

Exam Logical Verification

January 14, 2008

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:

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

(5 points)

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

(5 points)

- c. Complete the following simply typed λ -terms:

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

(5 points)

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` showing that the following formula is a tautology:

$$(\forall x. P(x) \rightarrow Q(x)) \rightarrow (\forall x. P(x)) \rightarrow (\forall x. Q(x))$$

(5 points)

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

(5 points)

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

(5 points)

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:

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

(5 points)

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

(5 points)

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

(5 points)

Exercise 4. This exercise is concerned with dependent types.

- a. What is the type in Coq of `natlist_dep`, the constructor for dependent lists? And what is the type in Coq of `(natlist_dep 5)`?

(5 points)

- b. The function `length : natlist -> nat` takes as input a (non-dependent) list of natural numbers and gives as output its length. Give the type of the corresponding function `length_dep` on dependent lists of natural numbers.

(5 points)

Exercise 5. This exercise is concerned with encodings.

- a. Give an impredicative definition of `false` in `prop2`, call it `new_false`.

Show in `prop2` the following: $\forall c. (\text{new_false} \rightarrow c)$.

(Unfold the definition of `new_false` in your proof.)

(5 points)

- b. The impredicative definition of `and` in `prop2` is as follows:

`new_and A B = $\forall c. ((A \rightarrow B \rightarrow c) \rightarrow c)$.`

Show `(new_and A B) $\rightarrow A$` in `prop2`.

(Unfold the definition of `new_and` in your proof.)

(5 points)

- c. Consider the following setting for an encoding of propositional logic:

```

(* prop representing the propositions is declared as a Set *)
Parameter prop : Set.
(* implication on prop is a binary operator *)
Parameter imp : prop -> prop -> prop.
(* we can use infix notation => for imp *)
Infix "=>" := imp (right associativity, at level 85).
(* 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 types of the parameters `imp_introduction` and `imp_elimination` for the introduction rule and elimination rule of the implication `=>`.

(5 points)

Exercise 6. This exercise is concerned with Coq.

- a. Give the definition of an inductive datatype `two` with exactly two elements.
(5 points)
- b. Give the induction principle for the datatype `two`.
(5 points)
- c. Consider the following inductive definition of the predicate `even`:

```

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

```

Give, if possible, an inhabitant of the following:

```

even 0
even 1
even 2

```

(5 points)

- d. Give a fixed point definition of a function that takes as input a `natlist` and gives as output the sum of its elements (and `0` if the input-list is empty).
(5 points)

The final note is (the total amount of points plus 10) divided by 10.