POLYTECH
PARIS-SUD

2021

Cycle Ingénieur – 2ème année
Département Informatique

Verification and Validation

Part IV :

Deductive Verification (I)

Burkhart Wolff
Département Informatique
Université Paris-Saclay / LMF

How to do Verification ?

In the sequel, we concentrate on

Deductive Verification

(Proof Techniques)

Motivation: Hoare - Logic

Motivation: Hoare - Logic

- A means to reason over all input and all states: Is there

Motivation: Hoare - Logic

- A means to reason over all input and all states: Is there

A Logic for Programs ???

Motivation: Hoare - Logic

- A means to reason over all input and all states: Is there

A Logic for Programs ???

- We consider the Hoare-Logic, technically an inference system PL + E + A + Hoare

Recall: Proof Systems

Recall: Proof Systems

- An Inference System (or *Logical Calculus*) allows to infer formulas from a set of *elementary facts* (axioms) and inferred facts by rules:

Recall: Proof Systems

- An Inference System (or *Logical Calculus*) allows to infer formulas from a set of *elementary facts* (axioms) and inferred facts by rules:

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

Recall: Proof Systems

- An Inference System (or *Logical Calculus*) allows to infer formulas from a set of *elementary facts* (axioms) and inferred facts by rules:

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

- "from the *assumptions* A_1 to A_n, you can infer the *conclusion* A_{n+1}."

  A rule with n=0 is an *elementary fact*. Variables occurring in the formulas A_n can be arbitrarily substituted.

Recall: Proof Systems

- An Inference System (or *Logical Calculus*) allows to infer formulas from a set of *elementary facts* (axioms) and inferred facts by rules:

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

- "from the *assumptions* A_1 to A_n, you can infer the *conclusion* A_{n+1}."

  A rule with n=0 is an *elementary fact*. Variables occurring in the formulas A_n can be arbitrarily substituted.

- Assumptions and conclusions are terms in a logic containing variables

Recall: Rule instances

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

$\{x \mapsto 1+2,$
$y \mapsto 2+1,$
$z \mapsto 3\}$

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

$\{x \mapsto 1+2,$
$y \mapsto 2+1,$
$z \mapsto 3\}$

$$\frac{1 + 2 = 2 + 1 \quad 2 + 1 = 3}{1 + 2 = 3}$$

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

$\{x \mapsto 1+2,\ y \mapsto 2+1,\ z \mapsto 3\}$

$$\frac{1 + 2 = 2 + 1 \quad 2 + 1 = 3}{1 + 2 = 3}$$

$\{x \mapsto 1+2,\ y \mapsto a,\ z \mapsto 3\}$

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

{x↦1+2,
y↦2+1,
z↦3}

$$\frac{1+2 = 2+1 \quad 2+1 = 3}{1+2 = 3}$$

{x↦1+2,
y↦a,
z↦3}

$$\frac{1+2 = a \quad a = 3}{1+2 = 3}$$

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

{x↦1+2, y↦2+1, z↦3}

$$\frac{1+2 = 2+1 \quad 2+1 = 3}{1+2 = 3}$$

{x↦1+2, y↦a, z↦3}

$$\frac{1+2 = a \quad a = 3}{1+2 = 3}$$

{x↦τ*5, y↦5*τ}

Recall: Rule instances

The variables in an inference rule can be replaced by a substitution. The substituted inference rule is called an instance (of this rule).

$$\frac{x = y \quad y = z}{x = z}$$

{x↦1+2, y↦2+1, z↦3}

$$\frac{1+2 = 2+1 \quad 2+1 = 3}{1+2 = 3}$$

{x↦1+2, y↦a, z↦3}

$$\frac{1+2 = a \quad a = 3}{1+2 = 3}$$

{x↦τ*5, y↦5*τ}

$$\frac{\tau * 5 = 5 * \tau \quad 5 * \tau = z}{\tau * 5 = z}$$

Recall: Formal Proofs

Recall: Formal Proofs

- A *Formal Proof* (or : *Derivation*) is a tree with rule instances as nodes

Recall: Formal Proofs

- A *Formal Proof* (or : *Derivation*)
  is a tree with rule instances as nodes

$$
\dfrac{\dfrac{\dfrac{f(a,b)=a}{a=f(a,b)} \qquad \dfrac{f(a,b)=a \quad f(f(a,b),b)=c}{f(a,b)=c}}{a=c} \qquad \dfrac{}{g(a)=g(a)}}{g(a)=g(c)}
$$

Recall: Formal Proofs

- A *Formal Proof* (or : *Derivation*) is a tree with rule instances as nodes

$$\dfrac{\dfrac{f(a,b)=a}{a=f(a,b)} \quad \dfrac{f(a,b)=a \quad f(f(a,b),b)=c}{f(a,b)=c}}{a=c} \qquad \dfrac{}{g(a)=g(a)}$$

$$\dfrac{a=c \qquad g(a)=g(a)}{g(a)=g(c)}$$

- The non-elementary facts at the leaves are the *global assumptions* (here $f(a,b) = a$ and $f(f(a,b),b) = c$).

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- Basis: The mini-language „IMP“,
  (following Glenn Wynskell's Book)

Hoare - Logic: A Proof System for Programs

- Basis: The mini-language „IMP",
  (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)

Hoare – Logic: A Proof System for Programs

- Basis: The mini-language „IMP“,
  (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)
  - the empty command SKIP

Hoare - Logic: A Proof System for Programs

- Basis: The mini-language „IMP",
  (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)
  - the empty command SKIP
  - the assignment x:== E $(x \in V)$

Hoare – Logic: A Proof System for Programs

- Basis: The mini-language „IMP“,
  (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)
  - the empty command   SKIP
  - the assignment   x:== E $(x \in V)$
  - the sequential compos. $c_1 \; ; \; c_2$

Hoare - Logic: A Proof System for Programs

- Basis: The mini-language „IMP",
  (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)
  - the empty command — SKIP
  - the assignment — x:== E $(x \in V)$
  - the sequential compos. — c_1 ; c_2
  - the conditional — IF cond THEN c_1 ELSE c_2

Hoare - Logic: A Proof System for Programs

- Basis: The mini-language „IMP“, (following Glenn Wynskell's Book)

- We have the following commands (*cmd*)
  - the empty command SKIP
  - the assignment $x := = E \quad (x \in V)$
  - the sequential compos. $c_1 \; ; \; c_2$
  - the conditional IF cond THEN c_1 ELSE c_2
  - the loop WHILE cond DO c

  where c, c_1, c_2, are cmd's, V variables,

  E an arithmetic expression, and cond a boolean expression.

Hoare – Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P
  - a post-condition Q

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P
  - a post-condition Q
  - and a piece of program cmd

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P
  - a post-condition Q
  - and a piece of program cmd
  - the triple (P,cmd,Q) is written:

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P
  - a post-condition Q
  - and a piece of program cmd
  - the triple (P,cmd,Q) is written:

$$\vdash \{P\}\ cmd\ \{Q\}$$

Hoare - Logic: A Proof System for Programs

- Core Concept: A Hoare Triple consisting ...
  - of a pre-condition P
  - a post-condition Q
  - and a piece of program cmd
  - the triple (P,cmd,Q) is written:

$$\vdash \{P\}\ cmd\ \{Q\}$$

  - *P* and *Q* are formulas over the variables *V*, so they can be seen as set of possible states.

