Type Theory and Coq 2012

23-01-2013

1. (a) Prove the formula

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

in minimal propositional logic. Indicate whether the proof has any detours.

$$
\begin{prooftree}
\AxiomC{$[a \to a \to c^x]$}
\AxiomC{$[a^w]$}
\RightLabel{$E{\to}$}
\BinaryInfC{$a \to c$}
\AxiomC{$[a^w]$}
\RightLabel{$E{\to}$}
\BinaryInfC{c}
\RightLabel{$I[w]{\to}$}
\UnaryInfC{$a \to c$}
\AxiomC{$[b \to a^y]$}
\AxiomC{$[b^z]$}
\BinaryInfC{a}
\RightLabel{$E{\to}$}
\BinaryInfC{c}
\RightLabel{$I[z]{\to}$}
\UnaryInfC{$b \to c$}
\RightLabel{$I[y]{\to}$}
\UnaryInfC{$(b \to a) \to b \to c$}
\RightLabel{$I[x]{\to}$}
\UnaryInfC{$(a \to a \to c) \to (b \to a) \to b \to c$}
\end{prooftree}
$$

This proof has one detour, the $E{\to}$ elimination right after the the $I[w]{\to}$ introduction.

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

$$\lambda x : a{\to}a{\to}c.\, \lambda y : b{\to}a.\, \lambda z : b.\, (\lambda w : a.\, xww)(yz)$$

2. (a) Prove the formula

$$a \to \forall b.\, (\forall c.\ a \to c) \to b$$

in second order propositional logic.

$$
\begin{prooftree}
\AxiomC{$[\forall c.\ a \to c^{H_2}]$}
\RightLabel{$E\forall$}
\UnaryInfC{$a \to b$}
\AxiomC{$[a^{H_1}]$}
\RightLabel{$E{\to}$}
\BinaryInfC{b}
\RightLabel{$I[H_2]{\to}$}
\UnaryInfC{$(\forall c.\ a \to c) \to b$}
\RightLabel{$I\forall$}
\UnaryInfC{$\forall b.\, (\forall c.\ a \to c) \to b$}
\RightLabel{$I[H_1]{\to}$}
\UnaryInfC{$a \to \forall b.\, (\forall c.\ a \to c) \to b$}
\end{prooftree}
$$

(b) Give the lambda term of $\lambda 2$ that corresponds to this proof, and give its type.

$$\begin{array}{c} \lambda H_1 : a.\, \lambda b : *.\, \lambda H_2 : (\Pi c : *.\, a \to c).\, H_2\, b\, H_1 \\ : \\ a \to \Pi b : *.\, (\Pi c : *.\, a \to c) \to b \end{array}$$

3. The rules for the eight systems from the Barendregt cube are given by the following table:

$\lambda{\to}$	$\mathcal{R} = \{(*, *)\}$
λP	$\mathcal{R} = \{(*, *), (*, \Box)\}$
$\lambda 2$	$\mathcal{R} = \{(*, *), (\Box, *)\}$
$\lambda P2$	$\mathcal{R} = \{(*, *), (*, \Box), (\Box, *)\}$
$\lambda\underline{\omega}$	$\mathcal{R} = \{(*, *), (\Box, \Box)\}$
$\lambda P\underline{\omega}$	$\mathcal{R} = \{(*, *), (*, \Box), (\Box, \Box)\}$
$\lambda\omega$	$\mathcal{R} = \{(*, *), (\Box, *), (\Box, \Box)\}$
λC	$\mathcal{R} = \{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}$

in which (s_1, s_2) is an abbreviation of (s_1, s_2, s_2).

Furthermore, the PTS product and abstraction rules are:

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

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

Finally we have the typings:

$$\begin{array}{lcl} \mathsf{nat} & : & * \\ \mathsf{vec} & : & \mathsf{nat} \to * \end{array}$$

For each of the following three terms, list in which of the systems from the Barendregt cube the term is typable:

(a)

$$\mathsf{nat} \to \mathsf{nat}$$

All eight systems.

(b)

$$\lambda a : *.\ a \to a$$

The systems that extend $\lambda\underline{\omega}$, i.e.: $\lambda\underline{\omega}$, $\lambda P\underline{\omega}$, $\lambda\omega$, λC.

The type of this term is $* \to *$ and to have that type one needs the rule $(\Box, \Box)$.

(c)

$$\Pi n : \mathsf{nat}.\ \mathsf{vec}\ n$$

The systems that extend λP, i.e.: λP, $\lambda P2$, $\lambda P\underline{\omega}$, λC.

