Lecture 25
Wednesday April 1

Hoare Triple as a Predicate

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp(S, R)$$

prog is f

mandatory req. Q

do S

ensure R

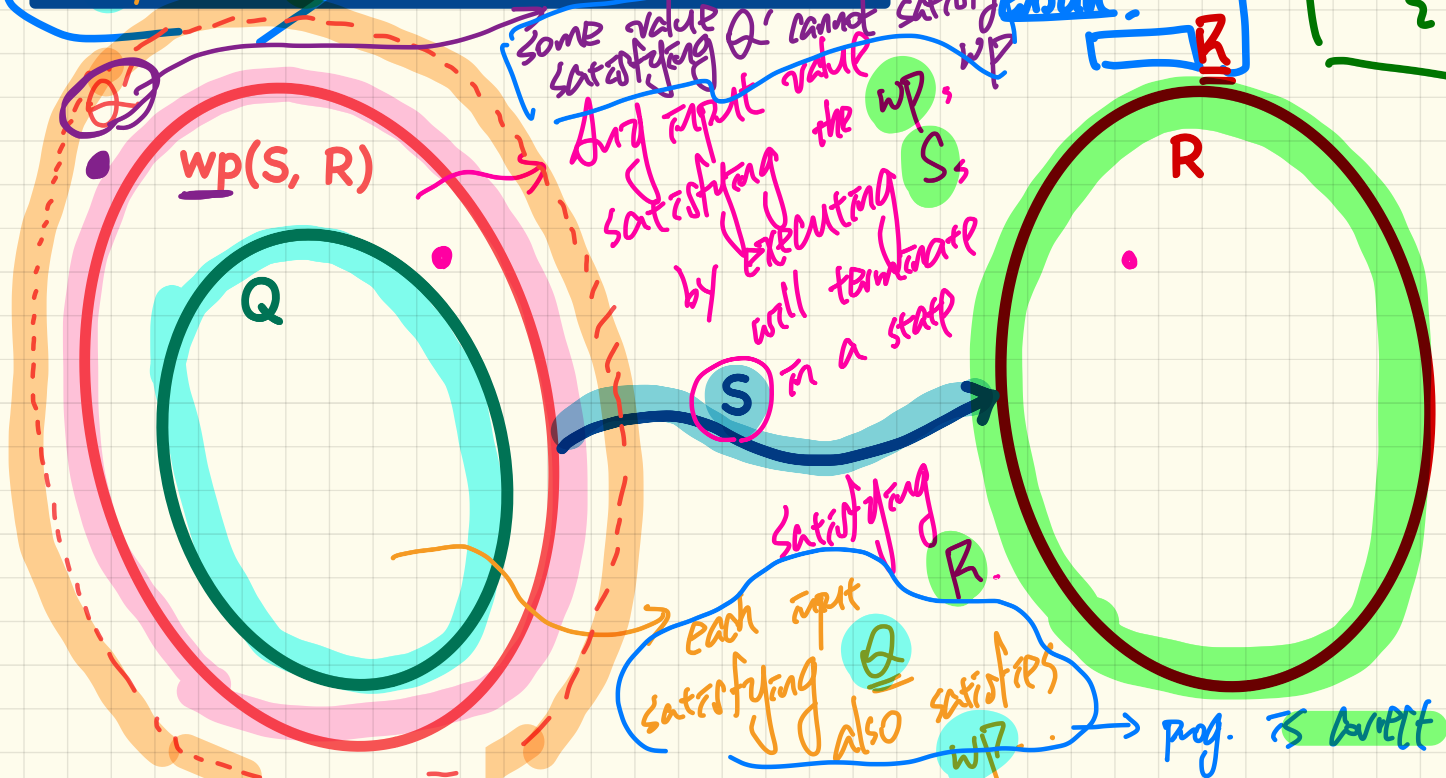
BOOLEAN (PREDICA)

{ } _ { }

some value satisfying Q' cannot satisfy wp

Any input value satisfying the wp, S, by executing S will terminate in a state satisfying R.

