

Exam Logical Verification

April 7, 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:

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

(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:

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

(5 points)

Exercise 2. This exercise is concerned with first-order predicate logic (pred1).

- a. Give the two detour elimination rules for first-order predicate logic.

(5 points)

- b. Give a proof of $(\forall x. P(x)) \rightarrow P(a)$ with a detour for $\forall$.

(5 points)

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

(5 points)

- d. Give a closed inhabitant in λP of the answer to 2c that contains a β -redex.

(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:

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

(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. Consider the definition of dependent lists:

```
Inductive natlist_dep : nat -> Set :=
| nil_dep : natlist_dep 0
| cons_dep : forall n : nat,
    nat -> natlist_dep n -> natlist_dep (S n).
```

Use this definition to give the term corresponding to the dependent list consisting of the three elements 1, 2, 3.

(5 points)

- b. The function `append : natlist -> natlist -> natlist` takes as input two (non-dependent) lists of natural numbers and gives as output their concatenation. Give the type of the corresponding function `append_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 *or* in prop2 is as follows:

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

Show $A \rightarrow (\text{new_or } A B)$ in prop2.

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

(5 points)

Exercise 6. This exercise is concerned with Coq.

- a. Give the definition of an inductive data-type `natpair` of pairs of natural numbers (with `nat` the type of natural numbers).
(5 points)
- b. Give the induction principle for `natpair`.
(5 points)
- c. Give the definition of a function that takes as input a `natpair` and gives as output the sum of the two elements.
(You can use the built-in addition of natural numbers in Coq.)
(5 points)
- d. Consider the following inductive definition of the predicate `le`:

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

Give if possible an inhabitant of the following:

```
forall n, n <= n  
0 <= 0  
0 <= (S 0)
```

(5 points)

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

