Type Theory and Coq 2021-2022

19-11-2021

1. Give a proof in intuitionistic propositional logic of the formula

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

$$
\dfrac{
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a \wedge (b \vee c)^x]}{a}\,El\wedge \qquad [b^y]}{a \wedge b}\,I\wedge}{(a \wedge b) \vee (a \wedge c)}\,Il\vee}{b \to (a \wedge b) \vee (a \wedge c)}\,I[y]\!\to
\quad
\dfrac{[a \wedge (b \vee c)^x]}{b \vee c}\,Er\wedge
\quad
\dfrac{\dfrac{\dfrac{\dfrac{[a \wedge (b \vee c)^x]}{a}\,El\wedge \qquad [c^z]}{a \wedge c}\,I\wedge}{(a \wedge b) \vee (a \wedge c)}\,Ir\vee}{c \to (a \wedge b) \vee (a \wedge c)}\,I[z]\!\to
}{(a \wedge b) \vee (a \wedge c)}\,E\vee}{a \wedge (b \vee c) \to (a \wedge b) \vee (a \wedge c)}\,I[x]\!\to
$$

2. Replace the question marks in the following type judgment with simple types, such that it becomes derivable in simple type theory, and give the full type derivation:

$$x : ?\,,\ y : ?\ \vdash (\lambda z : ?.\, xyz) : ?$$

Define

$$\Gamma := x : a \to b \to c\,,\ y : a\,, z : b$$

The derivation then is:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : a \to b \to c} \quad \dfrac{}{\Gamma \vdash y : a}}{\Gamma \vdash xy : b \to c} \qquad \dfrac{}{\Gamma \vdash z : b}}{\Gamma \vdash xyz : c}}{x : a \to b \to c\,,\ y : a\ \vdash (\lambda z : b.\, xyz) : b \to c}
$$

3. Consider the following two inductive types:

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

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 Coq term using these definitions that corresponds to the statement $2 \leq 4$, and give a Coq term that proves this.

```
le_S 2 3 (le_S 2 2 (le_n 2))
     : le 2 4
```

Or, if we write out the natural numbers:

```
le_S (S (S O)) (S (S (S O)))
  (le_S (S (S O)) (S (S O)) (le_n (S (S O))))
     : le (S (S O)) (S (S (S (S O))))
```

4. Consider the proof term

$$\lambda x : D.\, (\lambda y : D.\, \lambda H : py.\, H)\, x$$

for the formula of predicate logic

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

Give the natural deduction proof that corresponds to this term.

Does this proof contain a *detour*? If so, explain why, what connective the detour is for, and normalize this proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{[p(y)^H]}{p(y) \to p(y)}\, I[H]{\to}}{\forall y.\, p(y) \to p(y)}\, I\forall}{p(x) \to p(x)}\, E\forall}{\forall x.\, p(x) \to p(x)}\, I\forall$$

A detour is an elimination rule that directly follows the corresponding introduction rule. In this case the detour is for the universal quantifier, the $E\forall$ rule that follows the $I\forall$ rule:

$$\begin{array}{c} \vdots \\ \dfrac{\dfrac{p(y) \to p(y)}{\forall y.\, p(y) \to p(y)}\, I\forall}{p(x) \to p(x)}\, E\forall \\ \vdots \end{array}$$

The proof in normal form corresponds to the normal form of the proof term:

$$\lambda x : D.\, \lambda H : px.\, H$$

and is:

$$\dfrac{\dfrac{[p(x)^H]}{p(x) \to p(x)}\, I[H]{\to}}{\forall x.\, p(x) \to p(x)}\, I\forall$$

5. Give a derivation in $\lambda 2$ of the type judgment

$$a : * \vdash (* \to a) : *$$

For the rules of $\lambda 2$ see page 6.

$$\dfrac{\dfrac{\dfrac{}{\vdash * : \square} \qquad \dfrac{}{\vdash * : \square}}{a : * \vdash * : \square} \qquad \dfrac{\dfrac{\dfrac{}{\vdash * : \square}}{a : * \vdash a : *} \qquad \dfrac{\dfrac{}{\vdash * : \square} \qquad \dfrac{}{\vdash * : \square}}{a : * \vdash * : \square}}{a : *,\, b : * \vdash a : *}}{a : * \vdash (* \to a) : *}$$

