Type Theory and Rocq 2025-2026

24-10-2025
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. (a) Give an inhabitant of the simple type $a \to (a \to b) \to b$, as a term of Church-style simple type theory.
   (You should not give a type derivation for this term, later in the exam there will be another exercise that asks for a type derivation in simple type theory.)
   (b) Give the proof in minimal propositional logic that corresponds to this term.
   (c) Does your proof contain a detour? Explain your answer.
   (d) Give a Rocq version of this proof, using tactics. You may only use the tactics `intro`, `intros`, `apply`, `exact` and `assumption`. The proof script should be in the place of the dots in:

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

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

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

$$\lambda xy.x(\lambda z.xyz)$$

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

3. Give the full reduction graph of the untyped lambda term

$$\mathsf{2IK}$$

   We use the customary abbreviations:

$$\mathsf{I} := \lambda x.x$$
$$\mathsf{K} := \lambda xy.x$$
$$\mathsf{2} := \lambda fx.f(fx)$$

Note that we do not reduce like in combinatory logic here. This means that, although the term $\mathsf{2}$ expects two arguments as a combinator, we reduce for example:

$$\mathsf{2\,I} \to_\beta \lambda x.\mathsf{I}\,(\mathsf{I}x)$$

If there are multiple beta steps between the same terms, draw this as multiple arrows. We recommend underlining each beta redex, to keep track of what is going on.

4. Consider the following proposition of predicate logic:

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

Now answer the following three 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 type that corresponds to this proposition. For this, the context will have to contain λP variables for the existential quantifier and for the corresponding introduction rule. Therefore, this type will need to be well-typed in the context:

$$\begin{array}{rcl} D & : & * \\ \mathsf{ex} & : & (D \to *) \to * \\ \mathsf{ex\_intro} & : & \prod P : D \to *.\ \prod x : D.\,Px \to \mathsf{ex}\,P \\ p & : & D \to * \end{array}$$

You are allowed to use mathematical notation for this type, or to use Rocq syntax.

(Note that we just ask for the type as an expression, and *not* for a λP type derivation.)

(c) Give the λP proof term that corresponds to the proof that you gave in sub-question (a), using the same context.

(Note that we just ask for the proof term as an expression, and *not* for a λP type derivation.)

5. Consider the following four preterms:

$$\lambda a : *.\ a$$
$$\lambda a : *.\ *$$
$$\prod a : *.\ a$$
$$\prod a : *.\ *$$

Or in Rocq syntax:

```
fun a : Prop => a
fun a : Prop => Prop
forall a : Prop, a
forall a : Prop, Prop
```

Now answer the following four sub-questions:

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

(b) For the ones that are well-typed: give their type.

(c) Of the ones that are well-typed: which are types?

(d) Of the ones that are well-typed types: which are inhabited in an empty context? For the inhabited types, give an inhabitant.

6. (a) Give a type derivation in simple type theory of the judgment:

$$\vdash (\lambda x : a.\, x) : a \to a$$

(b) Give a type derivation in the pure type system $\lambda\!\to$ of the judgment:

$$a : * \vdash (\lambda x : a.\, x) : a \to a$$

For the rules of $\lambda\!\to$ see page 5.

7. Consider the following Rocq definition of a type for Booleans:

```
Inductive bool : Set :=
| true : bool
| false : bool.
```

We want to define a function `if_then_else` that chooses which argument to return based on a Boolean. It has type:

```
if_then_else
     : forall A : Set, bool -> A -> A -> A
```

Now answer the following three sub-questions:

(a) Define the function `if_then_else` using `Definition` and `match`.

(b) Give the type of the dependent recursion principle `bool_rec` of the type `bool`.

(c) Define the function `if_then_else` using an application of `bool_rec` to appropriate arguments.

8. We can also use an impredicative encoding in $\lambda 2$ of the Booleans, which is defined as:

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

Now answer the following two sub-questions:

(a) Give appropriate definitions of constants:

$$\begin{aligned} \mathsf{true}_2 &: \mathsf{bool}_2 \\ \mathsf{false}_2 &: \mathsf{bool}_2 \\ \mathsf{if\_then\_else}_2 &: \prod a : *.\, \mathsf{bool}_2 \to a \to a \to a \end{aligned}$$

(b) Give the two reductions that show that $\mathsf{if\_then\_else}_2$ behaves like an eliminator. For both, you should only give the start and end terms, and not all the intermediate beta steps.

9. Consider the proof of strong normalization of simply typed lambda calculus, in which the types are mapped to sets of untyped lambda terms.

   Now answer the following three sub-questions:

   (a) Give the definition of the set $[\![a \to a]\!]$, where a is a base type.

   (b) Show that this set has the property $[\![a \to a]\!] \neq \varnothing$.

   (c) Show that this set has the property $[\![a \to a]\!] \neq \mathsf{SN}$.

   (In these two last sub-questions, you may use the lemma and the proposition about the semantics from the lecture.)

Typing rules of $\lambda\!\rightarrow$

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

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

abstraction

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

product

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

conversion

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