A Logical Approach to Discrete Math

English to math (All types are integers.)

x is positive.	$x > 0$
x is negative.	$x < 0$
x is non-negative.	$\neg(x < 0)$ or $x \geq 0$
x is even.	$(\exists i \mid : x = 2 \cdot i)$
x is odd.	$(\exists i \mid : x = 2 \cdot i + 1)$
x divides y. $x \mid y$	$(\exists i \mid : x \cdot i = y)$
x is a power of 2.	$(\exists i \mid : x = 2^i)$

The element 13 is in $b[j..k]$.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
b	23	14	-6	5	-7	13	23	4	19	-2

$j = 4$ $\quad$ $k = 7$

$$(\exists i \mid j \le i \le k : b[i] = 13)$$

$x \in b[0..n-1]$ $\qquad$ $(\exists i \mid 0 \le i < n : x = b[i])$

Hoare Triple

Recall from Sec. 1.6 that a *state* is a set of identifier-value pairs. Further, the Hoare triple $\{Q\}\ S\ \{R\}$, where S is a program statement, Q is the precondition, and R is the postcondition, has the interpretation

> Execution of S begun in any state in which Q is *true* is guaranteed to terminate, and R is *true* in the final state.

Formal Specification

To specify a program is to say <u>what</u> it should do, not <u>how</u> it should do it.

Formal Specification

A specification of a program should give:

- a precondition Q (say): a boolean expression that describes the initial states for which execution of the program is being defined,
- a list x (say) of variables that may be assigned to, and
- a postcondition R (say): a boolean expression that characterizes the final states, after execution of the program.

$$\{Q\}\ x := ?\ \{R\}$$

Specification examples (All types are integers.)

Specify "Set x to y's value."

$\{true\} \quad x := ? \quad \{x = y\}$

Specify "Set y to x's value."

$\{true\} \quad y := ? \quad \{x = y\}$

Specify "Set x and y to have the same value."

$\{true\} \quad x, y := ? \quad \{x = y\}$

Specify "Swap the values of x and y."

This specification requires a <u>rigid variable</u>.

A rigid variable defines the initial value of a variable in the precondition, so it can be used in the postcondition.

$$\{x = \mathtt{X} \wedge y = \mathtt{Y}\} \quad x, y := ? \quad \{x = \mathtt{Y} \wedge y = \mathtt{X}\}$$

Specify "Set z to its own absolute value."

$\{z = \mathtt{Z}\} \quad z := ? \quad \{z = \mid \mathtt{Z} \mid\}$

Specify "Set z to the maximum of integers x and y."

$\{true\} \quad z := ? \quad \{(x \geq y \Rightarrow z = x) \wedge (y \geq x \Rightarrow z = y)\}$

Arrays

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	
b									$n = 8$

Abbreviation

$$x \in b[0..n-1] \quad \text{means} \quad (\exists i \mid 0 \leq i < n : x = b[i])$$

Specify "Set i to the index of x assuming x is in b."

$\{0 < n \wedge x \in b[0..n-1]\} \quad i := ? \quad \{0 \leq i < n \wedge x = b[i]\}$

Specify "Set i to the index of x if it is in b and to n if it is not."

$\{0 \leq n\} \quad i := ? \quad \{(0 \leq i < n \wedge x = b[i]) \vee (i = n \wedge x \notin b[0..n-1])\}$

Specify "If x is in b set boolean c to $true$ and i to the index of x. Otherwise set c to $false$."

$\{0 \leq n\} \quad i, c := ? \quad \{(c \equiv x \in b[0..n-1]) \ \wedge \ (c \Rightarrow x = b[i])\}$

Appropriate preconditions

Sum

It makes sense to have the sum of an empty range.

For the sum of $b[j..k-1]$, the precondition should include $j \leq k$.

Max

There is no maximum in an empty range.

For the maximum of $b[j..k-1]$, the precondition should include $j < k$.

Counting

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	
b	12	93	0	14	6	0	0	21	$n = 8$

How many zeros are in $b[0..7]$?

$$(\Sigma i \mid 0 \leq i < n \wedge b[i] = 0 : 1) = 3$$

Weakest precondition

Suppose

$$\{P\}x := E\{R\}$$

and

$$\{Q\}x := E\{R\}$$

are two valid Hoare triples with the same program statements and the same postconditions.
P is called the weakest precondition if

$$Q \Rightarrow P$$

for all Q that make the Hoare triple valid.