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Idea: We consider the specification (precond, postcond) and the program together

Hoare – Logic: A Proof System for Programs

- Idea: We consider the specification (precond, postcond) and the program together
- The Hoare-Triple says : The program "is conform" to the specification

Hoare – Logic: A Proof System for Programs

- Idea: We consider the specification (precond, postcond) and the program together
- The Hoare-Triple says : The program "is conform" to the specification
- More precisely:

Hoare – Logic: A Proof System for Programs

- Idea: We consider the specification (precond, postcond) and the program together
- The Hoare-Triple says : The program "is conform" to the specification
- More precisely:

$$\vdash \{P\}\ cmd\ \{Q\}$$

Hoare – Logic: A Proof System for Programs

- Idea: We consider the specification (precond, postcond) and the program together
- The Hoare-Triple says : The program "is conform" to the specification
- More precisely:

$$\vdash \{P\}\ cmd\ \{Q\}$$

If a program *cmd* starts in a state admitted by *P* if it terminates, that the program must reach a state that satisfies *P*.

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}} \qquad \frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \quad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \mathrm{IF}\ cond\ \mathrm{THEN}\ c\ \mathrm{ELSE}\ d\{Q\}}$$

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} := = \mathrm{E}\{P\}}$$

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \quad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \mathrm{IF}\ cond\ \mathrm{THEN}\ c\ \mathrm{ELSE}\ d\{Q\}}$$

$$\frac{\vdash \{P\}\ c\ \{Q\} \quad \vdash \{Q\}\ d\ \{R\}}{\vdash \{P\}\ c;\ d\ \{R\}}$$

Hoare – Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}} \qquad \frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x :== E}\{P\}}$$

$$\frac{\vdash \{P \wedge \mathit{cond}\}\ c\ \{Q\} \quad \vdash \{P \wedge \neg \mathit{cond}\}\ d\ \{Q\}}{\vdash \{P\}\ \mathrm{IF}\ \mathit{cond}\ \mathrm{THEN}\ c\ \mathrm{ELSE}\ d\{Q\}}$$

$$\frac{\vdash \{P\}\ c\ \{Q\} \quad \vdash \{Q\}\ d\ \{R\}}{\vdash \{P\}\ c;\ d\ \{R\}} \qquad \frac{\vdash \{P \wedge \mathit{cond}\}\ c\ \{P\}}{\vdash \{P\}\ \mathrm{WHILE}\ \mathit{cond}\ \mathrm{DO}\ c\ \{P \wedge \neg \mathit{cond}\}}$$

Hoare - Logic: A Proof System for Programs

- PL + E + A + Hoare (simplified binding) at a glance:

$$\frac{}{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}} \qquad \frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

$$\frac{\vdash \{P \land cond\}\ c\ \{Q\} \quad \vdash \{P \land \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \mathrm{IF}\ cond\ \mathrm{THEN}\ c\ \mathrm{ELSE}\ d\{Q\}}$$

$$\frac{\vdash \{P\}\ c\ \{Q\} \quad \vdash \{Q\}\ d\ \{R\}}{\vdash \{P\}\ c;\ d\ \{R\}} \qquad \frac{\vdash \{P \land cond\}\ c\ \{P\}}{\vdash \{P\}\ \mathrm{WHILE}\ cond\ \mathrm{DO}\ c\ \{P \land \neg cond\}}$$

$$\frac{P \rightarrow P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \rightarrow Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Hoare – Logic: A Proof System for Programs

Let's consider it one by one …

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- The SKIP-rule for the empty statement:

Hoare - Logic: A Proof System for Programs

- The SKIP-rule for the empty statement:

$$\overline{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

Hoare – Logic: A Proof System for Programs

- The SKIP-rule for the empty statement:

$$\overline{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

well, states do not change ...

Hoare – Logic: A Proof System for Programs

- The SKIP-rule for the empty statement:

$$\overline{\vdash \{P\}\ \mathrm{SKIP}\ \{P\}}$$

well, states do not change ...

Therefore, valid states remain valid.

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- The assignment rule:

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} := \mathrm{E} \{P\}}$$

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\} \text{ x :== E}\{P\}}$$

- Example (1):

$\vdash \{1 \leq x \wedge x \leq 10\}$ x:== x+2 $\{3 \leq x \wedge x \leq 12\}$

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (1):

$$\vdash \{1 \leq x \wedge x \leq 10\}\ x :== x+2\ \{3 \leq x \wedge x \leq 12\}$$

- Is this really an *instance* of the assignment rule ? We calculate:

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (1):

$$\vdash \{1 \leq x \wedge x \leq 10\}\ x :== x+2\ \{3 \leq x \wedge x \leq 12\}$$

- Is this really an *instance* of the assignment rule ? We calculate:

$$(3 \leq x \wedge x \leq 12)\ [x \mapsto x+2]$$

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\overline{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (1):

$$\vdash \{1 \leq x \wedge x \leq 10\}\ x :== x+2\ \{3 \leq x \wedge x \leq 12\}$$

- Is this really an *instance* of the assignment rule ? We calculate:

$(3 \leq x \wedge x \leq 12)\ [x \mapsto x+2]$

$\equiv 3 \leq (x+2) \wedge (x+2) \leq 12$

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (1):

$$\vdash \{1 \leq x \wedge x \leq 10\}\ x :== x+2\ \{3 \leq x \wedge x \leq 12\}$$

- Is this really an *instance* of the assignment rule ? We calculate:

$(3 \leq x \wedge x \leq 12)\ [x \mapsto x+2]$

$\equiv 3 \leq (x+2) \wedge (x+2) \leq 12$

$\equiv 1 \leq x \wedge x \leq 10$

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

Hoare - Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (2):

$\vdash$ {true} x:== 2 {x=2}

Hoare - Logic: A Proof System for Programs

- The *assignment* rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (2):

$\vdash$ {true} x:== 2 {x=2}

- Is this really an *instance* of the assignment rule ? We calculate:

Hoare – Logic: A Proof System for Programs

- The assignment rule:

$$\frac{}{\vdash \{P[x \mapsto E]\}\ \mathrm{x} :== \mathrm{E}\{P\}}$$

- Example (2):

$\vdash$ {true} x:== 2 {x=2}

- Is this really an *instance* of the assignment rule ? We calculate:

(x=2) [x↦2]

Hoare - Logic: A Proof System for Programs

- The *assignment* rule:

$$\frac{}{\vdash \{P[x \mapsto E]\} \text{ x :== E}\{P\}}$$

- Example (2):

$\vdash$ {true} x:== 2 {x=2}

- Is this really an *instance* of the assignment rule ? We calculate:

(x=2) [x↦2]

≡ 2=2 ≡ true (reflexivity…)

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

The conditional-rule:

Hoare – Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \land cond\}\ c\ \{Q\} \quad \vdash \{P \land \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Hoare - Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \qquad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \mathrm{IF}\ cond\ \mathrm{THEN}\ c\ \mathrm{ELSE}\ d\{Q\}}$$

Example (3):

Hoare - Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \qquad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Example (3):

$$\vdash \{true\}\ \text{IF}\ 0 \leq x\ \text{THEN SKIP ELSE}\ x :== -x\ \{0 \leq x\}$$

Hoare – Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \land cond\}\ c\ \{Q\} \quad \vdash \{P \land \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Example (3):

Hoare - Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \land cond\}\ c\ \{Q\} \quad \vdash \{P \land \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Example (3):

This can be extended to the formal proof:

Hoare – Logic: A Proof System for Programs

The conditional-rule:

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \quad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Example (3):

Hoare - Logic: A Proof System for Programs

- The conditional-rule:

$$\frac{\vdash \{P \wedge cond\}\ c\ \{Q\} \qquad \vdash \{P \wedge \neg cond\}\ d\ \{Q\}}{\vdash \{P\}\ \text{IF}\ cond\ \text{THEN}\ c\ \text{ELSE}\ d\{Q\}}$$

Example (3):

$$\frac{\dfrac{}{\vdash \{true \wedge 0 \leq x\}\ \text{SKIP}\ \{0 \leq x\}} \qquad \dfrac{\cdots}{\vdash \{true \wedge \neg(0 \leq x)\}\ x :== -x\ \{0 \leq x\}}}{\vdash \{true\}\ \text{IF}\ 0 \leq x\ \text{THEN SKIP ELSE}\ x :== -x\ \{0 \leq x\}}$$

Hoare – Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- The *sequence* rule:

Hoare - Logic: A Proof System for Programs

- The sequence rule:

$$\frac{\vdash \{P\}\ c\ \{Q\} \quad \vdash \{Q\}\ d\ \{R\}}{\vdash \{P\}\ c;\ d\ \{R\}}$$

Hoare - Logic: A Proof System for Programs

- The sequence rule:

$$\frac{\vdash \{P\}\ c\ \{Q\} \qquad \vdash \{Q\}\ d\ \{R\}}{\vdash \{P\}\ c;\ d\ \{R\}}$$

- essentially a relational composition on state sets.

