An Introduction to Program Verification

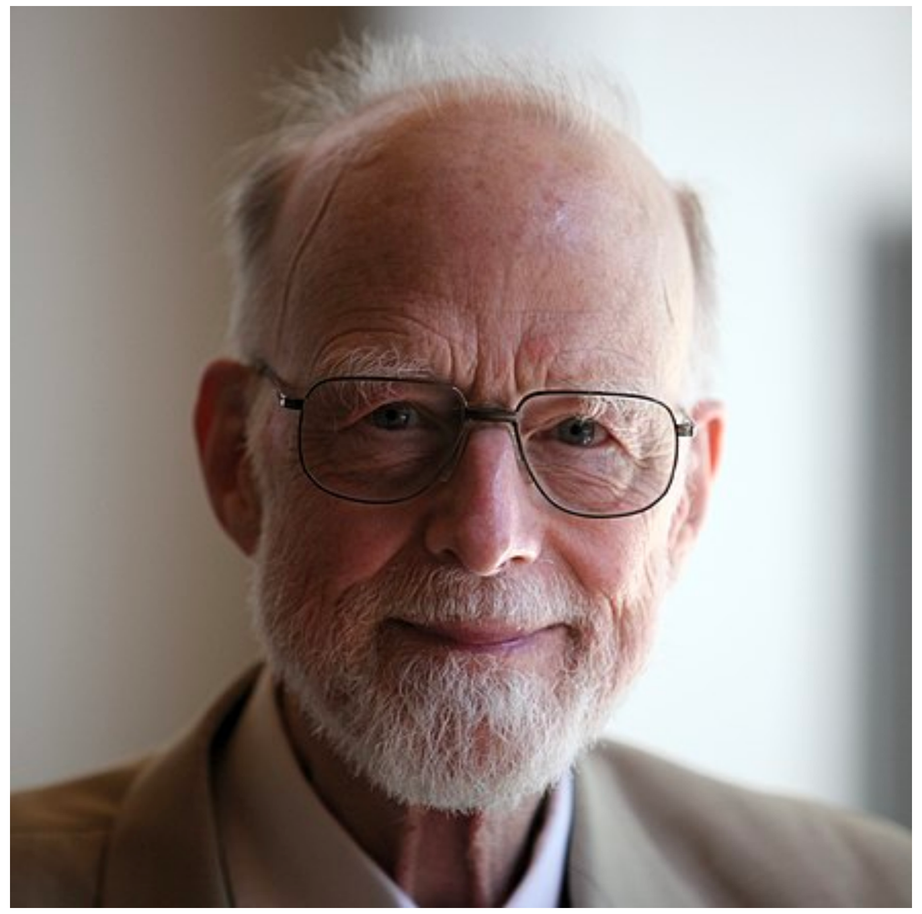
Nuno Macedo

Program Verification

- Main goal: to prove that an **implementation** is correct
- Verify (imperative) code against a formal specification
- **Undecidable** problem in general
- Two main approaches:
  - **Model checking** (imposes a bound on search space)
  - **Deductive verification** (semi-automatic, may require user input)

Hoare Logic

- A logic to reason about computer programs
- Hoare logic considers programs explicitly (*exogenous*)
  - More natural when verifying sequential programs
  - Contrast with temporal logics, that only consider states (*endogenous*)

