CS 4110 – Programming Languages and Logics
Homework #4

ChangeLog

- Version 1 (Wednesday, September 24): Initial release.
- Version 2 (Friday, September 26): Fixed typo in informal description of **do**-**until** loops. A **do**-**until** loop (**do** c **until** b) executes c one or more times until b becomes **true**, not until b becomes **false**.

Due Wednesday, October 1, 2014 at 11:59pm.

Instructions This assignment may be completed with one partner. You and your partner should submit a single solution on CMS. Please do not offer or accept any other assistance on this assignment. Late submissions will not be accepted.

Exercise 1.

(a) Extend the denotational semantics of IMP to handle the following commands:

$$c ::= \cdots \mid \textbf{if } b \textbf{ then } c \mid \textbf{do } c \textbf{ until } b$$

Operationally, these commands behave as follows. A one-armed conditional (**if** b **then** c) executes the body c only if b evaluates to **true**, whereas a **do**-**until** loop (**do** c **until** b) executes c one or more times until b becomes **true**.

(b) Extend Hoare logic with rules to handle one-armed conditionals and **do**-**until** loops.

Exercise 2. A simple way to prove two programs equivalent is to show that they denote the same mathematical object. In particular, this is often dramatically simpler than reasoning using the operational semantics. Using the denotational semantics, prove the following equivalences:

- $(x := x + 21; x := x + 21) \sim x := x + 42$
- $(x := 1; \textbf{do } x := x + 1 \textbf{ until } x < 0) \sim (\textbf{while true do } c)$
- $(x := x) \sim (\textbf{if } (x = x + 1) \textbf{ then } x := 0)$

Exercise 3. Find a suitable invariant for the loop in the following program:

```
$\{x = n \wedge y = m\}$
$r := x;$
$q := 0;$
while $(y \leq r)$ do {
    $r := r - y;$
    $q := q + 1;$
}
$\{r < m \wedge n = r + m * q\}$
```

Note: you do *not* have to give the proof of this partial correctness statement in Hoare Logic, but you may wish to complete the proof to convince yourself your invariant is suitable.

Exercise 4. Prove the following partial correctness specifications in Hoare Logic. You may write your proofs using a "decorated program", in which you indicate the assertion at each program point, invariants for loops, and any implications needed for the CONSEQUENCE rule.

(a)

```
$\{\exists n.\ x = 2 \times n + 1\}$
while $y > 0$ do
    $\{\exists n.\ x = 2 \times n + 1 \wedge y > 0\}$
    $x := x + 2$
    $\{\exists n.\ x = 2 \times n + 1 \wedge y > 0\}$
$\{\exists n.\ x = 2 \times n + 1 \wedge \neg(y > 0)\} \Rightarrow$
$\{\exists n.\ x = 2 \times n + 1 \wedge y \leq 0)\} \Rightarrow$
$\{\exists n.\ x = 2 \times n + 1\}$
```

(b)

```
$\{x = i \wedge y = j \wedge 0 \leq i\} \Rightarrow$
$\{0 = 0 \wedge x = i \wedge y = j \wedge 0 \leq i\}$
$z := 0;$
$\{z = 0 \wedge x = i \wedge y = j \wedge 0 \leq i\} \Rightarrow$
$\{z = (i - x) \times y \wedge x \geq 0\}$
while $(x > 0)$ do {
$\{z = (i - x) \times y \wedge x \geq 0 \wedge x > 0\} \Rightarrow$
$\{z = (i - x) \times y \wedge x + 1 \geq 0\}$
    $z := z + y;$
$\{z = (i - (x + 1)) \times y \wedge x + 1 \geq 0\}$
    $x := x - 1;$
};
$\{z = (i - x) \times j \wedge x \geq 0\}$
$y := 0;$
$\{thingshere\}$
$\{x = 0 \wedge y = 0 \wedge z = i \times j\}$
```

Exercise 5.

(a) If we replaced the WHILE rule with the following rule,

$$\textsc{While-Alt1}\ \frac{\vdash \{P\}\ c\ \{(b \Rightarrow P) \wedge (\neg b \Rightarrow Q)\}}{\vdash \{(b \Rightarrow P) \wedge (\neg b \Rightarrow Q)\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{Q\}}$$

would the logic still be (relatively) complete? Prove it or give a counterexample.

(b) What if we replaced it with the following rule instead?

$$\textsc{While-Alt2}\ \frac{\vdash \{P\}\ c\ \{P\}}{\vdash \{P\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{P \land \neg b\}}$$

Prove it or give a counterexample.

Debriefing

(a) How many hours did you spend on this assignment?
(b) Would you rate it as easy, moderate, or difficult?
(c) How deeply do you feel you understand the material it covers (0%100%)?
(d) If you have any other comments, we would like to hear them! Please write them here or send email to `jnfoster@cs.cornell.edu`.