COMSM0067: Advanced Topics in Programming Languages
Problem Sheet: Hoare Logic versus Symbolic Execution — Solutions

What does each one tell you?

What we need

Parts (a): weakest preconditions and verification conditions.

$$\mathsf{wp}(x := a,\ Q) = Q[a/x] \qquad \mathsf{wp}(c_1;\ c_2,\ Q) = \mathsf{wp}(c_1,\ \mathsf{wp}(c_2,\ Q))$$

To prove $\{P\}\ c_0;\ \mathsf{while}\ b\ \mathsf{do}\ c\ \{Q\}$ with invariant I, show

VC1	(init)	$P \Rightarrow \mathsf{wp}(c_0,\ I)$
VC2	(preserve)	$I \wedge b \Rightarrow \mathsf{wp}(c,\ I)$
VC3	(exit)	$I \wedge \neg b \Rightarrow Q$

In both problems, $c_0 \equiv i := 0;\ s := 0$.

Parts (b): symbolic execution. A symbolic state is (σ, π). An assignment updates σ; a guard b forks into $\pi \wedge \sigma(b)$ and $\pi \wedge \neg\sigma(b)$; a path with unsatisfiable π is infeasible. The postcondition Q can fail on a path iff

$$\pi \wedge \neg\sigma(Q)$$

is satisfiable, and a model is a concrete input on which Q fails.

Solutions

1. (a) Here $b \equiv i < n$ and $I \equiv s = i \cdot i \wedge 0 \le i \le n$.

VC1: $n \ge 0 \Rightarrow \mathsf{wp}(i := 0;\ s := 0,\ I)$

$\mathsf{wp}(\dots) = (0 = 0 \cdot 0 \wedge 0 \le 0 \le n) \equiv n \ge 0$	assignment, twice	✓

VC2: $I \wedge i < n \Rightarrow \mathsf{wp}(s := s + 2i + 1;\ i := i + 1,\ I)$

$\mathsf{wp}(\dots) = \big(s + 2i + 1 = (i+1)(i+1) \wedge 0 \le i + 1 \le n\big)$	assignment, twice	
$s + 2i + 1 = i^2 + 2i + 1 = (i+1)^2$	using $s = i \cdot i$	
$0 \le i + 1 \le n$	using $0 \le i$ and $i < n$	✓

VC3: $s = i \cdot i \wedge 0 \le i \le n \wedge \neg(i < n) \Rightarrow s = n \cdot n$

$i \le n \wedge i \ge n$, so $i = n$ and $s = n \cdot n$		✓

All three VCs hold, so the triple is valid.

(b) Start from $\pi = \alpha \ge 0$, $\sigma(i) = \sigma(s) = 0$. Each lap adds $2i + 1$ to s.

laps	π	$\sigma(s)$	$\pi \wedge \sigma(s) \neq \alpha \cdot \alpha$
0	$\alpha \ge 0 \wedge \neg(0 < \alpha) \equiv \alpha = 0$	0	unsat
1	$\alpha = 1$	1	unsat
2	$\alpha = 2$	4	unsat
3	$\alpha = 3$	9	unsat

Notes.

- After k laps, π is $\alpha \geq 0 \wedge 0 < \alpha \wedge \cdots \wedge k-1 < \alpha \wedge \neg(k < \alpha)$, which simplifies to $\alpha = k$.
- The path that enters a fourth lap ($\alpha \geq 4$) is cut off by the bound and not explored.
- No query is satisfiable, so the postcondition cannot fail on any explored path.

(c)
- (a) proves that *every* terminating run from $n \geq 0$ ends with $s = n \cdot n$.
- (b) proves the same only for $n \leq 3$, and says nothing about $n \geq 4$.

(a) is partial correctness: it says nothing about runs that do not terminate (here every run terminates, since $n - i$ decreases).

2. (a) Here $b \equiv i \leq n$ and $I \equiv s = i(i-1)/2 \wedge 0 \leq i \leq n+1$.

VC1: $n \geq 0 \Rightarrow \mathsf{wp}(i := 0;\ s := 0,\ I)$

$\mathsf{wp}(\dots) = (0 = 0 \cdot (-1)/2 \wedge 0 \leq 0 \leq n+1) \equiv n \geq -1$	follows from $n \geq 0$ ✓

VC2: $I \wedge i \leq n \Rightarrow \mathsf{wp}(s := s + i;\ i := i + 1,\ I)$

$\mathsf{wp}(\dots) = \big(s + i = (i+1)i/2 \wedge 0 \leq i + 1 \leq n + 1\big)$	assignment, twice
$s + i = (i^2 - i + 2i)/2 = i(i+1)/2$	using $s = i(i-1)/2$
$0 \leq i + 1 \leq n + 1$	using $0 \leq i \leq n$ ✓

VC3: $s = i(i-1)/2 \wedge 0 \leq i \leq n+1 \wedge \neg(i \leq n) \Rightarrow s = n(n-1)/2$

$i \leq n+1 \wedge i > n$, so $i = n+1$ and $s = (n+1)n/2$	
$(n+1)n/2 = n(n-1)/2 \iff n = 0$	**fails**

VC3 fails, e.g. for $n = 1$, $i = 2$, $s = 1$. The proof does not go through.

(b) Start from $\pi = \alpha \geq 0$, $\sigma(i) = \sigma(s) = 0$. Each lap adds i to s; after k laps the loop exits iff $\neg(k \leq \alpha)$.

laps	π	$\sigma(s)$	$\pi \wedge \sigma(s) \neq \alpha(\alpha-1)/2$
0	$\alpha \geq 0 \wedge \neg(0 \leq \alpha)$		infeasible
1	$\alpha = 0$	0	unsat
2	$\alpha = 1$	1	**sat**: $\alpha = 1$
3	$\alpha = 2$	3	**sat**: $\alpha = 2$

Notes.

- The path that enters a fourth lap ($\alpha \geq 3$) is not explored.
- The postcondition fails for $n = 1$: the run gives $s = 0 + 1 = 1$, but $n(n-1)/2 = 0$.
- The loop runs once too often and computes $0 + 1 + \cdots + n$. Fix: $i < n$.

(c)
- (a) proves *nothing* about P_2: it shows only that *this invariant* is not enough. A failed VC means either the program is wrong or the invariant is too weak; the counterexample to VC3 need not be a reachable state.
- (b) proves that P_2 *has a bug*: $n = 1$ is a concrete failing input, and every path symbolic execution reports is a real run. It says nothing about $n \geq 3$.