CS655 – Homework Assignment 4
(Solutions due February 16)

February 9, 2005

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

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

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 a command c and a post-condition B and a state σ such that $\sigma \models \mathrm{VC}(c, B)$ and $\langle c, \sigma\rangle \Downarrow \sigma'$ but $\sigma' \not\models B$.

Exercise 3: Write a sound and complete Hoare rule for $\mathsf{do}\ c\ \mathsf{while}\ b$. This statement has the standard semantics (e.g., 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 VC(while...).

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

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

$$\frac{\vdash \{X\}\ c\ \{X\}}{\vdash \{X\}\mathsf{while}\ b\ \mathsf{do}\ c\ \{X \wedge \neg b\}}\ \mathsf{river}$$

All three rules are sound, but only one rule is complete. Identify the two incomplete rules. For each incomplete rule give a A, B, σ, σ' and c such that $\langle c, \sigma \rangle \Downarrow \sigma'$, $\sigma \models A$ and $\sigma' \models B$ but 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. Think about your project proposal; it is due on Tuesday, February 21.