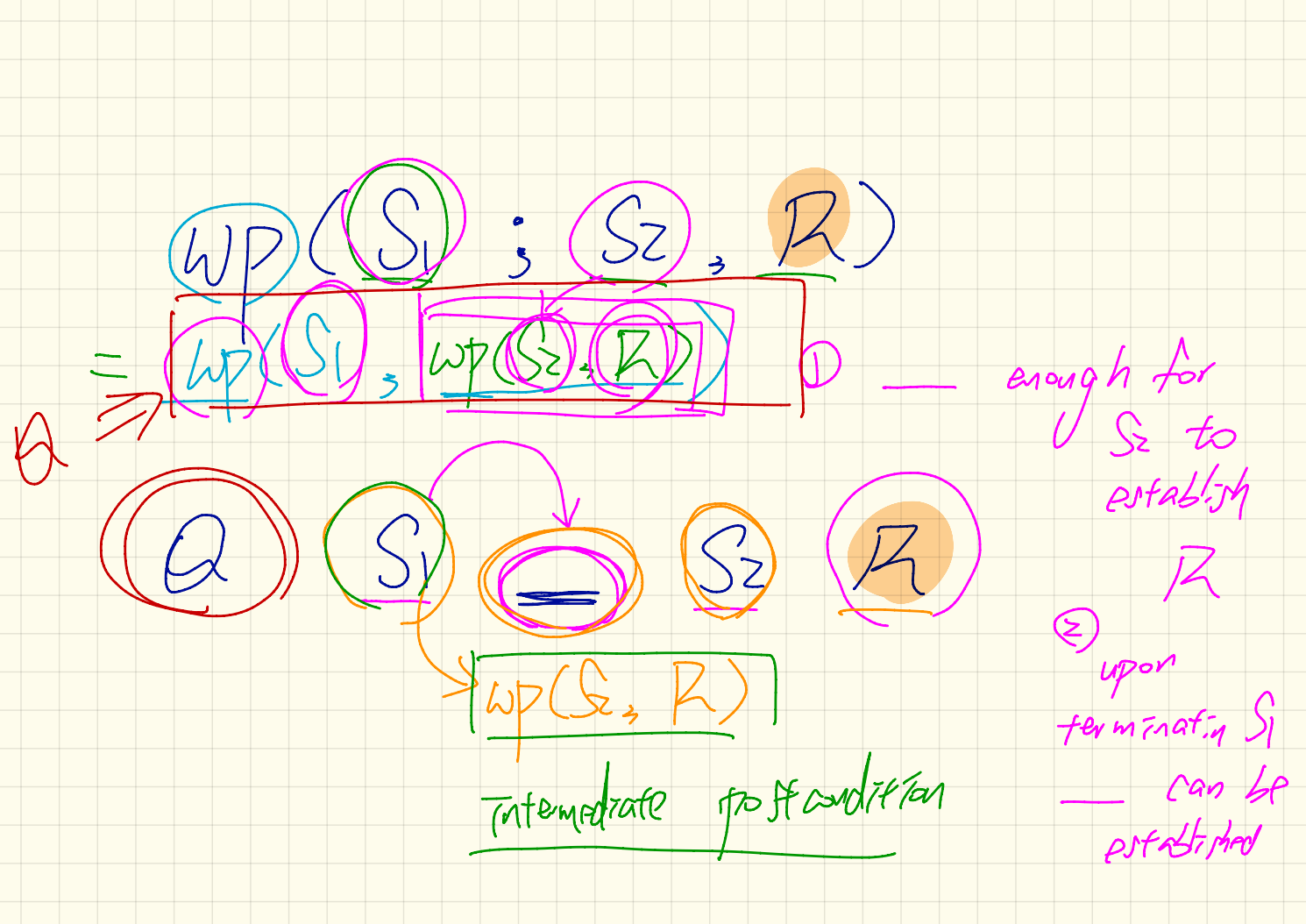
Friday April 5
Lecture 24


wp(S1 ; S2, R)
=
wp(S1, wp(S2, R))
①
—
enough for S2 to establish R
Q
S1
=
S2
R
wp(S2, R)
intermediate postcondition
②
upon terminating S1
—
can be established

Correctness of Program : Sequental Composition

Is $\{ \textbf{True} \}$ `tmp := x; x := y; y := tmp` $\{ x > y \}$ correct?

