

Exam Logical Verification with Answers

January 16, 2009

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 λ -calculus ($\lambda \rightarrow$).

- a. Give a proof in prop1 showing that the following formula is a tautology:

$$((A \rightarrow B \rightarrow A) \rightarrow B) \rightarrow B$$

A proof:

$$\frac{\frac{\frac{[(A \rightarrow B \rightarrow A) \rightarrow B^x]}{B} \quad \frac{\frac{\frac{[A^y]}{B \rightarrow A} I[z] \rightarrow}{A \rightarrow B \rightarrow A} I[y] \rightarrow}{E \rightarrow}}{I[x] \rightarrow}}{((A \rightarrow B \rightarrow A) \rightarrow B) \rightarrow B}$$

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

The corresponding typing derivation:

$$\frac{\frac{\frac{\Gamma_0 \vdash x : (A \rightarrow B \rightarrow A) \rightarrow B \quad \frac{\frac{\Gamma_2 \vdash y : A}{\Gamma_1 \vdash \lambda z : B. y : B \rightarrow A}}{\Gamma_0 \vdash \lambda y : A. \lambda z : B. y : A \rightarrow B \rightarrow A}}{\Gamma_0 \vdash (x(\lambda y : A. \lambda z : B. y)) : B}}{\vdash \lambda x : ((A \rightarrow B \rightarrow A) \rightarrow B). (x(\lambda y : A. \lambda z : B. y)) : ((A \rightarrow B \rightarrow A) \rightarrow B) \rightarrow B}}$$

We use:

$$\begin{aligned} \Gamma_0 &= \{x : (A \rightarrow B \rightarrow A) \rightarrow B\} \\ \Gamma_1 &= \{x : (A \rightarrow B \rightarrow A) \rightarrow B, y : A\} \\ \Gamma_2 &= \{x : (A \rightarrow B \rightarrow A) \rightarrow B, y : A, z : B\} \end{aligned}$$

c. Give closed inhabitants in simply typed λ -calculus of the following types:

$$\begin{aligned} &(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B \\ &(A \rightarrow B \rightarrow C) \rightarrow B \rightarrow A \rightarrow C \\ &((B \rightarrow A \rightarrow B) \rightarrow A) \rightarrow A \end{aligned}$$

Closed inhabitants:

$$\begin{aligned} &\lambda x : A \rightarrow A \rightarrow B. \lambda y : A. (x y) y \\ &\lambda x : A \rightarrow B \rightarrow C. \lambda y : B. \lambda z : A. (x z) y \\ &\lambda x : (B \rightarrow A \rightarrow B) \rightarrow A. x (\lambda u : B. \lambda v : A. u) \end{aligned}$$

Exercise 2. This exercise is concerned with first-order predicate logic (**pred1**) and λ -calculus with dependent types (λP).

a. Give a proof in **pred1** with a \forall -detour showing that the following formula is a tautology:

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

A proof with a \forall -detour:

$$\frac{\frac{\frac{[\forall y. P(y) \rightarrow A^u]}{P(x) \rightarrow A} E\forall}{\frac{A}{\forall z. A} I\forall} E\forall}{\frac{A}{(\forall y. P(y) \rightarrow A) \rightarrow A} I[v] \rightarrow} I[u] \rightarrow}{\forall x. (P(x) \rightarrow (\forall y. P(y) \rightarrow A) \rightarrow A)} \forall i$$

b. Give the λP -term corresponding to the formula in 2a.

$$\Pi x : \mathbf{Terms}. (P x \rightarrow (\Pi y : \mathbf{Terms}. P y \rightarrow A) \rightarrow A)$$

c. Give a closed inhabitant in λP of the answer to 2b.

$$\lambda x : \mathbf{Terms}. \lambda u : (P x). \lambda v : (\Pi y : \mathbf{Terms}. P y \rightarrow A). v x u$$

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

- a. Give a proof in **prop2** showing that the following formula is a tautology:

$$(\forall c. ((a \rightarrow b \rightarrow c) \rightarrow c)) \rightarrow a$$

A proof:

$$\frac{\frac{\frac{[\forall c. ((a \rightarrow b \rightarrow c) \rightarrow c)^u]}{(a \rightarrow b \rightarrow a) \rightarrow a} \quad E\forall \quad \frac{\frac{[a^x]}{b \rightarrow a} \quad I[y]}{a \rightarrow b \rightarrow a} \quad I[x]}{a} \quad E \rightarrow}{(\forall c. ((a \rightarrow b \rightarrow c) \rightarrow c)) \rightarrow a} \quad I[u] \rightarrow$$