Weakest precondition

Example

$$\{x = 4\}x := x + 1\{x < 7\} \quad \text{valid}$$

$$\{x < 6\}x := x + 1\{x < 7\} \quad \text{valid}$$

Note that

$$x = 4 \Rightarrow x < 6$$

Any precondition that makes this $S\{R\}$ valid implies $x < 6$.

$$\underbrace{x = 4}_{\text{strong}} \Rightarrow \underbrace{x < 6}_{\text{weak}}$$

A Logical Approach to Discrete Math

Notation for weakest precondition

$$wp.S.post \equiv P$$

means that

$$\{P\}S\{post\}$$

is valid, and for every Q satisfying

$$\{Q\}S\{post\}$$

Q must be stronger than P. That is,

$$Q \Rightarrow P$$

<u>Example</u>

$$wp.(x := x+1).(x < 7) \equiv x < 6$$

A Logical Approach to Discrete Math

A Theory of Programs

(p.1) **Axiom, Excluded miracle:** $wp.S.\mathit{false} \equiv \mathit{false}$

(p.2) **Axiom, Conjunctivity:** $wp.S.(X \wedge Y) \equiv wp.S.X \wedge wp.S.Y$

(p.3) **Monotonicity:** $(X \Rightarrow Y) \Rightarrow (wp.S.X \Rightarrow wp.S.Y)$

(p.4) **Definition, Hoare triple:** $\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp.S.R$

(p.4.1) $\{wp.S.R\}\ S\ \{R\}$

(p.5) **Postcondition rule:** $\{Q\}\ S\ \{A\} \wedge (A \Rightarrow R) \Rightarrow \{Q\}\ S\ \{R\}$

(p.6) **Definition, Program equivalence:** $S = T \equiv$ (For all $R, wp.S.R \equiv wp.T.R$)

(p.7) $(Q \Rightarrow A) \wedge \{A\}\ S\ \{R\} \Rightarrow \{Q\}\ S\ \{R\}$

(p.8) $\{Q0\}\ S\ \{R0\} \wedge \{Q1\}\ S\ \{R1\} \Rightarrow \{Q0 \wedge Q1\}\ S\ \{R0 \wedge R1\}$

(p.9) $\{Q0\}\ S\ \{R0\} \wedge \{Q1\}\ S\ \{R1\} \Rightarrow \{Q0 \vee Q1\}\ S\ \{R0 \vee R1\}$

A Logical Approach to Discrete Math

Prove (p.3) Monotonicity: $(X \Rightarrow Y) \;\Rightarrow\; (wp.S.X \Rightarrow wp.S.Y)$

Proof

$$
\begin{array}{ll}
 & wp.S.X \Rightarrow wp.S.Y \\
= & \langle (3.60) \rangle \\
 & wp.S.X \wedge wp.S.Y \equiv wp.S.X \\
= & \langle (\mathrm{p}.2) \rangle \\
 & wp.S.(X \wedge Y) \equiv wp.S.X \\
\Leftarrow & \langle (3.83)\ \text{Leibniz with } E, e, f := wp.S.z,\ X \wedge Y,\ X \\
 & \quad X \wedge Y = X \Rightarrow (wp.S.z)[z := X] = (wp.S.z)[z := X \wedge Y] \\
 & \quad X \wedge Y = X \Rightarrow wp.S.X = wp.S.(X \wedge Y) \rangle \\
 & X \wedge Y = X \\
= & \langle (3.60) \rangle \\
 & X \Rightarrow Y \quad //
\end{array}
$$

Prove (p.4.1) $\{wp.S.R\}\ S\ \{R\}$

Proof

$$\{wp.S.R\}\ S\ \{R\}$$

$= \quad \langle(\text{p.4})\rangle$

$$wp.S.R \Rightarrow wp.S.R$$

which is (3.71) Reflexivity if $\Rightarrow$ //

(p.6) **Definition, Program equivalence:** $S = T \quad \equiv \quad (\text{For all } R, wp.S.R \equiv wp.T.R)$

In (p.6), you cannot use the $\forall$ symbol because R is an expression, not a dummy variable.
S and T are programs statements

Sets

See (11.4) and (11.11b).
To prove set S equals set T, let v be an arbitrary element, and prove

$$v \in S \equiv v \in T$$

Programs

To prove program S equals program T, let R be an arbitrary postcondition, and prove

$$wp.S.R \equiv wp.T.R$$

(p.10) **Definition, skip:** $wp.skip.R \equiv R$

(p.11) $\{Q\}\ skip\ \{R\} \equiv Q \Rightarrow R$

The *skip* statement does nothing.
If R is true and you execute *skip*,
R is guaranteed to be true.

(p.12) **Definition, abort:** $wp.abort.R \equiv false$

(p.13) $\{Q\}\ abort\ \{R\} \equiv Q \equiv false$

The *abort* statement causes the program to fail.
An abort statement can never establish its postcondition because its precondition can never be true.
A program that executes *abort* is erroneous.

(p.14) **Definition, Composition:** $wp.(S;T).R \equiv wp.S.(wp.T.R)$

(p.15) $\{Q\}\ S\ \{H\} \ \wedge\ \{H\}\ T\ \{R\} \ \Rightarrow\ \{Q\}\ S;T\ \{R\}$

(p.14) says that if you execute S and then execute T, the postcondition of S is the precondition of T.

