Program Correctness

Literatuur

Verification of Sequential and Concurrent Programs.
Krzysztof R. Apt, Frank S. de Boer, Ernst-Rüdiger Olderog.
Series: Texts in Computer Science. Springer.
3rd ed. 2nd Printing.
ISBN: 978-1-84882-744-8.

Te behandelen stof

Course Towards Object-Oriented Program Verification
(zie *Preface*: *Outlines of One-Semester Courses* en slides).
Uit bovenstaand boek behandelen we de hoofdstukken 2, 3, 4, en 5 onderverdeeld in de volgende blokken B1-3:

	Onderwerp	Secties
B1	Partiële Correctheid While Programma's	2.1, 2.2, 2.4, 2.5, 2.7, 3.1, 3.3, 3.4, 3.10, 3.11.
B2	Totale Correctheid While Programma's	3.3 en 3.4.
B3	Partiële Correctheid Recursieve Programma's	4.1, 4.3, 5.1, 5.2, 5.3.

What? Correctness? Bugs!

The TimSort Bug

See

- EU project Envsiage
- Wikipedia

Industrial Relevance

'Softwarefouten kosten Nederlandse economie jaarlijks 1,6 miljard euro'

Managerial Misconceptions:

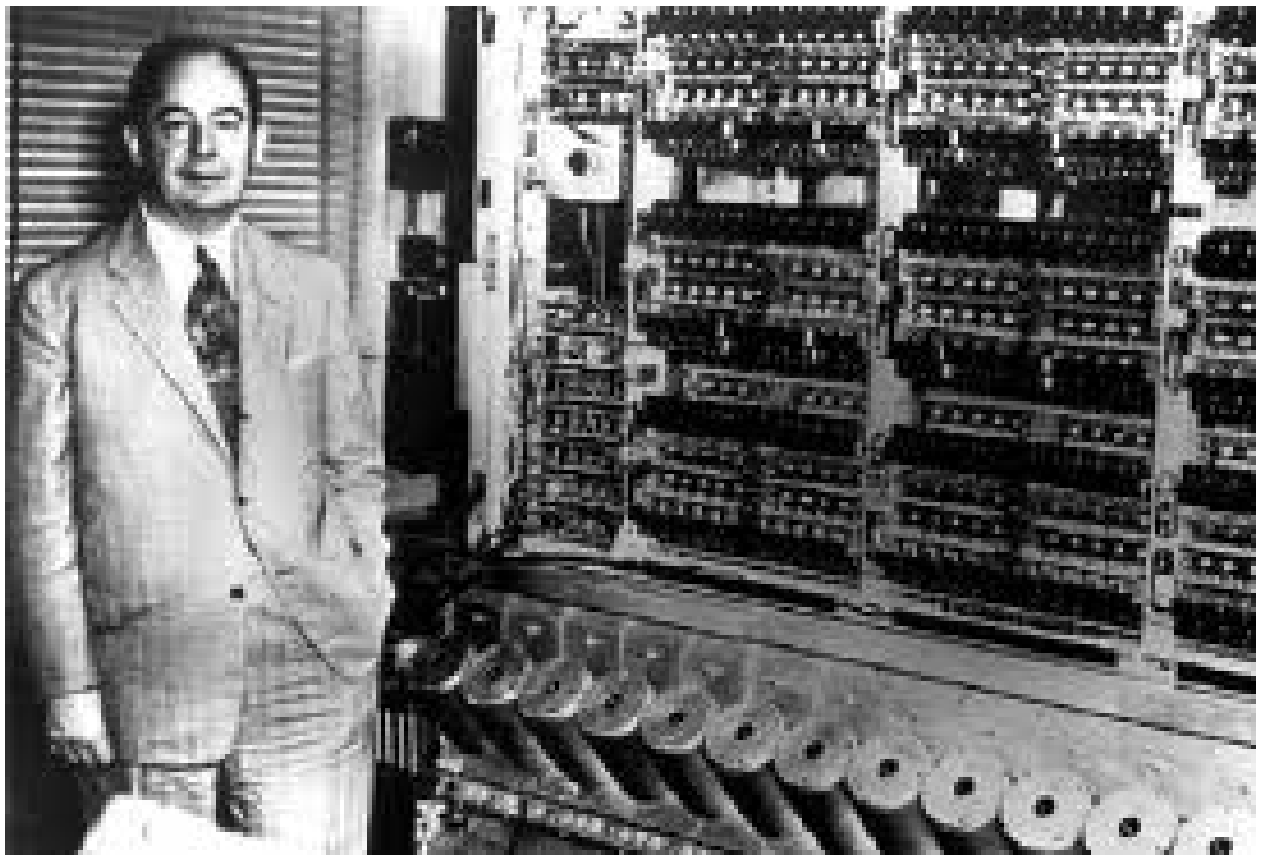
Software development is not an art, and programmers are not artists, despite any claims to the contrary.

Management has come to believe the first and most important misconception: that it is impossible to ship software devoid of errors in a cost-effective way.

What Makes Software Buggy?

An imperative program describes how a problem can be solved by a computer.