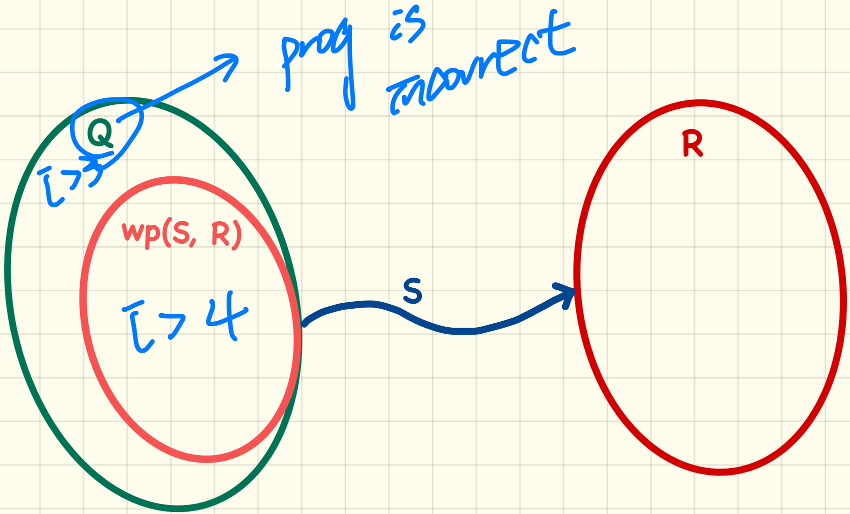
each input satisfying Q also satisfies wp. → prog. is correct.

