[A] → A → Bool

6) a) elem_e_x:xs = (⟨N_i : 0 ≤ i < #xs : xs.i = e⟩ mod 2 == 0)

b) elem e . x:xs = ⟨∀i : 0 ≤ i < #xs : (xs.i = e ∧ par i) ∨ (xs.i ≠ e ∧ par i)⟩

Anotación de Programa.

n predicado en un punto del programa que es sahsfecho por todos los estados posibles en este punto del programa

Obs 1: las anotaciones no son parte de los programas

obs 2: El predicado puede apuntar mas estados que los posibles

corolario: True es siempre una anotación válida

Ejemplo:

```
Const N : Int
Var i, r : Int
{N ≥ 0}                                  ⟶ PRECONDICIÓN
i, r := 0, 0 ;
{i = 0 ∧ r = 0}
do i ≤ N →
{0 ≤ i ≤ N}
  if p.i ⟶ {0 ≤ i ≤ N ∧ p.i}
     r := r + 1
  □ ¬p.i ⟶ {0 ≤ i ≤ N ∧ ¬p.i}
     skip
  fi ;
  i := i + 1
od
```

{más fuerte, más específico}.

{~~i ≥ N~~ / i = N+1 ∧ R = ⟨N_i : 0 ≤ i ≤ N : p.i⟩}

POST CONDICIÓN.

Anotaciones particulares que nos interesan mucho.

$bs = [\,] \wedge sum\ bs = 0$

$\equiv$ { lógica de listas

(4) $max\ \langle Max_{a,as,bs,cs} : x = a \wedge xs = as \# bs \# cs \wedge iga.e.x:xs : \#bs \rangle$

$\equiv$ { anidado, def iga

(4) $max\ \langle Max_{a} : x = a : \langle Max_{as,bs,cs} : xs = as \# bs \# cs \wedge x = e \wedge iga\ e.xs \#bs \rangle$

$\equiv$ { rango unitario

(4) $max\ \langle Max_{as,bs,cs} : xs = as \# bs \# cs \wedge x = e \wedge iga\ e.xs : \#bs \rangle$

$\equiv$ { análisis por casos.

$x = e$	$\neg x = e$
$\equiv$ {neutro $\wedge$, HI}	$=$ {abs (^), rangovacio
$max\ iga\ e\ xs$	$-\infty$

(*) $= \langle Max_{as,bs,cs} : x:xs = as \# bs \# cs \wedge max\ iga.e.x:xs \wedge as = [\,] : \#bs \rangle$

$\equiv$ { leibnitz

$\langle Max_{as,bs,cs} : x:xs = bs \# cs \wedge max\ iga\ e\ x:xs : \#bs \rangle$

HAY QUE MODULARIZAR

$maxiga\ e.xs \mid x = e \doteq m:e.x:xs\ max\ maxiga\ e.xs$

$\mid x \neq e \doteq m.e.x:xs$

Precondicion: Anotación en el punto inicial del programa. (describe el conjunto de posibles estados iniciales)

Postcondicion: Anotación en el punto final del programa. Describe el conjunto de estados finales posibles

Ejemplo: Dado un predicado $P : Int \to Bool$
$N \geq 0$, quiero contar la cantidad de veces que vale p entre 0 y N

Especificación

Const N: Int
Var n: Int
$\{N \geq 0\}$
S
$\{r = \langle N i : 0 \leq i \leq N : p.i \rangle\}$

Ejemplo: Dados m y n con $n > 0$, quiero calcular el cociente y el resto de dividir m por n

Especificación: Const m, n: Int
Var q, r: Int
$\{n > 0\}$
S
$\{m = q * n + r\}$

$q, r := 0, m$ → por sentencia (sentencia del programa) ✓

→ si yo especifico

$[\{m = q \times n + r \wedge 0 \leq r < n\}]$ otra especificación válida.

la post condición no siempre tiene la forma r = algo.

Ejemplo: Dado un arreglo A de N números, calcular la suma de los elementos

Esp: Const N: Int , A: array [0, N) of Num
Var r: Num
{N ≥ 0}
S
{r = ⟨Σ i : 0 ≤ i < N : A.i⟩} → índice

Especificación: Descripción formal del problema a resolver a través de una precondición y una postcondición. Notación:

{P} → precondición (constantes y variables)
S → una letra que simboliza el programa (la incógnita a resolver)
{Q} → postcondición

Terna de Hoare:

La terna de {P} S {Q} (Si tengo esta P, al ejecutar S obtengo Q) se llama terna de Hoare y se interpreta de la siguiente manera: "Dado un estado inicial que satisface P, la ejecución de S termina en un determinado estado final que satisface Q"

Ejemplo:

Var x: Int
{x > 0}
x := x * x
{true}

{x ≠ 100}
x := x * x
{x ≥ 0}

assing sing assing assigning

Derivación: Dada una especificación con pre P y post Q. una derivación es encontrar un programa S tq vale $\{P\}\ S\ \{Q\}$

Demostración: Dados una terna de Hoare, demostrar que vale

Para que una terna no valga, hay que encontrar un contra-ejemplo en donde a partir de p, aplicando S, no se satisfaga Q

$\{true\}$ $x := x * x$ $\{x \geq 8\}$ ✗	$\{x = 100\}$ $x := x * x$ $\{x \geq 8\}$ ✓	$\{x \geq 100\}$ $x := x * x$ $\{x \geq 8\}$

$\{x \geq 3\}$ ✓ el predicado mas debil posible.

$x := x * x$

$\{x \geq 8\}$

Precondicion MAS DÉBIL Dado un programa S y una postcondicion $\{Q\}$ la precondicion mas debil de ~~esta~~ S y Q es el predicado más debil q' satisface p, s, y q. (que el programa sea lo mas util posible). denotamos $P = wp.S.Q$ (weakest precondition)
Todo otro P' que satisface p, s y q, ~~satisface~~ es mas fuerte que P.

Debilidad y Fortaleza de predicados: Dados dos predicados P y q.

- Decimos que P es mas débil que Q, si y solo si $Q \rightarrow P$
- Decimos que P es mas fuerte que Q, si y sólo si $P \rightarrow Q$

Obs 1: False es el predicado mas fuerte que existe
Obs 2: True es el predicado mas débil

1) d

Var r : Int

$[\![\sigma_0 : (r \to 3)]\!]$

do $r \neq 0 \to$

$[\![\sigma_1', r \to 3, r \to 2, r \to 1$

if $r < 0 \to$

$[\![$ ——— $]\!]$

$r := r + 1$

$[\![$ ——— $]\!]$

$[\!]\ r > 0 \to$

$[\![r \to 3, r \to 2, r \to 1$

$r := r - 1$

$[\![r \to 2, r \to 1, r \to 0$

fi

$[\![r \to 2, r \to 1, r \to 0]\!]$

od

$[\![r \to 2, r \to 1, r \to 0$ $]\!]$

2) los 3 hacen las mismas cosas, pero en distinto orden.

b) NO, porque utiliza el resultado de la asignación anterior.

c) NO mas determinismo, y como ninguno satisfacería el programa, no tendría sentido

d)

5)

const N : Nat;

var i, s : Nat;

i, s := 1, 1;

do $i \leq N$

i, s := i + 1, s * i

od

Repaso Anotación de programa.

Anotación de programa
postcondicion - precondicion
Especificación
- declaración de variable y ctte
- Precondición P, post Q

Var - Const
{P}
S
{Q}

Terna de Hoare $\{P\}\{S\}\{Q\}$

Precondicion más débil [weakest precondition]

$\{P\}$
$x := x * x$
$\{x \geq 9\}$

1. $x \geq 3$
2. $x \leq -3$
3. $|x| \geq 3$ ← más débil.
4. $x = 10$
5. $x \geq 100 \wedge x \bmod 2 = 1$

Definición: Dado una sentencia S. y una post-condicion Q, la precondición más débil (weakest condition) de S y Q es el predicado mas débil P tq vale la terna de Hoare. $(\{P\}\ S\ \{Q\})$
Denotamos $P = w.p.\ S.Q$.

Obs 1: $\{w.p.\ S.Q\}\ S\ \{Q\}$

Obs 2: Sup $P' \longrightarrow w.p.\ S.\ Q$
luego $\{P'\}\ S\ \{Q\}$

Obs 3: $\{P'\}\ S\ \{Q\}$
luego $P' \longrightarrow w.p.\ S.Q$

Teorema $\{P\}\ Q\{Q\} = P \longrightarrow w.p.\ S.Q$

Demostracion de programas imperativos

Sup que quiero demostrar $\{P\}\ S\ \{Q\}$
Para P, S y q dados.

Dos formas

1. Usando ternas de Hoare. Demostracion directa de la terna de Hoare.
2. Usando el teorema para pasar a w.p., calular la w.p. y probar que es implicada por P.

SKIP: $\{P\}$ SKIP $\{Q\}$

1. $\{P\}$ SKIP $\{Q\} \equiv P \rightarrow Q$ (cuando P abarca menos posibilidades, osea P es mas fuerte)
   $\{P\}$ SKIP $\{P\}$ ✓
   $\{P\}$ SKIP $\{True\}$ ✓
   $\{x=10\}$ SKIP $\{x \geq 0\}$ ✓

2. $wp.\ SKIP.\ Q \overset{?}{\equiv} Q$

Asignacion :=

reemplazar todas las variables por la expresion

1. $\{P\}\ V_0, \dots, V_n := E_1, \dots E_N\ \{Q\} \equiv P \rightarrow Q(V_1 \leftarrow E_1, V_2 \leftarrow E_2, \dots, V_n \leftarrow E_n)$
2. $wp.(V_1 \dots, V_n := E_1, \dots, E_n.\ Q \equiv Q(V_1 \leftarrow E_1, V_2 \leftarrow E_2, \dots, V_n \leftarrow E_n)$

Ejemplos

$\{x \geq 4\}$ $x := x * x$ $\{x \geq 9\}$

$\{P\}$ $x := x + x$ $\{x \geq 9\}$

1. $x \geq 4$
2. $x \geq 3$
3. $|x| \geq 3$ ← wp

Como puede uno calular sistemáticamente lea wp $|x+x| \geq 3$

↓ aparece la asignacion

$\equiv x + x \geq 9$

Entonces, calculemos wp (2)

$wp\ (x := x * x).\ (x \geq 9)$
$\equiv$ { def de wp para :=}
$x + x \geq 9$
$\equiv$ { algebra
$|x| \geq 3$

13/10

Otro ejemplo $\{P\}$

$x, y := y, x$ SWAP

$\{x = B \wedge y = A\}$

$wp\ (x, y := y, x) \cdot (x = B \wedge y = A\}$

$= \{$ def wp para :=

$y = B \wedge x = A$ ✓

Como se demuestra la terna de Hoare?

$P \rightarrow Q\ (V_1 \leftarrow E_1, V_2 \rightarrow E_2, \ldots, V_n \leftarrow E_n)$.

(1 y 2) son con lo mismo

Concatenación ; (o secuenciación)

1- $\{P\}\ S ; T\ \{Q\} \equiv$ existe un R talque $\{P\}\ S\ \{R\}$ y $\{R\}\ T\ \{Q\}$

(Porque nos falta el final de S y el inicio de T, y deberían ser los mismos)

Hay que demostrar 2 subternas de Hoare.

2 $wp \cdot (S ; T) \cdot Q \equiv$ w.p. S (wp. T. Q) Calcular 2 wp':

w.p. S. R (donde R = wp. T. Q)

Empezamos de abajo para arriba

$\{$w.p. S . wp. T. Q$\}$ → actúa como postcondición

S ;

$\{$w.p. T. Q$\}$ → R (un buen candidato)

T

$\{Q\}$

léase así ↑

Ejemplo; $\{x \geq 4\}$

$x := x * x$; → postcondición más fuerte $\{x \geq 16\} = R$

$x := -x$

$\{x < -3\}$

1 Usando la técnica 1., debo probar

a) $\{x \geq 4\}\ x := x * x\ \{x \geq 16\}$

b) $\{x \geq 16\}\ x := -x\ \{x < -3\}$

a) Debo probar que $x \geq 4 \rightarrow wp(x := x * x)(x \geq 16)$

$\equiv$ { def wp

$x \geq 4 \rightarrow x * x \geq 16$ ✓ vale la implicación

$\equiv$ { álgebra

True.

Técnica 2.

2. Probar que la precondición $\rightarrow$ wp

$x \geq 4 \rightarrow wp(S_1; S_2)(x < -3)$

$\equiv$ { definición de wp. ;

$x \geq 4 \rightarrow wp(\underbrace{x := x * x}_{S_1})(\underbrace{wp(x := -x).(x < -3)}_{S_2.})$ (sentencia, postcondición.)

$\equiv$ { Def wp := (primero la de adentro)

$x \geq 4 \rightarrow wp(x := x * x).(-x < -3)$

$\equiv$ { Def wp

$x \geq 4 \rightarrow -(x * x) < -3$

$\equiv$ { Álgebra

$x \geq 4 \rightarrow x * x > 3$

$\equiv$ { Álgebra.

True.

