# Type Theory and Coq 2017-2018

**02-07-2018**

Write your name and student number on each paper that you hand in. This test has 6 exercises, with 30 subexercises. Every subexercise is worth 3 points, the first 10 points are free, and the final mark is the number of points divided by 10. Write proofs, terms and types according to the conventions of Femke's course notes. Good luck!

1. This exercise is about *simply typed lambda calculus* and *propositional logic.*

   (a) Give the most general type of the untyped lambda term:

   $$\lambda xyz.\,(\lambda v.\,yx)(\lambda w.\,zw)$$

   (b) Write the term from the previous subexercise in Church-style typed lambda calculus, i.e., with explicit types in the lambda bindings, such that it has the type from your answer to the previous subexercise.

   (c) Give a proof of the propositional formula

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

   that contains a detour.

   (d) Give the normal form of this proof.

   (e) Give the proof term that correspond to the normalized proof from the previous subexercise.

   (f) Give the full type derivation in simply typed lambda calculus of the term from the previous subexercise.

   You may use abbreviations for contexts, if convenient.

   (g) Is it possible to have two *different* $\lambda{\to}$ terms $M_1$ and $M_2$, i.e., with $M_1 \neq_\alpha M_2$, that are both in normal form and that are convertible, i.e., with $M_1 =_\beta M_2$? If so, give an example of two such terms. If not, explain why this is not possible.

2. This exercise is about *dependent types* and *predicate logic.*

   (a) Give a proof in predicate logic of the formula:

   $$\forall x.\,((\forall y.\,\neg p(y)) \to \neg\forall y.\,p(y))$$

   In this formula we have used the abbreviation $\neg A := A \to \bot$.

(b) Give the proof term in $\lambda P$ that corresponds to this proof.

(c) Give the full context needed to type the term from the previous subexercise in $\lambda P$.

(d) Give two $\lambda P$ proof terms and their context for the predicate logic formula:

$$\forall x.\, p(x) \to p(x)$$

Give one proof term that uses the Curry-Howard isomorphism for $\lambda P$, and another proof term in which $\lambda P$ is used as a logical framework with context

$$\begin{array}{rcl}
\multicolumn{3}{l}{\Gamma_{\mathsf{pred}} :=} \\
\mathsf{prop} & : & * \\
\mathsf{proof} & : & \mathsf{prop} \to * \\
D & : & * \\
\mathsf{implies} & : & \mathsf{prop} \to \mathsf{prop} \to \mathsf{prop} \\
\mathsf{forall} & : & (D \to \mathsf{prop}) \to \mathsf{prop} \\
\mathsf{implies\_intro} & : & \Pi A : \mathsf{prop}.\, \Pi B : \mathsf{prop}.\, (\mathsf{proof}\ A \to \mathsf{proof}\ B) \to \mathsf{proof}\ (\mathsf{implies}\ A\ B) \\
\mathsf{implies\_elim} & : & \Pi A : \mathsf{prop}.\, \Pi B : \mathsf{prop}.\, \mathsf{proof}\ (\mathsf{implies}\ A\ B) \to \mathsf{proof}\ A \to \mathsf{proof}\ B \\
\mathsf{forall\_intro} & : & \Pi P : (D \to \mathsf{prop}).\, (\Pi x : D.\, \mathsf{proof}\ (P x)) \to \mathsf{proof}\ (\mathsf{forall}\ P) \\
\mathsf{forall\_elim} & : & \Pi P : (D \to \mathsf{prop}).\, \mathsf{proof}\ (\mathsf{forall}\ P) \to \Pi x : D.\, \mathsf{proof}\ (P x)
\end{array}$$

3. This exercise is about *polymorphism* and *second order propositional logic.*

   We define disjunction impredicatively in $\lambda\omega$:

$$\mathsf{or}_2 := \lambda a : *.\, \lambda b : *.\, \Pi c : *.\, (a \to c) \to (b \to c) \to c$$

(a) Give the type of $\mathsf{or}_2$ in $\lambda\omega$.

(b) Give the formula of second order propositonal logic that corresponds to the type

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

after expanding $\mathsf{or}_2$, where $a$ and $b$ are variables of type $*$.

