Homework 6: Axiomatic Semantics and Hoare-style Verification

17-355/17-665/17-819: Program Analysis
Rohan Padhye

Due: Tuesday, 03/29/21 at 11:59 pm EST

100 points total

Assignment Objectives:

- Demonstrate understanding of verification condition generation and use in verifying programs.
- Write new Hoare Rules/axiomatic semantics for a particular language.
- Reason about soundness/completeness of a system for axiomatic semantics.

Handin Instructions. Submit your assignment through the Gradescope link on Canvas (supports PDF and jpgs/photos) by the due date. When submitting, indicate which pages of the PDF correspond to each homework question. Putting page breaks between questions makes this simpler. Typesetting is not required, but is strongly suggested; you may submit photos or scans of handwritten answers, but they must be clear and legible.

Question 1, VCGen, *(24 points).* Consider the following rules for VCGen. We saw the first two in class; consider the third one as a proposed rule for `let`:

$$\begin{array}{lcl} VC(S_1; S_2, Q) & = & VC(S_1, VC(S_2, Q)) \\ VC(x := e, Q) & = & [e/x]Q \\ VC(\texttt{let}\ x = e\ \texttt{in}\ S, Q) & = & [e/x]VC(S, Q) \end{array}$$

The rule for `let` is incorrect.

(a) Explain why, briefly, in English prose,

(b) Give a correct rule for `let`.

Question 2, Let rule soundness, *(18 points).* Given $\{P\}\ S\ \{Q\}$, we desire that $P \Rightarrow VC(c, Q) \Rightarrow WP(c, Q)$. We say that our VC rules are *sound* if $\vDash \{VC(S, Q)\}\ S\ \{Q\}$. Demonstrate the unsoundness of the buggy `let` rule above by giving/showing the following six things:

(a) a statement S and

(b) a post-condition Q and

(c) a state E, all such that

(d) $E \models VC(S, Q)$ and

(e) $\langle S, E\rangle \Downarrow E'$ but

(f) $E' \not\models Q$

Question 3, Do-while, *(10 points).* Write a sound and complete Hoare rule for `do` S `while` b. The statement has the standard semantics (i.e., S is executed at least once, before b is tested). You do not need to formally prove soundness/completeness, but make sure the rule makes sense.

Question 4, Loop proof obligations, *(24 points).* Consider the following program:

```
{N > 0 }
i := 0;
while (i < N) do
     i := i + 1
{i=N}
```

1. What is a suitable loop invariant for proving this post-condition holds?
2. Write out the verification condition for this program (don't forget to use the precondition!). You don't have to solve/prove the verification condition, just write it out neatly; it should be formatted so that it's especially clear which parts correspond to the three parts of the proof obligations for the loop:
   - Invariant is initially true:
   - Invariant is preserved by the loop body:
   - Invariant and exit condition imply postcondition:

Hint: you can get full credit on the second part even if your loop invariant (answer to the first part) is incorrect.

Question 5, Soundness/Completeness, *(24 points).* Consider the following three Hoare rules:

$$\frac{\vdash \{X\}\ S\ \{b \Rightarrow X \wedge \neg b \Rightarrow Y\}}{\vdash \{b \Rightarrow X \wedge \neg b \Rightarrow Y\}\ \texttt{while}\ b\ \texttt{do}\ S\ \{Y\}}\ \text{rule1} \qquad \frac{\vdash \{X \wedge b\}\ S\ \{X\}}{\vdash \{X\}\ \texttt{while}\ b\ \texttt{do}\ S\ \{X\}}\ \text{rule2}$$

$$\frac{\vdash \{X\}\ S\ \{X\}}{\{X\}\ \texttt{while}\ b\ \texttt{d}\ S\ \{X \wedge \neg b\}}\ \text{rule3}$$

Recall that a system of axiomatic semantics is *sound* if everything we can prove is also true: if $\vdash \{P\}\ S\ \{Q\}$ then $\models \{P\}\ S\ \{Q\}$. A system of axiomatic semantics is *complete* if we can prove all true things: if $\models \{P\}\ S\ \{Q\}$ then $\vdash \{P\}\ S\ \{Q\}$. All three of the rules above are sound, but none are complete.

For two of the identified incomplete rules, give/show example P, Q, E, S, and E' such that $\langle S, E\rangle \Downarrow E'$ and both $E \vDash P$ and and $E' \vDash Q$, but it is not possible given the rule to prove $\vdash \{P\}\ S\ \{Q\}$. Be clear about which rules you have chosen.

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

Acknowledgement: Thanks to Selva Samuel for finding a bug in the original formulation of this question back in 2016.