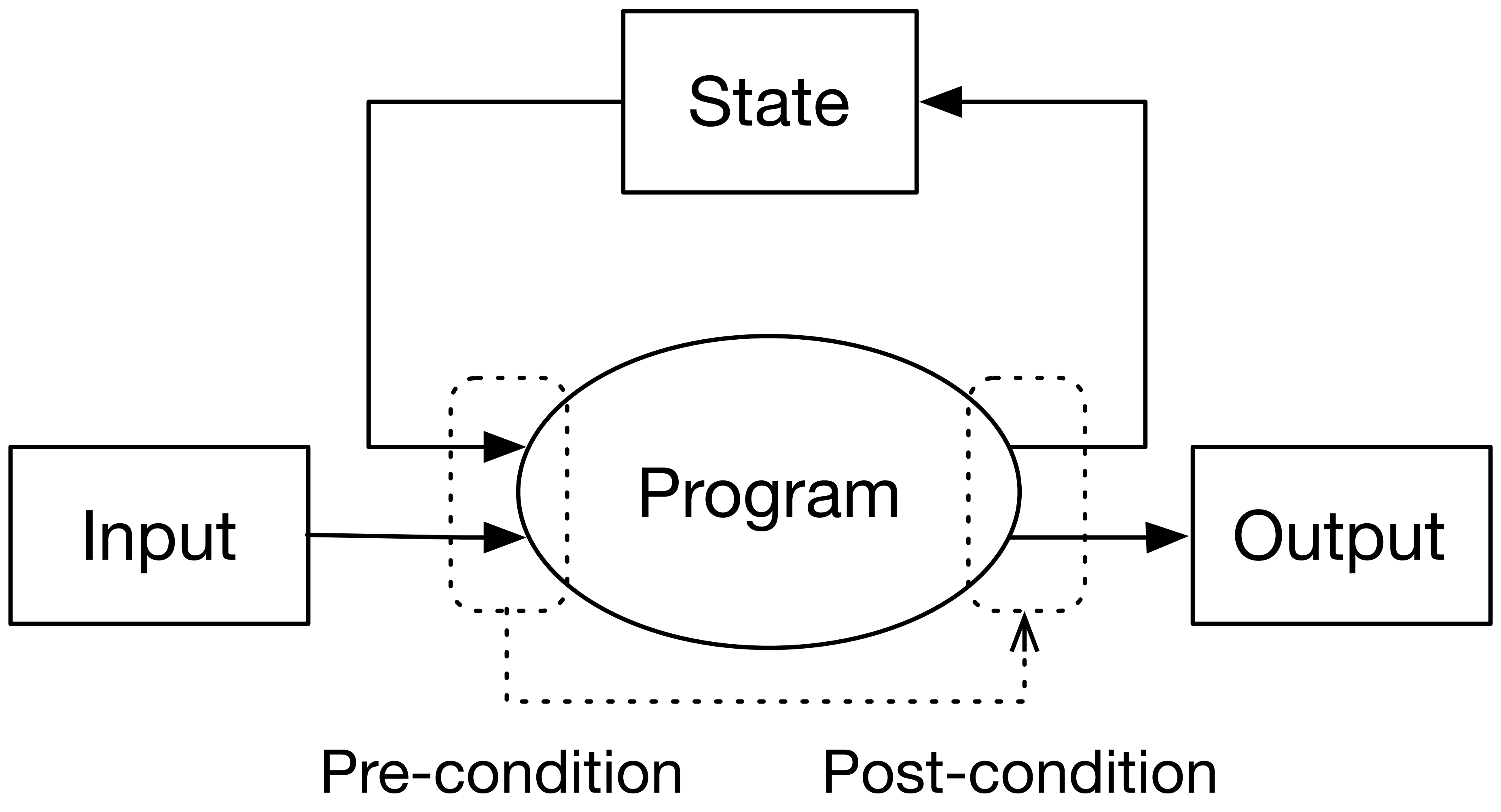


“*In the development of the understanding of complex phenomena, the most powerful tool available to the human intellect is **abstraction**.*“

Tony Hoare, Turing Award

Hoare Logic

- Allows the (formal) specification of the expected behaviour of a program fragment
- Dealing with sequential executions, annotations take the shape of **pre** and **postconditions**
- Conditions over the program **state** and **input/ output** variables
- Postconditions may refer to the initial state
- Usually in some restricted first-order logic



Hoare Logic

double sqrt(**double** x)

Preconditions:

-

Postconditions:

-

void sort(T[] a)

Preconditions:

-

Postconditions:

-

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

-

void sort(T[] a)

Preconditions:

-

Postconditions:

-

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

- The square of the result is x

void sort(T[] a)

Preconditions:

-

Postconditions:

-

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

- The square of the result is x
- Non-negative result?

void sort(T[] a)

Preconditions:

-

Postconditions:

-

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

- The square of the result is x
- Non-negative result?

void sort(T[] a)

Preconditions:

- Non-null a
- No null elements in array

Postconditions:

-

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

- The square of the result is x
- Non-negative result?

void sort(T[] a)

Preconditions:

- Non-null a
- No null elements in array

Postconditions:

- Result is sorted

Hoare Logic

double sqrt(**double** x)

Preconditions:

- Non-negative x

Postconditions:

- The square of the result is x
- Non-negative result?

void sort(T[] a)

Preconditions:

- Non-null a
- No null elements in array

Postconditions:

- Result is sorted
- Result is a permutation of a

Hoare Triples

- In Hoare logic, this information is represented as a **Hoare triple**

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

  - ***P*:** precondition on program states
  - ***S*:** a statement (code snippet)
  - ***Q*:** postcondition on program states

