Lecture 17 - April 6

Program Verification

Contracts of Loops: Invariant vs. Variant
Correctness of Loops

Announcements

- Lab4 released
- Exam guide released

Program Verification

- Predicates : Stronger $\Rightarrow$ Weaker
- Hoare Triple $\{Q\}\ S\ \{R\}$

$\iff Q \Rightarrow wp(S, R)$

① :=
② if then else
③ ;

① Show wp calculation ② Prove

one example done in the math review lecture (equational style)

$$= \{ \text{ justification } \}$$

⋮

$$\equiv \{ \ldots \}$$

$$\equiv \{ \ldots \}$$

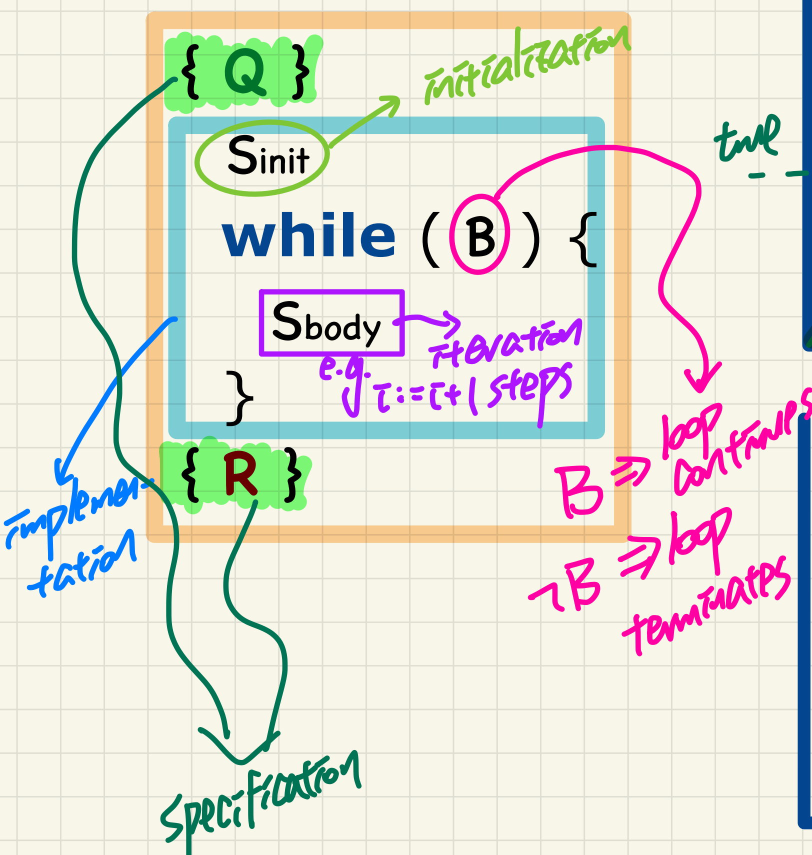
⋮

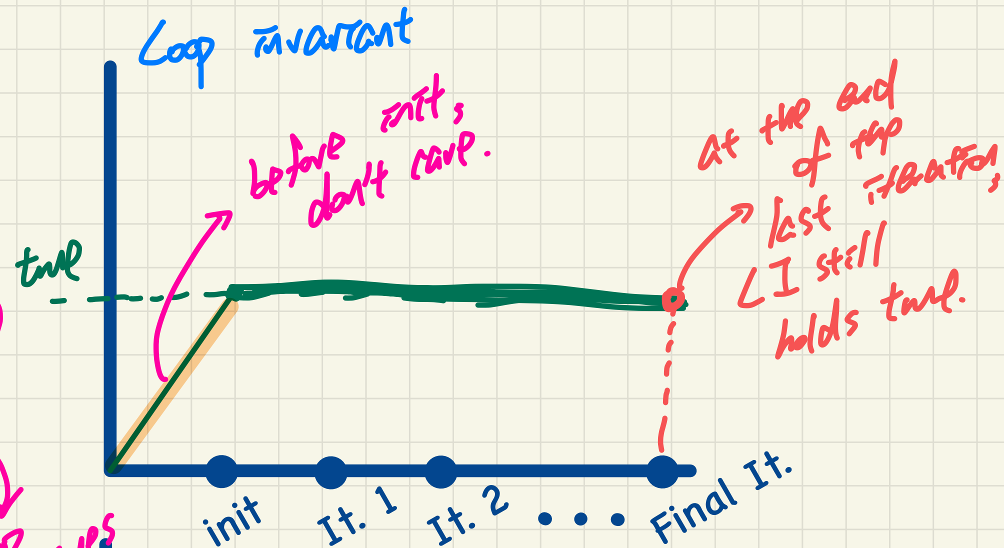
Lecture

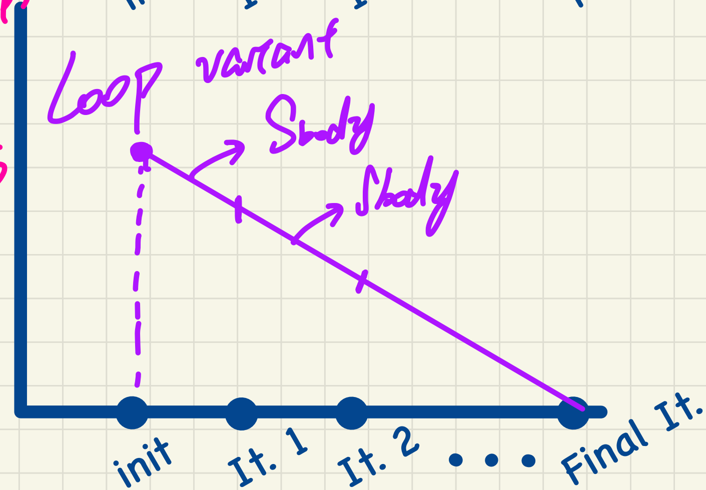
Program Verification

Contracts of Loops

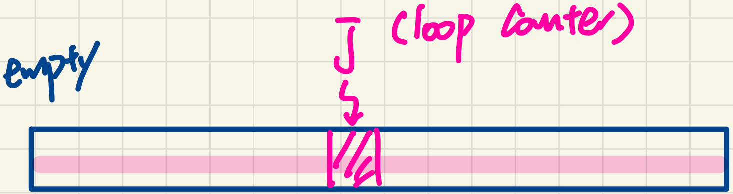
Correctness of Loops







non-empty

Input: 

Output: index i s.t. input[i] is max.