Prove (p.16a) $S\ ;\ skip = S$

Proof

Let R be an arbitrary postcondition, and prove that

$wp.(S\ ;\ skip).R \equiv wp.S.R$

$$\begin{array}{ll} & wp.(S\ ;\ skip).R \\ = & \langle (p.14) \rangle \\ & wp.S.(wp.\ skip\ .R) \\ = & \langle (p.10) \rangle \\ & wp.S.R \quad // \end{array}$$

(p.18) **Definition, Assignment:** $wp.(x := E).R \equiv R[x := E]$

Example

Compute the weakest precondition P for the following program. A and B are program constants, not rigid variables.

int x, y
const int A, B
$\{P\}\ x := x + y\ ;\ y := x - y\ \{x = A \wedge y = B\}$

A Logical Approach to Discrete Math

$$\{P\}\; x := x + y\; ;\; y := x - y\; \{x = A \wedge y = B\}$$

$$
\begin{array}{ll}
 & wp.(x := x + y\; ;\; y := x - y).(x = A \wedge y = B) \\
= & \quad \langle \text{(p.14)} \rangle \\
 & wp.(x := x + y)\; .\; (wp.(y := x - y).(x = A \wedge y = B)) \\
= & \quad \langle \text{(p.18) and t.s.} \rangle \\
 & wp.(x := x + y).(x = A \wedge x - y = B) \\
= & \quad \langle \text{(p.18) and t.s.} \rangle \\
 & x + y = A \wedge x + y - y = B \\
= & \quad \langle \text{Math} \rangle \\
 & x + y = A \wedge x = B \\
= & \quad \langle \text{(3.84a) Substitution} \rangle \\
 & x = B \wedge y = A - B \qquad //
\end{array}
$$

$$\{x = B \land y = A - B\}\ x := x + y\ ;\ y := x - y\ \{x = A \land y = B\}$$

<u>Example</u>

$A = 7, B = 4$

$$\{x = 4 \land y = 3\}\ x := x + y\ ;\ y := x - y\ \{x = 7 \land y = 4\}$$

Two applications

Program derivation

Given an assignment statement in a program with an unknown expression in the assignment, solve for the unknown expression.

Program correctness

Given a program, prove that it satisfies its specification.
In other words, prove that the program is correct.

Program derivation example

Solve for unknown E in the program

int x

$\{true\}\ x := E\ \{x = 4\}$

$$
\begin{array}{ll}
 & \{true\}\ x := E\ \{x = 4\} \\
= & \langle \text{(p.4)} \rangle \\
 & true \Rightarrow wp.(x := E).(x = 4) \\
= & \langle \text{(3.73)} \rangle \\
 & wp.(x := E).(x = 4) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & E = 4
\end{array}
$$

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

Program derivation example

From the division algorithm, where q is the quotient and r is the remainder when you divide x by y. Solve for unknown E in the program

int x, y, q, r

$$\{0 \leq x \wedge 0 < y\}\ q, r := E, x\ \{0 \leq r \wedge q * y + r = x\}$$

By (p.4) we must have

$$0 \leq x \wedge 0 < y \Rightarrow wp.(q, r := E, x).(0 \leq r \wedge q * y + r = x)$$

Use (4.4) Deduction (assume the conjuncts of the antecedent)

$$0 \leq x \wedge 0 < y \Rightarrow wp.(q, r := E, x).(0 \leq r \wedge q * y + r = x)$$

$$\begin{array}{ll}
& wp.(q, r := E, x).(0 \leq r \wedge q * y + r = x) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
& 0 \leq x \wedge E * y + x = x \\
= & \langle \text{Assume conjunct } 0 \leq x \rangle \\
& true \wedge E * y + x = x \\
= & \langle \text{(3.39) and math} \rangle \\
& E * y = 0 \\
= & \langle \text{Conjunct } 0 < y \text{ and math} \rangle \\
& E = 0
\end{array}$$

$$\{0 \leq x \wedge 0 < y\}\ q, r := 0, x\ \{0 \leq r \wedge q * y + r = x\}$$

int x, y

$\{x = \mathtt{X} \wedge y = \mathtt{Y}\}$

$x := E \; ; \; y := x + y$

$\{x = \mathtt{X} - \mathtt{Y} \wedge y = \mathtt{X}\}$

Rigid variables cannot occur in E.

$$
\begin{array}{ll}
 & wp.(x := E \; ; \; y := x+y).(x = \mathtt{X} - \mathtt{Y} \wedge y = \mathtt{X}) \\
= & \langle \text{(p.14)} \rangle \\
 & wp.(x := E).(wp.(y := x+y).(x = \mathtt{X} - \mathtt{Y} \wedge y = \mathtt{X})) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & wp.(x := E).(x = \mathtt{X} - \mathtt{Y} \wedge x + y = \mathtt{X}) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & E = \mathtt{X} - \mathtt{Y} \wedge E + y = \mathtt{X} \\
= & \langle \text{Assume conjuncts } x = \mathtt{X} \text{ and } y = \mathtt{Y} \rangle \\
 & E = x - y \wedge E + y = x \\
= & \langle \text{(3.38)} \rangle \\
 & E = x - y
\end{array}
$$

