CS615 – Homework Assignment 4

Exercise 1: In class we gave the following rules for the (backward) verification condition generation of assignment and let:

$$\begin{aligned}
\mathrm{VC}(c_1; c_2, B) &= \mathrm{VC}(c_1, \mathrm{VC}(c_2, B)) \\
\mathrm{VC}(x := e, B) &= [e/x]\ B \\
\mathrm{VC}(\mathsf{let}\ x = e\ \mathsf{in}\ c, B) &= [e/x]\ \mathrm{VC}(c, B)
\end{aligned}$$

That rule for let has a bug. Give a correct rule for let.

Exercise 2: Extra Credit. Given $\{A\}c\{B\}$ we desire that $A \Rightarrow \mathrm{VC}(c, B) \Rightarrow \mathrm{WP}(c, B)$. We say that our VC rules are *sound* if $\models \{\mathrm{VC}(c, B)\}\ c\ \{B\}$. Demonstrate the unsoundness of the buggy let rule by giving the following six things:

1. a command c and

2. a post-condition B and

3. a state σ such that

4. $\sigma \models \mathrm{VC}(c, B)$ and

5. $\langle c, \sigma \rangle \Downarrow \sigma'$ but

6. $\sigma' \not\models B$.

Exercise 3: Write a sound and complete Hoare rule for $\mathsf{do}\ c\ \mathsf{while}\ b$. This statement has the standard semantics (i.e., c is executed at least once, before b is tested).

Exercise 4: Give the (backward) verification condition rule for the command $\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b$ with respect to a post-condition P. The invariant Inv is true before and after c is executed. Your answer may not be defined in terms of $\mathrm{VC}(\mathsf{while}...)$.

Exercise 5: Consider the following three alternate while Hoare rules (named mal, jayne, and river):

$$\frac{\vdash \{X\}\ c\ \{b \Rightarrow X \land \neg b \Rightarrow Y\}}{\vdash \{b \Rightarrow X \land \neg b \Rightarrow Y\}\text{while } b \text{ do } c\ \{Y\}}\ \text{mal} \qquad \frac{\vdash \{X \land b\}\ c\ \{X\}}{\vdash \{X\}\text{while } b \text{ do } c\ \{X\}}\ \text{jayne}$$

$$\frac{\vdash \{X\}\ c\ \{X\}}{\vdash \{X\}\text{while } b \text{ do } c\ \{X \land \neg b\}}\ \text{river}$$

All three rules are sound, but only one rule is complete. Identify the two incomplete rules. For each incomplete rule provide the following:

1. the name of the rule and

2. A and

3. B and

4. σ and

5. σ' and

6. c such that

7. $\langle c, \sigma \rangle \Downarrow \sigma'$ and

8. $\sigma \models A$ and

9. $\sigma' \models B$ but

10. it is not possible to prove $\vdash \{A\}\ c\ \{B\}$.

Flavor text: Incompleteness in an axiomatic semantics or type system is typically not as dire as unsoundness. An incomplete system cannot prove all possible properties or handle all possible programs. Many research results that claim to work for the C language, for example, are actually incomplete because they do not address `setjmp/longjmp` or bitfields. (Many of them are also unsound because they do not correctly model unsafe casts, pointer arithmetic, or integer overflow.)

Exercise 6: No coding component. Instead, tell me how long you spent on this, something you like about the class (or the work, or the project, or whatever), something you do not like, and something I do not know about you. All non-null answers receive full credit. Think about and work on your class project.