- Exercise. Write an assertion for the postcondition.

- Exercise 2: loop invariant.
  - ↳ Hint: loop counter
  - Hint: inclusive of j or not?

Contracts of Loops

Syntax

input

output

```
CONSTANT ... (* input list *)
I(var_list) == ...
V(var_list) == ...
--algorithm MyAlgorithm {
  variables ..., variant_pre = 0, variant_post = 0
  {
    assert Q; (* Precondition *)
    S_init
    assert I(...); (* Is LI established? *)
    while ( B ) {
      variant_pre := V(...);
      S_body
      variant_post := V(...);

      assert variant_post >= 0;
      assert variant_post < variant_pre;
      assert I(...); (* Is LI preserved? *)
    }
    assert R; (* Postcondition *)
  }
}
```

precond

LI established and maintained

body of loop

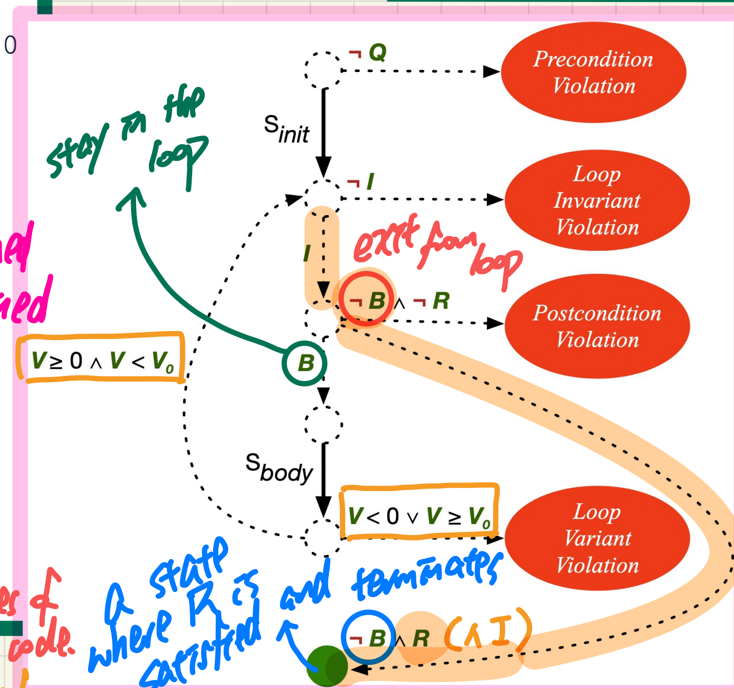
post cond

* R only needs to be established after these lines of code.