The Von Neumann Architecture of Imperative Programming



Assembly Language

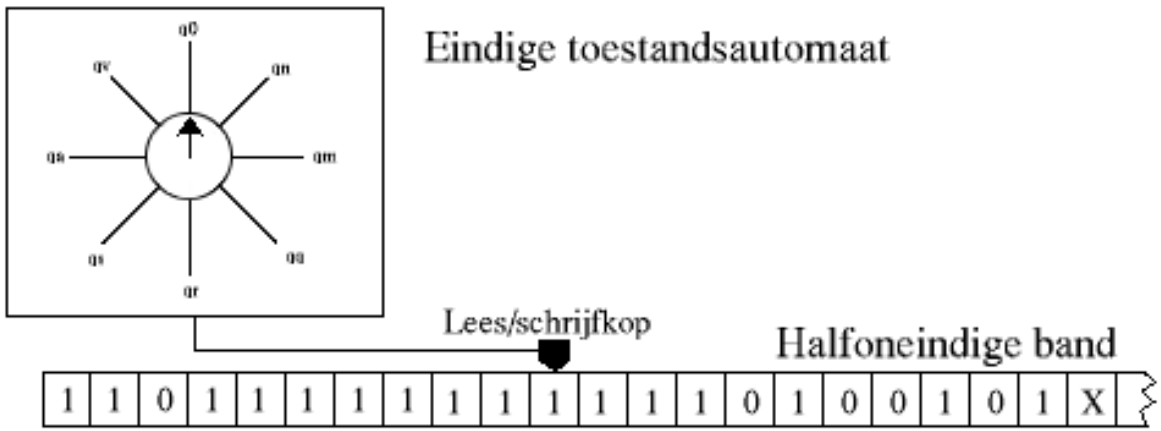
```
; Example of IBM PC assembly language
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

FIGURE 17. Assembly language

One of the Founding Fathers of Computer Science: Alan Turing

The Turing Machine



Edsger Dijkstra Introduced Structured Programming

Debugging only shows that a program is incorrect.

What The Hack Are You Doing?

What does the following program compute,
assuming that the initial value of x is greater than or equal to 0?

$y := 0; u := 0; v := 1;$
while $u + v \leq x$
do $y := y + 1;$
$u := u + v;$
$v := v + 2$
od

Debugging: Let it Flow

x	y	u	v
13	0	0	1
13	1	1	3
13	2	4	5
13	3	9	7
$\vdots$	$\vdots$	$\vdots$	$\vdots$

What's the relation between the values of x, y, u and v?

Robert Floyd Introduced Assertions For Program Specification in the Seventies

$$y^2 \leq x < (y+1)^2$$

Sir. Tony Hoare Developed a First Programming Logic

Design by Contract

Caller = Client and Callee = Supplier
in
Method calls in object-oriented programs

Designer must formally specify for each method:

- What does it expect? (precondition)
- What does it guarantee?(postcondition)
- What does it maintain? (invariant)

Main idea:

Formal specification of contracts by assertions, i.e. logical formulas

Design by Contract in Practice

- Object-oriented programming language Eiffel introduced by the company Eiffel Software.
- The Java Modelling Language JML supports run-time assertion checking.
- Spec# is a formal language for API contracts developed and used by Microsoft.

Correctness Formulas

$$\{p\}S\{q\}$$

where

- S is a (programming) statement
- p and q are assertions
- p is the precondition
- q is the postcondition

Informal Meaning

Every terminating computation of S
in a state which satisfies the precondition p
results in a final state which satisfies the postcondition q

Specifying Correctness of Assignments

- $\{\textbf{true}\}x := 0\{x = 0\}$
  (Java syntax: $\{\textbf{true}\}x = 0\{x == 0\}$)
- $\{\textbf{true}\}x := y + 1\{x = y + 1\}$
- $\{y = 0\}x := y + 1\{x = 1 \wedge y = 0\}$

Some Exercises

- Does in general $\{\mathbf{true}\}x := e\{x = e\}$ hold, e any side-effect free expression?
- For which precondition p does $\{p\}x := x + 1\{x = y\}$ hold?
- For which precondition p does $\{p\}x := x + 1\{a[x] = 0\}$ hold, where a is an array int[]?
- For which precondition p does $\{p\}x := x + 1\{x = y + 1\}$ hold?
- For which precondition p does $\{p\}a[i] := 0\{a[j] = 0\}$ hold, where a is an array int[]?
- For which precondition p does $\{p\}x := y.val\{x = y.val\}$ hold?
- For which precondition p does $\{p\}x := y \ div \ z\{x = y \ div \ z\}$ hold?
- For which postcondition q does $\{x = \mathbf{null}\}y := x.val\{q\}$ hold?
- For which statements S does $\{\mathbf{true}\}S\{\mathbf{false}\}$ hold?

Specifying Correctness of The Sequential Composition of Statements

- $\{x = y\}x := x + 1; y := y + 1\{x = y\}$
  Question: what holds in between?
- $\{x = q \cdot y + r \wedge r \geq y\}r := r - y; q := q + 1\{x = q \cdot y + r\}$
  Question: what holds in between?

Specifying Correctness of Conditional Statements

$\{\mathbf{true}\}$

- **if** $x > y$ **then** $m := x$ **else** $m := y$ **fi**

$\{(m = x \wedge x > y) \vee (m = y \wedge x \leq y)\}$

$\{\mathbf{true}\}$

- **if** $y \neq 0$ **then** $z := x \ div \ y$ **else** *skip* **fi**

$\{y \neq 0 \rightarrow z = x \ div \ y\}$

Specifying Correctness of While Statements

- $$\{\textbf{true}\}\textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}\{a[x] = 0\}$$
- $$\{a[n+1] = 0 \wedge \forall i \in [x : n] : \ a[i] \neq 0\}$$
  $$\textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}$$
  $$\{x = n + 1\}$$
- $$\{\forall n : a[n] = b[n]\}\textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}\{\forall n : a[n] = b[n]\}$$

Validating Correctness Formulas

Two methods to validate $\{p\}S\{q\}$:

- Testing: select input values for the program variables which satisfy p, run the S and check upon termination q.
- Verification: Axioms and proof rules.

Syntax of While Programs

S	$::=$	$skip$	skip statement
	$\mid$	$u := t$	assignment
	$\mid$	$S_1; S_2$	sequential composition
	$\mid$	**if** B **then** S_1 **else** S_2 **fi**	choice
	$\mid$	**while** B **do** S_1 **od**	iteration

Example:

$$x := a[i]; a[i] := a[j]; a[j] := x$$

Types

Basic types:

- **integer**,
- **Boolean**.

Higher types:

- $T_1 \times \ldots \times T_n \to T$,
  where
  - $T_1, \ldots, T_n, T$ are basic types.
  - $T_1, \ldots, T_n$ are *argument* types and T is the *value* type.

Variables

We distinguish two sorts of variables:

- *simple* variables (basic type),
- *array* variables or just arrays (higher type).

We denote the set of all simple and array variables by Var.

Constants

- constants of basic type,
- constants of higher type.

Examples:

- $+, -, \cdot, \mathit{min}, \mathit{max}, \mathit{div}, \mathit{mod}$ of type
  $\mathbf{integer} \times \mathbf{integer} \rightarrow \mathbf{integer}$,
- $=, <$ of type
  $\mathbf{integer} \times \mathbf{integer} \rightarrow \mathbf{Boolean}$,
- $\neg$ of type
  $\mathbf{Boolean} \rightarrow \mathbf{Boolean}$,
- $=, \vee, \wedge, \rightarrow, \leftrightarrow$ of type
  $\mathbf{Boolean} \times \mathbf{Boolean} \rightarrow \mathbf{Boolean}$.

Expressions

Expressions are defined by induction as follows:

- a simple variable of type T is an expression of type T,
- a constant of a basic type T is an expression of type T,
- if $s_1, \ldots, s_n$ are expressions of type $T_1, \ldots, T_n$, respectively, and op is a constant of type $T_1 \times \ldots \times T_n \to T$, then $op(s_1, \ldots, s_n)$ is an expression of type T,
- if $s_1, \ldots, s_n$ are expressions of type $T_1, \ldots, T_n$, respectively, and a is an array of type $T_1 \times \ldots \times T_n \to T$, then $a[s_1, \ldots, s_n]$ is an expression of type T,
- if B is a Boolean expression and s_1 and s_2 are expressions of type T, then **if** B **then** s_1 **else** s_2 **fi** is an expression of type T.

Infix notation

$$(s_1 \ op \ s_2)$$

Syntax of Assertions

$$
\begin{array}{lcll}
p & ::= & B & \text{Boolean expression} \\
 & | & (p \wedge q) & \text{conjunction} \\
 & | & \neg p & \text{negation} \\
 & \vdots & & \\
 & | & \exists x : p & \text{quantification}
\end{array}
$$

Example:

$$\forall n : a[n] \leq a[n+1]$$

Axioms for SKIP and Assignment

AXIOM 1: SKIP

$$\{p\}\ \mathit{skip}\ \{p\}$$

AXIOM 2: ASSIGNMENT

$$\{p[u := t]\}\ u := t\ \{p\}$$

Example

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

Substitution Subscripted Variables

$\{(a[y] = 1)[a[x] := 0]\}\ a[x] := 0\ \{a[y] = 1\}$
Example:

$$
\begin{array}{ll}
(a[y] = 1)[a[x] := 0] & \equiv \\
(a[y])[a[x] := 0] = (1[a[x] := 0]) & \equiv \\
\textbf{if } y[a[x] := 0] = x \textbf{ then } 0 \textbf{ else } a[y[a[x] := 0]] \textbf{ fi} = 1 & \equiv \\
\textbf{if } y = x \textbf{ then } 0 \textbf{ else } a[y] \textbf{ fi} = 1 &
\end{array}
$$

We derive

$$\{\textbf{if } y = x \textbf{ then } 0 \textbf{ else } a[y] \textbf{ fi} = 1\}\ a[x] := 0\ \{a[y] = 1\}$$

Consequence Rule

RULE 6: CONSEQUENCE

$$\frac{p \to p_1, \{p_1\}\ S\ \{q_1\}, q_1 \to q}{\{p\}\ S\ \{q\}}$$

Example: Let $p \equiv$ **if** $y = x$ **then** 0 **else** $a[y]$ **fi** $= 1$.

$$\frac{(y \neq x \wedge a[y] = 1) \to p, \{p\}\ a[x] := 0\ \{a[y] = 1\}, a[y] = 1 \to a[y] = 1}{\{y \neq x \wedge a[y] = 1\}\ a[x] := 0\ \{a[y] = 1\}}$$

Sequential Composition

RULE 3: COMPOSITION

$$\frac{\{p\}\ S_1\ \{r\}, \{r\}\ S_2\ \{q\}}{\{p\}\ S_1;\ S_2\ \{q\}}$$

Example

$$\frac{\{x+1=y+1\}\ x := x+1\ \{x=y+1\}, \{x=y+1\}\ y := y+1\ \{x=y\}}{\{x+1=y+1\}\ x := x+1;\ y := y+1\ \{x=y\}}$$

Application consequence rule:

$$\{x=y\}\ x := x+1;\ y := y+1\ \{x=y\}$$

since

$$x = y \rightarrow x+1 = y+1$$

Conditional

RULE 4: CONDITIONAL

$$\frac{\{p \wedge B\}\ S_1\ \{q\}, \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \{q\}}$$

Example

$$\frac{\{x \leq y\}\ z := y\ \{z = max(x,y)\}, \{\neg(x \leq y)\}\ z := x\ \{z = max(x,y)\}}{\{\textbf{true}\}\ \textbf{if}\ x \leq y\ \textbf{then}\ z := y\ \textbf{else}\ z := x\ \textbf{fi}\ \{z = max(x,y)\}}$$

Note: in the above premises we have abbreviated $\textbf{true} \wedge x \leq y$ and $\textbf{true} \wedge \neg(x \leq y)$.

Loop

RULE 5: LOOP

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

Example

$$\frac{\{x \leq y \wedge x < y\}\ x := x + 1\ \{x \leq y\}}{\{x \leq y\}\ \textbf{while}\ x < y\ \textbf{do}\ x := x + 1\ \textbf{od}\ \{x \leq y \wedge \neg(x < y)\}}$$

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 2)
2. $\{\forall n \leq i < x : a[i] \neq 0\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 2)
2. $\{\forall n \leq i < x : a[i] \neq 0\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 5: 3)
3. $\{\forall n \leq i < x : a[i] \neq 0 \wedge a[x] \neq 0\}x := x + 1\{\forall i : n \leq i < x :$
   $a[i] \neq 0\}$

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 2)
2. $\{\forall n \leq i < x : a[i] \neq 0\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 5: 3)
3. $\{\forall n \leq i < x : a[i] \neq 0 \wedge a[x] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$

4. $\{\forall n \leq i < x + 1 : a[i] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$
   (AXIOM 2)

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 2)
2. $\{\forall n \leq i < x : a[i] \neq 0\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 5: 3)
3. $\{\forall n \leq i < x : a[i] \neq 0 \wedge a[x] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 4)
4. $\{\forall n \leq i < x + 1 : a[i] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$
   (AXIOM 2)

Correctness Zero Search

Let S denote **while** $a[x] \neq 0$ **do** $x := x + 1$ **od**, to prove

$$\{x = n\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$$

Proof:

1. $\{x = n\}S\{a[x] = 0 \wedge \forall n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 2)
2. $\{\forall n \leq i < x : a[i] \neq 0\}S\{a[x] = 0 \wedge \forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 5: 3)
3. $\{\forall n \leq i < x : a[i] \neq 0 \wedge a[x] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$
   (RULE 6: 4)
4. $\{\forall n \leq i < x + 1 : a[i] \neq 0\}x := x + 1\{\forall i : n \leq i < x : a[i] \neq 0\}$
   (AXIOM 2)

- $(a[x] \neq 0 \wedge \forall n \leq i < x : a[i] \neq 0) \rightarrow \forall n \leq i < x + 1 : a[i] \neq 0$
- $x = n \rightarrow \forall n \leq i < x : a[i] \neq 0$

Correctness DIV

To prove

$$\{x \geq 0 \wedge y \geq 0\}\ DIV\ \{q \cdot y + r = x \wedge 0 \leq r < y\}$$

where DIV denotes

