

# Exam Logical Verification

January 15, 2007

**There are six (6) exercises.**

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

**Exercise 1.** This exercise is concerned with first-order propositional logic (`prop1`) and simply typed  $\lambda$ -calculus ( $\lambda \rightarrow$ ).

- a. Give a derivation in `prop1` showing that

$$((A \rightarrow B) \rightarrow C \rightarrow D) \rightarrow C \rightarrow B \rightarrow D$$

is a tautology.

(5 points)

- b. Give the type derivation in  $\lambda \rightarrow$  corresponding to the proof in (1a.).

(5 points)

- c. Complete the following three simply typed  $\lambda$ -terms:

$$\begin{aligned} \lambda x :?. \lambda y :?. (x y) \\ \lambda x :?. \lambda y :?. \lambda z :?. (\lambda u :?. y) x z \\ \lambda x :?. \lambda y :?. x ((\lambda z :?. y) y) \end{aligned}$$

(5 points)

**Exercise 2.** This exercise is concerned with first-order predicate logic (`pred1`) and  $\lambda$ -calculus with dependent types ( $\lambda P$ ).

- a. Give a derivation in `pred1` showing that

$$(\forall x. P(x) \rightarrow A) \rightarrow (\forall y. P(y)) \rightarrow A$$

is a tautology.

(5 points)

- b. Give the  $\lambda P$ -term corresponding to the formula of (2a.).

(4 points)

- c. Assume  $z : \text{Terms}$ .  
 Give an inhabitant of the  $\lambda P$ -term found in (2b.) that may contain  $z$  but no other free variables.  
*NB: You do not have to give a type derivation.*  
 (4 points)
- d. Is the formula  

$$(\forall x. P(x) \rightarrow A) \rightarrow (\forall y. P(y)) \rightarrow A$$
 valid in case the domain of terms is empty?  
 (2 points)

**Exercise 3.** This exercise is concerned with second-order propositional logic (**prop2**) and polymorphic  $\lambda$ -calculus ( $\lambda 2$ ).

- a. Give a derivation in **prop2** showing that  $\forall a. a \rightarrow \forall b. b \rightarrow a$  is a tautology.  
 (3 points)
- b. Give the  $\lambda 2$  type corresponding to the formula  $\forall a. a \rightarrow \forall b. b \rightarrow a$ .  
 (3 points)
- c. Give a closed inhabitant of the type found in (3b.).  
 (3 points)
- d. Show how the polymorphic identity is instantiated to the identity on  $B : \star$  by means of application and  $\beta$ -reduction.  
 (3 points)
- e. Give the proof corresponding to the application of (3d.), and indicate the detour (corresponding to the  $\beta$ -redex).  
 (3 points)

**Exercise 4.** This exercise is concerned with impredicative encodings in  $\lambda 2$ .

- a. Consider the following type which encodes false:

$$F = \Pi a : \star. a$$

Assume  $f : F$ . Show how  $f$  can be used to construct an inhabitant of  $A : \star$ .  
 (4 points)

- b. Suppose  $A : \star$  and  $B : \star$ . Consider the following definition:

$$\text{And } AB = \Pi c : \star. (A \rightarrow B \rightarrow c) \rightarrow c$$

Assume  $M : \text{And } AB$ .

Use  $M$  to construct an inhabitant of  $A$ .

(6 points)

**Exercise 5.** This exercise is concerned with the Curry–Howard–De Bruijn isomorphism.

- a. What is the type inhabitation problem?  
(3 points)
- b. What is the corresponding problem in logic?  
(3 points)
- c. Is the type inhabitation problem decidable in  $\lambda \rightarrow$ ?  
(3 points)
- d. Is the type inhabitation problem decidable in  $\lambda P$ ?  
(3 points)
- e. What is the principle of program extraction?  
(3 points)

**Exercise 6.** This exercise is concerned with Coq.

- a. Consider an inductive definition of finite lists of natural numbers:

```
Inductive natlist : Set :=  
| nil : natlist  
| cons : nat -> natlist -> natlist .
```

Give the type of `natlist_ind`, which is used to give proofs by induction.  
(5 points)

- b. Give an inductive definition of polymorphic lists (finite lists with elements from  $X : \text{Set}$ ).  
(5 points)

- c. Consider the definition of an inductive predicate for less-than-equal:

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

Note the types of the constructors:

```
le_n : forall n : nat, le n n  
le_S : forall n m : nat, le n m -> le n (S m)
```

What is the type of `le 1 0`?

Give an inhabitant of `le 0 0`.

Give an inhabitant of `le 0 1`.

(5 points)

- d. Give a definition of an inductive predicate `sorted` on elements of `natlist`, using `le` for sorting.

*Hint: distinguish cases for the empty list, lists with exactly one element, and lists with two elements or more.*

(5 points)