COMP 4161 S2/11
Advanced Topics in Software Verification

Assignment 2

This assignment starts on Tuesday, 16.8.2011 and is due on Thursday, 25.8.2011, 23:59h. Submit using `give` on a CSE machine:

```
give cs4161 a2 file1 file2 ...
```

Your submission must include an Isabelle theory (.thy) file, and may also include a text (.txt) or PDF (.pdf) file.

1 Proofs in Predicate Logic (28 marks)

Prove or disprove in Isabelle

(a) $(\forall x.\ \forall y.\ R\ x\ y) = (\forall y.\ \forall x.\ R\ x\ y)$ (4 marks)

(b) $((\forall x.\ P\ x) \wedge (\forall x.\ Q\ x)) = (\forall x.\ (P\ x \wedge Q\ x))$ (4 marks)

(c) $(\forall P.\ P = \mathsf{True} \vee P = \mathsf{False}) \longrightarrow (\forall P.\ (\neg P \longrightarrow P) \longrightarrow P)$ (5 marks)

(d) $(\forall P.\ P = \mathsf{True} \vee P = \mathsf{False}) \longrightarrow (\forall P.\ (\neg\neg P) = P)$
[Hint: use the previous result, with forward reasoning.] (6 marks)

(e) $((\forall P.\ \forall x.\ (R\ x \wedge P\ x) \longrightarrow P\ (SOME\ x.\ P\ x)) \wedge$
$(\forall x\ y.\ Q\ x\ y \longrightarrow R\ y)) \longrightarrow$
$((\forall x.\ \exists y.\ Q\ x\ y) \longrightarrow (\exists f.\ \forall x.\ Q\ x\ (f\ x)))$ (9 marks)

You may only use `notI`, `notE`, `conjI`, `conjE`, `disjE`, `impI`, `impE`, `mp`, `iffI`, `iffE`, `iffD1`, `iffD2`, `allI`, `allE`, `spec`, `exI`, `exE` and `TrueI` in single step rule applications with the proof methods `rule`, `erule`, `drule`, `frule`, `rule_tac`, `erule_tac`, `drule_tac`, `frule_tac` and `assumption`.

2 Drinker Formula (20 marks)

(a) The *drinker formula* (sometimes incorrectly called the *drinker paradox* — it is not a paradox) is a theorem of classical logics (such as Isabelle's HOL) that, in words, states:

> There is a drinker in the pub such that, if he is drinking then everyone in the pub is drinking.

Formally, this theorem is written

$$\exists x.\ D\ x \longrightarrow (\forall x.\ D\ x).$$

Prove this theorem using only the proof methods `rule`, `erule`, `rule_tac`, `erule_tac`, `assumption` and `case_tac`, using only the rules `exE`, `exI`, `allI`, `allE`, `impI`, `impE`, `ccontr` and `notE`. (8 marks)

(b) In the above formalisation, the idea that the pub is non-empty is implicit. However, it is made more explicit in the following formula:

$$\exists x.\ P\ x \wedge (D\ x \longrightarrow (\forall x.\ P\ x \longrightarrow D\ x)).$$

Prove this formula in Isabelle under the restrictions from part (a), or disprove it by proving an instance of its negation. You may try to use automated methods to disprove it. If you disprove it, then modify it by adding an extra assumption that makes it true and then prove the modified result under the restrictions from part (a), relaxed to also allow the use of the rule `conjI`. (12 marks)

3 Higher-Order Abstract Syntax and Unification (10 marks)

(a) Does Isabelle consider the terms $(\forall x.\ P\ x)$ and $(\forall y.\ P\ y)$ to be equal? (1 mark)

(b) Use Isabelle to confirm your answer, and explain how you did so. (3 marks)

(c) Explain why your answer to part (a) is correct, in terms of how $\forall$ is implemented. (4 marks)

(d) Is the same true for $\exists$ and *SOME*? (2 marks)

4 Normalising Expressions (42 marks)

In classical logics (such as Isabelle's HOL), an expression involving the operators $\vee, \wedge$, $\neg$, and $\longrightarrow$ and the quantifiers $\forall$ and $\exists$, can be rewritten equivalently as a formula that begins with a (possibly empty) string of $\forall$ and $\exists$ quantifiers followed by a subformula that contains no $\forall$s or $\exists$s. For instance, the formula

$$\neg(\forall x.\ x \vee (\exists z.\ (\forall y.\ y \wedge x) \longrightarrow z))$$

can be rewritten in this way, producing the equivalent formula

$$\exists x.\ \forall z\ y.\ \neg(x \vee (y \wedge x \longrightarrow z)).$$

Design a set of terminating, unconditional rewrite rules that rewrite such expressions in this way. Avoid including any unnecessary rules (i.e. any rule that could be removed from the set of rewrite rules without preventing them from rewriting expressions in the way described above.)

(a) Implement your system in Isabelle by proving the rules as lemmas. You may use automated methods. (14 marks)

(b) Sketch an informal proof by-hand that your rewrite system is terminating. (20 marks)

Specifically:

(i) Define a function that measures the size of terms in some way, and use this function to define a well-founded order on terms. (8 marks)

(ii) Show that this order is monotonic with respect to the structure of terms. (6 marks)

(iii) Show that the right-hand side of each of your rewrite rules is smaller than its left-hand side, under this order. (6 marks)

(c) Is your rewrite system confluent? Explain why. (6 marks)

(d) Run your rewrite system using `(simp only:..)` on two non-trivial examples of your choice to make sure you haven't missed any rules that are necessary. (2 marks)