① calculate $wp(\underbrace{tmp := x}_{S_1}; \underbrace{x := y; y := tmp}_{S_2}, x > y)$

$= \{$ wp rule for ; seq. comp. $\}$

$wp(tmp := x, wp(\underbrace{x := y}_{S_1}; \underbrace{y := tmp}_{S_2}, x > y))$

$= \{$ wp rule for ; $\}$

$wp(tmp := x, wp(x := y, wp(y := tmp, x > y)))$

$= \{$ wp for := $\}$

$wp(tmp := x, wp(x := y, x > tmp))$

$= \{$ wp for := $\}$

$wp(tmp := x, y > tmp)$

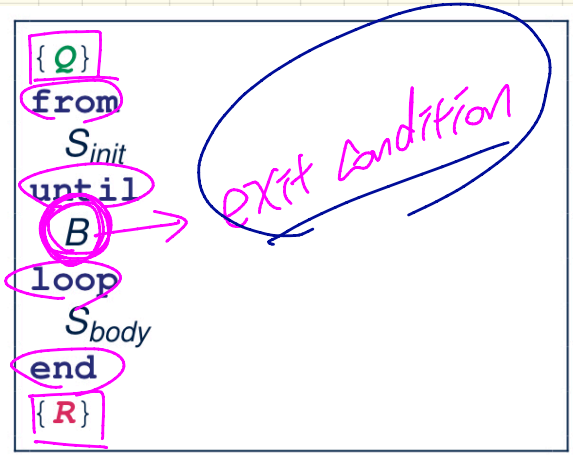
$= \{$ wp rule for := $\}$

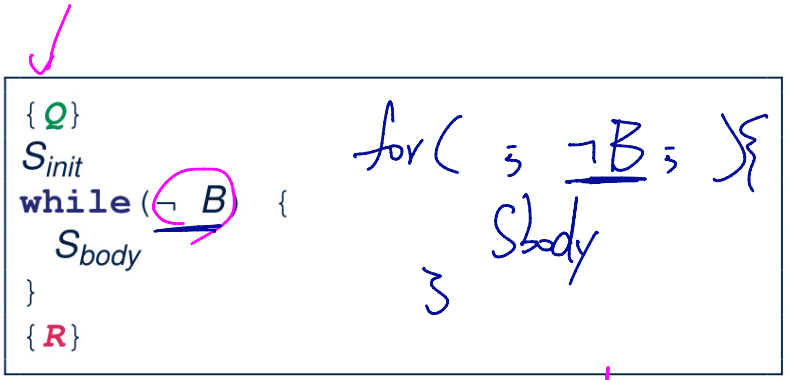
$y > x$

② counter ex $y = 1, x = 2$ ⇒ True ⇒ $x > y$

Swap without introducing an intermediate variable.

