Tentamen Toegepaste Logica

20 april 2001

Answers may be given in Dutch or English. Good luck!

Exercise 1. This exercise is concerned with first-order minimal propositional logic and $\lambda^{\rightarrow}$ (simply typed λ-calculus).

a. Show that the formula $(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B$ is a tautology of first-order minimal propositional logic.

   (5 points)

b. Give the (formal) type derivation in simply typed λ-calculus corresponding to the proof of 1a.

   (5 points)

c. Give a proof of $A \rightarrow (B \rightarrow B) \rightarrow A$ with a detour, and with all assumptions cancelled.

   (5 points)

d. Give the typing derivation corresponding to the proof of 1c, and reduce the term to β-normal form.

   (5 points)

Exercise 2. This exercise is concerned with first-order minimal predicate logic and λP (λ-calculus with dependent types).

a. Show that the formula $(\forall x.(P(x) \rightarrow Q(x))) \rightarrow (\forall y.P(y)) \rightarrow \forall z.Q(z)$ is a tautology of first-order minimal predicate logic.

   (5 points)

b. Give the typing derivation in λP that corresponds exactly to the proof of 2a.

   (5 points)

c. Explain the encoding of algebraic terms of first-order minimal predicate logic in λP in detail.

   (5 points)

d. The application rule of λP (in a slightly informal form) is

$$\frac{F : \Pi x{:}A.\, B \qquad M : A}{F\, M : B[x := M]}$$

Explain the correspondence between this rule and (two) rules of first-order predicate logic according to the Curry-Howard-De Bruijn isomorphism.

(5 points)

Exercise 3. This exercise is concerned with second-order minimal propositional logic and $\lambda 2$ (λ-calculus with polymorphic types).

a. Suppose we have a constructor Polylist such that $\mathsf{Polylist}\, a$ is the type of finite lists of terms of type a, for every a.

For finite lists of natural numbers we have a function

$$\mathsf{natmap} : (\mathsf{Nat} \to \mathsf{Nat}) \to \mathsf{Natlist} \to \mathsf{Natlist}$$

with specification

$$\begin{array}{lcl} \mathsf{natmap}\, f\, \mathsf{nil} & = & \mathsf{nil} \\ \mathsf{natmap}\, f\, (\mathsf{cons}\, h\, t) & = & \mathsf{cons}\, (f\, h)\, (\mathsf{natmap}\, f\, t) \end{array}$$

What is the type of a polymorphic version polymap of natmap?

(5 points)

b. Explain the encoding of formulas of second-order minimal propositional logic in $\lambda 2$ in detail.

(5 points)

c. Give the schematic form of the detours in second-order minimal propositional logic.

(5 points)

d. Consider the two product rules of $\lambda 2$ from its formal typing system:

$$\frac{\Gamma \vdash A : \star \qquad \Gamma, x : A \vdash B : \star}{\Gamma \vdash \Pi x{:}A.\, B : \star} \qquad \frac{\Gamma \vdash A : \Box \qquad \Gamma, x : A \vdash B : \star}{\Gamma \vdash \Pi x{:}A.\, B : \star}$$

Do for each rule the following: explain briefly what it means and illustrate its use by means of an example.

(5 points)

Exercise 4. This exercise is concerned with Gödel's **T**. Besides the rule for β-reduction we also use the rules for the recursor for natural numbers:

$$\begin{array}{lll} \mathsf{r}_{\mathsf{Nat}}(N, F, \mathsf{z}) & \to & N \\ \mathsf{r}_{\mathsf{Nat}}(N, F, \mathsf{s}(P)) & \to & F\, \mathsf{r}_{\mathsf{Nat}}(N, F, P)\, P \end{array}$$

Here $N : \mathsf{Nat}$, $F : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$, and $P : \mathsf{Nat}$. We suppose we have defined a term plus that implements addition of two natural numbers. We want to define a term mul implementing multiplication of two natural numbers.

a. Formulate the two requirements on the term mul (Use plus and use recursion in the first argument of mul.)

   (5 points)

b. Show that the term

   $$\mathsf{mul} = \lambda m{:}\mathsf{Nat}.\, \lambda n{:}\mathsf{Nat}.\, \mathsf{r}_{\mathsf{Nat}}(\mathsf{z}, \lambda x{:}\mathsf{Nat}.\, \lambda y{:}\mathsf{Nat}.\, \mathsf{plus}\ x\ n, m)$$

   satisfies the specification given in 4a. (Give the reduction in detail.)

   (5 points)

Exercise 5.

a. Explain briefly what program extraction is.

   (5 points)

b. Give an example of a formula that is a tautology of classical first-order logic, but not a tautology of intuitionistic first-order logic.

   (5 points)

Exercise 6.

a. Give and explain the definition of an inductive type `natlist` of finite lists of natural numbers in Coq. (The type `nat` of natural numbers may be used.)

   (5 points)

b. Give and explain the definition of an inductive predicate `even` of type `nat` $\to$ `Prop` in Coq. (The natural numbers are built using the constructors `O` for zero and `S` for successor.)

   (5 points)

Het tentamencijfer is (het totaal aantal punten plus 10) gedeeld door 10.