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 $\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b$ 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 $\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b$ 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 (second choice)

$$\text{VC}(\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b) = (Inv1 \rightarrow \text{VC}(c, Inv2)) \wedge ((Inv2 \wedge b) \rightarrow \text{VC}(c, Inv2)) \wedge ((Inv2 \wedge \neg b) \rightarrow P)$$

In plain language, my logic is as follows, starting with the post-condition:
First, when *Inv2* holds and b is false, then the post-condition must follow.
Whenever *Inv2* holds and b is true, executing c must reestablish *Inv2*.
Finally, *Inv1* is true before c is first executed, so given that invariant, the VC(c, *Inv2*) must follow.
While we do not know the behavior of c, we know that *Inv2* must be true before each evaluation of b, so *Inv1* being true or *Inv2* and b being true must lead to a verification condition that must hold for the command c and the loop invariant *Inv2*.

Question assigned to the following page: 3

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

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

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

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. σ and
5. σ' 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 $\vdash \{A\}\ c\ \{B\}$.

Answer (stark)

1. **Name:** stark
2. **A**: $x = 0$
3. **B**: $x = 1$
4. **Initial State:** $\sigma(x) = 0$
5. **Final State:** $\sigma'(x) = 1$
6. **Command** c : $\mathsf{while}\ (x < 1)\ \mathsf{do}\ (x := x + 1)$
7. **Execution:** We execute c once, starting with $x = 0$ and executing the command $x := x + 1$, leaving $x = 1$. Therefore $\langle c, \sigma\rangle \Downarrow \sigma'$
8. $\sigma \models$ **A**: In the initial state, $x = 0$, so $\sigma \models (x = 0)$
9. $\sigma' \models$ **B**: In the final state, $x = 1$, so $\sigma' \models (x = 1)$
10. We aim to show that it is not possible to prove using the stark rule that:

$$\vdash \{x = 0\}\ (\ \mathsf{while}\ (x < 1)\ \mathsf{do}\ (x := x + 1))\ \{x = 1\}$$

The alternate rule requires finding a loop invariant X such that $\{X\}$ is true before and after each iteration of the loop body. That is, the stark rule requires the loop's pre-condition and post-condition

Question assigned to the following page: 3

to be exactly the same invariant X. It leaves room for incompleteness because the alternate rule does not require that the loop guard in not true in the conclusion of the rule. That is:

$$\vdash \{X\} \text{ while } b \text{ do } c \{X\}$$

To examine this, we can apply logic in a forward direction, choosing a value for the invariant that matches the initial condition A and confirm that after running the command c that the condition B is also satisfied, or we can apply logic in a backwards direction and choose a value for X that satisfies the post-condition B first and work backwards. In the forward direction, if we choose the invariant X as $x = 0$ to cover the initial state σ, then the command $x := x + 1$ makes the invariant not true. We can choose the invariant X as $x = 1$ to cover the final state σ', but then we fail to cover the initial state σ. Therefore no invariant X exists to satisfy the stark rule for this scenario, even though the actual execution shows a clearly valid transition from σ to σ'. This shows that the alternate stark rule is incomplete.

Answer (targaryen)

1. **Name:** targaryen
2. **A**: $x = 0$
3. **B**: $(x = 1)$
4. **Initial State:** $\sigma(x) = 0$
5. **Final State:** $\sigma'(x) = 1$
6. **Command** c : while $(x < 1)$ do $(x := x + 1)$
7. **Execution:** We execute c once, starting with $x = 0$ and executing the command $x := x + 1$, leaving $x = 1$. Therefore $\langle c, \sigma \rangle \Downarrow \sigma'$
8. $\sigma \models$ **A**: In the initial state, $x = 0$, so $\sigma \models (x = 0)$
9. $\sigma' \models$ **B**: In the final state, $x = 1$, so $\sigma' \models (x = 1)$
10. While we use the same scenario as stark, targaryen is incomplete because it is required that the loop body command preserves the invariant for all states satisfying the invariant X unconditionally, that is:

$$\vdash \{X\}\ c\ \{X\}$$

We can set the invariant to something that will lead to the desired post-condition $(x = 1)$, but then the invariant will not be preserved by $x := x + 1$ when we start from $x = 0$, as required in its premise $\vdash \{X\}\ c\ \{X\}$. Therefore, it is not possible to prove $\vdash \{A\}\ c\ \{B\}$.