Logical Verification 2016–2017
Vrije Universiteit Amsterdam
Lecturer: dr. J. C. Blanchette

Exam
Thursday 1 June 2017, 15:15–18:00, WN-M143
7 problems, 90 points
Answers may be given in English or Dutch

Problem 1. Simply typed λ-calculus (3+5+3+3 points)

Consider the term $M \equiv \lambda x : ?.\, \lambda y : ?.\, \lambda z : ?.\, z\, x\, y\, (y\, x)$.

a. Specify the missing types for x, y, and z so that the term is legal in the simply typed λ-calculus, replacing the question marks ('?') above.

Answer:

$x : a$
$y : a \to b$
$z : a \to (a \to b) \to b \to c$

b. Show the type derivation for M, based on your answer to the previous question.

Answer:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : a \to b}\quad\dfrac{}{\Gamma \vdash x : a}}{\Gamma \vdash y\, x : b}\,(1)}{\Gamma \vdash z\, x\, y\, (y\, x) : c}}{x : a,\, y : a \to b \vdash \lambda z.\, z\, x\, y\, (y\, x) : (a \to (a \to b) \to b \to c) \to c}}{x : a \vdash \lambda y.\, \lambda z.\, z\, x\, y\, (y\, x) : (a \to b) \to (a \to (a \to b) \to b \to c) \to c}}{\vdash \lambda x.\, \lambda y.\, \lambda z.\, z\, x\, y\, (y\, x) : a \to (a \to b) \to (a \to (a \to b) \to b \to c) \to c}$$

where $\Gamma = x : a,\, y : a \to b,\, z : a \to (a \to b) \to b \to c$ and (1) is

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash z : a \to (a \to b) \to b \to c}\quad\dfrac{}{\Gamma \vdash x : a}}{\Gamma \vdash z\, x : (a \to b) \to b \to c}\quad\dfrac{}{\Gamma \vdash y : a \to b}}{\Gamma \vdash z\, x\, y : b \to c}$$

c. Is the formula $a \to (a \to b) \to (a \to (a \to b) \to b \to c) \to c$ a tautology of *classical* first-order propositional logic? Justify your answer (formally or informally).

Answer:

It is easy to see with a truth table. Another way to prove it is to exploit the Curry–Howard–De Bruijn correspondence. Since there exists a closed

term of type $a \to (a \to b) \to (a \to (a \to b) \to b \to c) \to c$, the formula corresponding to that type is a theorem of intuitionistic propositional logic. A fortiori, it is a theorem of classical propositional logic.

d. Is the formula $a \to (a \to b) \to (a \to (a \to b) \to b \to c) \to c$ a tautology of *intuitionistic* first-order propositional logic? Justify your answer (formally or informally).

Answer:

Yes. See answer to question c above.

Problem 2. Polymorphic λ-calculus (2+2+2+4+4+3+3 points)

Consider the following impredicative characterizations:

$$\begin{aligned}\mathbf{B} &\equiv \Pi a : *.\ a \to a \to a\\ \mathtt{T} &\equiv \lambda a : *.\ \lambda t : a.\ \lambda f : a.\ t\\ \mathtt{F} &\equiv \lambda a : *.\ \lambda t : a.\ \lambda f : a.\ f\end{aligned}$$

a. How many different inhabitants (in closed normal form) does the type $\mathbf{B}$ have?

Answer:

2.

b. What is the type of $\mathtt{F}$?

Answer:

$\mathbf{B}$.

c. What is the type of $\mathbf{B}$?

Answer:

$*$.

d. Give a closed term $\mathtt{XOR}$ such that $\mathtt{XOR\ T\ T} = \mathtt{XOR\ F\ F} = \mathtt{T}$ and $\mathtt{XOR\ T\ F} = \mathtt{XOR\ F\ T} = \mathtt{F}$.

