Exam Logical Verification

January 18, 2012

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

Exercise 1. *(5+5+4+4 points)*
This exercise is concerned with first-order propositional logic (prop1) and simply typed λ-calculus ($\lambda\!\rightarrow$).

a. Show that the following formula is a tautology of minimal prop1:
$((A \rightarrow B) \rightarrow C) \rightarrow B \rightarrow C$.

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

c. Give three different closed inhabitants in $\lambda\!\rightarrow$ of the following type:

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

d. Replace in the following three terms the ?'s by simple types, such that we obtain typable λ-terms. (NB: it is not asked to give the type derivations.)
$\lambda x : ?.\, \lambda y : ?.\, \lambda z : ?.\, (x\, z)\, (y\, z)$
$\lambda x : ?.\, \lambda y : ?.\, x\, (x\, y)$
$\lambda x : ?.\, \lambda y : ?.\, \lambda z : ?.\, (y\, x)\, (x\, z)$

Exercise 2. *(5+3+5+3+3 points)*
This exercise is concerned with first-order predicate logic (pred1) and λ-calculus with dependent types (λP).

a. Show that the following formula is a tautology of minimal pred1:
$\forall x.\, (P(x) \rightarrow (\forall y.\, P(y) \rightarrow A) \rightarrow A)$.

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

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

d. What is the type checking problem? Is it decidable for λP?

e. What is the type inhabitation problem ? Is it decidable for λP?

Exercise 3. *(5+3+5 points)*
This exercise is concerned with second-order propositional logic (prop2) and polymorphic λ-calculus ($\lambda 2$).

a. Show that the following formula is a tautology of minimal prop2:

$$\forall a. ((\forall c. ((a \to c) \to c)) \to a)$$

b. Give the $\lambda 2$-term corresponding to the formula in 3a.

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

Exercise 4. *(3+3+4+5 points)*
This exercises is concerned with encodings.

a. Give an impredicative definition of *false* in prop2, call it false.

b. Show that $\forall c. \mathsf{false} \to c$ is a tautology of prop2.
(with false your answer to 4a).

c. We define $\mathsf{and}\, A\, B$ with $A : *$ and $B : *$ in $\lambda 2$ as follows:

$$\mathsf{and}\, A\, B := \Pi c : *. (A \to B \to c) \to c$$

Assume an inhabitant $P : \mathsf{and}\, A\, B$. Give an inhabitant of A.

d. First-order propositional logic can be encoded in Coq using dependent types as follows:

```
(* prop representing the propositions is declared as a Set *)
Variable prop:Set.
(* T expresses if a proposion in prop is valid
   if (T p) is inhabited then p is valid
   if (T p) is not inhabited then p is not valid *)
Variable T: prop -> Prop.
(* conjunction is a binary operator represented by conj *)
Variable conj : prop -> prop -> prop .
```

Give the types of the variables

```
conj_intro
conj_elim_l
conj_elim_r
```

modeling introduction of conjunction, and left- and right elimination of conjunction.

Exercise 5. *(4+4+5 points)*
This exercise is concerned with inductive definitions in Coq.

a. Give the inductive definition of the datatype `boollist` of lists of booleans (booleans in Coq are of type `bool`).

b. Give the type of `boollist_ind` which is used to give proofs by induction on the structure of such lists of booleans.

c. Give a recursive definition of a function `andblist : boollist -> bool` that computes the conjunction of all booleans in the input-list, and returns `true` if the input is the empty list. You may wish to use the following:

```
true  : bool
false : bool
andb  : bool -> bool -> bool (* for conjunction *)
```

Exercise 6. *(4+4+4 points)*
This exercise is concerned with inductive predicates in Coq.

a. Consider the inductive predicate for less-than-equal in Coq:

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

Give if possible an inhabitant of the following, if it is not possible explain shortly why not:

```
le O (S O)
le (S O) O
```

b. Give the definition in Coq of an inductive predicate `tre` on natural numbers that holds exactly if the number can be divided by 3.

(The constructors for the natural numbers are `O : nat` and `S : nat -> nat`.)

c. Complete the following definition of conjunction in Coq:

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

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