Type theory and proof assistants
21 January 2009, 15.30–17.30

This test has 15 exercises, and each exercise is worth 6 points. The first 10 points are free, and the final mark is the number of points divided by ten.

Good luck!

1. Give the term in simply typed lambda calculus that under the Curry-Howard isomorphism corresponds to a proof of the formula of minimal propositional logic
$$(a \to b \to c) \to b \to a \to c$$

2. Give a type derivation of the type judgment for the term that is your answer to the previous exercise.

3. Give a proof of minimal propositional logic that contains a detour and give the corresponding proof after detour elimination. What reduction in the simply typed lambda calculus corresponds to this detour elimination?

4. Someone wants to have a definition of lists with

   ```
   list : Set -> nat -> Set
   ```

   so that

   ```
   list A n
   ```

   is the type of lists with elements in `A` that have length `n`. Furthermore the same someone wants to be able to write

   ```
   nil A
   ```

   for the empty list in `list A 0`. What will be the type of the `nil` function, or in other words, what should be in the place of the question mark in

   ```
   nil : ?
   ```

   Write this type both in mathematical notation (with $*$, $\Box$, λ, Π, $\to$) as well as in Coq notation (with `Set`, `Type`, `fun`, `forall`, `->`).

5. Someone defines a type of binary trees in Coq as

   ```
   Inductive tree : Set :=
   | leaf : tree
   | node : tree -> tree -> tree.
   ```

Give the induction principle that belongs to this type (in Coq notation).

6. Give a derivation of the judgment

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

in the lambda calculus with dependent types λP. For the typing rules of λP, see page 4.

7. Give the term in the polymorphic lambda calculus that corresponds under the Curry-Howard isomorphism to a proof of the formula of minimal second order propositional logic

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

Give also the type of this term.

8. What is the impredicative definition in the polymorphic lambda calculus of the type of natural numbers consisting of the Church numerals? What is a term in this type that corresponds to the number 2?

9. Which of the following type judgments belong to $\lambda{\to}$, which belong to λP and which belong to $\lambda 2$?

$$\begin{array}{lcl} A : * & \vdash & A : * \\ A : * & \vdash & (\Pi x : A.\, A) : * \\ A : * & \vdash & (\Pi x : *.\, A) : * \\ A : * & \vdash & * : \Box \\ A : * & \vdash & (\Pi x : A.\, *) : \Box \\ A : * & \vdash & (\Pi x : *.\, *) : \Box \end{array}$$

10. Give the Coq definition of an inductive predicate of type

```
even : nat -> Prop
```

that says whether a natural number is even or not. (The constructors of the type `nat` are called `O` and `S`.)

11. Give the order-preserving function Φ_{even} on the complete lattice on the powerset of the natural numbers that corresponds to the inductive predicate from the previous exercise. Explain what it means that Φ_{even} is order-preserving. What is the least fixpoint of Φ_{even}?

12. Give an example of a complete lattice L and a map $\Phi : L \to L$ which is *not* order-preserving, and such that Φ does *not* have a fixpoint. What is the set H of pre-fixpoints of this map, and what is $\bigvee_L H$?

13. What are the type checking, type synthesis and type inhabitation problems? Which of these problems is decidable for λP?

14. A substitution lemma might read

$$\Gamma, x : B \vdash M : A \quad \text{and} \quad \Gamma \vdash P : B \quad \text{then} \quad \Gamma \vdash M[x := P] : A[x := P]$$

Explain why for a pure type system this cannot be proved with a straight-forward induction on the derivation of $\Gamma, x : B \vdash M : A$. Furthermore give a more general version of this lemma that *can* be proved with a straight-forward induction.

15. We define for each type A of the simply typed lambda calculus a set of untyped lambda terms $[\![A]\!]$ by

$$\begin{array}{rcl} [\![a]\!] & := & \mathsf{SN} \\ [\![A \to B]\!] & := & \{M \mid \forall N \in [\![A]\!].\, MN \in [\![B]\!]\} \end{array}$$

where SN is the set of strongly normalizing lambda terms.

Prove by induction on the structure of A that from $M[x := N]\vec{P} \in [\![A]\!]$ and $N \in \mathsf{SN}$ it follows that $(\lambda x.\, M)\, N\vec{P} \in [\![A]\!]$.

Derivation rules of the Pure Type Systems λP and $\lambda 2$

In these rules the variable s ranges over the set of sorts $\{*, \Box\}$. The product rule differs between λP and $\lambda 2$.

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 (λP)

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

product ($\lambda 2$)

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

conversion

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