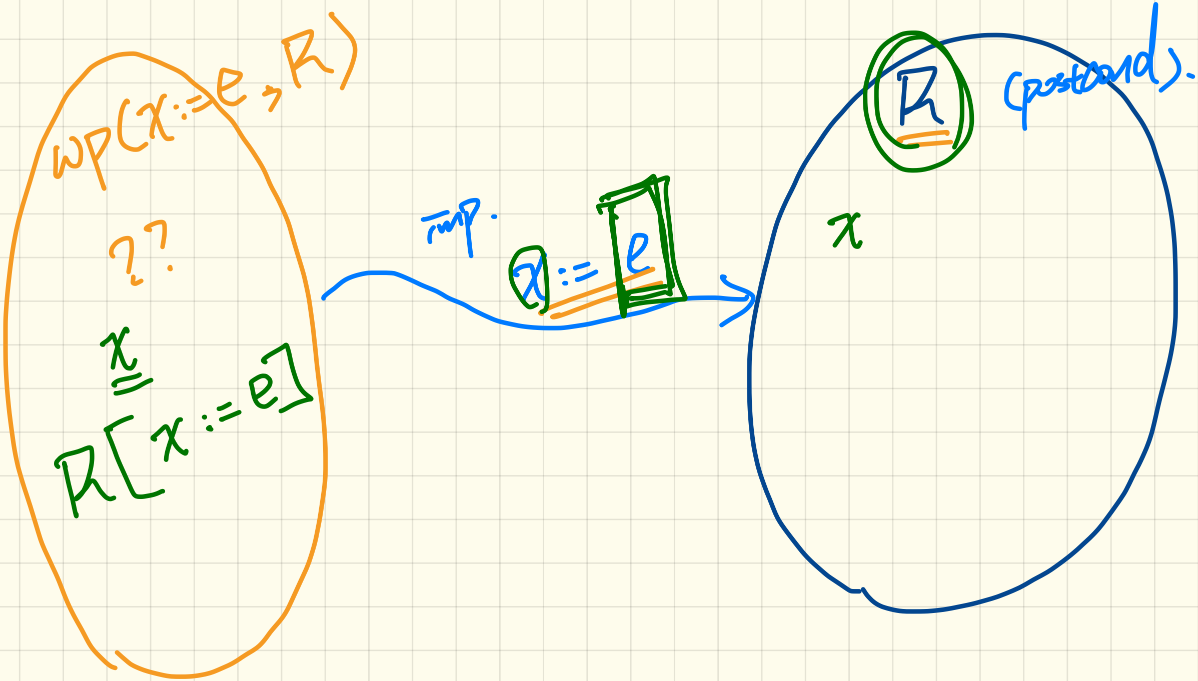


Program Correctness: Revisiting Example (1)

```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 3
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp(S, R)$$




wp(x := e, R)
??
x
R[x := e]
Imp.
x := e
x
R
(postcond).

Correctness of Programs: Assignment (2)

What is the weakest precondition for a program `x := x + 1` to establish the postcondition ~~$x > x_0$~~ $x = 23$?

$\{??\}$ `x := x + 1` $\{x = 23\}$

$$wp(x := x+1,\ x = 23)$$

$$= \{\text{def. of } wp \text{ for } :=\}$$

$$\boxed{x} = 23\,[x := x+1]$$

$$= x + 1 = 23 \equiv \boxed{x = 22}$$

$x = 22$

x

prog not correct

$x = 21$

Rules of Weakest Precondition: Conditionals

wp(**if** B **then** S1 **else** S2 **end**, R) ??

if B then

[S_1

else

[S_2

end

{R}

$B \Rightarrow wp(S_1, R)$

① $\wedge$ ② $\vee$??

if : –

$\neg B \Rightarrow wp(S_2, R)$

else : –

y=1,x=-4

Rules of Weakest Precondition: Conditionals

wp(if B then S1 else S2 end, R)

B ⇒ wp(S1, R)
∨
¬B ⇒ wp(S2, R)

incorrect

wp will just be T

vs.

B ⇒ wp(S1, R)
∧
¬B ⇒ wp(S2, R)

correct

if branch establishes R and else branch establishes R

??

should this program be correct?

post cond. violation

x ≥ 0

Consider:

wp(if y > 0 then x := x + 1 else x := x − 1 end, x >= 0)

1>0 T ⇒

$y>0 \Rightarrow wp(x:=x+1,\ x \geq 0)$

∨

$y \leq 0 \Rightarrow wp(x:=x-1,\ x \geq 0)$

$1 \leq 0$ F ⇒ ___ ≡ T

wp should not evaluate to T if an input value can violate postcond.

mistakenly, wp will still evaluate to T

y=1 x=−4

Rules of Weakest Precondition: Summary

$$wp(\mathtt{x} \ := \ \mathtt{e}, \ \boldsymbol{R}) = \boldsymbol{R}[x := e]$$

$$wp(\mathbf{if}\ \boxed{B}\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end},\ \boldsymbol{R}) = \begin{pmatrix} \boxed{B} \Rightarrow wp(S_1,\ \boldsymbol{R}) \\ \wedge \\ \neg \boxed{B} \Rightarrow wp(S_2,\ \boldsymbol{R}) \end{pmatrix}$$

$$wp(S_1 \ ; \ S_2,\ \boldsymbol{R}) = wp(S_1,\ wp(S_2,\ \boldsymbol{R}))$$

Proof Rules using **Weakest Precondition**

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp(S, R)$$

2 ways to proving correctness

① prove

$Q \Rightarrow wp(\_\_\_\_, R)$

② $\{Q \wedge B\}\ S_1\ \{R\}$

$\wedge$

$\{Q \wedge \neg B\}\ S_2\ \{R\}$

Q

if B

S_1

else

S_2

$\{R\}$

$$\{Q\}\ x := e\ \{R\} \iff Q \Rightarrow \underbrace{R[x := e]}_{wp(x\ :=\ e,\ R)}$$

$$\{Q\}\ \texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \texttt{end}\ \{R\}$$

$$\iff \left(\begin{array}{l}\{\ Q \wedge B\ \}\ S_1\ \{\ R\ \} \\ \wedge \\ \{\ Q \wedge \neg B\ \}\ S_2\ \{\ R\ \}\end{array}\right) \iff \left(\begin{array}{l}(Q \wedge B) \Rightarrow wp(S_1, R) \\ \wedge \\ (Q \wedge \neg B) \Rightarrow wp(S_2, R)\end{array}\right)$$

$$\{Q\}\ S_1\ ;\ S_2\ \{R\} \iff Q \Rightarrow \underbrace{wp(S_1, wp(S_2, R))}_{wp(S_1\ ;\ S_2,\ R)}$$

Correctness of Programs: Conditionals

Is this program correct?

$\{x > 0 \wedge y > 0\}$

```
if x > y then
  bigger := x ; smaller := y
else
  bigger := y ; smaller := x
end
```

$\{bigger \geq smaller\}$

$\rightarrow S$

$x \le y$

To prove, follow 2 steps.

① calculate $wp(S, b \geq s)$

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

$x > y \Rightarrow wp(b := x ; s := y, b \geq s)$

$x \le y \Rightarrow wp(b := y ; s := x, b \geq s)$

② Prove:

$\boxed{x > 0 \wedge y > 0} \Rightarrow \boxed{??}$

$wp(S_1, wp(S_2, R))$

$$wp(\underline{\underline{S_1}}\ ;\ S_2, R)$$

$\perp$

$\downarrow$

?

$\underline{\underline{S_1}}$; S_2 $\underline{R}$ $\cancel{x}$

$wp(S_2, R)$

S_1 terminates before S_2 is executed

$$wp(\underline{S_1}, wp(\underline{S_2}, \underline{R}))$$

phase 2

Correctness of Programs: Sequential Composition

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

① Step1: Calculate $wp(tmp := x; x := y; y := tmp, x > y)$

$= \{$ def. of wp for ; $\}$

$wp(tmp := x, wp(x := y; \boxed{y := tmp, x > y}))$

$= \{$ def. of wp for ; $\}$

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

$= \{$ def. of wp for := $\}$

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

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

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

$= \{$ def. of wp for := $\}$ $y > x$

Prove

② True $\Rightarrow y > x$

$= \{$ identity of $\Rightarrow \}$

$\boxed{y > x}$

$\boxed{\text{not}}$ a tautology (theorem)

counterexample: any x, y satisfying $\neg(y > x)$ e.g. $y = 3$, $x = 4$

Loops: Eiffel vs. Java

