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...).

If we choose the second option, the verification condition for $\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b$ is:

$$VC(\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b) = Inv1 \wedge VC(c, Inv2) \wedge (\forall x_1 ... x_n . Inv2 \implies ((b \implies VC(c, Inv2)) \wedge (\neg b \implies P)) \quad (1)$$

The logic behind this is as follows. First, Inv1 must hold on entry. Then, Inv2 must be true after executing c, so VC(c, Inv2) must hold on entry as well. The rest of the formula is similar to the VC(while...) rule given in lecture, where if b is true, then c executes again, and Inv2 must be re-established after c finishes. Thus VC(c, Inv2) must hold at the beginning of each iteration. This repeats until b is false, upon which we arrive at the post condition 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 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 $\vdash \{A\}\ c\ \{B\}$.

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
2. $A := x < 2$
3. $B := x \geq 2$
4. $\sigma[x = 1]$
5. $\sigma'[x = 2]$

6. $c =$ while $x < 2$ do $x := x + 1$
7. $x = 1$ to begin, and the while loop will increment x until $x = 2$, so $< c, \sigma > \Downarrow \sigma'$.
8. $x = 1$ in σ, so $x < 2$. Thus, $\sigma \models A$.
9. $x = 2$ in σ', so $x \geq 2$. Thus, $\sigma' \models B$.
10. It is not possible to prove $\vdash \{A\}\ c\ \{B\}$ as shown below:

$$\frac{\vdash x < 2 \implies x \leq 3 \quad \vdash \{x \leq 3\}\ \text{while } x < 2 \text{ do } x := x + 1\{x \leq 3\} \quad \vdash \{x \leq 3 \implies ???\}}{\vdash \{x < 2\}\ \text{while } x < 2 \text{ do } x := x + 1\{x \geq 2\}}$$

Here, we use the rule of consequence to get the form of the Stark rule. At the ???, we need $x \leq 3 \implies x \geq 2$ to get the post condition B, which cannot be proven. Thus, this derivation cannot be completed and this rule is incomplete. For this rule to work, we need the Stark rule to have post condition $\{X \wedge \neg b\}$, so then at the ??? above, we would have $x \leq 3 \wedge x \geq 2 \implies x \geq 2$, which can be proven.

---

1. Targaryen
2. $A := x > 10$
3. $B := x > 10 \wedge \neg(x < 5)$
4. $\sigma[x = 12]$
5. $\sigma'[x = 12]$
6. $c =$ while $x < 5$ do $x := 6$
7. Since $x = 12$ in state σ, the while loop in c will not run, so x is still 12 in σ'. Thus $< c, \sigma > \Downarrow \sigma'$.
8. $x = 12$ in σ, so $x > 10$. Thus, $\sigma \models A$.
9. $x = 12$ in σ', so $x > 10 \wedge \neg(x < 5)$. Thus, $\sigma' \models B$.
10. It is not possible to prove $\vdash \{A\}\ c\ \{B\}$ as shown below:

$$\frac{\vdash \{x > 10\}\ x := 6\ \{x > 10\}}{\vdash \{x > 10\}\ \text{while } x < 5 \text{ do } x := 6\{x > 10 \wedge \neg(x < 5)\}}$$

We start with $x = 12$, so the precondition to $x := 6$ holds and we execute that command. However, the post-condition of $x := 6$ is $x > 10$. This is not true, so we can not continue with this derivation. Thus, this rule is incomplete.

Submission. Turn in the formal component of the assignment as a single PDF document via the `gradescope` website. Your name and Michigan email address must appear on the first page of your PDF submission but may not appear anywhere else.

Peer Review ID: 72932004 — enter this when you fill out your peer evaluation via gradescope

3 5F-3 VCGen Mistakes

- **0 pts** Correct