

# Verification & Testing

## Hoare Logic

Roderick Bloem  
IAIK

# Today

- Undecidability
- Manual proofs with Hoare Logic

# Motivation

Proving correctness of programs is undecidable

- You can do it only by hand
- Model checking does not (always) work: infinite state space

Hoare logic: notation plus set of rules that allows you to prove programs correct by hand.

- We use very simple version: no function calls, no mallocs, etc

We will use Hoare logic later to compute *abstractions*

# Interlude: The Meta Game

Something is a *game* if (and only if) it fulfills the following:

1. It has two players, A and B
2. A starts, turns alternate
3. always ends (in win or draw)

Example: tic-tac-toe, connect-four, but *not* chess

The “meta-game,” played by two players

Turns (A starts):

1. Player picks a game,
2. Play the game (other player starts),
3. add one to score of winner (if draw, point for player who did not choose.)

Alternate turns until one player has 5 points

Is the meta-game a game?

# More Paradoxes

$S = \{A \mid A \notin A\}$  (All sets that do not contain themselves)

The Barber's paradox

# The Halting Problem

Does this program halt?

```
int main() {
    BigInt i;
    i << cin; // cin > 0
    while(i != 1) {
        if(i is even)
            i = i/2;
        else
            i = 3*i + 1;
    }
}
```

# Halting Problem

## Halting problem is undecidable:

There is no program  $H(G)$  that decides, given a program  $G$ , whether it halts

- This holds for programs without input, for programs with a fixed input, for the question whether the programs holds for all inputs, etc.

## Proof sketch:

- Suppose there is an algorithm  $H$  with as input a program  $P$  that outputs true iff  $P$  halts (on all inputs)
- Take this program: `weird() { if (H(weird)) while(1); }`
- Is  $H(\text{weird})$  true or false?
- There is no correct implementation for  $H$ !

# Reduction

Problem  $A$  *reduces to* problem  $B$  if you can use an algorithm for  $B$  to solve  $A$

- If  $B$  is decidable, so is  $A$
- If  $A$  is not decidable, neither is  $B$

More undecidable problems:

- Can  $G$  reach location  $l$ ?
- Can  $G$  reach location  $l$  with  $d=0$ ?
- In  $G$ , can  $d$  ever be 0?

The halting problem *reduces to* these problems.

- For instance,  $R(G,l) = \text{“can } G \text{ reach location } l\text{”}$  can be used to solve the halting problem
- $H(G) = R(G,l)$  where  $l$  is the last line in the program

# Ways Out

- Don't prove correctness
- Incomplete Verification
  - Closing the program by providing inputs (test, JPF)
  - Abstraction and refinement (SLAM, BLAST)
  - Verify only *some* programs
- Manual proof using Hoare Logic

# Hoare Logic

A **Hoare triple**:

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

P is the precondition

Q is the postcondition

S is a program

**Meaning:** if P holds before execution and S finishes, then Q holds afterwards.

Note: we prove **partial correctness**. If S runs forever,  $\{P\}S\{Q\}$  holds.

Example:

1.  $\{x = 1\} x := x + 1 \{x = 2\}$
2.  $\{x > 9\} x := x + 1 \{x > 10\}$
3.  $\{x > 100\} x := x + 1 \{x > 10\}$

Example 1 and 2 give the **weakest** precondition. We normally prefer that (it gives all circumstances under which the program is correct)

In the following we will assume that variables are integer.