Hoare - Logic: A Proof System for Programs

0

Hoare – Logic: A Proof System for Programs

The rule for the sequence.

Example (4):

0

Hoare - Logic: A Proof System for Programs

The rule for the sequence.

Example (4):

$$\frac{}{\vdash \{true\}\ tm :== 1; (sum :== 1; i :== 0)\ \{tm = 1 \land sum = 1 \land i = 0\}}$$

Hoare - Logic: A Proof System for Programs

The rule for the sequence.

Example (4):

$$\frac{}{\vdash \{true\}\ tm :== 1; (sum :== 1; i :== 0)\ \{tm = 1 \land sum = 1 \land i = 0\}}$$

This can be extended to the formal proof:

Hoare - Logic: A Proof System for Programs

The rule for the sequence.

Example (4):

$$\dfrac{\dfrac{}{\vdash \{true\} tm :== 1 \{tm = 1\}} \qquad \dfrac{\dfrac{}{\vdash \{tm = 1\} sum :== 1 \{B\}} \qquad \dfrac{}{\vdash \{B\}\ i :== 0\ \{A\}}}{\vdash \{tm = 1\}\ sum :== 1; i :== 0\ \{A\}}}{\vdash \{true\}\ tm :== 1; (sum :== 1; i :== 0)\ \{tm = 1 \land sum = 1 \land i = 0\}}$$

where $A = tm = 1 \land sum = 1 \land i = 0$ and where $B = tm = 1 \land sum = 1$.

Hoare - Logic: A Proof System for Programs

The rule for the sequence.

Example (4):

$$\frac{\dfrac{}{\vdash \{true\} tm :== 1 \{tm = 1\}} \qquad \dfrac{\dfrac{}{\vdash \{tm = 1\} sum :== 1 \{B\}} \qquad \dfrac{}{\vdash \{B\}\ i :== 0\ \{A\}}}{\vdash \{tm = 1\}\ sum :== 1; i :== 0\ \{A\}}}{\vdash \{true\}\ tm :== 1; (sum :== 1; i :== 0)\ \{tm = 1 \land sum = 1 \land i = 0\}}$$

where $A = tm = 1 \land sum = 1 \land i = 0$ and where $B = tm = 1 \land sum = 1$.

It is often practical to introduce abbreviations.

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- The while-rule.

Hoare - Logic: A Proof System for Programs

- The while-rule.

$$\frac{\vdash \{P \wedge cond\}\ c\ \{P\}}{\vdash \{P\}\ \text{WHILE}\ cond\ \text{DO}\ c\ \{P \wedge \neg cond\}}$$

Hoare - Logic: A Proof System for Programs

- The while-rule.

$$\frac{\vdash \{P \land cond\}\ c\ \{P\}}{\vdash \{P\}\ \mathrm{WHILE}\ cond\ \mathrm{DO}\ c\ \{P \land \neg cond\}}$$

- This works like an induction: if some P is true after n traversals of the loop and remain true for the n+1 traversal, it must be always true.

Hoare - Logic: A Proof System for Programs

- The while-rule.

$$\frac{\vdash \{P \land cond\}\ c\ \{P\}}{\vdash \{P\}\ \text{WHILE}\ cond\ \text{DO}\ c\ \{P \land \neg cond\}}$$

- This works like an induction: if some P is true after n traversals of the loop and remain true for the n+1 traversal, it must be always true.
- When exiting the loop, the condition cond can on longer hold.

Hoare - Logic: A Proof System for Programs

- The while-rule.

$$\frac{\vdash \{P \wedge cond\}\ c\ \{P\}}{\vdash \{P\}\ \text{WHILE}\ cond\ \text{DO}\ c\ \{P \wedge \neg cond\}}$$

- This works like an induction: if some P is true after n traversals of the loop and remain true for the n+1 traversal, it must be always true.
- When exiting the loop, the condition cond can on longer hold.
- The predicate P is called an invariant. Note that an invariant can be maintained even if the concrete state changes ! See:

Hoare - Logic: A Proof System for Programs

- The while-rule.

$$\frac{\vdash \{P \wedge cond\}\ c\ \{P\}}{\vdash \{P\}\ \text{WHILE}\ cond\ \text{DO}\ c\ \{P \wedge \neg cond\}}$$

- This works like an induction: if some P is true after n traversals of the loop and remain true for the n+1 traversal, it must be always true.
- When exiting the loop, the condition cond can on longer hold.
- The predicate P is called an invariant. Note that an invariant can be maintained even if the concrete state changes ! See:

$\vdash$ {1≤x ∧ x≤10} WHILE x < 10 DO x:== x+1 {¬ (x < 10) ∧ 1≤x ∧ x≤10}

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

$$\frac{P \rightarrow P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \rightarrow Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

$$\frac{P \to P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \to Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Reflects the intuition that P' is a subset of legal states P and Q is a subset of legal states Q'.

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

$$\frac{P \rightarrow P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \rightarrow Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Reflects the intuition that P' is a subset of legal states P and Q is a subset of legal states Q'.

This is the only rule that is not determined by the syntax of the program; it can be applied anywhere in the (Hoare-) proof.

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

$$\frac{P \rightarrow P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \rightarrow Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Example (5) (the continuation of Example (3)):

Hoare - Logic: A Proof System for Programs

- The consequence-rule:

$$\frac{P \to P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' \to Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Example (5) (the continuation of Example (3)):

$$\frac{true \land \neg(0 \leq x) \to (0 \leq -x) \quad \dfrac{}{\vdash \{(0 \leq x)[x \mapsto -x]\}\ x :== -x\ \{0 \leq x\}} \quad 0 \leq x \to 0 \leq x}{\vdash \{true \land \neg(0 \leq x)\}\ x :== -x\ \{0 \leq x\}}$$

Hoare – Logic: A Proof System for Programs

- The Hoare calculus has a number of implicit consequences, i.e. rules that can be derived from the other ones.