Ejemplo: $\{x = A \wedge y = B\}$ $x := y$; $y := x$ $\{x = B \wedge y = A\}$ $\Big\}$ NO VALE

$x = B \wedge y = B \leftarrow$

$\{x = A \wedge y = B\}$ $x := y$; $y := x$ $\{x = B \wedge y = B\}$ $\Big\}$ Sirve ✓

Obs: Si A y B no están declaradas como constantes, se llaman Constantes de especificación.

No pueden usarse en el programa, sólo en la especificación

Sirven para guiar los valores que deberían tomar x e y

Otro ej

$\{x = A \wedge y = B\}$

$x = A \wedge y = B \wedge a = A\}$ ← $a := x$; (guardo el valor de x en a, que si es variable de programa) lo cambio sin miedo

$x = B \wedge y = B \wedge a = A\}$ ← $x := y$; → valor guardado.

$x = B \wedge y = A \wedge a = A\}$ $y := a$

$\{x = B \wedge y = A\}$

Con 1 : Probar 3 ternas

Con 2 : Probar 3 wp's anidadas.

Condicional (if)

→ estas son las 3 sentencias.

la terna de Hoare va a depender de cada uno de las ternas de Hoare para q'un if valga, alguna guarda ha de ser true. ↗ al menos 1 guarda vale.

1. $\{P\}$ if $B_1 \to S_1$ $\{Q\}$ $\equiv P \to (B_1 \vee B_2 \vee \dots \vee B_n)$

   □ $B_2 \to S_2$ $\wedge$ $\{P \wedge B_1\}$ S_1 $\{Q\}$

   ⋮ $\{P \wedge B_2\}$ S_2 $\{Q\}$

   □ $B_n \to S_n$ ⋮ ⋮ ⋮

   fi $\{P \wedge B_n\}$ S_n $\{Q\}$

Todas las ramas del if, tienen que garantizarme que voy a llegar a Q.

2 $wp\ (\text{if} \dots \text{fi}) \cdot Q \equiv (B_1 \vee B_2 \vee \dots \vee B_n) \wedge (B_1 \to wp\ S_1 \cdot Q \wedge$
$B_2 \to wp\ S_2 \cdot Q \wedge \dots \wedge$
$B_n \to wp\ S_n\ Q)$

$\sum_{i=0}^{N} i = \frac{N(N+1)}{2}$

6)

```
Const N : Num
var i, n : Num
i, n := 0, 1 ;
do i ≤ N →
   n, i := n + i, i + 1
od
```

```
const N : Num
var n : Num
n := (N * (N+1)) / 2
```

7)

```
Const A : array [0, N) of Int
Var i, p, n : Int
i, p, n := 0, 0, 0 ;
do i < N
   if  A.i > 0 → p := p + 1
   □ A.i < 0 → n := n + 1
   □ A.i = 0 → SKIP
   fi ;
   i := i + 1
od
```

8)

```
Const A : array [0, N) of Int
var i, p, n : Int
i, p, n := 0, 0, 0 ;
do i < N
   if A.i ≥ 0 → p := A.i + p
   □ A.i < 0 → n := A.i + n
   fi
   i := i + 1
od
```

(a, finish, (x 9, Node y 9) 7

$x := 1$

case fn Sent st of
(st', Finish) →
(st', To Exec sent') →

Algoritmos

18/10

Repaso: Demostraciones de programas imperativos
$\{P\}\ S\ \{Q\}$

Dos estrategias para hacer las pruebas:

✓ Trabajar directamente sobre la terna de Hoare

✓ Usar el teorema $\{P\}\ S\ \{Q\} \equiv \{P\} \to wp\ S\ .Q.$

SKIP:

1 - $\{P\}\ skip\ \{Q\} \equiv P \to Q$

2 - $wp.skip.Q \equiv Q$

Asignación:

1 - $\{P\}\ x_1, \dots x_n := E_1, \dots, E_n\ \{Q\} \equiv P \to Q(x_1 \to E_1, \dots x_n \to E_n)$

2. $wp(x_1, \dots, x_n := E_1, \dots, E_n).Q \equiv Q(x_1 \to E_1, \dots, x_n \to E_n)$

Composición

1. $\{P\}\ S\ ;T\ \{Q\} \equiv \exists R$ tq $\{P\}S\{R\}$ y $\{R\}T\{Q\}$

Para probar la terna de Hoare de una comp, hay que probar 2 ternas

2 - $wp\ (S;T).Q \equiv wp.S.(wp.T.Q)$

Condicional (IF)

1 - $\{P\}$ if $B_1 \to S_1$ $\{Q\} \equiv (P \to B_1 \vee B_2 \vee \dots \vee B_n) \wedge$

$\square\ B_2 \to S_2$ $\quad B_1 \to \{P \wedge B_1\}\ S_1\ \{Q\}$

$\vdots$ $\quad \wedge\ B_2 \to \{P \wedge B_2\}\ S_2\ \{Q\}$

$\square\ B_n \to S_n$ $\quad \vdots$

fi $\quad \wedge\ B_n \to \{P \wedge B_n\}\ S_n\ \{Q\}$

(↑ Probar alguna guarda verdadera.)

2 - $wp.($if $B_1 \to S_1).Q \equiv (B_1 \vee B_2 \vee \dots \vee B_n) \wedge (B_1 \to wp.S_1.Q)$

fi $\quad \wedge (B_2 \to wp.S_2.Q)$

$\vdots$

Ejercicio: Verificar que
$① \equiv P \rightarrow wp.(if \ldots fi).Q$

DO

Repetición:

Const N : Int;
Var i, r : Int;
$\{N > 0\}$ → P
$i, r := 0, 0;$
do $i \leq N$ →
  if $p.i \rightarrow r := r+1$
  $\square \neg p.i \rightarrow$ skip
  fi
  $i := i + 1$ $\{I\}$
od
$\{r = \langle N_j : 0 \leq j \leq N : p.j \rangle\}$ → Q

S

$\{i \leq N \wedge r = \langle N_j : 0 \leq j < i : p.j \rangle\}$ I

quiero agregar algo mas fuerte

(porque me el i-ésimo num)

me permite relacionar el ciclo con su post condición.

cuando termino, $i = N+1 \equiv 0 \leq j \leq N$

$0 \leq j <$

Otro.

Ejemplo (Ejercicio 5: Calcular N!)

Especificación:

Const N : Nat;
Var r : Nat
$\{N \geq 0\}$
S
$\{r = N!\}$

$r = 1 \times 2 \times 3 \times \ldots \times N$
$= \langle \Pi_i : 1 \leq i \leq N : i \rangle$

Const N : Nat
Var n, r : Nat
$\{N \geq 0\}$
$n, r := 1, 1;$ ← $\{n \Rightarrow 1 \wedge r \Rightarrow 1\}$
do $n \leq N$ → $\{r = (n-1)!\}$
  $n, r := n+1, r \times n$
  $\{r = (n-1)!\}$
od

variable que acumula resultados

variable que recorre

cuando se cumpla la negación de la guarda, va a ser $r = (N+1)-1!$

$r = N!$

Invariante, no varía con la ejecución del ciclo, tiene que valer al principio y al final del ciclo

Programa equivalente:

① $\{r = (n-1)!\}$
$r := r * n$
② $\{r = n!\}$
$n := n + 1$
③ $\{r = (n-1)!\}$

ciclo	n	r	
1°	4	3·2·1	$r = (n-1)!$
2°	4	4·3·2·1	$n! = r$
3°	5	4·3·2·1	$(n-1)! = r$

Invariante de ciclo:

Es un predicado con las siguientes características:

(A) Vale siempre al principio del cuerpo del ciclo.

Obs a: Debe valer la primera vez que entro al ciclo. ie. la precondicion del ciclo debe implicar a la invariante. $P \to I$

b: Debe valer el resto de las veces que entre al ciclo. Osea debe valer como post-condición del cuerpo del ciclo.

c: En el medio del cuerpo del ciclo el invariante se puede romper (puede no valer como anotación de programa)

(B) Al terminar el ciclo, implica a la postcondición del ciclo.

Terna de Hoare $1.\ \{P\}\ do\ B \to S\ od\ \{Q\} \equiv$ Existe invariante I tal que

1) $P \to I$ (obs a)
2) $I \wedge \neg B \to Q$ (B) (negación de la guarda)
3) $\{I \wedge B\}\ S\ \{I\}$ el cuerpo del ciclo preserva el invariante (A, b)
4) el ciclo termina

Terminación de ciclo

t: Estado → Int la función de cota siempre decrece.

i) $I \wedge B \implies t \geq 0$

ii) $\{I \wedge B \wedge t = T\}\ S\ \{t < T\}$ → variable de especificación (me sirve para "guardar" el valor de t)

es una función que se aplica a las variables.

i) se puede interpretar como que cuando $t<0$ el ciclo termina ya que
$\neg t<0 \rightarrow \neg I \vee \neg B$
{recordemos $P \rightarrow Q \equiv \neg Q \rightarrow P$}
y $\neg I$ no puede ser, luego, $\neg B$

ii) dice que la cota decrece cada vez que ejecuto el cuerpo del ciclo.
Esto es, si fijo el valor de la cota en T, al finalizar el cuerpo del ciclo, valdrá la cota es menor que T.

Ejemplo. función t para N!

$t = -n$ ← porque n crece, $-n$ va a decrecer.
(cumple ii, pero no i

$t = N - n$ → calculo cuanto me falta para llegar al final.

$N - n < 0 \equiv N < n$

si vale esto, vale la negación de la guarda ↑ ↑ negación de la warda

Función de cota para N! $t = N - n$

Repaso: Demostración

de programación imperativa

Repetición: Demostración de la terna de Hoare

$\{P\}$ do $\{S\}$ od $\{Q\} \equiv$ existe invariante I tq

1) $P \rightarrow I$
2) $I \wedge \neg B \rightarrow Q$
3) $\{I \wedge B\}\ S\ \{I\}$
4) existe una función de cota $t: Estado \rightarrow Int$
   i) $I \wedge B \rightarrow t \geq 0$
   ii) $\{I \wedge B \wedge t = T\}\ S\ \{t < T\}$

Const N : Nat;
Var i, r : Nat;

P: $\{N \geq 0\}$

$\{R: i = 0 \wedge r = 0\}$

S_2 { $i, r := 0, 0$;
do $i \leq N \rightarrow$
S_3 { if $p.i \rightarrow r := r+1$ □ $\neg p.i \rightarrow$ skip } S_4 fi ;
$i := i+1$ } S_5
od } S

Q: $\{r = \langle N_j : 0 \leq j \leq N : p.j \rangle\}$

Queremos demostrar $\{P\}\ S\ \{Q\}$ con $S = S_1; S_2$ Sea $R \equiv i = 0 \wedge r = 0$

Debo probar (por ternas de ;)

Ⓐ $\{P\}\ S_1\ \{R\}$ (Asignación)

Ⓑ $\{R\}\ S_2\ \{Q\}$ (Do)

(A) Debo ver que $P \to wp.S_1.R$

Sup P y vemos la w.p

$wp(i,r:=0,0).(i=0 \wedge r=0)$

$= \{$ def w.p $\}$

$R(i \to 0, r \to 0)$

$\equiv \{$ sustitucion $\}$

$0=0 \wedge 0=0$

$\equiv \{$ logica $\}$

True.

(B) i) $P \to I$

ii) $I \wedge \neg B \to Q$

iii) $\{I \wedge B\}\ S\ \{I\}$

En primer lugar, I debe ser lo suficientemente fuerte como para satisfacer ②, osea:

$I \wedge i > N \to r = \langle N j : 0 < j \leq N : P.j \rangle$

$\downarrow$

$\{ i \leq N+1 \wedge r = \langle N j : 0 \leq j < i : P.j \rangle : I \}$ $\{R : i=0, r=0\}$.

$< N+1$

② Queda como ejercicio

Faltan ① y ③. Veamos con $R \to I$

I

$\equiv \{$ def I $\}$

$i \leq N+1 \wedge r = \langle N j : 0 \leq j < i : P.j \rangle$

$\equiv \{$ Hip $\}$

$0 \leq N+1 \wedge 0 = \langle N j : 0 \leq j < 0 : 0.Pj \rangle$

$\equiv \{$ rango vacio

$0 \leq N+1 \wedge 0 = 0$

$\{$ Hip $N \geq 0$

True ∎

Vamos con ③ $\{I \wedge B\}\ S_3\ \{I\}$ con $S_3 = S_4 ; S_5$ usando wp debo ver que.

Ⓒ $\{I \wedge B\}\ S_4\ \{\overbrace{wp.S_5.I}^{R}\}$ (*)

Ⓓ $\{\underbrace{wp.S_5.I}_{R}\}\ S_5\ \{I\}$ si $R = wp.S_5.I$, tengo garantizada Ⓓ

(*) Por terna de Hoare del ; debo dar R tq

$\{I \wedge B\}\ S_4\ \{R\}$ y

$\{R\}\ S_5\ \{I\}$

Si elijo $R = w.p.\ S_5 . I$ queda Ⓒ $\{I \wedge B\}\ S_4\ \{wp.S_5.I\}$ → conviene post condición débil, porque wp vale.

