CS 205 Sections 07 and 08
Homework 4 – Accepted for grading 4/12

1. Prove that whenever $p_1, \ldots p_n$ is a list of two or more propositions,

$$\neg(p_1 \vee p_2 \vee \ldots \vee p_n)$$

is logically equivalent to

$$\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n$$

Use mathematical induction, and the fact that $\neg(p \vee q)$ is equivalent to $\neg p \wedge \neg q$ (De Morgan's law).

Answer:
Proof. Basis step: $n = 2$. In this case, $\neg(p_1 \vee p_2)$ is equivalent to $\neg p_1 \wedge \neg p_2$ by De Morgan.

Inductive step. Suppose $\neg(p_1 \vee p_2 \vee \ldots \vee p_k)$ is equivalent to $\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_k$. Consider $\neg(p_1 \vee p_2 \vee \ldots \vee p_k \vee p_{k+1})$. By De Morgan, this is equivalent to $\neg(p_1 \vee p_2 \vee \ldots \vee p_k) \wedge \neg p_{k+1}$. By the induction hypothesis, this is equivalent to $\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_k \wedge \neg p_{k+1}$. This is what we had to show.

We complete the proof by mathematical induction.

2. Prove by induction that if $a \equiv b(mod\ m)$ then $a^n \equiv b^n(mod\ m)$ for all $n \geq 0$.

   Answer:
   Proof. The key fact for the proof is that if $a \equiv b(mod\ m)$ and $c \equiv d(mod\ m)$ then $ac \equiv bd(mod\ m)$. This is Theorem 10 in the text on page 163.

   Basis step: $n = 0$. In this case, $a^0 = 1$ and $b^0 = 1$ and $1 \equiv 1(mod\ m)$.

   Inductive step. Suppose $a^k \equiv b^k(mod\ m)$. Consider $a^{k+1} = a(a^k)$ and $b^{k+1} = b(b^k)$. Since $a \equiv b(mod\ m)$ and by hypothesis $a^k \equiv b^k(mod\ m)$, by Theorem 10, $a^{k+1} \equiv b^{k+1}(mod\ m)$.

   We complete the proof by mathematical induction.

3. Verify that the program segment

   if $x < y$ **then**
   $m := x$
   else
   $m := y$

   is correct with respect to the initial asssertion **T** and the final assertion

$$(x \le y \wedge m = x) \vee (x > y \wedge m = y)$$

Answer:
We use the rule for verifying conditional programs by verifying each branch. We consider the branches in turn.

First, it must be shown that if the initial assertion is true and $x < y$, then after we execute $m := x$, it's true that $(x \le y \wedge m = x) \vee (x > y \wedge m = y)$. By inertia, $x < y$ is true after we execute $m := x$. And by implication, $x \le y$ is true. By assignment, $m = x$ is true after we execute $m := x$. So by logic, $x \le y \wedge m = x$ is true and thus $(x \le y \wedge m = x) \vee (x > y \wedge m = y)$.

Second, it must be shown that if the initial assertion is true and $y \le x$, then after we execute $m := y$, it's true that $(x \le y \wedge m = x) \vee (x > y \wedge m = y)$. By inertia, $y \le x$ is true after we execute $m := y$. By assignment, $m = y$ is true after we execute $m := y$. So we have $y \le x \wedge m = y$. We consider cases for $y \le x$: either $x > y$ or $y = x$. So either $(x > y \wedge m = y) \vee (y = x \wedge m = y)$. Looking at the second disjunct, $y = x \wedge m = y$ is equivalent to $y = x \wedge m = x$ and thus entails $x \le y \wedge m = x$. So we conclude $(x \le y \wedge m = x) \vee (x > y \wedge m = y)$.

This completes the proof.

4. This program computes quotients and remainders:

   $r := a$
   $q := 0$
   while $r \geq d$
   begin
   $\quad r := r - d$
   $\quad q := q + 1$
   end

   The program assumes that $d > 0$ and $a > 0$.

   Prove that

   $$d > 0 \ \wedge \ 0 \leq r \leq a \ \wedge \ a = dq + r$$

   is a *loop invariant* for the **while** loop. In other words, show that if

   $$d > 0 \ \wedge \ 0 \leq r \leq a \ \wedge \ a = dq + r \ \wedge \ r \geq d$$

   is true at the beginning of any iteration of the loop, then

   $$d > 0 \ \wedge \ 0 \leq r \leq a \ \wedge \ a = dq + r$$

   is true afterwards.

   Answer:
   Each iteration of the loop carries out the two instructions I_1 of $r := r - d$ and I_2 of $q := q + 1$. First we show that if $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r \wedge r \geq d$ is true before I_1 then $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r + d$ is true afterwards. Since d does not change in I_1, and we know $d > 0$ before I_1, we know $d > 0$ after I_1. Initially r has some value: call it r_0. We know initially $a = dq + r_0$. This does not depend on r, so it holds after I_1 by inertia. Likewise $0 \leq r_0 \leq a$ holds after I_1 by inertia. Meanwhile, by assignment, we know that after I_1, $r = r_0 - d$. Since $d > 0$ and $d \leq r_0 < a$ we know $0 \leq r \leq a$ after I_1. Finally, by algebra $r_0 = r + d$ and thus $a = dq + r + d$. By logic then, after I_1, $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r + d$.

   Next we show that if $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r + d$ is true before I_2, then $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r$ is true afterwards. Neither d nor r nor a is affected by assignment to q so that means $d > 0 \wedge 0 \leq r \leq a$ is true after I_2. Before I_2 q has some value: call it q_0. We know initially $a = dq_0 + r + d$. This does not depend on q, so it holds after I_2 by inertia. Meanwhile, by assignment, we know that after I_2, $q = q_0 + 1$. By algebra $q_0 = q - 1$ so $a = d(q-1) + r + d = dq - d + r + d = dq + r$. Thus by logic $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r$ is true after I_2.

   We complete the proof by observing that since the two instructions are run in sequence, these two arguments suffice to show that if $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r \wedge r \geq d$ is true before any iteration, then $d > 0 \wedge 0 \leq r \leq a \wedge a = dq + r$ is true afterwards.

5. Briefly, why does this invariant guarantee that the program can only terminate with a correct answer.

   Answer:
   When the loop completes, we have $a = dq + r$ and $0 \leq r$ by the loop invariant and $r < d$ by the termination condition. That makes r the remainder and q the quotient, by the Division Algorithm.