6. We define in minimal second order propositional logic:

$$\begin{aligned} \mathsf{and}_2\, A\, B &:= \forall c.\, ((A \to B \to c) \to c) \\ \mathsf{or}_2\, A\, B &:= \forall c.\, ((A \to c) \to (B \to c) \to c) \end{aligned}$$

Give a proof in this logic of the formula

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

$$\dfrac{\dfrac{\dfrac{\dfrac{[a \to c^y] \qquad \dfrac{\dfrac{[\forall c.\, ((a \to b \to c) \to c)^x]}{(a \to b \to a) \to a}\, E\forall \qquad \dfrac{\dfrac{[a^v]}{b \to a}\, I[w]{\to}}{a \to b \to a}\, I[v]{\to}}{a}\, E{\to}}{c}\, E{\to}}{(b \to c) \to c}\, I[z]{\to}}{(a \to c) \to (b \to c) \to c}\, I[y]{\to}$$

$$\dfrac{\dfrac{(a \to c) \to (b \to c) \to c}{\forall c.\, ((a \to c) \to (b \to c) \to c)}\, I\forall}{\mathsf{and}_2\, a\, b \to \mathsf{or}_2\, a\, b}\, I[x]{\to}$$

7. Give the polymorphic identity function and its type as a term and type of $\lambda 2$.

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

Or, in a different notation:

$$(\Lambda a.\, \lambda x : a.\, x) \;:\; (\forall a.\, a \to a)$$

8. Consider the inductive type for the Cartesian product $A \times B$:

```
Inductive prod (A B : Set) : Set :=  pair : A -> B -> prod A B.
```

Coq defines the *dependent* recursion principle for this type automatically, with type:

```
prod_rec
     : forall (A B : Set) (P : prod A B -> Set),
       (forall (a : A) (b : B), P (pair a b)) ->
       forall p : prod A B, P p
```

The exercise is to give the type of the *non-dependent* recursion principle `prod_rec_nondep`.

```
prod_rec_nondep
     : forall (A B P : Set), (A -> B -> P) -> prod A B -> P
```

9. Consider the lambda term:

$$\lambda xyz.\, xy(zyx)$$

Use the algorithm PT to find whether this term is typable, and if so, give the principal type and show how the algorithm computed it.

Assign type variables to all variables and to all applicative subterms:

$$\lambda x^a.\, \lambda y^b.\, \lambda z^c.\, \underbrace{\underbrace{x^a\, y^b}_{e}\, \overbrace{(\overbrace{z^c\, y^b}^{g}\, x^a)}^{f}}_{d}$$

The applicative subterms then give the following equations:

$$\begin{aligned} a &= b \to e \\ e &= f \to d \\ c &= b \to g \\ g &= a \to f \end{aligned}$$

Solving this allows us to express all types in terms of b, d and f:

$$\begin{aligned} a &= b \to f \to d \\ c &= b \to (b \to f \to d) \to f \\ e &= f \to d \\ g &= (b \to f \to d) \to f \end{aligned}$$

The type of the annotated term is:

$$a \to b \to c \to d$$

which expands to the principal type:

$$(b \to f \to d) \to b \to (b \to (b \to f \to d) \to f) \to d$$

10. One of the cases in the definition of the function $\mathsf{Type}_{\_}(-)$ is:

$$\mathsf{Type}_\Gamma(MN) \quad = \quad \begin{cases} \text{if} & \mathsf{Type}_\Gamma(M) = C \text{ and } \mathsf{Type}_\Gamma(N) = D \\ \text{then} & \text{if } C \twoheadrightarrow_\beta \Pi x : A.\, B \text{ and } A =_\beta D \\ & \text{then } B[x := N] \text{ else 'false'} \\ \text{else} & \text{'false'} \end{cases}$$