Ⓓ $\{wp.S_5.I\}\ S_5\ \{I\}$

Ⓓ vale por def de w.p. Falta probar Ⓒ

Veamos $wp.S_5\ I$:

$w.p.S_5\ .\ I$

$\equiv$ {def wp :=}

$I\ (i := i+1)$

$\equiv$ {sustitución $i \to i+1$}

$i+1 \leq N+1 \wedge r = \langle N\, j : 0 \leq j < i+1 : p.j \rangle$

Ⓒ es $\{I \wedge B\}\ S_4\ \{I\ (i \to i+1)\}$

Con S_4 = if $B_1 \to S_6$

$\square \neg B_1 \to S_7$

fi

Usando terna del if, debo probar $B = i \leq N$ $B_1 = p.i$

Ⓔ $I \wedge B \to B_1 \vee \neg B_1$ → tercero excluido (asumo $I \wedge B$ y pruebo si alguna guarda vale)

Ⓕ $\{(I \wedge B) \wedge B_1\}\ S_6\ \{I\ (i \to i+1)\}$ (precondición del if)

Ⓖ $\{(I \wedge B) \wedge \neg B_1\}\ S_7\ \{I\ (i \to i+1)\}$

Veamos (F)

$$\{I \wedge B \wedge B_1\}\ r := r+1\ \{I\ (i \to i+1)\}$$

Supongamos $I \wedge B \wedge B_1$ y veamos wp.

$wp\ (r := r+1).\ I\ (i \to i+1)$

$\equiv$ { def wp }

$I\ (i \to i+1)(r \to r+1)$

$\equiv$ { sustitucion.

$i+1 \leq N+1 \wedge r+1 = \langle N j : 0 \leq j < i+1 : p.j \rangle$

true (por B)

$\equiv$ { por B $i+1 \leq N+1$ y logica

$r+1 = \langle N j : 0 \leq j < i \vee j = i : p.j \rangle$

$\equiv$ { particion de rango }

$r+1 = \langle N j : 0 \leq j < i : p.j \rangle + \langle N j : i = j : p.j \rangle$

$\equiv$ { Hipotesis I y rango unitario y B_1

$r+1 = r+1$ ■

Hipótesis

$I : i \leq N+1 \wedge r = \langle N j : 0 \leq j < i : p.j \rangle$

$B : i \leq N$

$B_1 : p.i$

Def de conteo

$\langle \Sigma j : j = i \wedge p.j : 1 \rangle$

$\equiv$ { Leibniz

$\langle \Sigma j : j = i \wedge p.i : 1 \rangle$

$\equiv$ { $p.i \equiv$ true (por B_1), Rango unitario

$\langle \Sigma j : j = i : 1 \rangle$

$\equiv$ { Rango unitario

1

(G) $\{I \wedge B \wedge \neg B_1\}$ skip $\{I\ (i \to i+1)\}$

O sea ver que $\{I \wedge B \wedge \neg B_1\}_P \to \{I\ (i \to i+1)\}_Q$ Sup P y probamos Q

$I\ (i \to i+1)$

$\equiv$ { def I y sust

$i+1 \leq N+1 \wedge r = \langle N j : 0 \leq j < i+1 : p.j \rangle$

$\equiv$ { por Hipotesis, mismos pasos que en (F)

$r = r + \langle \Sigma j : j = i \wedge p.i : 1 \rangle$

$\equiv$ { Rango vacio.

$r = r + 0$ ■

Hipotesis

I :

B :

$\neg B_1$:

SÓLO FALTA EL 4 del do

4) Exist t t_q

1) $I \wedge B \to t \geq 0$ ✓

2) $\{I \wedge B \wedge t = T\}\ S_3\ \{t < T\}$ ✓

vale $t \geq 0$

$t = N - i$ cuando i crece, la diferencia se achica.

1) $wp\ (x := x+y).(x \neq 6 \wedge y = 5) \equiv (6 = x+y \wedge y = 5)$

$\Leftarrow$ {Leibniz}

$6 = x+5 \wedge y = 5$

$\boxed{1 = x} \quad \wedge\ y = 5$

b) $wp.\ x := 8.(x = 8)$

$\equiv$ {def wp}

$8 = 8 \equiv \boxed{true}$ ✓

c) $wp.(x := 8).(x = 7)$

$\equiv$ {def wp.}

$8 = 7 \equiv$ {false}

d) $wp.(x, y := y, x).\{x = B \wedge y = A\}$

$\equiv$ {def wp}

$\{y = B \wedge x = A\}$ WP

e) $wp(a, x := x, y\ ;\ y := a).\{X = B \wedge y = A\}$

$\equiv$ {def wp}

$wp.a, x := x, y.(\underline{wp.(y := a).\{x = B \wedge y = A\}})$

$\equiv$ {def wp :=}

$wp.(a, x := x, y).(x = B \wedge a = A)$

$\equiv$ {def wp :=}

$\{y = B \wedge x = A\}$ wp!

f) $wp.\ \text{if } x \geq y \to x := 0 \ \square\ x \leq y \to x := 2 \cdot (x = 0 \vee x = 2) \wedge y = 1$

$= \{ \text{def wp}$

(A) $(x \geq y \vee x \leq y) \wedge (x \geq y \to wp.\ x := 0 \cdot (x = 0 \vee x = 2) \wedge y = 1) \wedge$ (B)

$\to x \leq y \to wp.\ x := 2 \cdot \{(x = 0 \vee x = 2) \wedge y = 1\}$ (C)

A ≡ True

(B) Sup $x \geq y$ $\quad wp.\ x := 0 \cdot (x \neq 0 \vee x = 2) \wedge y = 1$

$= \{ \text{def wp}$

$(0 = 0) \vee (2 = 2) \wedge y = 1$

$\text{True} \wedge y = 1 \equiv \boxed{y = 1}$ ←

$x \geq y \to y = 1$

(C) Sup $x \leq y \to wp\ x := 2 \cdot (x = 0 \vee x = 2) \wedge y = 1$

$= \{ \text{def wp}$

$2 = 0 \vee 2 = 2 \wedge y = 1$

$\text{True} \wedge y = 1$

$x \leq y \to \boxed{y = 1}$ ←

$x \geq y \Rightarrow y = 1 \wedge x \leq y \Rightarrow y = 1$

$a \to b \wedge b \to c$

$a \vee b \to c$

$x \geq y \vee x \leq y \to y = 1$

$\text{True} \to y = 1$

$\boxed{y = 1}$

Heh True this Gertrudis

25/10

Repaso Ejemplo de demostración de programa completo.

Const N : Nat
var i, r : Nat
$\{N \geq 0\} \leftarrow P$

$$S \begin{cases} S_1 \; [\, i, r := 0, 0 \,] \longrightarrow \{R : i = 0 \wedge r = 0\} \\ S_2 \; \text{do } i \leq N \xrightarrow{\text{kB}} \longrightarrow \{I : r = \langle N_j : 0 \leq j < i : p.j \rangle \wedge i \leq N\} \\ \quad S_4 \; [\, \text{if } p.i \to r := r+1 \;\square\; \neg p.i \to \text{SKIP} \,\text{fi}\,; \{R'\}; \\ \quad i := i+1 \,] S_5 \\ \text{od} \end{cases} S_3$$

$\longrightarrow \{r = \langle N_i : 0 \leq i \leq N : p.i \rangle\} \leftarrow Q$

Quiero probar $\{P\} S_1; S_2 \{Q\}$ Dar R tq
Ⓐ $\{P\}\ S_1\ \{R\}$ $(P \to wp.S_1.R)$
Ⓑ $\{R\}\ S_2\ \{Q\}$ (Invariante)

Pienso el resultado parcial a cada momento (es parecido a la post condición)

Para probar Ⓑ debo ver que
Ⓒ $R \to I$ ⟶ (Probar con $0..I3i$)
Ⓓ $I \wedge \neg(i \leq N) \to Q$
Ⓔ $\{I \wedge i \leq N\}\ S_3\ \{I\}$
Ⓕ El ciclo termina

Para probar Ⓔ debo dar R' tq Ⓖ $\{I \wedge i \leq N\}\ S_4\ \{R'\}$
Ⓗ $\{R'\}\ S_5\ \{I\}$ ✓ (vale por def de wp)

Elijo R' como $wp.S_5.I. \equiv I(i \to i+1)$

Ⓖ Para probar G debo ver:
Ⓘ $I \wedge i \leq N \to pi \vee \neg p.i$ ✓ (vale por tercero excluido)
Ⓙ $\{pi \wedge I \wedge i \leq N\}\ r := r+1\ \{R'\}$ ✓ (ver wp)
Ⓚ $\{\neg p.i \wedge I \wedge i \leq N\}$ SKIP $\{R'\}$ ✓ (ver que pre → post)

Prueba de ③

$\{p.i \wedge I \wedge i \leq N\}\ r := r+1\ \{R'\}$.

es lo mismo probar

$p.i \wedge I \wedge i \leq N \Rightarrow wp\ (r := r+1).R'$

Luego, suponemos $(p.i \wedge I \wedge i \leq N)$ y veamos la w.p.

$wp.(r := r+1).R'$

$\equiv$ {def R'

$wp.r := r+1.\ I(i \to i+1) \equiv wp\ r := r+1.(wp.S_3.I)$

$i := i+1$

$\equiv$ {def wp :=

$wp.(r := r+1).I(i \to i+1)$

{def wp :=

$I(i \to i+1)(r \to r+1)$

$\equiv$ {sustitución

$r+1 = \langle Nj : 0 \leq j < i+1 : p.j \rangle \wedge i+1 \leq N+1$

↗ vale por hipótesis

$\equiv$ {part. de rango

$r+1 = \langle Nj : 0 \leq j < i : p.j \rangle + \langle Nj : j = i : p.j \rangle$

$\equiv$ {hipótesis I

$r+1 = r + \langle Nj : j = i : p.j \rangle$

$\equiv$ {rango unitario de cuerpo

$r+1 = r + (p.i \to 1 \ \square\ \neg p.i \to 0)$

$\equiv$ {por hipótesis, vale p.i

$r+1 = r+1$ ✓ True.

Función de cota. Dar t : Estado → Int tq i) $I \wedge B \to t \geq 0$

ii) $\{I \wedge B \wedge t = T\}\ S_3\ \{t < T\}$

Sea $t \doteq N - i$

Veamos 1) Sup $I \wedge B$

$t \geq 0$

$\equiv$ {def t.

$N - i \geq 0$

$\equiv$ {Algebra

$N \geq i$

$\equiv$ {HP(B) True

Ahora ii) $I \wedge B \wedge t = T \longrightarrow wp.S_3.t < T$

Sup $I \wedge B \wedge t = t$ vemos la wp.

$wp\ S_3.\ t < T$

$\equiv$ { def wp t

$wp.S_3\ (N-i < T)$

$\equiv$ { def wp para ;

$wp.S_4.(wp.S_5.(N-i<T))$

$\equiv$ { def wp S_5 }

$wp.S_4.N-(i+1)<T$

$\equiv$ { wp if.

$(p.i \vee \neg p.i) \wedge (p.i \Rightarrow wp(r := r+1).(N-(i+1)<T))$ ①

$\wedge (\neg p.i \Rightarrow wp.skip.(N-(i+1)<P)$ ②

TNW

$\equiv$ { def wp. para := y skip }

$p.i \Rightarrow N-(i+1)<T \wedge$ $\quad A \Rightarrow B \wedge$

$\neg p.i \Rightarrow (N-(i+1)<T)$ $\quad \neg A \rightarrow B \equiv B$

$\equiv$ { lógica

$N-(i+1)<T$

$\equiv$ { hipótesis $t = T$

$N-(i+1) < t$

$\equiv$ { def $\overline{t}$

$N-(i+1) < N-i$

$\equiv$ { algebra (resto $N-i$ a ambos lados)

$-1 < 0$ V TRUE.

DERIVACIÓN DE PROGRAMAS.

Imperativos.

Dada una especificación:

$\{P\}$ ⟶ Dada

S ⟶ Incógnita

$\{Q\}$ ⟶ Dada

encontrar un programa S que la satisfaga

¿Qué tipo de sentencia debe ser S? ⟶ Creatividad.

1. ¿es skip? $P \Rightarrow Q$
2. ¿es una asignación?

Ejemplos. (Ej 3a práctico 1)

Var x, y : Nat

$\{True\}$

$S \longrightarrow x, y := E, F$

$\{x = y+1\}$

S va a ser de la forma $x, y := E, F$

Sup la pre True y veamos la WP.

$wp.\ x, y := E, F\ .\ x = y+1$

$\equiv$ { def wp :=

$E = F+1$ (*)

$\equiv$ { elijo $E = y+1$, $F = y$

$y+1 = y+1$

Resultado $x, y := y+1, y$

O sea $x := y+1$

Alternativas.

(*) = { elijo

$E = 1$ $F = 0$

$1 = 0+1$

Resultado $x, y := 1, 0$

Otro ejemplo

Const N: Nat

Var i, r : Nat

$\{N \geq 0\}$

S

$\{r = \langle N j : 0 \leq i < i : P.j \rangle \wedge i \leq N+1\}$

Supongamos $N \geq 0$ y veamos la wp

$wp.(r, i := E, F).I$

$\equiv$ {def wp

$E = \langle Nj : 0 \leq j < F : P.j \rangle \wedge F \leq N+1$

$\equiv$ {Elijo $F = 0$}

$E = \langle Nj : 0 \leq j < 0 : P.j \rangle \wedge 0 \leq N+1$

$\equiv$ {Rango vacío.