Loops: Eiffel vs. Java





from
until
not (x ≥ 0)
loop
end

x ≤ 0

while (x ≥ 0){
[...
}

stay condition

Contracts of Loops

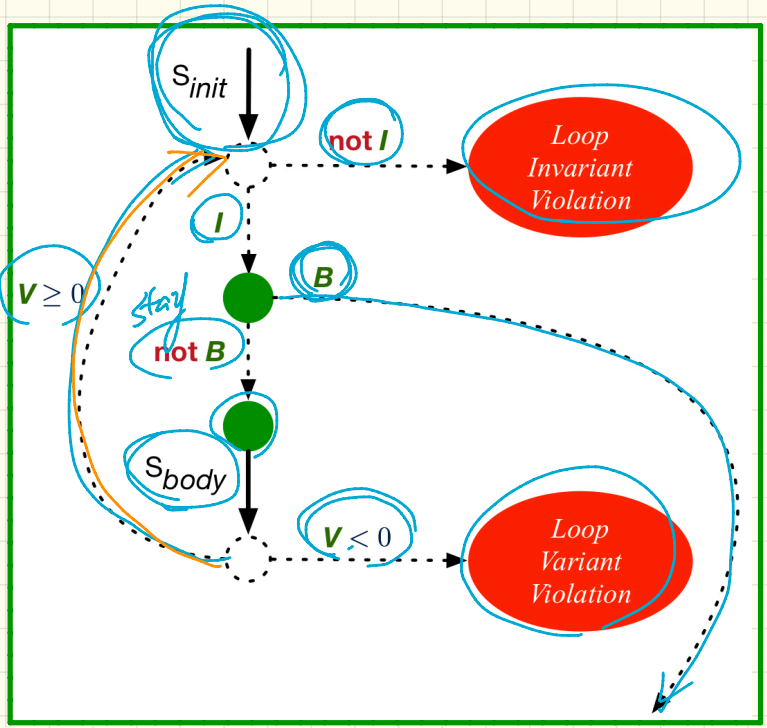
Syntax

```
from
  S_init
invariant
  invariant_tag: I
until
  B
loop
  S_body
variant
  variant_tag: V
end
```

established

maintained

Runtime Checks



Contracts of Loops: Example

Example

```
test
  local
    i: INTEGER
  do
    from
      i := 1
    invariant
      1 <= i and i <= 6
    until
      i > 5
    loop
      io.put_string ("iteration " + i.out
      i := i + 1
    variant
      6 - i
    end
end
```

1 > 5 F
2 > 5 F
6 > 5 T

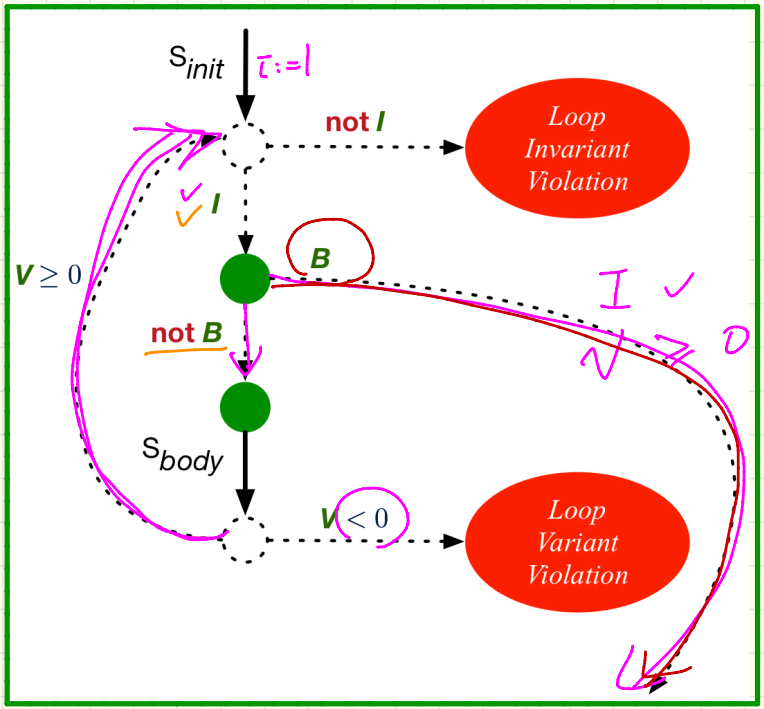
6 - 2 = 4
6 - 6 = 0

i

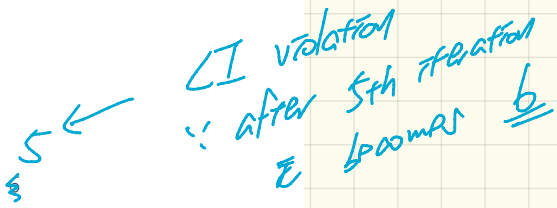
Iteration 1
Iteration 2
3
4
5

6

Runtime Checks



```
test
  local
    i: INTEGER
  do
    from
      i := 1
    invariant
      1 <= i and i <= 5
    until
      i > 5
    loop
      io.put_string ("iteration " + i.out
      i := i + 1
    variant
      6 - i
    end
end
```