$\{x = \mathtt{X} \wedge y = \mathtt{Y}\} x := x - y \; ; \; y := x + y \{x = \mathtt{X} - \mathtt{Y} \wedge y = \mathtt{X}\}$

Deriving sequential compositions

$$\{x = \mathtt{X} \wedge y = \mathtt{Y}\}\ y := E\ ;\ x := F\ \{x = \mathtt{Y} \wedge y = \mathtt{X} + \mathtt{Y}\}$$

$$\begin{array}{ll}
 & wp.(y := E\ ;\ x := F).(x = \mathtt{Y} \wedge y = \mathtt{X} + \mathtt{Y}) \\
= & \langle \text{(p.14) Definition, Composition} \rangle \\
 & wp.(y := E\ .\ wp.(x := F).(x = \mathtt{Y} \wedge y = \mathtt{X} + \mathtt{Y})) \\
= & \langle \text{(p.18) and textual substitution} \rangle \\
 & wp.(y := E).(F = \mathtt{Y} \wedge y = \mathtt{X} + \mathtt{Y}) \\
= & \langle \text{(p.18) and textual substitution} \rangle \\
 & F^y_E = \mathtt{Y} \wedge E = \mathtt{X} + \mathtt{Y} \\
= & \langle \text{Assume conjuncts } x = \mathtt{X} \text{ and } y = \mathtt{Y} \rangle \\
 & F^y_E = y \wedge E = x + y \\
= & \langle \text{(3.84a) Substitution } (e = f) \wedge E^z_e \ \equiv\ (e = f) \wedge E^z_f \rangle \\
 & F^y_{x+y} = y \wedge E = x + y
\end{array}$$

Deriving sequential compositions

$$\{x = \mathtt{X} \wedge y = \mathtt{Y}\}\ y := E\ ;\ x := F\ \{x = \mathtt{Y} \wedge y = \mathtt{X} + \mathtt{Y}\}$$

$$F^{y}_{x+y} = y \wedge E = x + y$$

$F = y - x$

because

$F^{y}_{x+y} = F[y := x + y] = (y - x)[y := x + y] = x + y - x = y$

So, the program is

$y := x + y\ ;\ x := y - x$

Invariant

Invariant

An invariant is a conjunct that appears in both the precondition and the postcondition.

Example

int x, y, q, r

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

$q, r := ?$

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

invariant

Abbreviation

$P1: \quad 0 \leq r \wedge q \cdot y + r = x$

int x, y, q, r

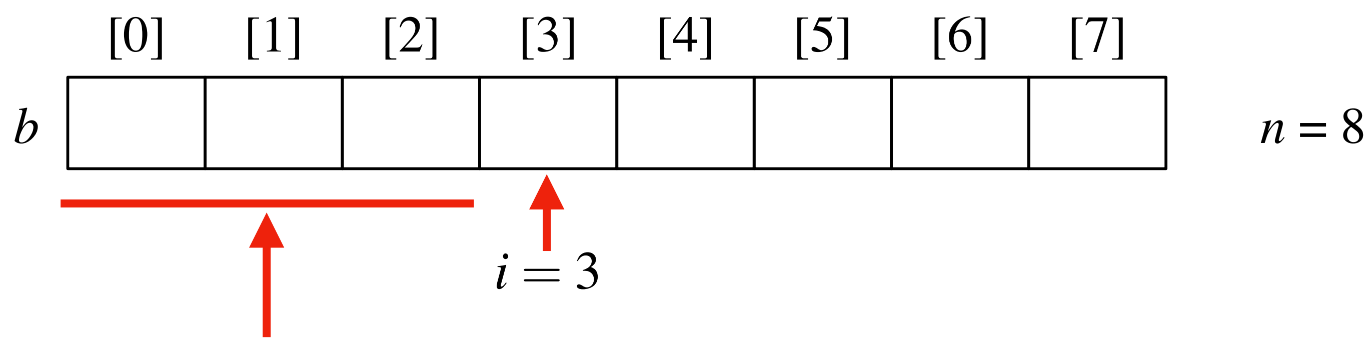
$\{P1\}$

$q, r := ?$

$\{P1 \wedge r < y\}$

Example

$P1: \quad x = (\Sigma k \mid 0 \leq k < i : b[k])$



$$x = b[0] + b[1] + b[2]$$

We want to increment i by 1 and to maintain the invariant.
Afterwords, we want $i = 4$ and $x = b[0] + b[1] + b[2] + b[3]$

const int n
int $i, x, b[n]$
$\{P1\}\ i, x := i+1, E\ \{P1\}$

A Logical Approach to Discrete Math

$$P1: \quad x = (\Sigma k \mid 0 \leq k < i : b[k])$$

$$\{P1\}\ i,x := i+1, E\ \{P1\}$$

$$\begin{array}{ll} & wp.(i,x := i+1, E).P1 \\ = & \langle \text{(p.18) and t.s.} \rangle \\ & E = (\Sigma k \mid 0 \leq k < i+1 : b[k]) \\ = & \langle \text{Split off last term} \rangle \\ & E = (\Sigma k \mid 0 \leq k < i : b[k]) + b[i] \\ = & \langle \text{Assume conjunct } P1 \rangle \\ & E = x + b[i] \end{array}$$

$$\{P1\}\ i,x := i+1, x+b[i]\ \{P1\}$$

Program correctness

(p.19) **Proof method for assignment:**
To show that $x := E$ is an implementation of $\{Q\} x := ? \{R\}$, prove $Q \Rightarrow R[x := E]$.