$E = 0 \wedge 0 \leq N+1$

$\equiv$ {Elijo $E = 0$}

$0 = 0 \wedge 0 \leq N+1$

$\equiv$ {Hipótesis $N \geq 0$

True.

Otro ejemplo:

$\{true\}$

$x, y := x+1, E$

$\{x = y+1\}$

Sup la pre y veamos la wp.

$wp\ (x, y := x+1, E).(x = y+1)$

$\equiv$ {def wp

$x+1 = E+1$

$\equiv$ {Algebra

$x = E$

$\equiv$ {Elijo $E = x$}

True.

Otro ejemplo:

$P: \{x = q * y + r\}$

$q, r := q+1, E$

$Q: \{x = q * y + r\}$

Supongamos P y veamos la wp.

$wp.(q, r := q+1, E).Q$

$\equiv$ {def wp}

$x = (q+1) * y + E$

$\equiv$ {Algebra

$x = qy + y + E$

$\equiv$ {Algebra

$E = x - qy - y$

$\equiv$ {Elijo $E = x - qy - y$} Por hipótesis P.

Podemos obtener algo más sencillo.

$E = qy + r - qy - y$

$\equiv$ {Algebra

$E = r - y$.

$\equiv$ {Elijo $E = r - y$.

True ∎

Resultado.

$P: \{x = q \times y + r\}$

$q, r := q+1, r-y$

$\{x = q \times y + r\}$

Resumen

1 - Primero plantear la asignación como incognita

Obs 1 : si no sé cuál variables hay que asignar, probar asignar todas

Obs 2 : Algunas Incognitas pueden ser previamente conocidas por creatividad

2 - Hacer la demostración de la forma, Eligiendo para las incognitas expresiones programables de manera que la demostración sea true (correcta)

Clase que viene veremos mas.

Guia 4 :

2) a)

$$S \begin{cases} \text{if } x \geq 1 \longrightarrow x := x+1 \\ \square\ x \leq 1 \longrightarrow x := x-1 \\ \text{fi} \end{cases}$$

$\{True\}$ wp $\{x \neq 1\}$

$wp.\ S.\ x \neq 1$

$(x \geq 1 \vee x \leq 1) \wedge (x \geq 1 \longrightarrow wp.\ x := x+1 .\ x \neq 1) \wedge (x \leq 1 \longrightarrow wp.\ x := x-1 .\ x \neq 1)$

$(x \geq 1 \vee x \leq 1)$ = True

① $B_1 \longrightarrow wp.\ S_1.\ Q$

② $B_2 \longrightarrow wp.\ S_2.\ Q$

① Sup $x \geq 1$ y pruebo $wp\ x := x+1 .\ x \neq 1$

$wp\ x := x+1 .\ x \neq 1$

$=$ { def wp := }

$x + 1 \neq 1$

$=$ { algebra }

$x \neq 0$ por hip. $x \geq 1$ = True

$x \geq 1 \longrightarrow x \neq 0$

② $x \leq 1 \longrightarrow wp.\ x := x-1 .\ x \neq 1$. Sup $x \leq 1$, veamos la wp

$wp.\ x := x-1 .\ x \neq 1$

$=$ { def wp }

$x - 1 \neq 1$

$=$ { algebra }

$x \neq 2$ por hipotesis, $x \leq 1$, esto es verdadero

b)

1a $x \geq y \rightarrow wp.\ x := x - y.\ Q \quad x \geq 0 \wedge y \geq 0$

$\equiv$ {def wp.

$x - y \geq 0 \wedge y \geq 0$

$\equiv$ {lógica}

$x \geq y \wedge y \geq 0$

$\equiv$ {Hipótesis}

$\boxed{y \geq 0}$

Conclusión.

$wp.\ S_2\ .\ Q \equiv \boxed{y \geq 0 \wedge x \geq 0}$

1b. $x < y \rightarrow wp.\ y := y - x.\ x \geq 0 \wedge y \geq 0$

$\equiv$ {def wp :=

$x \geq 0 \wedge y - x \geq 0$

$\equiv$ {lógica, hipótesis

$\boxed{x \geq 0}$

Seguimos con ② $wp.\ S_1\ .\ (wp.\ S_2\ .\ Q)\ .\ \equiv wp.\ S_1\ .\ y \geq 0 \wedge x \geq 0$

$\equiv$ {def wp :=

$y * y \geq 0 \wedge x * x \geq 0$

$\equiv$ {álgebra

true.

d) $\{true\}$

if $\neg a \vee b \rightarrow a := \neg a$

$[]\ a \vee \neg b \rightarrow b := \neg b$

fi

$\{a \vee b\}$

Quero probar

$\equiv true \rightarrow ((\neg a \vee b) \vee (a \vee \neg b)) \wedge$

① $\{(\neg a \vee b) \wedge true\}\ a := \neg a\ \{a \vee b\} \wedge$

② $\{a \vee \neg b \wedge true\}\ b := \neg b\ \{a \vee b\}$

① $\{(\neg a \vee b)\}\ a := \neg a\ \{a \vee b\}$

$\equiv$ {def wp :=

$\neg a \vee b$ (por Hip $\equiv$ true)

② lo mismo

e) $\{N \geq 0\}$

$x := 0$;

do $x \neq N \rightarrow x := x+1$

od

$\{x = N\}$

Hay que ver si $\exists$ I tq

① $\{P\}\ S_1\ \{R\} \wedge \{R\}\ S_2\ \{Q\}$ ②

Sup $R = \{x = 0\}$ y probamos

① $\{N \geq 0\}\ x := 0\ \{x = 0\}$

② $\{x = 0\}$ do $x \neq N \rightarrow x := x+1$ od $\{x = N\}$

$\exists$ I tq

$2a\ P \Rightarrow I \ ,\ (\{I \wedge \neg B\}\ S\ \{I\}) \wedge (I \wedge \neg E \rightarrow Q)$

Sup $I = 0 \leq x \leq N$

2a) $0 \leq x \rightarrow 0 \leq x \leq N$

0

2b $\{0 \leq x \leq N \wedge x = N\}\ x := x+1\ \{x = N\}$

=$\{$wp

$x+1 = N$

27/10

Repaso Derivación de programas imperativos.

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

Dado

Tipos de sentencia • SKIP? $p \rightarrow q$?

• Asignación? ver $p \rightarrow wp.(v_1, \ldots v_n := E_1, \ldots, E_n).Q$

3a $\{True\}\ x, y := x+1, E\ \{y = x+1\}$

Sup precondición y vemos la wp.

wp. $x, y := x+1, E\ \{y = x+1\}$

=$\{$def wp asignación

$E = x+2$

=$\{$ Elyo $\boxed{E = x+2}$ $\}$.

True ■

otro Ejemplo : {const A: array [0, N) of Num
{r = ⟨Σ i : 0≤i<N : A.i⟩ ∧ N > 0}
S
{r = ⟨Σ i : 0≤i<N : A.i⟩ / N}

→ r := r/N plantear x := E como operación

- Condicional: Puede aparecer al intentar derivar una asignación y encontrar que necesito hacer un análisis por casos.

Ejemplo: (ej 11 practico 3)

```
Const N: Int , A: Array[0, N) of Num
Var i,c : Int ;
  i,c := 0,0;
  do i<N →
    {c = ⟨Nj : 0≤j<i : A.j>0⟩} ← I ∧ B
      S1 ——————→ (?)
    {c = ⟨Nj : 0≤j<i : A.j>0⟩}
  od
```

S1 es por lo menos una asignación i,c := E,F. (creatividad)
Mejor, sabemos que E = i+1. Tenemos que ver i,c := i+1, F

Sup I ∧ B y vemos la wp.

wp . i,c := i+1, f . I
≡ {def wp :=
F = ⟨Nj : 0≤j<i+1 : A.j>0⟩
≡ {partición de rango.
F = ⟨Nj : 0≤j<i : A.j>0⟩ + ⟨Nj : j=i : A.j>0⟩
≡ {por hipótesis.
F = c + ⟨Nj : j=i : A.j>0⟩
≡ {rango unitario}
F = c + (A.i>0 → 1
□ ¬A.i>0 → 0)

* Caso A.i > 0 B_1
≡ {aplico el caso
F = c+1
≡ {elijo F=c+1
Trivial → lo importante es llegar a true.

* Caso ¬(A.i > 0) ≡ A.i ≤ 0 B_2
≡ {aplico el caso.
F = c + 0
≡ {elijo F=c
Trivial.

27/10

Luego, el programa queda de la siguiente forma

$\{I \wedge B\}$

if $A.i > 0 \longrightarrow$

$i, c := i+1, c+1$

□ $A.i \leq 0 \longrightarrow$

$i, c := i+1, c$

fi

$\{I\}$

Obs: Al derivar una asignación $V_1, \dots, V_n = E_1, \dots, E_m$ si llego a que debo elegir $E_j = V_j$ puedo omitir esa asignación

Ejercicio: Derivar el ciclo p/el problema de contar p entre 0 y N.

Falso ⟶ algo ≡ true
algo ⟶ True ≡ True

- Derivación de ciclos

Recordemos que $\{P\}$ do B → S $\{Q\}$ ≡ ① ∃ I tal que

a) $P \rightarrow I$ (*) Débil

b) $I \wedge \neg B \rightarrow Q$ Fuerte

c) $\{I \wedge B\}\ S\ \{I\}$ (el invariante se preserva) (*) Débil

d) el ciclo termina.

Al derivar un ciclo, lo primero a determinar es el invariante y la guarda B

(*) a más débil el invariante, más fácil de demostrar

la condición que me pide fortaleza para el invariante es (b).

(b) es lo primero que hay que mirar para elegir I y B

Técnicas para determinar Invariante (Cap 19 del libro) en adelante.

1ra técnica: Tomar el elementos de una conjunción $\{P\}\ S\ \{Q\}$

Supongamos $Q \equiv Q_1 \wedge Q_2$

luego si elijo $I \equiv Q_1$ y $B \equiv \neg Q_2$ (Siempre que Q_2 sea programable)

Esto garantiza (B)

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

$$Q_1 \wedge \neg(\neg Q_2) \longrightarrow Q_1 \wedge Q_2$$

$$Q_1 \wedge Q_2 \longrightarrow Q_1 \wedge Q_2$$

Ahora debería verse $P \longrightarrow Q_1 \wedge \neg(\neg Q_2)$

Siempre hay que tener en mente el programa.

Ejemplo Algoritmo de la división.

Var x, y : Int

Var q, r : Int.

P: $\{y > 0 \wedge x \geq 0\}$

S

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

$0 \leq r \wedge r < y$ $\quad B \equiv \neg(r \geq y)$

Aplicando la técnica de tomar términos de conjunción mas un poco de creatividad, elegimos

$B \equiv r \geq y$

$I \equiv x = q * y + r \wedge 0 \leq r$ ←

El programa quedaría de la siguiente forma

$\{P\}$

S_0 ; $\{I\}$ $\{I \wedge B\}\ S_1\ \{Q\}$

do $r \geq y \rightarrow S_1$ od.

$\{Q\}$

Debemos encontrar S_0 y S_1. [La condición intermedia entre S_0 y el do es I ya que la precondición del do, que es I) ya que de esta manera garantizo la condición 1 $(P \rightarrow I)$]

la precondición del do.

Queda encontrar S_0 tq $\{P\}\ S_0\ \{I\}$ y S_1 tq $\{I \wedge B\}\ S_1\ \{I\}$

↑ Inicialización ↑ cuerpo del ciclo.

Derivamos S_0 Va a ser de la forma

$q, r := E, F$

Sup P y usamos la wp

$wp.(q, r \rightarrow E, F).I$

$\equiv$ {def wp}

$x = E * y + F \wedge 0 \leq F$

(0, x, x)

$\equiv$ {Elijo $E = 0 \wedge F = x$

$x = 0 * y + x \wedge 0 \leq x$

$\equiv$ {Aritmética y Hipótesis}

True.

Conclusión $q, r := 0, x$

Falta derivar S_1 tq $\{I \wedge B\}\ S_1\ \{I\}$

Obs: Si elijo skip, la terna vale pero el ciclo no termina. Debo asignar algo, hacer algo que cambie el estado.

Veamos con una asignación. Va a ser de la forma

$$q, r := E, F$$

Sup $I \wedge B$ y veamos la wp:

$wp.(q, r := E, F).I$

$\equiv$ { def wp.

$x = E * y + F \wedge 0 \leq F$ ⟵ lo mismo de la inicialización.

$\equiv$ { Sabemos que $r \geq y$, entonces $r - y \geq 0$. Elijo $F = r - y$ y estamos seguro que $0 \leq F$ se cumple }

$x = E * y + (r - y) \wedge 0 \leq r - y$

$\equiv$ { ~~Despejamos~~ Despejamos E

$E = \dfrac{x - r + y}{y}$ ~~(división de $\mathbb{R}$, no vale)~~ NO SIRVE

$\equiv$ { ~~factor común y~~

$x = (E - 1) * y + r$ ⟵ alternativa más rápida.

$\equiv$ { Hip. Invariante

$q * y + r = E * y + r - y$

$\equiv$ { Aritmética

$q * y = E * y - y$

$q * y + y = E * y$

$(q + 1) * y = E * y$ como sé $y > 0$

$q + 1 = E$

$\equiv$ { Elijo $E = q + 1$ ⟵

True.