$$q := 0; r := x; \textbf{while } r \geq y \textbf{ do } r := r - y;\ q := q + 1 \textbf{ od}$$

Loop Invariant

$$I \equiv q \cdot y + r = x \wedge r \geq 0$$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$
   (RULE 5: 2)

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$
2. $r := r - y; q := q + 1$

   $\{q \cdot y + r = x \wedge r \geq 0\}$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$
   (RULE 5: 2)

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

2. $r := r - y; q := q + 1$

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

3. $q := q + 1$ (AXIOM 2)

   $\{q \cdot y + r = x \wedge r \geq 0\}$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$
   (RULE 5: 2)

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

2. $r := r - y; q := q + 1$

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

3. $q := q + 1$ (AXIOM 2)

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + (r - y) = x \wedge (r - y) \geq 0\}$

4. $r := r - y$ (AXIOM 2)

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$
   (RULE 5: 2)

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

2. $r := r - y; q := q + 1$

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

3. $q := q + 1$ (AXIOM 2)

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + (r - y) = x \wedge (r - y) \geq 0\}$

4. $r := r - y$ (AXIOM 2)

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

5. $r := r - y$ (RULE 6: 4)

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

Invariance

1. $\{I\}$**while** $r \geq y$ **do** $r := r - y;\ q := q + 1$ **od**$\{I \wedge \neg(r \geq y)\}$
   (RULE 5: 2)

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

2. $r := r - y; q := q + 1$ (RULE 3: 3,5)

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

3. $q := q + 1$ (AXIOM 2)

   $\{q \cdot y + r = x \wedge r \geq 0\}$

   $\{(q + 1) \cdot y + (r - y) = x \wedge (r - y) \geq 0\}$

4. $r := r - y$ (AXIOM 2)

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

   $\{q \cdot y + r = x \wedge r \geq 0 \wedge r \geq y\}$

5. $r := r - y$ (RULE 6: 4)

   $\{(q + 1) \cdot y + r = x \wedge r \geq 0\}$

Initialisation

6. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$

Initialisation

6. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$

7. $\{q \cdot y + x = x \wedge x \geq 0\} r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (AXIOM 2)

Initialisation

6. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$

7. $\{q \cdot y + x = x \wedge x \geq 0\} r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (AXIOM 2)
8. $\{0 \cdot y + x = x \wedge x \geq 0\} q := 0 \{q \cdot y + x = x \wedge x \geq 0\}$
   (AXIOM 2)

Initialisation

6. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$

7. $\{q \cdot y + x = x \wedge x \geq 0\} r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (AXIOM 2)
8. $\{0 \cdot y + x = x \wedge x \geq 0\} q := 0 \{q \cdot y + x = x \wedge x \geq 0\}$
   (AXIOM 2)
9. $\{x \geq 0 \wedge y \geq 0\} q := 0 \{q \cdot y + x = x \wedge x \geq 0\}$
   (RULE 6: 8, $(x \geq 0 \wedge y \geq 0) \rightarrow 0 \cdot y + x = x \wedge x \geq 0$)

Initialisation

6. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (RULE 3: 7,9)
7. $\{q \cdot y + x = x \wedge x \geq 0\} r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (AXIOM 2)
8. $\{0 \cdot y + x = x \wedge x \geq 0\} q := 0 \{q \cdot y + x = x \wedge x \geq 0\}$
   (AXIOM 2)
9. $\{x \geq 0 \wedge y \geq 0\} q := 0 \{q \cdot y + x = x \wedge x \geq 0\}$
   (RULE 6: 8, $(x \geq 0 \wedge y \geq 0) \rightarrow 0 \cdot y + x = x \wedge x \geq 0$)

Conclusion

10. $\{x \geq 0 \wedge y \geq 0\}\ DIV\ \{q \cdot y + r = x \wedge 0 \leq r < y\}$

Conclusion

10. $\{x \geq 0 \wedge y \geq 0\}\ DIV\ \{q \cdot y + r = x \wedge 0 \leq r < y\}$
    (RULE 6: 11)
11. $\{x \geq 0 \wedge y \geq 0\} DIV \{q \cdot y + r = x \wedge r \geq 0 \wedge \neg(r \geq y)\}$
    (RULE 3: 1,6)

Correctness Summation Program

$$
\begin{aligned}
SUM \equiv\ & k := 0;\ x := 0; \\
& \textbf{while } k \neq N \textbf{ do} \\
& \quad x := x + a[k]; \\
& \quad k := k + 1 \\
& \textbf{od}.
\end{aligned}
$$

To prove

$$\{N \geq 0\} SUM \{x = \Sigma_{i=0}^{N-1}\ a[i]\}$$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$k := 0;\ x := 0;$

while $k \neq N$ **do**

$x := x + a[k];$

$k := k + 1$

od.

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$k := 0;\ x := 0;$
$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$
while $k \neq N$ **do**

$x := x + a[k];$

$k := k + 1$

od.

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$\{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\}$

$k := 0;\ x := 0;$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

while $k \neq N$ **do**

$x := x + a[k];$

$k := k + 1$

od.

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$\{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\}$

$k := 0;\ x := 0;$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

while $k \neq N$ **do**

$x := x + a[k];$

$k := k + 1$

od.

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\}$

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$\{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\}$

$k := 0;\ x := 0;$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

while $k \neq N$ **do**

$x := x + a[k];$

$k := k + 1$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

od.

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\}$

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$\{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\}$

$k := 0;\ x := 0;$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

while $k \neq N$ **do**

$x := x + a[k];$

$\{0 \leq (k+1) \leq N \wedge x = \Sigma_{i=0}^{(k+1)-1}\ a[i]\}$

$k := k + 1$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

od.

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\}$

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$SUM \equiv \{N \geq 0\}$

$\{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\}$

$k := 0;\ x := 0;$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

while $k \neq N$ **do**

$\{0 \leq (k+1) \leq N \wedge x + a[k] = \Sigma_{i=0}^{(k+1)-1}\ a[i]\}$

$x := x + a[k];$

$\{0 \leq (k+1) \leq N \wedge x = \Sigma_{i=0}^{(k+1)-1}\ a[i]\}$

$k := k + 1$

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\}$

od.

$\{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\}$

$\{x = \Sigma_{i=0}^{N-1}\ a[i]\}$

Proof Outline for the Summation Program

$$
\begin{aligned}
SUM \equiv\ & \{N \geq 0\} \\
& \{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\} \\
& k := 0;\ x := 0; \\
& \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \textbf{while } k \neq N \textbf{ do} \\
& \quad \{0 \leq k < N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \quad \{0 \leq (k+1) \leq N \wedge x + a[k] = \Sigma_{i=0}^{(k+1)-1}\ a[i]\} \\
& \quad x := x + a[k]; \\
& \quad \{0 \leq (k+1) \leq N \wedge x = \Sigma_{i=0}^{(k+1)-1}\ a[i]\} \\
& \quad k := k + 1 \\
& \quad \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \textbf{od}. \\
& \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\} \\
& \{x = \Sigma_{i=0}^{N-1}\ a[i]\}
\end{aligned}
$$

Proof Outline for the Summation Program

$$
\begin{aligned}
SUM \equiv\ & \{N \geq 0\} \\
& \{0 \leq 0 \leq N \wedge 0 = \Sigma_{i=0}^{-1}\ a[i]\} \\
& k := 0;\ x := 0; \\
& \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \textbf{while } k \neq N \textbf{ do} \\
& \{0 \leq k \leq N \wedge k \neq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \quad \{0 \leq k < N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \quad \{0 \leq (k+1) \leq N \wedge x + a[k] = \Sigma_{i=0}^{(k+1)-1}\ a[i]\} \\
& \quad x := x + a[k]; \\
& \quad \{0 \leq (k+1) \leq N \wedge x = \Sigma_{i=0}^{(k+1)-1}\ a[i]\} \\
& \quad k := k + 1 \\
& \quad \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i]\} \\
& \textbf{od}. \\
& \{0 \leq k \leq N \wedge x = \Sigma_{i=0}^{k-1}\ a[i] \wedge \neg(k \neq N)\} \\
& \{x = \Sigma_{i=0}^{N-1}\ a[i]\}
\end{aligned}
$$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$

while $i < k$ **do**

$a[i] := b[i]$

$i := i + 1$

od

$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
while $i < k$ **do**

$$a[i] := b[i]$$

$$i := i + 1$$

od

$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
while $i < k$ **do**

$$a[i] := b[i]$$

$$i := i + 1$$

od
$\{\neg(i < k) \wedge \forall n : 1 \leq n < i : a[n] = b[n]\}$
$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
while $i < k$ **do**

$a[i] := b[i]$

$i := i + 1$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$

od
$\{\neg(i < k) \wedge \forall n : 1 \leq n < i : a[n] = b[n]\}$
$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
while $i < k$ **do**

$a[i] := b[i]$
$\{\forall n : 1 \leq n < i + 1 : a[n] = b[n]\}$
$i := i + 1$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$

od
$\{\neg(i < k) \wedge \forall n : 1 \leq n < i : a[n] = b[n]\}$
$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Proof-outline Array Copy

To prove $\{i = 1\}$**while** $i < k$ **do** $a[i] := b[i]; i := i + 1$ **od**$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$
we introduce the following proof-outline:

$\{i = 1\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
while $i < k$ **do** $\{\forall n : 1 \leq n < i : a[n] = b[n] \wedge i < k\}$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
$a[i] := b[i]$
$\{\forall n : 1 \leq n < i + 1 : a[n] = b[n]\}$
$i := i + 1$
$\{\forall n : 1 \leq n < i : a[n] = b[n]\}$
od
$\{\neg(i < k) \wedge \forall n : 1 \leq n < i : a[n] = b[n]\}$
$\{\forall n : 1 \leq n < k : a[n] = b[n]\}$

Justification

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Array assignment To this end we compute

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Array assignment To this end we compute

$$(\forall n : 1 \leq n < i + 1 : a[n] = b[n])[a[i] := b[i]]$$

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Array assignment

To this end we compute

$$(\forall n : 1 \leq n < i + 1 : a[n] = b[n])[a[i] := b[i]]$$
$$\equiv$$
$$\forall n : 1 \leq n < i + 1 : a[n][a[i] := b[i]] = b[n][a[i] := b[i]]$$

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Array assignment

To this end we compute

$$(\forall n : 1 \leq n < i + 1 : a[n] = b[n])[a[i] := b[i]]$$
$$\equiv$$
$$\forall n : 1 \leq n < i + 1 : a[n][a[i] := b[i]] = b[n][a[i] := b[i]]$$
$$\equiv$$
$$\forall n : 1 \leq n < i + 1 : \textbf{if } n = i \textbf{ then } b[i] \textbf{ else } a[n] \textbf{ fi} = b[n]$$

Justification

Initialization

$$i = 1 \rightarrow \forall n : 1 \leq n < i : a[n] = b[n]$$

Termination

$$\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k)$$