Example

Prove the correctness of the following program.

int x, y

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

Use (p.4) and deduction.

$$
\begin{array}{cl}
 & wp.(x,y := x+1, x+y).(x \geq y) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & x+1 \geq x+y \\
= & \langle \text{Assume antecedent } y = 1 \rangle \\
 & x+1 \geq x+1 \\
= & \langle \text{Math} \rangle \\
 & true \quad //
\end{array}
$$

Example

Prove the correctness of the following program.

int x, y

$\{x = \mathtt{X} \wedge y = \mathtt{Y}\}\ x := x + y\ ;\ y := x - y\ ;\ x := x - y\ \{x = \mathtt{Y} \wedge y = \mathtt{X}\}$

$$
\begin{array}{ll}
& wp.(x := x+y\ ;\ y := x-y\ ;\ x := x-y).(x = \mathtt{Y} \wedge y = \mathtt{X}) \\
= & \langle \text{(p.14)} \rangle \\
& wp.(x := x+y\ ;\ y := x-y).(wp.(x := x-y).(x = \mathtt{Y} \wedge y = \mathtt{X})) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
& wp.(x := x+y\ ;\ y := x-y).(x-y = \mathtt{Y} \wedge y = \mathtt{X}) \\
= & \langle \text{(p.14)} \rangle \\
& wp.(x := x+y).(wp.(y := x-y).(x-y = \mathtt{Y} \wedge y = \mathtt{X}))
\end{array}
$$

A Logical Approach to Discrete Math

$$
\begin{array}{ll}
 & wp.(x := x+y).(wp.(y := x-y).(x-y = \mathtt{Y} \wedge y = \mathtt{X})) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & wp.(x := x+y).(x-(x-y) = \mathtt{Y} \wedge x-y = \mathtt{X}) \\
= & \langle \text{Math} \rangle \\
 & wp.(x := x+y).(y = \mathtt{Y} \wedge x-y = \mathtt{X}) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & y = \mathtt{Y} \wedge x+y-y = \mathtt{X} \\
= & \langle \text{Math} \rangle \\
 & y = \mathtt{Y} \wedge x = \mathtt{X} \\
= & \langle \text{Assume conjuncts } x = \mathtt{X} \text{ and } y = \mathtt{Y} \rangle \\
 & \mathit{true} \quad //
\end{array}
$$

(p.19) **Proof method for assignment:** (p.19) is (10.2)

To show that $x := E$ is an implementation of $\{Q\}x :=?\{R\}$, prove $Q \Rightarrow R[x := E]$.

(p.20) $(x := x) = skip$

(p.21) IFG : (p.21) is (10.6)

if $B1 \rightarrow S1$

[] $B2 \rightarrow S2$

[] $B3 \rightarrow S3$

fi

(p.22) **Definition,** IFG**:** $wp.IFG.R \equiv (B1 \vee B2 \vee B3) \wedge$
$B1 \Rightarrow wp.S1.R \wedge B2 \Rightarrow wp.S2.R \wedge B3 \Rightarrow wp.S3.R$

(p.23) **Empty guard:** **if fi = ** *abort*

The alternative statement

(p.21) $IFG:$

$$\begin{array}{l} \textbf{if } B1 \rightarrow S1 \\ [\,]\ \ B2 \rightarrow S2 \\ [\,]\ \ B3 \rightarrow S3 \\ \textbf{fi} \end{array}$$

There are two key points with the alternative statement.

- Execution aborts if no guard is *true* .
- If more than one guard is *true* , only one of them is chosen (arbitrarily) and its corresponding command is executed.

Example

$$\begin{array}{l} \textbf{if } a < 18 \rightarrow t := 0 \\ [\,]\ \ 18 \leq a < 21 \rightarrow t := 5 \\ [\,]\ \ 21 \leq a < 65 \rightarrow t := 10 \\ \textbf{fi} \end{array}$$

Initial value of $a = 15 \quad \Rightarrow \quad$ final value of $t = 0$

Initial value of $a = 20 \quad \Rightarrow \quad$ final value of $t = 5$

Initial value of $a = 30 \quad \Rightarrow \quad$ final value of $t = 10$

Initial value of $a = 70 \quad \Rightarrow \quad$ abort

<u>Example</u>

$$\begin{array}{l} \textbf{if } a < 18 \rightarrow t := 0 \\ [] \;\; a < 21 \rightarrow t := 5 \\ \textbf{fi} \end{array}$$

Initial value of $a = 15 \;\;\Rightarrow\;\;$ final value of $t = 0$ or $t = 5$
because both guards are true.