Answer:

$\lambda b : \mathbf{B}.\ \lambda c : \mathbf{B}.\ \lambda a : *.\ \lambda t : a.\ \lambda f : a.\ b\, a\, (c\, a\, t\, f)\, (c\, a\, f\, t)$

e. Show that your definition is correct in the $\mathtt{XOR\ F\ T}$ case.

Answer:

$$\begin{aligned}&\mathtt{XOR\ F\ T}\\ &= (\lambda b.\ \lambda c.\ \lambda a.\ \lambda t.\ \lambda f.\ b\, a\, (c\, a\, t\, f)\, (c\, a\, f\, t))(\lambda a.\ \lambda t.\ \lambda f.\ f)\, (\lambda a.\ \lambda t.\ \lambda f.\ t)\\ &= \lambda a.\ \lambda t.\ \lambda f.\ (\lambda a.\ \lambda t.\ \lambda f.\ f)\, a\, ((\lambda a.\ \lambda t.\ \lambda f.\ t)\, a\, t\, f)\, ((\lambda a.\ \lambda t.\ \lambda f.\ t)\, a\, f\, t)\\ &= \lambda a.\ \lambda t.\ \lambda f.\ (\lambda a.\ \lambda t.\ \lambda f.\ f)\, a\, t\, f\\ &= \lambda a.\ \lambda t.\ \lambda f.\ f\\ &= \mathtt{F}\end{aligned}$$

where each step consists of either definition expansions or β-reductions.

f. How many different inhabitants (in closed normal form) does the type $\mathbf{O} \equiv \Pi a : *.\ a \to (\mathbf{B} \to a) \to a$ have?

Answer:

3.

g. Define the a type `option` as an inductive datatype in Coq corresponding to **O**, in the same way that the standard `bool` type corresponds to **B**. Recall the definition of `bool`:

```
Inductive bool : Set :=
  true : bool
| false : bool.
```

Answer:

```
Inductive option : Set :=
  None : option
| Some : bool -> option.
```

Problem 3. Second-order propositional logic (3+3+3+3+3 points)

Are the following formulas tautologies of intuitionistic second-order propositional logic? Justify briefly (formally or informally).

a. $\forall a.\, a$

Answer:

No because it implies false (by taking $a := \bot$).

b. $\forall a.\, a \vee \neg a$

Answer:

No because the law of the excluded middle doesn't hold intuitionistically.

c. $(\forall a.\, a) \to (\forall a.\, a \vee \neg a)$

Answer:

Yes because we can derive any formula from $\forall a.\, a$.

d. $(\forall a.\, a \vee \neg a) \to (\exists x.\, \mathsf{p}\, x \to (\forall x.\, \mathsf{p}\, x))$

Answer:

No. Yes in λC because in a classical setting, as imposed by the assumption, the drinker's paradox holds, but the formula isn't propositional!

e. $(\forall a.\, a \vee \neg a) \to \neg\neg b \to b$

Answer:

Yes because in a classical setting, as imposed by the assumption, $\neg\neg b$ is equivalent to b.

Problem 4. Logical frameworks (4+4+4 points)

Consider the following encoding of intuitionistic first-order propositional logic in Coq:

```
prop : Set
 imp : prop -> prop -> prop
   T : prop -> Prop

imp_intro : forall p q : prop, (T p -> T q) -> T (imp p q)
 imp_elim : forall p q : prop, T (imp p q) -> T p -> T q
```

a. Extend the above specification with a constant `false` corresponding to $\bot$ and associated introduction and/or elimination rules, as appropriate.

Answer:

```
false : prop

false_elim : forall p : prop, T false → T p
```

b. Extend the specification further with a constant `or` corresponding to $\vee$ and associated introduction and/or elimination rules, as appropriate. You may use the disjoint sum type (`sum` or $+$, with constructors `inl` and `inr`).