This product type itself only needs the rule $(*, *)$, but to type vec one also needs $(*, \Box)$.

4. (a) Consider the Coq definition

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

Give the *dependent* induction principle `nat_ind` of this type.

```
nat_ind :
  forall P : nat -> Prop,
    P O -> (forall n : nat, P n -> P (S n)) ->
    forall n : nat, P n
```

(b) Give the normal form of the term

```
nat_ind P c f (S (S O))
```

that uses the principle from the previous exercise. In this term the variables `P`, `c`, `f` and `n` are variables from the context.

$$\texttt{nat\_ind P c f (S (S O))} \to^* \texttt{f (S O) (f O c)}$$

(c) Give the *non-dependent* induction principle that corresponds to the induction principle from 4(a).

```
nat_ind :
  forall P : Prop,
    P -> (nat -> P -> P) ->
    nat -> P
```

5. (a) Consider the Coq definition

```
Inductive le (n : nat) : nat -> Prop :=
| le_n : le n n
| le_S : forall m : nat, le n m -> le n (S m).
```

Give the *non-dependent* induction principle `le_ind` of this type. (Hint: first determine the *dependent* induction principle, and then remove the dependence on the elements of `le n m` in the predicate.)

The dependent induction principle would have been:

```
le_ind :
  forall (n : nat)
    (P : forall m : nat, le n m -> Prop),
  P n (le_n n) ->
  (forall (m : nat) (H : le n m),
    P m H -> P (S m) (le_S n m H)) ->
  forall (m : nat) (H : le n m), P m H
```

But the induction principle in Coq is non-dependent, and therefore it is:

```
le_ind :
  forall (n : nat) (P : nat -> Prop),
  P n ->
  (forall m : nat, le n m -> P m -> P (S m)) ->
  forall m : nat, le n m -> P m
```

Note that this very much resembles the *dependent* induction principle for `nat`, but then for the natural numbers $\geq n$.

(b) Prove that $1 \leq 2$, i.e., give an inhabitant of

```
le (S O) (S (S O))
```

where `le` is the type from the previous exercise.

```
le_S (S O) (S O) (le_n (S O))
```

6. Which of the following four inductive definitions are allowed by Coq? For the definitions that are not allowed, explain what requirement is not satisfied.

   (a)
   ```
   Inductive T1 : Type :=
   | b1 : T1
   | c1 : (T1 -> T1) -> T1.
   ```
   Not allowed: the first `T1` in the type of `c1` does not occur positively.

   (b)
   ```
   Inductive T2 (A : Type) : Type :=
   | b2 : T2 A
   | c2 : T2 (A -> A) -> T2 A.
   ```
   Allowed.

   (c)
   ```
   Inductive T3 (A : Type) : Type :=
   | b3 : T3 A
   | c3 : T3 A -> T3 (A -> A).
   ```
   Not allowed: the parameter in the type of `c3` has to match the parameter in the definition.

   (d)
   ```
   Inductive T4 : Type :=
   | b4 : T4
   | c4 : (nat -> T4) -> T4.
   ```
   Allowed.

7. We recursively define an operation M^* on untyped lambda terms:

$$\begin{array}{rcll} x^* & := & x & \\ (\lambda x.M)^* & := & \lambda x.M^* & \\ ((\lambda x.M)N)^* & := & M^*[x := N^*] & \\ (MN)^* & := & M^* N^* & \qquad \text{when } MN \text{ is not a beta redex} \end{array}$$

and we inductively define a relation $M \Rightarrow N$ on untyped lambda terms:

$$x \Rightarrow x$$

$$\frac{M \Rightarrow M'}{\lambda x.M \Rightarrow \lambda x.M'}$$

$$\frac{M \Rightarrow M' \quad N \Rightarrow N'}{MN \Rightarrow M'N'}$$

$$\frac{M \Rightarrow M' \quad N \Rightarrow N'}{(\lambda x.M)N \Rightarrow M'[x := N']}$$

(a) State the diamond property for this relation $M \Rightarrow N$.

If $M \Rightarrow M_1$ and $M \Rightarrow M_2$ then there exists a term N such that $M_1 \Rightarrow N$ and $M_2 \Rightarrow N$.

(b) What is the relation between the M^* operation and the $M \Rightarrow N$ relation that allows one to prove this property? (Note that the exercise does *not* ask you to prove that this relation holds.)

If $M \Rightarrow N$ then $N \Rightarrow M^*$.