CIS 301: Logical Foundations of Programming, Exam I

March 7, 2008, 10:30-11:20am

General Notes

- Open textbook (Barwise & Etchemendy), open class notes, open solutions of homework assignments. No use of laptops or other computing devices.
- Please write your name on this page.

Good Luck!

NAME:

1. *7 points.* Consider the sentences P and Q given by

$$\begin{aligned} P &= A \leftrightarrow (B \wedge \neg C) \\ Q &= \neg((\neg B \vee C) \rightarrow A) \end{aligned}$$

Construct truth tables for P and Q. Then argue that P is a tautological consequence of Q, but Q is not a tautological consequence of P.

Consider the propositional rules of system $\mathcal{F}$, as listed pp. 557–559 in LPL.

1. Do these rules allow a proof of P from premise Q? Why, or why not? (*Hint:* look at Section 8.3.)

2. Do these rules allow a proof of Q from premise P? Why, or why not?

2. *7 points.* Consider the argument

$$\begin{array}{l} (P \to \neg R) \lor (Q \to \neg R) \\ \hline R \to \neg(P \land Q) \end{array}$$

An informal proof of its validity goes as follows: *We assume R with the goal of proving $\neg(P \land Q)$. We shall do so by contradiction, assuming $P \land Q$. We then do a proof by cases on the premise. If $P \to \neg R$ holds, modus ponens gives us $\neg R$, contradicting our assumption. Similarly, if $Q \to \neg R$ holds, we also get $\neg R$. Thus both cases lead to a contradiction, as desired.*

Your task is to convert the above into a formal proof in system F, using only the propositional rules listed pp. 557–559 in the textbook. Remember to annotate each step with the elimination or introduction rule you have used to derive that step, but you do *not* need to cite the supporting sentences.

3. *6 points.* Recall the factorial program from the notes:

$\{\mathtt{x} \geq 0\}$
```
    y := 1; z := 0;
    while z ≠ x do
        z := z + 1;
        y := y * z
    od
```
$\{\mathtt{y} = \mathrm{fac}(\mathtt{x})\}$

We gave the following correctness proof, using the invariant $\mathtt{y} = \mathrm{fac}(\mathtt{z})$.

1. The invariant is established by the preamble, since $1 = \mathrm{fac}(0)$.
2. The invariant is maintained by the loop body, since (with primes denoting "new" values) we have $\mathtt{y}' = \mathtt{y} \cdot \mathtt{z}' = \mathtt{y} \cdot (\mathtt{z}+1) = \mathrm{fac}(\mathtt{z})(\mathtt{z}+1) = \mathrm{fac}(\mathtt{z}+1) = \mathrm{fac}(\mathtt{z}')$.
3. At loop exit, $\mathtt{z} = \mathtt{x}$ so the invariant amounts to the postcondition.
4. The integer $\mathtt{z}$ is incremented by one in each iteration, and will therefore eventually equal the non-negative integer $\mathtt{x}$.

On the next page, we shall list some possible changes to the program. Each suggested change will invalidate (at least) one item in the above proof. Your task is to specify which item is invalidated, and how. To clarify what you are being asked for, the answer has been given to the first question.

Change 1: Replace the loop test by $\mathtt{z} \leq \mathtt{x}$.
This will invalidate item 3, since at loop exit we now have $\mathtt{z} > \mathtt{x}$ which does not enable us to infer $\mathtt{y} = \mathrm{fac}(\mathtt{x})$ from $\mathtt{y} = \mathrm{fac}(\mathtt{z})$.

Change 2: Replace the preamble by $\mathtt{y} := 0;\ \mathtt{z} := 0$.

Change 3: Replace the preamble by $\mathtt{y} := 1;\ \mathtt{z} := 1$.

Change 4: Swap the assignments in the loop body. That is, it becomes $\mathtt{y} := \mathtt{y} * \mathtt{z};\ \mathtt{z} := \mathtt{z} + 1$.