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 $\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b$ 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: We can define the following VC rule formula:

$$\text{VC}(\mathsf{do}_{\mathsf{Inv}}\ c\ \mathsf{while}\ b) = \text{VC}(c, \mathsf{Inv}) \wedge (\forall x_1 ... x_n.\ \mathsf{Inv} \Rightarrow (b \Rightarrow \text{VC}(c, \mathsf{Inv}) \wedge \neg b \Rightarrow P))$$

This rule states that $\text{VC}(c, \mathsf{Inv})$ must hold on entry into the do-while loop. This is because c always executes at least once, therefore the condition on entry is the same as the precondition for executing c once. Then, $b \Rightarrow \text{VC}(c, \mathsf{Inv})$ indicates that Inv must be preserved in an arbitrary iteration of the loop. Finally, $\neg b \Rightarrow P$ indicates that P holds when the loop terminates in an arbitrary iteration.

---

2 5F-2 VCGen Do-While

- **0 pts** Correct

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. Choose **two** incomplete rules. For each chosen incomplete 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 $\vdash \{A\}\ c\ \{B\}$.

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:

Firstly, we have:

1. targaryen
2. $A = \{\neg b\}$
3. $B = \{\neg b \wedge \neg b\}$
4. $\sigma = \{b = \mathsf{false}\}$

5. $\sigma' = \sigma$
6. $c =$ ‘$\mathsf{while}\ b\ \mathsf{do}\ b := \mathsf{true}$’
7. Naturally, since b is false, we have $\langle c, \sigma\rangle \Downarrow \sigma'$
8. $\sigma \models \{true\} \Leftrightarrow \sigma \models \{\neg false\} \Leftrightarrow \sigma \models \{\neg b\} \Leftrightarrow \sigma \models A$
9. $\sigma' \models B$ for the same reason that $\sigma \models A$
10. To prove $\{A\}c\{B\}$, we must show that $\{A\}b := \mathsf{true}\{A\}$, as the targaryen rule states. This is not possible as $\langle b := \mathsf{true}, \sigma_1[b = \mathsf{false}]\rangle \Downarrow \sigma_2[b = \mathsf{true}] \Rightarrow \sigma_2 \not\models \{\mathsf{false}\} \Leftrightarrow \sigma_2 \not\models \{\neg\mathsf{true}\} \Leftrightarrow \sigma_2 \not\models \{\neg b\} \Leftrightarrow \sigma_2 \not\models A$.

Secondly, we have:

1. stark
2. $A = \{\neg b\}$
3. $B = A$
4. $\sigma = \{b = \mathsf{false}\}$
5. $\sigma' = \sigma$
6. $c =$ ‘$\mathsf{while}\ b\ \mathsf{do}\ \mathsf{skip}$’
7. Naturally, since b is false, we have $\langle c, \sigma\rangle \Downarrow \sigma'$
8. $\sigma \models \{true\} \Leftrightarrow \sigma \models \{\neg false\} \Leftrightarrow \sigma \models \{\neg b\} \Leftrightarrow \sigma \models A$
9. $\sigma' \models B$ for the same reason that $\sigma \models A$
10. To show $\{A\}c\{B\}$, we must show $\{A \wedge b\}\mathsf{skip}\{A\}$. This is impossible because $A = \{\neg b\} \Rightarrow \{\neg b \wedge b\}\mathsf{skip}\{\neg b\} \Leftrightarrow \{\mathsf{false}\}\mathsf{skip}\{\neg b\}$.

---

3 5F-3 VCGen Mistakes

- **0 pts** Correct