$$\rightarrow \forall n : 1 \leq n < k : a[n] = b[n]$$

Array assignment To this end we compute

$$(\forall n : 1 \leq n < i + 1 : a[n] = b[n])[a[i] := b[i]]$$
$$\equiv$$
$$\forall n : 1 \leq n < i + 1 : a[n][a[i] := b[i]] = b[n][a[i] := b[i]]$$
$$\equiv$$
$$\forall n : 1 \leq n < i + 1 : \textbf{if } n = i \textbf{ then } b[i] \textbf{ else } a[n] \textbf{ fi} = b[n]$$

and observe that the resulting formula is logically equivalent to

$$\forall n : 1 \leq n < i : a[n] = b[n]$$

Case Study: Minimum-Sum Section Problem

Let $s_{i,j}$ denote the sum of section $a[i:j]$:

$$s_{i,j} = \Sigma_{k=i}^{j} a[k].$$

Case Study: Minimum-Sum Section Problem

Let $s_{i,j}$ denote the sum of section $a[i : j]$:

$$s_{i,j} = \Sigma_{k=i}^{j}\ a[k].$$

Design *MINSUM* such that

$$\{N > 0\} MINSUM \{sum = min\ \{s_{i,j} \mid 0 \leq i \leq j < N\}\}$$

Case Study: Minimum-Sum Section Problem

Let $s_{i,j}$ denote the sum of section $a[i:j]$:

$$s_{i,j} = \Sigma_{k=i}^{j}\ a[k].$$

Design *MINSUM* such that

$$\{N > 0\} MINSUM \{sum = min\ \{s_{i,j} \mid 0 \leq i \leq j < N\}\}$$

For example, the minimum-sum section of

$$a[0:4] = (5, -3, 2, -4, 1)$$

is

$$a[1:3] = (-3, 2, -4)$$

and its sum is -5.

Invariant

Let

$$s_k = min\ \{s_{i,j} \mid 0 \leq i \leq j < k\}.$$

Invariant

Let

$$s_k = min\ \{s_{i,j} \mid 0 \leq i \leq j < k\}.$$

Note that

$$min\ \{s_{i,j} \mid 0 \leq i \leq j < N\} = s_N$$

Invariant

Let

$$s_k = min\ \{s_{i,j} \mid 0 \leq i \leq j < k\}.$$

Note that

$$min\ \{s_{i,j} \mid 0 \leq i \leq j < N\} = s_N$$

We *construct* a loop with invariant

$$1 \leq k \leq N \wedge sum = s_k$$

While Body

s_{k+1}

While Body

$$
\begin{array}{ll}
 & s_{k+1} \\
= & \quad \{\text{definition of } s_{k+1}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k+1\})
\end{array}
$$

While Body

$$
\begin{array}{ll}
 & s_{k+1} \\
= & \quad \{\text{definition of } s_{k+1}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k+1\}) \\
= & \quad \{\text{definition of } s_{i,j}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k\} \cup \{s_{i,k} \mid 0 \leq i < k+1\})
\end{array}
$$

While Body

$$
\begin{array}{ll}
 & s_{k+1} \\
= & \quad \{\text{definition of } s_{k+1}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k+1\}) \\
= & \quad \{\text{definition of } s_{i,j}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k\} \cup \{s_{i,k} \mid 0 \leq i < k+1\}) \\
= & \quad \{\text{associativity of } min\} \\
 & min(min(\{s_{i,j} \mid 0 \leq i \leq j < k\}), min(\{s_{i,k} \mid 0 \leq i < k+1\}))
\end{array}
$$

While Body

$$
\begin{array}{ll}
 & s_{k+1} \\
= & \quad \{\text{definition of } s_{k+1}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k+1\}) \\
= & \quad \{\text{definition of } s_{i,j}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k\} \cup \{s_{i,k} \mid 0 \leq i < k+1\}) \\
= & \quad \{\text{associativity of } min\} \\
 & min(min(\{s_{i,j} \mid 0 \leq i \leq j < k\}), min(\{s_{i,k} \mid 0 \leq i < k+1\})) \\
= & \quad \{\text{definition of } t_{k+1}\} \\
 & min(s_k, t_{k+1})
\end{array}
$$

While Body

$$
\begin{array}{ll}
 & s_{k+1} \\
= & \quad \{\text{definition of } s_{k+1}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k+1\}) \\
= & \quad \{\text{definition of } s_{i,j}\} \\
 & min(\{s_{i,j} \mid 0 \leq i \leq j < k\} \cup \{s_{i,k} \mid 0 \leq i < k+1\}) \\
= & \quad \{\text{associativity of } min\} \\
 & min(min(\{s_{i,j} \mid 0 \leq i \leq j < k\}), min(\{s_{i,k} \mid 0 \leq i < k+1\})) \\
= & \quad \{\text{definition of } t_{k+1}\} \\
 & min(s_k, t_{k+1})
\end{array}
$$

where

$$t_k \equiv min\ \{s_{i,k-1} \mid 0 \leq i < k\}$$

Synthesis

$k := 1;\ \ sum := a[0];$

while $k \neq N$ **do**

$$sum := min(sum, t_{k+1});$$

$$k := k + 1$$

od

Synthesis

$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**

$$sum := min(sum, t_{k+1});$$

$$k := k + 1$$

od

Synthesis

$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**

$$sum := min(sum, t_{k+1});$$

$$k := k + 1$$

od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**

$$sum := min(sum, t_{k+1});$$

$$k := k + 1$$

od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**

$sum := min(sum, t_{k+1});$

$k := k + 1$
$\{1 \leq k \leq N \wedge sum = s_k\}$

od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**

$sum := min(sum, t_{k+1});$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k\}$

od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do**
$\{1 \leq k+1 \leq N \wedge min(sum, t_{k+1}) = s_{k+1}\}$
$sum := min(sum, t_{k+1});$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k\}$
od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do** $\{1 \leq k \leq N \wedge sum = s_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, t_{k+1}) = s_{k+1}\}$
$sum := min(sum, t_{k+1});$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k\}$
od

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do** $\{1 \leq k \leq N \wedge sum = s_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, t_{k+1}) = s_{k+1}\}$
$sum := min(sum, t_{k+1});$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k\}$
od
$\{1 \leq k \leq N \wedge sum = s_k \wedge \neg(k \neq N)\}$

Synthesis

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1\}$
$k := 1;\ sum := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k\}$
while $k \neq N$ **do** $\{1 \leq k \leq N \wedge sum = s_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, t_{k+1}) = s_{k+1}\}$
$sum := min(sum, t_{k+1});$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k\}$
od
$\{1 \leq k \leq N \wedge sum = s_k \wedge \neg(k \neq N)\}$
$\{sum = s_N\}$

Initialization

$$N > 0 \rightarrow (1 \leq k \leq N \wedge sum = s_k)[k, sum := 1, a[0]]$$

Initialization

$$N > 0 \rightarrow (1 \leq k \leq N \wedge sum = s_k)[k, sum := 1, a[0]]$$

Note that

$$(1 \leq k \leq N \wedge sum = s_k)[k, sum := 1, a[0]]$$
$$=$$
$$1 \leq 1 \leq N \wedge a[0] = s_1$$

Boolean Test

$$(1 \leq k \leq N \wedge sum = s_k \wedge k \neq N)$$
$$\rightarrow$$
$$(1 \leq k + 1 \leq N \wedge sum = s_k)$$

Finalization

$$1 \leq k \leq N \wedge sum = s_k \wedge k = N)$$
$$\rightarrow$$
$$sum = s_N$$

Computation of t_{k+1}

t_{k+1}

Computation of t_{k+1}

$$
\begin{array}{ll}
 & t_{k+1} \\
= & \quad \{\text{definition of } t_k\} \\
 & min\ \{s_{i,k} \mid 0 \leq i < k+1\}
\end{array}
$$

Computation of t_{k+1}

$$
\begin{array}{ll}
 & t_{k+1} \\
= & \quad \{\text{definition of } t_k\} \\
 & min\ \{s_{i,k} \mid 0 \leq i < k+1\} \\
= & \quad \{\text{associativity of } min\} \\
 & min(min\ \{s_{i,k} \mid 0 \leq i < k\},\ s_{k,k})
\end{array}
$$

Computation of t_{k+1}

$$
\begin{array}{ll}
 & t_{k+1} \\
= & \quad \{\text{definition of } t_k\} \\
 & min\ \{s_{i,k} \mid 0 \leq i < k+1\} \\
= & \quad \{\text{associativity of } min\} \\
 & min(min\ \{s_{i,k} \mid 0 \leq i < k\},\ s_{k,k}) \\
= & \quad \{s_{i,k} = s_{i,k-1} + a[k]\} \\
 & min(min\ \{s_{i,k-1} + a[k] \mid 0 \leq i < k\},\ a[k])
\end{array}
$$

Computation of t_{k+1}

$$
\begin{array}{ll}
 & t_{k+1} \\
= & \quad \{\text{definition of } t_k\} \\
 & min\ \{s_{i,k} \mid 0 \leq i < k+1\} \\
= & \quad \{\text{associativity of } min\} \\
 & min(min\ \{s_{i,k} \mid 0 \leq i < k\},\ s_{k,k}) \\
= & \quad \{s_{i,k} = s_{i,k-1} + a[k]\} \\
 & min(min\ \{s_{i,k-1} + a[k] \mid 0 \leq i < k\},\ a[k]) \\
= & \quad \{\text{property of } min\} \\
 & min(min\ \{s_{i,k-1} \mid 0 \leq i < k\} \ +\ a[k],\ a[k])
\end{array}
$$

Computation of t_{k+1}

$$
\begin{array}{ll}
 & t_{k+1} \\
= & \quad \{\text{definition of } t_k\} \\
 & min\ \{s_{i,k} \mid 0 \leq i < k+1\} \\
= & \quad \{\text{associativity of } min\} \\
 & min(min\ \{s_{i,k} \mid 0 \leq i < k\},\ s_{k,k}) \\
= & \quad \{s_{i,k} = s_{i,k-1} + a[k]\} \\
 & min(min\ \{s_{i,k-1} + a[k] \mid 0 \leq i < k\},\ a[k]) \\
= & \quad \{\text{property of } min\} \\
 & min(min\ \{s_{i,k-1} \mid 0 \leq i < k\} \ +\ a[k],\ a[k]) \\
= & \quad \{\text{definition of } t_k\} \\
 & min(t_k + a[k], a[k])
\end{array}
$$

Correctness by Construction

$k := 1;\ sum := a[0];\ x := a[0];$

while $k \neq N$
do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$k := k + 1$

od

Correctness by Construction

$k := 1;\ sum := a[0];\ x := a[0];$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

while $k \neq N$

do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$k := k + 1$

od

Correctness by Construction

$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$

$k := 1;\ sum := a[0];\ x := a[0];$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

while $k \neq N$

do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$k := k + 1$

od

Correctness by Construction

$\{N > 0\}$

$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$

$k := 1;\ sum := a[0];\ x := a[0];$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

while $k \neq N$

do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$k := k + 1$

od

Correctness by Construction

$\{N > 0\}$

$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$

$k := 1;\ sum := a[0];\ x := a[0];$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

while $k \neq N$

do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$k := k + 1$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

od

Correctness by Construction

$\{N > 0\}$

$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$

$k := 1;\ sum := a[0];\ x := a[0];$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

while $k \neq N$

do

$x := min(x + a[k], a[k]);$

$sum := min(sum, x);$

$\{1 \leq k + 1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$

$k := k + 1$

$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$

od

Correctness by Construction

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
while $k \neq N$
do

$x := min(x + a[k], a[k]);$
$\{1 \leq k + 1 \leq N \wedge min(sum, x) = s_{k+1} \wedge x = t_{k+1}\}$
$sum := min(sum, x);$
$\{1 \leq k + 1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$
$k := k + 1$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
od

Correctness by Construction

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
while $k \neq N$
do
$\{1 \leq k+1 \leq N \wedge min(sum, min(x + a[k], a[k])) = s_{k+1} \wedge$
$min(x + a[k], a[k]) = t_{k+1}\}$
$x := min(x + a[k], a[k]);$
$\{1 \leq k+1 \leq N \wedge min(sum, x) = s_{k+1} \wedge x = t_{k+1}\}$
$sum := min(sum, x);$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$
$k := k + 1$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
od

Correctness by Construction

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
while $k \neq N$
do $\{1 \leq k+1 \leq N \wedge sum = s_k \wedge x = t_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, min(x + a[k], a[k])) = s_{k+1} \wedge$