- Assumes input and output variables in the state

Program Correctness

- Given a Hoare Triple {*P*} *S* {*Q*}, two views on correctness
- **Partial correctness**: if the execution of *S* on a program state satisfying *P* terminates, then the resulting program state satisfies *Q*
- **Total correctness**: the execution of *S* on a program state satisfying *P* terminates and the resulting program state satisfies *Q*
- Total correctness = Partial correctness + Termination

Creating Hoare Triples

- For a program S, multiple P and Q conditions for which the triple is true
  - E.g., $\{\bot\}\ S\ \{Q\}$ is valid for any S and Q
  - What about $\{P\}\ S\ \{\top\}$?
- Usually we want a weak P (more inputs) and a strong Q (restricted output)
  - the **weakest precondition** such that S guarantees Q is wp(S, Q)
  - the **strongest postcondition** resulting from running S on P is sp(P, S)
- For conditions P and Q there are also multiple acceptable programs S
  - Allows programs to be **refined** from design until implementation

Simple Language

E ::= n | x | $E + E$ | $E * E$ | $E - E$ | … // *Expressions*

B ::= **true** | **false** | $E == E$ | $E < E$ | B && B | B || B | !B | … // *Conditionals*

C ::= $x := E$ // *Statements*

| C ; C

| **if** B **then** C **else** C

| **while** B **do** C

| **skip**

where $n \in \mathbb{Z}$ integers, $x \in \mathbb{V}$ variables

Examples

- Which of the following Hoare triples are true?
- Which have a weakest precondition?
  - $\{x = 0\}$ x := 1 $\{x = 1\}$
  - $\{x = y\}$ x := x+1 $\{x = y+1\}$
  - $\{\top\}$ z := x; x := y; y := z $\{x = y \wedge y = x\}$
  - $\{x = -1\}$ **if** x < 0 **then** x := -x **else skip** $\{x = 1\}$
  - $\{\top\}$ **if** x > y **then** x := y **else skip** $\{x \leq y\}$
  - $\{\top\}$ **while** i > 0 **do** i := i-1$\{i = 0\}$

Examples

- Which of the following Hoare triples are true?
- Which have a weakest precondition?
    - $\{\top\}\ x := 1\ \{x = 1\}$
    - $\{x = y\}\ x := x+1\ \{x = y+1\}$
    - $\{x = x_0 \wedge y = y_0\}\ z := x;\ x := y;\ y := z\ \{x = y_0 \wedge y = x_0\}$
    - $\{x = -1 \vee x = 1\}$ **if** $x < 0$ **then** $x := -x$ **else skip** $\{x = 1\}$
    - $\{\top\}$ **if** $x > y$ **then** $x := y$ **else skip** $\{x \leq y\}$
    - $\{I > 0\}$ **while** $i > 0$ **do** $i := i-1\{i = 0\}$

Inference Rules

- To mechanise proofs, there is a set of axioms and inference rules based on the semantics the statements

- They have the shape

$$\frac{\{P_1\}\ S_1\ \{Q_1\} \quad \ldots \quad \{P_2\}\ S_2\ \{Q_2\}}{\{P\}\ S\ \{Q\}}$$

- Meaning that if the **premises** $\{P_1\}\ S_1\ \{Q_1\}$, …, $\{P_1\}\ S_1\ \{Q_1\}$ are true, so is the **conclusion** $\{P\}\ S\ \{Q\}$

- Axioms are rules without premises

Inference Rules

$$\frac{}{\{P\}\ \textbf{skip}\ \{P\}}\ R_N$$

$$\frac{}{\{Q[E/x]\}\ x := E\ \{Q\}}\ R_A$$

$$\frac{\{P \land C\}\ S_1\ \{Q\} \qquad \{P \land \neg C\}\ S_2\ \{Q\}}{\{P\}\ \textbf{if}\ \mathrm{C}\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \{Q\}}\ R_C$$

$$\frac{\{P\}\ S_1\ \{M\} \qquad \{M\}\ S_2\ \{Q\}}{\{P\}\ S_1;\ S_2\ \{Q\}}\ R_S$$

where Q[E/x] means replacing occurrences of x by E

Inference Rules

$$\frac{\{I \ \wedge \ C \ \wedge \ V = V_0\}\ S\ \{I \ \wedge \ 0 \leq V < V_0\}}{\{I\}\ \textbf{while}\ C\ \textbf{do}\ S\ \{I \wedge \neg C\}}\ R_L$$

- *I*: loop **invariant**, holds before, after, and in every iteration of the loop
- *V*: loop **variant**, non-negative strictly decreasing integer, ensures termination

