The Invariant Game

Matteo Vaccari
m.vaccari@sourcesense.com
sourcesense

XP Day Eindhoven, 20 November 2008

Why?

Technical excellence enhances agility!

What?

Robert W. Floyd, *Assigning Meanings to Programs*, 1967
C.A.R. Hoare, *An Axiomatic Basis for Computer Programming*, 1969
E.W. Dijkstra, *A Discipline of Programming*, 1976

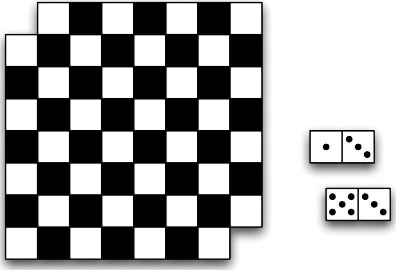
$\vdots$

Right here in Eindhoven!

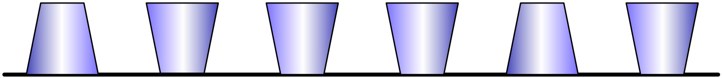
$\vdots$

Roland Backhouse, *Program Construction*, 2003

The case of the missing squares

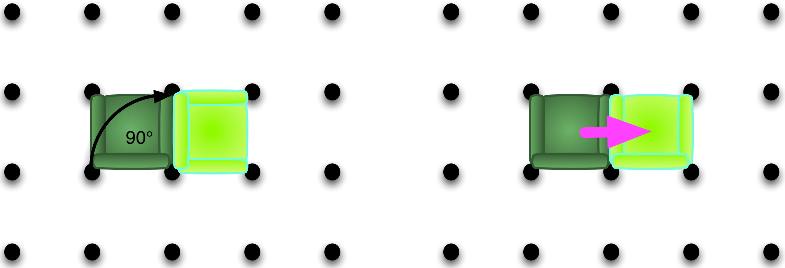


The case of the upside-down tumblers



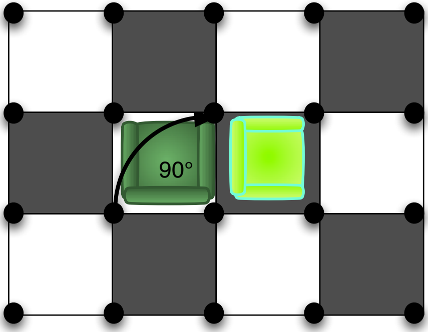
The case of the heavy armchair

?



from R. Backhouse, *Program Construction*

The case of the heavy armchair



Hoare triples

$$\{ P \} \, S \, \{ Q \}$$

if precondition P is true,
then program S will terminate,
and then postcondition Q will be true.

$$\begin{array}{ccc} \{ \text{true} \} & x := 42 & \{ x = 42 \} \\ \{ x = 3 \} & x := x + 1 & \{ x = 4 \} \\ \{ x > 0 \} & x := x + 1 & \{ x > 1 \} \end{array}$$

Triples as program specs

"Find program S that establishes Q starting from P"

$$\{\ P\ \}\ S\ \{\ Q\ \}$$

Example: spec for "the square root of x"

$$\{\ x \geq 0\ \}\ S\ \{\ |y^2 - x| < \epsilon\ \}$$

(Informally: x is given; the program should assign to y)

Loops

```
R          # initialization
while B    # guard
  S        # body
end
```

Solving problems with loops

```
{ P }            # precondition
R
while B
  S
end
{ Q }            # postcondition
```

How to find R, B, S ?

Strategy for solving loops (i)

Find predicate inv such that:

```
{ P }
R                         #   i. it can be established
{ inv }                   #      initially
while B
  S
end
{ Q }
```

Strategy for solving loops (ii)

Find predicate inv such that:

```
{ P }
R                               #   i. it can be established
{ inv }                         #      initially
while B                         #  ii. it's preserved by
  { B && inv } S { inv }        #      the loop body
end
{ Q }
```

Strategy for solving loops (iii)

Find predicate inv such that:

```
{ P }
R                              #   i. it can be established
{ inv }                        #      initially
while B                        #  ii. it's preserved by
  { B && inv } S { inv }       #      the loop body
end
{ !B && inv }                  # iii. at loop termination, it
{ Q }                          #      implies the postcondition
```

Example: sum the elements of an array

var a: array $[0, N)$ of integer

$\{\ \mathit{true}\ \}\ S\ \{\ s = \mathit{sum}[0, N)\ \}$

Where

$$\mathit{sum}[0, N) = (\Sigma : 0 \le i < N : a[i])$$

What is the idea?

Spec: $\{\ true\ \}\ S\ \{\ s = sum[0, N)\ \}$

0 N

summed

What is the idea?

Spec: $\{\ \mathsf{true}\ \}\ S\ \{\ s = \mathsf{sum}[0, N)\ \}$

0 N

summed

Introduce a new variable k

What is the idea?

Spec: $\{ \text{true} \}\ S\ \{ s = \mathrm{sum}[0, N) \}$

0	N
summed	

Introduce a new variable k

Invariant: $s = \mathrm{sum}[0, k)$

0	k	N
summed	not summed	

Does the invariant imply the postcondition?