- b. Give the $\lambda 2$ -type corresponding to the formula of 3a.

$$(\Pi c : *. ((a \rightarrow b \rightarrow c) \rightarrow c)) \rightarrow a$$

- c. Give a closed inhabitant in $\lambda 2$ of the answer to 3b.

A closed inhabitant:

$$\lambda u : (\Pi c : *. ((a \rightarrow b \rightarrow c) \rightarrow c)). u a (\lambda x : a. \lambda y : b. x)$$

Exercise 4. This exercise is concerned with encodings.

- a. The data-type of booleans is encoded in $\lambda 2$ as follows:

$$\mathbf{Bool} = \Pi a : *. a \rightarrow a \rightarrow a$$

Give two different closed inhabitants of **Bool** that can be used as encodings of *true* and *false*.

Two closed inhabitants:

$$\begin{aligned} \lambda a : *. \lambda x : a. \lambda y : a. x \\ \lambda a : *. \lambda x : a. \lambda y : a. y \end{aligned}$$

- b. The disjunction **Or** $A B$ is encoded in $\lambda 2$ as follows:

$$\mathbf{Or} \ A \ B = \Pi c : *. (A \rightarrow c) \rightarrow (B \rightarrow c) \rightarrow c$$

Assume $P : A$ and use P to give an inhabitant of **Or** $A B$.

An inhabitant of **Or** $A B$:

$$\lambda c : *. \lambda x : A \rightarrow c. \lambda y : B \rightarrow c. (x P)$$

c. Assume the following setting in Coq:

```
(* prop representing the propositions is declared as a Set *)  
Parameter prop : Set.
```

```
(* implication on prop is a binary operator *)  
Parameter imp : prop -> prop -> prop.
```

```
(* T expresses if a proposition in prop is valid  
   if (T p) is inhabited then p is valid  
   if (T p) is not inhabited then p is not valid *)  
Parameter T : prop -> Prop.
```

Give the type of `imp_introduction`, the variable that models the introduction rule for `imp`.

Here it is:

```
Parameter imp_introduction :  
forall p q : prop, (T p -> T q) -> T (p => q).
```

Exercise 5. This exercise is concerned with inductive data-types in Coq.

a. Give the definition of an inductive data-type `two` with exactly two elements.

```
Inductive two : Set :=  
| een : two  
| twee : two.
```

b. Give the induction principle for `two_ind`, for your data-type from 5a.

```
two_ind :  
forall P : two -> Prop,  
P een -> P twee -> forall t : two, P t
```

c. Give the definition of an inductive data-type `natpair` of pairs of natural numbers. (You can use the data-type `nat` of natural numbers.)

```
Inductive natpair : Set :=  
| pair : nat -> nat -> natpair .
```

Exercise 6. This exercise is concerned with inductive predicates in Coq.

- a. Complete the following definition of conjunction:

```
Inductive and (A : Prop) (B : Prop) : Prop :=
```

The complete definition:

```
Inductive and (A : Prop) (B : Prop) : Prop :=
conj : A -> B -> A /\ B.
```

- b. Consider the following inductive predicates:

```
Inductive ev : nat -> Prop :=
| ev0 : ev 0
| evS : forall n:nat , odd n -> ev (S n)
with odd : nat -> Prop :=
| oddS : forall n:nat , ev n -> odd (S n) .
```

Give inhabitants of the following:

```
ev 0
odd 1
ev 2
```

Inhabitants:

```
ev0 : ev 0
oddS 0 ev0 : odd 1
evS 1 (oddS 0 ev0) : ev 2
```

- c. Complete the following definition of the inductive predicate `even`:

```
Inductive even : nat -> Prop :=
| even0 :
| evenSS :
```

The complete definition:

```
Inductive even : nat -> Prop :=
| even0 : even 0
| evenSS : forall n:nat, (even n) -> (even (S (S n))) .
```