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- A handy derived rule, the False-rule:

Hoare - Logic: A Proof System for Programs

- A handy derived rule, the False-rule:

$$\frac{}{\vdash \{false\}\ cmd\ \{false\}}$$

Hoare - Logic: A Proof System for Programs

- A handy derived rule, the False-rule:

$$\frac{}{\vdash \{false\}\ cmd\ \{false\}}$$

- **Proof**: by induction over cmd ! (At the Blackboard)

Hoare - Logic: A Proof System for Programs

- A handy derived rule, the False-rule:

$$\frac{}{\vdash \{false\}\ cmd\ \{false\}}$$

- **Proof**: by induction over cmd ! (At the Blackboard)

- A very handy corollary of the False-rule and the consequence-rule is the FalseE-rule:

Hoare - Logic: A Proof System for Programs

- A handy derived rule, the False-rule:

$$\frac{}{\vdash \{false\}\ cmd\ \{false\}}$$

- **Proof**: by induction over cmd ! (At the Blackboard)

- A very handy corollary of the False-rule and the consequence-rule is the FalseE-rule:

$$\frac{}{\vdash \{false\}\ cmd\ \{P\}}$$

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Another handy corollary of the False-rule:

Hoare - Logic: A Proof System for Programs

- Another handy corollary of the False-rule:

$$\frac{}{\vdash \{P \land \neg cond\}\ \mathrm{WHILE}\ cond\ \mathrm{DO}\ c\ \{P \land \neg cond\}}$$

Hoare – Logic: A Proof System for Programs

- Another handy corollary of the False-rule:

$$\frac{}{\vdash \{P \wedge \neg cond\}\ \text{WHILE}\ cond\ \text{DO}\ c\ \{P \wedge \neg cond\}}$$

Proof:

by consequence-rule, while-rule,

P and cond-negation,

False-rule.

This means: If we can not enter into the WHILE-loop, it behaves like a SKIP.

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Yet another handy corollary of the consequence rule:

Hoare – Logic: A Proof System for Programs

- Yet another handy corollary of the consequence rule:

$$\frac{P = P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' = Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Hoare - Logic: A Proof System for Programs

- Yet another handy corollary of the consequence rule:

$$\frac{P = P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' = Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Proof:

by consequence rule and the fact that $P = P'$ (ou $P \equiv P'$) infers $P \rightarrow P'$

Hoare - Logic: A Proof System for Programs

- Yet another handy corollary of the consequence rule:

$$\frac{P = P' \quad \vdash \{P'\}\ cmd\ \{Q'\} \quad Q' = Q}{\vdash \{P\}\ cmd\ \{Q\}}$$

Proof:
by consequence rule and the fact that $P = P'$ (ou $P \equiv P'$) infers $P \rightarrow P'$

- *Note: We will allow to apply this rule implicitly, thus leveraging local "logical massage" of pre- and post-conditions.*

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Hoare – Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- Example (6):

Hoare – Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

$$\frac{true \rightarrow true \qquad \vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\} \qquad false \rightarrow x = 42}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

true ∧ ¬*true* ≡ *false*

$$\frac{true \rightarrow true \qquad \vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\} \qquad false \rightarrow x = 42}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

$true \wedge \neg true \equiv false$

$$\frac{true \rightarrow true^{\surd} \quad \vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\} \quad false \rightarrow x = 42}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

$true \wedge \neg true \equiv false$

$$\frac{true \rightarrow true^{\surd} \quad \vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\} \quad false \rightarrow x = 42^{\surd}}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

$true \wedge \neg true \equiv false$

$$\frac{true \rightarrow true^{\surd} \qquad \dfrac{\dfrac{}{\vdash \{true \wedge false\}\text{SKIP}\{false\}}}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\}} \qquad false \rightarrow x = 42^{\surd}}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO}\ SKIP\ \{x = 42\}}$$

Proof (bottom up):

$$\frac{true \rightarrow true \qquad \dfrac{\dfrac{}{\vdash \{true \wedge false\}\text{SKIP}\{false\}}}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{false\}} \qquad false \rightarrow x = 42}{\vdash \{true\}\ \text{WHILE}\ true\ \text{DO SKIP}\ \{x = 42\}}$$

Hoare - Logic: A Proof System for Programs

- Example (6):

$$\frac{}{\vdash \{true\} \text{ WHILE } true \text{ DO } SKIP \; \{x = 42\}}$$

<u>Note</u>:

Hoare-Logic is a calculus for

partial correctness; for non-terminating

programs, it is possible to prove *anything*!

Hoare - Logic: A Proof System for Programs

- Example (7):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare – Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\frac{true \rightarrow I \qquad \dfrac{\dfrac{I \wedge x < 2 \rightarrow I'' \qquad \dfrac{}{\vdash \{I''\}\ x :== x+1\ \{I'\}} \qquad I' \rightarrow I}{\vdash \{I \wedge x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \wedge \neg(x<2)\}} \qquad I \wedge \neg(x<2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\dfrac{I \land x < 2 \to I'' \qquad \dfrac{}{\vdash \{I''\}\ x :== x+1\ \{I'\}} \qquad I' \to I}{\vdash \{I \land x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \land \neg(x < 2)\}} \qquad I \land \neg(x<2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \leq x\}}$$

We can't apply the WHILE-rule directly — the only other choice is the consequence rule. Instantiating the invariant variable P by a fresh variable I allows us to bring the triple into a shape that we can apply the WHILE rule later

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\frac{true \rightarrow I \qquad \overline{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

We can't apply the WHILE-rule directly — the only other choice is the consequence rule. Instantiating the invariant variable P by a fresh variable I allows us to bring the triple into a shape that we can apply the WHILE rule later

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \rightarrow I \qquad \overline{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\frac{true \rightarrow I \qquad \overline{\vdash \{I\}\ \text{WHILE } x < 2 \text{ DO } x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE } x < 2 \text{ DO } x :== x + 1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\frac{true \rightarrow I \qquad \dfrac{}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \land \neg(x < 2)\}} \qquad I \land \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Now we can apply the while rule.

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\frac{true \to I \qquad \overline{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \land \neg(x < 2)\}} \qquad I \land \neg(x < 2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \rightarrow I \qquad \dfrac{\dfrac{}{\vdash \{I \wedge x < 2\}\ x :== x + 1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare – Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \rightarrow I \qquad \dfrac{\overline{\vdash \{I \wedge x < 2\}\ x :== x + 1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\overline{\vdash \{I \land x < 2\}\ x :== x + 1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \land \neg(x < 2)\}} \qquad I \land \neg(x < 2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

To be sure (entering the while loop) we apply again the consequence rule. For the missing bit, we instantiate I".

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\dfrac{I \wedge x < 2 \to I'' \qquad \vdash \{I''\}\; x :== x+1\; \{I'\} \qquad I' \to I}{\vdash \{I \wedge x < 2\}\; x :== x+1\; \{I\}}}{\vdash \{I\}\ \mathrm{WHILE}\ x < 2\ \mathrm{DO}\ x :== x+1\; \{I \wedge \neg(x<2)\}} \qquad I \wedge \neg(x<2) \to 2 \leq x}{\vdash \{true\}\ \mathrm{WHILE}\ x < 2\ \mathrm{DO}\ x :== x+1\; \{2 \leq x\}}$$

To be sure (entering the while loop) we apply again the consequence rule. For the missing bit, we instantiate I''.

Hoare – Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\dfrac{I \wedge x < 2 \to I'' \qquad \vdash \{I''\}\ x :== x + 1\ \{I'\} \qquad I' \to I}{\vdash \{I \wedge x < 2\}\ x :== x + 1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \rightarrow I \qquad \dfrac{\dfrac{I \wedge x < 2 \rightarrow I'' \qquad \vdash \{I''\}\ x :== x+1\ \{I'\} \qquad I' \rightarrow I}{\vdash \{I \wedge x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \mathrm{WHILE}\ x < 2\ \mathrm{DO}\ x :== x+1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \mathrm{WHILE}\ x < 2\ \mathrm{DO}\ x :== x+1\ \{2 \leq x\}}$$

Now, in order to make the assignment rule "fit", we must have $I'' \equiv I'[x \mapsto x+1]$.