Inference Rules

$$\frac{P_1 \rightarrow P_2 \qquad \{P_2\}\ S\ \{Q\}}{\{P_1\}\ S\ \{Q\}}\ R_{SP}$$

$$\frac{Q_1 \rightarrow Q_2 \qquad \{P\}\ S\ \{Q_1\}}{\{P\}\ S\ \{Q_2\}}\ R_{WQ}$$

Weakest Precondition Calculus

- In practice, inference rules are not applied directly when mechanising proofs

- Instead, a technique (by Dijkstra) based on **calculating weakest preconditions** is used

- Main idea: calculate the weakest precondition for *S*, and and see if the provided *P* is as strong

$$\{P\}\ S\ \{Q\} \text{ is true} \quad \text{iff} \quad P \rightarrow \mathrm{wp}(S, Q)$$

Weakest Precondition Calculus

W_N $\quad$ wp(**skip**, Q) = Q

W_A $\quad$ wp(x := E, Q) = $Q[E/x]$

W_S $\quad$ wp(S_1; S_2, Q) = wp(S_1, wp(S_2, Q))

W_C $\quad$ wp(**if** C **then** S_1 **else** S_2, Q) = $(C \wedge \mathrm{wp}(S_1,Q)) \vee (\neg C \wedge \mathrm{wp}(S_2,Q))$

W_L $\quad$ wp(**while** C **do** S, Q) = $P_0 \vee P_1 \vee \ldots$ where

$P_0 = \neg C \wedge Q$, $P_k = C \wedge \mathrm{wp}(S, P_{k-1})$ for $k > 0$

Proof Procedure (no loops)

- Without loops, proofs can be built automatically by tools
- To prove $\{P\}\ S\ \{Q\}$:
  1. Calculate wp(S,Q) using the first four rules above
  2. Prove that $P \rightarrow$ wp(S,Q)

Example (no loops)

$\{x > 0\}$ x := x+y; y := x-y $\{y > 0\}$

1. wp(S,Q)

   wp(x := x+y; y := x-y, y > 0)

   = wp(x := x+y, wp(y := x-y, y > 0)) W_S

2. $P \rightarrow$ wp(S,Q)

$\{x > 0\}$

x := x+y;

y := x-y

$\{y > 0\}$

Example (no loops)

{x > 0} x := x+y; y := x-y {y > 0}

1. wp(S,Q)

   wp(x := x+y; y := x-y, y > 0)

   = wp(x := x+y, wp(y := x-y, y > 0)) W_S

   = wp(x := x+y, x-y > 0) W_A

2. $P \rightarrow$ wp(S,Q)

{x > 0}

x := x+y;

y := x-y

{y > 0}

Example (no loops)

$\{x > 0\}\ x := x+y;\ y := x-y\ \{y > 0\}$

1. wp(S,Q)

	wp(x := x+y; y := x-y, y > 0)	
=	wp(x := x+y, wp(y := x-y, y > 0))	W_S
=	wp(x := x+y, x-y > 0)	W_A
=	x+y-y > 0	W_A
=	x > 0	*simp*

2. $P \rightarrow$ wp(S,Q)

{x > 0}

x := x+y;

y := x-y

{y > 0}

Example (no loops)

$\{x > 0\}\ x := x+y;\ y := x-y\ \{y > 0\}$

1. wp(*S*,*Q*)

	wp(x := x+y; y := x-y, y > 0)	
=	wp(x := x+y, wp(y := x-y, y > 0))	W_S
=	wp(x := x+y, x-y > 0)	W_A
=	x+y-y > 0	W_A
=	x > 0	*simp*

2. $P \rightarrow$ wp(*S*,*Q*)

   $x > 0 \rightarrow x > 0$

{x > 0}

x := x+y;

y := x-y

{y > 0}

Example (no loops)

$\{x > 0\}$ x := x+y; y := x-y $\{y > 0\}$

1. wp(S,Q)

$$
\begin{array}{lll}
 & \text{wp}(x := x+y;\ y := x-y,\ y > 0) & \\
= & \text{wp}(x := x+y,\ \text{wp}(y := x-y,\ y > 0)) & W_S \\
= & \text{wp}(x := x+y,\ x-y > 0) & W_A \\
= & x+y-y > 0 & W_A \\
= & x > 0 & \textit{simp}
\end{array}
$$

2. $P \rightarrow$ wp(S,Q)

$$
\begin{array}{ll}
 & x > 0 \rightarrow x > 0 \\
\equiv & \top
\end{array}
$$

$\{x > 0\}$

x := x+y;

y := x-y

$\{y > 0\}$

Example (no loops)