El programa queda

```
Var x, y, q, r : Num
{x ≥ 0 ∧ y > 0}
q, r := 0, x
{I}
do r ≥ y →
        {I ∧ ¬B}
        q, r := q + 1, r - y
```

Hay que demostrar que termina.

Función de cota →

4) $\exists$ $t: (st) \to Num$ tal que

1) $I \wedge B \to T \geq 0$

2) $\{I \wedge B \wedge t = T\}\ S_1\ \{t < T\}$

Obs Sirve tratar de reescribir la guarda B de la forma $t \geq 0$

$t = r - y$ ✓

$t = r$

---

1 e) $\{N \geq 0\}$

$x := 0;$

do $x \neq N \to x := x + 1$

od

$\{x = N\}$

Debo ver $\{P\}\ S_1; S_2\ \{Q\}$

luego hay que ver si $\exists R$ tq

1 $\{P\}\ S_1\ \{R\} \wedge^2 \{R\}\ S_2\ \{Q\}$

Sup. $R = \{x = 0\}$

1) $\{N \geq 0\}\ x := 0\ \{x = 0\}$

veamos si $P \to wp.\ x := 0\ \{x = 0\}$ Sup. P

$\equiv$ {def wp}

$0 = 0$ True. $P \to$ True ✓

2) $\{R\}\ S_2\ \{Q\}$ Para probar el de. $\exists I$ tq

Sup $I = \{x \leq N\}$ $\{0 \leq x \leq N\}$

1) $R \to I$

2) $I \wedge \neg B \to Q$

3) $\{I \wedge B\}\ S_2\ \{I\}$

4) Cota t_0

1) $x = 0 \to 0 \leq 0 \leq N$ ✓

2) $0 \leq x \leq N \wedge \neg(x \neq N) \to \{x = N\}$

$\{x = N\} \to \{x = N\}$ ✓

3) $\{0 \leq x \leq N \wedge x \neq N\}\ x := x + 1\ \{0 \leq x \leq N\}$

Probamos wp

wp. $x := x + 1$. $0 \leq x \leq N$

$\equiv$ {def wp}

$0 \leq x + 1 \leq N$ $(x > N \vee x < N) \wedge (0 \leq x \leq N) \to 0 \leq x + 1 \leq N$

4) $t = N - x$

f) {True}
r := N ;
do r ≠ 0 →
if r < 0 → r := r + 1
[] r > 0 → r := r - 1
fi
od
{r = 0}

{True} r := N ; do r ≠ 0 if [] fi od {r = 0}

Derivacion de ciclos 1/10

Recordemos demostracion : {P} do B → S od {Q}.
≡ Existe un invariante tq
1) P ⇒ I
2) I ∧ ¬B ⇒ Q ⊛
3) {I ∧ B} S {I}
4) Existe cota.

Obs Para dar un invariante, es mejor empezar por 2. (necesita mas fortaleza)
1 y 3 cuanto más débiles, mejor

Derivacion : Tenemos esp {P} S {Q}
1. Dar invariante I y guarda B talque I ∧ ¬B → Q
2- Plantear el esquema del programa.

{P}
S0 → Inicialización
{I}
do B →
{I ∧ B}
S1
{I}
od
{Q}

3- Derivar S0 con la especificacion
{P} S0 {I} → asignacion
4 Derivar S1 con especificacion
{I ∧ B} S1 {I}
↑ Lo mejor es intentar asignacion
↓ si yo pusiera skip, el programa sería correcto, pero el ciclo no terminaría

5 - Dar funcion de cota.

Obs 1 : En ④ no se puede poner skip (hay que cambiar el estado)

Técnicas para Determinar Invariantes

- Técnica 1: Tomar elementos de una conjunción (de la postcondicion)
  Tomamos una parte como I, y la otra como $\neg B$. Luego, garantizamos $I \wedge \neg B \rightarrow Q$ (Q es lo más difícil)

Ejemplo
const x, y : Int
Var q, r : Int
$\{P: x \geq 0 \wedge y > 0\}$
S
$\{Q: x = qy + r \wedge 0 \leq r < y\}$
$\underbrace{x = qy + r \wedge 0 \leq r}_{I} \wedge \underbrace{r < y}_{\neg B}$.

Otro ejemplo con la misma técnica (está en el libro!) Búsqueda lineal
Sup. que tenemos un predicado $f: Int \rightarrow Bool$ y sabemos que algún número > 0 satisface esta condición. Quiero encontrar el más chico que satisface f.

Especificación Var r : Int
$\rightarrow \{P: \langle \exists i : 0 \leq i : f.i \rangle\}$ → ponemos como pre, nuestro único dato.
S
$\rightarrow \{r = \underbrace{\langle Min\ i : 0 \leq i \wedge f.i : i \rangle}_{f.e}\}$ → nunca va a dar rango vacío porque sabemos que alguno lo satisface.
↳ r es el número tq " "

Reescribimos la post. $Q \equiv \underbrace{0 \leq r \wedge f.r}_{\text{r satisface la condicion}} \wedge \underbrace{\langle \forall i : 0 \leq i < r : \neg f.i \rangle}_{\text{todo número que este antes de r no lo satisface}}$

Elijo $I \equiv \langle \forall i : 0 \leq i < r : \neg f.i \rangle \wedge 0 \leq r$
$B = \neg f.r$

Esquema del programa $\{P\}$
S_0 ;
$\{I\}$
do $\neg f.r \rightarrow S_1$ od.
$\{Q\}$

(Intuitivamente)
r := 0
do $\neg f.r \rightarrow$
r := r + 1
od

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

• Inicialización: Demar S_0 tal que $\{P\}\ S_0\ \{I\}$
Probamos $r := E$. Supongamos P y veamos ~~wp.r:=E.I~~

$$\begin{aligned} & wp.r := E \bullet I \\ \equiv & \{ \text{Def wp.} \\ & \langle \forall i : 0 \le i < E : \neg f.i \rangle \wedge 0 \le E \\ \equiv & \{ \text{Elijo } E = 0 \\ & \text{True} \ \blacksquare \end{aligned}$$

• Cuerpo del ciclo: Encontrar S_1 tq $\{I \wedge B\}\ S_1\ \{I\}$ Probamos
$r := E$. sup $I \wedge B$ $wp.r := t.I$
$\equiv$ {def wp. → a veces se plantea $r+F$ (como que tiene que avanzar)
$\langle \forall i : 0 \le i < E : \neg f.i \rangle \wedge 0 \le E$ hace falta creatividad.
$\equiv$ {Elijo $E = r+1$
$\langle \forall i : 0 \le i < r+1 : \neg f.i \rangle \wedge 0 \le r+1$
$\equiv$ {partición de rango}
$\langle \forall i : 0 \le i < r : \neg f.i \rangle \wedge \langle \forall i : i = r : \neg f.i \rangle \wedge 0 \le r+1$
$\equiv$ {Hip I
True $\wedge$ $\langle \forall i : i = r : \neg f.i \rangle \wedge 0 \le r+1$
{Rango unitario
$\neg f.r \wedge 0 \le r+1$
$\equiv$ {vale por HB
$0 \le r+1$
$\equiv$ {vale por el Invariante
True.

• Función de cota: Como es algo teórico, puedo decir que $X \overset{\geq 0}{tq}\ f.X$.
luego, $t = X - r$
$r < X$ (si estoy dentro del ciclo).

Ejemplo en un parcial Cota: la cota es $t = X - r$
i) $I \wedge B \Rightarrow t \geq 0$
$t \geq 0$ ya que si vale I, entonces $r < X$, y luego
$0 < X - r = t$
ii) la cota se achica en el cuerpo del ciclo ya que "X" no cambia y

2da técnica

- Reemplazo de constante por variables. (libro / cap 19)

Ejemplo Suma de los elementos de un arreglo

Especificación

Const N: Int;
A: array [0,N) of Int
Var r: Int
{P: N ≥ 0}
S
{Q: r = ⟨Σ i : 0 ≤ i < N : A.i⟩}

- Usando reemplazo de constantes por variables **Elijo** $I = (\Sigma i : 0 \leq i < n : A.i) \wedge n \leq N$ (nueva variable)

para tener los índices definidos.

$B \stackrel{?}{\equiv} n \neq N$ (cuando lo niegue, va a implicar la postcondición)

Queremos $(I \wedge \neg B) \rightarrow Q$ ✓

Esquema del programa.

```
{P}
S0 ⟶ n,r := E,F        → 0 por el invariante, por rango acotado
{I}
n,r := 0,T
do n≠N →                → porque es índice, entonces se asume que empieza en 0 y aumenta 1
{I ∧ B}
  S1 ⟶ n,r := E,F
       n,r := n+1,T     → se supone que tengo guardada la suma hasta tal elemento r + A.n
 {I}
od
{Q}
```

Ejercicio: Escribir la inicialización y el cuerpo del ciclo.

cota: N − n

¿Qué pasa si $I \equiv r = \langle \Sigma i : n < i < N : A.i \rangle \wedge n \geq 0$

$B = n \neq 0$ ya que de esa forma $I \wedge \neg B \rightarrow Q$

cota $t = n$

```
n,r := N,0
do n≠0 →
  n,r := n-1, r + A.(n-1)
od
```

Ejemplo Exponenciacion.

Const x, y : Int.
Var r : Int
{P : x > 0 ∧ y ≥ 0}
S
{Q : r = x^y}

A ojo. (va a tener un do)

r, n := 1, 0 ;
do n < y → r, n := r*x, n+1
od.

↑ invariante
$r = x^n$ ↘ es un reemplazo de constantes por variables
∧ n ≤ y

Usando reemplazo de constantes por variables, elijo I ≡ $r = x^n \wedge x \leq y$ (para la cota.)
B ≡ n ≠ y

↘ cuando usamos esta técnica, solemos poner const ≠ var para garantizar I ∧ ¬B → Q.

Inicialización {P} S_0 {I} S_0 es de la forma r, n := E, F.
Ejercicio. 1 0

Cuerpo del ciclo S_1 va a ser de la forma r, n := E, n+1
Sup I ∧ B y veamos la wp. {I ∧ B} S_1 {I}

wp (r, n := E, n+1)
≡ {def wp
$E = x^{n+1} \wedge n+1 \leq y$
≡ {aritmética
$E = x^n * x \wedge n+1 \leq y$
≡ {Hipótesis Inv.
$E = r * x \wedge n+1 \leq y$
≡ {Elijo E = r * x
n+1 ≤ y
≡ {Hipótesis n ≤ y, B ≡ n ≠ y} → n < y, luego n+1 ≤ y.
True

Cota = t = y − n

Repaso. Derivación de ciclos

Técnicas para determinar el invariante

1er técnica: tomar términos de una conjunción

2da técnica: reemplazo de constantes por variables.

Resumen! de la segunda técnica

$Q \equiv r = \langle \Sigma i : 0 \leq i < N : A.i \rangle$

1. Elegir una de las constantes que aparecen en la postcondición (por ejemplo N).
2. Declarar una nueva variable (por ejemplo "n") y elegir el invariante (I) como Q reemplazando la constante por la variable y B como el predicado que $n \neq N$ (decir que la constante es distinta de la variable)
3. Si hace falta, fortalecer el invariante con restricciones para la nueva variable (En esta sumatoria, para restringir los valores del arreglo (no pasarme), y para la función de cota) — demostrar

En el ejemplo dado. $I \equiv \langle \Sigma i : 0 \leq i < n : A.i \rangle \wedge n \leq N$

Obs: La elección de la constante a reemplazar es creativa

* Puede no funcionar (ej. $Q \equiv r = \langle \Sigma i : 0 \leq i < N : A.i \rangle$, falla reemplazar la base)
* Pueden obtenerse diferentes algoritmos que calculan lo mismo pero de distinta manera (ej $Q \equiv \langle \Sigma i : 0 \leq i < N : A.i \rangle$ empezaría desde el final.

3er Técnica: Fortalecimiento de Invariantes.

Ejemplo: Ej 12 (Práctico 5)

"sumant" para determinar si un elemento del arreglo A es igual a la suma de los anteriores

```
Const N : Int ; Const A : Array(0, N] of Int
Var r : Bool
{P: N ≥ 0}
S
{Q ≡ ⟨∃i : 0 ≤ i < N : A.i = ⟨Σj : 0 ≤ j < i : A.j⟩⟩}
{Q ≡ r = ⟨∃i : 0 ≤ i < N : A.i = ⟨Σj : 0 ≤ j < i : A.j⟩⟩}
```

Usamos reemplazo de constantes x var.
(reemplazamos N!)

```
r, n := False, 0
do n ≠ N
    r, n := E, n+1 → {ver si A.n es igual a la suma de los anteriores}
od
```

S, m

```
S, m := 0, 0 ;
do m ≠ n →
    S, m := S + A.m, m+1
od ;
r, n := r ∨ (A.n = S), n+1
```

od

Intuitivamente

optimiza

```
r, n := False, 0 ;
do n ≠ N ∧ (¬r) →
    S, n := 0, 0 ;
    do m ≠ n →
        S, m := S + A.m, m+1
    od ;
    r, n := (A.n = S), n+1 ;
od
```

para asegurarnos que no se pasa del arreglo

→ Reemplazo de ctte por variables

$I \equiv r = \langle \exists i : 0 \le i < n : A.i = \langle \Sigma j : 0 \le j < i : A.j \rangle \rangle \wedge n \le N$

$B \equiv n \neq N$

Inicialización: n, r := 0, false

cuerpo del ciclo: r, n := E, n+1

Sup. $I \wedge B$ y veamos la wp de Q → ⊛

wp r, n := E, n+1.