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \rightarrow I \qquad \dfrac{\dfrac{I \wedge x < 2 \rightarrow I'' \qquad \overline{\vdash \{I''\}\; x :== x+1\; \{I'\}} \qquad I' \rightarrow I}{\vdash \{I \wedge x < 2\}\; x :== x+1\; \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \wedge \neg(x<2)\}} \qquad I \wedge \neg(x<2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \leq x\}}$$

Now, in order to make the assignment rule "fit", we must have
$I'' \equiv I'[x \mapsto x+1]$.

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\dfrac{I \wedge x < 2 \to I'' \qquad \dfrac{}{\vdash \{I''\}\ x :== x+1\ \{I'\}} \qquad I' \to I}{\vdash \{I \wedge x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \leq x\}}$$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\cfrac{true \rightarrow I \qquad \cfrac{\cfrac{I \wedge x < 2 \rightarrow I'' \qquad \overline{\vdash \{I''\}\ x :== x + 1\ \{I'\}} \qquad I' \rightarrow I}{\vdash \{I \wedge x < 2\}\ x :== x + 1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x < 2) \rightarrow 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

Additionally, in order that this constitutes a Hoare-Proof, we must have all the implications.

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof (bottom up):

$$\dfrac{true \to I \qquad \dfrac{\dfrac{I \wedge x < 2 \to I'' \qquad \dfrac{}{\vdash \{I''\}\ x :== x+1\ \{I'\}} \qquad I' \to I}{\vdash \{I \wedge x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \wedge \neg(x < 2)\}} \qquad I \wedge \neg(x<2) \to 2 \leq x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \leq x\}}$$

Additionally, in order that this constitutes a Hoare-Proof, we must have all the implications.

Hoare - Logic: A Proof System for Programs

- Example (7):

$$\frac{}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x + 1\ \{2 \leq x\}}$$

So, we have a Hoare Proof iff we have a solution to the following list of constraints:

$I'' \equiv I'[x \mapsto x+1]$

$A \equiv true \rightarrow I$

$B \equiv I \wedge \neg(x < 2) \rightarrow 2 \leq x$

$C \equiv I \wedge x < 2 \rightarrow I'[x \mapsto x+1]$

Hoare - Logic: A Proof System for Programs

- Example (7):

  Proof:

$$I'' \equiv I'[x \mapsto x+1]$$

$$A \equiv true \rightarrow I$$

$$B \equiv I \wedge \neg(x < 2) \rightarrow 2 \leq x$$

$$C \equiv I \wedge x < 2 \rightarrow I'[x \mapsto x+1]$$

$$D = I' \rightarrow I$$

  - I must be *true*, this solves A, B, D
  - we are fairly free for a solution for I'; e.g. $x \leq 2$ or $x \leq 5$ would do the trick !

Hoare - Logic: Some facts.

Assume that we have a reasonably well-defined "compiler function" that maps a program to a relation from input to output states:

$$C : \text{cmd} \rightarrow (\sigma\times\sigma)\text{Set}$$

(See Winskell's Book)

Then we can define the "validity" of a specification:

$$\models \{P\}\ cmd\ \{Q\} \equiv \forall \sigma, \sigma'.(\sigma, \sigma') \in C(cmd) \rightarrow P(\sigma) \rightarrow Q(\sigma')$$

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications

    (worst case: ask the user ...)

Hoare – Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications

    (worst case: ask the user ...)

  - ➢ Essence of all: again, we reduced a program verification problem to a constraint resolution problem of formulas …

Hoare - Logic: A Proof System for Programs

- Remarks:

  This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications

    (worst case: ask the user ...)

  - Essence of all: again, we reduced a program verification problem to a constraint resolution problem of formulas …
  - … provided we have solutions for the invariants.

Hoare - Logic: Some facts.

Theorem: Correctness of the Hoare-Calculus:

$$\vdash \{P\}\ cmd\ \{Q\} \rightarrow\ \models \{P\}\ cmd\ \{Q\}$$

... so, whenever there is a proof, it is also valid wrt. C.

Hoare - Logic: Some facts.

Theorem: Relative Completeness of the Hoare-Calculus

$$\models \{P\}\ cmd\ \{Q\} \ \rightarrow\ \vdash \{P\}\ cmd\ \{Q\}$$

Amazingly, this holds also the other way round:
whenever a specification is valid, (and we can solve all the implications on arithmetics), there is a Hoare-Proof.

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:

Hoare – Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary

Hoare – Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)

Hoare – Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule

Hoare - Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications

    (worst case: ask the user ...)

Hoare - Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications

    (worst case: ask the user ...)
  - ➢ Essence of all: again, we reduced a program verification problem to a constraint resolution problem of formulas ...

Hoare - Logic: A Proof System for Programs

- This proof rises the idea of particular construction method of Hoare-Proofs, which can be automated:
  - apply bottom-up all rules following the cmd-syntax; introduce fresh variables for the wholes where necessary
  - apply the consequence rule only at entry points of loops (this is deterministic!)
  - extract the implications used in these consequence rule
  - try to find solutions for these implications
    (worst case: ask the user ...)
  - ➢ Essence of all: again, we reduced a program verification problem to a constraint resolution problem of formulas ...
  - ➢ ... provided we have solutions for the invariants.

Hoare – Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

$\}\equiv$ prelude

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

$\}\equiv$ prelude

$\}\equiv$ body

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

$\}$ ≡ prelude

$\}$ ≡ body

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

} ≡ prelude

} ≡ body

- Program and Specification in a Hoare Triple

Hoare - Logic: A Proof System for Programs

- Another Example (8) : The integer square-root

```
int i = 0;
int tm = 1;
int sum = 1;
WHILE sum ≤ a DO
  i  := i+1;
  tm := tm + 2;
  sum:= sum + tm;
```

} ≡ prelude

} ≡ body

- Program and Specification in a Hoare Triple

$$\vdash \{a \geq 0\}\ \text{prelude; WHILE sum} \leq a\ \text{DO body}\ \{\text{post}\}$$

where $\text{post} \equiv i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first:

Hoare – Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first: $\vdash \{a \geq 0\}$ prelude $\{a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1\}$

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first: $\vdash \{a \geq 0\}$ prelude $\{a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1\}$
  - and:

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first: $\vdash \{a \geq 0\}$ prelude $\{a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1\}$
  - and:

$$\vdash \{a \geq 0 \wedge A\} \text{ WHILE } sum \leq a \text{ DO } body \ \{i^2 \leq a \wedge a < (i+1)^2\}$$

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first: $\vdash \{a \geq 0\}$ prelude $\{a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1\}$

We know that already ....

  - and:

$$\vdash \{a \geq 0 \wedge A\} \text{ WHILE } sum \leq a \text{ DO } body \ \{i^2 \leq a \wedge a < (i+1)^2\}$$

Hoare - Logic: A Proof System for Programs

- We cut it into 2 parts (sequence rule):
  - first: $\vdash \{a \geq 0\}$ prelude $\{a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1\}$

$$\frac{\dfrac{}{\vdash \{true\} tm :== 1 \{tm = 1\}} \qquad \dfrac{\dfrac{}{\vdash \{tm = 1\} sum :== 1 \{B\}} \quad \dfrac{}{\vdash \{B\}\ i :== 0\ \{A\}}}{\vdash \{tm = 1\}\ sum :== 1; i :== 0\ \{A\}}}{\vdash \{true\}\ tm :== 1; (sum :== 1; i :== 0)\ \{tm = 1 \wedge sum = 1 \wedge i = 0\}}$$

We know that already ....

where $A = tm = 1 \wedge sum = 1 \wedge i = 0$ and where $B = tm = 1 \wedge sum = 1$.