```
{Q}
from
  S_init
until
  B
loop
  S_body
end
{R}
```

exit condition

```
{Q}
S_init
while (¬ B) {
  S_body
}
{R}
```

stay condition

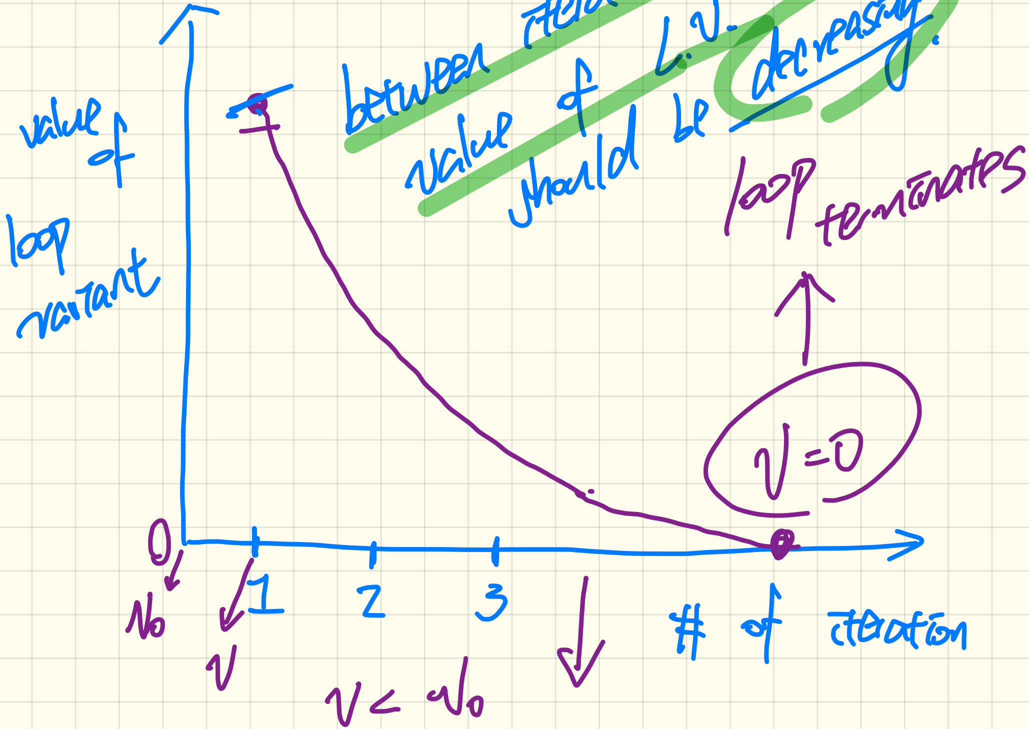
```
from
  i := 1
until
  i = 10
loop print(i)
  i := i+1
end
```

1~9

```
int i = 1;
while ( ¬(i = 10) ) {
  print(i)
  i++;
}
```

Loop Variant

between iterations, value of L.V. should be **decreasing**.



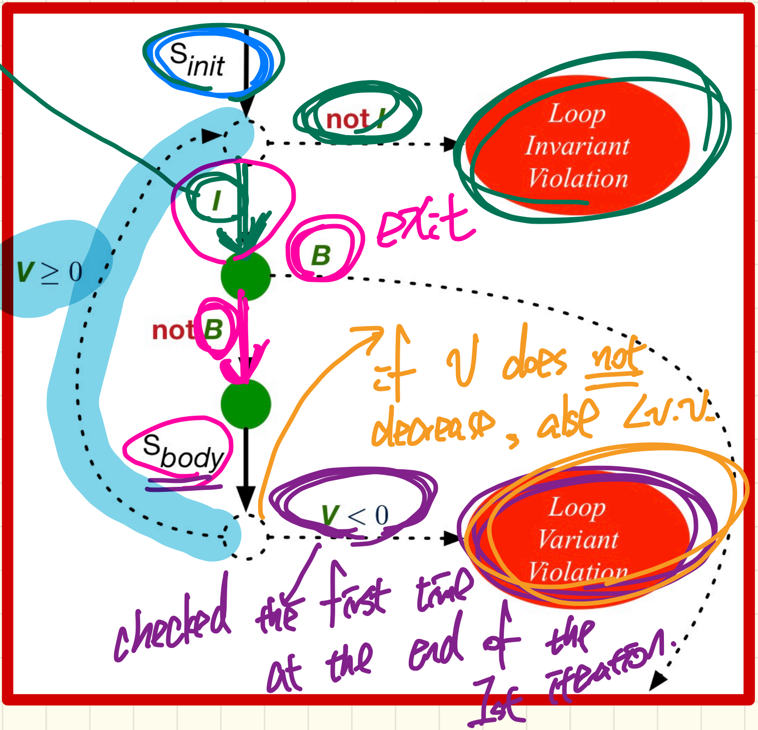
Contracts of Loops

Syntax

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

checked for the first time before the 1st iteration.

Runtime Checks



Contracts of Loops: Example

Syntax