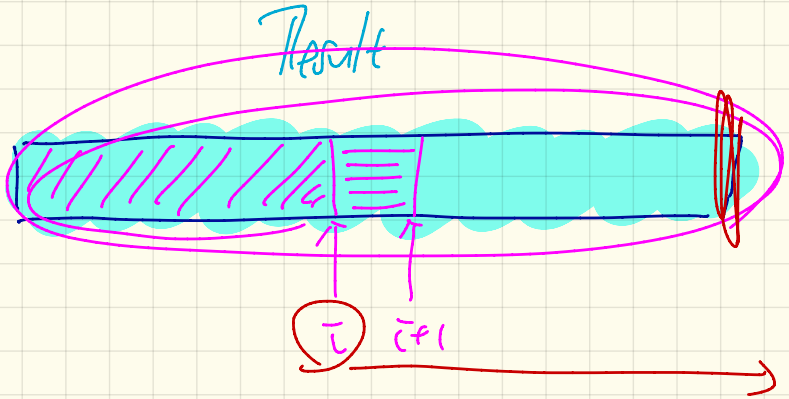


```
test
  local
    i: INTEGER
  do
    from
      i := 1
    invariant
      1 <= i and i <= 6
    until
      i > 5
    loop
      io.put_string ("iteration " + i.out
      i := i + 1
    variant
      5 + 1 - i
    end
end
```

5th iteration

i becomes 6

5 - 6 = -1 LV violation.


Result
i
i+1
Result

Contracts of Loops: Violations

Example

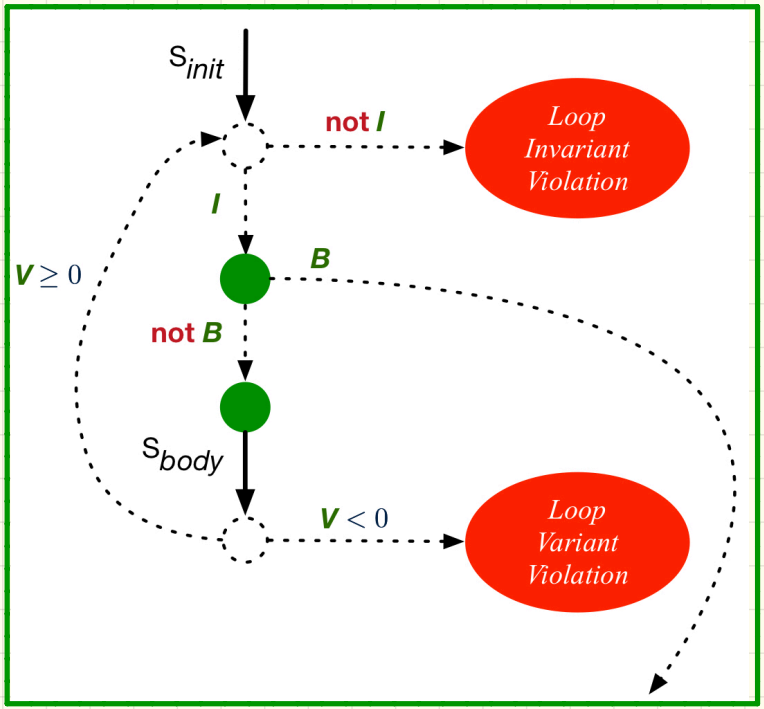
```
test
  local
    i: INTEGER
  do
    from
      i := 1
    invariant
      1 <= i and i <= 6
    until
      i > 5
    loop
      io.put_string ("iteration " + i.out
      i := i + 1
    variant
      6 - i
    end
end
```