Esquema del programa (para aclarar cosas)

```
{P}
So;
{I}
do n≠N →
    {I∧B}
    S1
    {I}
od
{Q}
```

$P \Rightarrow wp.S.I$

Tenemos que ver esta terna de hoare.

$I \wedge B \Rightarrow wp.S.I$

continuamos ...

$wp.\ r,n := E, n+1$

$\equiv$ {def wp

$E = \langle \exists i : 0 \le i < n+1 : A.i = \langle \Sigma j : 0 \le j < i : A.j \rangle \rangle \wedge n+1 \le N$

$0 \le i < n+1 \equiv 0 \le i < n \vee i = n$

$\equiv$ {partición de rango y rango unitario

$E \equiv \left(\langle \exists i : 0 \le i < n : A.i = \langle \Sigma j : 0 \le j < i : A.j \rangle \rangle \vee An = \langle \Sigma j : 0 \le j < n : A.j \rangle \right) \wedge n+1 \le N$

$\equiv$ { Hip I

$E \equiv (r \vee A.n = \langle \Sigma j : 0 \le j < n : A.j \rangle) \wedge \underline{n+1 \le N}$

$\equiv$ { por hip $n \le N$, $n \neq N$

$E \equiv (r \vee A.n = \underbrace{\langle \Sigma j : 0 \le j < n : A.j \rangle}_{\text{no es programable}})$

Me trabo! No puedo encontrar E programable. Si hubiera tenido como Hipótesis $S = \langle \Sigma j : 0 \le j < n : A.j \rangle$

"Fortalezco el Invariante" = (**)

$I' \equiv r = \langle \exists i : 0 \le i < n : A.i = \langle \Sigma j : 0 \le j < i : A.j \rangle \rangle \wedge S = \langle \Sigma j : 0 \le j < n : A.j \rangle \wedge n \le N$

(agrego una nueva declaración de variable s: Int

(Ahora en el esquema donde decía $I \to I'$

Debo hacer de nuevo la inicialización $r, S, n := E, F, 0$

(llegar a $E = False$ $F = 0$)

1) $\varphi \Rightarrow wp.S_1.I$

2) $I' \wedge \neg B \Rightarrow Q$ ——— no hay que pensar porque es mejor $F \rightarrow$ algo (True)

Cuerpo del ciclo $\{I' \wedge B\}$

$r,s,n := E,F,n+1$

$\{I'\}$

Sup $I' \wedge B$ y veamos la wp.

$wp(r,s,n := E,F,n+1).I'$

$\equiv$ {def wp}

$E = (**) \wedge F = \langle \Sigma j : 0 \le j < n+1 : A.j \rangle \wedge n+1 \le N$

$\equiv$ {mismos pasos}

$(E = r \vee A.n = \langle \Sigma j : 0 \le j < n : A.j \rangle) \wedge (F = \langle \Sigma j : 0 \le j < n+1 : A.j \rangle)$

$\equiv$ {Partición de rango y rango unitario}

$(E \equiv r \vee A.n = \langle \Sigma j : 0 \le j < n : A.j \rangle) \; F = \langle \Sigma j : 0 \le j < n : A.j \rangle + A.n$

$\equiv$ {Hip nueva}

$E \equiv r \vee (A.n = s) \wedge F = \langle \Sigma j : 0 \le j < n : A.j \rangle + A.n$

$\equiv$ {Hip nueva (de nuevo)}

$(E \equiv r \vee (A.n = s)) \wedge F = s + A.n$

$\equiv$ Elijo $\boxed{E = r \vee A.n = s \quad y \quad F = s + A.n}$

True

$t = ?$

Programa:

```
Const N: Int ; A: array[0, N) of Int
Var r: Bool ; s, n: Int;
{N ≥ 0}
r, s, n := false, 0, 0 ;
do n ≠ N →
    r, s, n := r ∨ (A.n = s), s + A.n, n+1
od                                  ↓ cota
{Q}
```

$\boxed{t = N - n}$ no cambia la cota

i) $I \wedge B \rightarrow t \ge 0$ vale pues I dice $n \le N \rightarrow 0 \le N - n = t$

ii) Decrece pues n crece

Resumen 1- Derivado el cuerpo del ciclo me trabo porque hay una parte de las incógnitas que no puedo elegir de tal forma que sea programable

2- Declaro una nueva variable y fortalezco el invariante con el predicado que dice que la nueva variable vale esa parte que no es programable, fortalezco sólo la parte NO programable.

$I' \equiv I \wedge (S = \text{algo no programable})$

3- Derivo de nuevo la inicialización y el cuerpo del ciclo, ambas considerando la nueva variable

Observaciones: la derivación anterior del ciclo se tira (ya no sirve)

- la nueva derivación del ciclo sale por los mismos pasos que la vieja pero esta vez no me trabo porque tengo una hipótesis nueva
- Puede suceder que me trabe de nuevo y haga falta fortalecer de nuevo

4) a)

Var a, b, r: Num

{True}

S

{r = a → a < b
b → b < a}

True

if a > b → r := b
□ a ≤ b → r := a

if

b)

Var a, r: Num

{True}

S ⟶

{r = |a|}

if a > 0 → r := a
□ a ≤ 0 → r := a · (-1)

Repaso: Fortalecimiento de invariantes

Invariantes

1- Me trabo derivando el cuerpo de un ciclo (no puedo llegar a la hipótesis porque hay algo de más)
2. Declaro una nueva variable y fortalezco el invariante.
- $(I \wedge \neg B) \Rightarrow Q$ (sigue valiendo, no hace falta probarla).
3- Debo derivar de nuevo la inicialización y también debo derivar de nuevo el cuerpo del ciclo (con la hipótesis adicional)

Problemas de bordes:

Ejemplo (Ej 16 prácticos) Segmento de suma máxima de un arreglo A

Const N: Int ; A: Array [0,N) of Int
Var r: Int
$\{P\}$ N ≥ 0
S
$\{Q = r = \langle \text{Max } p,q : 0 \le p < q \le N : \text{sum } p\ q \rangle\}$
$|[\text{sum } p\ q = \langle \Sigma i : p \le i < q : A.i \rangle]$

- Voy a derivar un ciclo usando la técnica de reemplazo de constantes por variables con N ← n

$I \doteq r = \langle \text{Max } p,q : 0 \le p \le q \le n : \text{sum } p\ q \rangle \wedge (n \le N)$ → para no pasarme de los elementos del arreglo.

$B \doteq n \neq N$ $(I \wedge \neg B \rightarrow Q)$ ✓

[Seguiría la inicialización, pero vamos a hacer cuerpo del ciclo por si hay que fortalecer, y no hacer trabajo de más]

r, n := E, 0
do n ≠ N
  r, n := E, n + 1
od

Cuerpo del ciclo $\{I \wedge B\}\ r;A := E, n+1\ \{I\}$.

Sup $I \wedge B$ y veamos la wp.

$wp.(r, n := n+1).I$

$\equiv$ { por def wp.

$E = \langle Max\ p,q : 0 \le p \le q \le n+1 : sum\ p.q \rangle \wedge n+1 \le N$

$\equiv$ { Hip $n \le N$ y $n \ne N$, $n+1 \le N \equiv true$ }

$E = \langle Max\ p,q : 0 \le p \le q \le n+1 : sup\ p.q \rangle$

$\equiv$ { particion de rango, por lógica.

$E = \langle Max\ p,q : 0 \le p \le q \wedge ((q \le n) \vee (q = n+1)) : sum\ p.q \rangle$

$\equiv$ { distributiva, y particion de rango.

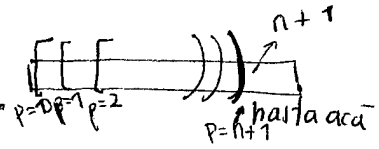
$E = \langle Max, p,q : 0 \le p \le q \wedge q \le n : sum\ p.q \rangle\ max\ \langle Max\ pq : 0 \le p \le q \wedge q = n : sum\ p.q \rangle$

$\equiv$ { Hip }

$E = r\ max\ \langle Max\ p,q : 0 \le p \le q \ne n+1 : sum\ p.q \rangle$

$\equiv$ { Anidado y rango unitario }

$E = r\ max\ \langle Max\ p : 0 \le p \le n+1 : sum.p.(n+1) \rangle$



Si yo fortalezco ahora, tengo que

$I' = I \wedge S = \langle Max\ p : 0 \le p \le n+1 : sum.p.(n+1) \rangle$

$\textcircled{4} \wedge n \le N$ $\qquad$ $\langle \Sigma i : p \le i < n+1 : A.i \rangle$

i puede llegar a valer N, otra A.N (no tiene sentido, undefine

Tengo un problema de borde: Si $n = N$, me queda $sum.p.(N+1) = \langle \Sigma i : p \le i < N+1 : A.i \rangle$ y me estoy yendo de los bordes del arreglo.

Todavía no hay que fortalecer, hay que seguir trabajando.

$\equiv$ { nueva partición de rango.

$E = r\ max\ \langle Max\ p : 0 \le p \le n : sum.p.n+1 \rangle\ max\ \langle Max : p = n+1 : sum.p.n+1 \rangle$

$\equiv$ { rango unitario

$E = * \langle Max * \rangle * \ max\ sum(n+1).(n+1)$

$\equiv$ { por def de sum, queda rango vacío

$E = r\ max\ \langle Max\ p : 0 \le p \le n : sum.p.(n+1) \rangle\ max\ 0$

$\equiv$ { def definición de sum

$E = r\ max\ \langle Max\ p : 0 \le p \le n : \langle \Sigma i : p \le i < n+1 : A.i \rangle \rangle\ max\ 0$

$\equiv$ { partición rango, rango unitario $\longrightarrow p \le i < n \vee i = n$

$E = \langle Max\ p : 0 \le p \le n : \langle \Sigma i : p \le i < n : A.i \rangle + A.n \rangle$

$\equiv$ { def sum }

$E = \langle Max\ p : 0 \le p \le n : sum\ p\ n + A.n \rangle\ max\ r\ max\ 0$

acá esto está bien definido pues no es invariante y $B = n \neq N$, sino.

Si fortalezco tengo

$I' \equiv I \wedge S = \langle Max\ p : 0 \leq p \leq n : sum.p.n + A.n \rangle$

[veamos $I \wedge \neg B \Rightarrow Q$

$I' \neg B$

$\equiv r = \langle \cdots \rangle \wedge n \leq N \wedge S = \langle \cdots \rangle \wedge n = N$ ¡NO DEFINIDO!

Leibnitz $\equiv r = \langle \cdots \rangle \wedge n = N \wedge S = \langle Max\ p : 0 \leq p \leq N : sum.p.N + A.N \rangle$]

$\equiv$ { distributividad suma con max

$E = (\langle Max\ p : 0 \leq p < n : sum.p.n \rangle + A.n)\ max\ r\ max\ 0$

Ahora sí puedo fortalecer. Tengo

$I' \equiv I \wedge S = \langle Max\ p : 0 \leq p \leq n : sum.p.n \rangle$

Demuestro el nuevo ciclo.

$\{I' \wedge B\}\ r, s, n := E, F, n+1\ \{I'\}$

Sup $I' \wedge B$ y vemos la wp.

$wp.(r, s, n := E, F, n+1).I'$

$E = \langle max \cdots \rangle \wedge n+1 \leq N \wedge F = \langle Max\ p : 0 \leq p \leq n+1 : sum.p.(n+1) \rangle$

$\equiv$ { Por mismos pasos

$E = (\langle Max\ p : 0 \leq p \leq n : sum.p.n \rangle + A.n)\ max\ r\ max\ 0 \wedge F = \langle Max : \cdots \rangle$

$\equiv$ { por hipótesis nueva,

$E = (S + A.n)\ max\ r\ max\ 0 \wedge F = \langle Max\ p : 0 \leq p \leq n+1 : sum.(n+1) \rangle$

$\equiv$ { Ejercicio }

$E = (S + A.n)\ max\ r\ max\ 0 \wedge F = 0\ max\ (S + A.n)$

$\equiv$ { elijo E y F convenientemente } True!

Nos quedó

$r, s, n := 0, 0, 0$

do $n \neq N$

$r, s, n := (S + A.n)\ max\ r\ max\ 0, (S + A.n)\ max\ 0, n+1$

od.

