

# Tentamen Toegepaste Logica

17 januari 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  $\lambda$ -calculus).

- a. Show that the formula  $((A \rightarrow B \rightarrow A) \rightarrow B) \rightarrow B$  is a tautology of first-order minimal propositional logic.  
(5 points)
- b. Give the (formal) type derivation in simply typed  $\lambda$ -calculus corresponding to the proof of 1a.  
(5 points)
- c. Give a proof of the formula  $A \rightarrow B \rightarrow A$  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  $\beta$ -normal form.  
(5 points)

**Exercise 2.** This exercise is concerned with first-order minimal predicate logic and  $\lambda P$  ( $\lambda$ -calculus with dependent types).

- a. Show that the formula  $(\forall x.(P(x) \rightarrow \forall y.Q(y))) \rightarrow \forall z.\forall w.(P(z) \rightarrow Q(w))$  is a tautology of first-order minimal predicate logic.  
(5 points)
- b. Give the typing derivation in  $\lambda P$  that corresponds exactly to the proof of 2a.  
(5 points)
- c. Assume that an algebraic term is encoded by a term in `Terms` in  $\lambda P$ . Explain the encoding of formulas of first-order minimal predicate logic in  $\lambda P$  in detail.  
(5 points)

d. The formal typing system for  $\lambda P$  contains the conversion rule:

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : \mathbf{Set}/\square}{\Gamma \vdash A : B'} \quad \text{with } B =_{\beta} B'$$

Explain why this rule is necessary.

(5 points)

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

a. Give the identity function on `bool` and give the polymorphic identity function in  $\lambda 2$ .

What is the formula in second-order minimal propositional logic corresponding to the type of the polymorphic identity?

(5 points)

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

(5 points)

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

$$\frac{\Gamma \vdash A : \mathbf{Set} \quad \Gamma, x : A \vdash B : \mathbf{Set}}{\Gamma \vdash \Pi x:A. B : \mathbf{Set}} \quad \frac{\Gamma \vdash A : \square \quad \Gamma, x : A \vdash B : \mathbf{Set}}{\Gamma \vdash \Pi x:A. B : \mathbf{Set}}$$

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  $\beta$ -reduction rule we also use the rules for the conditional:

$$\begin{aligned} c(M, N, \mathbf{t}) &\rightarrow M \\ c(M, N, \mathbf{f}) &\rightarrow N \end{aligned}$$

Here we take  $M : \mathbf{Bool}$  and  $N : \mathbf{Bool}$  and hence also  $c(M, N, \mathbf{t}) : \mathbf{Bool}$  and  $c(M, N, \mathbf{f}) : \mathbf{Bool}$ .

a. Let `or` =  $\lambda x:\mathbf{Bool}. \lambda y:\mathbf{Bool}. c(\mathbf{t}, y, x)$ . Show that  $(\mathbf{or} \ \mathbf{t} \ z) \rightarrow_{\mathbf{T}}^* \mathbf{t}$  (give the reduction step by step).

(5 points)

b. Explain why the term  $(\mathbf{or} \ z \ \mathbf{t})$  does *not* reduce to  $\mathbf{t}$ .

(5 points)

c. Define a term `or'` such that  $(\mathbf{or}' \ z \ \mathbf{t}) \rightarrow_{\mathbf{T}}^* \mathbf{t}$  (give the reduction step by step).

(5 points)

**Exercise 5.**

- a. Explain briefly what proof checking is and how it works.  
(5 points)
- b. Coq is based on intuitionistic (also called constructive) logic. Give three different ways to extend *intuitionistic* first-order minimal propositional logic to *classical* first-order minimal propositional logic.  
(5 points)

**Exercise 6.** This question is concerned with inductive types.

- a. Give and explain the definition of an inductive type `nat` of natural numbers in Coq.  
(5 points)
- b. Give and explain the type of the term `nat_ind` that is automatically generated by Coq once the inductive definition of `nat` is given.  
(5 points)

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