Lecture 15 - March 28

Program Verification

Stronger vs. Weaker Assertions
Total vs. Partial Correctness

Announcements

Lab3

↳ grace period until 12noon

- Bonus Opportunity - **Course Evaluation**
- **ProgTest1**: Echo (eMail, Zoom); Jackie (Office Hour)
- **Lab3** due tomorrow
- **ProgTest2**
- **Final Exam**: Review Q&A Sessions

↳ data sheet

↳ one side only; put anything you like

↳ computer-typed ≥ 10pt

Lecture

Program Verification

Correctness - Motivating Examples

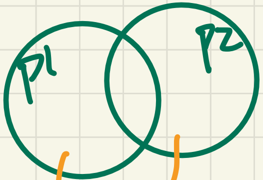
Assertions : precondition, postcondition, invariant

no overlap ⇒ can't say weaker or stronger

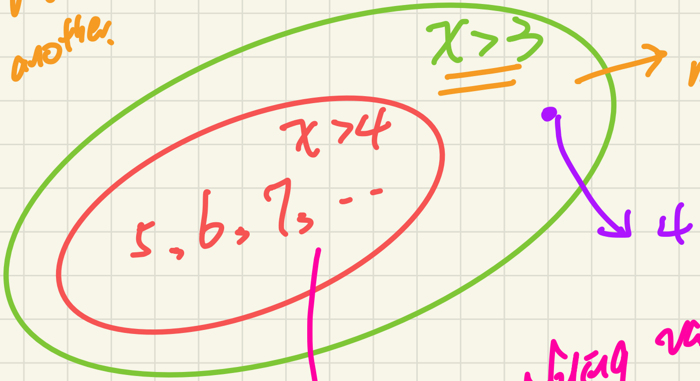
$x>4 \Rightarrow x>3$

$x>3 \rightarrow \{x \mid x>3\} = \{\boxed{4}, \underline{5, 6, 7}, \dots\}$

$x>4 \rightarrow \{x \mid x>4\} = \{\underline{5, 6, 7}, \dots\}$



In this case the sets of satisfying values are not subset of one another.



→ more satisfying values ⇒ weaker.

fewer satisfying values ⇒ stronger

p1 ⇒ p2

p1 stronger

p2 weaker

Weakest predicate : True

Strongest predicate: False

-- algorithm foo

assert □ → $x > 3$ vs. $x > 4$

$x > 3$ weaker

$x > 4$ stronger

more input values can be used for computation.

false precond.

↳ no input value accepted.

Imp.

Postcond. □ → P_1 vs. P_2

true post.

↳ no way to check computation

known: $P1 \Rightarrow P2$

P_2 weaker

↳ more computation outcomes are acceptable known: $P1 \Rightarrow \underline{P2}$

Preconditions

P_1 vs. P_2

stronger: require more (on input values)

wecker: require less (on input values)

search(int[] a, int tar)

↳ precond. for linear search:

a != null

↳ precond for binary search:

a != null

= a is sorted

known: $P1 \Rightarrow P2$

Postconditions

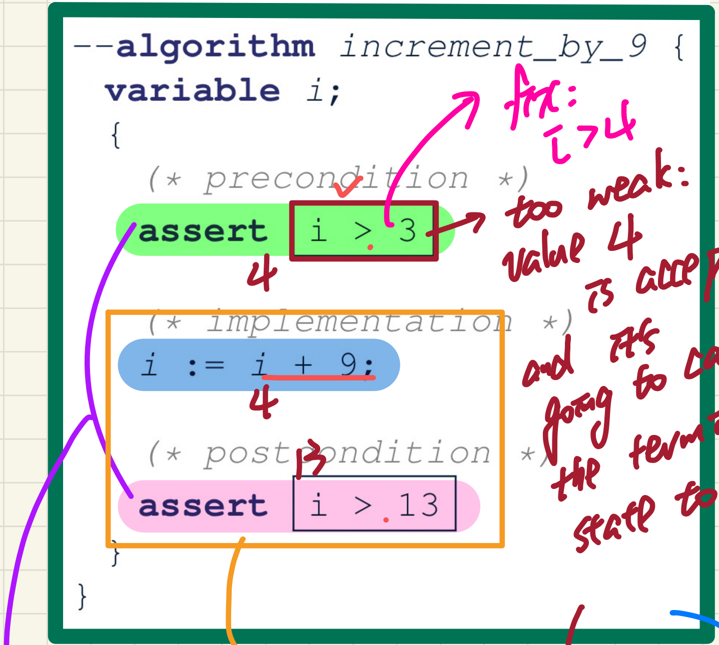
P_1 vs. P_2

P_1 ↳ stronger: ensure more on the output result

P_2 ↳ weaker: ensure less on the output result.

Program Correctness: Example (1)

traceability of specification



```
--algorithm increment_by_9 {
  variable i;
  {
    (* precondition *)
    assert i > 3
    (* implementation *)
    i := i + 9;
    (* postcondition *)
    assert i > 13
  }
}
```

too weak: value 4 is accepted, and it's going to cause the terminating state to violate the postcond.

specification

for wp calculations, assume imp. and postcond. are fixed

not correct!

Correctness

1. Relative concept

2. For input values satisfying the precondition, executing the implementation will:

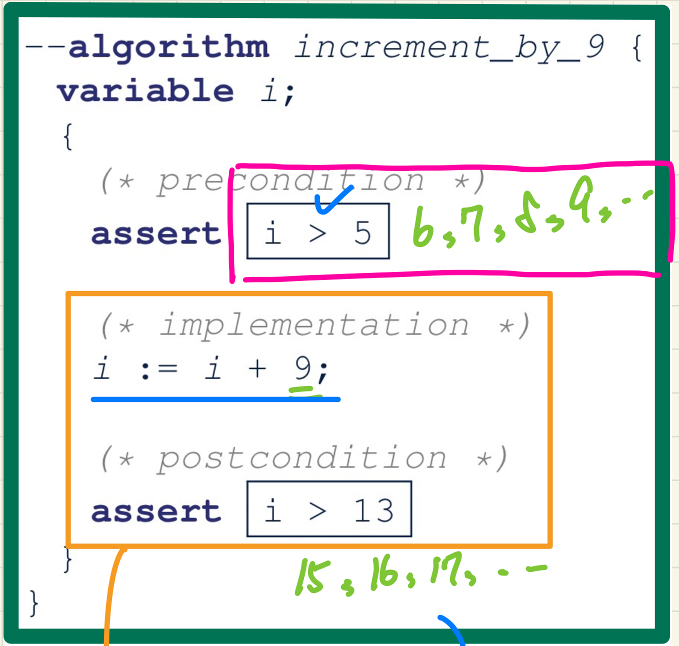
(1) terminate

(2) output/input satisfies the postcond upon termination.

Is this program correct?

$\{i > 3\}\ i := i + 9\ \{i > 13\}$ → unprovable.

Program Correctness: Example (2)



```
--algorithm increment_by_9 {
  variable i;
  {
    (* precondition *)
    assert i > 5

    (* implementation *)
    i := i + 9;

    (* postcondition *)
    assert i > 13
  }
}
```

6, 7, 8, 9, ...

15, 16, 17, ...

fixed.

Is this precond. too strong?

∵ 5 is disallowed by the precond.

need to check the REQ.

Is this program correct?

$\{i > 5\}\ i := i + 9\ \{i > 13\}$ → provable as a theorem

Lecture

Program Verification

Hoare Triple and Weakest Precondition

e.g. conjunction of a list of assertions

$$\{\ \underline{\qquad}\ \}\ \underline{\qquad S \qquad}\ \{\ \underline{\qquad}\ \}$$

can be transformed into a Boolean predicate

precondition

programming statement.

postcondition

PO1 ∧ ¬PO2 ⇒ incorrect.

Hoare Triple: $\{Q\}\ S\ \{R\}$

PO1

↳ Starting in a state satisfying Q, executing S will ✓ $\boxed{\text{terminate}}$ in a state satisfying R.

PO2

PO2 (without termination) ⇒ partial correctness

PO2 ∧ PO1 ⇒ total correctness