

Specification (schemas)

↳ refinement

Design (schemas)

Algorithm (abstract program)

↳ refinement

Code (guarded commands)

↳ translation

Code (programming language?)

Refinement Calculus

Specification statement

frame : [precondition, postcondition]

Example

available! : [true, *available*! = *free*(θBoxOffice)]

Example

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.1 < f(m') < 0.1 \\ a \leq b \quad , \quad a \leq m' \leq b \end{array} \right]$$

Guarded commands

```
if ... fi
do ... od
begin ... end
;,
```

Abort

```
abort = w : [false, true]
```

Choose

```
choose w = w : [true, true]
```

Skip

```
skip = [true, true]
```

Magic

```
magic = w : [true, false]
```

Assignment

$$x, y : \begin{bmatrix} x = X & x' = X - Y \\ y = Y, & y' = X \end{bmatrix}; \quad x := y - x$$

Refinement

If a program P is correctly refined by another program Q , then we write $P \sqsubseteq Q$; pronounced ' P is refined by Q '.

$$P \sqsubseteq Q \wedge Q \sqsubseteq R \Rightarrow P \sqsubseteq R$$

Strengthen postcondition

If

$$pre \wedge post_2 \Rightarrow post_1$$

then

$$w : [pre, post_1] \sqsubseteq w : [pre, post_2]$$

Example

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.1 < f(m') < 0.1 \\ a \leq b \quad , \quad a \leq m' \leq b \end{array} \right]$$

\sqsubseteq (strengthen postcondition)

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.01 < f(m') < 0.01 \\ a \leq b \quad , \quad a \leq m' \leq b \end{array} \right]$$

Weaken precondition

If

$$pre_1 \Rightarrow pre_2$$

then

$$w : [pre_1, post] \sqsubseteq w : [pre_2, post]$$

Example

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.01 < f(m') < 0.01 \\ a \leq b \quad , \quad a \leq m' \leq b \end{array} \right]$$

\sqsubseteq (weaken precondition)

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.01 < f(m') < 0.01 \\ , \quad a \leq m' \leq b \end{array} \right]$$

Feasibility check

$$pre \Rightarrow \exists x' : X; y' : Y \bullet post$$

Infeasible

$$m : \left[\begin{array}{l} f(a) * f(b) \leq 0 \quad -0.01 < f(m') < 0.01 \\ , \quad a \leq m' \leq b \end{array} \right]$$

Example

Introduce local block

If x does not appear in w , then

$$w : [pre, post] \equiv \begin{array}{l} \text{begin} \\ \quad \text{var } x : T \mid \text{inv}^\bullet \\ \quad w.x : [pre, post] \\ \text{end} \end{array}$$

\equiv (introduce local block)

begin

$$\begin{array}{l} \text{var } x, y, \dots \\ \quad x, y, m : \left[\begin{array}{ll} f(a) * f(b) \leq 0 & -0.1 < f(m') < 0.1 \\ a \leq b & , \quad a \leq m' \leq b \end{array} \right] \\ \text{end} \end{array}$$

Assignment introduction

If

$$pre \Rightarrow post[E/w', x/x']$$

then

$$w, x : [pre, post] \sqsubseteq w := E$$

Skip introduction

If

$$pre \Rightarrow post[w/w']$$

then

$$w : [pre, post] \sqsubseteq \text{skip}$$

Logical constants

Example

```
x,y :  $\begin{bmatrix} x = X & x' = X - Y \\ y = Y, & y' = X \end{bmatrix}; x := y - x$ 
```

```
begin con X • x : [x = X, x' > X]
```

```
begin  
con X • x : [x = X, x' > X]  
end;
```

```
begin  
con X • x : [x = X, x' > X]  
end
```

Scope

```
begin  
con X • x : [x = X, x' > X]; x : [x = X, x' > X]  
end
```

Angelic nondeterminism

```
begin con X • x : [x = X, x' > X] end
```

```
begin con X • x : [true, x' > x] end
```

Introduce logical constant

If

$pre_1 \Rightarrow (\exists C : T \bullet pre_2)$

and C is a fresh name, then

$w : [pre_1, post]$

$$\equiv \left\{ \begin{array}{l} \text{begin} \\ \quad \text{con } C : T \bullet prog \\ \text{end} \end{array} \right\} \subseteq prog$$

Eliminate logical constant

If C occurs nowhere in $prog$, then

$$\left\{ \begin{array}{l} \text{begin} \\ \quad \text{con } C : T \bullet prog \\ \text{end} \end{array} \right\} \subseteq prog$$

C-style constants

```
begin
  var pi : ℝ | pi = 22/7 •
  :
end
```

Sequential composition

```
w,x:[pre, post]
x:[pre, mid]

w,x:[mid[X/x, x/x'], post[X/x]]
```