∴ In case there's code between end of loop and "assert R";

* LV: ① ∈ N ② LV < LV₀

Runtime Checks



Contracts of Loops: Example

Assume: Q and R are true

Specification

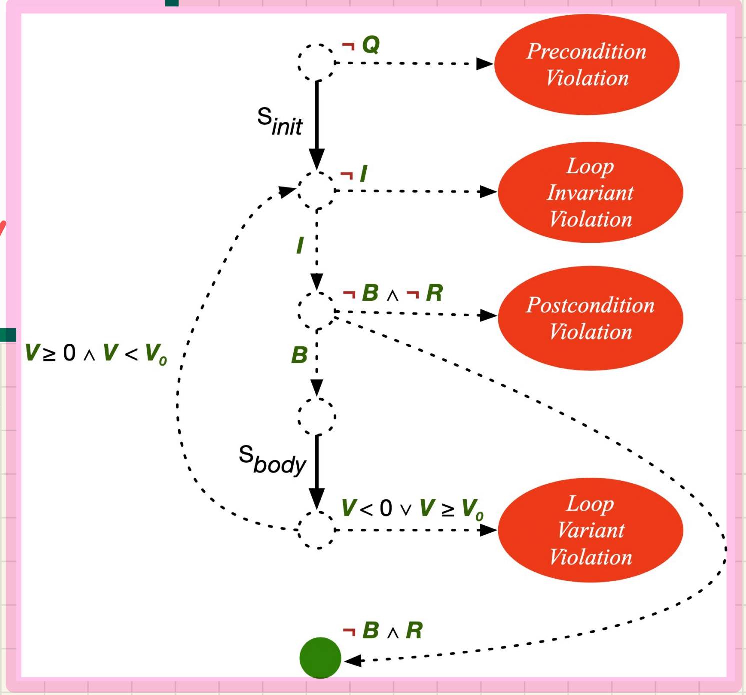
```
I(i) == (1 <= i) /\ (i <= 6)
V(i) == 6 - i
--algorithm loop_invariant_test
  variables i = 1, variant_pre = 0, variant_post = 0;
  {
    assert I(i);
    while (i <= 5) {
      variant_pre := V(i);
      i := i + 1;
      variant_post := V(i);
      assert variant_post >= 0;
      assert variant_post < variant_pre;
      assert I(i);
    } ;
  }
```

① precond: true

② Start is "i=1"

B

Runtime Checks



established

end of iteration	i	I	V	B
0 → Start →	1	T	5	T
1	2	T	4	T
2	3	T	3	T
3	4	T	2	T
4	5	T	1	T
5	6	T	0	F