$min(x + a[k], a[k]) = t_{k+1}\}$
$x := min(x + a[k], a[k]);$
$\{1 \leq k+1 \leq N \wedge min(sum, x) = s_{k+1} \wedge x = t_{k+1}\}$
$sum := min(sum, x);$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
od

Correctness by Construction

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
while $k \neq N$
do $\{1 \leq k+1 \leq N \wedge sum = s_k \wedge x = t_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, min(x + a[k], a[k])) = s_{k+1} \wedge$
$min(x + a[k], a[k]) = t_{k+1}\}$
$x := min(x + a[k], a[k]);$
$\{1 \leq k+1 \leq N \wedge min(sum, x) = s_{k+1} \wedge x = t_{k+1}\}$
$sum := min(sum, x);$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
od
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k \wedge k = N\}$

Correctness by Construction

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
while $k \neq N$
do $\{1 \leq k+1 \leq N \wedge sum = s_k \wedge x = t_k \wedge k \neq N\}$
$\{1 \leq k+1 \leq N \wedge min(sum, min(x + a[k], a[k])) = s_{k+1} \wedge$
$min(x + a[k], a[k]) = t_{k+1}\}$
$x := min(x + a[k], a[k]);$
$\{1 \leq k+1 \leq N \wedge min(sum, x) = s_{k+1} \wedge x = t_{k+1}\}$
$sum := min(sum, x);$
$\{1 \leq k+1 \leq N \wedge sum = s_{k+1} \wedge x = t_{k+1}\}$
$k := k+1$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
od
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k \wedge k = N\}$
$\{sum = s_N\}$

GCD

To prove

$$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$$
$$\textbf{while } x \neq y \textbf{ do if } x > y \textbf{ then } x := x - y \textbf{ else } y := y - x \textbf{ fi od}$$
$$\{x = y \wedge x = n\}$$

we introduce the following proof-outline (see next slide)

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$

while $x \neq y$

do

if $x > y$

then

$x := x - y$

else

$y := y - x$

fi

od

$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$

while $x \neq y$

do

if $x > y$

then

$x := x - y$

else

$y := y - x$

fi

od

$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$

else

$y := y - x$

fi

od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$

else

$y := y - x$

fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$

else

$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$

else

$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$

fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$

else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$
$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then

$x := x - y$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$
$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then
$\{x - y > 0 \wedge y > 0 \wedge n = ggd(x - y, y\}$
$x := x - y$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$
$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do

if $x > y$
then $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x > y\}$
$\{x - y > 0 \wedge y > 0 \wedge n = ggd(x - y, y\}$
$x := x - y$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$
$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
while $x \neq y$
do
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
if $x > y$
then $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x > y\}$
$\{x - y > 0 \wedge y > 0 \wedge n = ggd(x - y, y\}$
$x := x - y$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$
$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$
$y := y - x$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$
fi
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
od
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$

while $x \neq y$

do $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x \neq y\}$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$

if $x > y$

then $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x > y\}$

$\{x - y > 0 \wedge y > 0 \wedge n = ggd(x - y, y\}$

$x := x - y$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$

else $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x < y\}$

$\{x > 0 \wedge y - x > 0 \wedge n = ggd(x, y - x\}$

$y := y - x$

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y\}$

fi

$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$

od

$\{x = y \wedge n = ggd(x, y)\}$

$\{x = y \wedge x = n\}$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\textbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\textbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\textbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\textbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

$$(a[i] = a[j])[a[i] := a[j]] \qquad \equiv$$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\textbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

$$\begin{array}{ll} (a[i] = a[j])[a[i] := a[j]] & \equiv \\ a[i][a[i] := a[j]] = a[j][a[i] := a[j]] & \equiv \end{array}$$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\mathbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

$$\begin{array}{ll} (a[i] = a[j])[a[i] := a[j]] & \equiv \\ a[i][a[i] := a[j]] = a[j][a[i] := a[j]] & \equiv \\ \mathbf{if}\ i = i\ \mathbf{then}\ a[j]\ \mathbf{else}\ a[i]\ \mathbf{fi} = \mathbf{if}\ j = i\ \mathbf{then}\ a[j]\ \mathbf{else}\ a[j]\ \mathbf{fi} & \leftrightarrow \end{array}$$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\mathbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

$$\begin{array}{ll}
(a[i] = a[j])[a[i] := a[j]] & \equiv \\
a[i][a[i] := a[j]] = a[j][a[i] := a[j]] & \equiv \\
\textbf{if } i = i \textbf{ then } a[j] \textbf{ else } a[i] \textbf{ fi} = \textbf{if } j = i \textbf{ then } a[j] \textbf{ else } a[j] \textbf{ fi} & \leftrightarrow \\
a[j] = a[j] & \leftrightarrow
\end{array}$$

Eerste deeltentamen: Voorbeelden

Bewijs de correctheidsbewering

$$\{\mathbf{true}\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

waar a een array is van type **integer** $\rightarrow$ **integer**.

Uitwerking

Assignment Axiom:

$$\{(a[i] = a[j])[a[i] := a[j]]\}\ a[i] := a[j]\ \{a[i] = a[j]\}$$

We berekenen de preconditie:

$$\begin{array}{ll}
(a[i] = a[j])[a[i] := a[j]] & \equiv \\
a[i][a[i] := a[j]] = a[j][a[i] := a[j]] & \equiv \\
\mathbf{if}\ i = i\ \mathbf{then}\ a[j]\ \mathbf{else}\ a[i]\ \mathbf{fi} = \mathbf{if}\ j = i\ \mathbf{then}\ a[j]\ \mathbf{else}\ a[j]\ \mathbf{fi} & \leftrightarrow \\
a[j] = a[j] & \leftrightarrow \\
\mathbf{true} &
\end{array}$$

Bewijs

$$\{n \geq 0\}$$
$$k := 0; \textbf{while } k \leq n \textbf{ do } a[k] := b[n-k]; k := k+1 \textbf{ od}$$
$$\{\forall i \in [0:n] : a[i] = b[n-i]\}$$

$\{n \geq 0\}$

$k := 0$

while $k \leq n$
do

$a[k] := b[n - k];$

$k := k + 1$

od

$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$

$k := 0$
$\{\forall i \in [0 : k-1] : a[i] = b[n-i] \wedge k \leq n+1\}$
while $k \leq n$
do

$a[k] := b[n-k];$

$k := k+1$

od

$\{\forall i \in [0 : n] : a[i] = b[n-i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do

$a[k] := b[n - k];$

$k := k + 1$

od

$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do

$a[k] := b[n - k];$

$k := k + 1$

od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do

$a[k] := b[n - k]$;

$k := k + 1$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do

$a[k] := b[n - k]$;

$\{\forall i \in [0 : (k + 1) - 1] : a[i] = b[n - i] \wedge k + 1 \leq n + 1\}$
$k := k + 1$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$

od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do

$a[k] := b[n - k]$;
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge a[k] = b[n - k] \wedge k \leq n\}$
$\{\forall i \in [0 : (k + 1) - 1] : a[i] = b[n - i] \wedge k + 1 \leq n + 1\}$
$k := k + 1$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$

od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do
$\{\forall i \in [0 : k - 1] : \textbf{if } i = k \textbf{ then } b[n - k] \textbf{ else } a[i] \textbf{ fi} = b[n - i] \wedge$
$\textbf{if } k = k \textbf{ then } b[n - k] \textbf{ else } a[k] \textbf{ fi} = b[n - k] \wedge k \leq n\}$
$a[k] := b[n - k];$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge a[k] = b[n - k] \wedge k \leq n\}$
$\{\forall i \in [0 : (k + 1) - 1] : a[i] = b[n - i] \wedge k + 1 \leq n + 1\}$
$k := k + 1$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

$\{n \geq 0\}$
$\{\forall i \in [0 : -1] : a[i] = b[n - i] \wedge 0 \leq n + 1\}$
$k := 0$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
while $k \leq n$
do $\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n \wedge k \leq n + 1\}$
$\{\forall i \in [0 : k - 1] : \textbf{if } i = k \textbf{ then } b[n - k] \textbf{ else } a[i] \textbf{ fi} = b[n - i] \wedge$
$\textbf{if } k = k \textbf{ then } b[n - k] \textbf{ else } a[k] \textbf{ fi} = b[n - k] \wedge k \leq n\}$
$a[k] := b[n - k];$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge a[k] = b[n - k] \wedge k \leq n\}$
$\{\forall i \in [0 : (k + 1) - 1] : a[i] = b[n - i] \wedge k + 1 \leq n + 1\}$
$k := k + 1$
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1\}$
od
$\{\forall i \in [0 : k - 1] : a[i] = b[n - i] \wedge k \leq n + 1 \wedge \neg(k \leq n)\}$
$\{\forall i \in [0 : n] : a[i] = b[n - i]\}$

Bewijs

$$\{x \geq 0 \wedge y \geq 0\}$$
$$p := 0; c := 0; \textbf{while } c > 0 \textbf{ do } p := p + x; c := c + 1 \textbf{ od}$$
$$\{p = x \times y\}$$

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

$p := 0; c := y$

while $c > 0$
do

$p := p + x;$

$c := c - 1$

od

$\{p = x \times y\}$

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

$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$p := p + x;$

$c := c - 1$

od

$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$p := p + x;$

$c := c - 1$

od

$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$p := p + x;$

$c := c - 1$

od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$p := p + x;$

$c := c - 1$
$\{p = x \times (y - c) \wedge c \geq 0\}$
od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$p := p + x;$
$\{p = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$c := c - 1$
$\{p = x \times (y - c) \wedge c \geq 0\}$
od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do

$\{p + x = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$p := p + x;$
$\{p = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$c := c - 1$
$\{p = x \times (y - c) \wedge c \geq 0\}$

od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do
$\{p + x = x \times (y - c) + x \wedge c - 1 \geq 0\}$
$\{p + x = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$p := p + x;$
$\{p = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$c := c - 1$
$\{p = x \times (y - c) \wedge c \geq 0\}$
od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

$\{x \geq 0 \wedge y \geq 0\}$
