## 1 5F-1 Bookkeeping

**- 0 pts** Correct

**Exercise 5F-2. VCGen Do-While [8 points].** Choose exactly *one* of the two options below. (If you are not certain, pick the first. The answers end up being equivalent, but the first may be easier to grasp for some students and the second easier to grasp for others.)

- Give the (backward) verification condition formula for the command $\textsf{do}_{Inv}\ c\ \textsf{while}\ b$ with respect to a post-condition $P$. The invariant *Inv* is true before each evaluation of the predicate $b$. Your answer may not be defined in terms of VC(while...).
- Give the (backward) verification condition formula for the command $\textsf{do}_{Inv1,Inv2}\ c\ \textsf{while}\ b$ with respect to a post-condition $P$. The invariant *Inv1* is true before $c$ is first executed. The invariant *Inv2* is true before each evaluation of the loop predicate $b$. Your answer may not be defined in terms of VC(while...).

**Answer:**
We will try to solve it in the second way. First we unwind the command.

$$\textsf{do}_{Inv1,Inv2}\ c\ \textsf{while}\ b \qquad \Leftrightarrow \qquad \textsf{assert}\ Inv1\ ;\ c\ ;\ \textsf{while}_{Inv2}\ b\ \textsf{do}\ c$$

So we have

$$\begin{aligned} & VC(\textsf{do}_{Inv1,Inv2}\ c\ \textsf{while}\ b, B) \\ = \ & \mathrm{VC}(\textsf{assert}\ Inv1\ ;\ c\ ;\ \textsf{while}_{Inv2}\ b\ \textsf{do}\ c, B) \\ = \ & \mathrm{VC}(\textsf{assert}\ Inv1, \mathrm{VC}(c\ ;\ \textsf{while}_{Inv2}\ b\ \textsf{do}\ c, B)) \\ = \ & Inv1 \wedge \mathrm{VC}(c, \mathrm{VC}(\textsf{while}_{Inv2}\ b\ \textsf{do}\ c, B)) \end{aligned}$$

According to the lecture, we have

$$\begin{aligned} & VC(\textsf{do}_{Inv1,Inv2}\ c\ \textsf{while}\ b, B) \\ = \ & Inv1 \wedge \mathrm{VC}(c, VC(\textsf{while}_{Inv2}\ b\ \textsf{do}\ c, B)) \\ = \ & Inv1 \wedge \mathrm{VC}(c, Inv2 \wedge (\forall x_1 \dots x_n . Inv2 \Rightarrow (b \Rightarrow \mathrm{VC}(c, Inv2)) \wedge \neg b \Rightarrow B)) \end{aligned}$$

2 5F-2 VCGen Do-While

**- 0 pts** Correct

**Peer Review ID: 72123356** — enter this when you fill out your peer evaluation via gradescope

**Exercise 5F-3. VCGen Mistakes [20 points].** Consider the following three alternate while Hoare rules (named lannister, stark, and targaryen):

$$\frac{\vdash \{X\}\ c\ \{b \implies X \ \wedge\ \neg b \implies Y\}}{\vdash \{b \implies X \ \wedge\ \neg b \implies Y\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{Y\}}\ \textsf{lannister} \qquad \frac{\vdash \{X \ \wedge\ b\}\ c\ \{X\}}{\vdash \{X\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{X\}}\ \textsf{stark}$$

$$\frac{\vdash \{X\}\ c\ \{X\}}{\vdash \{X\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{X \ \wedge\ \neg b\}}\ \textsf{targaryen}$$

All three rules are sound but incomplete. Choose **two** incomplete rules. For each chosen rule provide the following:

1. the name of the rule and
2. $A$ and
3. $B$ and
4. $\sigma$ and
5. $\sigma'$ 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.)

**Answer:**

First, we do the `targaryen` one.