maintained

① 5>4

②

dec

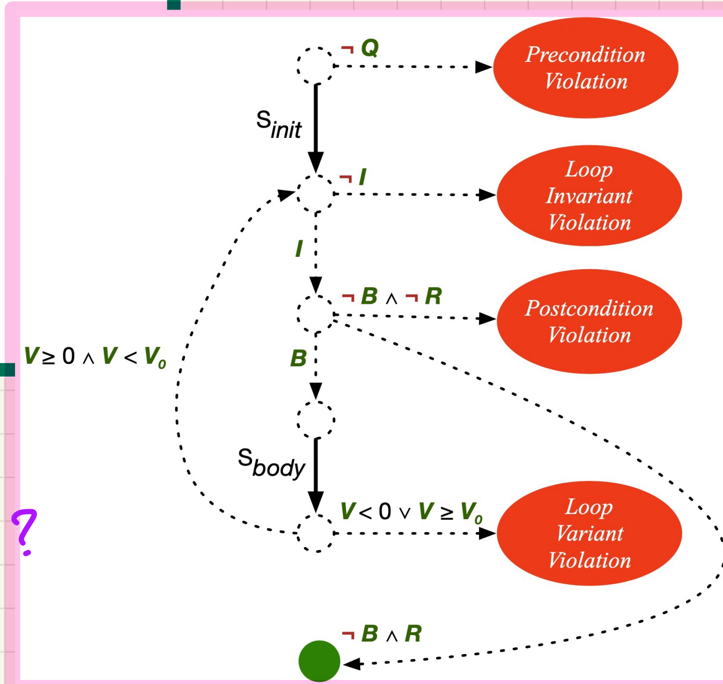
Contracts of Loops: Violations

Assume: Q and R are true

Specification

```
I(i) == (1 <= i) /\ (i <= 6)
V(i) == 6 - i
--algorithm loop_invariant_test
 variables i = 1, variant_pre = 0, variant_post = 0;
 {
  assert I(i);
  while (i <= 5) {
   variant_pre := V(i);
   i := i + 1;
   variant_post := V(i);
   assert variant_post >= 0;
   assert variant_post < variant_pre;
   assert I(i);
  } ;
 }
```

Runtime Checks



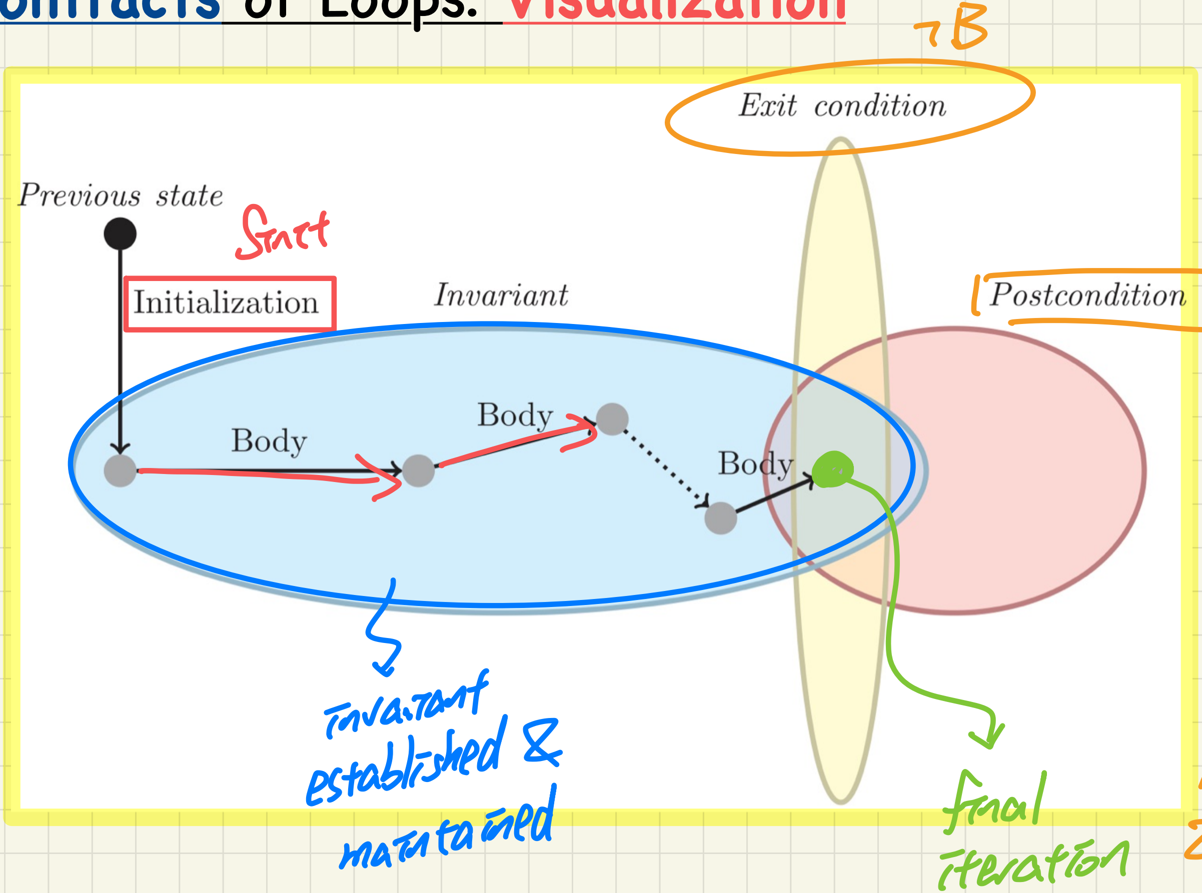
LI violation at the end of iteration 5 violation?