(c) Give a proof in second order propositional logic of the formula from the previous subexercise.

(d) Give a $\lambda\omega$ term with type

$$\Pi a : *.\, \Pi b : *.\, \mathsf{or}_2\, a\, b \to \mathsf{or}_2\, b\, a$$

4. This exercise is about the typing rules of *pure type systems* and the *lambda cube.*

   (a) Give a full type derivation of

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

   in the calculus of constructions.

   See below for the lambda cube and the typing rules of the pure type systems from the lambda cube. If you want, you may first give sub-derivations first, instead of replicating them all the time.

   (b) In which of the systems of the lambda cube is this type derivation allowed?

5. This exercise is about *inductive types* and *recursive functions.*

   (a) Give the type of the dependent recursor nat_rec of the inductively defined natural numbers:

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

   (b) Use this recursor nat_rec to define the predecessor function

   $$\mathsf{pred}(n) := \begin{cases} 0 & \text{if } n = 0 \\ n - 1 & \text{if } n > 0 \end{cases}$$

   (c) Give the $\iota$-reduction rules for nat_rec.

   (d) Give another definition of the predecessor function, this time using `Fixpoint` and `match`.

   (e) We now want to model the *partial* predecessor function that is *undefined* on input 0 in the form of a relation pred_rel. Give an inductive Coq definition of this relation.

   (f) Give both the dependent and non-dependent induction principles that belong to the relation pred_rel from the previous exercise.

6. This exercise is about the *CPS translation.*

   The types of the two systems that we will study in this exercise are:

   $$A ::= a \mid \bot \mid A \to A$$

   where $\neg A := A \to \bot$ will be the type of continuations. These systems will have Curry-style typing, so the terms of the systems are untyped lambda terms.

   The terms, values and evaluation contexts differ between CBN and CBV versions of the system. For CBN these are:

   $$\begin{aligned} M &::= V \mid x \mid MM \mid \mathsf{callcc}\, M \mid \mathsf{throw}\, M \\ V &::= \lambda x.\, M \\ E &::= \Box \mid EM \end{aligned}$$

   For CBV these are:

   $$\begin{aligned} M &::= V \mid MM \mid \mathsf{callcc}\, M \mid \mathsf{throw}\, M \\ V &::= x \mid \lambda x.\, M \\ E &::= \Box \mid EM \mid VE \end{aligned}$$

   The reduction rules of the system are for CBN:

   $$\begin{aligned} E[(\lambda x.\, M)N] &\to E[M[x := N]] \\ E[\mathsf{callcc}\, M] &\to E[M(\lambda x.\, E[x])] \\ E[\mathsf{throw}\, M] &\to M \end{aligned}$$

   For CBV they are the same, but the beta rule is more restricted:

   $$\begin{aligned} E[(\lambda x.\, M)V] &\to E[M[x := V]] \\ E[\mathsf{callcc}\, M] &\to E[M(\lambda x.\, E[x])] \\ E[\mathsf{throw}\, M] &\to M \end{aligned}$$

   Now answer the following questions:

   (a) Give the reduction paths to normal form, both under CBN reduction and under CBV reduction, of

   $$\mathsf{callcc}\,(\lambda k.\,(\lambda x.\, \mathsf{true})(\mathsf{throw}\,(k\, \mathsf{false})))$$

   where $\mathsf{true} := \lambda xy.\, x$ and $\mathsf{false} := \lambda xy.\, y$ , In both reductions give for each reduction step explicitly the evaluation context.

(b) Give an untyped lambda term that is normalizing under CBN reduction, but not normalizing under CBV reduction.

(c) Give the types of $\mathsf{callcc}\, M$ and $\mathsf{throw}\, M$ in terms of the type of $M$.

(d) In this simply typed lambda calculus with $\mathsf{callcc}$ and $\mathsf{throw}$, is there a term with type $\neg\neg a \to a$? Explain your answer.