$\{x > 0\}\ x := x+y;\ y := x-y\ \{y > 0\}$

1. $wp(S,Q)$

$$
\begin{array}{lll}
 & wp(x := x+y;\ y := x-y,\ y > 0) & \\
= & wp(x := x+y,\ wp(y := x-y,\ y > 0)) & W_S \\
= & wp(x := x+y,\ x-y > 0) & W_A \\
= & x+y-y > 0 & W_A \\
= & x > 0 & \textit{simp}
\end{array}
$$

2. $P \rightarrow wp(S,Q)$

$$
\begin{array}{ll}
 & x > 0 \rightarrow x > 0 \\
\equiv & \top
\end{array}
$$

$\{x > 0\}$

$x := x+y;$

$\{x-y > 0\}$

$y := x-y$ $\qquad W_A$

$\{y > 0\}$

Example (no loops)

$$\{x > 0\}\ x := x+y;\ y := x-y\ \{y > 0\}$$

1. $wp(S,Q)$

$$
\begin{array}{lll}
 & wp(x := x+y;\ y := x-y,\ y > 0) & \\
= & wp(x := x+y,\ wp(y := x-y,\ y > 0)) & W_S \\
= & wp(x := x+y,\ x-y > 0) & W_A \\
= & x+y-y > 0 & W_A \\
= & x > 0 & \textit{simp}
\end{array}
$$

2. $P \rightarrow wp(S,Q)$

$$
\begin{array}{ll}
 & x > 0 \rightarrow x > 0 \\
\equiv & \top
\end{array}
$$

$\{x > 0\}$

$\{x > 0\}$

simp

$\{x+y-y > 0\}$

$x := x+y;$

W_A

$\{x-y > 0\}$

$y := x-y$

$\{y > 0\}$

Example (no loops)

{x > 0} x := x+y; y := x-y {y > 0}

1. wp(S,Q)

   wp(x := x+y; y := x-y, y > 0)

   = wp(x := x+y, wp(y := x-y, y > 0)) W_S

   = wp(x := x+y, x-y > 0) W_A

   = x+y-y > 0 W_A

   = x > 0 *simp*

2. $P \rightarrow$ wp(S,Q)

   x > 0 → x > 0

   ≡ ⊤

{x > 0} →

{x > 0}

{x+y-y > 0}

x := x+y;

{x-y > 0}

y := x-y

{y > 0}

Example (no loops)

{⊤} **if** x < 0 **then** x := -x **else skip** {x ≥ 0}

1. wp(*S*,*Q*)

   wp(**if** x < 0 **then** x := -x **else skip**, x ≥ 0)

2. *P* → wp(*S*,*Q*)

```
{⊤}

if x < 0 then

    x := x-1

else

    skip

{x ≥ 0}
```

Example (no loops)

$\{\top\}$ **if** x < 0 **then** x := -x **else skip** $\{x \geq 0\}$

1. wp(S,Q)

   wp(**if** x < 0 **then** x := -x **else skip**, x ≥ 0)

   = (x < 0 ∧ wp(x := -x, x ≥ 0)) ∨

   (x ≥ 0 ∧ wp(**skip**, x ≥ 0) W_C

2. $P \rightarrow$ wp(S,Q)

$\{\top\}$

if x < 0 **then**

x := x-1

else

skip

$\{x \geq 0\}$

Example (no loops)

$\{\top\}$ **if** x < 0 **then** x := -x **else skip** $\{x \geq 0\}$

1. wp(S,Q)

$$
\begin{array}{lll}
 & \text{wp}(\textbf{if } x < 0 \textbf{ then } x := -x \textbf{ else skip}, x \geq 0) & \\
= & (x < 0 \land \text{wp}(x := -x, x \geq 0)) \lor & \\
 & (x \geq 0 \land \text{wp}(\textbf{skip}, x \geq 0) & W_C \\
= & (x < 0 \land -x \geq 0) \lor (x \geq 0 \land x \geq 0) & W_A, W_N \\
= & x < 0 \lor x \geq 0 & \textit{simp} \\
= & \top & \textit{simp}
\end{array}
$$

2. $P \rightarrow$ wp(S,Q)

$\{\top\}$

if x < 0 **then**

x := x-1

else

skip

$\{x \geq 0\}$

Example (no loops)

{⊤} **if** x < 0 **then** x := -x **else skip** {x ≥ 0}

1. wp(S,Q)

   wp(**if** x < 0 **then** x := -x **else skip**, x ≥ 0)