Initial value of $a = 20 \;\;\Rightarrow\;\;$ final value of $t = 5$

Initial value of $a = 30 \;\;\Rightarrow\;\;$ abort

Example

$$\begin{array}{l} \textbf{if } a < 18 \rightarrow t := 0 \\ [] \;\; 18 \leq a < 21 \rightarrow t := 5 \\ [] \;\; 21 \leq a \rightarrow \textit{skip} \\ \textbf{fi} \end{array}$$

Cannot abort

(p.24) **Proof method for IFG:** (p.24) is (10.7)

To prove $\{Q\}IFG\{R\}$, it suffices to prove

(a) $Q \Rightarrow B1 \vee B2 \vee B3$,

(b) $\{Q \wedge B1\}\ S1\ \{R\}$,

(c) $\{Q \wedge B2\}\ S2\ \{R\}$, and

(d) $\{Q \wedge B3\}\ S3\ \{R\}$.

(p.25) $\neg(B1 \vee B2 \vee B3) \Rightarrow\ IFG\ =\ \mathit{abort}$

(p.26) **One-guard rule:** $\{Q\}$ **if** $B \rightarrow S$ **fi** $\{R\}\ \Rightarrow\ \{Q\}\ S\ \{R\}$

(p.27) **Distributivity of program over alternation:**

if $B1 \rightarrow S1; T$ [] $B2 \rightarrow S2; T$ **fi** $=$ **if** $B1 \rightarrow S1$ [] $B2 \rightarrow S2$ **fi** $; T$

Example

Verify, the correctness of the following program.

$$
\begin{array}{l}
\text{int } x, y, z \\
\{x > z\} \\
\textbf{if } x > y \rightarrow x, y := y, x \\
[] \;\; y > z \rightarrow y, z := z, y \\
\textbf{fi} \\
\{x \leq y \vee y \leq z\}
\end{array}
$$

By (p.24), must prove

(a) $x > z \;\Rightarrow\; x > y \;\vee\; y > z$

(b) $\{x > z \wedge x > y\}\; x, y := y, x\; \{x \leq y \vee y \leq z\}$

(c) $\{x > z \wedge y > z\}\; y, z := z, y\; \{x \leq y \vee y \leq z\}$

Proof of (a)

$$
\begin{array}{ll}
 & x > z \;\Rightarrow\; x > y \;\vee\; y > z \\
= & \langle \text{Contrapositive} \rangle \\
 & \neg(x > y \;\vee\; y > z) \Rightarrow \neg(x > z) \\
= & \langle \text{De Morgan and math} \rangle \\
 & x \leq y \;\wedge\; y \leq z \Rightarrow x \leq z \\
= & \langle \text{Math, transitivity of } \leq \rangle \\
 & \mathit{true} \quad //
\end{array}
$$

Proof of (b)

$\{x > z \wedge x > y\}\ x,y := y,x\ \{x \leq y \vee y \leq z\}$

$$
\begin{array}{ll}
 & wp.(x,y := y,x).(x \leq y \vee y \leq z) \\
= & \quad \langle \text{(p.18) and t.s.} \rangle \\
 & y \leq x \vee x \leq z \\
= & \quad \langle \text{Assume conjunct } x > z \text{ and math} \rangle \\
 & y \leq x \vee false \\
= & \quad \langle \text{(3.30) Identity of } \vee \rangle \\
 & y \leq x \\
= & \quad \langle \text{Math} \rangle \\
 & y < x \vee y = x \\
= & \quad \langle \text{Assume conjunct } x > y \rangle \\
 & true \vee y = x \\
= & \quad \langle \text{(3.29) Zero of } \vee \rangle \\
 & true \quad //
\end{array}
$$

A Logical Approach to Discrete Math

Proof of (c)

$\{x > z \wedge y > z\}\ y,z := z,y\ \{x \le y \vee y \le z\}$

$$
\begin{array}{ll}
 & wp.(y,z := z,y).(x \le y \vee y \le z) \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & x \le z \vee z \le y \\
= & \langle \text{Assume conjunct } x > z \text{ and math} \rangle \\
 & false \vee z \le y \\
= & \langle \text{(3.30) Identity of } \vee \rangle \\
 & z \le y \\
= & \langle \text{Math} \rangle \\
 & z < y \vee z = y \\
= & \langle \text{Assume conjunct } y > z \rangle \\
 & true \vee z = y \\
= & \langle \text{(3.29) Zero of } \vee \rangle \\
 & true \quad //
\end{array}
$$

The alternative statement in the Promela language

```
active proctype P() {
byte a = 5, b = 5;
byte max, branch;
if
  :: a >= b -> max = a; branch = 1
  :: a <= b -> max = b; branch = 2
fi
}
```

(p.28) $DO: \quad \textbf{do } B \rightarrow S \textbf{ od}$

(p.29) **Fundamental Invariance Theorem.** (p.29) is (12.43)

Suppose

- $\{P \wedge B\}\ S\ \{P\}$ holds—i.e. execution of S begun in a state in which P and B are $true$ terminates with P $true$—and
- $\{P\}\ \textbf{do } B \rightarrow S \textbf{ od}\ \{true\}$—i.e. execution of the loop begun in a state in which P is $true$ terminates.