1. targaryen
2. $A = \ x \geq 0$
3. $B = \ x = 0$
4. $\sigma(x) = 5$



Peer Review ID: 72123356 — enter this when you fill out your peer evaluation via gradescope

5. $\sigma'(x) = 0$

6. $c =$ while $x > 0$ do $x := x - 1$

7. $\langle c, \sigma \rangle \Downarrow \sigma'$ as we proved in the class that $<$ while $b$ do $c, \sigma >\Downarrow \sigma'$ where $\sigma \models A$ and $\sigma' \models A \wedge \neg b$. Here, $b = x > 0$, so $A \wedge \neg b \;=\; x = 0$. Thus, $\langle c, \sigma \rangle \Downarrow \sigma'$.

8. $\sigma \models A$ as $\sigma(x) = 5 \geq 0$

9. $\sigma' \models B$ as $\sigma'(x) = 0 = 0$ but

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

$$\vdash \{x \geq 0\}\text{while } x > 0 \text{ do } x := x - 1\{x = 0\}$$

The targaryen rule cannot be the last rule we use, because its postcondition is textually the same as precondition $\wedge \neg b$. When targaryen is not the last rule, we can use the rule of inversion to get

$$\frac{D_1 ::\vdash x \geq 0 \Rightarrow X \quad \dfrac{D_2 ::\vdash \{X\}x := x - 1\{X\}}{\vdash \{X\}\text{while}x > 0\text{do}x := x - 1\{X \wedge \neg(x > 0)\}} \quad D_3 ::\vdash X \wedge \neg(x > 0) \Rightarrow x = 0}{\vdash \{x \geq 0\}\text{while } x > 0 \text{ do } x := x - 1\{x = 0\}}$$

According to $D_1$ and $D_3$, we have

$$X \subseteq \; x \geq 0$$
$$X \supseteq \; x = 0$$

Thus, consider $x = 0$, which will break $D_2$. So $D_2$ cannot be sound. So it is not possible to prove $\vdash \{A\}\ c\ \{B\}$.

Second, we do the `stark` one.

1. stark

2. $A = \; true$

3. $B = \; x = 0$

4. $\sigma(x) = 3$

5. $\sigma'(x) = 0$

6. $c =$ while $x \neq 0$ do $x := 0$

7. $\langle c, \sigma \rangle \Downarrow \sigma'$ as we proved in the class that $<$ while $b$ do $c, \sigma >\Downarrow \sigma'$ where $\sigma \models A$ and $\sigma' \models A \wedge \neg b$. Here, $b = x \neq 0$, so $A \wedge \neg b \;=\; x = 0$. Thus, $\langle c, \sigma \rangle \Downarrow \sigma'$.

8. $\sigma \models A$ as $A = true$

9. $\sigma' \models B$ as $\sigma'(x) = 0 = 0$ but

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

$$\vdash \{true\}\textsf{while } x \neq 0 \textsf{ do } x := x - 1\{x = 0\}$$

The $\textsf{stark}$ rule cannot be the last rule we use, because it requires precondition and postcondition the same. When $\textsf{stark}$ is not the last rule, we can use the rule of inversion to get

$$\dfrac{D_1 :: \vdash true \Rightarrow X \qquad \dfrac{\vdash \{X \land x \neq 0\}x := 0\{X\}}{\vdash \{X\}\textsf{while}x \neq 0\textsf{do}x := 0\{X\}} \qquad D_2 :: \vdash X \Rightarrow x = 0}{\vdash \{x \geq 0\}\textsf{while } x \neq 0 \textsf{ do } x := 0\{x = 0\}}$$

According to $D_1$ and $D_2$, we have

$$\vdash true \Rightarrow X \Rightarrow x = 0$$

Such $X$ does not exist. So it is not possible to prove $\vdash \{A\}\ c\ \{B\}$.

## 3 5F-3 VCGen Mistakes

**- 0 pts** Correct

Peer Review ID: 72123356 — enter this when you fill out your peer evaluation via gradescope