

Exam Logical Verification

January 15, 2003

There are six (6) exercises.

Answers may be given in Dutch or English. Good luck!

Exercise 1.

- a. Show that the formula $((A \rightarrow B) \rightarrow (C \rightarrow D)) \rightarrow C \rightarrow B \rightarrow D$ is a tautology of first-order minimal propositional logic. (Give a proof in natural deduction with all assumptions canceled.)
(4 points)
- b. Give the (formal) type derivation in simply typed λ -calculus corresponding to the proof of 1a.
(4 points)
- c. Give the correspondence between terms in simply typed λ -calculus and proofs in first-order minimal propositional logic in detail.
(6 points)
- d. What is the inhabitation problem in simply typed λ -calculus?
What is the corresponding problem in first-order minimal propositional logic?
(4 points)

Exercise 2.

- a. Give the polymorphic identity, its type, and the formula corresponding to this type.
(6 points)
- b. Give the polymorphic version of the following function:
 $\lambda f:\text{nat} \rightarrow \text{bool}. \lambda g:\text{bool} \rightarrow \text{nat}. \lambda x:\text{nat}. g(f\ x).$
(In the polymorphic variant neither `nat` nor `bool` occurs.)
(5 points)
- c. Explain briefly the principle of program extraction.
(4 points)

Exercise 3.

- a. Show that the following formula is a tautology of first-order predicate logic: $(\forall x.(P(x) \rightarrow Q(x))) \rightarrow (\forall x.P(x) \rightarrow \forall y.Q(y))$.
(5 points)
- b. Indicate the error(s) in the following proof:

$$\frac{\frac{\frac{[\exists x.P(x, y)^u] \quad \frac{\frac{[P(x, y)^v]}{P(x, y) \rightarrow P(x, y)} I[v] \rightarrow}{P(x, y)} E\exists}{\forall x.P(x, y)} I\forall}{\forall y.\forall x.P(x, y)} I\forall}{\exists x.P(x, y) \rightarrow \forall y.\forall x.P(x, y)} I[u] \rightarrow$$

(6 points)

- c. Give the two different detours in minimal first-order predicate logic in schematic form.
(4 points)

Exercise 4. This exercise is concerned with Coq.

- a. Give the inductive type `natlist` of finite lists of natural numbers.
(4 points)
- b. Give the type of `natlist_ind`, for induction on `natlist`.
(4 points)
- c. Consider the following definition in Coq:

```
Inductive positive : Set :=
  build_odd  : positive -> positive
| build_even : positive -> positive
| one       : positive.
```

Explain how this defines the positive integers (i.e. $\{1, 2, 3, \dots\}$).

(Hint: positive integers are even or odd.)

(4 points)

- d. In addition to the type `positive` given in 4c we now also have:

```
Inductive Z : Set :=
  ZERO : Z | POS : positive -> Z | NEG : positive -> Z.
```

Explain how this defines the integers.

(4 points)

Exercise 5. This exercise is concerned with sequent calculus. The rules of the sequent calculus are given in the appendix.

- a. What is the interpretation of a sequent $A_1, \dots, A_m \vdash B_1, \dots, B_n$?
(2 points)
- b. Prove the following sequent: $\vdash A \wedge (B \vee C) \rightarrow (A \wedge B) \vee (A \wedge C)$.
(4 points)
- c. Prove the following sequent: $\vdash \forall x.P(x) \rightarrow \neg \exists y.(\neg P(y))$
(4 points)

Exercise 6. This exercise is concerned with PVS.

- a. Give an abstract datatype specification of the type consisting of the finite lists where the elements are of some unspecified type t . That is, complete the following:

```
lists [t:TYPE] : DATATYPE
BEGIN

END lists
```

(The precise syntax is not important. It is important to make clear what are the constructors, accessors, and recognizers.)

(6 points)

- b. The following is a (naive) specification of the integers.

```
integers: THEORY
BEGIN
int      : TYPE
zero     : int
succ     : int -> int
pred     : int -> int
END integers
```

What axiom(s) is (are) natural to add? Why?

(6 points)

- c. What is a predicate in PVS? Give an example of a predicate subtype.
(4 points)

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