The CPS translation also differs between the CBN and CBV versions of the systems. We will use the notation $|V|$ for the translation of values, and $[\![M]\!]$ as the translation of arbitrary terms ('computations'). For CBN these are:

$$\begin{aligned}
[\![V]\!] &= \lambda k.\, k|V| \\
[\![x]\!] &= x \\
[\![M_1 M_2]\!] &= \lambda k.\, [\![M_1]\!](\lambda v_1.\, v_1 [\![M_2]\!] k)) \\
[\![\mathsf{callcc}\, M]\!] &= \lambda k. [\![M]\!](\lambda v.\, v(\lambda k'.\, k'(\lambda xy.xk))k) \\
[\![\mathsf{throw}\, M]\!] &= \lambda k. [\![M]\!](\lambda v.\, v) \\
|\lambda x.\, M| &= \lambda x.\, [\![M]\!]
\end{aligned}$$

For CBV these are:

$$\begin{aligned}
[\![V]\!] &= \lambda k.\, k|V| \\
[\![M_1 M_2]\!] &= \lambda k.\, [\![M_1]\!](\lambda v_1.\, [\![M_2]\!](\lambda v_2.\, v_1 v_2 k)) \\
[\![\mathsf{callcc}\, M]\!] &= \lambda k. [\![M]\!](\lambda v.\, v(\lambda xy.\, kx)k) \\
[\![\mathsf{throw}\, M]\!] &= \lambda k. [\![M]\!](\lambda v.\, v) \\
|x| &= x \\
|\lambda x.\, M| &= \lambda x.\, [\![M]\!]
\end{aligned}$$

The remainder of the exercise consists of the following questions:

(e) Give the CPS translation

$$[\![(\lambda x.\, x)(\lambda y.\, y)]\!]$$

for the CBN system, and normalize this term (where you also reduce under lambdas).

(f) We have type translations (with corresponding translations of the types in the context) that correspond to the term translations given above, i.e., such that if

$$\Gamma \vdash V : A$$

then

$$|\Gamma| \vdash |V| : |A|$$

(note that the term translation $|V|$ is different from the type translation $|A|$, despite the identical notation), and if

$$\Gamma \vdash M : A$$

then for CBN

$$[\![\Gamma]\!] \vdash [\![M]\!] : [\![A]\!]$$

and for CBV

$$|\Gamma| \vdash [\![M]\!] : [\![A]\!]$$

Give these type translations both for the CBN translation and for the CBV translation in the form

$$\begin{aligned} |a| &= \dots \\ |\bot| &= \dots \\ |A \to B| &= \dots \\ [\![A]\!] &= \dots \end{aligned}$$

(g) Why might the CPS translation be used in some compilers for functional languages?

# Typing rules of the lambda cube

In these rules the variables $s$, $s_1$, $s_2$ and $s_3$ range over the set of sorts $\{*, \Box\}$.

*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*

$$\frac{\Gamma \;\vdash\; A : s_1 \qquad \Gamma,\, x : A \;\vdash\; B : s_2}{\Gamma \;\vdash\; \Pi x : A.\, B : s_3} \quad \text{where } (s_1, s_2, s_3) \in \mathcal{R}$$

*conversion*

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

The sets of rules $\mathcal{R}$ for the eight systems of the lambda cube are:

| | |
|---|---|
| $\lambda{\to}$ | $\mathcal{R} = \{(*, *, *)\}$ |
| $\lambda P$ | $\mathcal{R} = \{(*, *, *), (*, *, \Box)\}$ |
| $\lambda 2$ | $\mathcal{R} = \{(*, *, *), (\Box, \Box, *)\}$ |
| $\lambda P2$ | $\mathcal{R} = \{(*, *, *), (*, *, \Box), (\Box, \Box, *)\}$ |
| $\lambda\underline{\omega}$ | $\mathcal{R} = \{(*, *, *), (\Box, \Box, \Box)\}$ |
| $\lambda P\underline{\omega}$ | $\mathcal{R} = \{(*, *, *), (*, *, \Box), (\Box, \Box, \Box)\}$ |
| $\lambda\omega$ | $\mathcal{R} = \{(*, *, *), (\Box, \Box, *), (\Box, \Box, \Box)\}$ |
| $\lambda P\omega = \lambda C$ | $\mathcal{R} = \{(*, *, *), (*, *, \Box), (\Box, \Box, *), (\Box, \Box, \Box)\}$ |