$\{0 = x \times (y - y) \wedge y \geq 0\}$
$p := 0; c := y$
$\{p = x \times (y - c) \wedge c \geq 0\}$
while $c > 0$
do $\{p = x \times (y - c) \wedge c \geq 0 \wedge c > 0\}$
$\{p + x = x \times (y - c) + x \wedge c - 1 \geq 0\}$
$\{p + x = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$p := p + x;$
$\{p = x \times (y - (c - 1)) \wedge c - 1 \geq 0\}$
$c := c - 1$
$\{p = x \times (y - c) \wedge c \geq 0\}$
od
$\{p = x \times (y - c) \wedge c \geq 0 \wedge \neg(c > 0)\}$
$\{p = x \times y\}$

Total Correctness Interpretation

$$\{p\}S\{q\}$$

iff

Every computation of S in a state which satisfies precondition p terminates and does so in a state which satisfies the postcondition q

Examples

- $\{x \geq 0\}$ **while** $x \neq 0$ **do** $x := x - 1$ **od** $\{\textbf{true}\}$

Examples

- $\{x \geq 0\}$ **while** $x \neq 0$ **do** $x := x - 1$ **od** $\{\textbf{true}\}$
  But not

$$\{\textbf{true}\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

Examples

- $\{x \geq 0\}$ **while** $x \neq 0$ **do** $x := x - 1$ **od** $\{\textbf{true}\}$
  But not

$$\{\textbf{true}\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- For which precondition p we have

$$\{p\}\ \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}\ \{\textbf{true}\}$$

Examples

- $\{x \geq 0\}$ **while** $x \neq 0$ **do** $x := x - 1$ **od** $\{\textbf{true}\}$
  But not

$$\{\textbf{true}\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- For which precondition p we have

$$\{p\}\ \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}\ \{\textbf{true}\}$$

  Answer: $\exists n : a[n] = 0 \wedge x \leq n$

Exercise

For which statements S we have

- $\models_{tot} \{\mathbf{true}\} S \{\mathbf{false}\}$
- $\models_{tot} \{\mathbf{true}\} S \{\mathbf{true}\}$
- $\models_{tot} \{\mathbf{false}\} S \{\mathbf{true}\}$

Verification Total Correctness

RULE : LOOP II

$$\frac{\{p \wedge B\}\ S\ \{p\},\quad \{p \wedge B \wedge t = z\}\ S\ \{t < z\},\quad p\ \rightarrow t \geq 0}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg\ B\}}$$

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?
  Answer: take for p the assertion $x \geq 0$ itself.

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?
  Answer: take for p the assertion $x \geq 0$ itself.

Proof obligations

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?
  Answer: take for p the assertion $x \geq 0$ itself.

Proof obligations

1. $\{x \geq 0 \wedge x \neq 0\}\ x := x - 1\ \{x \geq 0\}$

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?
  Answer: take for p the assertion $x \geq 0$ itself.

Proof obligations

1. $\{x \geq 0 \wedge x \neq 0\}\ x := x - 1\ \{x \geq 0\}$
2. $\{x \geq 0 \wedge x \neq 0 \wedge x = z\}\ x := x - 1\ \{x < z\}$

Example

Prove total correctness of

$$\{x \geq 0\}\ \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - 1\ \textbf{od}\ \{\textbf{true}\}$$

- What's the bound t?
  Answer: take for t variable x.
- What's the invariant p such that p implies $x \geq 0$?
  Answer: take for p the assertion $x \geq 0$ itself.

Proof obligations

1. $\{x \geq 0 \wedge x \neq 0\}\ x := x - 1\ \{x \geq 0\}$
2. $\{x \geq 0 \wedge x \neq 0 \wedge x = z\}\ x := x - 1\ \{x < z\}$
3. $x \geq 0 \rightarrow x \geq 0$

Total Correctness Zero Search

To prove total correctness of

$\{\exists n : a[n] = 0 \wedge x \leq n\}$**while** $a[x] \neq 0$ **do** $x := x + 1$ **od**$\{a[x] = 0\}$

- What is the bound function t?
- What is the invariant p?

Solution: Instantiation

1. Proof total correctness of

   $\{a[n] = 0 \wedge x \leq n\}$**while** $a[x] \neq 0$ **do** $x := x + 1$ **od**$\{a[x] = 0\}$

2. ($\exists$-INTRODUCTION RULE:1)

   $\{\exists n : a[n] = 0 \wedge x \leq n\}$**while** $a[x] \neq 0$ **do** $x := x + 1$ **od**$\{a[x] = 0\}$

Zero Search: Bound and Invariant

Define

Zero Search: Bound and Invariant

Define

- bound t by $n - x$

Zero Search: Bound and Invariant

Define

- bound t by $n - x$
- invariant p by $a[n] = 0 \wedge x \leq n$

Zero Search: Bound and Invariant

Define

- bound t by $n - x$
- invariant p by $a[n] = 0 \wedge x \leq n$

Proof obligations:

Zero Search: Bound and Invariant

Define

- bound t by $n - x$
- invariant p by $a[n] = 0 \wedge x \leq n$

Proof obligations:

- $\{a[n] = 0 \wedge x \leq n \wedge a[x] \neq 0\}\ x := x + 1\ \{a[n] = 0 \wedge x \leq n\}$

Zero Search: Bound and Invariant

Define

- bound t by $n - x$
- invariant p by $a[n] = 0 \wedge x \leq n$

Proof obligations:

- $\{a[n] = 0 \wedge x \leq n \wedge a[x] \neq 0\}\ x := x + 1\ \{a[n] = 0 \wedge x \leq n\}$
- $a[n] = 0 \wedge x \leq n \rightarrow n - x \geq 0$

Zero Search: Bound and Invariant

Define

- bound t by $n - x$
- invariant p by $a[n] = 0 \wedge x \leq n$

Proof obligations:

- $\{a[n] = 0 \wedge x \leq n \wedge a[x] \neq 0\}\ x := x + 1\ \{a[n] = 0 \wedge x \leq n\}$
- $a[n] = 0 \wedge x \leq n \rightarrow n - x \geq 0$
- $\{a[n] = 0 \wedge x \leq n \wedge a[x] \neq 0 \wedge n - x = z\}\ x := x + 1\ \{n - x < z\}$

Some Auxiliary Rules

$\exists$-INTRODUCTION RULE

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x : p\}\ S\ \{q\}}$$

where x does not occur in S or in $\mathit{free}(q)$.

Some Auxiliary Rules

$\exists$-INTRODUCTION RULE

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x : p\}\ S\ \{q\}}$$

where x does not occur in S or in $\mathit{free}(q)$.

DISJUNCTION RULE

$$\frac{\{p\}\ S\ \{q\}, \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

Some Auxiliary Rules

$\exists$-INTRODUCTION RULE

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x : p\}\ S\ \{q\}}$$

where x does not occur in S or in $free(q)$.

DISJUNCTION RULE

$$\frac{\{p\}\ S\ \{q\}, \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

CONJUNCTION RULE

$$\frac{\{p_1\}\ S\ \{q_1\}, \{p_2\}\ S\ \{q_2\}}{\{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}}$$

Some Auxiliary Rules

∃-INTRODUCTION RULE

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x : p\}\ S\ \{q\}}$$

where x does not occur in S or in $free(q)$.

DISJUNCTION RULE

$$\frac{\{p\}\ S\ \{q\}, \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

CONJUNCTION RULE

$$\frac{\{p_1\}\ S\ \{q_1\}, \{p_2\}\ S\ \{q_2\}}{\{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}}$$

SUBSTITUTION RULE

$$\frac{\{p\}\ S\ \{q\}}{\{p[\bar{z} := \bar{t}]\}\ S\ \{q[\bar{z} := \bar{t}]\}}$$

where $(\{\bar{z}\} \cap var(S)) \cup (var(\bar{t}) \cap change(S)) = \emptyset$.

Total Correctness: Decomposition

RULE : DECOMPOSITION

$$\frac{\vdash_p \{p\}\ S\ \{q\},\quad \vdash_t \{p\}\ S\ \{\mathbf{true}\}}{\{p\}\ S\ \{q\}}$$

where the provability signs $\vdash_p$ and $\vdash_t$ refer to proof systems for partial and total correctness for the considered program S, respectively.

Example Decomposition Total Correctness

To prove total correctness of

$$\{x \geq 0 \wedge y > 0\}\ DIV\ \{q \cdot y + r = x \wedge 0 \leq r < y\}$$

where DIV denotes

$$q := 0; r := x; \textbf{while } r \geq y \textbf{ do } r := r - y;\ q := q + 1 \textbf{ od}$$

it suffices to prove its partial correctness and total correctness of

$$\{x \geq 0 \wedge y > 0\}\ DIV\ \{\textbf{true}\}$$

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$

while $rem \geq y$ **do**

$rem := rem - y;\ quo := quo + 1$

od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$

while $rem \geq y$ **do**

$rem := rem - y;\ quo := quo + 1$

od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

$rem := rem - y;\ quo := quo + 1$

od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

z:=rem

$rem := rem - y;\ quo := quo + 1$

od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

z:=rem

$rem := rem - y;\ quo := quo + 1$
$\{rem \geq 0 \wedge y > 0 \wedge rem < z\}$
od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

z:=rem

$\{rem - y \geq 0 \wedge y > 0 \wedge rem - y < z\}$
$rem := rem - y;\ quo := quo + 1$
$\{rem \geq 0 \wedge y > 0 \wedge rem < z\}$
od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

z:=rem
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y \wedge rem = z\}$
$\{rem - y \geq 0 \wedge y > 0 \wedge rem - y < z\}$
$rem := rem - y;\ quo := quo + 1$
$\{rem \geq 0 \wedge y > 0 \wedge rem < z\}$
od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**

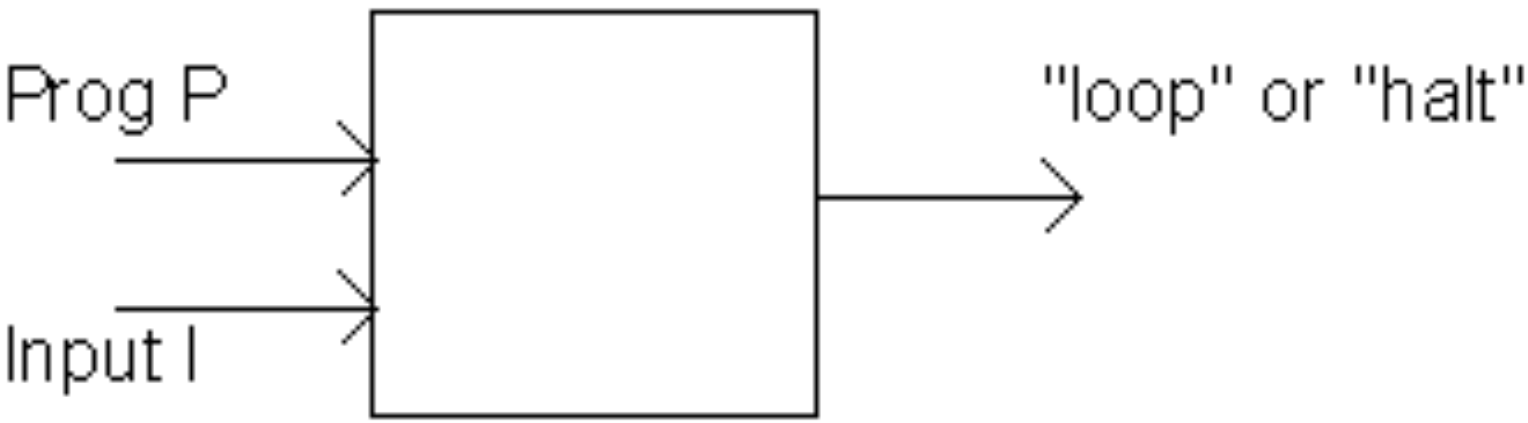
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y \wedge z = z\}$
z:=rem
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y \wedge rem = z\}$
$\{rem - y \geq 0 \wedge y > 0 \wedge rem - y < z\}$
$rem := rem - y;\ quo := quo + 1$
$\{rem \geq 0 \wedge y > 0 \wedge rem < z\}$
od
$\{\mathbf{true}\}$.

Total Correctness Proof Outlines: An Example

$\{x \geq 0 \wedge y > 0\}$
$quo := 0;\ rem := x;$
$\{rem \geq 0 \wedge y > 0\}\ \{\mathbf{bd} : rem\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq 0\}$
while $rem \geq y$ **do**
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y\}$
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y \wedge z = z\}$
z:=rem
$\{rem \geq 0 \wedge y > 0 \wedge rem \geq y \wedge rem = z\}$
$\{rem - y \geq 0 \wedge y > 0 \wedge rem - y < z\}$
$rem := rem - y;\ quo := quo + 1$
$\{rem \geq 0 \wedge y > 0 \wedge rem < z\}$
od
$\{\mathbf{true}\}$.

Halting Problem

Suppose there exists a (Java) program that checks whether a given Java program P terminates for a given input I:



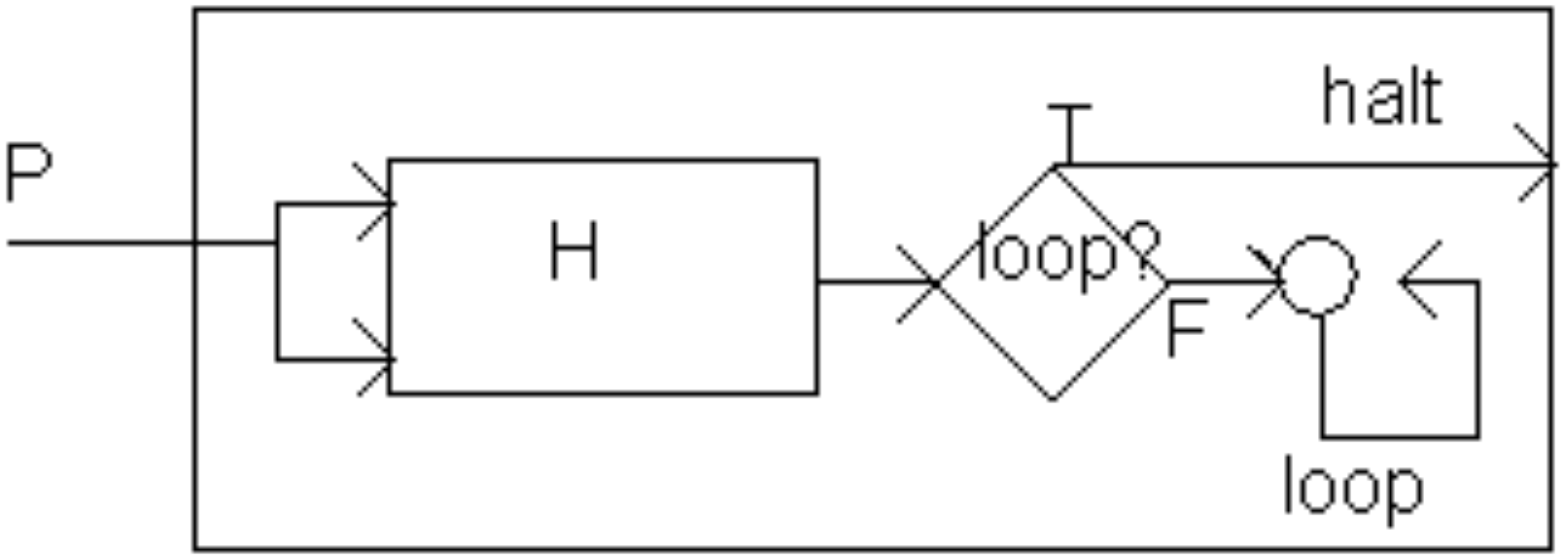
Formally, let H be such that

$$H(P, I) = \begin{cases} 1 & \text{if } P \text{ terminates on } I \\ 0 & \text{otherwise} \end{cases}$$

The Barber of Russell Shaves Only Those Men Who Don't Shave Themselves

The barber of Russell shaves himself
iff
he does not shave himself.

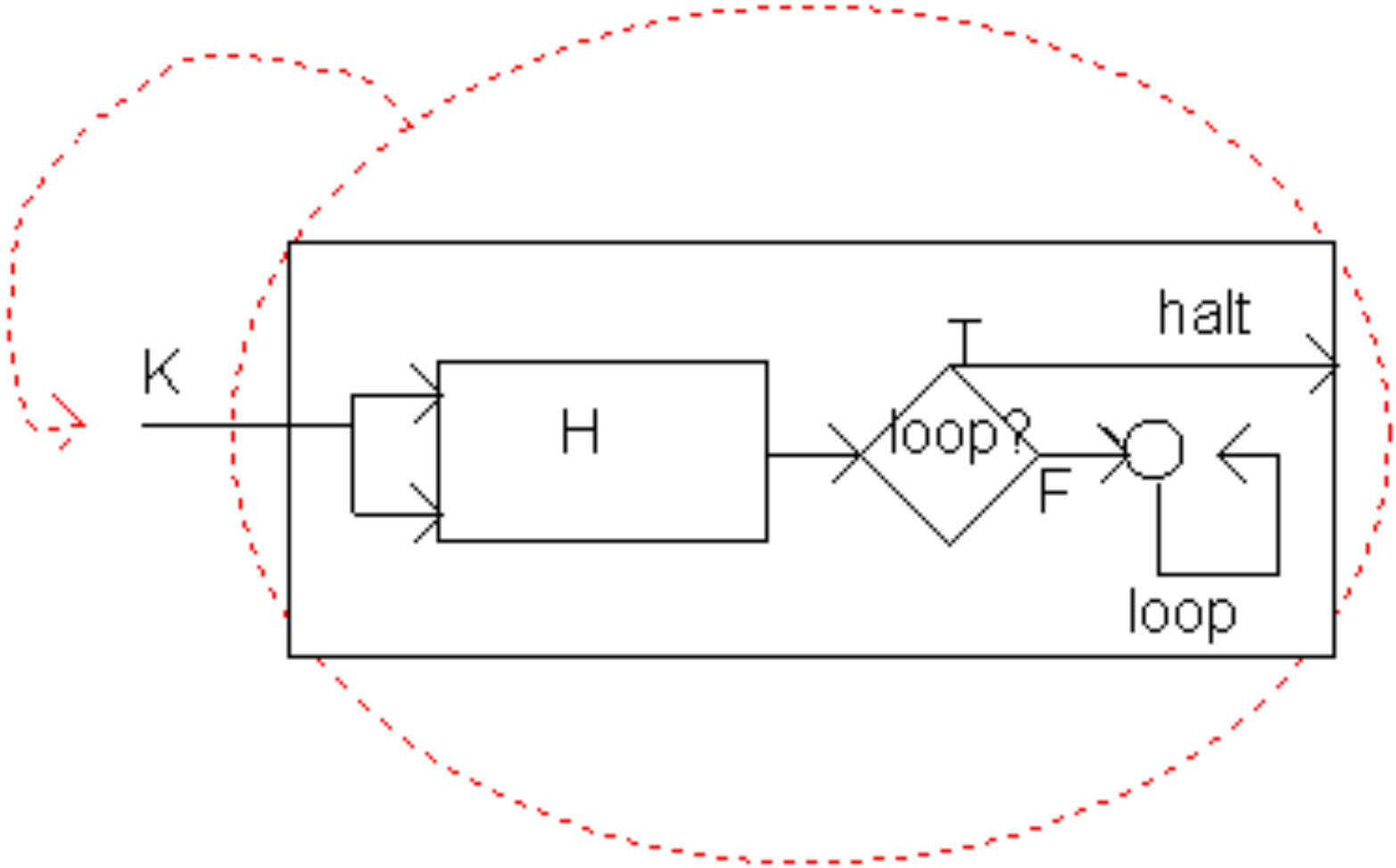
The Barber of Russell Revisited



Formally, let H' be such that

$$H'(P) \equiv H(P, P); \textbf{ if } \mathrm{result} = 1 \textbf{ then while true do } \mathrm{skip} \textbf{ od fi}$$

The Halting Paradox



This program terminates on itself
iff
it does not terminate on itself (as input)

Formally,

H' terminates on H'
iff
$H(H', H') = 0$
iff
H' does not terminate on H'

Recursive Programs

Given a set of *procedure identifiers* with typical elements

$$P, P_1, \ldots,$$

we extend the syntax of **while** programs by the following clause:

$$S ::= P$$

Note: no parameters.

Procedure declarations

$$P :: S$$

Programs

$$D \mid S$$

where D is a set of procedure declarations and S is the *main* statement.

Factorial Program

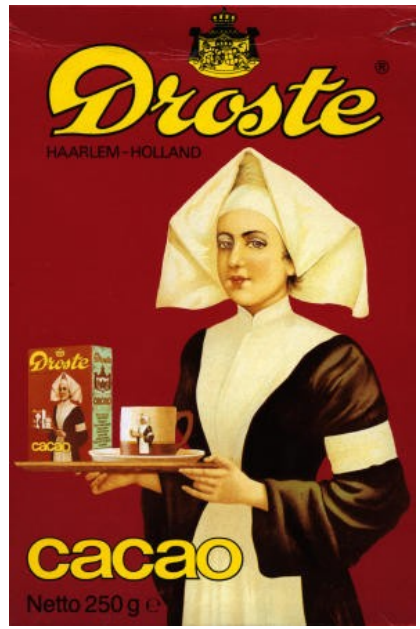
Fac ::
if $x = 0$
then $y := 1$
else $x := x - 1;\ Fac;\ x := x + 1;\ y := y \cdot x$
fi
$|\ \ Fac$

Verification Recursive Programs

The rule

$$\frac{\{p\}\ S\ \{q\}}{\{p\}\ P\ \{q\}}$$