Invariant Violation : $1 \leq i \leq 5$

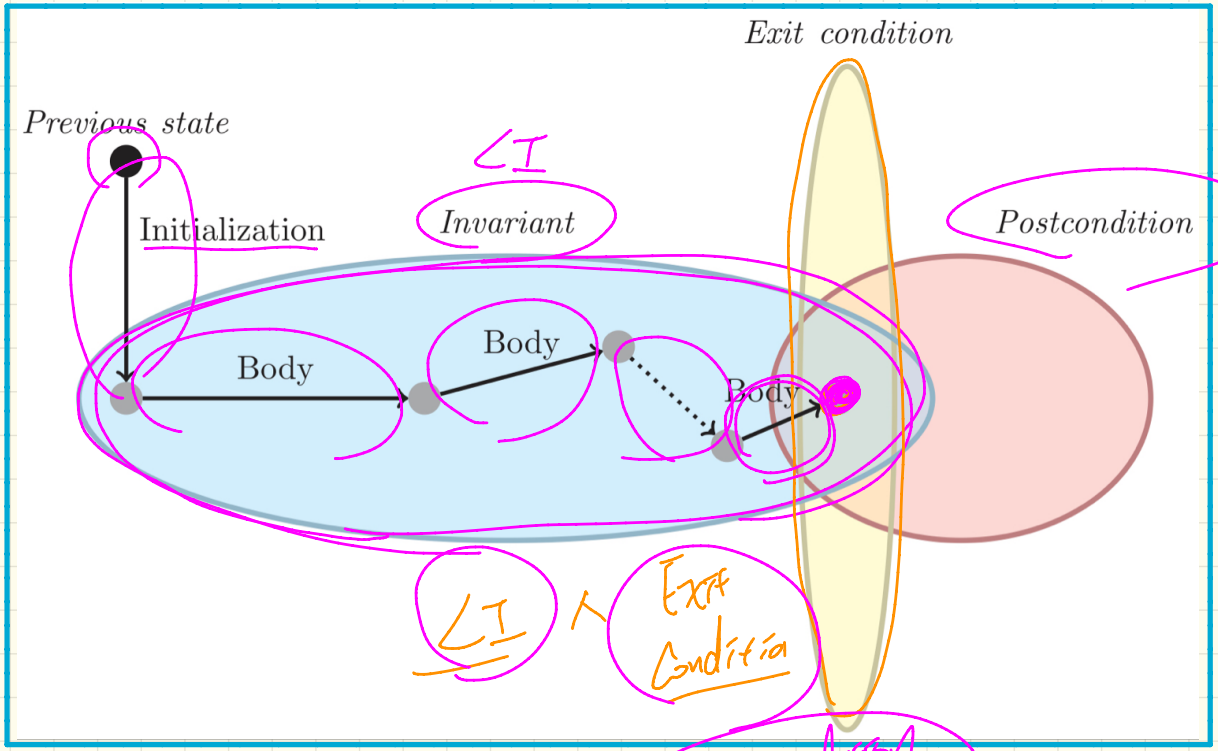
Variant Violation: $5 - i$

Skipping Loop Body : $i > 0$

Runtime Checks

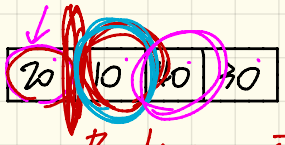


Contracts of Loops: Visualization



Finding Max : V1

```
find_max (a: ARRAY [INTEGER]): INTEGER
  local i: INTEGER
  do
    from
      i := a.lower; Result := a[i]
    invariant
      loop_invariant: -- ∀j | a.lower ≤ j ≤ i • Result ≥ a[j]
      across a.lower |..| i as j all Result >= a [j.item] end
    until
      i > a.upper
    loop
      if a [i] > Result then Result := a [i] end
      i := i + 1
    variant
      loop_variant: a.upper - i + 1
    end
  ensure
    correct_result: -- ∀j | a.lower ≤ j ≤ a.upper • Result ≥ a[j]
    across a.lower |..| a.upper as j all Result >= a [j.item]
  end
end
```