Inicialización (Queda como ejercicio probarla)

$r, s, n := 0, 0, 0$

$t = N - n$

Ejercicio Máxima distancia entre dos elementos de un arreglo A

Const N: Int, A: array [0,N) of Int

Var r: Int

$I \equiv r = \langle Max\ p,q : 0 \le p < q < n : A.p - A.q \rangle \wedge n \le N$

$B \equiv n \neq N$

Cuerpo: sup $I \wedge B$ Veamos wp

$wp.(r,n := t, n+1).I$

$\equiv$ { def wp

$E = \langle Max\ p,q : 0 \le p < q < n+1 : A.p - A.q \rangle$

$\equiv$ { parto rango $\longrightarrow 0 \le p < q \wedge (q < n \vee q = n)$

$E = \langle Max\ p,q : 0 \le p < q < n : A.p - A.q \rangle\ max\ \langle Max\ p,q : 0 \le p < q = n : A.p - A.q \rangle$

$\equiv$ { Hipr, anidado y rango unitario

$E = r\ max\ \langle Max\ p : 0 \le p < n : A.p - A.n \rangle$ → Acá no se puede fortalecer por el An que se me va del rango

$\equiv$ { Distrib + con max (para sacarlo afuera)

$E = r\ max (\langle Max\ p : 0 \le p < n : A.p \rangle - A.n)$

fortalecimiento. Ahora

$I' \equiv I \wedge s = \langle Max\ p : 0 \le p < n : A.p \rangle$ el número mas grande que he visto hasta ahora.

Ejercicio: Terminar de derivar.

do

r, s, n := r max (s - A.n), s max A.n, n+1

od.

Inicialización

r, s, n := A.0 - A.1, A.0 max A.1, 2 // no funciona 0,0,0.

↳ hacemos el proceso en los 2 primeros elementos

Básicamente NO FORTALECER SI HAY UN A.n. (o Sum n,) o lo que sea con n. (n=N)

6) S calcula la suma de los elementos de un array
si cumple las anotaciones pero se traba pues no avanza el índice

b) No pues avanza antes de sumar luego, no calcula la primera posición

c) Sí funciona y sí cumple

d) Calcula si hay algún número igual a e. Sí funciona.

7) a) Sí, pues no cambia el valor de i ni a

b) Sí, pues vale en todo el ciclo

c) Sí

d) No pues no cumple I ∧ N con el mismo, estaría contando hasta i

e) Sí el primer elemento a ≥ 0, el invariante vale, sino, no

8) const X, Y : Int
var x, y : Int;
{x > 0 ∧ Y > 0 ∧ x = X ∧ y = Y}
S
{x = min X, Y}

Ver en el libro

→

10) 1) exp. x. y = (y = 0 → 1
□ y ≠ 0 → x * (exp x. (y-1))
)

{x = X ∧ y = Y ∧ x ≥ 0 ∧ y ≥ 0}
S
{r = exp x.y} ←

I = {y ≥ 0 ∧ r * x^y = X^Y} ←

Veamos I ∧ ¬B → Q (para elegir la guarda)
y ≥ 0 ∧ r * x^y = X^Y → r = exp X . Y
= X^Y

Cota por hip
X^Y * x^y = X^Y
si elijo y = 0, sigo sin ver.

I = y ≥ 0 ∧ r * x^y = X^Y
B = y ≠ 0
Cuerpo del ciclo
do y ≠ 0 →
r, x, y := E, F, y-1
od
Veamos hay que probar la terna {I ∧ B} S {I}. U
Veamos por wp
{(y ≥ 0 ∧ r * x^y = X^Y) ∧ y ≠ 0} r, x, y := E, F, y-1 {y ≥ 0 ∧ r * x^y = X^Y}
= {def wp
(y-1 ≥ 0) ∧ E * F^(y-1) = X^Y
= {por hipótesis y ≠ 0, y-1 ≥ 0 ≡ true
E * F^(y-1) = X^Y
= {hip x^y : r

NO SE COMO SEGUIR HACIENDO

do y ≠ 0 →
r, x, y := r * x, x, y-1
od

Repaso: Problemas de borde al fortalecer invariantes

Ej 1 $Q: r = \langle Max\ pq : 0 \le p < q < N : A.p - A.q \rangle$

$I: r = \langle Max\ pq : 0 \le p < q < n : A.p - A.q \rangle \wedge n \le N$

$B: n \neq N$

Ciclo: $Sup.\ I \wedge B.\ wp(r, n := E, n+1).\ I:$

$E = r\ max\ \langle Max\ p : 0 \le p < n : A.p - A.n \rangle$ → esto es un PROBLEMA porque hacé que me salga del arreglo sia invariante. y NO se puede fortalecer.

Según lo visto, llegamos a algo programable para E y tenemos que recurrir a fortalecimiento de invariantes.

Entonces, aplicamos distributividad de suma con max y fortalecemos solo esa parte

$E = r\ max\ \langle Max\ p : 0 \le p < n : A.p \rangle - A.n$

fortalecemos esto, ya que estamos seguros que p nunca llega a N

ESTO ESTA MAL

$I' \equiv I \wedge S \equiv \langle Max\ p : 0 \le p < n : A.p - A.n \rangle$

cuerpo ciclo. $Sup\ I' \wedge B$

$wp.(r, s, n := E, F, n+1).\ I'$

$= \{$ muchos pasos

$E = r\ max\ s \wedge F = \langle Max\ p : 0 \le p < n+1 : A.p - A.n \rangle$ (se destapa un problema)

$= \{$ partición de rango. $0 \le p < n$ y rango unitario. $p = n$

$E = r\ max\ S \wedge F = \langle Max\ p : 0 \le p < n : A.p - A.(n+1) \rangle\ max\ (A.n - A.(n+1))$

ESTO no es hipótesis. NO hay forma de llegar a la hipótesis.

Ni se ocurra fortalecer de nuevo. va a ser lo mismo pero con $n+2$.

→ Resumen de la técnica: Al derivar el cuerpo del ciclo con arreglos, veo que necesito fortalecer el invariante.

2. También veo que SI fortalezco de inmediato usando la parte no programable voy a agregar al invariante algo que puede no estar definido ($A.n$, indexar $A.i$ con i fuera del $[0, N)$)

3. Debo trabajar más sobre la expresión a fortalecer (sobre la derivación)

para fortalecer con una expresion que quede bien definida

4. Una vez que trabajé la expresión y me deshago del pedacito que molestaba, fortalezco y sigo normalmente. (TODO DE NUEVO)

ESTADOS INTERMEDIOS (no está en el libro pero parece ser importante)

Ejercicio 9: Calcular varianza de los elementos de un arreglo

Especificación. Const N : Int, Array [0,N) of Real

Var r : Real

{ N ≥ 0 }

S

$\{ r = \frac{\langle \Sigma i : 0 \leq i < N : (A.i - X)^2 \rangle}{N}$

$\land X = \frac{\langle \Sigma i : 0 \leq i < N : A.i \rangle}{N}$ (N si; N no) → si yo usara la técnica de reemplazo de const x var, reemplazo la N del rango pero no la del término.

Uno podría primero resolver el problema del promedio, guardarlo en alguna variable para usarlo en el programa. Primero calculamos el promedio.

Necesitamos n para recorrer y s para acumular el resultado

$s, n := 0, 0$

do $n \neq N \rightarrow$

$s, n := s + A.n, n+1$

od; ← proc. promedio

$\{ s = \Sigma i : 0 \leq i < N : A.i \rangle \}$ ← estado intermedio.

$s := \frac{s}{N}$

$\{ s = X \}$ ⟶ precondición.

Ahora tenemos que recorrer de nuevo el programa inicializamos de nuevo pero no cambiamos S ← $r, n := 0, 0$

do $n \neq N \rightarrow$

$r, n := r + (A.n - s)(A.n - s), n+1$

od

$\{ r = \langle \Sigma i : 0 \leq i < N : (A.i - X)^2 \rangle$ ⟶ estos estados intermedios me ayudan a programar como por partes.

$r := \frac{r}{N}$

{Q}

(estos estados: otra forma de derivar con máxima creatividad.)

Ejemplo (más difícil porque el otro era más fácil ..... no.)

Dado un arreglo de numeros A y un numero X, calcular la posicion de X en el arreglo. Si no está, dar -1 como resultado de posicion.

ordenarlo mejor → Const N: Int, Array [0, N) of int; Var r: Int; const X: Int;

{N ≥ 0}

S

{ X = A.r ∨ (r = -1 ∧ ⟨∀i : 0 ≤ i < N : A.i ≠ X⟩) } → mejor forma pero quiere agregar más

{(0 ≤ r < N ∧ X = A.r) ∨ (r = -1 ∧ ⟨∀i : 0 ≤ i < N : A.i ≠ N⟩)} → óptima

Intuitivamente (por Carlos)

do n ≠ N

if A.n ≠ X → r, n := -1, n+1 (o podemos asignarlo al principio.)

□ A.n = X → r, n := n, N

siempre que yo entre acá quiere continuar, entonces podríamos poner esta guarda en B

if ; od. (Cuando aparece el elemento forzo que termine el ciclo.)

esto poco lo que denuncia programas porque es común mucho.

optimizado.

r, n := -1, 0;

do n ≠ N ∧ A.n ≠ N → n := n+1

od;

{I ∧ (n = N ∨ A.n = X)}

I = ⟨∀i : 0 ≤ i < n : A.i ≠ N⟩ ∧ n ≤ N

↓

lo interesante es que si n = N se cumple la Q óptima.

if n = N → skip

□ n ≠ N → r := n

fi;

casi el programa que le gusta

situacion de estado intermedio por creatividad

Cambio la actualizacion y el if

n := 0

if n = N → r := -1

Empezando al revés!

```
r:=N-1;
do r≥0 ∧ A.r≠X →
    r:=r-1
od
```

Va a dar la última vez que aparece

$I \equiv \langle \forall i : r < i < N : A.i \neq X \rangle \wedge r \geq -1$

Práctico 5

1) Algoritmo de la división

```
Const x, y : Int;
Var q, r : Int;
{P: x ≥ 0 ∧ y > 0}
S
{Q: x = q*y + r ∧ 0 ≤ r ∧ r < y}
```

$0 \leq r < y$

Ver: Usamos la técnica de usar partes de una conjunción

Sea $I = \{x = q*y + r \wedge 0 \leq r\}$;

$B: r \geq y$

So q, r := t, t

```
do r ≥ y → q, r := t, f   S1
od
```

S_0 Veamos la wp de S_0

```
q, r := 0, x;
do r ≥ y →
    q, r := t, f
od
```

$wp.q,r := t,f \ \{x = q*y + r \wedge 0 \leq r\}$

$= \{\text{def wp}\}$

$x = t*y + f \wedge 0 \leq f$

$= \{\text{aritmética}\}$

$x = 0*y + x \wedge 0 \leq f$

$= \{x \geq 0 \wedge f = 0\}$

Veamos S_1 $\quad q, r := E, F.\ \{I \equiv x = q*y + r \wedge 0 \leq r\}$ $\quad r = x - q*y$

$= \{\text{def wp}\}$

$x = E*y + F \wedge 0 \leq F$

$= \{\text{hipótesis: } r \geq y \text{ es decir } r - y \geq 0 \text{ es decir } r - y > -1\}$

$x = E*y + r - y \wedge -1 < r - y$

$= \{\text{hipótesis inductiva}\}$

$q*y - x = E*y - y \wedge -1 < r - y$

$q*y - y = E*y - y$
$q*y - y = x - y$
$y(q+1) = x - y$
elijo $q + 1 = E$

True

2) Búsqueda lineal dada $f : Nat \to Bool$ y suponiendo $(\exists n : 0 \le n : f.n)$ especificar y derivar un programa que calcule el mínimo x tq $f.x$.

Var $x : Int$

$\{\exists n : 0 \le n : f.n\}$

$\{x : \langle Min_i : n < i : f.i \rangle \wedge \langle \forall k : 0 \le k < x : \neg f.k \rangle\}$ $x \ge 0$

Sea:

do $B \to S$

od

el invariante = $\langle$ grada : $B \cdot \neg f.i$

menor original

$I = \langle \forall k : 0 \le k < x : \neg f.k \rangle$

Intuitivamente

$x := 0$

do $\neg f.x \to x := x + 1$

od

Veamos el invariante

1) $P \to I$

2) $I \wedge \neg B \to Q$

3) $\{I \wedge B\}\ S\ \{I\}$

2) $I \wedge \neg B \to Q$

$\langle \forall k : 0 \le k < x : \neg f.k \rangle \wedge f.x \to Q$

1) $x := 0 \to \langle \forall k : 0 \le k < x : \neg f.k \rangle$

$\langle \forall k : 0 \le k < 0 : \neg f.k \rangle$

Para búsqueda lineal $\langle Min_i : R.i : i \rangle \equiv R.x \wedge \langle \forall i : i < x : \neg R.i \rangle$

Cota? Supongo lo mismo que ∃ lat mín y le llamo X

$x := x$

para 4.

12, 13 y 14 son de fortalecimiento
15, 16, 17, 18 Son problemas de bordes
5, 6, 7 Reemplazo cte x variable.

Clase de consulta.

$Q = r \equiv \langle N p,q : 0 \le p < q < N : A.p \lor A.q \ge 0 \rangle \leftarrow$

Con reemplazo de cte x variable introduzco una nueva variable n.

$I \equiv r = \langle N p,q : 0 \le p < q < n : A.p \lor A.q \ge 0 \rangle \wedge n \le N$ → nos interesa que n chica sea igual que N para la post.

