \Box$$

(b) Give the $\lambda 2$ type of this type (its *kind*).

(c) Give an inhabitant of this type, in the given context.

(d) Give the typing judgment of this inhabitant. (Note that you only need to *state* the judgment, and do not need to *derive* it.)

The next two exercises are concerned with inductive types.

9. In Coq the natural numbers are defined by:

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

Its dependent recursion principle `nat_rec` has type:

```
nat_rec :
  forall A : nat -> Set,
    A O ->
    (forall n : nat, A n -> A (S n)) ->
      forall n : nat, A n
```

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

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

The recursion equations that we want to capture are:

```
pred  O    = O
pred (S m) = m
```

In other words, for this function we define the predecessor of zero to be zero.

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

(b) Define this function `pred` as an application of the recursor `nat_rec`.

10. (a) Define an inductive type `list` of polymorphic lists, with constructors `nil` and `cons`.

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

(c) Give the type of the non-dependent recursor `list_rec_nondep` of this type.

The next exercise is concerned with the Church-Rosser proof by Takahashi.

11. Consider the untyped lambda term:

$$(\lambda x.xII)(II)$$

where we abbreviate, as always:

$$I := \lambda x.x$$

(a) Give the full reduction graph for this term.

(b) For each term M in this graph, give the result of the full development M^*, as defined by:

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

The next exercise is concerned with the strong normalization proof using saturated sets.

12. (a) Give the recursive definition of the semantics $[\![A]\!]_\rho$ from the strong normalization proof for $\lambda 2$.

(b) Explain what A, ρ and $[\![A]\!]_\rho$ are in this definition.

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

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