

**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  $\text{do}_{Inv} c \text{ while } b$  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  $\text{VC}(\text{while} \dots)$ .
- Give the (backward) verification condition formula for the command  $\text{do}_{Inv1, Inv2} c \text{ while } b$  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  $\text{VC}(\text{while} \dots)$ .

I choose the first option. VC of  $\text{do}_{Inv} c \text{ while } b$  is

$$\boxed{VC(c, Inv) \wedge [\forall x_1, \dots, x_n. Inv \implies (b \implies VC(c, Inv) \wedge \neg b \implies P)]}$$

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**):

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

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

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.  $\sigma$  and
5.  $\sigma'$  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**

### Example for the **stark** Rule

1. **Name:** stark.
2.  $A$ :  $x \geq 0$ .
3.  $B$ :  $x = 0$ .

Question assigned to the following page: [3](#)

4.  $\sigma$ : A state with  $x = 3$ .
5.  $\sigma'$ : The state with  $x = 0$ .
6.  $c$ : **while**  $(x > 0)$  **do**  $(x := x - 1)$ .
7. **Execution**:  $\langle c, \sigma \rangle \Downarrow \sigma'$ .
8.  $\sigma \models A$  and  $\sigma' \models B$ .
9. **Issue**: No choice of invariant  $X$  for the **stark** rule allows one to derive the stronger postcondition  $x = 0$  from  $x \geq 0$ .

### Example for the **targaryen** Rule

1. **Name**: targaryen.
2.  $A$ :  $x \geq 0$ .
3.  $B$ :  $x = 0$  (equivalently,  $x \geq 0 \wedge \neg(x > 0)$ ).
4.  $\sigma$ : A state with  $x = 3$ .
5.  $\sigma'$ : The state with  $x = 0$ .
6.  $c$ : **while**  $(x > 0)$  **do**  $(x := x - 1)$ .
7. **Execution**:  $\langle c, \sigma \rangle \Downarrow \sigma'$ .
8.  $\sigma \models A$  and  $\sigma' \models B$ .
9. **Issue**: The **targaryen** rule requires proving  $\{x \geq 0\} (x := x - 1) \{x \geq 0\}$ , which fails when  $x = 0$  even though the loop never executes in that state.

**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.