$B \equiv n \neq N$

$(I \wedge \neg B \Rightarrow Q$ sale trivialmente)

Para la inicialización es la prop 1 de I

Ahora B_0 (Inicialización) y S (cuerpo del ciclo)

$\{P\}$

S_0; → $r, n := 0, 0$

do $n \neq N$

S

od

$\{Q\}$

Como lo único me importa es el signo del producto, puedo hacer una definición por casos. para definir los casos importa el SIGNO de los términos $\ge 0, \le 0$

Cuerpo del ciclo

Encontrar E tq

$\{I \wedge B\}\ r, n := E, n+1\ \{I\}$ — por precaución y es obvio.

Suponemos $I \wedge \neg B \Rightarrow wp(r, n := E, n+1).I$

$\equiv$ def wp

$E = \langle N p,q : 0 \le p < q < n+1 : A.p \lor A.q \ge 0 \rangle \wedge$

$n+1 \le N$ ← hipótesis invariante y guarda.

$=$ {Partición de rango. Precaución con las reglas, es notación de Σ

$E = \langle N p,q : 0 \le p < q < n : A.p \lor A.q \ge 0 \rangle +$ ⊛

$\langle N p,q : 0 \le p < q = n : A.p \lor A.q \ge 0 \rangle$

$=$ {hipótesis e invariante

$E = r + \langle N p,q : 0 \le p < q = n : A.p \lor A.q \ge 0 \rangle$

$=$ {Andado y rango unitario → problem.

$E = r + \langle N p : 0 \le p < n : A.p \lor A.n \ge 0 \rangle$ → NO SE PUEDE FORTALECER

* Caso $A.n > 0$

$=$ {regla signos. $A.p \lor A.n \ge 0 \equiv A.p \ge 0$

$E = r + \langle N p : 0 \le p < n : A.p \ge 0 \rangle$ fortalezco con S → Cuenta los elementos positivos.

* Caso $A.n = 0$

$=$ {algebra

$E = r + \langle N p : 0 \le p < n : A.p \lor (A.n \ge 0) \rangle$ ($A.n \to 0$, $0 \ge 0$ true) → por muchos rangos unitarios y porque por cada 0 tenemos n productos 0.

$E = r + \langle N p : 0 \le p < n : True \rangle$ elijo $E = r + n$

$r, n := E, n+1$

⊛ $0 \le p < q \wedge (q < n \lor q \neq r)$

otra forma $0 \le p < n \wedge q = n$

$\langle Np: 0 \le p < n : True \rangle \Longrightarrow n.$

$\langle \Sigma p: 0 \le p < n : 1 * 1 \rangle$

$\langle \Sigma p: 0 \le p < 3 : 1 \rangle \rightarrow \{0,1,2\} = 1+1+1 \rangle$

$\equiv$ Caso $n < 0$

$\equiv$ { Regla signos $Ap * A.p \ge 0 \equiv Ap \le 0$

~~$E = \langle A \langle Np: 0 \le p < 0 : A p \rangle A \cdot 0 \rangle \equiv \langle A p \le 0 \rangle$~~

$E = r + \underbrace{\langle Np: 0 \le p < n : A p \le 0 \rangle}_{\text{portalizo con } t}$

Nuevo Invariante

$I' \equiv I \wedge s = \langle N.p : 0 \le p < n : A.p \ge 0 \rangle \wedge t = \langle Np: 0 \le p < n : A.p \le 0$

Nuevo Cuerpo del ciclo.

tema de hoare nueva.

$\{I' \wedge B\}\ r,s,t,n := E,F,G,n+1\ \{I'\}$

Supongamos $I' \wedge B$ vemos la wp.

$wp.\ r,s,t,n := E,F,G,n+1\ \{I'\}$

$\equiv$ { definicion de wp y mismos pasos

$E = r + \langle Np: 0 \le p < n : A.p * An \ge 0 \rangle \wedge$

$F = \langle Np: 0 \le p < n+1 : A.p \ge 0 \rangle \wedge$

$G = \langle Np: 0 \le p < n+1 : A.p \le 0 \rangle$

$\equiv$ { particion de rango e hipótesis

$E = \dots \wedge F = s + \langle Np: p = n : A.p \ge 0 \rangle \wedge G = t + \langle N.p : p = n : A.p \ge 0 \rangle$

↓ rango unitario ←

$T_i : i = n : t_i \equiv$
$T.n \rightarrow 1$
$\square \rightarrow T.n \rightarrow 0$
)

* Caso $A.n > 0$

regla de signos $\equiv$ { muchas cuentas (rango unitario de conteo) pasos anteriores que nos llevaron a los casos

$E = r + \langle Np: 0 \le p < n : A.p \ge 0 \rangle \wedge F = s + 1 \wedge G = t + 0$

$\equiv$ { hue sobre s del invariante

$E = r + s \wedge F = s + 1 \wedge G = t$

* Caso $A.n \not< 0$ (por mismos pasos (predicados de dos hoy)

$E = r + n \wedge F = s + 1 \wedge G = t + 1$

$\{P\} := \{R\}$ do $\{K\}$ do $\{Q\}$

* Caso $A.n < 0$

$\equiv$ { mismos pasos (usando los rangos vistos), en el caso simétrico a $A.n > 0$

$E = r + t \wedge F = S \wedge G = t + 1$

Elijo Cosas convenientemente

$\{N \geq 0\}$

$r, s, t, n := 0, 0, 0, 0$ ⟶ rangos vacíos de sumatorias

do $n \neq N$

if $A.n > 0$ ⟶ E F G

$r, s, t, n := r+s, s+1, t, n+1$

□ $A.n = 0$ ⟶

$r, s, t, n := r+n, s+1, t+1, n+1$

$A.n < 0$ ⟶

$r, s, t, n := r+t, s, t+1, n+1$

fi

od.

$t = N - n$

algoritmo mcd

```
do x ≠ y →
   if x < y →
      y := y - x
   □ x > y →
      x := x - y
   fi
od.
{X = mcd X.Y}
```

$I = \{x > 0 \wedge y > 0 \wedge mcd\, x.y = mcd\, X.Y\}$

y } veo lo que se achica.
x

lo que se puede hacer es asignar una cota para cada rama y ver como se juntan

Como armo una cota a partir de las dos cotas? $t = y + x$ o se achica una o se achica la otra.

$ot = max(x, y)$

3)

$\{N \geq 0\}$

S

$\{r : \langle \exists_i : 0 \leq i < N : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle$

Reemplazo de cte por variable

$I = \langle \exists_i : 0 \leq i < n : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle$

$B = N \neq n$

Cuerpo ciclo. $\{I \wedge B\}\ S\ \{I\}$

$\{\langle \exists_i : 0 \leq i < n : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle \wedge n \neq N\}\ r, n := E, n+1\ \{I\}$

asumo la precondicion y vamos la wp.

$E = \langle \exists_i : 0 \leq i < n+1 : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle$

$\equiv$ { particion de rango

$\equiv \langle \exists_i : 0 \leq i < n : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle \vee \langle \exists_i : i = n : A.n = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle$

$\equiv$ { hipotesis I, rango unitario.

$\equiv r \vee A.n = \langle \Sigma_k : 0 \leq k < n : A.k \rangle$

Bueno hay que fortalecer [illegible]

[illegible] parametro "s" tq

$I' = r = \langle \exists_i : 0 \leq i < n : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle \wedge s = \langle \Sigma_k : 0 \leq k < n : A.k \rangle$

$B = n \neq N$

$\{I' \wedge B\}\ n, r, s := n+1, R, S\ \{I'\}$

[illegible]

$R = \langle \exists_i : 0 \leq i < n+1 : A.i = \langle \Sigma_k : 0 \leq k < i : A.k \rangle \rangle \wedge S = \langle \Sigma_k : 0 \leq k < n+1 : A.k \rangle$

{ mismo paso

$R = r \vee A.n = \langle \Sigma_k : 0 \leq k < n : A.k \rangle \wedge S = \langle \Sigma_k : 0 \leq k < n+1 : A.k \rangle$

$\equiv$ { particion de rango [illegible]

$R = r \vee A.n = \langle \Sigma_k : 0 \leq k < n : A.k \rangle \wedge S = \langle \Sigma_k : 0 \leq k < n : A.k \rangle + A.n$

$\equiv$ { hipótesis I

$\equiv r \vee A.n = s \wedge S = s + A.n$

Conclusion

Const N : Int, A : array [0, N) of Int;

Var r : Bool, Var n : Int; Var s

$n, r, s := 0, false, 0;$

do $n \neq N \rightarrow$

$n, r, s := n+1, r \vee A.n = s, s + A.n$

od

Invariante con $P \to I'$

$r, s, n := R, S, \#N \longrightarrow R = \langle \exists i : 0 \le i < n : A.i = \langle \exists k : 0 \le k < \#s : s.k \rangle \rangle \wedge S = \langle \forall k : 0 \le k < n : A.k \rangle$

$= \{ n = 0$ y algún convencimiento $\}$

$r, s, n := false, 0, 0$

Dado un número, devuelve la posición en la que se encuentra en un arreglo. Si el número no se encuentra en el arreglo devuelve -1

$P \{ N \ge 0 \}$

$Q = \{ (\langle \exists i : 0 \le i < N : a.i = A \rangle \wedge r = i) \vee (\neg \langle \exists i : 0 \le i < N : a.i = A \rangle \wedge r = -1) \}$

$\langle \forall i : 0 \le i < N : a.i \neq A \rangle$

$Q = \langle \exists i : 0 \le i < N : A.i = A \rangle$

$I : 0 \le n \le N$

$B = n \neq N$

$T = N - n$

```
n := 0 ;
do n ≠ N →
   if (a.n = A) → r := n
   □ (a.n ≠ A) → skip
fi
n := n + 1
od ;
if (n = N) → r := -1
□ n ≠ N → skip
fi
```

El programa

S_1 [$n := 0$;

S_2 [do $n \neq N \wedge a.n \neq A \to$ $n := n + 1$ od ;

S_3 [if $(n = N) \to$ ...
□ $(n \neq N) \to$ ...
fi

entre S_2 y S_3

$I = \langle \forall i : 0 \le i < n : a.i \neq A \rangle \wedge 0 \le n < N$

$\neg B = (n = N \vee a.n = A)$

$\{ I \wedge \neg B \}$ ...

Demostracion formal

$\{N \geq 0\}$

S_0:
```
n := 0 ;
do n ≠ N ∧ a.n ≠ A →
    n := n + 1
od ;
```

S_1:
```
if n = N →
    r := -1
□ n ≠ N →
    r := n
fi
```

$\{P\}\ S_0 ; S_1\ \{Q\}$ Dar R tq

① $\{P\}\ S_0\ \{R\}$ ←

② $\{R\}\ S_1\ \{Q\}$

cuantificador

donde $R = I \wedge \neg B$

$I = \langle \forall i : 0 \leq i < n : A.i \neq A \rangle \wedge 0 \leq n \leq N$

$\neg B \equiv n = N \vee a.n = A$

① $\{P\}\ n := 0\ \{n = 0\}$ ③

$\{n = 0\}\ S_2\ \{R\}$ ④

③ Sale como wp.

④ Queda

a) $\{n = 0\} \Rightarrow I$ ✓

b) $I \wedge \neg B \Rightarrow R = I \wedge \neg B$ ✓ (trivialmente)

c) $\{I \wedge B\}\ n := n + 1\ \{I\}$

d) cota $t = N - n$

para c) suponemos $I \wedge B$ y vemos la wp

$\equiv$ { def wp

$\langle \forall i : 0 \leq i < n+1 : A.i = A \rangle \wedge 0 \leq n+1 \leq N$

= { partición de rango y rango unitario

$\langle \forall i : 0 \leq i < n : A.i \neq A \rangle \wedge a.n \neq A \wedge 0 \leq n+1 \leq N$

sale por guarda e invariante

= { hipótesis I

$true \wedge A.n \neq A$

= { guarda. $A.n \neq A$

true !

Sólo falta el if

25.5

② Sup $R = I \wedge \neg B$ y vemos la wp.

$wp.S_1.Q$

$\equiv$ { wp if

$(@(n=N) \vee N=n) \wedge n=N \Rightarrow$ ⓑ $wp.(n:=-1)\,Q \wedge$

$n \neq N \Rightarrow wp.(r:=n)\,Q$ ⓒ

a) True!

b) Sup $n=N$ y veamos la wp.

$wp.(r:=(-1)).Q$

$\equiv$ { def wp }

$(0 \leq -1 < N \wedge a.(-1) = A) \vee (\langle \forall i : 0 \leq i < N : a.i \neq A \rangle \wedge -1 = -1)$ (false, true)

$\equiv$ { lógica

$\langle \forall i : 0 \leq i < N : a.i \neq A \rangle$

$\equiv$ { hip 1 y $n=N$

c) $n \neq N \Rightarrow wp.(r:=n)\,[Q]$

$(0 \leq n < N \wedge a.n = A) \vee (\langle \forall i : 0 \leq i < n : a.i \neq A \rangle \wedge n = -1)$

$\equiv$ { hipótesis con $n \geq 0$

$0 \leq n < N \wedge a.n = A$

$\equiv$ { hipótesis $0 \leq n < N \wedge n \leq N$

$a.n = A$

$\equiv$ { negación de la guarda $n=N \vee a.n \neq A$, además $n \neq N$, luego ...

True. ∎ Demasiado complicado

$0 \leq n < N \wedge n \neq N \ldots \; 0 \leq n < N$