where $P :: S \in D$, gives rise to an infinite regression:



Example: try to prove

$$\{\mathbf{true}\}\ P\ \{\mathbf{true}\}$$

where $P :: P \in D$.

The Simplified Recursion Rule

Assume what you want to prove.

Let $D = P :: S$.

$$\frac{\{p\}\ P\ \{q\} \vdash \{p\}\ S\ \{q\}}{\{p\}\ P\ \{q\}}$$

Correctness Factorial Program

We prove

$$\{x \geq 0\}\ \mathit{Fac}\ \{y = x!\} \vdash \{x \geq 0\}\ S\ \{y = x!\}$$

where S denote the body of Fac.

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$

else

$x := x - 1;$

$Fac;$

$x := x + 1;$

$y := y \cdot x$

fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else

$x := x - 1;$

$Fac;$

$x := x + 1;$

$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else

$x := x - 1;$

$Fac;$

$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else

$x := x - 1;$

$Fac;$

$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else

$x := x - 1;$
$\{x \geq 0\}$
$Fac;$
$\{y = x!\}$
$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else
$\{x - 1 \geq 0\}$
$x := x - 1;$
$\{x \geq 0\}$
$Fac;$
$\{y = x!\}$
$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then

$y := 1$
$\{y = x!\}$
else $\{x \geq 0 \wedge x \neq 0\}$
$\{x - 1 \geq 0\}$
$x := x - 1;$
$\{x \geq 0\}$
$Fac;$
$\{y = x!\}$
$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then
$\{1 = x!\}$
$y := 1$
$\{y = x!\}$
else $\{x \geq 0 \wedge x \neq 0\}$
$\{x - 1 \geq 0\}$
$x := x - 1;$
$\{x \geq 0\}$
$Fac;$
$\{y = x!\}$
$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

$\{x \geq 0\}$
if $x = 0$
then $\{x \geq 0 \wedge x = 0\}$
$\{1 = x!\}$
$y := 1$
$\{y = x!\}$
else $\{x \geq 0 \wedge x \neq 0\}$
$\{x - 1 \geq 0\}$
$x := x - 1;$
$\{x \geq 0\}$
$Fac;$
$\{y = x!\}$
$\{y \cdot (x + 1) = (x + 1)!\}$
$x := x + 1;$
$\{y \cdot x = x!\}$
$y := y \cdot x$
$\{y = x!\}$
fi
$\{y = x!\}$

Proving Invariant Properties

We want to prove the correctness formula

$$\{z = x\}\ \mathit{Fac}\ \{z = x\}$$

Try it!

Does Not Work, Hey?!

We need the Substitution Rule:

$$\frac{\{p\}\ S\ \{q\}}{\{p[z := t]\}\ S\ \{q[z := t]\}}$$

where

- z does not appear in the program
- the variables of t are *read-only*

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then

$y := 1$

else

$x := x - 1;$

$Fac;$

$x := x + 1;$

$y := y \cdot x$

fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$

$Fac;$

$x := x + 1;$

$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$

$Fac;$

$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$

$Fac;$

$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$

$Fac;$
$\{z - 1 = x\}$
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$

$Fac;$
$\{z - 1 = x\}$ $(\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$
$\{z - 1 = x\}$
$Fac;$
$\{z - 1 = x\}$ $(\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$

fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else

$x := x - 1;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$Fac;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else
$\{z - 1 = x - 1\}$
$x := x - 1;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$Fac;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$

else $\{z = x\}$
$\{z - 1 = x - 1\}$
$x := x - 1;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$Fac;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then
$y := 1$
$\{z = x\}$
else $\{z = x\}$
$\{z - 1 = x - 1\}$
$x := x - 1;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$Fac;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Another Proof-Outline for the Factorial Program

$\{z = x\}$
if $x = 0$
then $\{z = x\}$
$y := 1$
$\{z = x\}$
else $\{z = x\}$
$\{z - 1 = x - 1\}$
$x := x - 1;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$Fac;$
$\{z - 1 = x\}$ ($\equiv (z = x)[z := z - 1]$)
$\{z = x + 1\}$
$x := x + 1;$
$\{z = x\}$
$y := y \cdot x$
$\{z = x\}$
fi
$\{z = x\}$

Summing Up Recursively

Let P be declared by

$$\textbf{if } n \neq k-1 \textbf{ then } sum := sum + a[k]; k := k+1; P \textbf{ fi}$$

Prove

$$\{k = 1\}\ sum := 0; P\ \{sum = \Sigma_{i=1}^{n} a[i]\}$$

To prove

$$\{k = 1 \wedge sum = 0\}\ P\ \{sum = \Sigma_{i=1}^{n} a[i]\}$$

Generalized assumption:

$$\{sum = \Sigma_{i=1}^{k-1} a[i]\}\ P\ \{sum = \Sigma_{i=1}^{n} a[i]\}$$

Note that

$$(k = 1 \wedge sum = 0) \rightarrow sum = \Sigma_{i=1}^{k-1} a[i]$$

and ...

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
if $n \neq k - 1$
then

$sum := sum + a[k];$

$k := k + 1;$

P

else

$skip$

fi
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
if $n \neq k - 1$
then

$sum := sum + a[k];$

$k := k + 1;$

P

else

$skip$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
fi
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
if $n \neq k - 1$
then

$sum := sum + a[k];$

$k := k + 1;$

P

else

$\{sum = \Sigma_{i=1}^{n} a[i]\}$
$skip$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

fi
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
if $n \neq k - 1$
then

$sum := sum + a[k];$

$k := k + 1;$

P

else $\{n = k - 1 \wedge sum = \Sigma_{i=1}^{k-1} a[i]\}$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
$skip$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
fi
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
if $n \neq k-1$
then

$sum := sum + a[k];$

$k := k + 1;$
$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$
P
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
else $\{n = k-1 \wedge sum = \Sigma_{i=1}^{k-1} a[i]\}$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
$skip$
$\{sum = \Sigma_{i=1}^{n} a[i]\}$
fi
$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

if $n \neq k - 1$

then

$sum := sum + a[k];$

$\{sum = \Sigma_{i=1}^{k} a[i]\}$

$k := k + 1;$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

P

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

else $\{n = k - 1 \wedge sum = \Sigma_{i=1}^{k-1} a[i]\}$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$skip$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

fi

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

if $n \neq k - 1$

then

$\{sum + a[k] = \Sigma_{i=1}^{k} a[i]\}$

$sum := sum + a[k];$

$\{sum = \Sigma_{i=1}^{k} a[i]\}$

$k := k + 1;$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

P

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

else $\{n = k - 1 \wedge sum = \Sigma_{i=1}^{k-1} a[i]\}$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$skip$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

fi

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

if $n \neq k-1$

then $\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

$\{sum + a[k] = \Sigma_{i=1}^{k} a[i]\}$

$sum := sum + a[k];$

$\{sum = \Sigma_{i=1}^{k} a[i]\}$

$k := k + 1;$

$\{sum = \Sigma_{i=1}^{k-1} a[i]\}$

P

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

else $\{n = k-1 \wedge sum = \Sigma_{i=1}^{k-1} a[i]\}$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

$skip$

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

fi

$\{sum = \Sigma_{i=1}^{n} a[i]\}$

Case Study: Binary Search

$$
\begin{array}{lll}
\mathit{BinSearch} :: & m := (f' + l')\ \mathit{div}\ 2; & \\
& \textbf{if } f' \neq l' & \\
& \textbf{then if } a[m] < v & \\
& \qquad \textbf{then } f' := m + 1;\ \mathit{BinSearch} & \\
& \qquad \textbf{else if } a[m] > v & \\
& \qquad\qquad \textbf{then } \ l' := m;\ \mathit{BinSearch} & \\
& \qquad\qquad \textbf{fi} & \\
& \qquad \textbf{fi} & \\
& \textbf{fi} &
\end{array}
$$

Case Study: Binary Search

$$
\begin{array}{lll}
\mathit{BinSearch} :: & m := (f' + l')\ \mathit{div}\ 2; & \\
& \textbf{if } f' \neq l' & \\
& \textbf{then if } a[m] < v & \\
& \qquad \textbf{then } f' := m + 1;\ \mathit{BinSearch} & \\
& \qquad \textbf{else if } a[m] > v & \\
& \qquad\qquad \textbf{then } \ l' := m;\ \mathit{BinSearch} & \\
& \qquad\qquad \textbf{fi} & \\
& \qquad \textbf{fi} & \\
& \textbf{fi} &
\end{array}
$$

Correctness Binary Search

$$\{ \qquad\qquad \} \qquad\qquad \} \qquad\qquad \}$$

$$\textit{BinSearch}$$

$$\{ \qquad\qquad \} \qquad\qquad \}$$

Correctness Binary Search

$$\{f = f' \land l' = l \land f' \leq l' \land sorted(a)\}$$
$$BinSearch$$
$$\{ \qquad \}$$

Correctness Binary Search

$$\{f = f' \wedge l' = l \wedge f' \leq l' \wedge \mathit{sorted}(a)\}$$
$$\mathit{BinSearch}$$
$$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$$

Correctness Binary Search

$$\{f = f' \wedge l' = l \wedge f' \leq l' \wedge \mathit{sorted}(a)\}$$
$$\mathit{BinSearch}$$
$$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$$

where

- $\mathit{sorted}(a) \equiv \forall n : a[n] \leq a[n+1]$
- $v \in a[f : l] \equiv \exists n \in [f : l] : v = a[n]$

Does Not Fit!

$\{f = f' \wedge l' = l \wedge f' \leq l' \wedge sorted(a)\}$
$m := (f' + l')\ div\ 2;$
if $f' \neq l'$
then if $a[m] < v$
then $f' := m + 1;$
$\{f = f' \wedge l' = l \wedge f' \leq l' \wedge sorted(a)\}$
$BinSearch$
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$
else if $a[m] > v$
then $l' := m;$
$\{f = f' \wedge l' = l \wedge f' \leq l' \wedge sorted(a)\}$
$BinSearch$
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$
fi
fi
fi
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$

So? Let's Weaken The Precondition.

$\{f \leq f' \leq l' \leq l \wedge \mathit{sorted}(a)\}$
$m := (f' + l')\ \mathit{div}\ 2;$
if $f' \neq l'$
then if $a[m] < v$
then $f' := m + 1;$
$\{f \leq f' \leq l' \leq l \wedge \mathit{sorted}(a)\}$
$\mathit{BinSearch}$
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$
else if $a[m] > v$
then $l' := m;$
$\{f \leq f' \leq l' \leq l \wedge \mathit{sorted}(a)\}$
$\mathit{BinSearch}$
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$
fi
fi
fi
$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$

Still Does Not Fit, Hey?

What's missing?

Still Does Not Fit, Hey?

What's missing?
Well, this:

$$v \in a[f : l] \rightarrow v \in a[f' : l']$$

Still Does Not Fit, Hey?

What's missing?
Well, this:

$$v \in a[f : l] \rightarrow v \in a[f' : l']$$

Suffices to prove

$$\{f \leq f' \leq l' \leq l \wedge (v \in a[f : l] \rightarrow v \in a[f' : l']) \wedge \mathit{sorted}(a)\}$$
$$\mathit{BinSearch}$$
$$\{f \leq m \leq l \wedge (v \in a[f : l] \rightarrow a[m] = v)\}$$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;$ *BinSearch*
  else if $a[m] > v$ **then**

    $l' := m;$ *BinSearch*
    else
    fi
  fi
else
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;$ *BinSearch*
  else if $a[m] > v$ **then**

    $l' := m;$ *BinSearch*
    else
    fi
  fi
else $\{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;$ *BinSearch*
  else if $a[m] > v$ **then**

    $l' := m;$ *BinSearch*
    else
    fi
  fi
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}$ $\{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;$ *BinSearch*