# Hoare Logic: Rules

Axioms to find the weakest precondition

- Assignment:  $x := e$
- Consecution:  $S1; S2$
- if-statement:  $\text{if } b \text{ then } S1 \text{ else } S2$
- Loops:  $\text{while } b \text{ do } S \text{ od}$
  
- Plus
  - extra “glue” rules to make things work
  - Function calls, mallocs, pointers, etc

# Axiom of Assignment

$$\frac{}{\{P[x \rightarrow e]\} x := e \{P\}}$$

$P[x \rightarrow e]$  means that  $x$  is replaced by  $e$  in  $P$

Example:

$$\{y = 4\} x := y \{x = 4\}$$

$$\{x+1 = 4\} x := x + 1 \{x = 4\}$$

$$\{x = 4\} x := 2 * x \{x = 8\}$$

$$\{x < 4\} x := 2 * x \{x < 8\}$$

This rule gives the *weakest precondition*, i.e.,  $\{P[x \rightarrow e]\}$  holds before  $S$  **if and only if**  $P$  holds afterwards

# Sequencing Rule (Consecution)

$$\frac{\{P\} S1 \{Q\} \quad \{Q\} S2 \{R\}}{\{P\} S1; S2 \{R\}}$$

Example:

$\{x+1 = 4\} x := x + 1 \{x = 4\}$

$\{x = 4\} x := x * 2 \{x = 8\}$

Conclusion:

$\{x = 3\} x := x + 1; x := x * 2 \{x = 8\}$

*The horizontal line means: if everything above the line is true, then so is everything below the line.*

# Conditional Rule

$$\frac{\{P \wedge c\} S1 \{Q\} \quad \{P \wedge \neg c\} S2 \{Q\}}{\{P\} \text{ if } c \text{ then } S1 \text{ else } S2 \text{ fi } \{Q\}}$$

Example:

$\{x \geq 0\}$  skip  $\{x \geq 0\}$

$\{x < 0\}$   $x = -x$   $\{x \geq 0\}$

$\{\text{true}\}$  if( $x \geq 0$ ) then skip else  $x = -x$  fi  $\{x \geq 0\}$

## Conditional Rule (Alternative)

$$\frac{\{P1\} S1 \{Q\} \quad \{P2\} S2 \{Q\}}{\{c \wedge P1 \vee \neg c \wedge P2\} \text{ if } c \text{ then } S1 \text{ else } S2 \text{ fi } \{Q\}}$$

Example:

$\{x \geq 0\}$  skip  $\{x \geq 0\}$

$\{x < 0\}$   $x = -x$   $\{x \geq 0\}$

$\{x \geq 0 \vee x < 0\}$  if( $x \geq 0$ ) then skip else  $x = -x$  fi  $\{x \geq 0\}$

# While Rule

$$\frac{\{P \wedge c\} S \{P\}}{\{P\} \text{ while } c \text{ do } S \text{ od } \{P \wedge \neg q\}}$$

Example

$\{x < 4\} x = x + 1 \{x \leq 4\}$

$\{x \leq 4\} \{\text{while}(x < 4) \text{ do } x = x + 1 \text{ od } \{x = 4\}$

*This is the hardest rule: how do you find P?*

Notes:

$P = x \leq 4$

$q = x < 4$

$\{x+1 \leq 4\} = \{x < 4\}$

$\{P \wedge q\} = \{x \leq 4 \wedge x < 4\} = \{x < 4\}$

$\{P \wedge \neg q\} = \{x \leq 4 \wedge \neg(x < 4)\} = \{x = 4\}$

# Consequence Rule

## Strengthening the precondition

$$\frac{\{P\} S \{Q\} \quad P' \rightarrow P}{\{P'\} S \{Q\}}$$

## Weakening the postcondition

$$\frac{\{P\} S \{Q\} \quad Q \rightarrow Q'}{\{P\} S \{Q'\}}$$

Example:

$\{x = -5\}$  if( $x \geq 0$ ) then skip else  $x = -x$  fi  $\{x \geq 0\}$

$\{\text{true}\}$  if( $x \geq 0$ ) then skip else  $x = -x$  fi  $\{x \neq -10\}$

# Proof Example I

```
{true}
if (a > b) {
  t = a;
  a = b;
  b = t;
} else {
  skip;
}
{b>a}
```

# Proof Example II

Proving a program using Hoare logic requires creativity. Let's try a simple example.

```
{x ≥ 0 ∧ y > 0}
```

```
r := x; q := 0;
```

```
while (r ≥ y) do
```

```
    r := r - y;
```

```
    q := q + 1;
```

```
od
```