Invariant: $s = sum[0, k)$

0 k N

summed	not summed

Does the invariant imply the postcondition?

Invariant: $s = sum[0, k)$

0 $\qquad$ k $\qquad$ N

summed	not summed

Yes! when $k = N$,

$$s = sum[0, k) = sum[0, N)$$

Does the invariant imply the postcondition?

Invariant: $s = sum[0, k)$

0	k	N
summed	not summed	

Yes! when $k = N$,

$$s = sum[0, k) = sum[0, N)$$

The shape of the loop:

while $k \neq N$
    S
    $k := k + 1$
end

Can we establish the invariant initially?

Invariant: $s = sum[0, k)$

0	k	N
summed	not summed	

Can we establish the invariant initially?

Invariant: $s = sum[0, k)$

0 $\qquad\qquad\qquad$ k $\qquad\qquad\qquad\qquad$ N

summed	not summed

Yes! when $k = 0$,

$$s = sum[0, 0) = 0$$

Can we establish the invariant initially?

Invariant: $s = sum[0, k)$

0 $\qquad\qquad$ k $\qquad\qquad$ N

summed	not summed

Yes! when $k = 0$,

$$s = sum[0, 0) = 0$$

The initial statement is

$$s := 0; k := 0$$

Can we preserve the invariant?

Invariant: $s = sum[0, k)$

$0 \qquad\qquad\qquad k \qquad\qquad\qquad\qquad N$

summed	not summed

$$
\begin{array}{l}
s := 0; k := 0 \\
\text{while } k \neq N \\
\quad s := E \\
\quad k := k + 1 \\
\text{end}
\end{array}
$$

Can we preserve the invariant?

Invariant: $s = sum[0, k)$

0	k	N
summed	not summed	

$s := 0; k := 0$

while $k \neq N$

  $s :=$ E

  $k := k + 1$

end

Observe:

$sum[0, k+1) = sum[0, k) + a[k]$

Can we preserve the invariant?

Invariant: $s = \mathrm{sum}[0, k)$

0 summed	k not summed N

$s := 0; k := 0$

while $k \neq N$

$s := E$

$k := k + 1$

end

Yes! by choosing $E = s + a[k]$

$s := 0; k := 0$

while $k \neq N$

$s := s + a[k]$

$k := k + 1$

end

Observe:

$\mathrm{sum}[0, k+1) = \mathrm{sum}[0, k) + a[k]$

So, the standard solution is *correct* :-)

$\{ \text{true} \}$

$s := 0; k := 0$

$\{ \text{sum}[0, 0) = 0 \}$

while $k \neq N$

$\quad \{ s = \text{sum}[0, k) \wedge k \neq N \}$

$\quad s := s + a[k]$

$\quad k := k + 1$

$\quad \{ \text{sum}[0, k + 1) = \text{sum}[0, k) + a[k] \}$

$\quad \{ s = \text{sum}[0, k) \}$

end

$\{ k = N \wedge s = \text{sum}[0, k) \}$

$\{ s = \text{sum}[0, N) \}$

Warmup 0: assign 0 to all elements of an array

Example: given $[1, 2, 3, 4]$, return $[0, 0, 0, 0]$
Spec as a pic?
Spec as a formula?
Invariant?
Implementation?

Warmup 1: make a random permutation of an array

Example: given $[1, 2, 3, 4]$, return (for instance) $[2, 4, 3, 1]$

Spec as a pic?

Spec as a formula?

Invariant?

Implementation?

Warmup 2: Separate odd and even numbers

Rearrange an array in place so that the even values are to the left and the odd values to the right.
Examples:

- input [], output []
- input [1,2,3,4,5], output [2,4,1,3,5]

Spec as a pic?
Spec as a formula?
Invariant?
Implementation?

Let's play now!

Rules of the game:

- Split in teams

Let's play now!

Rules of the game:

- Split in teams
- For every problem, you must produce the spec (pic and formula), the invariant (pic and formula), and the implementation

Let's play now!

Rules of the game:

- Split in teams
- For every problem, you must produce the spec (pic and formula), the invariant (pic and formula), and the implementation
- Every problem solved earns you points! (o:

Let's play now!

Rules of the game:

- Split in teams
- For every problem, you must produce the spec (pic and formula), the invariant (pic and formula), and the implementation
- Every problem solved earns you points! (o:
- You may ask for help, but that will cost you points!)o:

Let's play now!

Rules of the game:

- Split in teams
- For every problem, you must produce the spec (pic and formula), the invariant (pic and formula), and the implementation
- Every problem solved earns you points! (o:
- You may ask for help, but that will cost you points!)o:
- You must convince your opponents that your solution is valid

Let's play now!

Rules of the game:

- Split in teams
- For every problem, you must produce the spec (pic and formula), the invariant (pic and formula), and the implementation
- Every problem solved earns you points! (o:
- You may ask for help, but that will cost you points!)o:
- You must convince your opponents that your solution is valid
- Team with largest score wins!

For more information

Two books by Roland Backhouse:

- *Algorithmic Problem Solving*
- *Program Construction*

Thank you. Any questions?