Then $\{P\}\ \textbf{do } B \rightarrow S \textbf{ od}\ \{P \wedge \neg B\}$ holds.

A Logical Approach to Discrete Math

Example

int x, i

$x, i := 0, 0$;

do $i < 4 \;\rightarrow\; i, x := i+1, x+i$ **od**

Guard $i < 4$	i	x	$x = (\Sigma k \mid 0 \leq k < i : k)$
?	?	?	
	0	0	$0 = (\Sigma k \mid 0 \leq k < 0 : k)$
true			
	1	0	$0 = (\Sigma k \mid 0 \leq k < 1 : k)$
true			
	2	1	$1 = (\Sigma k \mid 0 \leq k < 2 : k)$
true			
	3	3	$3 = (\Sigma k \mid 0 \leq k < 3 : k)$
true			
	4	6	$6 = (\Sigma k \mid 0 \leq k < 4 : k)$
false			
Terminate			

What does the loop do?

It sets x to $0+1+2+3$

That is, $x = (\Sigma k \mid 0 \leq k < 4 : k)$

Question

What is the invariant of statement

$$i, x := i+1, x+i$$

That is, what is $P1$ in

$$\{P1\}\; i, x := i+1, x+i \;\{P1\}$$

Answer

$$x = (\Sigma k \mid 0 \leq k < i : k)$$

Check by verifying $P1$ at each step.

A Logical Approach to Discrete Math

Check by correctness proof of:

$P1: \quad x = (\Sigma k \mid 0 \leq k < i : k)$

$\{P1\}\ i,x := i+1, x+i\ \{P1\}$

$$
\begin{array}{ll}
 & wp.(i,x := i+1, x+i).P1 \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & x+i = (\Sigma k \mid 0 \leq k < i+1 : k) \\
= & \langle \text{Split off last term} \rangle \\
 & x+i = (\Sigma k \mid 0 \leq k < i : k) + i \\
= & \langle \text{Assume antecedent} \rangle \\
 & x+i = x+i \\
= & \langle \text{Reflexivity of } = \rangle \\
 & true \quad //
\end{array}
$$

(p.30) **Proof method for DO:** (p.30) is (12.45)

To prove $\{Q\}\ initialization;\ \{P\}$ **do** $B \rightarrow S$ **od** $\{R\}$,
it suffices to prove

(a) P is $true$ before execution of the loop: $\{Q\}\ initialization;\ \{P\}$,

(b) P is a loop invariant: $\{P \wedge B\}\ S\ \{P\}$,

(c) Execution of the loop terminates, and

(d) R holds upon termination: $P \wedge \neg B \Rightarrow R$.

(p.31) **False guard:** **do** $false \rightarrow S$ **od** = *skip*

The multiplication algorithm

(12.42) $\{Q:\ 0 \le n\}$

$i,p := 0,0;$

$\{P:\ 0 \le i \le n \ \wedge\ p = i \cdot x\}$

$\textbf{do}\ i \ne n \to i,p := i+1, p+x\ \textbf{od}$

$\{R:\ p = n \cdot x\}$

<u>Multiplication algorithm proof checklist</u>

(a) Prove $0 \le n \Rightarrow wp.(i,p := 0,0).P$

(b) Prove $P \wedge (i \ne n) \Rightarrow wp.(i,p := i+1, p+x).P$

(c) Prove the loop terminates.

(d) Prove $P \wedge \neg(i \ne n) \Rightarrow p = n \cdot x$

Multiplication algorithm

$P: \quad 0 \leq i \leq n \wedge p = i \cdot x$

(a) Prove $0 \leq n \Rightarrow wp.(i, p := 0, 0).P$

$$
\begin{array}{ll}
 & wp.(i, p := 0, 0).P \\
= & \quad \langle \text{(p.18) and t.s.} \rangle \\
 & 0 \leq 0 \leq n \wedge 0 = 0 \cdot x \\
= & \quad \langle \text{Math} \rangle \\
 & 0 \leq 0 \leq n \wedge true \\
= & \quad \langle \text{(3.39) Identity of } \wedge \rangle \\
 & 0 \leq 0 \leq n \\
= & \quad \langle \text{Conjunctive meaning of } \leq \rangle \\
 & 0 \leq 0 \wedge 0 \leq n \\
= & \quad \langle \text{Assume antecedent } 0 \leq n, \text{(3.39)} \rangle \\
 & true \quad //
\end{array}
$$

Multiplication algorithm

$P: \quad 0 \leq i \leq n \wedge p = i \cdot x$

(b) Prove $P \wedge (i \neq n) \Rightarrow wp.(i, p := i+1, p+x).P$

$$
\begin{array}{cl}
 & wp.(i, p := i+1, p+x).P \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & 0 \leq i+1 \leq n \wedge p+x = (i+1) \cdot x \\
= & \langle \text{Conjunctive meaning, math} \rangle \\
 & 0 \leq i+1 \wedge i+1 \leq n \wedge p = i \cdot x \\
= & \langle \text{Assume conjunct } p = i \cdot x \text{, math} \rangle \\
 & -1 \leq i \wedge i+1 \leq n \\
= & \langle \text{Assume conjunct } 0 \leq i \rangle \\
 & i+1 \leq n \\
= & \langle \text{Assume conjuncts } i \leq n \text{ and } i \neq n \rangle \\
 & true \quad //
\end{array}
$$

Multiplication algorithm

(c) Prove the loop terminates.