# Proof Example

This proof shows all the important data but it is hard to follow how it was built. See the next slide for a deduction.

```
{x ≥ 0 ∧ y > 0}
r := x; q := 0;
{x = (y·q + r) ∧ 0 ≤ r ∧ y ≥ 0}
while(r ≥ y) do
  {r ≥ y ∧ x = (y·q + r) ∧ 0 ≤ r ∧ y ≥ 0}
  r := r - y;
  q := q + 1;
  {x = (y·q + r) ∧ 0 ≤ r ∧ y ≥ 0}
od
{x = (y·q + r) ∧ 0 ≤ r ∧ r < y ∧ y ≥ 0}
```

# Proof Example: Deductive

We've split the proof into three parts, mainly because of space.

Let

$$L = (\text{while}(r \geq y) \text{do } r := r - y; q := q + 1; \text{od})$$

$$S = (r := x; q := 0; L),$$

and let

$$R1 = \{x \geq 0 \wedge y > 0\} r := x; q := 0 \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\},$$

$$R2 = \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\} L \{x = yq + r \wedge r \geq 0 \wedge y \geq 0 \wedge r < y\}.$$

First, we prove that if  $R1$  and  $R2$  are correct, then so is the program. We use the axiom of consecution and strengthening of the precondition.

$$\frac{\frac{R1 \quad R2}{\{x \geq 0 \wedge y \geq 0\} S \{x = yq + r \wedge 0 \leq r < y\}}}{\{x \geq 0 \wedge y > 0\} S \{x = yq + r \wedge 0 \leq r < y\}}$$

Then we proof  $R1$ . We use the axiom of assignment twice and the axiom of consecution once.

$$\frac{\overline{\{x \geq 0 \wedge y \geq 0\} r := x; \{x = r \wedge r \geq 0 \wedge y \geq 0\}} \quad \overline{\{x = r \wedge r \geq 0 \wedge y \geq 0\} q := 0 \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\}}}{\{x \geq 0 \wedge y \geq 0\} r := x; q := 0 \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\}}$$

Then, we proof  $R2$  using the axiom of assignment (twice), the axiom for consecution and that for loops. For the latter, we have  $P = (x = yq + r \wedge r \geq 0 \wedge y \geq 0)$  and  $p = (r \geq y)$ .

$$\frac{\overline{\{x = yq + r \wedge r \geq 0 \wedge y \geq 0 \wedge r \geq y\} r := r - y \{x = y(q+1) + r \wedge r \geq 0 \wedge y \geq 0\}} \quad \overline{\{x = y(q+1) + r \wedge r \geq 0 \wedge y \geq 0\} q := q + 1 \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\}}}{\frac{\{x = yq + r \wedge r \geq 0 \wedge y \geq 0 \wedge r \geq y\} r := r - y; q := q + 1 \{x = yq + r \wedge r \geq 0 \wedge y \geq 0\}}{\{x = yq + r \wedge r \geq 0 \wedge y \geq 0\} L \{x = yq + r \wedge r \geq 0 \wedge y \geq 0 \wedge r < y\}}}$$

# More Examples

```
x = a;  
y = 0;  
while (x != 0) {  
    x = x - 1;  
    y = y + 2;  
}  
assert (y == 2*a);
```

Input:  
a ... integer

```
s = 0;
i = 0;
while (i != n) {
    s = s + a[i];
    i = i + 1;
}
assert (s ==  $\sum_{j=0}^{n-1} a[j]$ );
```

Input:  
a ... array of  
integers  
n ... length of a

```

r = false;
i = 0;
while(i != n) {
    if(a[i] == x) {
        r = true;
    }
    i = i + 1;
}

```

```

assert(r == ( $\bigvee_{j=0}^{n-1} a[j] == x$ ));

```

Input:

a ... array

n ... length of a

x ... value to look  
for in a

Hint:

$(\bigvee_{j=0}^{-1} \Phi) == \text{false}$