  else if $a[m] > v$ **then**

    $l' := m;$ *BinSearch*
    else
    fi
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}$ $\{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**

    $l' := m;$ $BinSearch$
    else
    fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**

    $l' := m;\ \{p\}\ BinSearch\ \{q\}$
    else $\{q\}$
    fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**

    $l' := m;\ \{p\}\ BinSearch\ \{q\}$
    else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
    fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**

    $\{p[l' := m]\}$
    $l' := m; \{p\}\ BinSearch\ \{q\}$
    else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
    fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**
      $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] > v\}$
      $\{p[l' := m]\}$
      $l' := m; \{p\}\ BinSearch\ \{q\}$
      else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
      fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**

    $\{p[f' := m + 1]\}$
    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**
      $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] > v\}$
      $\{p[l' := m]\}$
      $l' := m;\ \{p\}\ \ BinSearch\ \{q\}$
      else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
      fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$

if $f' \neq l'$ **then**
  if $a[m] < v$ **then**
    $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] < v\}$
    $\{p[f' := m + 1]\}$
    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**
      $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] > v\}$
      $\{p[l' := m]\}$
      $l' := m;\ \{p\}\ BinSearch\ \{q\}$
      else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
      fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

$\{p\}$
$m := (f' + l')\ div\ 2;$
$\{p \wedge m = (f' + l')\ div\ 2\}$
if $f' \neq l'$ **then**
  if $a[m] < v$ **then**
    $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] < v\}$
    $\{p[f' := m + 1]\}$
    $f' := m + 1;\ \{p\}\ BinSearch\ \{q\}$
  else if $a[m] > v$ **then**
      $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' \neq l' \wedge a[m] > v\}$
      $\{p[l' := m]\}$
      $l' := m;\ \{p\}\ BinSearch\ \{q\}$
      else $\{p \wedge m = (f' + l')\ div\ 2 \wedge a[m] = v\}\ \{q\}$
      fi $\{q\}$
  fi $\{q\}$
else $\{p \wedge m = (f' + l')\ div\ 2 \wedge f' = l'\}\ \{q\}$
fi
$\{q\}$

Soundness

A provable correctness formula is true.

Truth?

How To Prove Soundness

1. prove that all axioms are true
2. prove that all rules preserve truth

Completeness

Every true *correctness formula is* provable*.*

Gödel's Incompletness Theorem

There exist no complete proof system for deriving all arithmetic truths.

But

$$p \text{ is true iff } \{\textbf{true}\}\ \textit{skip}\ \{p\} \text{ is true}$$

Relative Completeness

We simply assume

Relative Completeness

We simply assume

- all valid assertions (in the consequence rule)

$$\models \{p\}\ S\ \{q\} \text{ iff } T \vdash \{p\}\ S\ \{q\}$$

where T consists of all valid assertions, and

Relative Completeness

We simply assume

- all valid assertions (in the consequence rule)

$$\models \{p\}\ S\ \{q\} \text{ iff } T \vdash \{p\}\ S\ \{q\}$$

  where T consists of all valid assertions, and
- *expressivity* of the weakest precondition $wp(S, q)$:

  $wp(S, q)$ holds in a state iff
  all final states of computations of S starting from
  that state satisfy postcondition q.

Intermezzo: Formal Compositional Input/Output Semantics

We define

$$R(S) \subseteq \Sigma \times \Sigma$$

where Σ is the set of states σ, e.g.,

- $\sigma(x) \in \mathbb{Z}$, where x is an integer variable,
- $\sigma(a) \in \mathbb{Z} \to \mathbb{Z}$, where a is an array variable of type **integer** $\to$ **integer**.

Intermezzo: Formal Compositional Input/Output Semantics

- $R(u := t) = \{\langle \sigma, \sigma[u := t] \rangle \mid \ \sigma \in \Sigma\}$

Intermezzo: Formal Compositional Input/Output Semantics

- $R(u := t) = \{\langle \sigma, \sigma[u := t] \rangle \mid \ \sigma \in \Sigma\}$
- $R(S_1; S_2) = R(S_1) \circ R(S_2)$,
  where $\circ$ denotes the composition operator on relations.

Intermezzo: Formal Compositional Input/Output Semantics

- $R(u := t) = \{\langle \sigma, \sigma[u := t]\rangle \mid \ \sigma \in \Sigma\}$
- $R(S_1; S_2) = R(S_1) \circ R(S_2)$,
  where $\circ$ denotes the composition operator on relations.
- $R(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}) = (R(b) \circ R(S_1)) \cup (R(\neg b) \circ R(S_2))$,
  where $R(p) = \{\langle \sigma, \sigma \langle | \ \sigma \models p\}$, for any assertion p.

Intermezzo: Formal Compositional Input/Output Semantics

- $R(u := t) = \{\langle \sigma, \sigma[u := t] \rangle \mid \ \sigma \in \Sigma\}$
- $R(S_1; S_2) = R(S_1) \circ R(S_2)$,
  where $\circ$ denotes the composition operator on relations.
- $R(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}) = (R(b) \circ R(S_1)) \cup (R(\neg b) \circ R(S_2))$,
  where $R(p) = \{\langle \sigma, \sigma \langle | \ \sigma \models p\}$, for any assertion p.
- $R(\textbf{while } b \textbf{ do } S \textbf{ od}) = (R(b) \circ R(S))^* \circ R(\neg b)$,
  where R^* denotes the reflexive, transitive closure of the relation R.

Formal Semantics Correctness Formulas

Partial Correctness

$$\models \{p\}\ S\ \{q\} \text{ iff}$$
$$\sigma \models p \text{ and } \langle \sigma, \sigma' \rangle \in R(S) \text{ implies } \sigma' \models q.$$

Total Correctness (note $R(S)$ is deterministic)

$$\models \{p\}\ S\ \{q\} \text{ iff}$$
$$\sigma \models p \text{ implies that there exists } \langle \sigma, \sigma' \rangle \in R(S) \text{ and } \sigma' \models q.$$

Weakest Precondition (Partial Correctness)

$$wp(S, q) = \{\sigma \mid \langle \sigma, \sigma' \rangle \in R(S) \text{ implies } \sigma' \models q\}.$$

General Properties Weakest Precondition

General Properties Weakest Precondition

- $\models \{wp(S, q)\}\ S\ \{q\}$

General Properties Weakest Precondition

- $\models \{wp(S, q)\}\ S\ \{q\}$
- $\{p\}\ S\ \{q\}$ iff $p \rightarrow wp(S, q)$

General Properties Weakest Precondition

- $\models \{wp(S, q)\}\ S\ \{q\}$
- $\{p\}\ S\ \{q\}$ iff $p \rightarrow wp(S, q)$
- $wp(u := t, q) \leftrightarrow q[u := t]$

Completeness Proof

Induction on S ($\models \{p\}\ S\ \{q\}$ implies $T \vdash \{p\}\ S\ \{q\}$):

Completeness Proof

Induction on S ($\models \{p\}\ S\ \{q\}$ implies $T \vdash \{p\}\ S\ \{q\}$):

- Let $\models \{p\}\ u := t\ \{q\}$.
  It follows that $\models p \to q[u := t]$.

Completeness Proof

Induction on S ($\models \{p\}\ S\ \{q\}$ implies $T \vdash \{p\}\ S\ \{q\}$):

- Let $\models \{p\}\ u := t\ \{q\}$.
  It follows that $\models p \rightarrow q[u := t]$.
- Let $\models \{p\}\ S_1; S_2\ \{q\}$.
  It follows that $\models \{p\}\ S_1\ \{wp(S_2, q)\}$ and
  $\models \{wp(S_2, q)\}\ S_2\ \{q\}$.

Completeness Proof

Induction on S ($\models \{p\}\ S\ \{q\}$ implies $T \vdash \{p\}\ S\ \{q\}$):

- Let $\models \{p\}\ u := t\ \{q\}$.
  It follows that $\models p \to q[u := t]$.
- Let $\models \{p\}\ S_1; S_2\ \{q\}$.
  It follows that $\models \{p\}\ S_1\ \{wp(S_2, q)\}$ and
  $\models \{wp(S_2, q)\}\ S_2\ \{q\}$.
- Let $\models \{p\}$ **if** b **then** S_1 **else** S_2 **fi** $\{q\}$.
  It follows that $\models \{p \wedge b\}\ S_1\ \{q\}$ and $\models \{p \wedge \neg b\}\ S_2\ \{q\}$.

Completeness Proof

Induction on S ($\models \{p\}\ S\ \{q\}$ implies $T \vdash \{p\}\ S\ \{q\}$):

- Let $\models \{p\}\ u := t\ \{q\}$.
  It follows that $\models p \to q[u := t]$.
- Let $\models \{p\}\ S_1; S_2\ \{q\}$.
  It follows that $\models \{p\}\ S_1\ \{wp(S_2, q)\}$ and
  $\models \{wp(S_2, q)\}\ S_2\ \{q\}$.
- Let $\models \{p\}$ **if** b **then** S_1 **else** S_2 **fi** $\{q\}$.
  It follows that $\models \{p \wedge b\}\ S_1\ \{q\}$ and $\models \{p \wedge \neg b\}\ S_2\ \{q\}$.
- Let $\models \{p\}$ **while** b **do** S **od** $\{q\}$.
  It follows that $\models$
  $\{wp(\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od}, q) \wedge b\}\ S\ \{wp(\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od}, q)\}$
  and
  $\models (wp(\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od}, q) \wedge \neg b) \to q$.

Weakest Precondition While Statement

We show that
$\models \{wp(\textbf{while } b \textbf{ do } S \textbf{ od}, q) \wedge b\}\ S\ \{wp(\textbf{while } b \textbf{ do } S \textbf{ od}, q)\}$:
Let $\sigma \models wp(\textbf{while } b \textbf{ do } S \textbf{ od}, q) \wedge b$ and $\langle \sigma, \sigma' \rangle \in R(S)$.
Note that $\sigma \models b$ and $\langle \sigma, \sigma' \rangle \in R(S)$ implies $\langle \sigma, \sigma' \rangle \in R(b) \circ R(S)$.
To prove: $\sigma' \models wp(\textbf{while } b \textbf{ do } S \textbf{ od}, q)$, that is,
$\langle \sigma', \sigma'' \rangle \in R(\textbf{while } b \textbf{ do } S \textbf{ od})$ implies $\sigma'' \models q$.
Let $\langle \sigma', \sigma'' \rangle \in R(\textbf{while } b \textbf{ do } S \textbf{ od}) = (R(b) \circ R(S))^* \circ R(\neg b)$.
$\langle \sigma, \sigma' \rangle \in R(b) \circ R(S)$ and $\langle \sigma', \sigma'' \rangle \in (R(b) \circ R(S))^* \circ R(\neg b)$
implies
$\langle \sigma, \sigma'' \rangle \in (R(b) \circ R(S)) \circ (R(b) \circ R(S))^* \circ R(\neg b)$.
Note that
$(R(b) \circ R(S)) \circ (R(b) \circ R(S))^* \circ R(\neg b) \sqsubseteq (R(b) \circ R(S))^* \circ R(\neg b)$.
And so, $\langle \sigma, \sigma'' \rangle \in (R(b) \circ R(S))^* \circ R(\neg b) = R(\textbf{while } b \textbf{ do } S \textbf{ od})$.
Since $\sigma \models wp(\textbf{while } b \textbf{ do } S \textbf{ od}, q)$, we conclude $\sigma'' \models q$.

But This Even Won't DO In All Cases

For a language containing (e.g., Algol)

- Procedures as parameters (higher-order)
- Local variable and procedure declarations
- Static scoping
- Recursion

there does not exist a relative complete proof system!!!!!

> *Edmund M. Clarke: Programming Language Constructs for Which it is Impossible to Obtain "Good" Hoare-Like Axiom Systems. POPL 1977: 10-20*

Program Correctness in Practice

- Microsoft Spec#
- The Eiffel programming language
- The Java Modeling Language (JML)
- The Object Constraint Language (OCL)
- Run-time Assertion Checking
- The KeY Project: Integrated Deductive Software Design
- Separation Logic AnalYzER
- Temporal Logic of Actions, Leslie Lamport (Turing Award 2013)

Major Challenges

- Tool support (e.g., theorem proving, proof reuse, counter-example generation, ...)
- Object-orientation (heaps, sub-typing, ...)
- Multithreading (interference, locks, ...)