logical verification 2008-2009
exercises 1

`prop1` **and simply typed λ-calculus**

Exercise 1.

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

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

Exercise 2.

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

b. Give the type derivation in simply typed λ-calculus corresponding to the proof of 2a.

Exercise 3.

a. Show that the formula $((A \to B \to A) \to B) \to B$ is a tautology of first-order minimal propositional logic.

b. Give the type derivation in simply typed λ-calculus corresponding to the proof of 3a.

Exercise 4.

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

b. Give the type derivation in simply typed λ-calculus corresponding to the proof of 4a.

Exercise 5.

a. What is the definition of a detour in a natural deduction proof?

b. Give a proof of $A \to A \to A$ in first-order minimal propositional logic that contains a detour.

c. Give the λ-term that corresponds to the proof of 5b.
Which part corresponds to the detour?
Give the normal form of the λ-term.

Exercise 6. Show that the following formulas are tautologies:

a. $(((A \to \bot) \to A) \to A) \to \neg\neg A \to A$,

b. $\neg\neg(((A \to B) \to A) \to A)$,

c. $(A \vee \neg A) \to ((\neg A \to B) \wedge (\neg A \to \neg B)) \to A$,

d. $\neg\neg((A \vee \neg A) \to ((\neg A \to B) \wedge (\neg A \to \neg B)) \to A)$.

simply typed λ-calculus

Exercise 7. Replace in the following terms the ?'s by simple types, such that we obtain typable λ-terms.

a. $\lambda x{:}?.\, \lambda y{:}?.\, x$

b. $\lambda x :?.\, \lambda y :?.(x\, y)$

c. $\lambda x{:}?.\, \lambda y{:}?.\, x\, y\, y$

d. $\lambda x{:}?.\, \lambda y{:}?.\, x\, (x\, y)$

e. $\lambda x{:}?.\, \lambda y{:}?.\, \lambda z{:}?.\, x\, (y\, z)$

f. $\lambda x{:}?.\, \lambda y{:}?.\, \lambda z{:}?.\, y\, (\lambda u{:}?.\, x)$

g. $\lambda x :?.\, \lambda y :?.\, \lambda z :?.\, (\lambda u :?.\, y)\, x\, z$

h. $\lambda x :?.\, \lambda y :?.\, x\, ((\lambda z :?.\, y) y)$

i. $\lambda x{:}?.\, \lambda y{:}?.\, \lambda z{:}?.\, z\, ((\lambda u{:}?.\, y)\, x)$

Exercise 8. Give four different closed normal forms of type $(A \to A) \to A \to A$.

Coq

Exercise 9. Consider the definition of `nat`:

```
Inductive nat : Set :=  O : nat | S : nat->nat.
```

a. What are the constructors of `nat`?

b. Describe the elements of `nat`.

c. Give the type of `nat_ind`.

Exercise 10. Consider the following definition:

```
Inductive A : Set :=
| a : A -> A
| b : A -> A -> A.
```

How many elements does the set `A` have?

Exercise 11.

a. Consider the definition of `natlist` for lists of natural numbers:

```
Inductive natlist : Set :=
| nil : natlist
| cons : nat -> natlist -> natlist.
```

Give the type of `natlist_ind`, which is used to give proofs by induction.

b. Give the definition of an inductive predicate `last_element` such that `(last_element n l)` means that `n` is the last element of `l`.

Exercise 12.

a. Give the inductive definition of the datatype `natbintree` of binary trees with unlabeled nodes and natural numbers at the leafs.

b. The Coq function for appending two lists is defined as follows:

```
Fixpoint append (l k : natlist) {struct l} : natlist :=
  match l with
    nil => k
  | cons n l' => cons n (append l' k)
  end.
```

In what argument is the recursion? Why is the recursive call (intuitively) safe?

c. Give the definition of a recursive function `flatten : natbintree -> natlist` which flattens a tree into a list that contains the nodes from left to right.

You may use append.

d. Give a recursive definition of a function `count` that takes as input a natbintree and gives as output the number of nodes of the tree.

Exercise 13. Consider the definition of an inductive predicate for even:

```
Inductive even : nat -> Prop :=
| even_zero    : even O
| even_greater : forall n:nat, even n -> even (S (S n)).
```

a. What is the type of `even O`?

b. Give an inhabitant of `even O`.

c. Give an inhabitant of `even 2`.

pred1

Exercise 14. What is the type of the function that can be extracted from the proof of the following theorem:

```
forall l : natlist,
{l' : natlist | Permutation l l' /\ Sorted l'}.
```

Exercise 15.

a. Give an example of a proof that is incorrect because the side-condition for the introduction rule for $\forall$ is violated.

b. The rule for elimination of an existential quantifier is:

$$\frac{\exists x.\, A \qquad \forall x.\, (A \to B)}{B}\; E\exists$$

What is the side-condition for this rule?

Exercise 16. Show that the following formulas are tautologies of first-order intuitionistic predicate logic.

a. $(\forall x.\, \neg P(x)) \to \neg(\exists x.\, P(x))$

Hint: use the existential quantification elimination rule as early as possible.

b. $\forall x.\, (P(x) \to \neg\forall y.\, (\neg P(y)))$.

c. $(\forall x.\, P(x)) \to \neg\exists y.\, \neg P(y)$.

d. $((\exists x.\, P(x)) \to (\forall y.\, Q(y))) \to \forall z.\, (P(z) \to Q(z))$.