invariant: $1 <= i <= 5$ which iteration?

variant: $5 - i$

LV violation at the end of iteration 5

Contracts of Loops: Visualization



1. LI
2. ¬B
3. R

Lecture

Program Verification

Correctness Proofs of Loops

Correct Loops: Proof Obligations

- A loop is ***partially*** *correct* if:
  - Given precondition **Q**, the initialization step S_{init} establishes **LI** I.

    $\{Q\}\ S_{init}\ \{I\}$

    $\{Q\}\ S_{init}\ \{I\}$
  - At the end of S_{body}, if not yet to exit, **LI** I is maintained.

    $\{I \land B\}\ S_{body}\ \{I\}$

    $\{I \land B\}\ S_{body}\ \{I\}$
  - If ready to exit and **LI** I maintained, postcondition **R** is established.

    $\neg B \land I \Rightarrow R$

    $I \land \neg B \Rightarrow R$
- A loop ***terminates*** if:
  - Given **LI** I, and not yet to exit, S_{body} maintains **LV** V as non-negative.

    $\{I \land B\}\ S_{body}\ \{V \geq 0\}$

    $\{I \land B\}\ S_{body}\ \{V \geq 0\}$
  - Given **LI** I, and not yet to exit, S_{body} decrements **LV** V.

    $\{I \land B\}\ S_{body}\ \{V < V_0\}$

    $\{I \land B\}\ S_{body}\ \{V < V_0\}$

means variant_post at the end of iteration

```
{Q}
S_init
assert I(...);
while( B ) {
  variant_pre := V(...);
  S_body
  variant_post := V(...);
  assert variant_post >= 0;
  assert variant_post < variant_pre;
  assert I(...);
}
{R}
```

Correct Loops: Proof Obligations

Example

Specification

```
I(i) == (1 <= i) /\ (i <= 6)
V(i) == 6 - i
--algorithm loop_invariant_test
  variables i = 1, variant_pre = 0, variant_post = 0;
  {
    assert I(i);
    while (i <= 5) {
      variant_pre := V(i);
      i := i + 1;
      variant_post := V(i);
      assert variant_post >= 0;
      assert variant_post < variant_pre;
      assert I(i);
    } ;
  }
```

→ not to be included in

Start

$1 <= i \land$

① $\{True\}\ i := 1\ \{i \leq 6\}$

② $\{1 \leq i \land i \leq 6 \land i \leq 5\}$
$i := i + 1$
$\{1 \leq i \land i \leq 6\}$

③ $1 \leq i \land i \leq 6 \land \neg(i <= 5) \Rightarrow True$

④ $\{1 \leq i \land i \leq 6 \land i \leq 5\}\ i := i+1\ \{6 - i \geq 0\}$

⑤ $\{1 \leq i \land i \leq 6 \land i \leq 5\}\ i := i+1\ \{6 - i < 6 - i_0\}$

- A loop is ***partially*** *correct* if:
  - Given precondition **Q**, the initialization step S_{init} establishes **LI** I.
    ① $\boxed{\{Q\}\ S_{init}\ \{I\}}$
  - At the end of S_{body}, if not yet to exit, **LI** I is maintained.
    ② $\boxed{\{I \land B\}\ S_{body}\ \{I\}}$
  - ③ If ready to exit and **LI** I maintained, postcondition **R** is established.
    $\boxed{I \land \neg B \Rightarrow R}$
- A loop ***terminates*** if:
  - ④ Given **LI** I, and not yet to exit, S_{body} maintains **LV** V as non-negative.
    $\boxed{\{I \land B\}\ S_{body}\ \{V \geq 0\}}$
  - ⑤ Given **LI** I, and not yet to exit, S_{body} decrements **LV** V.
    $\boxed{\{I \land B\}\ S_{body}\ \{V < V_0\}}$

$\{Q\}$

loop

$\boxed{S_{extra}}$ assume: no loop

$\{R\}$

$wp(S_{extra}, R)$ -