```
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
```

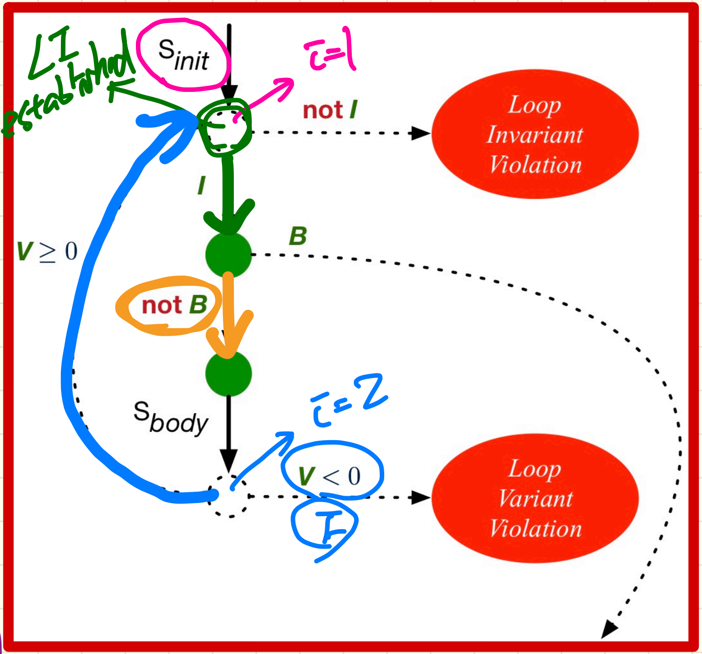
i > 5
as soon as i gets to 6, we exit.

6 - 2 = 4.

the last time LV is checked.
6 - 6 = 0.

i = 1
2
3
4
5 || 6 exit

Runtime Checks

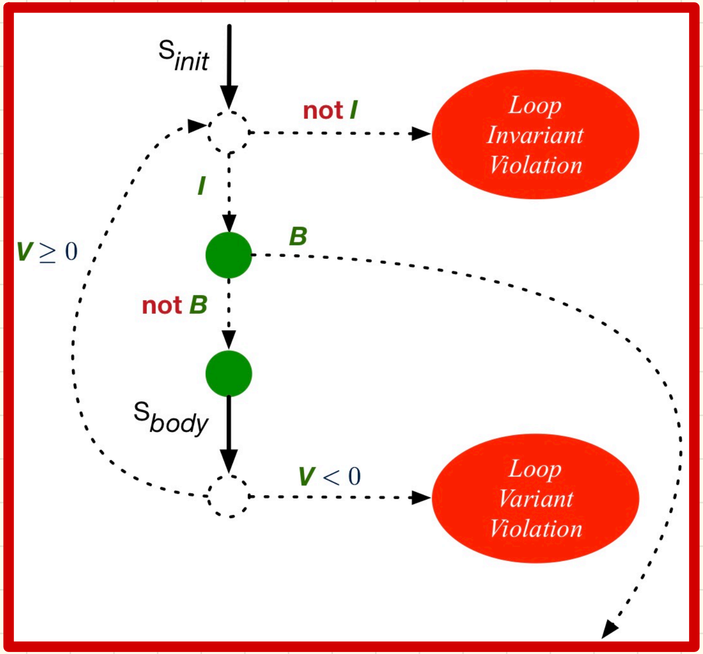


Contracts of Loops: Violations

Syntax

```
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
```

Runtime Checks



exit condition: i > 0

invariant: 1 <= i <= 5

variant: 5 - i

Contracts of Loops: Visualization