By $Q: \ 0 \leq n, \quad n$ cannot be negative.

By the initialization $i, p := 0, 0, \quad$ the initial value of i cannot be greater than n.

Each time through the loop, i increases by 1, and n does not change.

Therefore, i must eventually equal n, $i \neq n$ will be false, and the loop will terminate.

Multiplication algorithm

$P: \;\; 0 \leq i \leq n \wedge p = i \cdot x$

(d) Prove $P \wedge \neg(i \neq n) \Rightarrow p = n \cdot x$

$$
\begin{array}{cl}
 & p = n \cdot x \\
= & \langle \text{Assume conjunct } p = i \cdot x \rangle \\
 & i \cdot x = n \cdot x \\
= & \langle \text{Assume conjunct } \neg(i \neq n) \text{ and double negation} \rangle \\
 & n \cdot x = n \cdot x \\
= & \langle \text{Reflexivity of } = \rangle \\
 & true \quad //
\end{array}
$$

The division algorithm

$$\begin{array}{ll} (12.46) & \{Q:\ b \geq 0\ \wedge\ c > 0\} \\ & q, r := 0, b; \\ & \{\text{invariant } P:\ b = q \cdot c + r\ \wedge 0 \leq r\} \\ & \textbf{do } r \geq c \rightarrow q, r := q+1, r-c \textbf{ od} \\ & \{R:\ b = q \cdot c + r\ \wedge 0 \leq r < c\} \end{array}$$

Division algorithm proof checklist

(a) Prove $b \geq 0 \wedge c > 0 \Rightarrow wp.(q,r := 0,b).P$

(b) Prove $P \wedge (r \geq c) \Rightarrow wp.(q,r := q+1, r-c).P$

(c) Prove the loop terminates.

(d) Prove $P \wedge \neg(r \geq c) \Rightarrow b = q \cdot c + r \wedge 0 \leq r < c$

Division algorithm

$P: \;\; b = q \cdot c + r \wedge 0 \leq r$

(a) Prove $b \geq 0 \wedge c > 0 \Rightarrow wp.(q, r := 0, b).P$

$$
\begin{array}{ll}
 & wp.(q, r := 0, b).P \\
= & \quad \langle \text{(p.18) and t.s.} \rangle \\
 & b = 0 \cdot c + b \wedge 0 \leq b \\
= & \quad \langle \text{Math, (3.39) Identity of } \wedge \rangle \\
 & 0 \leq b \\
= & \quad \langle \text{Assume conjunct } 0 \leq b \rangle \\
 & \mathit{true} \quad //
\end{array}
$$

Division algorithm

$P: \quad b = q \cdot c + r \wedge 0 \leq r$

(b) Prove $P \wedge (r \geq c) \Rightarrow wp.(q, r := q+1, r-c).P$

$$
\begin{array}{ll}
 & wp.(q, r := q+1, r-c).P \\
= & \langle \text{(p.18) and t.s.} \rangle \\
 & b = (q+1) \cdot c + r - c \wedge 0 \leq r - c \\
= & \langle \text{Math} \rangle \\
 & b = q \cdot c + r \wedge c \leq r \\
= & \langle \text{Assume conjuncts } b = q \cdot c + r \text{ and } r \geq c \rangle \\
 & true \quad //
\end{array}
$$

Division algorithm

(c) Prove the loop terminates.

By $Q: \ b \geq 0 \wedge c > 0, \quad c$ must be positive.

Regardless of the initial value of r, each time through the loop
it decreases by c, and c does not change.

Therefore, r must eventually equal be less than c, $r \geq c$ will be false,
and the loop will terminate.

Division algorithm

$P: \quad b = q \cdot c + r \wedge 0 \leq r$

(d) Prove $P \wedge \neg(r \geq c) \Rightarrow b = q \cdot c + r \wedge 0 \leq r < c$

$$
\begin{array}{ll}
 & b = q \cdot c + r \wedge 0 \leq r < c \\
= & \langle \text{Assume conjunct } b = q \cdot c + r \rangle \\
 & 0 \leq r < c \\
= & \langle \text{Conjunctive meaning} \rangle \\
 & 0 \leq r \wedge r < c \\
= & \langle \text{Assume conjunct } 0 \leq r \rangle \\
 & r < c \\
= & \langle \text{Assume conjunct } \neg(r \geq c) \text{ and math} \rangle \\
 & \mathit{true} \quad //
\end{array}
$$