Type Theory and Coq 2014

26-01-2015

Write your name on each paper that you hand in. Each subexercise is worth 3 points, 10 points are free, and the final mark is the number of points divided by 10. Write proofs, terms and types in this test according to the conventions of Femke's course notes. Good luck!

1. Consider the term of the untyped lambda calculus:

$$(\lambda x.x)(\lambda y.\,(\lambda z.z)\,y\,(\lambda w.w))$$

   (a) Give the normal form of this term.

   (b) Give a most general type of this term, where the term is taken to be a term of Curry-style simple type theory. (You do not need to explain how you obtained this type, nor why it is a most general type.)

   (c) Give the term of Church-style simple type theory that corresponds to the untyped lambda term and that has the type from part (b).

   (d) Give a type of this term in Curry-style simple type theory that is *not* a most general type.

2. Consider the formula of first order propositional logic:

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

   (a) Give a proof in first order propositional logic of this formula. (Write all the names of the proof rules in the proof tree.)

   (b) Give the proof term of Church-style simple type theory of this proof.

   (c) Give the type judgment for the term from part (b).

   (d) Give a derivation of the type judgment from part (c). (You do not need to give names for the typing rules in the derivation tree, and you may use abbreviations for contexts.)

3. Consider the formula of first order predicate logic:

$$(\forall x.\forall y.R(x,y)) \to (\forall x.\forall y.R(y,x))$$

   (a) Give a proof in first order predicate logic of this formula. (Write all the names of the proof rules in the proof tree.)

   (b) Which of the rules in this proof has a variable condition, what is this condition, and why is it satisfied?

   (c) Give the type of λP that corresponds to the formula.

   (d) Give a λP proof term for the type from part (c).

4. Consider the term of λC:

$$\mathsf{or}_2 \quad :\equiv \quad \lambda A : *.\, \lambda B : *.\, \Pi C : *.\, (A \to C) \to (B \to C) \to C$$

   (a) Give the type of or_2 in λC. (See page 5 for the typing rules of λC, in case you need those.)

   (b) Is or_2 also typeable in $\lambda 2$? Explain your answer.

   (c) Give a term of λC that inhabits the following λC type:

$$\Pi A : *.\, \Pi B : *.\, A \to \mathsf{or}_2\, A\, B$$

   (d) Give a term of λC that inhabits the following λC type:

$$\Pi A : *.\, \Pi B : *.\, \Pi C : *.\, \mathsf{or}_2\, A\, B \to (A \to C) \to (B \to C) \to C$$

5. Consider the λC type $a \to a$ in the context $a : *$.

   (a) Give the λC typing judgment (without a derivation) that gives the *kind* of this type.

   (b) Give a derivation in λC of the judgment from part (a). (See page 5 for the typing rules of λC. You do not need to give names for the typing rules in the derivation tree.)

   (c) Give also an *inhabitant* in λC of this type.

   (d) Give the λC typing judgment (without a derivation) for this inhabitant.

6. Consider the Coq inductive type for binary positive numbers:

```
Inductive positive : Set :=
| xH : positive
| xO : positive -> positive
| xI : positive -> positive.
```

In this representation of binary numbers, `xH` stands for the number 1, `xO` stands for the function $\lambda n.2n$ (which adds a zero at the end of the number), and `xI` stands for the functin $\lambda n.2n + 1$ (which adds a one).

(a) Give a Coq term that represents the binary form of the decimal number 18.

(b) Give the type of the dependent induction principle `positive_rect` for this inductive type. (You can write this induction principle using Coq syntax or using PTS syntax, whatever is your preference.)

(c) Give the type of the corresponding non-dependent induction principle `positive_rect_nondep` for this inductive type.

(d) Write a function `succ : positive -> positive` that adds one to its argument using a combination of `Fixpoint` and `match`.

For instance

```
succ (xI xH) = xO (xO xH)
```

should hold, because `succ`$(3) = 4$.

(e) Now also write the `succ` function using the non-dependent induction priciple used as a primitive recursor.

7. We define the extended (untyped) lambda terms and the subclass called *values* by:

$$t ::= x \mid t_1 t_2 \mid v$$
$$v ::= \lambda x.t \mid [\tilde{x}\, v_1 \ldots v_n]$$

On these terms we define weak reduction by the three rules:

$$(\lambda x.t)v \to_v t[x := v] \qquad (\beta_v)$$
$$[\tilde{x}\, v_1 \ldots v_n]v \to_v [\tilde{x}\, v_1 \ldots v_n\, v] \qquad (\beta_s)$$
$$E_v(t) \to_v E_v(t') \quad \text{if } t \to_v t' \qquad (\text{context}_v)$$

where the one step contexts are defined as:

$$E_v(\Box) ::= \Box\, v \mid t\, \Box$$

We write $\mathcal{V}(t)$ for the unique normal form of t under $\to_v$ reduction (if it exists), the function $\mathcal{V}$ is a *partial* function. Using this we define functions $\mathcal{R}(t)$ and $\mathcal{N}(v)$ recursively by:

$$\mathcal{N}(t) = \mathcal{R}(\mathcal{V}(t)) \tag{1}$$

$$\mathcal{R}(\lambda x.t) = \lambda y.\, \mathcal{N}((\lambda x.t)\, [\tilde{y}]) \quad (y \text{ fresh}) \tag{2}$$

$$\mathcal{R}([\tilde{x}\, v_1 \dots v_n]) = x\, \mathcal{R}(v_1) \dots \mathcal{R}(v_n) \tag{3}$$

Finally we define the term t_7 by

$$t_7 := (\lambda x.x)(\lambda y.\, (\lambda z.z)\, y\, (\lambda w.w))$$

(a) Does there exist an extended lambda term that has two different one step $\to_v$ reductions? If so, give an example.

(b) Is there an extended lambda term t for which $\mathcal{V}(t)$ does not exist (because the reduction does not terminate)? If so, give an example.

(c) Is there an extended lambda term t that is typable in Curry-style simple type theory (and therefore does not contain subterms of the shape $[\tilde{x}\, v_1 \dots v_n]$) for which $\mathcal{V}(t)$ does not exist? If so, give an example.

(d) Show how the value $\mathcal{V}(t_7)$ is calculated, and give the result.
Write down all the $\to_v$ reduction steps that are used in this calculation.

(e) Show how the value $\mathcal{N}(t_7)$ is calculated, and give the result.
Whenever in this calculation you calculate a weak normal form $\mathcal{V}(t)$, write down all the $\to_v$ reduction steps like in part (d).

In these rules the variables s, s_1 and s_2 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

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma,\, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

conversion

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