   = (x < 0 ∧ wp(x := -x, x ≥ 0)) ∨

   (x ≥ 0 ∧ wp(**skip**, x ≥ 0) *W_C*

   = (x < 0 ∧-x ≥ 0) ∨ (x ≥ 0 ∧ x ≥ 0) *W_A, W_N*

   = x < 0 ∨ x ≥ 0 *simp*

   = ⊤ *simp*

2. P → wp(S,Q)

   ⊤ → ⊤

   ≡ ⊤

{⊤}

if x < 0 **then**

  x := x-1

else

  skip

{x ≥ 0}

Example (no loops)

$\{\top\}$ **if** $x < 0$ **then** $x := -x$ **else skip** $\{x \geq 0\}$

1. wp(S,Q)

$$
\begin{array}{lll}
 & \mathrm{wp}(\textbf{if } x < 0 \textbf{ then } x := -x \textbf{ else skip}, x \geq 0) & \\
= & (x < 0 \wedge \mathrm{wp}(x := -x, x \geq 0)) \quad \vee & \\
 & (x \geq 0 \wedge \mathrm{wp}(\textbf{skip}, x \geq 0) & W_C \\
= & (x < 0 \wedge -x \geq 0) \vee (x \geq 0 \wedge x \geq 0) & W_A, W_N \\
= & x < 0 \vee x \geq 0 & \textit{simp} \\
= & \top & \textit{simp}
\end{array}
$$

2. $P \rightarrow \mathrm{wp}(S,Q)$

$$
\begin{array}{ll}
 & \top \rightarrow \top \\
\equiv & \top
\end{array}
$$

$\{\top\}$

if $x < 0$ **then**

$x := x-1$

$\{x \geq 0\}$

else

W_C

skip

$\{x \geq 0\}$

$\{x \geq 0\}$

Example (no loops)

$\{\top\}$ **if** x < 0 **then** x := -x **else skip** $\{x \geq 0\}$

1. wp(S,Q)

	wp(**if** x < 0 **then** x := -x **else skip**, x ≥ 0)	
=	(x < 0 ∧ wp(x := -x, x ≥ 0)) ∨	
	(x ≥ 0 ∧ wp(**skip**, x ≥ 0)	W_C
=	(x < 0 ∧-x ≥ 0) ∨ (x ≥ 0 ∧ x ≥ 0)	$W_{A,}$ W_N
=	x < 0 ∨ x ≥ 0	*simp*
=	⊤	*simp*

2. $P \rightarrow$ wp(S,Q)

	⊤ → ⊤
≡	⊤

$\{\top\}$

if x < 0 **then**

  {x ≥ 0}

  x := x-1 W_A

  {x ≥ 0}

else

  {x ≥ 0}

  skip W_N

  {x ≥ 0}

{x ≥ 0}

Example (no loops)

$\{\top\}$ **if** x < 0 **then** x := -x **else skip** $\{x \geq 0\}$

1. wp(S,Q)

   wp(**if** x < 0 **then** x := -x **else skip**, x ≥ 0)

   $= (x < 0 \wedge \text{wp}(x := -x, x \geq 0)) \vee$

   $(x \geq 0 \wedge \text{wp}(\textbf{skip}, x \geq 0)$ — W_C

   $= (x < 0 \wedge -x \geq 0) \vee (x \geq 0 \wedge x \geq 0)$ — W_A, W_N

   $= x < 0 \vee x \geq 0$ — *simp*

   $= \top$ — *simp*

2. $P \rightarrow \text{wp}(S,Q)$

   $\top \rightarrow \top$

   $\equiv \top$

$\{\top\}$

$\{\top\}$ *simp*

$\{(x < 0 \wedge -x \geq 0) \vee (x \geq 0 \wedge x \geq 0)\}$

if x < 0 **then**

$\{x \geq 0\}$

x := x-1

$\{x \geq 0\}$

else W_C

$\{x \geq 0\}$

skip

$\{x \geq 0\}$

$\{x \geq 0\}$

Example (no loops)

$\{\top\}$ **if** $x < 0$ **then** $x := -x$ **else skip** $\{x \geq 0\}$

1. $\mathrm{wp}(S, Q)$

$$
\begin{aligned}
& \mathrm{wp}(\textbf{if } x < 0 \textbf{ then } x := -x \textbf{ else skip}, x \geq 0) & \\
= \; & (x < 0 \wedge \mathrm{wp}(x := -x, x \geq 0)) \;\; \vee & \\
& (x \geq 0 \wedge \mathrm{wp}(\textbf{skip}, x \geq 0)) & W_C \\
= \; & (x < 0 \wedge -x \geq 0) \vee (x \geq 0 \wedge x \geq 0) & W_A, W_N \\
= \; & x < 0 \vee x \geq 0 & \mathit{simp} \\
= \; & \top & \mathit{simp}
\end{aligned}
$$

