1 4F-1 Bookkeeping

- 0 pts Correct

Peer Review ID: 70933850 — enter this when you fill out your peer evaluation via gradescope

Exercise 4F-2. VCGen for Let [6 points]. The bug in the let rule is that it does not reassign the initial value of x back to x after the command is finished. In order to ensure the original value is assigned back to x, we can provide the rule for let as a sequence of commands where we:

1. Store the initial value of x is to a new variable a that hasn't been used yet and is not used in e and c.
2. Assign e to x.
3. Execute c.
4. Reassign a, which stores the original value of x, to x.

Below is a correct rule for let, where a is a variable that is not in scope before the let command is executed and is not used in e and c:

$$\begin{aligned}\text{VC}(\textsf{let } x = e \textsf{ in } c, B) &= \text{VC}(a := x \ ; \ x := e \ ; \ c \ ; \ x := a, B) \\ &= \text{VC}(a := x,\ \text{VC}(x := e,\ \text{VC}(c,\ \text{VC}(x := a, B)))) \\ &= \text{VC}(a := x,\ \text{VC}(x := e,\ \text{VC}(c,\ [a/x]\ B))) \\ &= \text{VC}(a := x,\ [e/x]\ \text{VC}(c,\ [a/x]\ B)) \\ &= [x/a]\ [e/x]\ \text{VC}(c,\ [a/x]\ B)\end{aligned}$$

2 4F-2 VCGen for Let

- 0 pts Correct

Exercise 4F-3. VCGen Mistakes [6 points].

1. Command: $c = \mathsf{let}\ x = y + 1\ \mathsf{in}\ \mathsf{skip}$
2. Post-condition: $B = \{x > 0\}$
3. State: $\sigma = [x := 0, y := 0]$
4. We see that $\sigma \models VC(c, B)$:

$$\begin{aligned}&\text{VC}(\mathsf{let}\ x = y + 1\ \mathsf{in}\ \mathsf{skip}, \{x > 0\})\\ &= [y + 1/x]\ \text{VC}(\mathsf{skip}, \{x > 0\})\\ &= [y + 1/x]\ \{x > 0\}\\ &= \{y + 1 > 0\},\end{aligned}$$

   and because $\sigma(y) = 0$, $\sigma(y) + 1 = 1 > 0$.

5. We apply the operational semantics rule for let to obtain σ'. We have $\sigma' = \sigma = [x := 0, y := 0]$ because the original value of $\sigma(x) = 0$ is restored after the let command is finished, and no variables were modified inside the body of the let command.

$$\dfrac{\dfrac{}{\langle x, \sigma\rangle \Downarrow 0} \quad \dfrac{\dfrac{\dfrac{}{\langle y, \sigma\rangle \Downarrow 0} \quad \dfrac{}{\langle 1, \sigma\rangle \Downarrow 1}}{\langle y + 1, \sigma\rangle \Downarrow 1}}{\langle x := y + 1, \sigma\rangle \Downarrow \sigma[x := 1]} \quad \dfrac{}{\langle \mathsf{skip}, \sigma[x := 1]\rangle \Downarrow \sigma[x := 1]}}{\langle \mathsf{let}\ x = y + 1\ \mathsf{in}\ \mathsf{skip}, \sigma\rangle \Downarrow \sigma}$$

6. $\sigma' \not\models B = \{x > 0\}$ because $\sigma'(x) = 0 \not> 0$.

Peer Review ID: 70933850 — enter this when you fill out your peer evaluation via gradescope

3 4F-3 VCGen Mistakes

- 0 pts Correct

Peer Review ID: 70933850 — enter this when you fill out your peer evaluation via gradescope

Exercise 4F-4. Axiomatic Do-While [6 points]. We present the following Hoare rule for do c while b:

$$\frac{\vdash \{A\}\ c\ \{B\} \quad \vdash \{B\}\ \textsf{while}\ c\ \textsf{do}\ b\ \{B \wedge \neg b\}}{\vdash \{A\}\ \textsf{do}\ c\ \textsf{while}\ b\ \{B \wedge \neg b\}}$$

In this rule, we execute c with pre-condition $\{A\}$ and post-condition $\{B\}$. After executing c once, the rest of the do c while b command reduces to a normal while b do c command. Thus, the post-condition $\{B\}$ after executing c serves as the loop invariant for the rest of the while loop. Using the Hoare rule for while, we know that the post-condition of the rest of the loop should be $\{B \wedge \neg b\}$, the invariant B with the additional condition that the boolean expression b is false. Overall, we denote this sequence of operations to have precondition $\{A\}$ with the post-condition after executing the rest of the loop $\{B \wedge \neg b\}$.

4 4F-4 Axiomatic Do-While

- 0 pts Correct