CSCI 5535: Homework Assignment 7

Due Thursday, March 18, 2010

Exercise 1: Indicate in a sentence or two how much time you spent on this homework, how difficult you found it subjectively, and what you found to be the hardest part. Tell me something about yourself that I do not already know. Any non-empty answer will receive full credit.

Also, if your opinions have changed since the last assignment, indicate one thing you like about the class so far and one thing you would change about it.

Exercise 2: Verification condition for let. In class, we gave the following definitions for the (backward) verification condition generation of sequencing and assignment:

\[
\begin{align*}
\text{vc}(c_1; c_2, B) & \overset{\text{def}}{=} \text{vc}(c_1, \text{vc}(c_2, B)) \\
\text{vc}(x := e, B) & \overset{\text{def}}{=} [e/x]B
\end{align*}
\]

Consider the following definition for let:

\[
\text{vc(let } x = e \text{ in } c, B) \overset{\text{def}}{=} [e/x]\text{vc}(c, B)
\]

The above definition for let has a bug.

- Give a correct definition for let.

Given \{A\} c \{B\}, recall that we desire that

\[ A \Rightarrow \text{vc}(c, B) \Rightarrow \text{wp}(c, B) \]

holds. We say that our vc definitions are sound if

\[ \models \{\text{vc}(c, B)\} c \{B\} . \]

- Explain briefly why the buggy let definition is unsound and demonstrate the unsoundness by giving the following six things:

  1. a command \(c\),
2. a post-condition $B$,
3. a state $\sigma$, and
4. a state $\sigma'$ such that
5. $\sigma \models vc(c, B)$ and
6. $\langle c, \sigma \rangle \Downarrow \sigma'$ but
7. $\sigma' \not\models B$.

**Exercise 3:** Consider the following three alternate Hoare rules for *while* (named *huey*, *dewey*, and *louie*):

\[
\frac{\vdash \{X\} c \{b \Rightarrow X \land \neg b \Rightarrow Y\}}{\vdash \{b \Rightarrow X \land \neg b \Rightarrow Y\} \text{while } b \text{ do } c \{Y\}} \quad \text{huey}
\]

\[
\frac{\vdash \{X \land b\} c \{X\}}{\vdash \{X\} \text{while } b \text{ do } c \{X\}} \quad \text{dewey}
\]

\[
\frac{\vdash \{X\} c \{X\}}{\vdash \{X\} \text{while } b \text{ do } c \{X \land \neg b\}} \quad \text{louie}
\]

All three rules are sound, but only one rule is complete. Identify the one complete rule and the two incomplete rules.

1. For the complete rule, do the following:
   (a) Provide the name of the rule.
   (b) Show that the system of axioms remains complete if we replace the old rule for *while* with this one. You must show that any derivation that uses the old rule for *while* can be written with this rule instead. Hint: begin by considering a derivation that ends in the old rule for *while*.

2. For each incomplete rule, do the following:
   (a) Provide the name of the rule.
   (b) Give a counterexample by providing the following:
      i. an assertion $A$,
      ii. an assertion $B$,
      iii. a state $\sigma$,
      iv. a state $\sigma'$, and
      v. a command $c$ such that
      vi. $\langle c, \sigma \rangle \Downarrow \sigma'$,
      vii. $\sigma \models A$, and
viii. $\sigma' \models B$, but
ix. it is not possible to derive $\vdash \{A\} c \{B\}$.

Commentary: 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).