Exam Logical Verification

January 18, 2006

There are six (6) exercises.
Answers may be given in Dutch or English. Good luck!

Exercise 1. This exercise is concerned with first-order minimal propositional logic and simply typed λ-calculus.

a. Show that $(A \to A \to B) \to (C \to A) \to C \to B$ is a tautology.
(5 points)

b. Give the type derivation in simply typed λ-calculus corresponding to the proof of 1a.
(5 points)

c. Replace in the following three terms the ?'s by simple types, such that we obtain typable λ-terms.
$\lambda z{:}?.\, \lambda y{:}?.\, (\lambda x{:}?.\, y)\, z$
$(\lambda x{:}?.\, x)\, (\lambda y{:}?.\, \lambda z{:}?.\, z\, y)$
$\lambda x{:}?.\, \lambda y{:}?.\, \lambda z{:}?.\, x\, (x\, y)$
(6 points)

d. Give a proof of $A \to B \to A$ with a detour.
(4 points)

Exercise 2. This exercise is concerned with inductive definitions in Coq.

a. Give the inductive definition of the datatype `natlist` of lists of natural numbers.
(5 points)

b. Give the type of `natlist_ind` which is used to give proofs by induction on the structure of such lists of natural numbers.
(5 points)

c. The predicate `le` is defined as follows:

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

This defines a family of predicates `le n` (e.g. `le 0 : nat -> Prop`). Give an inhabitant or explain (very shortly) why there is no inhabitant for:

(i) `le 0 0`

(ii) `le (S 0) 0`

(iii) `le 0 (S 0)`

(5 points)

d. Define a predicate `lelist : nat -> natlist -> Prop` such that `lelist n l` holds if `n` is smaller than or equal to the head of `l`. Also `lelist n l` holds if `l` is the empty list.

(You may use 2c.)

(5 points)

Exercise 3. This exercise is concerned with first-order predicate logic.

a. Give the two kinds of detours of minimal first-order predicate logic.

(5 points)

b. Show that $((\forall x.\, P(x)) \vee (\forall x.\, Q(x))) \rightarrow \forall x.\, (P(x) \vee Q(x))$ is a tautology.

(5 points)

Exercise 4. This exercise is concerned with dependent types. We use the following definition in Coq:

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

a. Give the type of `natlist_dep` in λP-syntax (with $*$ and $\Box$).

(3 points)

b. What is the type of the answer to 4a?

(2 points)

c. What is the type of `nil_dep`?

(2 points)

d. Give the Coq-definition of `length_dep` that gives the length of a dependent list; its type is `forall n : nat, natlist_dep n -> nat`.

(3 points)

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

a. What is the type checking problem?
(3 points)

b. What is the corresponding (to type checking) problem in logic?
(3 points)

c. Give an example of a simple type that is not inhabited by a closed λ-term.
(4 points)

Exercise 6. This exercise is concerned with second-order propositional logic and polymorphic λ-calculus ($\lambda 2$).

a. Show that the formula $\forall a.\, a \to a$ is a tautology.
(3 points)

b. Give the $\lambda 2$-term corresponding to the formula $\forall a.\, a \to a$.
(3 points)

c. Give a $\lambda 2$-term that is an inhabitant of the answer to 6b.
(3 points)

d. Give the $\lambda 2$-derivation of $a : * \vdash a : *$
(3 points)

e. Give the $\lambda 2$-derivation of $a : * \vdash \lambda x{:}a.\, x : a \to a$
(You may use the earlier derivation.)
(4 points)

f. Give the $\lambda 2$-derivation corresponding to the proof of 6a.
(You may use the earlier derivations.)
(4 points)

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

appendix to the exam logical verification 2005 - 2006

Typing rules of the simply typed lambda calculus.
The environment is a finite set of declarations.

variable $$\frac{}{\Gamma, x : A \vdash x : A}$$

abstraction $$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x{:}A.\, M) : A \to B}$$

application $$\frac{\Gamma \vdash F : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash (F\, N) : B}$$

Typing rules in the style of pure type systems (PTSs).
In these rules the variable s ranges over the set of sorts $\{*, \Box\}$.
The environment is a finite list of declarations.

start $$\frac{}{\vdash * : \Box}$$

variable $$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$$

weakening $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}$$

product ($\lambda \to$) $$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A.\, B : *}$$

product (λP) $$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x{:}A.\, B : s}$$

product ($\lambda 2$) $$\frac{\Gamma \vdash A : s \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A.\, B : *}$$

abstraction $$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x{:}A.\, B : s}{\Gamma \vdash \lambda x{:}A.\, M : \Pi x{:}A.\, B}$$

application $$\frac{\Gamma \vdash F : \Pi x{:}A.\, B \qquad \Gamma \vdash M : A}{\Gamma \vdash (F\, M) : B[x := M]}$$

conversion $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \quad \text{with } B =_\beta B'$$