2. $P \rightarrow \mathrm{wp}(S, Q)$

$$
\begin{aligned}
& \top \rightarrow \top \\
\equiv \; & \top
\end{aligned}
$$

$\{\top\}$ $\rightarrow$

$\{\top\}$

$\{(x < 0 \wedge -x \geq 0) \vee (x \geq 0 \wedge x \geq 0)\}$

if $x < 0$ **then**

    $\{x \geq 0\}$

    $x := x-1$

    $\{x \geq 0\}$

else

    $\{x \geq 0\}$

    skip

    $\{x \geq 0\}$

$\{x \geq 0\}$

Proof Procedure (loops)

- The weakest precondition of a loop is too difficult to find
- Procedure relies the inference rule, which requires additional **user intervention**
- To prove $\{P\}$ **while** C **do** S $\{Q\}$:
  1. Find a loop invariant I (based on loop understanding and proof obligations below)
  2. Find a loop invariant V (based on loop understanding, variables o C updated in S)
  3. Prove that
     A. $P \rightarrow I$ *invariant holds in the beginning*
     B. $I \wedge \neg C \rightarrow Q$ *postcondition holds when the loop terminates*
     C. $\{I \wedge C \wedge V = V_0\}\ S\ \{I \wedge 0 \leq V < V_0\}$ *invariant is preserved and variant decreases*

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

1. $I = n \geq 0$ (looking at P and C)
2. $V = n$ (looking at what the loop does)
3. Proofs

    A. $P \rightarrow I$

    B. $I \wedge \neg C \rightarrow Q$

    C. $\{I \wedge C \wedge V = V_0\}$ S $\{I \wedge 0 \leq V < V_0\}$

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

1. I = n ≥ 0 (looking at P and C)
2. V = n (looking at what the loop does)
3. Proofs

   A. $P \rightarrow I$

      $n \geq 0 \rightarrow n \geq 0$

   B. $I \wedge \neg C \rightarrow Q$

   C. $\{I \wedge C \wedge V = V_0\}\ S\ \{I \wedge 0 \leq V < V_0\}$

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n\text{-}1$ $\{n = 0\}$

1. I = n ≥ 0 (looking at P and C)
2. V = n (looking at what the loop does)
3. Proofs

   A. $P \rightarrow I$

      $n \geq 0 \rightarrow n \geq 0 \equiv \top$

   B. $I \wedge \neg C \rightarrow Q$

   C. $\{I \wedge C \wedge V = V_0\}$ S $\{I \wedge 0 \leq V < V_0\}$

Example (loops)

{n ≥ 0} **while** n > 0 **do** n := n-1 {n = 0}

1. I = n ≥ 0 (looking at P and C)
2. V = n (looking at what the loop does)
3. Proofs
   A. $P \rightarrow$ I

      n ≥ 0 → n ≥ 0 ≡ ⊤

   B. $I \wedge \neg C \rightarrow Q$

      n ≥ 0 ∧ ¬(n > 0) → n = 0

   C. $\{I \wedge C \wedge V = V_0\}$ S $\{I \wedge 0 \leq V < V_0\}$

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

1. $I = n \geq 0$ (looking at P and C)
2. $V = n$ (looking at what the loop does)
3. Proofs

   A. $P \rightarrow I$

   $n \geq 0 \rightarrow n \geq 0 \equiv \top$

   B. $I \land \neg C \rightarrow Q$

   $n \geq 0 \land \neg(n > 0) \rightarrow n = 0 \equiv n \geq 0 \land n \leq 0 \rightarrow n = 0 \equiv n = 0 \rightarrow n = 0 \equiv \top$

   C. $\{I \land C \land V = V_0\}$ S $\{I \land 0 \leq V < V_0\}$

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

1. $I = n \geq 0$ (looking at P and C)
2. $V = n$ (looking at what the loop does)
3. Proofs

   A. $P \rightarrow I$

   $n \geq 0 \rightarrow n \geq 0 \equiv \top$

   B. $I \wedge \neg C \rightarrow Q$

   $n \geq 0 \wedge \neg(n > 0) \rightarrow n = 0 \equiv n \geq 0 \wedge n \leq 0 \rightarrow n = 0 \equiv n = 0 \rightarrow n = 0 \equiv \top$

   C. $\{I \wedge C \wedge V = V_0\}$ S $\{I \wedge 0 \leq V < V_0\}$

   $\{n \geq 0 \wedge n > 0 \wedge n = V_0\}$ $n := n-1$ $\{n \geq 0 \wedge 0 \leq n < V_0\}$

