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...).

I choose the first option
First rewrite the $\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b$ as

$$\mathsf{assert}\ (Inv_1); c; \mathsf{while}\ _{Inv_2}\ b\ \mathsf{do}\ c$$

Here we introduce new invariant of while loop Inv_2, the reason is the first execution of c might influence Inv_1. In addition, we have to make sure $Inv_2 \Rightarrow Inv_1$. Therefore

$$\mathrm{VC}(\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b, B) =$$

$$\mathrm{VC}(\mathsf{assert}\ (Inv_1 \wedge Inv_2 \Rightarrow Inv_1); c; \mathsf{while}\ _{Inv_2}\ b\ \mathsf{do}\ c, B) =$$

$$Inv_1 \wedge Inv_2 \Rightarrow Inv_1 \wedge \mathrm{VC}(c; \mathsf{while}\ _{Inv_2}\ b\ \mathsf{do}\ c), B) =$$

$$Inv_1 \wedge Inv_2 \Rightarrow Inv_1 \wedge \mathrm{VC}(c, \mathrm{VC}(\mathsf{while}\ _{Inv_2}\ b\ \mathsf{do}\ c, B))) =$$

Let $x_1, x_2 \cdots x_n$ be the modified variables when executing c

$$Inv_1 \wedge Inv_2 \Rightarrow Inv_1 \wedge \mathrm{VC}(c, Inv_2 \wedge (\forall x_1, x_2 \cdots x_n Inv_2 \Rightarrow (b \Rightarrow \mathrm{VC}(c, Inv_2) \wedge \neg b \Rightarrow B)))$$

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):

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

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

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. 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.)

stark rule is incomplete
Use the count to six example

1. the name of the rule and
2. $A = \mathsf{true}$
3. $B = x = 6$
4. $\sigma(x) = 1$
5. $\sigma'(x) = 6$ and
6. $c = \ \mathsf{while}\ x < 6\ \mathsf{do}\ x := x + 1$
7. $\langle c, \sigma \rangle \Downarrow \sigma'$, it's clear that it's true
8. $\sigma \models A$ and
9. $\sigma' \models B$
10. it's inpossible to prove $\vdash \{A\}\ c\ \{B\}$ using stark rule.

We will prove that with contraction. Assume there is a derivation rule D such that

$$D ::\ \ \vdash \{\mathsf{true}\}\ \mathsf{while}\ x < 6\ \mathsf{do}\ x := x + 1\{x = 6\}$$

By inversion, we know that the last rule we use must be stark rule or the rule of consequence

1. The last rule is stark rule. We know stark rule have the pre-condition and post-condition to be the same, but in this case $\{\mathsf{true}\}$ is not same as $\{x = 6\}$, which causes contradiction
2. The last rule is the rule of consequence. Therefore D should be in the following form

   $$\frac{\vdash \mathsf{true} \Rightarrow C \quad D' ::\ \ \vdash \{C\}\ \mathsf{while}\ x < 6\ \mathsf{do}\ x := x + 1\{C\} \quad \vdash C \Rightarrow \{x = 6\}}{\vdash \{\mathsf{true}\}\ \mathsf{while}\ x < 6\ \mathsf{do}\ x := x + 1\{x = 6\}}$$

   However, there is no such C that $\vdash \{\mathsf{true}\} \Rightarrow C \Rightarrow \{x = 6\}$, which also raises contradiction

Therefore, such derivation rule D doesn't exist, and thus stark rule is incomplete

targaryen rule is incomplete
Use the count to six example

1. the name of the rule and
2. $A = x < 7$
3. $B = x = 6$
4. $\sigma(x) = 1$
5. $\sigma'(x) = 6$ and
6. $c =$ while $x < 6$ do $x := x + 1$
7. $\langle c, \sigma \rangle \Downarrow \sigma'$, it's clear that it's true
8. $\sigma \models A$
9. $\sigma' \models B$
10. it's inpossible to prove $\vdash \{A\}\ c\ \{B\}$ using targaryen rule.

We will prove that with contraction. Assume there is a derivation rule D such that

$$D :: \ \vdash \{x < 7\} \text{ while } x < 6 \text{ do } x := x + 1 \{x = 6\}$$

By inversion, we know that the last rule we use must be targaryen rule or the rule of consequence

1. The last rule is targaryen rule. We know targaryen rule have the post-condition to be pre-condition $\wedge\neg(x < 6)$, which causes contradiction
2. The last rule is the rule of consequence. Therefore D should be in the following form

$$\dfrac{D_1 :: \ \vdash \{x < 7\} \Rightarrow C \quad D_2 :: \dfrac{D_3 :: \ \vdash \{C\} x := x + 1 \{C\}}{\vdash \{C\} \text{ while } x < 6 \text{ do } x := x + 1 \{C \wedge \neg(x < 6)\}} \quad D_4 :: \ \vdash C \wedge \neg(x < 6) \Rightarrow \{x = 6\}}{\vdash \{x < 7\} \text{ while } x < 6 \text{ do } x := x + 1 \{x = 6\}}$$

   Based on D_1 and D_4 we know that C should be $x \leq 6$.
   However, in this case $D_3 :: \ \vdash \{C\} x := x+1 \{C\}$ can't exist, which causes contradiction.

Therefore, such derivation rule D doesn't exist, and thus targaryen rule is incomplete

3 5F-3 VCGen Mistakes

- 0 pts Correct