  - and:

$$\vdash \{a \geq 0 \wedge A\} \text{ WHILE } sum \leq a \text{ DO } body\ \{i^2 \leq a \wedge a < (i+1)^2\}$$

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\vdash \{a \geq 0 \wedge A\} \text{ WHILE sum} \leq \text{a DO body } \{post\}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\frac{a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\} \text{ WHILE sum} \leq a \text{ DO body } \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\} \text{ WHILE sum} \leq a \text{ DO body } \{post\}}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$
\dfrac{a \geq 0 \wedge A \longrightarrow I \qquad \dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ \mathrm{WHILE}\ sum \leq a\ \mathrm{DO}\ body\ \{a < sum \wedge I\}} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ \mathrm{WHILE}\ sum \leq a\ \mathrm{DO}\ body\ \{post\}}
$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\dfrac{\dfrac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}}{a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post}$$

$$\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\frac{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum \mapsto sum+tm]\} \qquad \vdash \{I[sum \mapsto sum+tm]\}\ sum:=sum+tm\ \{I\}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\}}$$

$$\frac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\frac{a \geq 0 \wedge A \longrightarrow I \qquad \dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\}} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\dfrac{\vdash \{I'\}\ i := i{+}1;\ tm := tm{+}2\ \{I[sum \mapsto sum{+}tm]\} \qquad \dfrac{}{\vdash \{I[sum \mapsto sum{+}tm]\}\ sum{:=}sum{+}tm\ \{I\}}}{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i{+}1;\ tm := tm{+}2;\ sum{:=}sum{+}tm\ \{I\} \qquad I \longrightarrow I}$$

$$\dfrac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i{+}1;\ tm := tm{+}2;\ sum{:=}sum{+}tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\dfrac{a \geq 0 \wedge A \longrightarrow I \qquad \dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ \text{WHILE}\ sum \leq a\ \text{DO}\ body\ \{a < sum \wedge I\}} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ \text{WHILE}\ sum \leq a\ \text{DO}\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\frac{\vdash \{I'\}\ i := i+1\ \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\} \qquad \vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\ tm := tm+2\ \{I[sum \mapsto sum+tm]\}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum \mapsto sum+tm]\}}$$

$$\frac{}{\vdash \{I[sum \mapsto sum+tm]\}\ sum:=sum+tm\ \{I\}}$$

$$\frac{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum \mapsto sum+tm]\} \qquad \vdash \{I[sum \mapsto sum+tm]\}\ sum:=sum+tm\ \{I\}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\}}$$

$$\frac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\frac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\}}$$

$$\frac{a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\vdash \{I'\}\ i := i+1\ \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\} \qquad \overline{\vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\ tm := tm+2\ \{I[sum \mapsto sum+tm]\}}$$

$$\frac{}{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum \mapsto sum+tm]\} \qquad \overline{\vdash \{I[sum \mapsto sum+tm]\}\ sum:=sum+tm\ \{I\}}}$$

$$\frac{}{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\} \qquad I \longrightarrow I}$$

$$\frac{}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\frac{}{a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post}$$

$$\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\vdash \{I'\}\ i := i+1\ \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\} \qquad \vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\ tm := tm+2\ \{I[sum \mapsto sum+tm]\}$$

$$\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum \mapsto sum+tm]\} \qquad \vdash \{I[sum \mapsto sum+tm]\}\ sum := sum+tm\ \{I\}$$

$$I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum := sum+tm\ \{I\} \qquad I \longrightarrow I$$

$$\vdash \{I \wedge sum \leq a\}\ body\ \{I\}$$

$$a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post$$

$$\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\dfrac{\dfrac{\dfrac{I' \longrightarrow I''[i \mapsto i+1] \qquad \vdash \{I''[i \mapsto i+1]\}\, I := I+1\, \{I''\} \qquad I'' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2]}{\vdash \{I'\}\ i := i+1\, \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}} \qquad \dfrac{}{\vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\, tm := tm+2\, \{I[sum \mapsto sum+tm]\}}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2\, \{I[sum \mapsto sum+tm]\}} \qquad \dfrac{}{\vdash \{I[sum \mapsto sum+tm]\}\, sum := sum+tm\ \{I\}}}{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum := sum+tm\ \{I\} \qquad I \longrightarrow I}$$

$$\dfrac{\dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post}}{\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\dfrac{I' \longrightarrow I''[i\mapsto i+1] \qquad \overline{\vdash \{I''[i\mapsto i+1]\}\ I := I+1\ \{I''\}} \qquad I'' \longrightarrow I[sum\mapsto sum+tm][tm\mapsto tm+2]}{\vdash \{I'\}\ i := i+1\ \{I[sum\mapsto sum+tm][tm\mapsto tm+2]\}}$$

$$\dfrac{\vdash \{I'\}\ i := i+1\ \{I[sum\mapsto sum+tm][tm\mapsto tm+2]\} \qquad \overline{\vdash \{I[sum\mapsto sum+tm][tm\mapsto tm+2]\}\ tm := tm+2\ \{I[sum\mapsto sum+tm]\}}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum\mapsto sum+tm]\}}$$

$$\dfrac{\vdash \{I'\}\ i := i+1;\ tm := tm+2\ \{I[sum\mapsto sum+tm]\} \qquad \overline{\vdash \{I[sum\mapsto sum+tm]\}\ sum:=sum+tm\ \{I\}}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\}}$$

$$\dfrac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum:=sum+tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\dfrac{a \geq 0 \wedge A \longrightarrow I \qquad \dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\}} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$I' \longrightarrow I''[i \mapsto i+1] \qquad \vdash \{I''[i \mapsto i+1]\}\, I := I+1\, \{I''\} \qquad I'' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2]$$

$$\vdash \{I'\}\ i := i+1\, \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\} \qquad \vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\, tm := tm+2\, \{I[sum \mapsto sum+tm]\}$$

$$\vdash \{I'\}\ i := i+1;\ tm := tm+2\, \{I[sum \mapsto sum+tm]\} \qquad \vdash \{I[sum \mapsto sum+tm]\}\, sum := sum+tm\ \{I\}$$

$$I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum := sum+tm\ \{I\} \qquad I \longrightarrow I$$

$$\vdash \{I \wedge sum \leq a\}\ body\ \{I\}$$

$$a \geq 0 \wedge A \longrightarrow I \qquad \vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\} \qquad a < sum \wedge I \longrightarrow post$$

$$\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}$$

Hoare - Logic: A Proof System for Programs

- so, for the body, we derive bottom-up:

$$\dfrac{I' \longrightarrow I''[i \mapsto i+1] \qquad \dfrac{}{\vdash \{I''[i \mapsto i+1]\}\, I := I+1\, \{I''\}} \qquad I'' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2]}{\vdash \{I'\}\ i := i+1\, \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}}$$

$$\dfrac{\vdash \{I'\}\ i := i+1\, \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\} \qquad \dfrac{}{\vdash \{I[sum \mapsto sum+tm][tm \mapsto tm+2]\}\, tm := tm+2\, \{I[sum \mapsto sum+tm]\}}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2\, \{I[sum \mapsto sum+tm]\}}$$

$$\dfrac{\vdash \{I'\}\ i := i+1;\ tm := tm+2\, \{I[sum \mapsto sum+tm]\} \qquad \dfrac{}{\vdash \{I[sum \mapsto sum+tm]\}\, sum := sum+tm\ \{I\}}}{\vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum := sum+tm\ \{I\}}$$

$$\dfrac{I \wedge sum \leq a \longrightarrow I' \qquad \vdash \{I'\}\ i := i+1;\ tm := tm+2;\ sum := sum+tm\ \{I\} \qquad I \longrightarrow I}{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}$$

$$\dfrac{a \geq 0 \wedge A \longrightarrow I \qquad \dfrac{\vdash \{I \wedge sum \leq a\}\ body\ \{I\}}{\vdash \{I\}\ WHILE\ sum \leq a\ DO\ body\ \{a < sum \wedge I\}} \qquad a < sum \wedge I \longrightarrow post}{\vdash \{a \geq 0 \wedge A\}\ WHILE\ sum \leq a\ DO\ body\ \{post\}}$$

