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 $\mathsf{do}_{\mathit{Inv}}\ c\ \mathsf{while}\ b$ 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 $\mathsf{do}_{\mathit{Inv1},\mathit{Inv2}}\ c\ \mathsf{while}\ b$ 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...).

We will give the backward verification formula for the command $\mathsf{do}_{\mathit{Inv}}\ c\ \mathsf{while}\ b$ with respect to a post-condition P.
First we rewrite

$$\mathsf{do}_{\mathit{Inv}}\ c\ \mathsf{while}\ b$$

as

$$c; \mathsf{while}_{\mathit{Inv}}\ b\ \mathsf{do}\ c$$

The VC of this becomes $VC(c; \mathsf{while}_{\mathit{Inv}}\ b\ \mathsf{do}\ c, P)$ Which we can reduce using the rules presented in class for while loop and concatenation into:

$$VC(c, inv) \wedge (\forall x_1 ... x_n . Inv \Rightarrow ((b \Rightarrow VC(c, Inv)) \wedge (\neg b \Rightarrow P)))$$

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 $\mathsf{lannister}$, stark, and $\mathsf{targaryen}$):

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

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

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.)

1. Stark Rule
2. $A := x > 0$
3. $B := x < 0$
4. $\sigma(x) = 1$
5. $\sigma'(x) = -1$
6. $c :=$ while $x > 0$ do $x = -1$
7. $\langle c, \sigma \rangle \Downarrow \sigma'$ since the while loop will execute once if x=1 originally in sigma which means $\sigma'(x) = -1$ as the loop will set x to -1 before terminating since after setting x to -1, $x > 0$ will be false and we will exit the loop
8. $\sigma \models A$ evaluates to $1 > 0$ which is true
9. $\sigma' \models B$ evaluates to $-1 < 0$ which is true
10. But it is not possible to prove $\vdash \{A\}\ c\ \{B\}$ using only the Stark rule since it has no way of proving a post-condition that doesn't match the precondition.

For the Targaryen rule:

1. Targaryen Rule
2. $A := x > 0 \&\& y > 0$
3. $B := x < 0 \&\& y < 0$
4. $\sigma(x) = 1 \&\& \sigma(y) = 1$
5. $\sigma'(x) = -1 \&\& \sigma(y) = -1$
6. $c :=$ while $x > 0$ do $x = -1; y = -1$
7. $\langle c, \sigma \rangle \Downarrow \sigma'$ since the while loop will execute once if x=1 originally in sigma which means $\sigma'(x) = -1$ as the loop will set x to -1 before terminating since after setting x to -1, $x > 0$ will be false and we will exit the loop. Additionally, $\sigma'(y) = -1$ since the while loop sets y to -1 as well.
8. $\sigma \models A$ evaluates to $1 > 0 \&\& 1 > 0$ which is true
9. $\sigma' \models B$ evaluates to $-1 < 0 \&\& -1 < 0$ which is true
10. But it is not possible to prove $\vdash \{A\}\ c\ \{B\}$ using only the Targaryen rule since it has no way of proving a post-condition on the command in the while loop that doesn't match the precondition, namely it has no way of constructing $y < 0$

3 5F-3 VCGen Mistakes

- 0 pts Correct

Peer Review ID: 72953158 — enter this when you fill out your peer evaluation via gradescope