Sequential composition introduction

$$\frac{w, x : [pre, post]}{\begin{cases} \text{begin} \\ \quad \text{con } X \bullet \\ \quad x : [pre, mid]; \\ w, x : [mid[X/x, x/x'], post[X/x]] \\ \text{end} \end{cases}}$$

Example

$$\frac{x : [true, x' = x + 2] \atop \sqsubseteq (\text{sequential composition introduction})}{\begin{array}{l} \text{con } X \bullet \\ x : [x = X, x' = x + 1]; \\ x : [x = X + 1, x' = X + 2] \end{array}}$$

Swapping the hard way

```
x,y:[true, x' = y \wedge y' = x]
      □ (sequential composition introduction)
con X1, Y1 : ℤ •
x,y:[true, x' = x - y \wedge y' = x];
      □
x,y:[x = X1 - Y1 , x' = Y1 ]
y = X1 , y' = X1
      □
x := x - y
```

Flattened

$x, y : [x = X \wedge y = Y, x = Y \wedge y = X]$

\dagger
 \sqsubseteq (assignment introduction)

$y := x + y$

\dagger

\sqsubseteq (assignment introduction)

$x := y - x$

$x := y - x$

end

$x := y - x$

end

\sqsubseteq

$x := x - y ;$

$y := x + y ;$

$x := y - x$

Simple sequential composition

If the predicates *mid* and *post* make no reference to before variables, then

$w, x : [pre, post]$

$\sqsubseteq x : [pre, mid] ; w, x : [mid[x/x'], post]$

Leading assignment

$w, x : [pre[E/x], post[E/x]]$

$\sqsubseteq x := E ; w, x : [pre, post]$

Following assignment

$w, x : [pre, post]$

$\sqsubseteq w : [pre, post[E/x']] ; x := E$

Conditional introduction

If

$$pre \Rightarrow (G_1 \vee G_2 \vee \dots \vee G_n)$$

then

$$\begin{aligned} & \text{if } G_1 \rightarrow com_1 \\ & \Box G_2 \rightarrow com_2 \\ & \vdots \\ & \Box G_n \rightarrow com_n \\ \text{fi} \quad & w : [pre, post] \quad \sqsubseteq \quad \left\{ \begin{array}{l} \text{if } G_1 \rightarrow w : [G_1 \wedge pre, post] \\ \Box G_2 \rightarrow w : [G_2 \wedge pre, post] \\ \vdots \\ \Box G_n \rightarrow w : [G_n \wedge pre, post] \end{array} \right. \end{aligned}$$

\sqsubseteq (conditional introduction)

$$\begin{aligned} \text{Example} \quad & \text{if } x \leq y \rightarrow x, y : \left[\begin{array}{l} x \leq y \wedge x' = x \wedge y' = y \\ \vee \\ true \end{array} \right] \quad [\lhd] \\ x, y : & \left[\begin{array}{l} x \leq y \wedge x' = x \wedge y' = x \\ \vee \\ x \leq y' \wedge x' = y \wedge y' = y \\ \vee \\ y \leq x \rightarrow x, y : \left[\begin{array}{l} y \leq x \wedge y' = x \\ \vee \\ y \leq x \wedge x' = y \wedge y' = x \end{array} \right] \quad [\dagger] \end{array} \right] \\ \text{fi} \quad & \end{aligned}$$

\sqsubseteq (strengthen postcondition)

$$x, y : [x \leq y, x \leq y \wedge x' = x \wedge y' = y]$$

\sqsubseteq (skip introduction)

skip

$$\begin{aligned} & + \\ & \sqsubseteq \text{ (strengthen postcondition)} \\ & x, y : [y \leq x, y \leq x \wedge x' = y \wedge y' = x] \\ & \sqsubseteq \text{ (assignment introduction)} \\ & x, y := y, x \end{aligned}$$

if $x \leq y \rightarrow \text{skip}$

\square $y \leq x \rightarrow x, y := y, x$

fi

$w, x : [pre, post] \sqsubseteq w : [pre, post[x/x']]$

Contract frame

Expand frame

$w : [pre, post] = w, x : [pre, post \wedge x' = x]$

do $G_1 \rightarrow com_1$
 $\square G_2 \rightarrow com_2$
 \vdots
 $\square G_n \rightarrow com_n$
od

Iteration

Loop introduction

$w : \begin{bmatrix} \neg G[w'/w] \\ inv', inv[w'/w] \end{bmatrix}$

Point at the target

$i : [target \in \text{ran } s, s(i') = target]$

$$\sqsubseteq \begin{cases} \text{do} & G \rightarrow \\ & w : \begin{bmatrix} G & inv[w'/w] \end{bmatrix} \\ \text{od} & \end{cases}$$

```

begin
  conI : 1 .. #\$ •
  i : [ s(I) = target, s(I') = target ]
end

i : [ s(I) = target, s(I') = target ]
      ,   i ≤ I , i' ≤ I
      [ s(I') = target
        s(I) = target
      ]
      ,   i ≤ I , i' ≤ I