Answer:

```
or : prop → prop → prop

 or_intro_left : forall p : prop, forall q : prop, T p → T (or p q)
or_intro_right : forall p : prop, forall q : prop, T q → T (or p q)
       or_elim : forall p : prop, forall q : prop, T (or p q) → T p + T q
```

c. Let a : `prop`. Give a proof term for $a \vee \bot \to a$ using the above encoding. As in Coq, you may use _ (underscore) to avoid specifying arguments that can be inferred from other arguments.

Answer:
The desired property is $P \equiv$ `imp (or` a `false)` a. The following proof term is an inhabitant of `T` P.

```
imp_intro _ _ (fun x : T (or a false) ⇒
  match or_elim _ _ x with
    inl y → y
  | inr y → false_elim a y
  end)
```

Problem 5. Strong normalization and confluence (4+2 points)

a. Prove or disprove (informally): If $M_1, \ldots, M_n$ are strongly normalizing untyped λ-terms and x is a variable, then the term $x\, M_1 \ldots M_n$ is strongly normalizing.

Answer:

The proof is by contraposition. Assume that there exists an infinite chain

$$x\, M_1 \ldots M_n \longrightarrow_\beta N_1 \longrightarrow_\beta N_2 \longrightarrow_\beta \cdots$$

Each β-reduction step can only take place in one of the arguments M_1 to M_n. Since n is finite, one of the arguments M_i must be rewritten an infinite number of times. Hence M_i is not strongly normalizing. QED.

b. What is the definition of confluence for β-reduction?

Answer:

For all M, N_1, N_2 such that $M \to_\beta^* N_1$ and $M \to_\beta^* N_2$, there exists a term P such that $N_1 \to_\beta^* P$ and $N_2 \to_\beta^* P$.

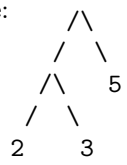
Problem 6. Inductive datatypes (3+3+4+4 points)

a. Define an inductive datatype, `nattree`, of binary trees in Coq. The leaf nodes are labeled with a natural number (of type `nat`). The inner nodes always have two children.

Answer:

```
Inductive nattree : Set :=
  Leaf : nat -> nattree
| Inner : nattree -> nattree -> nattree.
```

b. Represent the following ASCII-art tree, with three leaf nodes, using your Coq datatype:



Answer:

```
Inner (Inner (Leaf 2) (Leaf 3)) (Leaf 5)
```

c. What is the induction principle associated with the above tree type (i.e., the type of `tree_ind`)?

Answer:

```
forall P : nattree -> Prop,
  (forall n, P (Leaf n)) ->
  (forall l r, P l -> P r -> P (Inner l r)) ->
  (forall t, P t)
```

d. Define a Coq function `sum_tree` that, given a tree, computes the sum of the labels stored in the tree's leaf nodes.

Answer:

```
Fixpoint sum_tree (t : nattree) : nat :=
  match t with
    Leaf n -> n
  | Inner l r -> sum_tree l + sum_tree r
  end.
```

Problem 7. Inductive predicates (4+5 points)

Consider the following type of polymorphic lists:

```
Inductive list (A : Set) : Set :=
  nil : list A
| cons : A -> list A -> list A.
```

a. Complete the following inductive definition of a predicate `all` A P that tests whether a list of A elements consists only of elements satisfying P:

```
Inductive all (A : Set) (P : A -> Prop) : list A -> Prop :=
  ...
```

Answer:

```
  all_nil : all _ P (nil _)
| all_cons : forall x xs, P x -> all _ P xs ->
    all _ P (cons _ x xs).
```

b. Define an inductive predicate `mem` A x that tests whether a list over A contains the element $x : A$.

Answer:

```
Inductive mem (A : Set) (x : A) : list A -> Prop :=
  mem_hd : forall xs, mem _ x (cons x xs)
| mem_tl : forall y ys, mem _ x ys -> mem _ x (cons y ys).
```

The grade for the exam is the total amount of points divided by 10, plus 1.