$\forall j \mid a.lower \le j \le i \bullet Result \ge a[j]$

i only to be considered in the coming iteration

i = 1, Result = 20

$\forall j \mid 1 \le j \le 1 \bullet 20 \ge a[j]$

$20 \ge a[1]$

i = 2, Result = 20

$\forall j \mid 1 \le j \le 2 \bullet 20 \ge a[j]$

$20 \ge a[1]$

$20 \ge a[2]$

i = 3, Result = 20

$\forall j \mid 1 \le j \le 3 \bullet 20 \ge a[j]$

$20 \ge a[1]$ $20 \ge a[2]$

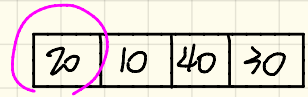
LI ✓

After Iteration	i	Result	LI	Exit (i > a.upper)?	LV
Initialization	1	20	✓	×	–
1st	●	●	●	●	●
2nd	●	●	●	●	●

Finding Max : v2

```
find_max (a: ARRAY [INTEGER]): INTEGER
  local i: INTEGER
  do
    from
      i := a.lower ; Result := a[i]
    invariant
      loop_invariant: -- ∀j | a.lower ≤ j < i • Result ≥ a[j]
        across a.lower |..| (i - 1) as j all Result >= a [j.item] end
    until
      i > a.upper
    loop
      if a [i] > Result then Result := a [i] end
      i := i + 1
    variant
      loop_variant: a.upper - i
    end
  ensure
    correct_result: -- ∀j | a.lower ≤ j ≤ a.upper • Result ≥ a[j]
      across a.lower |..| a.upper as j all Result >= a [j.item]
  end
end
```

i | Result
1 | 20



F

$\forall j \mid 1 \le j \le 0 \bullet$

$20 \ge A[j]$.

After Iteration	i	Result	LI	Exit (*i* > *a.upper*)?	LV
Initialization	1	20	✓	×	–
1st	2	20	✓	×	2
2nd	3	20	✓	×	1
3rd	4	40	✓	×	0
4th	●	●	●	●	●

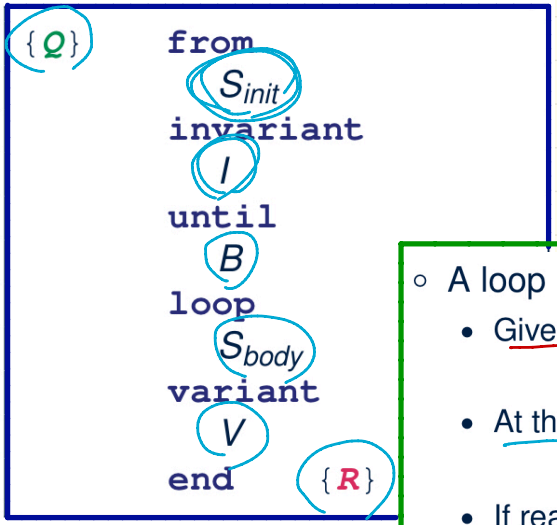
$$\forall x \mid \underline{\underline{F}} \cdot P(x) \qquad T$$

$$\forall x \mid R(x) \cdot P(x)$$

$$\equiv \forall x \cdot \underline{\underline{R(x)}} \Rightarrow P(x)$$

$$F$$

Proof Obligations for Correct Loops



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

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

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

    $$I \wedge B \Rightarrow R$$

- A loop *terminates* if:
  - Given ***LI*** *I*, and not yet to exit, S_{body} maintains ***LV*** *V* as non-negative.

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

    $$\{I \wedge \neg B\}\ S_{body}\ \{V < V_0\}$$