```

```

do
  s(i) ≠ target →
  i : [ s(I) = target      s(I') = target
        i ≤ I           i' ≤ I
        [ s(i) ≠ target , 0 ≤ I - i' < I - i ]
  od

```

An integer array

$ar : (1..n) \rightarrow \mathbb{N}$

```

Init = ar : [ true, ran ar' = {0} ]
Init = ar : [ true, ∀ j : 1..n • ar' j = 0 ]
zeroed(i, ar) = ∀ j : 1..i • ar j = 0

```

$ar : [\text{true}, \text{zeroed}(n, ar')]$

\sqsubseteq

$\text{var } j \mid 1 \leq j \leq n + 1 \bullet$

$j, ar : [\text{true}, \text{zeroed}(n, ar')]$

\sqsubseteq (simple sequential composition)
 $j, ar : [\text{true}, \text{zeroed}(j' - 1, ar')];$ [\lhd]
 $j, ar : [\text{zeroed}(j - 1, ar), \text{zeroed}(n, ar')]$ [\dagger]

\sqsubseteq (assignment introduction)
 $j := 1$

\dagger

\sqsubseteq (strengthen postcondition)
 $j, ar : [\text{zeroed}(j - 1, ar), \text{zeroed}(j' - 1, ar') \wedge j' = n + 1]$

\sqsubseteq (following assignment)

$ar : \left[\begin{array}{ll} j \neq n + 1 \\ \text{zeroed}(j - 1, ar) , \text{zeroed}(j, ar') \end{array} \right] ;$ [\lhd]

$j := j + 1$

$j, ar : \left[\begin{array}{ll} j \neq n + 1 & 0 \leq n - j' + 1 < n - j + 1 \\ \text{zeroed}(j - 1, ar) , \text{zeroed}(j' - 1, ar') & \end{array} \right]$

od

\sqsubseteq (assignment introduction)

$ar := ar \oplus \{ j \mapsto 0 \}$

Init

$$\sqsubseteq$$

```

begin
  var j | 1 ≤ j ≤ n + 1 •
  j := 1;
  do j ≠ n + 1 →
    ar := update(ar,j,0);
    j := j + 1
  od
end

```

PROCEDURE Init ;
 BEGIN
 FOR j := 1 TO n DO ar[j] := 0
 END

Base conversion

10011100

$$\begin{aligned}
 & 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + \\
 & 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 \\
 = & 128 + 16 + 8 + 4 \\
 = & 156
 \end{aligned}$$

Recurrence relation

$$\sum_{i=1}^n a_i * \beta^{i-1} = H_{1,n}$$

where

$$\begin{aligned}
 H_{n,n} &= a_n \\
 H_{i,n} &= a_i + \beta * H_{i+1,n}
 \end{aligned}$$

for $i < n$

$d : [\text{true}, d' = H_{1,n}]$

\sqsubseteq

$\text{var } j : 1 .. n \bullet$

$d,j : [\text{true}, d' = H_{1,n}]$

\sqsubseteq (sequential composition introduction)

$d,j : [\text{true}, d' = H_{j,n}] ;$

$d,j : [d = H_{j,n}, d' = H_{1,n}]$

\sqsubseteq (assignment introduction)

$d,j := a_n, n$

\dagger

\sqsubseteq (strengthen postcondition)

$d,j : [d = H_{j,n}, d' = H_{1,n} \wedge j' = 1]$

\sqsubseteq (strengthen postcondition)

$d,j : [d = H_{j,n}, d' = H_{j,n} \wedge j' = 1]$

\sqsubseteq (loop introduction)

do

$j \neq 1 \rightarrow$

$d,j : \left[\begin{array}{ll} j \neq 1 & 0 \leq j' < j \\ d = H_{j,n} & d' = H_{j,n} \end{array} \right]$

od

\sqsubseteq (leading assignment)

$j := j - 1;$

$d,j : \left[\begin{array}{ll} j \neq 0 & 0 \leq j' \leq j \\ d = H_{j+1,n} & d' = H_{j,n} \end{array} \right]$

\sqsubseteq (contract frame)

$d : \left[\begin{array}{ll} j \neq 0 & 0 \leq j \leq j' \\ d = H_{j+1,n} & d' = H_{j,n} \end{array} \right]$

\sqsubseteq (strengthen postcondition)

$d : [j \neq 0 \wedge d = H_{j+1,n}, d' = a_j + \beta * H_{j+1,n}]$

\sqsubseteq (strengthen postcondition)

$d : [j \neq 0 \wedge d = H_{j+1,n}, d' = a_j + \beta * d]$

\sqsubseteq (assignment introduction)

$d := a_j + \beta * d$

```

begin
var j : 1..n •
d,j := an,n;
do j ≠ 1 →
j := j - 1;
d := aj + x * d
od
end

```

```

PROCEDURE Translate ;
BEGIN
d := a[n] ;
FOR j := n DOWNTO 1 DO
d := a[j] + x * d
END

```