Compute

$$\mathsf{Type}_{\Gamma_{\mathsf{zeroes}}}(\mathsf{zeroes}\ \mathsf{O})$$

where

$$\Gamma_{\mathsf{zeroes}} := \mathsf{nat} : *,\ \mathsf{O} : \mathsf{nat},\ \mathsf{vec} : \mathsf{nat} \to *,\ \mathsf{zeroes} : (\Pi n : \mathsf{nat}.\, \mathsf{vec}\ n)$$

and explain what M, N, x, A, B, C and D are in the above definition when working out this specific type.

$$\mathsf{Type}_{\Gamma_{\mathsf{zeroes}}}(\mathsf{zeroes}\ \mathsf{O}) : \mathsf{vec}\ \mathsf{O}$$

The instances in the definition for this case are:

$$\begin{aligned} M &= \mathsf{zeroes} \\ N &= \mathsf{O} \\ x &= n \\ A &= \mathsf{nat} \\ B &= \mathsf{vec}\ n \\ C &= \Pi n : \mathsf{nat}.\, \mathsf{vec}\ n \\ D &= \mathsf{nat} \end{aligned}$$

11. In the proof of $\mathsf{CR}(\beta)$ for the untyped lambda calculus, the notion of *parallel reduction* is defined, which is written as $M \Longrightarrow P$.

We denote the reflexive transitive closure of $\Longrightarrow$ by $\Longrightarrow^*$.

Now give all inclusions that hold between the four relations $\to_\beta$, $\twoheadrightarrow_\beta$ (which is the same as $\to_\beta^*$), $\Longrightarrow$ and $\Longrightarrow^*$. You do not need to explain why this is the case.

As an example of what we mean with 'inclusions': the inclusion $\twoheadrightarrow_\beta \subseteq \to_\beta$ does *not* hold, because $x \twoheadrightarrow_\beta x$, but not $x \to_\beta x$.

$$\to_\beta \subset \Longrightarrow \subset \twoheadrightarrow_\beta = \Longrightarrow^*$$

12. In the proof of SN for $\lambda{\to}$, a model for $\lambda{\to}$ is defined consisting of *saturated sets* of untyped lambda terms. The definition is:

$$\begin{aligned} [\![a]\!] &:= \mathsf{SN} && \text{for all base types } a \\ [\![A \to B]\!] &:= \{M \mid \forall N \in [\![A]\!].\, MN \in [\![B]\!]\} \end{aligned}$$

Now for a given type A, consider the four sets:

$$
\begin{aligned}
&[\![A]\!] \\
&\mathsf{SN} \\
\mathcal{T}_0(A) &:= \{M \mid \vdash M : A\} \\
\mathcal{T}_1(A) &:= \{M \mid \exists\Gamma.\, \Gamma \vdash M : A\}
\end{aligned}
$$

In the definitions of the sets $\mathcal{T}_0(A)$ and $\mathcal{T}_1(A)$, the typing is in Curry-style simple type theory, so these sets consist of untyped lambda terms that are typable with type A.

Now give all inclusions that hold between these four sets for all types A. You do not need to explain why these inclusions hold.

However, for all inclusions that are not equalities for all types A, you should give such a type A and an (untyped) term that shows this.

$$\mathcal{T}_0(A) \subset \mathcal{T}_1(A) \subset [\![A]\!] \subseteq \mathsf{SN}$$

None of these inclusions are equalities for all types A (in fact, as we suggested in our notation, one can prove that two of them are *never* equalities; to show this was not required in the exercise though). This can be shown with the following examples:

$x \in \mathcal{T}_1(a)$, because we have $x : a \vdash x : a$.
$x \notin \mathcal{T}_0(a)$, because terms in $\mathcal{T}_0(a)$ cannot have free variables.

$(\lambda x.\, x) \in [\![a]\!] = \mathsf{SN}$.
$(\lambda x.\, x) \notin \mathcal{T}_1(a)$, as $(\lambda x.\, x)$ has to be typed with a function type.

$(\lambda x.\, xx) \in \mathsf{SN}$.
$(\lambda x.\, xx) \notin [\![a \to a]\!]$, as $(\lambda x.\, xx) \in [\![a]\!] = \mathsf{SN}$ but $(\lambda x.\, xx)(\lambda x.\, xx) \notin [\![a]\!] = \mathsf{SN}$.