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- Our proof boils down to the constraints:

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I' \longrightarrow I''[i \mapsto i+1]$

$I'' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2]$

$I \wedge sum \leq a \longrightarrow I'$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I' \longrightarrow I''[i \mapsto i+1]$

$I'' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2]$

Solution $I'' \equiv I[sum \mapsto sum+tm][tm \mapsto tm+2]$

$I \wedge sum \leq a \longrightarrow I'$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I'$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$I \wedge sum \leq a \longrightarrow I'$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I' \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

Solution $I' \equiv I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$I \wedge sum \leq a \longrightarrow I'$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$ "Invariant is preserved in body"

$a \geq 0 \wedge A \longrightarrow I$

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$ “Invariant is preserved in body”

$a \geq 0 \wedge A \longrightarrow I$ “Invariant initially holds at loop entry”

$a < sum \wedge I \longrightarrow post$

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$ “Invariant is preserved in body”

$a \geq 0 \wedge A \longrightarrow I$ “Invariant initially holds at loop entry”

$a < sum \wedge I \longrightarrow post$ “Invariant at loop exit implies post”

Hoare - Logic: A Proof System for Programs

- Our proof boils down to the constraints:

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$ "Invariant is preserved in body"

$a \geq 0 \wedge A \longrightarrow I$ "Invariant initially holds at loop entry"

Recall: $\ldots \equiv a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$

$a < sum \wedge I \longrightarrow post$ "Invariant at loop exit implies post"

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$i \geq 0$

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$i \geq 0$

$tm \geq 1$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

$i \geq 0$

$tm \geq 1$

$sum \geq 1$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

$i \geq 0$

$tm \geq 1$

$sum \geq 1$

$tm = 2*i + 1$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

$i \geq 0$

$tm \geq 1$

$sum \geq 1$

$tm = 2*i + 1$

$sum = \sum_{k=0}^{i} (2*k + 1)$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

$i \geq 0$

$tm \geq 1$

$sum \geq 1$

$tm = 2*i + 1$

$sum = \sum_{k=0}^{i} (2*k + 1)$

$sum = (i+1)^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

$i \geq 0$

$tm \geq 1$

$sum \geq 1$

$tm = 2*i + 1$

$sum = \sum_{k=0}^{i} (2*k + 1)$

$sum = (i+1)^2$

$a \geq i^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

- Our proof boils further down to finding the invariant I

$$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (1)

$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$

$a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$

$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$

Hoare - Logic: A Proof System for Programs

- We check our invariant (1)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv \text{sum} = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \wedge sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1$$

$$I \land sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum+tm+2 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land tm+2 = 2*(i+1) + 1 \land tm+2 \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$I \wedge sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \wedge sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \wedge sum \leq a$$
$$\longrightarrow sum+tm+2 = ((i+1)+1)^2 \wedge a \geq (i+1)^2 \wedge tm+2 = 2*(i+1) + 1 \wedge tm+2 \geq 1$$

$$\equiv \quad sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \wedge sum \leq a$$
$$\longrightarrow (i+1)^2 + 2*(i+1)+1 = ((i+1)+1)^2 \wedge a \geq (i+1)^2 \wedge (2*i + 1)+2 = 2*(i+1) + 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1$$

$$I \land sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum+tm+2 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land tm+2 = 2*(i+1) + 1 \land tm+2 \geq 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow (i+1)^2 + 2*(i+1)+1 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land (2*i + 1)+2 = 2*(i+1) + 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow a \geq (i+1)^2$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1$$

$$I \land sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum+tm+2 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land tm+2 = 2*(i+1) + 1 \land tm+2 \geq 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow (i+1)^2 + 2*(i+1)+1 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land (2*i + 1)+2 = 2*(i+1) + 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow a \geq (i+1)^2$$

$$\equiv \text{True}$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 1)

$$I \equiv sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1$$

$$I \land sum \leq a \longrightarrow I[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1[sum \mapsto sum+tm][tm \mapsto tm+2][i \mapsto i+1]$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow sum+tm+2 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land tm+2 = 2*(i+1) + 1 \land tm+2 \geq 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow (i+1)^2 + 2*(i+1)+1 = ((i+1)+1)^2 \land a \geq (i+1)^2 \land (2*i + 1)+2 = 2*(i+1) + 1$$

$$\equiv \quad sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \land sum \leq a$$
$$\longrightarrow a \geq (i+1)^2$$

$$\equiv \text{True}$$

Invariant preserved

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow 1 = (0+1)^2 \wedge a \geq 0^2 \wedge 1 = 2*0 + 1 \wedge 1 \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \;\equiv\; sum = (i+1)^2 \;\wedge\; a \geq i^2 \;\wedge\; tm = 2*i + 1 \;\wedge\; tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow 1 = (0+1)^2 \wedge a \geq 0^2 \wedge 1 = 2*0 + 1 \wedge 1 \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow a \geq 0 \wedge 1 = 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow 1 = (0+1)^2 \wedge a \geq 0^2 \wedge 1 = 2*0 + 1 \wedge 1 \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow a \geq 0 \wedge 1 = 1$$

$$\equiv \text{True}$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 2)

$$I \;\equiv\; sum = (i+1)^2 \;\wedge\; a \geq i^2 \;\wedge\; tm = 2*i + 1 \;\wedge\; tm \geq 1$$

$$a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1 \longrightarrow I$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow 1 = (0+1)^2 \wedge a \geq 0^2 \wedge 1 = 2*0 + 1 \wedge 1 \geq 1$$

$$\equiv \quad a \geq 0 \wedge a \geq 0 \wedge i=0 \wedge tm=1 \wedge sum=1$$
$$\longrightarrow a \geq 0 \wedge 1 = 1$$

$$\equiv \text{True}$$

Invariant initially holds

Hoare - Logic: A Proof System for Programs

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv \text{sum} = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$\equiv \quad a < sum \wedge sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$\equiv \quad a < sum \wedge sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$\equiv \quad a < sum \wedge sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \longrightarrow a < sum$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1$$

$$a < sum \wedge I \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$\equiv \quad a < sum \wedge sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \longrightarrow i^2 \leq a \wedge a < (i+1)^2$$

$$\equiv \quad a < sum \wedge sum = (i+1)^2 \wedge a \geq i^2 \wedge tm = 2*i + 1 \wedge tm \geq 1 \longrightarrow a < sum$$

$$\equiv \quad \text{True}$$

Hoare - Logic: A Proof System for Programs

- We check our invariant (constraint 3)

$$I \equiv sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1$$

$$a < sum \land I \longrightarrow i^2 \leq a \land a < (i+1)^2$$

$$\equiv \quad a < sum \land sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \longrightarrow i^2 \leq a \land a < (i+1)^2$$

$$\equiv \quad a < sum \land sum = (i+1)^2 \land a \geq i^2 \land tm = 2*i + 1 \land tm \geq 1 \longrightarrow a < sum$$

$$\equiv \quad \text{True}$$

Invariant implies post-condition

Hoare - Logic: A Proof System for Programs

Hoare – Logic: A Proof System for Programs

- We check termination:

Hoare – Logic: A Proof System for Programs

- We check termination:
  - We provide a function m that decreases for the program state (a, i , tm, sum) for any possible loop traversal (i.e. $sum \leq a \wedge I$), i.e.

