COMP4161 T3/2020
Advanced Topics in Software Verification

Assignment 1

This assignment starts on Monday 28th September 2020 and is due on Monday 5th October 2020 6pm. We will accept plain text (.txt) files, PDF (.pdf) files, and Isabelle theory (.thy) files.

The assignment is take-home. This does NOT mean you can work in groups. Each submission is personal. For more information, see the plagiarism policy: https://student.unsw.edu.au/plagiarism

Submit using `give` on a CSE machine:

```
give cs4161 a1 files ...
```

For example:

```
give cs4161 a1 a1.thy a1.pdf
```

1 Types (12 marks)

Provide the most general type for the term $\lambda x\ y\ z.\ x\ z\ (y\ (f\ z))$ and construct a type derivation tree that shows the term is type correct.

Each node of the tree should correspond to the application of a *single* typing rule, indicating which typing rule is used at each step.

Under which contexts is the term type correct?

2 λ-Calculus (18 marks)

Recall the encoding of natural numbers and addition in lambda calculus (Church Numerals) seen in the lecture:

`0`	$\equiv$	$\lambda f\ x.\ x$
`1`	$\equiv$	$\lambda f\ x.\ f\ x$
`2`	$\equiv$	$\lambda f\ x.\ f\ (f\ x)$
`3`	$\equiv$	$\lambda f\ x.\ f\ (f\ (f\ x))$
`add`	$\equiv$	$\lambda m\ n.\ \lambda f\ x.m\ f\ (n\ f\ x)$

(a) What are the β normal forms for `add 2 x` and `add x 2`? Justify your answer by providing the single β reduction steps leading from the term to its normal form. There should be one reduction per step. Underline the redex being reduced. (12 marks)

(b) What does the result mean for the commutativity of `add` in this system? If `add x 2` is always β-convertible into `add 2 x` in this system, explain why. If it is not, give a counter example and, if possible, describe conditions under which it is. Informal reasoning is enough, no formal proof required for this question. (6 marks)

3 Propositional Logic (36 marks)

Prove each of the following statements, using only the proof methods `rule`, `erule`, `assumption`, and `cases`; and using only the proof rules `impI`, `impE`, `conjI`, `conjE`, `disjI1`, `disjI2`, `disjE`, `notI`, `notE`, `iffI`, `iffE`, `iffD1`, `iffD2`, `ccontr`, `classical`, `FalseE`, `TrueE`, `conjunct1`, `conjunct2`, and `mp`. You do not need to use all of these methods and rules.

(a) $(\neg\ B \longrightarrow A) = (A \vee B)$ (5 marks)

(b) $(A \longrightarrow B \longrightarrow C) = (B \longrightarrow A \longrightarrow C)$ (5 marks)

(c) $(P \longrightarrow Q) = (\neg\ Q \longrightarrow \neg\ P)$ (5 marks)

(d) $((A \vee B) \wedge C) = (A \wedge C \vee C \wedge B)$ (6 marks)

(e) $((P \longrightarrow ((Q \longrightarrow R) \longrightarrow Q) \longrightarrow Q) \longrightarrow P) \longrightarrow P$ (5 marks)

(f) $\neg\ \neg\ P \Longrightarrow P$ (3 marks)

(g) $(\neg\ A \wedge (B \longrightarrow C)) = (A \vee B \longrightarrow \neg\ A \wedge C)$ (7 marks)

4 Higher-Order Logic (34 marks)

Prove the following statements, using only the proof methods: `rule`, `erule`, `assumption`, `frule`, `drule`, `rule_tac`, `erule_tac`, `frule_tac`, `drule_tac`, `rename_tac`, and `case_tac`; and using only the proof rules: `impI`, `impE`, `conjI`, `conjE`, `disjI1`, `disjI2`, `disjE`, `notI`, `notE`, `iffI`, `iffE`, `iffD1`, `iffD2`, `ccontr`, `classical`, `FalseE`, `TrueI`, `allI`, `allE`, `exI`, and `exE`. You do not need to use all of these methods and rules. You may use rules proved in earlier parts of the question when proving later parts.

(a) $((\forall x.\ P\ x) \wedge (\forall x.\ Q\ x)) = (\forall x.\ Q\ x \wedge P\ x)$ (6 marks)

(b) $\forall x.\ P\ x \longrightarrow Q \Longrightarrow (\exists x.\ P\ x) \longrightarrow Q$ (3 marks)

(c) $(\neg\ (\forall x.\ P\ x)) = (\exists x.\ \neg\ P\ x)$ (6 marks)

(d) $(\forall\, a.\ P\ a\ a) \longrightarrow \neg\ (\forall\, a\ b.\ P\ b\ a \longrightarrow \neg\ P\ a\ b)$ (6 marks)

(e) $(\forall\, P\ x.\ P\ x) = \mathit{False}$ (4 marks)

(f) $\forall\, P\ h.\ (\forall\, x.\ \neg\ P\ x \longrightarrow P\ (h\ x)) \longrightarrow (\exists\, x.\ P\ x \wedge P\ (h\ (h\ x)))$ (9 marks)