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!

bug_1223345.jpg

The TimSort Bug

http://www.envisage-project.eu

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

Neumann.jpeg

Assembly Language

assembly.jpg

One of the Founding Fathers of Computer Science: Alan Turing

Turing.jpg

The Turing Machine

TuringMachine.png

Edsger Dijkstra Introduced Structured Programming

Dijkstra.jpeg

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 $y := y + 1;$
$\quad u := u + v;$
$\quad v := v + 2$
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

Floyd.jpg

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

Sir. Tony Hoare Developed a First Programming Logic

Hoare.jpeg

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

- $\{\mathbf{true}\}$
  $\mathbf{if}\ x > y\ \mathbf{then}\ m := x\ \mathbf{else}\ m := y\ \mathbf{fi}$
  $\{(m = x \land x > y) \lor (m = y \land x \leq y)\}$
- $\{\mathbf{true}\}$
  $\mathbf{if}\ y \neq 0\ \mathbf{then}\ z := x\ \mathit{div}\ y\ \mathbf{else}\ \mathit{skip}\ \mathbf{fi}$
  $\{y \neq 0 \to z = x\ \mathit{div}\ y\}$

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

$$
\begin{array}{rcll}
S & ::= & \mathit{skip} & \text{skip statement} \\
 & \mid & u := t & \text{assignment} \\
 & \mid & S_1; S_2 & \text{sequential composition} \\
 & \mid & \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} & \text{choice} \\
 & \mid & \textbf{while } B \textbf{ do } S_1 \textbf{ od} & \text{iteration}
\end{array}
$$

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
  $\textbf{integer} \times \textbf{integer} \rightarrow \textbf{integer}$,
- $=, <$ of type
  $\textbf{integer} \times \textbf{integer} \rightarrow \textbf{Boolean}$,
- $\neg$ of type
  $\textbf{Boolean} \rightarrow \textbf{Boolean}$,
- $=, \vee, \wedge, \rightarrow, \leftrightarrow$ of type
  $\textbf{Boolean} \times \textbf{Boolean} \rightarrow \textbf{Boolean}$.

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

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

Example:

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

Axioms for SKIP and Assignment

AXIOM 1: SKIP

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

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

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

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

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

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:

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

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

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. $\{x \geq 0 \wedge y \geq 0\} q := 0; r := x \{q \cdot y + r = x \wedge r \geq 0\}$
   (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* $\{q \cdot y + r = x \wedge 0 \leq r < y\}$
    (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

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

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

Initialization

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

Termination

$$\begin{aligned} &\forall 1 \leq n < i : a[n] = b[n] \wedge \neg(i < k) \\ &\rightarrow \forall n : 1 \leq n < k : a[n] = b[n] \end{aligned}$$

Array assignment To this end we compute

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

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 $a[i:j]$:

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

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

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

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

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

where

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

Synthesis

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

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

Note that

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

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

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

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

$\{N > 0\}$
$\{1 \leq 1 \leq N \wedge a[0] = s_1 \wedge a[0] = t_1\}$
$k := 1;\ sum := a[0];\ x := a[0];$
$\{1 \leq k \leq N \wedge sum = s_k \wedge x = t_k\}$
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 $\{x > 0 \wedge y > 0 \wedge n = ggd(x, y) \wedge x \neq y\}$
$\{x > 0 \wedge y > 0 \wedge n = ggd(x, y)\}$
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
$\{x = y \wedge n = ggd(x, y)\}$
$\{x = y \wedge x = n\}$

Exercises

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:

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

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

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