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

1. $I = n \geq 0$ (looking at P and C)
2. $V = n$ (looking at what the loop does)
3. Proofs

   A. $P \rightarrow I$

   $n \geq 0 \rightarrow n \geq 0 \equiv \top$

   B. $I \wedge \neg C \rightarrow Q$

   $n \geq 0 \wedge \neg(n > 0) \rightarrow n = 0 \equiv n \geq 0 \wedge n \leq 0 \rightarrow n = 0 \equiv n = 0 \rightarrow n = 0 \equiv \top$

   C. $\{I \wedge C \wedge V = V_0\}$ S $\{I \wedge 0 \leq V < V_0\}$

   $\{n \geq 0 \wedge n > 0 \wedge n = V_0\}$ $n := n-1$ $\{n \geq 0 \wedge 0 \leq n < V_0\} \equiv \{n > 0 \wedge n = V_0\}$ $n := n-1$ $\{0 \leq n < V_0\}$

   $\equiv (n > 0 \wedge n = V_0) \rightarrow (0 \leq n-1 < V_0) \equiv n > 0 \rightarrow (n \geq 1 \wedge n-1 < n) \equiv \top$

Example (loops)

{n ≥ 0} **while** n > 0 **do** n := n-1 {n = 0}

{n ≥ 0}

while n > 0 **do**

n := n-1

{n = 0}

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n-1$ $\{n = 0\}$

$\{n \geq 0\}$	
$\{n \geq 0\}$	$P \rightarrow I$ (A)
while $n > 0$ **do**	
$\{n \geq 0 \wedge n > 0 \wedge n = V_0\}$	$\{I \wedge C \wedge V = V_0\}$ (C)
$n := n-1$	
$\{n \geq 0 \wedge 0 \leq n < V_0\}$	$\{I \wedge 0 \leq V < V_0\}$ (C)
$\{n \geq 0 \wedge \neg(n > 0)\}$	$I \wedge \neg C \rightarrow Q$ (B)
$\{n = 0\}$	

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n\text{-}1$ $\{n = 0\}$

$\{n \geq 0\}$	
$\{n \geq 0\}$	$P \rightarrow I$ (A)
while $n > 0$ **do**	
$\{n \geq 0 \wedge n > 0 \wedge n = V_0\}$	$\{I \wedge C \wedge V = V_0\}$ (C)
$\{n > 0 \geq \wedge n = V_0\}$	
$\{n \geq 1 \wedge n\text{-}1 < V_0\}$	
$\{n\text{-}1 \geq 0 \wedge 0 \leq n\text{-}1 < V_0\}$	W_A
$n := n\text{-}1$	
$\{n \geq 0 \wedge 0 \leq n < V_0\}$	$\{I \wedge 0 \leq V < V_0\}$ (C)
$\{n \geq 0 \wedge \neg(n > 0)\}$	$I \wedge \neg C \rightarrow Q$ (B)
$\{n = 0\}$	

Example (loops)

$\{n \geq 0\}$ **while** $n > 0$ **do** $n := n\text{-}1$ $\{n = 0\}$

$\{n \geq 0\}$ $\rightarrow$	
$\{n \geq 0\}$	$P \rightarrow I$ (A)
while $n > 0$ **do**	
$\quad \{n \geq 0 \wedge n > 0 \wedge n = V_0\}$	$\{I \wedge C \wedge V = V_0\}$ (C)
$\quad \{n > 0 \geq \wedge\ n = V_0\}$ $\rightarrow$	
$\quad \{n \geq 1 \wedge n\text{-}1 < V_0\}$	
$\quad \{n\text{-}1 \geq 0 \wedge 0 \leq n\text{-}1 < V_0\}$	W_A
$\quad n := n\text{-}1$	
$\quad \{n \geq 0 \wedge 0 \leq n < V_0\}$	$\{I \wedge 0 \leq V < V_0\}$ (C)
$\{n \geq 0 \wedge \neg(n > 0)\}$ $\rightarrow$	$I \wedge \neg C \rightarrow Q$ (B)
$\{n = 0\}$	

Proof Tableux

- A proof with statements interleaved with conditions is called a **proof tableu**
- Intermediary conditions are **midconditions,** often with justifications to be read top-down
- Derived automatically bottom-up by weakest precondition calculus (except for loops)
- In general, the steps are:
  1. Identify and insert loop invariants and variants
  2. Calculate preconditions bottom-up
  3. Prove implications of consecutive conditions
- Tools usually automate steps 2 and 3