Hoare – Logic: A Proof System for Programs

- We check termination:
  - We provide a function m that decreases for the program state (a, i , tm, sum) for any possible loop traversal (i.e. $sum \leq a \wedge I$), i.e.

$$sum \leq a \wedge I \longrightarrow m(a, i, tm, sum) > m(a, i+1, tm+2, sum+tm)$$

Hoare - Logic: A Proof System for Programs

- We check termination:
  - We provide a function m that decreases for the program state (a, i , tm, sum) for any possible loop traversal (i.e. $sum \leq a \wedge I$), i.e.

    $$sum \leq a \wedge I \longrightarrow m(a, i, tm, sum) > m(a, i+1, tm+2, sum+tm)$$

  - Iff such a function m (a measure) exists, the loop will terminate.

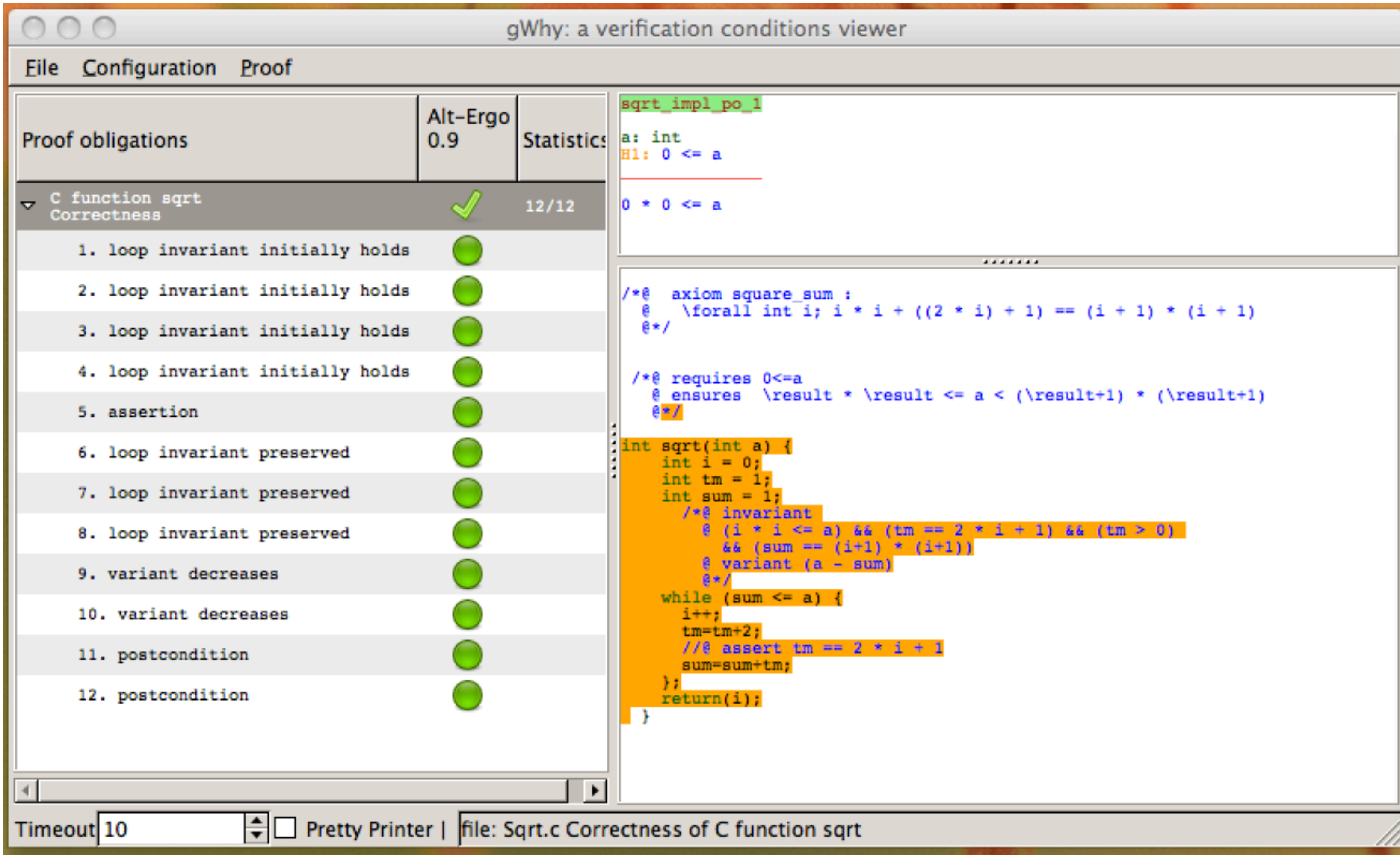
Hoare – Logic: A Proof System for Programs

- We check termination:
  - We provide a function m that decreases for the program state (a, i , tm, sum) for any possible loop traversal (i.e. $sum \leq a \wedge I$), i.e.

    $$sum \leq a \wedge I \longrightarrow m(a, i, tm, sum) > m(a, i+1, tm+2, sum+tm)$$

  - Iff such a function m (a measure) exists, the loop will terminate.
  - A candidate for m: $m(a, i, tm, sum) \equiv a - i$ which obviously decreases.

Tools: gwhy and Squareroot



Hoare-Logic : Summary

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)
  - prove partial correctness of the spec for the program via a Hoare-Calculus (or an alternative such as the wp-calculus)

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)
  - prove partial correctness of the spec for the program via a Hoare-Calculus (or an alternative such as the wp-calculus)

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)
  - prove partial correctness of the spec for the program via a Hoare-Calculus (or an alternative such as the wp-calculus)

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)
  - prove partial correctness of the spec for the program via a Hoare-Calculus (or an alternative such as the wp-calculus)

total correctness = partial correctness + termination ...

Hoare-Logic : Summary

- Note: Validity is a « partial correctness notion »

  proof under condition that the program terminates.
  For non-terminating programs, the calculus allows
  to prove anything

- The Deductive Proof-Method is therefore two-staged:
  - verify termination (find mesures for loops and recursive calls that strictly decrease for each iteration)
  - prove partial correctness of the spec for the program via a Hoare-Calculus (or an alternative such as the wp-calculus)

total correctness = partial correctness + termination ...

Hoare - Logic: Summary

- … in the essence, the Hoare Calculus is an entirely syntactic game that constructs a labelling of the program with assertions …

Hoare - Logic: Summary

Hoare - Logic: Summary

Formal Proof

- Can be very hard - up to infeasible (nobody will probably ever prove the correctness of MS Word!)

- But still, the proof-task can be automated to a large extent.

Recall: What are the limits of tests

Recall: What are the limits of tests

- Assumptions on „Testability“

  (system under test must behave deterministically,
  or have controlled non-determinism, must be initializable)

Recall: What are the limits of tests

- Assumptions on „Testability“

  (system under test must behave deterministically,
  or have controlled non-determinism, must be initializable)

- Assumptions like Test-Hypothesis

  (Uniform / Regular behaviour is sometimes
  a „realistic“ assumption, but not always)

Recall: What are the limits of tests

- Assumptions on „Testability“

  (system under test must behave deterministically,
  or have controlled non-determinism, must be initializable)

- Assumptions like Test-Hypothesis

  (Uniform / Regular behaviour is sometimes
  a „realistic“ assumption, but not always)

- Limits in perfection:

  We know only up to a given "certainty" that the
  program meets the specification ...